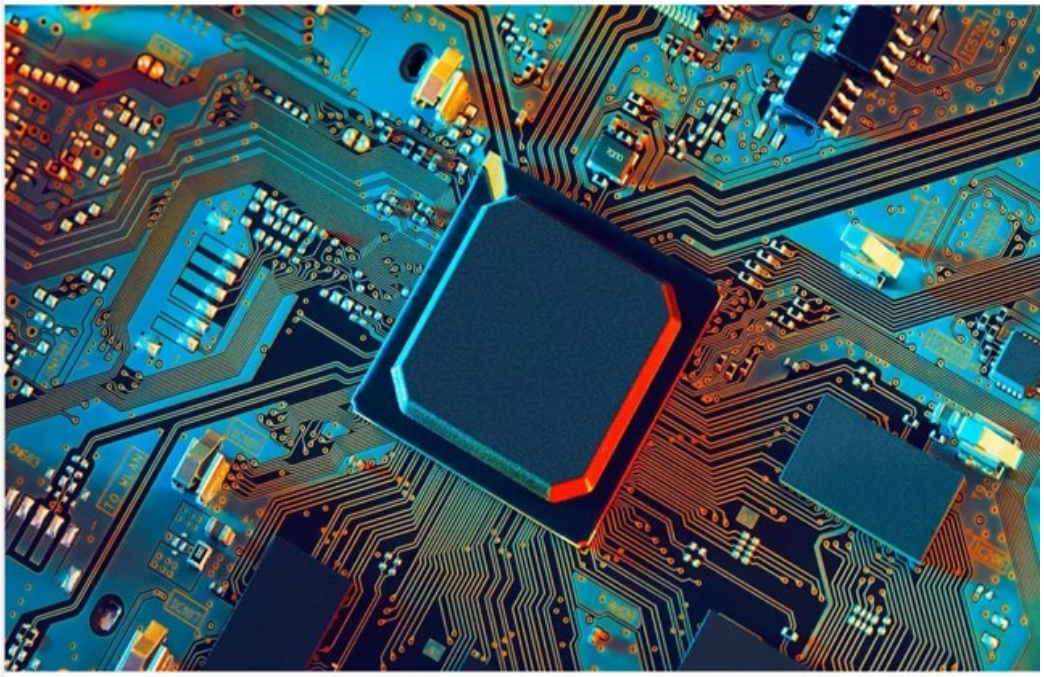
КОМП'ЮТЕРНА СХЕМОТЕХНІКА
частина 2

Коваленко О.Є., Волошин С.М., Гусєв Б.С.
Нікітенко Є.В., Матієвський В.В.



КОМП'ЮТЕРНА СХЕМОТЕХНІКА
частина 2
«ТИПОВІ ВУЗЛИ КОМП'ЮТЕРНИХ СИСТЕМ»

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Київ
2023

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Кафедра комп'ютерних систем, мереж та кібербезпеки

Коваленко О.Є., Волошин С.М., Гусєв Б.С., Нікітенко Є.В., Матієвський В.В.

КОМП'ЮТЕРНА СХЕМОТЕХНІКА

частина 2

«ТИПОВІ ВУЗЛИ КОМП'ЮТЕРНИХ СИСТЕМ»

(навчальний посібник для самостійної роботи студентів з курсів «Комп'ютерна схемотехніка», «Компонентна база та схемотехніка в системах захисту інформації »)

КОМПРІНТ
КИЇВ - 2023

УДК 004.3(072)
ББК 39.97
К 63

Рекомендовано до друку Вченою радою Національного університету біоресурсів і природокористування України

Рецензенти:

Криворучко О.В. – доктор технічних наук, професор, завідувач кафедри інженерії програмного забезпечення та кібербезпеки Київського національного торговельно-економічного університету;

Цюцюра С.В. – доктор технічних наук, професор, завідувач кафедри інформаційних технологій Київського національного університету будівництва і архітектури;

Болбот І.М. – доктор технічних наук, доцент, професор кафедри автоматики та робототехнічних систем ім. академіка І.І. Мартиненка Національного університету біоресурсів і природокористування України.

Коваленко О.Є., Волошин С.М., Гусєв Б.С., Нікітенко Є.В., Матієвський В.В.
К 63 Комп'ютерна схемотехніка. Частина 2 [навчальний посібник] / О.Є.Коваленко, С.М.Волошин, Б.С.Гусєв, Є.В., Нікітенко, В.В.Матієвський // - К.: НУБіП України, 2023.- 331с.

Навчальний посібник «Комп'ютерна схемотехніка» (частина 2) призначений для використання здобувачами вищої освіти за спеціальностями 125 «Кібербезпека та захист інформації» і 123 «Комп'ютерна інженерія» освітнього ступеня «Бакалавр» під час самостійної теоретичної підготовки в галузі розробки та проєктування типових вузлів на основі тригерних схем в операційних автоматах комп'ютерних систем.

В навчальному посібнику розглянуті порядок синтезу, структурна організація типових цифрових послідовністних вузлів, які побудовані на основі тригерних схем та використовуються в операційних автоматах комп'ютерних систем загального і спеціалізованого призначення, мікроконтролерних системах. Для цих вузлів аналізуються характеристики, властивості функціонування та визначаються динамічні параметри.

В результаті опанування навчальним матеріалом, представленим в посібнику, здобувачі вищої освіти мають можливість проєктувати послідовністні пристрої, виконуючи процедуру їх синтезу, в тому числі розробку таблиці переходів, визначення функцій збудження базових тригерів, та проводити аналіз функціонування і визначення параметрів цих пристроїв.

Кожний підрозділ посібника містить контрольні завдання та запитаннями, що дозволяє здобувачам вищої освіти отримувати необхідні практичні вміння і навички.

ЗМІСТ

ПЕРЕДМОВА

Швидкий розвиток інформаційних технологій, зокрема в галузі проєктування апаратних засобів комп'ютерних систем, вимагає від здобувачів вищої освіти за спеціальностями «Комп'ютерна інженерія» і «Кібербезпека та захист інформації» отримання сучасної якісної підготовки для забезпечення проведення розробки, застосування та супроводження комп'ютерних систем загального та спеціалізованого призначення.

Це може бути досягнуто за рахунок впровадження відповідної організації проведення навчального процесу, метою якого є забезпечення підготовки висококваліфікованих фахівців, які мають високий рівень компетенцій, сучасних знань і вмінь в галузі інформаційних технологій, що відповідають зазначеним вище спеціальностям, а зокрема в галузі проєктування, застосування та супроводження апаратних засобів комп'ютерних систем.

Вирішення цих задач потребує забезпечення сучасної теоретичної і практичної підготовки здобувачів вищої освіти з метою виконання синтезу цифрових пристроїв та проведення аналізу їх функціонування для визначення параметрів цих пристроїв, отриманих в процесі синтезу.

Навчальний посібник призначений для підготовки здобувачів вищої освіти за спеціальностями 123 «Комп'ютерна інженерія» і 125 «Кібербезпека та захист інформації» під час здобування освітнього ступеня «Бакалавр» при опануванні дисциплінами «Комп'ютерна схемотехніка», «Компонентна база та схемотехніка в системах захисту інформації».

В посібнику розглядаються основні методи розробки та варіанти апаратної реалізації цифрових послідовністних пристроїв, що побудовані на основі тригерних схем та є базою для використання в будь-якій комп'ютерній системі в складі операційних автоматів, а також електронного обрамлення мікроконтролерних систем.

В навчальному посібнику, який є продовженням першої частини аналогічного посібника, з метою забезпечення якісної підготовки фахівців, які будуть працювати в галузі розробки і застосування цифрових пристроїв, детально розглянуті наступні розділи:

- синтез тригерних схем на базі стандартних тригерних схем;
- аналіз роботи синтезованих тригерних схем в статичному і динамічному режимах роботи;
- регістри з керованою синхронізацією на базі тригерів;
- регістри з некерованою синхронізацією на базі тригерів;
- регістри на базі інтегральних схем регістрів;
- цифрові лічильники;
- способи організації переносів в двійкових лічильниках;
- синтез лічильників з довільним модулем ліку;
- лічильники на базі лічильної схеми;
- багатофункціональні пристрої, до складу яких входять як регістри, так і лічильники.

В першому розділі посібника розглядається процедура синтезу тригерних схем, які будуються на основі стандартних *D*-, *RCS*-, *JK*-, *TC*- або *T*- тригерів, що дозволяє виконувати взаємні перетворення стандартних тригерів між собою, а також розробляти власні тригери, що відрізняються від стандартних.

Другий розділ присвячений синтезу регістрів як на основі окремих тригерів, так і на базі інтегральних схем регістрів. Зокрема розглядається синтез регістрів з керованою і некерованою синхронізацією, які виконують не тільки завантаження інформації із зовнішніх джерел, але й мікрооперації зсуву даних. Для цього детально роз’яснюються види зсувів, що використовуються для обробки інформації. Наприкінці розділу розглядається організація двотактових регістрів. Для всіх типів регістрів, що розглянуті в цьому розділі, приведені переваги і недоліки різних способів структурної і логічної реалізації цих пристроїв.

Третій розділ присвячений розгляду проведення процедур синтезу і аналізу функціонування цифрових лічильників. Спочатку аналізуються різні способи логічного побудування двійкових лічильників з точки зору реалізації різних способів організації переносів між розрядами лічильника: лічильники з паралельним і послідовним трактами розповсюдження переносу, лічильники з безпосереднім переносом. При цьому кожний з цих типів переносів розглядається як для асинхронних, так і для синхронних лічильників.

Далі розглядаються різні способи синтезу лічильників з довільним модулем ліку: класичний спосіб синтезу, побудова підсумовувальних лічильників за модулем 2^k+1, послідовне з'єднання лічильників, лічильники з асинхронним і синхронним встановленням початкового стану, а також лічильники на базі лічильної схеми.

Характеристики і властивості цифрових пристроїв розглядаються на структурному, структурно-функціональному та логічному рівнях їх побудови.

У всіх розділах наведені приклади синтезу вказаних вище цифрових вузлів, а в якості перевірки коректності проведеного синтезу та наведених тверджень виконується моделювання їх функціонування з використанням системи схемотехнічного проєктування MicroCap.

Навчальний матеріал, що наведено в посібнику, дозволяє здобувачам вищої освіти, що навчаються за спеціальностями 123 і 125, вивчати теоретичні засади з розробки, аналізу функціонування цифрових вузлів та отримувати практичні вміння під час виконання наступних етапів розробки цифрових вузлів:

- синтез послідовністних цифрових пристроїв з використанням таблиць переходів та отримання логічних виразів, що описують функціонування синтезованих вузлів;

- мінімізація отриманих логічних виразів та їх перетворення до заданого логічного базису;

- визначення динамічних параметрів типових вузлів, аналізуючи їх логічну, структурно-функціональну та структурну організацію;

- аналіз функціонування синтезованих пристроїв за допомогою систем автоматизованого проєктування.

Всі підрозділи посібника містять контрольні запитання для забезпечення більш кращого розуміння навчального матеріалу, а виконання контрольних завдань дозволяють здобувачам вищої освіти детальніше засвоїти методи проєктування цифрових пристроїв.

Навчальний матеріал, наведений в посібнику, є базовими і дозволяє набути початкові вміння та навички з розробки типових цифрових вузлів на базі тригерних схем.

Крім того, в посібнику також приведений перелік літературних і довідникових джерел, які можуть бути використані під час самостійного опрацювання навчального матеріалу, а також можуть бути застосовані для вдосконалення отриманих знань для виконання складних технічних завдань.

Навчальний матеріал, поданий в посібнику, базується на знаннях, вміннях і навичках, отриманих здобувачами вищої освіти при вивченні курсу «Комп’ютерна логіка», а також передбачає ознайомлення з першою частиною навчального посібника.

Автори вдячні шановним колегам та рецензентам за слушні зауваження та підтримку, що сприяло поліпшенню змісту посібника, послідовності і методики викладання навчального матеріалу.

1 СИНТЕЗ ТРИГЕРНИХ СХЕМ НА ОСНОВІ ТИПОВИХ ТРИГЕРІВ

Основи структурної організації, схемної реалізації, синтезу та властивості тригерних схем, а також визначення значень їх динамічних параметрів детально проаналізовано в [1], де на структурно-функціональному і логічному рівнях представлення були розглянуті класифікація, синтез тригерних схем (ТС), аналіз функціонування ТС в статичному та динамічному режимах роботи (в тому числі й за допомогою логічного моделювання), визначення функцій збудження тригерів та параметрів, які визначають динамічні властивості ТС, реакція ТС на виникнення завад.

Під час проєктування цифрових пристроїв для забезпечення збереження інформації виникає задача розробки вузлів цих пристроїв на базі тригерів. При цьому необхідно відзначити наступне:

1. При проєктуванні апаратних засобів комп'ютерних систем (наприклад, мікроконтролерних систем) може існувати необхідність розробки апаратної частини такої системи, яка, фактично, є її електронним обрамленням. До складу такої частини можуть входити як окремі тригери, так і тригери у складі регістрів, в які записується певна інформація з вихідних портів мікроконтролера або, навпаки, з яких інформація потрапляє в мікроконтролер.

У зв'язку з тим, що в інтегральному виконанні випускаються тільки *JK*- або *D*-тригери (маються на увазі синхронні тригери), то в разі потреби використовування *RCS*-, *TC*- або *T*-тригерів необхідно виконувати перетворення схем тригерів, що є в наявності, в тригери потрібного типу.

2. Така ж задача може виникати при використовуванні ***FPGA***-технологій (***FPGA – Field-Programmable Gate Array*:** ПЛІС – програмована логічна інтегральна схема) в разі необхідності створення необхідних схем

тригерів, використовуючи готові моделі тригерів з відповідної бібліотеки системи проєктування.

3. Під час проєктування може виникнути потреба використовувати ті чи інші нестандартні тригери, наприклад, *E*-тригер, який є різновидом *RS*- тригерів та виконує мікрооперацію збереження інформації при двох однакових вхідних сигналах (*R = S)*.

Виходячи з цього, розглянемо методику синтезу будь-якого тригера на базі типової тригерної схеми.

Типовий тригер, на основі якого проєктується заданий тригер, будемо називати базовим тригером (БТ).

Тригер, який необхідно розробити, будемо називати підсумковим тригером (ПТ).

Узагальнена структурна схема підсумкового тригера приведена на рис.1.1.

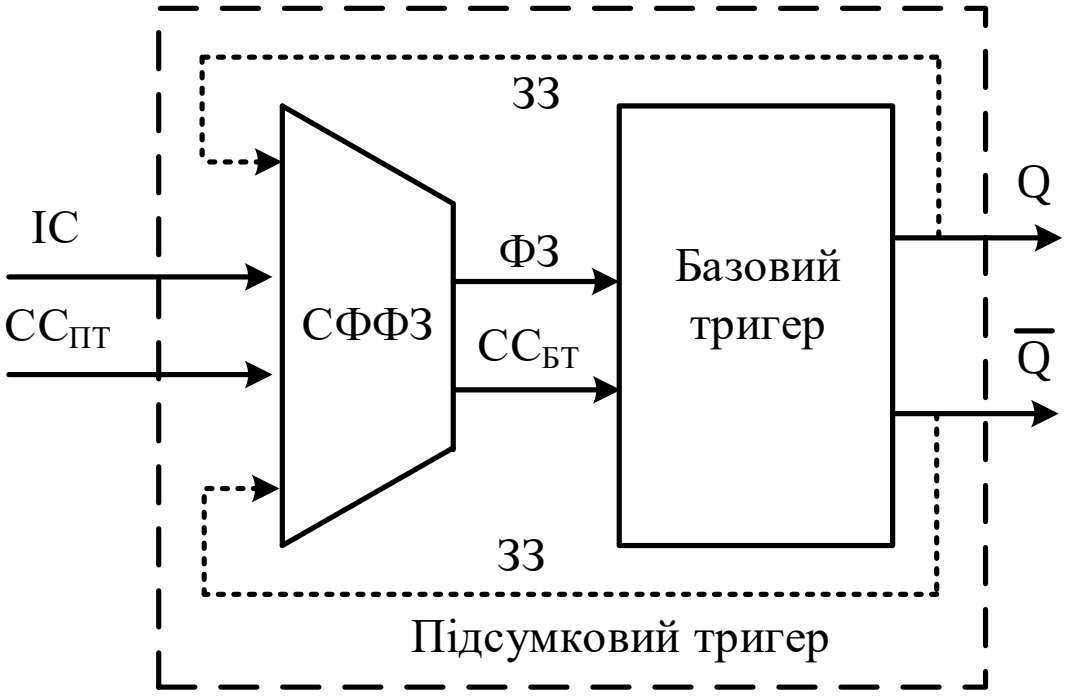


Рисунок 1.1 – Структурна схема підсумкового тригера

На рис.1.1 використовуються наступні скорочення:

- *ІС* – інформаційні сигнали підсумкового тригера;
- $СС_{ПТ}$ – сигнали синхронізації підсумкового тригера;

- $СС_{БТ}$ – сигнали синхронізації базового тригера;
- *ЗЗ* – лінії зворотного зв’язку;
- *СФФЗ* – схема формування функцій збудження;
- *ФЗ* – функції збудження базового тригера;
- Q, $\overline{Q}$ – вихідні сигнали підсумкового тригера.

Лінії зворотного зв'язку можуть бути відсутні, тому показані пунктиром.

Схема формування функцій збудження забезпечує перетворення вхідних і вихідних сигналів підсумкового тригера таким чином, щоб забезпечити переключення базового тригера у відповідний стан згідно з його таблицею переходів.

В якості базового тригера зазвичай використовується будь-який стандартний тригер. В зв’язку з цим для виконання процедури проєктування підсумкового тригера необхідно синтезувати схему формування функцій збудження, яка є комбінаційною схемою: $ФЗ = f(IC, CC, Q^t)$.

Методика синтезу підсумкового тригера полягає в наступному (звичайно вважається, що принцип функціонування базового тригера є відомим):

1. Виконання розробки таблиці переходів ПТ.
2. Визначення функцій збудження БТ.
3. Мінімізація функцій збудження.
4. Виконання перетворення функцій збудження в заданий базис логічних елементів.
5. Реалізація логічної схеми та перевірка коректності функціонування схеми підсумкового тригера.

Розглянемо приклади синтезу підсумкового тригера на основі заданого базового тригера.

Приклад 1.1. Виконати синтез *DE*-тригера на базі *RES*-тригера в будь- якому базисі логічних елементів.

Розв'язок

Виконаємо розробку таблиці переходів підсумкового тригера та в цій же таблиці визначимо функції збудження базового тригера, тобто поєднаємо перший і другий кроки методики синтезу (табл.1.1).

Таблиця 1.1 складається з двох частин. В стовпчиках 1-4 приведена таблиця переходів підсумкового тригера, тобто *DE*-тригера (крок 1 методики синтезу), а в стовпчиках 5-7 визначені функції збудження базового тригера, тобто *RES*-тригера (крок 2 методики синтезу).

В зв'язку з тим, що обидва тригери мають вхід *E*, то ці входи будемо розрізняти за допомогою відповідних нижніх індексів E_D (колонка 1) і E_{RES} (колонка 5).

Таблиця 1.1 – Таблиця переходів *DE*-тригера і функції збудження *RES*- тригера

Номери наборів	$\mathbf{E_D}$	**D**	$\mathbf{Q^t}$	$\mathbf{Q^{t+1}}$	$\mathbf{E_{RES}}$	**R**	**S**
	1	**2**	**3**	**4**	**5**	**6**	**7**
0	1	0	0	0	1	*	0
1	1	0	1	0	1	1	0
2	1	1	0	1	1	0	1
3	1	1	1	1	1	0	*
4	0	0	0	0	0	*	*
5	0	0	1	1	0	*	*
6	0	1	0	0	0	*	*
7	0	1	1	1	0	*	*

Метою виконання першого кроку методики є заповнення стовпчика 4 таблиці переходів

Нагадаємо принцип функціонування *DE*-тригера. Відповідно до [1] формування наступного стану *DE*-тригера визначається за логічним виразом $Q^{t+1} = \overline{E} \cdot Q^t \vee E \cdot D$, тобто за неактивного значення сигналу синхронізації ($E = 0$) тригер перебуває в стані збереження інформації ($Q^{t+1} = Q^t$), а за активного значення сигналу синхронізації ($E = 1$) значення сигналу зі входу

D з’являється на виході тригера ($Q^{t+1} = D$). Отже, в таблиці переходів для номерів наборів 4-7 ($E = 0$) значення сигналу на виході *Q* (Q^{t+1}) повторює попереднє значення на цьому виході (Q^t). Для номерів вхідних наборів 0-3 ($E = 1$) відповідне значення Q^{t+1} дорівнює значенню сигналу на вході *D*.

Наступним кроком (крок 2) є заповнення стовпчиків 5-7 таблиці 1.1. В зв’язку з тим, що вхід синхронізації визначає мікрооперацію, яку виконує тригер (збереження або завантаження інформації), то визначення функцій збудження починається із заповнення колонки 5. Існує кілька варіантів заповнення цієї колонки, але будемо використовувати такий, при якому налаштування підсумкового тригера на виконання відповідної мікрооперації (збереження або завантаження інформації) забезпечує виконання такої ж мікрооперації й базовим тригером. Таким чином, значення сигналів в колонці 5 буде збігатися з аналогічними значеннями в колонці 1.

Далі заповнюємо стовпці 6, 7. Для вхідних наборів 4-7 значення сигналу на вході синхронізації *RES*-тригера є неактивним, тобто стан цього тригера не залежить від значень сигналів на інформаційних входах *S* і *R*, тому в стовпцях 6, 7 на цих наборах записуємо символи «*», які означають, що сигнали на цих входах можуть бути будь-якими.

Значення сигналів на входах *S* і *R* для вхідних наборів 0-3 визначаються таким же чином, як і для асинхронного *RS*-тригера [1].

Наприклад, розглянемо вхідний набір 3, відповідно до якого *DE*- тригер перебуваючи в одиничному стані в цьому ж стані й залишається, тобто відбувається виконання мікрооперації збереження одиниці. Це може бути досягнуто двома шляхами: на входи базового тригера задати комбінацію встановлення в одиничний стан ($S = 1$; $R = 0$) або підключити неактивні сигнали на інформаційні входи ($S = 0$; $R = 0$). В результаті можна отримати, що для забезпечення збереження одиниці на вхід *R* обов’язково необхідно підключати нуль, а на вхід *S* – будь-який сигнал, що відображено в таблиці 1.1 символом «*».

Далі визначаємо логічні вирази функцій збудження базового тригера та в разі потреби виконуємо мінімізацію цих виразів.

В зв'язку з тим, що колонки 1, 5 таблиці 1.1 однакові, то відразу можна отримати: $E_{RES} = E_D$.

Для визначення логічних виразів функцій збудження інформаційних сигналів будемо використовувати мінімізацію цих виразів за допомогою карт Карно (рис.1.2).

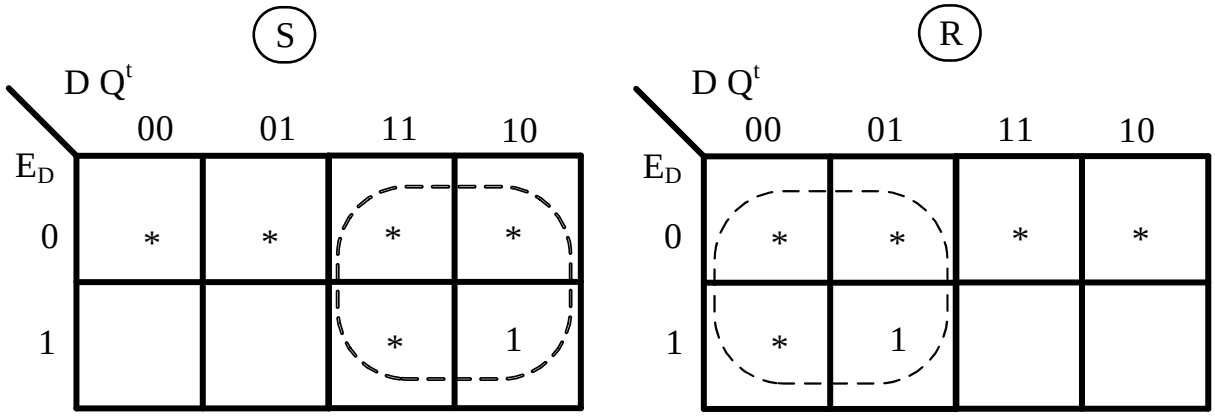


Рисунок 1.2 – Карти Карно для визначення ФЗ *RES*-тригера

В результаті мінімізації отримаємо: $S = D$; $R = \overline{D}$.

З отриманих виразів витікає, що для реалізації схеми підсумкового тригера достатньо одного додаткового інвертора, тобто перетворення до будь-якого базису логічних елементів не потрібно.

Функціональна схема підсумкового тригера приведена на рис.1.3.

Схема для моделювання підсумкового тригера в середовищі системи схемотехнічного проєктування ***MicroCap*** приведена на рис.1.4, а результати моделювання в статичному режимі – на рис.1.5.

Аналізуючи результати моделювання можна побачити, що при нульовому стані сигналу на вході *E* пристрій не переключається, виконуючи мікрооперацію збереження інформації, а при *E = 1* вихідний сигнал повторює значення сигналу на вході *D*, в результаті чого можна зробити висновок, що синтезований пристрій дійсно є *DE*-тригером.

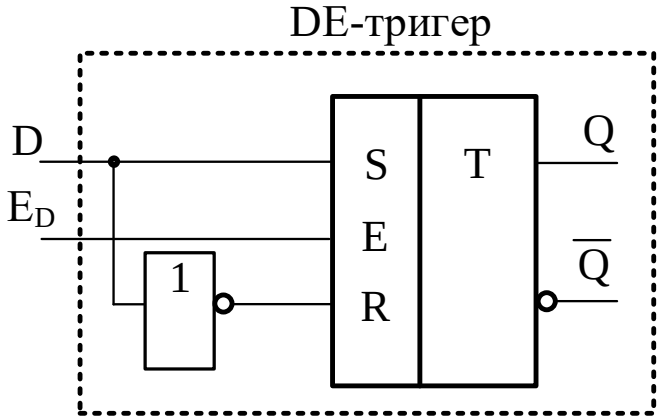


Рисунок 1.3 – Функціональна схема *DE*-тригера на базі *RES*-тригера

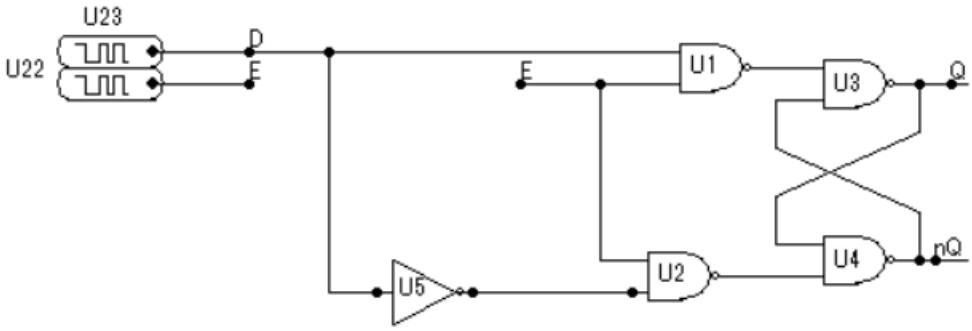


Рисунок 1.4 – Схема для моделювання *DE*-тригера на базі *RES*- тригера

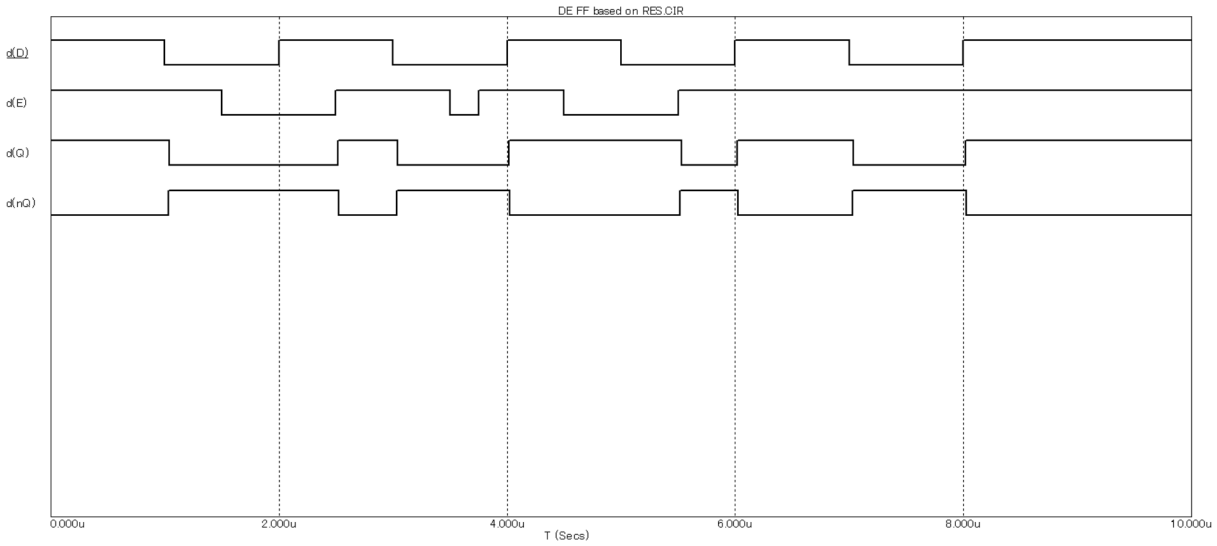


Рисунок 1.5 – Результати моделювання *DE*-тригера на базі *RES*- тригера

Далі розглянемо приклад синтезу нестандартного тригера.

Приклад 1.2. На базі *JK*-тригера виконати синтез X_1CX_2-тригера з використанням базису Шефера. Умовне графічне позначення (УГП) X_1CX_2- тригера приведено на рис.1.6, а скорочена таблиця переходів цього тригера – в табл.1.2. Таблиця 1.2 відображує функціонування тригера при активному значенні сигналу на вході синхронізації *C*.

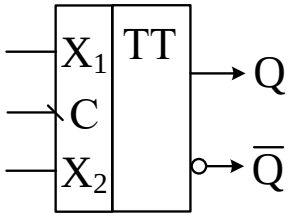


Рисунок 1.6 – УГП X_1CX_2-тригера

Таблиця 1.2 – Скорочена таблиця переходів ПТ

X_1	X_2	Q^{t+1}
0	***0***	***1***
0	***1***	$\overline{Q^t}$
1	***0***	***0***
1	***1***	Q^t

Розв’язок

Відповідно до УГП підсумкового тригера необхідно синтезувати двоступеневий X_1CX_2-тригер зі спрацьовуванням за заднім фронтом сигналу синхронізації.

Скорочена таблиця переходів підсумкового тригера відображує функціонування цього тригера в момент появи активного (заднього) фронту на вході *C*.

Відповідно до методики синтезу виконаємо розробку таблиці переходів підсумкового X_1CX_2-тригера та в цій же таблиці, як і в попередньому прикладі, визначимо функції збудження базового *JK*-тригера, поєднавши, таким чином, перші два кроки методики синтезу (табл.1.3).

Для того, щоб розрізняти сигнали на входах синхронізації заданих тригерів будемо використовувати позначення C_{x1x2} і C_{JK} відповідно для підсумкового і базового тригерів (колонки 1 і 6).

Таблиця 1.3 – Таблиця переходів X_1CX_2-тригера і функції збудження JK- тригера

Номери наборів	C_{x1x2}	X_1	X_2	Q^t	Q^{t+1}	C_{JK}	J	K
	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	*	*
1	0	0	0	1	1	0	*	*
2	0	0	1	0	0	0	*	*
3	0	0	1	1	1	0	*	*
4	0	1	0	0	0	0	*	*
5	0	1	0	1	1	0	*	*
6	0	1	1	0	0	0	*	*
7	0	1	1	1	1	0	*	*
8	1	0	0	0	1	1	1	*
9	1	0	0	1	1	1	*	0
10	1	0	1	0	1	1	1	*
11	1	0	1	1	0	1	*	1
12	1	1	0	0	0	1	0	*
13	1	1	0	1	0	1	*	1
14	1	1	1	0	0	1	0	*
15	1	1	1	1	1	1	*	0

Нагадаємо, що позначення логічної одиниці на входах C в таблиці переходів двоступеневих тригерів означає наявність активного фронту (для даного прикладу – заднього) на цих входах, а позначення логічного нуля – наявність будь-якого неактивного сигналу.

Таким чином, в першій половині таблиці (рядки 0-7) на вході C підсумкового тригера задане неактивне значення сигналу, тобто тригер буде виконувати мікрооперацію збереження інформації незалежно від значення сигналів на інформаційних входах ($Q^{t+1} = Q^t$), що відображено в стовпці 5.

В другій половині таблиці синхросигнал приймає активне значення, тобто підсумковий тригер спрацьовує відповідно скороченій таблиці переходів (табл.1.2), тобто за $X_1 = X_2 = 0$ тригер переключається в одиничний стан (рядки 8,9: $Q^{t+1} = 1$). Якщо $X_1 = 0$; $X_2 = 1$, то підсумковий тригер переключається в протилежний стан (рядки 10,11: $Q^{t+1} = \overline{Q^t}$). При $X_1 = 1$; $X_2 = 0$ тригер переключається нульовий стан (рядки 12,13: $Q^{t+1} = 0$), і, наприкінці, при $X_1 = X_2 = 1$ тригер виконує збереження інформації по інформаційним входам (рядки 14,15: $Q^{t+1} = Q^t$).

Далі виконується другий крок методики синтезу, на якому визначаються функції збудження *JK*-тригера, для чого необхідно заповнити стовпці 6-8 таблиці 1.3.

За аналогією з прикладом 1.1 визначення функцій збудження починається з заповнення стовпця 6, внаслідок того, що сигнал на вході C_{JK} базового тригера визначає режим функціонування цього тригера. Клітини стовпця 6 можуть бути заповнені кількома способами, наприклад, в клітини, які відповідають збереженню інформації, як по керуючому входу C_{JK}, так і за допомогою інформаційних входів *J* і *K*, заносяться неактивні значення сигналу C_{JK}.

Але для того, щоб функції збудження інформаційних входів базового тригера не залежали від значення сигналу синхронізації (наприклад, якщо базовий тригер є двоступеневим тригером з заборонним зв’язком [1], то цей тригер буде спрацьовувати не тільки за активним фронтом сигналу *C*, але й за фронтом інформаційних сигналів), доцільно в клітини для сигналу C_{JK} записувати такі значення сигналу, які забезпечують виконання мікрооперації збереження інформації тільки по керуючому сигналу без участі інформаційних входів. В результаті цього значення сигналів в колонці C_{JK} будуть збігатися з аналогічними значеннями в колонці C_{x1x2}.

Для визначення функцій збудження інформаційних входів заповнимо стовпці 7,8 таблиці 1.3.

В першій половині таблиці 1.2 для вхідних наборів 0-7 значення сигналу на вході синхронізації *JK*-тригера є неактивним. Це означає, що наступний стан базового тригера не залежить від значень сигналів на інформаційних входах *J* і *K*, тобто в стовпцях 7, 8 на цих наборах необхідно вказати символи «*», які зазначають, що сигнали на цих входах можуть бути будь-якими.

Функції збудження для входів *J* і *K* на вхідних наборах 8-15 визначаються таким же чином, як і для будь-якого *JK*-тригера, що було детально розглянуто в [1].

Наприклад, розглянемо вхідний набір 8, на якому підсумковий тригер переключається з нульового стану ($Q^t = 0$) в одиничний стан ($Q^{t+1} = 1$). Забезпечення цього переключення може бути досягнуто двома шляхами: на вхід *J* базового тригера задати одиничний сигнал, а на вхід *K* – нульовий ($J = 1$; $K = 0$), що відповідає вхідній комбінації ***Set*** або підключити на обидва входи одиниці ($J = 1$; $K = 1$), що відповідає переключенню тригера в протилежний стан. Звідси випливає, що для забезпечення переключення *JK*- тригера з нульового стану в одиничний на вхід *J* обов'язково необхідно підключати одиницю, а на вхід *K* – будь-який сигнал, що відзначено в таблиці 1.3 символом «*». Аналогічно визначаються інші функції збудження *JK*-тригера.

Далі виконаємо третій крок методики синтезу та визначимо логічні вирази, які описують функції збудження базового тригера.

Звертаючи увагу на те, що значення сигналів C_{JK} і C_{x1x2}, як вже було зазначено вище (стовпці 6 і 1 таблиці 1.3), є однаковими, то отримаємо:

$$C_{JK} = C_{x1x2}. \qquad (1.1)$$

Для визначення мінімальних логічних виразів функцій збудження інформаційних сигналів *JK*-тригера виконаємо мінімізацію цих виразів за допомогою карт Карно (рис.1.7).

На рис.1.7, а приведена карта Карно для визначення функції збудження входу *J*, а на рис.1.7, а – входу *K* .

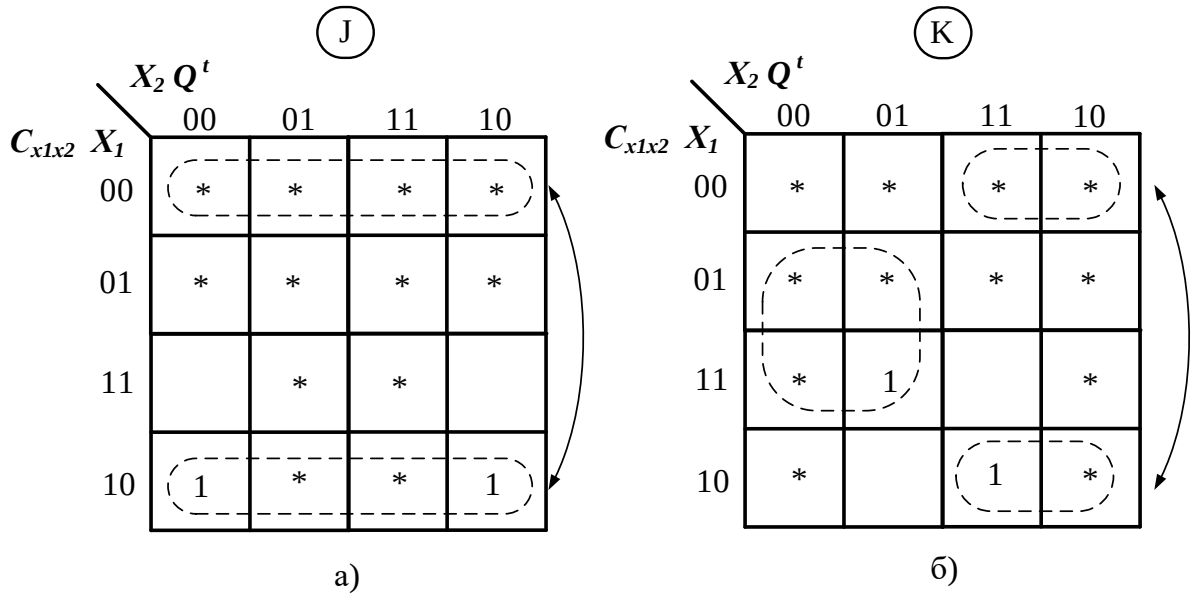


Рисунок 1.7 – Карти Карно для визначення ФЗ *JK*-тригера

В результаті мінімізації отримаємо наступні логічні вирази:

$$J = \overline{X_1}; \qquad K = \overline{X_1} \cdot X_2 \vee \overline{X_2} \cdot X_1 = X_1 \oplus X_2.$$

Відповідно до четвертого кроку методики виконаємо перетворення функцій збудження в заданий базис (згідно з завданням – в базис Шефера). В результаті, використовуючи закон де Моргана, отримуємо:

$$K = X_1 \oplus X_2 = \overline{X_1} \cdot X_2 \vee \overline{X_2} \cdot X_1 = \overline{\overline{\overline{X_1} \cdot X_2} \cdot \overline{\overline{X_2} \cdot X_1}}. \qquad (1.2)$$

Далі виконується останній крок методики синтезу, відповідно до якого необхідно здійснити реалізацію логічної схеми підсумкового тригера та перевірити коректність її функціонування, наприклад, за допомогою моделювання.

Логічна схема підсумкового тригера, побудована відповідно до виразів (1.1), (1.2) приведена на рис.1.8.

Схема для моделювання представлена на рис.1.9. В якості базового тригера будемо використовувати з бібліотеки типових вузлів системи схемотехнічного проєктування інтегральну схему (ІС) ***SN74107***, до складу якої входять два *JK*-тригера зі спрацьовуванням за заднім фронтом сигналу синхронізації і асинхронним входом скиду з інверсним керуванням ***nR***. Вхід

nR використовується для встановлення базового тригера в початковий стан. Умовне графічне позначення такого тригера можна побачити на рис.1.8, 1.9.

Результати моделювання підсумкового X_1CX_2-тригера приведені на рис.1.10.

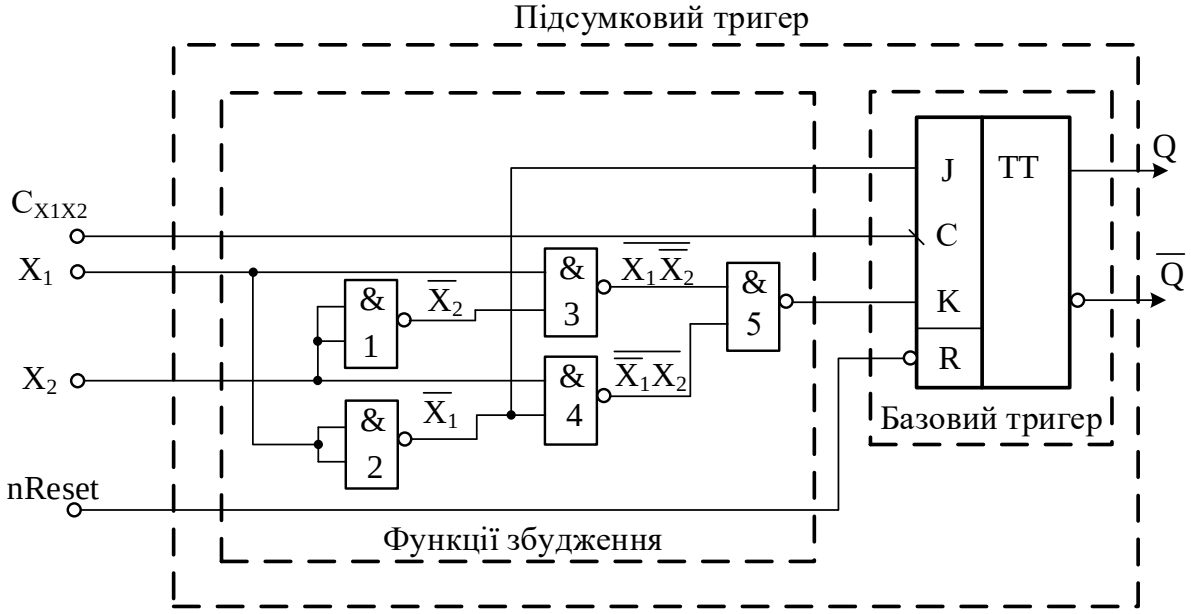


Рисунок 1.8 – Логічна схема підсумкового тригера в базисі Шефера

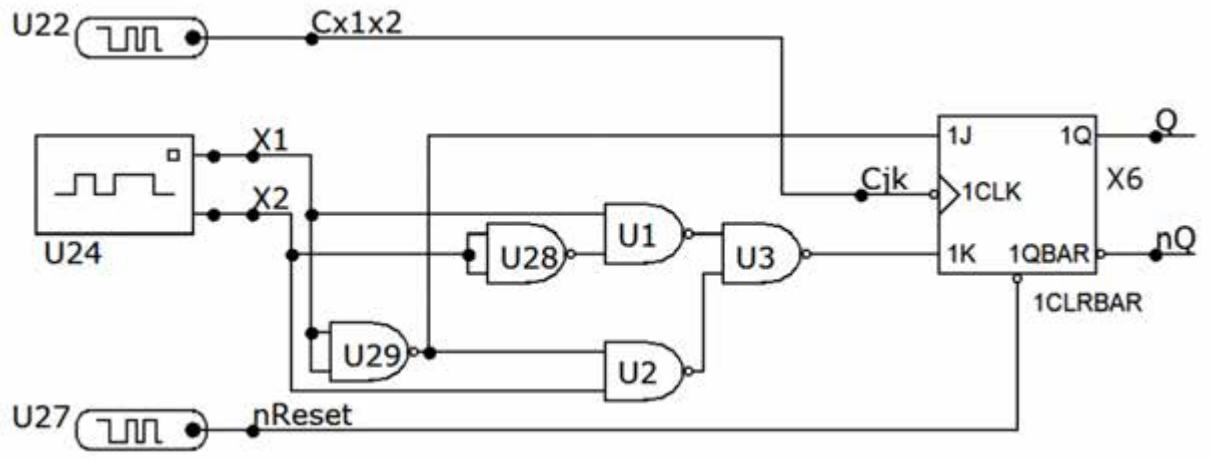


Рисунок 1.9 – Схема для моделювання підсумкового тригера

Часова діаграма на рис.1.10 розділена на такти з 0 до 9. Кожний такт відповідає періоду сигналу синхронізації та обмежений заднім фронтом цього сигналу, під час якого відбувається можливе переключення тригера. Для спрощення аналізу функціонування підсумкового тригера значення інформаційних сигналів X_1 і X_2 показані за допомогою функції ***bin()*** системи

моделювання. На часовій діаграмі в момент заднього фронту сигналу синхронізації використовуються скорочені позначення станів, в які переключається тригер, що досліджується:

- П1(П0) – переключення тригера в одиничний(нульовий) стан;
- ПС – переключення тригера в протилежний стан;
- Зб1(Зб0) – збереження одиничного (нульового) стану;

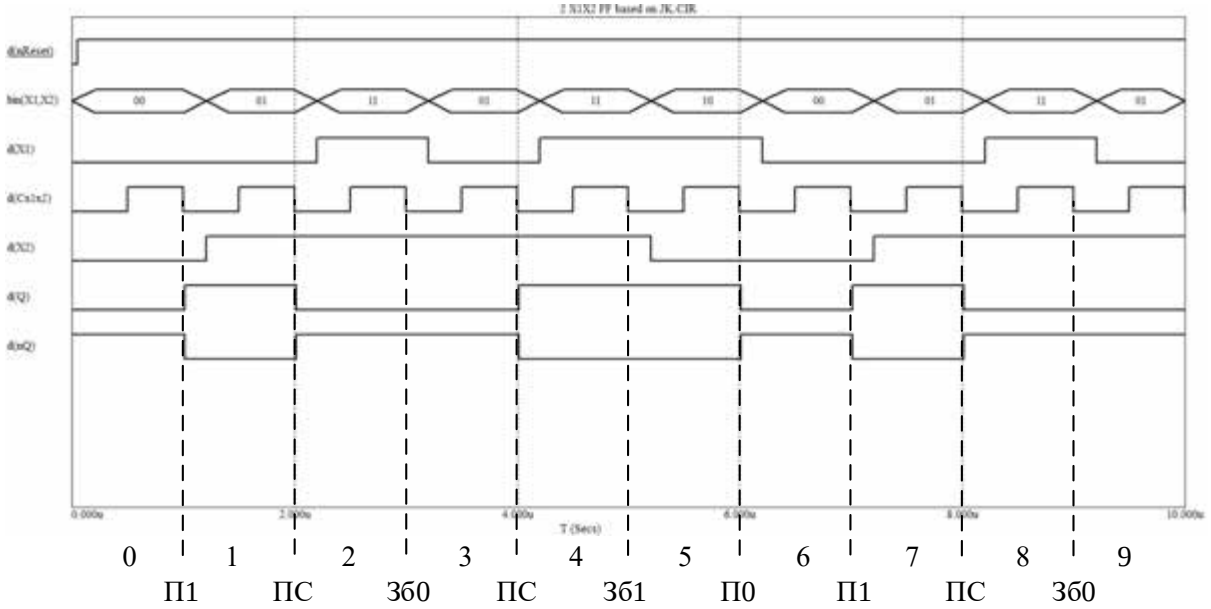


Рисунок 1.10 – Результати моделювання X_1CX_2-тригера

На початку моделювання (такт 0) на асинхронний вхід *nR* базового *JK*- тригера надходить активний (нульовий) рівень сигналу, в результаті чого базовий тригер і, звичайно, підсумковий тригер встановлюються в початковий нульовий стан. Після цього сигнал на вході *nR* переключається в неактивний рівень, дозволяючи переключення базового тригера за допомогою синхронних входів. Далі зміна значень сигналів на інформаційних входах X_1 і X_2 підібрані таким чином, щоб забезпечити перехід підсумкового тригеру у всі можливі стани: переключення в одиничний стан (такти 1, 7), переключення в нульовий стан (такт 6), переключення в протилежний (нульовий) стан (такти 2,8), переключення в протилежний (одиничний) стан (такт 4), збереження одиничного стану (такт 5), збереження нульового стану (такти 3, 9).

Зверніть також увагу, що для коректного спрацьовування базового двоступеневого тригера всі інформаційні сигнали підсумкового тригера переключаються тільки при неактивному для першого ступеню значенню сигналу синхронізації, щоб уникнути ситуацій захвату сигналу або проскакування фронту [1].

Далі розглянемо визначення динамічних параметрів підсумкового тригера на прикладі функціонування тригера в такті 3. Часова діаграма функціонування тригера в цьому такті приведена на рис.1.11. На цьому рисунку додатково показана часова діаграма вихідного сигналу функції збудження для керування входом *K*.

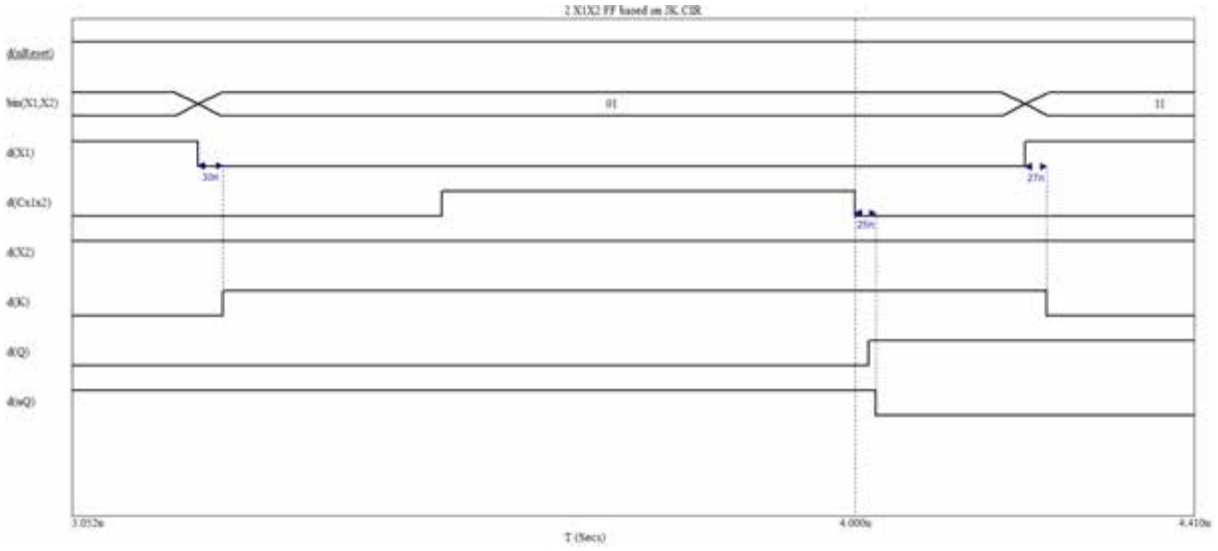


Рисунок 1.11 – Результати моделювання X_1CX_2-тригера в динамічному режимі

Процес спрацьовування підсумкового двоступеневого тригера може бути розділений на три послідовних етапи (рис.1.12).

На першому етапі спрацьовують елементи комбінаційної схеми формування функцій збудження (рис.1.8). Зі схеми на рис.1.8 випливає, що час затримки формування функцій збудження базового *JK*-тригера складається з формування сигналу на вході *K* ($\boldsymbol{t_K}$) базового тригера, в зв'язку з тим, що це коло є більш довгим, ніж коло формування сигналу *J* ($\boldsymbol{t_J}$):

$$t_J = t_2; \quad t_K = max(t_1,t_2) + max(t_3,t_4) + t_5; \quad t_{фз} = max(t_J,t_K),$$

де t_J (t_K) – час формування функцій збудження відповідно на вході *J* (*K*);

$t_{фз}$ – час формування функцій збудження базового тригера;

t_i – час затримки спрацьовування *i*-того логічного елемента;

max() – функція обчислення максимального значення серед заданих аргументів.

Очевидно, що за однакових затримок логічних елементів ($t_{зтр}$) отримаємо $t_J = t_{зтр}$; $t_K = 3t_{зтр}$. На часовій діаграмі момент закінчення формування функцій збудження позначено літерами *JK*.

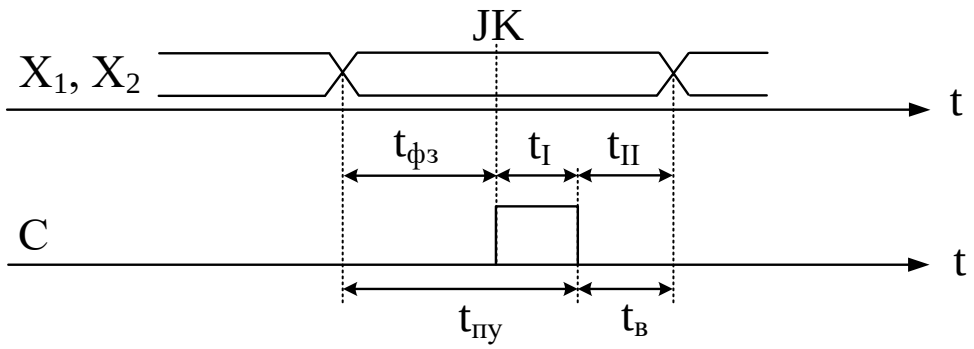


Рисунок 1.12 – Часова діаграма для визначення динамічних параметрів X_1CX_2-тригера

Після закінчення формування функцій збудження на другому етапі відбувається переключення сигналу синхронізації *C* в одиничний стан, в результаті чого перший ступінь базового тригера переключається у відповідний стан в залежності від значень сигналів на входах *J* і *K*. Очевидно, що тривалість одиничного стану сигналу *C* визначається часом спрацьовування першого ступеня базового тригера (***t*****$_I$**).

Таким чином, час попередньої установки підсумкового тригера ($t_{пу}$) [1] визначається як $t_{пу} = t_{фз} + t_I$, тобто впродовж цього інтервалу часу (рис.1.12) інформаційні сигнали на входах X_1 і X_2 не можна змінювати.

На третьому етапі сигнал *C* переключається в нульовий стан (формується задній фронт), в результаті чого переключається другий ступінь базового тригера. Тривалість цього етапу залежить від часу спрацьовування другого ступеня базового тригера (t_{II}) і визначає час витримки підсумкового тригера $t_в = t_{II}$ [1].

В результаті аналізу функціонування підсумкового тригера можна зробити висновок, що в інтервалі $t_{ny} + t_в$ заборонено змінювати інформаційні сигнали на входах X_1 і X_2.

Крім того, значення $t_{ny} + t_в$ визначає мінімальний період сигналу синхронізації, тобто визначає максимальну частоту надходження керуючого сигналу $f_{cmax} = 1/(t_{ny} + t_в)$.

З результатів моделювання (рис.1.11) можна побачити, що час спрацьовування схеми формування функцій збудження для третього такту (сигнал X_1 переключається в 0, а X_2 залишається в одиничному стані) складає *30*нс. При такому переключенні вхідних сигналів логічні елементи 2, 5 переключаються в одиницю, а елемент 4 – в нуль (послідовність переключення цих елементів $2 \rightarrow 4 \rightarrow 5$), тобто $t_{фз} = t_2^{01} + t_4^{10} + t_5^{01}$, де t_i^{10} (t_i^{01}) – час спрацьовування *i*-того логічного елемента при його переключенні з 1 в 0 (з 0 в 1).

Нагадаємо, що в середовищі ***MicroCap*** для моделювання динаміки будемо використовувати модель логічних елементів ***DLY_TTL*** (t^{10} = 8нс, t^{01} = 11нс), тобто, дійсно, $t_{фз} = 11 + 8 + 11 = 30$нс, що відповідає результатам моделювання.

Формування нуля на вході *K* базового тригера відбувається швидше (див. результати моделювання) $t_{фз} = t_2^{10} + t_4^{01} + t_5^{10} = 8 + 11 + 8 = 27$нс.

На часовій діаграмі (рис.1.11) можна побачити, що час спрацьовування базового тригера складає *25*нс, тобто, якщо базовий тригер є двоступеневим, то цей час відповідає часу спрацьовування другого ступеня цього тригера (більш детально див. [1]), тобто $t_{II} = 25$нс.

Вихід першого ступеня базового тригера недоступний користувачу, але, знаючи структурну організацію двоступеневих тригерів, можна зробити припущення, що $t_I \approx t_{II} = 25$нс.

Таким чином, мінімальний період сигналу синхронізації підсумкового тригера складає $27 + 25 + 25 = 77$нс.

На цьому виконання прикладу синтезу X_1CX_2-тригера закінчено.

Очевидно, що в результаті синтезу тип підсумкового тригера з точки зору структурної організації повністю співпадає з типом такої організації базового тригера, тобто, якщо базовий тригер є непрозорим (двоступеневим або з динамічним керуванням [1]), то і підсумковий тригер буде теж непрозорим. Активність рівня (для прозорих тригерів) або фронту сигналу синхронізації у базового і підсумкового тригерів співпадають.

Для того, щоб змінити активність сигналу синхронізації підсумкового тригера на протилежну достатньо проінвертувати цей сигнал перед підключенням до керуючого входу.

Якщо базовий тригер є прозорим (*DE*- або *RES*-тригери), а підсумковий тригер повинен бути непрозорим, то спочатку необхідно перетворити прозорий тригер у непрозорий двоступеневий тригер, а далі виконувати синтез (рис.1.13).

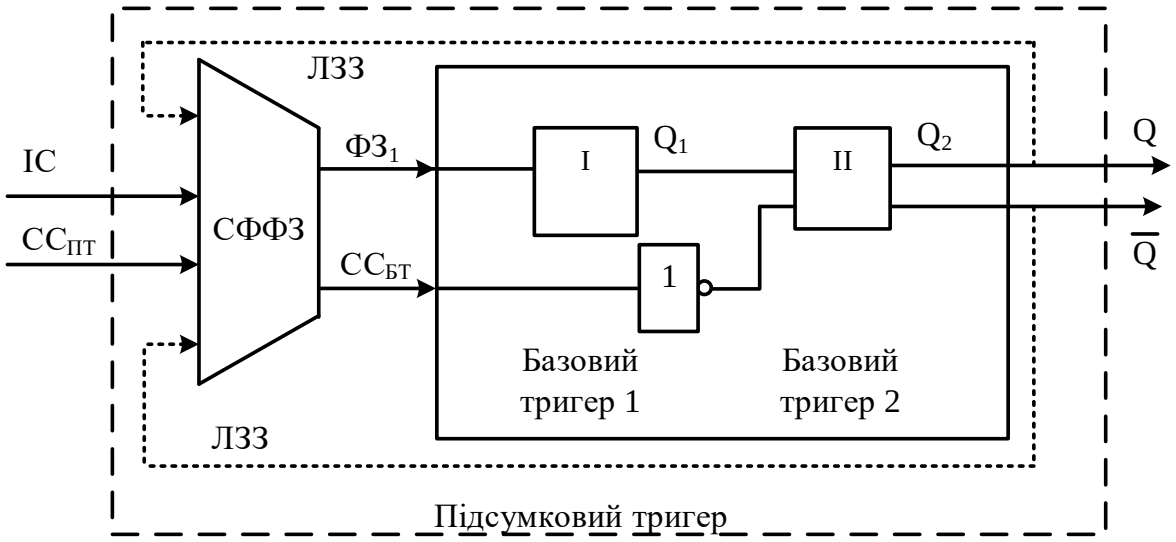


Рисунок 1.13 – Структурна схема непрозорого підсумкового тригера на базі прозорих тригерів

На структурній схемі використовуються ті ж позначення, що і на рис.1.1. Виключенням є відома структура організації двоступеневого тригера [1], на якій цифрами I (II) показані відповідно перший (другий) ступені, а *Q1* (*Q2*) позначають виходи цих ступенів. Звичайно, що в загальному випадку тригери першого і другого ступенів можуть відноситися до різних типів

прозорих тригерів. Крім того, в підсумковому тригері можуть бути присутні асинхронні входи *Set* і *Reset* з будь-яким типом керування (на рис.1.13 не показані, але вхід *nReset* показаний в прикладі на рис.1.8).

Далі розглянемо функціональні схеми, які використовуються для реалізації стандартних тригерів на основі інших стандартних тригерів. Це може стати в нагоді в зв'язку з тим, що в інтегральному виконання випускаються тільки *D*- і *JK*-тригери.

На початку розглянемо реалізацію стандартних тригерів на основі *JK*- тригера (рис.1.14).

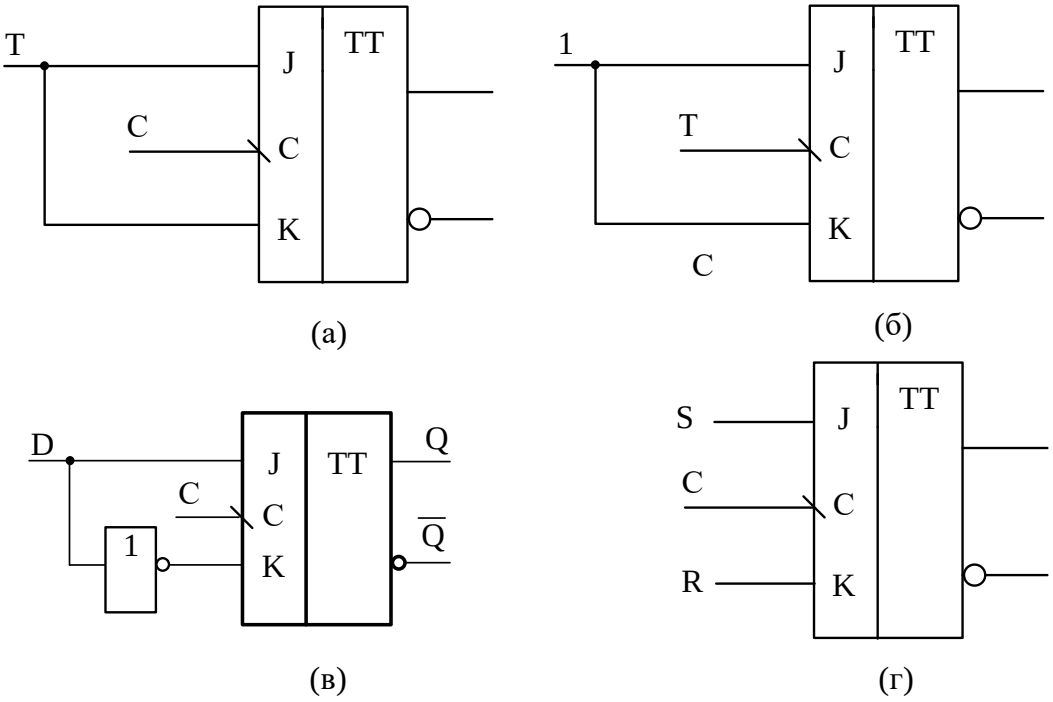


Рисунок 1.14 – Реалізація стандартних тригерів на основі *JK*-тригера

На рис.1.14,а, б, в, г приведені функціональні схеми *TC*-, *T*-, *D*- і *RCS*- тригерів відповідно. З наведених схем можна побачити, що реалізація будь-якого стандартного тригера на базі *JK*-тригера є доволі простою задачею (тільки в одному випадку використовується додатковий інвертор). Це є однією з причин, чому *JK*-тригер також називають ***універсальним*** тригером.

Далі розглянемо реалізацію стандартних тригерів (*D*-, *JK*-, *T*-) на основі *RCS*-тригера (рис.1.15).

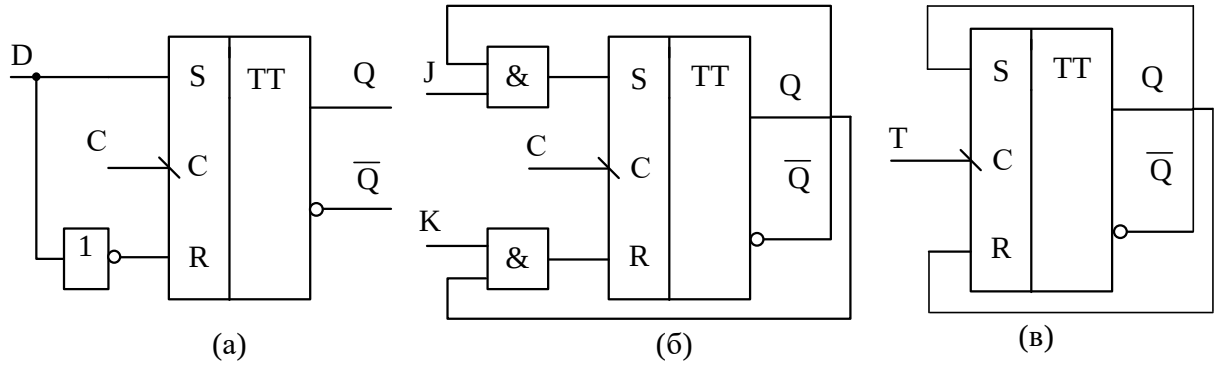


Рисунок 1.15 – Реалізація стандартних тригерів на основі *RCS*- тригера

І, наприкінці, розглянемо схеми перетворення *T*- і *DV*-тригерів на основі *D*-тригера (рис.1.16).

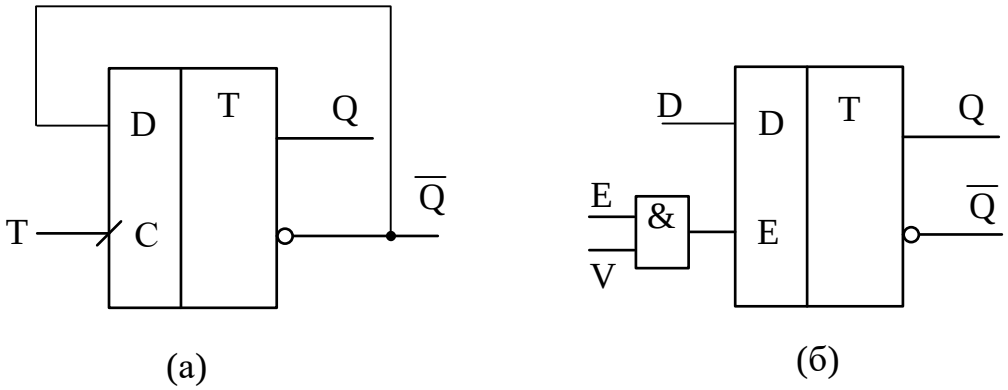


Рисунок 1.16 – Реалізація *T*-, *DV*-тригерів на базі *D*-тригера

Всі розглянуті вище схеми перетворення тригерів, звичайно, можуть бути отримані за допомогою методики синтезу тригерів на основі базових тригерних схем.

Контрольні завдання та запитання

1. Поясніть методику синтезу тригерів на базі типових тригерних схем.
2. З якою метою здійснюється синтез тригерів на базі типових тригерних схем?
3. Поясніть термін «базовий тригер».

4. Поясніть термін «підсумковий тригер».
5. Які типи синхронних тригерів випускаються в інтегральному виконанні?
6. Приведіть узагальнену структурну схему тригера, побудованого на основі типового тригера.
7. Поясніть призначення складових структурної схеми, яка приведена на рис.1.1.
8. З якою метою використовується схема формування функцій збудження?
9. Поясніть термін «функція збудження» тригера.
10. Яким чином необхідно визначати функції збудження базового тригера?
11. Чому лінії зворотного зв’язку на рис.1.1 показані пунктиром?
12. При виконанні яких типових мікрооперацій в схемі підсумкового тригера з’являються лінії глобального зворотного зв’язку?
13. В чому полягає різниця між функціями збудження та функцією переходів тригерів?
14. Яку схему необхідно синтезувати, щоб забезпечити проєктування тригера на основі типових тригерних схем?
15. До якого класу цифрових схем відноситься схема формування функцій збудження? Обґрунтуйте відповідь.
16. Поясніть, чому на наборах 4-7 в таблиці 1.1 значення функцій збудження інформаційних сигналів (колонки 6, 7) позначені символом «*»?
17. Поясніть визначення функцій збудження в колонках 6, 7 на наборах 0-3 таблиці 1.1.
18. Поясніть, яким чином заповнюється колонка 5 таблиці 1.1.
19. Прокоментуйте заповнення колонки 4 таблиці 1.1.
20. Як визначити мінімальні логічні вирази для реалізації функцій збудження базового тригера?

21. Як довести коректність синтезу підсумкового тригера?
22. Як виконати синтез нетипової тригерної схеми?
23. Як визначити значення динамічних параметрів підсумкового тригера?
24. Як визначити час переключення підсумкового тригера?
25. Як визначити величину мінімального періоду сигналу синхронізації?
26. Як визначити величину максимальної частоти надходження сигналу синхронізації?
27. Як розрахувати час затримки спрацьовування підсумкового тригера?
28. Прокоментуйте побудову функціональної схеми на рис.1.8.
29. Для чого використовується сигнал ***nReset*** в схемі на рис.1.8?
30. Яким чином необхідно задавати вхідні сигнали на підсумковий тригер для виконання перевірки коректності функціонування цього тригера?
31. Поясніть, яким чином задані вхідні сигнали на рис.1.10?
32. Поясніть взаємодію у часі інформаційних і керуючого сигналів підсумкового тригера на прикладі часової діаграми, приведеної на рис.1.10.
33. Виконання якої мікрооперації перевіряється в тактах 2, 4, 8 на часовій діаграмі на рис.1.10?
34. В чому полягає різниця у виконанні мікрооперацій в тактах 2, 4, 8 на часовій діаграмі на рис.1.10?
35. Виконання якої мікрооперації перевіряється в тактах 1, 3, 7 на часовій діаграмі на рис.1.10?
36. В чому полягає різниця у виконанні мікрооперацій в тактах 1, 3, 7 на часовій діаграмі на рис.1.10?
37. Виконання якої мікрооперації перевіряється в тактах 0, 6 на часовій діаграмі на рис.1.10?
38. Виконання якої мікрооперації перевіряється в такті 5 на часовій діаграмі на рис.1.10?

39. Охарактеризуйте послідовність переключення логічних елементів в тригері, схема якого приведена на рис.1.8.
40. Як визначити значення динамічних параметрів в схемі підсумкового тригера на рис.1.8?
41. Поясніть визначення динамічних параметрів на часовій діаграмі на рис.1.12.
42. Які динамічні параметри можна побачити на часовій діаграмі на рис.1.11?
43. Які динамічні параметри не приведені на часовій діаграмі на рис.1.11 і за якої причини?
44. Поясніть термін «час попереднього встановлення» тригера.
45. Поясніть термін «час витримки» тригера.
46. Чим характеризується інтервал часу $t_{пу} + t_в$ на рис.1.12?
47. Чим характеризується момент часу, позначений на рис.1.12 як *JK*?
48. Який тип підсумкового тригера з точки зору структурної організації може бути отриманий в результаті синтезу?
49. Як змінити активність фронту непрозорого підсумкового тригера на протилежну по зрівнянню з базовим тригером?
50. Як синтезувати непрозору тригерну схему на базі прозорого тригера?
51. Прокоментуйте структурну схему, приведену на рис.1.13.
52. Поясніть, чому *JK*-тригер називається універсальним тригером?
53. Яким чином визначається тип підсумкового тригера з урахуванням реакції на появу завад?
54. Виконати синтез *D*-тригера на основі *JK*-тригера.
55. Виконати синтез *RCS*-тригера на основі *JK*-тригера.
56. Виконати синтез *TC*-тригера на основі *JK*-тригера.
57. Виконати синтез *T*-тригера на основі *JK*-тригера.
58. Виконати синтез *RCS*-тригера зі спрацьовуванням за заднім фронтом синхросигналу на основі *JK*-тригера зі спрацьовуванням за переднім фронтом синхросигналу.

59. Виконати синтез *D*-тригера на основі *RCS*-тригера.
60. Виконати синтез *TC*-тригера на основі *RCS*-тригера.
61. Виконати синтез *T*-тригера на основі *RCS*-тригера.
62. Виконати синтез *JK*-тригера на основі *RCS*-тригера.
63. Виконати синтез *RCS*-тригера на основі *D*-тригера.
64. Виконати синтез *TC*-тригера на основі *D*-тригера.
65. Виконати синтез *T*-тригера на основі *D*-тригера.
66. Виконати синтез *JK*-тригера на основі *D*-тригера.
67. Виконати синтез *RCS*-тригера на основі *TC*-тригера.
68. Виконати синтез *D*-тригера на основі *TC*-тригера.
69. Виконати синтез *JK*-тригера на основі *TC*-тригера.
70. Виконати синтез *T*-тригера на основі *TC*-тригера.
71. Виконати синтез *JK*-тригера на основі *DE*-тригера.
72. Виконати синтез непрозорого *RCS*-тригера на основі *DE*-тригера.
73. Виконати синтез *TC*-тригера на основі *DE*-тригера.
74. Виконати синтез *T*-тригера на основі *DE*-тригера.
75. Виконати синтез *TC*-тригера на основі *RES*-тригера.
76. Виконати синтез *T*-тригера на основі *RES*-тригера.
77. Виконати синтез непрозорого *D*-тригера на основі *RES*-тригера.
78. Виконати синтез *JK*-тригера на основі *RES*-тригера.
79. Виконати синтез *DE*-тригера на основі *RES*-тригера.
80. Виконати синтез *DE*-тригера на основі *nRES*-тригера.
81. Виконати синтез *DE*-тригера на основі *REnS*-тригера.
82. Виконати синтез *DE*-тригера на основі *nREnS*-тригера.
83. Як визначити реакцію підсумкового тригера на підключення забороненої вхідної комбінації?
84. Чи може бути реалізований прозорий тригер на базі двоступеневого тригера за вказаною в розділі методикою? Обґрунтуйте відповідь.
85. Виконати синтез *DE*-тригера на основі *DV*-тригера.
86. Виконати синтез *DV*-тригера на основі *DE*-тригера.

87. Виконати синтез *RES*-тригера на основі *DV*-тригера.

88. Виконати синтез *DV*- тригера на основі *RES*-тригера.

2 РЕГІСТРИ

Регістр (***register***) – цифровий пристрій, який призначений для виконання мікрооперацій прийому та збереження інформації.

Нагадаємо, що будь-яка мікрооперація виконується за допомогою апаратури цифрового пристрою або вузла впродовж одного такту функціонування комп'ютерної системи.

Якщо порівняти визначення тригера (тригер – цифровий пристрій, який призначений для виконання мікрооперацій прийому та збереження одного біту інформації.) і регістра, то можна зробити висновок, що будь- який регістр являє собою сукупність тригерів.

Таким чином, однією з найважливіших характеристик регістра є його розрядність, тобто кількість базових тригерів, що входить до складу регістра (наприклад, якщо процесор є 64-розрядним, то це означає, що до складу кожного регістра цього процесора входить 64 тригери).

Будемо нумерувати тригери регістра, починаючи з нуля, причому розряд з більшим номером є більш старшим, тобто в *n*-розрядному регістрі розряди нумеруються від 0 до *n-1*. Таким чином, в 64-розрядному регістрі тригери нумеруються з 0 до 63, де розряд 63 є найстаршим, а розряд 0 – наймолодшим. Така нумерація розрядів в сучасних комп'ютерах при використанні чисел з фіксованою комою (цілих чисел) дозволяє ставити у відповідність номер розряду $\boldsymbol{k}$ і вагу його двійкового еквіваленту $\boldsymbol{2^k}$.

Як правило, регістри будуються з використанням одного типу базових тригерів, але в загальному випадку до складу регістра можуть входити тригери кількох типів.

Таким чином, регістри, як і тригери, відносяться до класу цифрових схем з пам'яттю (послідовнісних схем), тобто стан цих пристроїв залежить,

як від значень множини вхідних сигналів, так й від значення попереднього стану таких пристроїв.

В результаті, стан регістра може бути описаний відповідно до виразу $Q^{t+1} = f(X, Q^t)$, де X є множиною вхідних сигналів (як інформаційних, так і керуючих), а $Q^{t+1}(Q^t)$ визначає наступний (попередній) стан регістра.

Виходячи з визначення регістра, цей пристрій виконує дві мікрооперації:

- завантаження інформації;

- збереження інформації.

Як правило, виходи регістрів є доступними користувачу в будь-який момент часу, але в деяких випадках в складі регістра можуть бути присутні кола для зчитування інформації з виходів регістру, тобто кола, які забезпечують доступ користувача до виходів базових тригерів регістра при виконанні певних умов (наприклад, кола для керування третім станом регістра тощо).

Мікрооперація завантаження інформації до регістра може бути виконана кількома способами:

- паралельне завантаження;

- послідовне завантаження;

- послідовно-паралельне завантаження.

При використанні мікрооперації паралельного завантаження розрядність регістра збігається з розрядністю вхідної шини даних, тобто завантаження одного інформаційного слова відбувається за один такт (в сучасних комп’ютерних системах таким чином функціонують так звані паралельні порти, наприклад, в мікроконтролерних системах).

Для виконання послідовного завантаження в регістр використовується однорозрядна шина даних незалежно від розрядності регістра. При такому способі завантаження за один такт в регістр записується один біт інформації. В якості прикладу використання такого способу завантаження можна привести так званий послідовний порт (***Serial Port***, наприклад, зараз

використовується в мікроконтролерах), ***USB*** (***Universal Serial Bus*** – універсальна послідовна шина). В більшості комп'ютерних мереж також використовується послідовна передача інформації.

Послідовно-паралельне завантаження використовується достатньо рідко (розрядність вхідної шини даних ***m*** визначається виразом $1 < m < n$, де n – розрядність регістра). Такий спосіб може використовуватися в мікроконтролерних системах, якщо, наприклад, розрядність сенсора менша за розрядність порту.

2.1 Регістри на базі окремих тригерів

З точки структурної організації регістрів та способу завантаження інформації розрізняють:

- регістри з керованою синхронізацією;
- регістри з некерованою синхронізацією.

2.1.1 Регістри з керованою синхронізацією

Структурна схема n-розрядного регістра з керованою синхронізацією приведена на рис.2.1. На структурній схемі використовуються такі позначення та скорочення:

- *БТ* – базові тригери регістра;

n – розрядність регістра (кількість тригерів);

- n-$1…0$ – номери базових тригерів;
- *COM* – комутатор;

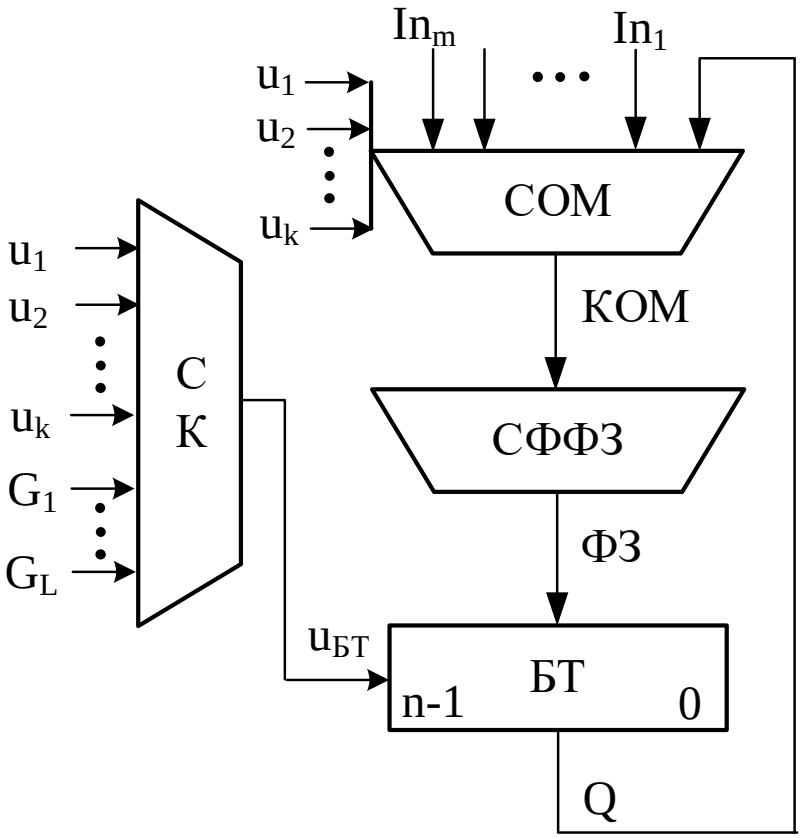


Рисунок 2.1 – Структурна схема регістра з керованою синхронізацією

- *КОМ* – вихідна шина комутатора;
- *СФФЗ* – схема формування функцій збудження базового тригера;
- *ФЗ* – функцій збудження базового тригера;
- *Q* – вихідний стан регістра, який є множиною станів окремих тригерів $Q = \{Q_{n-1}, Q_{n-2},\ldots, Q_1, Q_0\}$, де Q_i – стан *i*-того тригера регістра;
- In_i – вхідна шина даних для завантаження інформації з *i*- того напряму;
- *m* – кількість зовнішніх вхідних шин даних;
- u_i – сигнал керуючого автомату, який призначений для виконання *i*- тої мікрооперації;
- *k* – кількість мікрооперацій, які виконує регістр;
- *СК* – схема керування регістром;
- G_i – сигнал *i*-того генератора прямокутних імпульсів;
- *L* – кількість джерел прямокутних імпульсів;

- $u_{БТ}$ – внутрішній керуючий сигнал регістра, призначений для дозволу завантаження інформації в базові тригери.

Відповідно до позначення складових вузлів *СОМ*, *СФФЗ*, *СК* регістра можна побачити, що ці частини являють собою комбінаційні вузли.

Розглянемо призначення цих складових частин регістра.

Комутатор *СОМ* призначений для забезпечення вибору одного з вхідних джерел, інформацію з якого необхідно завантажити до регістра в залежності від активного значення керуючого сигналу u_i, що надходить з керуючого автомата. При цьому на виході комутатора з'являється стан ліній того джерела вхідної інформації, яке було обрано для завантаження регістра відповідним керуючим сигналом u_i (наприклад, за активного значення керуючого сигналу u_i на виході комутатора з'являється стан ліній вхідної шини даних In_i).

Схема формування функцій збудження *СФФЗ* призначена для забезпечення завантаження інформації з виходів комутатора в базові тригери регістра в залежності від типу цих тригерів.

Схема керування *СК* призначена для забезпечення функціонування комутатора, схеми формування функцій збудження та базових тригерів регістра.

Структурна схема на рис.2.1 є узагальненою, тобто в деяких випадках може бути спрощена:

- якщо регістр виконує тільки одну мікрооперацію, то в схемі регістра буде відсутній комутатор;

- якщо в якості базових тригерів використовуються *D*-тригери, то в регістрі буде відсутня *СФФЗ*;

- якщо регістр виконує завантаження інформації тільки із зовнішніх джерел, то в схемі регістра відсутні лінії зворотного зв'язку з виходів базових тригерів на входи комутатора.

Розглянемо також використання генераторів прямокутних імпульсів (ГПІ) для побудови регістра. Якщо для керування регістром застосовується

один сигнал ГПІ ($L = 1$), то система керування називається однофазною. У випадку використання двох сигналів ГПІ ($L = 2$) система керування називається двофазною. Якщо $L > 2$, то система керування називається багатофазною.

Як правило, для побудови регістра використовується однофазна система синхронізації. Двофазна система може застосовуватися, наприклад, при використанні двотактових базових тригерів [1].

В результаті розгляду структурної схеми та призначення складових вузлів регістра з керованою синхронізацією можна зробити висновок, що для проєктування такого регістра достатньо виконати синтез комбінаційних вузлів регістра *СОМ*, *СФФЗ*, *СК*. Звичайно, що синтез базових тригерів не здійснюється в зв'язку з тим, що під час побудови регістра використовуються вже готові тригерні схеми [1].

Таким чином, послідовність кроків для виконання синтезу регістра полягає в наступному:

1. Аналіз функціонування регістра при виконанні заданих мікрооперацій.
2. Синтез комутатора *СОМ*.
3. Синтез схеми керування *СК*.
4. Синтез схеми формування функцій збудження *СФФЗ* базових тригерів.
5. Перетворення отриманих в п.п.2, 3, 4 логічних виразів комбінаційних вузлів регістра в заданий базис логічних елементів (в разі необхідності).
6. Реалізація логічної схеми регістра.
7. Перевірка коректності синтезу за допомогою моделювання і/або фізичної реалізації пристрою.
8. Визначення динамічних параметрів регістра.

Далі детально розглянемо синтез комбінаційних вузлів регістра.

2.1.1.1 Синтез комутатора

Синтез комутатора може бути проведений двома способами:

- за допомогою таблиці істинності (класичний спосіб);

- за допомогою таблиці мікрооперацій (МО).

Класичний спосіб передбачає розробку таблиці істинності для одного i-того розряду комутатора $КОМ_i = f(u_1, u_2,\ldots, u_k, In_{1i}, In_{2i},\ldots, In_{mi})$, де In_{ji} – значення i-того розряду вхідної шини даних In_j ($j = 1, m$). Якщо регістр може завантажувати інформацію тільки із зовнішніх джерел, то $k = m$. Але використання цього методу обмежується кількістю вхідних змінних (для проведення мінімізації «вручну» за допомогою карт Карно кількість вхідних змінних, як правило, не повинно перевищувати 6 або необхідно використовувати комп'ютерні методи мінімізації).

Використання метода таблиць мікрооперацій базується на тому, що сукупність значень керуючих сигналів являє собою або унітарний код (в будь-який момент часу може бути активним не більше одного керуючого сигналу), або нульовий код – в разі неактивних значень всіх керуючих розрядів регістра.

До складу таблиці входять k рядків, де k – кількість мікрооперацій, які виконуються регістром, та n стовпців, де n – кількість розрядів (тригерів) в регістрі.

Узагальнений вид таблиці мікрооперацій приведений в табл.2.1.

Таблиця 2.1 – Узагальнена таблиця МО для синтезу комутатора

МО	$КОМ_{n-1}$	$КОМ_{n-2}$	…	$КОМ_1$	$КОМ_0$
u_1					
u_2					
…					
u_k					

Кожний стовпчик таблиці мікрооперацій відображує стан відповідної вихідної лінії комутатора в залежності від певного активного значення керуючого сигналу u_i.

Для виконання синтезу комутатора за допомогою таблиці МО необхідно виконати два кроки:

1. Заповнення клітин таблиці мікрооперацій.

2. Визначення логічних виразів для кожного виводу комутатора.

Заповнення клітин цієї таблиці виконується для кожного керуючого сигналу (кожного рядка) окремо незалежно від вмісту клітин інших рядків. Якщо регістр виконує паралельне завантаження інформації, то фрагмент заповненої таблиці мікрооперацій для керуючого сигналу u_i має вигляд, як приведено в табл.2.2.

Таблиця 2.2 – Фрагмент заповненої таблиці МО для керуючого сигналу u_i

МО	KOM_{n-1}	KOM_{n-2}	…	KOM_1	KOM_0
u_1	…	…	…	…	…
u_i	$In_{j\ n-1}$	$In_{j\ n-2}$	…	$In_{j\ 1}$	$In_{j\ 0}$
u_k	…	…	…	…	…

Вміст кожної клітини визначає стан певної лінії вхідної шини даних, яка була обрана комутатором в якості джерела інформації згідно з активним значенням відповідного керуючого сигналу.

Наприклад, клітина, що пофарбована в табл.2.2 сірим кольором, визначає стан розряду *n-2* комутатора за активності керуючого сигналу u_i. При цьому на виході цього розряду з'являється стан *n-2*-ої лінії вхідної шини даних In_j, що і відображено у цій клітині ($In_{j\ n-2}$).

Після заповнення таблиці мікрооперацій необхідно визначити логічні вирази, що описують функціонування комутатора.

Ці вирази складаються для кожної вихідної лінії комутатора, тобто для кожного стовпчика таблиці, і записуються у вигляді диз'юнктивної нормальної форми (ДНФ).

Нагадаємо, що ДНФ є способом запису будь-якої логічної функції у вигляді диз'юнкції термів, причому кожний терм є кон'юнктивним (кон'юнкцією вхідних змінних).

Кожний терм логічного виразу для виходу комутатора є кон'юнкцією вмісту поточної клітини і відповідного керуючого сигналу.

Наприклад, для таблиці мікроопераций (по сигналу u_1 виконується завантаження з вхідної шини In_1, по сигналу u_2 – з вхідної шини In_2 і т.д.), яка приведена в табл.2.3, логічні вирази для побудови комутатора будуть виглядати таким чином:

Таблиця 2.3 – Приклад заповненої таблиці мікроопераций

MO	KOM_{n-1}	KOM_{n-2}	…	KOM_1	KOM_0
u_1	$In_{1\ n-1}$	$In_{1\ n-2}$	…	$In_{1\ 1}$	$In_{1\ 0}$
…	…	…	…	…	…
u_i	$In_{i\ n-1}$	$In_{i\ n-2}$	…	$In_{i\ 1}$	$In_{i\ 0}$
…	…	…	…	…	…
u_k	$In_{k\ n-1}$	$In_{k\ n-2}$	…	$In_{k\ 1}$	$In_{k\ 0}$

$$KOM_{n-1} = In_{1\ n-1} \cdot u_1 \vee \ldots \vee In_{i\ n-1} \cdot u_i \vee \ldots \vee In_{k\ n-1} \cdot u_k ;\ i = 1, k.$$

$$\ldots \qquad (2.1)$$

$$KOM_0 = In_{1\ 0} \cdot u_1 \vee \ldots \vee In_{i\ 0} \cdot u_i \vee \ldots \vee In_{k\ 0} \cdot u_k ;\ i = 1, k.$$

Вирази (2.1) можна записати в загальному вигляді:

$$KOM_j = In_{1\ j} \cdot u_1 \vee \ldots \vee In_{i\ j} \cdot u_i \vee \ldots \vee In_{k\ j} \cdot u_k,\ j = 0, n-1;\ i = 1, k. \qquad (2.2)$$

Аналогічні логічні вирази можна отримати, використовуючи класичний метод синтезу за допомогою таблиці істинності.

Використання таблиці мікроопераций для синтезу комутатора є більш простішим і наочним, ніж використовування класичного методу синтезу особливо за великої кількості заданих мікроопераций.

2.1.1.2 Синтез схеми керування

Схема керування є комбінаційною схемою, тобто синтез будемо проводити за допомогою таблиці істинності.

Відповідно до структурної схеми регістра (рис.2.1) схема керування *СК* формує сигнал синхронізації $u_{БТ}$ базових тригерів, який забезпечує завантаження в ці тригери інформації, яка була обрана комутатором *СОМ* та перетворена схемою формування функцій збудження *СФФЗ*, тобто $u_{БТ} = f(u_1, u_2, \ldots, u_k, G_1, G_2, \ldots, G_L)$. При цьому, як правило, в регістрах використовується однофазна система синхронізації. В цьому разі можна записати $u_{БТ} = f(u_1, u_2, \ldots, u_k, G)$. Звідси можна побачити, що в загальному випадку кількість вхідних змінних може бути завелика для виконання синтезу за допомогою таблиці істинності.

Під час проєктування цифрових пристроїв зазвичай використовується невелика сукупність елементарних логічних функцій (кон'юнкція, диз'юнкція, додавання за модулем 2 та їх інверсії, а також функція заперечення), які утворюють більш складну за структурою і функціональним призначенням функцію, використовуючи метод ***суперпозиції***.

Для спрощення синтезу виконаємо декомпозицію [2, 3, 4] логічної функції $u_{БТ}$.

Нагадаємо, що декомпозиція являє собою операцію поділу будь-якої досліджуваної системи на множину окремих взаємопов'язаних підсистем, тобто система розділяється на більш прості частини. Звичайно, що до будь- якої такої частини можна також застосувати операцію декомпозиції.

В процесі проєктування цифрових пристроїв декомпозиція нерозривно пов'язана зі зворотним процесом композиції, тобто об'єднанням та узгодженням раніше декомпонованих окремих вузлів в єдиний пристрій з його аналізом коректності функціонування та визначення параметрів такого пристрою.

Декомпозиція логічних функцій проводиться за допомогою методу (операції) суперпозиції і передбачає утворення нової функції з кількох початкових функцій, тобто така функція представляється у вигляді суперпозиції двох або більшої кількості логічних функцій.

Це пов'язано з тим, що область значень логічних функцій $\{0,1\}$ збігається з множиною допустимих значень її аргументів $\{0,1\}$. Це означає, що будь-яка логічна функція може бути підставлена в якості аргументу в іншу логічну функцію. Така підстановка називається суперпозицією логічних функцій.

Таким чином, метод суперпозицій дає можливість заміни частини аргументів початкової функції іншою функцією від цих аргументів, тобто функція $f(a,\ b,\ c)$ може бути представлена у вигляді $f(a,\ f_{bc}(b, c))$, де $f_{bc}(b, c)$ – функція від аргументів (b, c). Наприклад, логічна функція $y = a{\cdot}b{\cdot}c{\cdot}d{\cdot}e$ може бути представлена, використовуючи проміжні функції y_{ab} та y_{de}: $y_{ab} = a{\cdot}b$; $y_{de} = d{\cdot}e$. В результаті початкова функція y отримає вигляд $y = y_{ab}{\cdot}c{\cdot}y_{de}$. Звичайно, що процес декомпозиції останньої функції далі може бути продовжений.

Декомпозиція логічних функцій, як правило, проводиться для можливості проведення процедури синтезу таких функцій за умови великої кількості змінних (при проведенні неавтоматизованого синтезу не більше шести) або для спрощення синтезу і подальшого аналізу функціонування логічної функції, яка має різні групи вхідних змінних за функціональним призначенням (наприклад, інформаційні та керуючі групи входів), і в цьому випадку виконується декомпозиція функції по цих групах входів.

Таким чином, логічна функція від великої кількості аргументів може бути реалізована на базі кількох логічних функцій з меншою кількістю аргументів. Така можливість дає змогу застосовувати обмежену сукупність логічних елементів з невеликою кількістю входів для схемної реалізації логічних функцій з будь-якою кількістю аргументів. В результаті, використання принципу суперпозиції логічних функцій приводить до

застосування каскадного з'єднання логічних елементів, які забезпечують апаратну реалізацію функції.

Одним зі способів декомпозиції логічної функції по змінній x_i використовується так зване розкладання (декомпозиція) Шеннона, відповідно до якого будь-яка логічна функція може бути розкладена на логічні функції меншої кількості аргументів. При цьому декомпозиція може бути виконана по будь-якому аргументу та за будь-якою кількістю аргументів.

Відповідно до розкладання Шеннона логічна функція може бути представлена наступним чином:

$$y = x_i \cdot f_{1i}(x_{n-1}, \ldots, x_{i+1}, 1, x_{i-1}, \ldots, x_0) \vee \overline{x_i} \cdot f_{0i}(x_{n-1}, \ldots, x_{i+1}, 0, x_{i-1}, \ldots, x_0); \quad (2.3)$$

$$y = (x_i \vee f_{0i}(x_{n-1}, \ldots, x_{i+1}, 0, x_{i-1}, \ldots, x_0)) \cdot (\overline{x_i} \vee f_{1i}(x_{n-1}, \ldots, x_{i+1}, 1, x_{i-1}, \ldots, x_0)), \quad (2.4)$$

де $f_{1i}(x_{n-1}, \ldots, x_{i+1}, 1, x_{i-1}, \ldots, x_0) = f_{1i}(x_{n-1}, \ldots, x_{i+1}, x_{i-1}, \ldots, x_0)$ – часткова логічна функція при $x_i = 1$ відносно аргументу x_i;

$f_{0i}(x_{n-1}, \ldots, x_{i+1}, 0, x_{i-1}, \ldots, x_0) = f_{0i}(x_{n-1}, \ldots, x_{i+1}, x_{i-1}, \ldots, x_0)$ – часткова логічна функція при $x_i = 0$ відносно аргументу x_i.

В зв'язку з тим, що терми у виразі (2.3) одночасно ніколи не дорівнюють одиниці, то операцію диз'юнкції можна замінити функцією додавання за модулем 2.

Далі для скорочення будемо позначати $f_{0i}(x_{n-1}, \ldots, x_{i+1}, x_{i-1}, \ldots, x_0) = f_{0xi}(x)$; $f_{1i}(x_{n-1}, \ldots, x_{i+1}, x_{i-1}, \ldots, x_0) = f_{1xi}(x)$.

Для визначення часткових логічних функцій за допомогою таблиці істинності необхідно скласти дві таблиці істинності і, відповідно, дві карти Карно для кожної з функцій $f_{0xi}(x)$ і $f_{1xi}(x)$.

Якщо є потреба, то операцію декомпозиції можна продовжити відносно іншого аргументу і т.д.

Продовжимо виконувати синтез схеми керування *СК* регістром.

Проведення процедури синтезу *СК* полягає у визначенні логічного виразу, який описує формування сигналу $u_{БТ} = f(u_1, u_2, \ldots, u_k, G)$ при використанні однофазної синхронізації.

Синтез схеми керування можна виконати класичним методом, тобто логічні вирази та схема формування сигналу $u_{БТ}$ може бути отримані за допомогою таблиці істинності.

В таблиці істинності активний стан сигналу $u_{БТ}$ будемо позначати логічною одиницею, а неактивний – логічним нулем. Нагадаємо, що активне значення сигналу синхронізації залежить від схеми базового тригера і може відповідати як певному рівню, так і фронту цього сигналу. Таким чином, в таблиці істинності активне значення керуючого сигналу позначаємо логічною одиницею.

Для заповнення таблиці істинності будемо використовувати такі правила функціонування *СК*:

1. За відсутності активного значення сигналу *G* базові тригери регістра повинні знаходитися в режимі зберігання інформації, тобто значення сигналу $u_{БТ}$ є неактивним.

2. В разі неактивного значення всіх керуючих сигналів навіть при активному *G* вихідний сигнал схеми керування $u_{БТ}$ також приймає неактивне значення.

3. Сигнал $u_{БТ}$ приймає активне значення, якщо тільки один з керуючих сигналів регістра є активним, звичайно за активності *G*.

4. Решта вхідних наборів таблиці істинності відповідає випадкам, коли за активності *G* два або більше керуючих сигналів є активними.

Заповнення таблиці істинності може бути здійснено двома основними способами, які розглянуті нижче.

Використання першого способу пов’язано з тим, що проєктування керуючого автомату проводиться таким чином, що керуючі сигнали, які відповідають за здійснення завантаження даних в регістр, можуть надходити тільки так, що тільки один з цих керуючих сигналів є активним або жодного,

тобто ситуація, коли надходять відразу два або більше цих активних сигналів є неможливою і забороненою. В цьому випадку логічна функція, що описує формування сигналу $u_{БТ}$ є недовизначеною, і таблиця істинності на цих наборах заповнюється символом «*». При використанні цього способу таблиця істинності у заповненому вигляді показана в табл.2.4.

Таблиця 2.4 – Таблиця істинності формування $u_{БТ}$ першим способом

Номери наборів	G	u_k	…	u_i	…	u_2	u_1	$u_{БТ}$
0	0	0	…	0	…	0	0	0
1	0	0	…	0	…	0	1	0
2	0	0	…	0	…	1	0	0
…	…	…	…	…	…	…	…	0
i	0	0	…	1	…	0	0	0
…	…	…	…	…	…	…	…	0
k	0	1	…	0	…	0	0	0
Решта наборів при $G = 0$	…	…	…	…	…	…	…	* або 0
2^k	1	0	…	0	…	0	0	0
2^k+1	1	0	…	0	…	0	1	1
2^k+2	1	0	…	0	…	1	0	1
…	…	…	…	…	…	…	…	1
2^k+i	1	0	…	1	…	0	0	1
…	…	…	…	…	…	…	…	1
2^k+k	1	1	…	0	…	0	0	1
Решта наборів при $G = 1$	…	…	…	…	…	…	…	*

Далі необхідно виконати мінімізацію логічного виразу $u_{БТ}$, наприклад, за допомогою карти Карно.

В результаті мінімізації (цей процес буде показаний пізніше під час розглядання прикладів синтезу регістра) може бути отриманий такий логічний вираз:

$$u_{БТ} = G \cdot (u_1 \vee u_2 \vee \ldots \vee u_i \vee \ldots \vee u_k),\ i = 1, k; \qquad (2.5)$$

За великої кількості керуючих сигналів регістра таблиця істинності і карта Карно набуває великих обсягів і є дуже незручними для «ручної»

обробки, тому в таких випадках доцільно проводити декомпозицію логічної функції.

Розглянемо декомпозицію логічної функції, яка описує формування сигналу $u_{БТ}$. Для цього розділимо вхідні змінні на групи за функціональним призначенням (G і сукупність керуючих сигналів u) та виконаємо декомпозицію і визначимо часткові функції по аргументу G.

Як вже було відзначено вище для того, щоб визначити часткові функції, необхідно скласти дві таблиці істинності для кожної з функцій $f_{0G}(u)$ і $f_{1G}(u)$, де u – сукупність керуючих сигналів регістра, що надходять з керуючого автомату. Для цього можна використати табл.2.4, причому перша половина цієї таблиці відповідає таблиці істинності часткової функції $f_{0G}(u)$, а друга половина – $f_{1G}(u)$.

З першої половини таблиці 2.4 (за неактивного G) випливає, що $f_{0G}(u) = 0$.

З другої половини таблиці 2.4 за допомогою карт Карно можна отримати $f_{1G}(u) = u_1 \vee u_2 \vee \ldots \vee u_i \vee \ldots \vee u_k$, $i = 1, k$.

Таким чином, відповідно до розкладання Шеннона (2.3) отримаємо

$$u_{БТ} = \overline{G}\cdot 0 \vee G\cdot(u_1 \vee u_2 \vee \ldots \vee u_i \vee \ldots \vee u_k) =$$
$$= G\cdot(u_1 \vee u_2 \vee \ldots \vee u_i \vee \ldots \vee u_k),$$

що відповідає (2.5).

Аналогічно можна використовувати розкладання (2.4):

$$u_{БТ} = (\overline{G} \vee (u_1 \vee u_2 \vee \ldots \vee u_i \vee \ldots \vee u_k)) \cdot (0 \vee G) =$$
$$= G\cdot(u_1 \vee u_2 \vee \ldots \vee u_i \vee \ldots \vee u_k).$$

Виконуючи аналіз логічних виразів (2.2) і (2.5), можна побачити, що керуючі сигнали $u_1, \ldots, u_k$ одночасно надходять на входи комутатора та схеми керування. Відповідно до структурної схеми регістра (рис.2.1) сигнали з виходів схеми керування і комутатора (через *СФФЗ*) надходять на входи базових тригерів, тобто в схемі спостерігаються розгалуження сигналів $u_1, \ldots, u_k$, що далі сходяться на інформаційних і керуючих входах тригерів.

Таким чином, в регістрі існують розгалуження сигналів, що сходяться, тобто функціонування регістру залежить від співвідношення затримок логічних елементів комутатора і схеми керування. Це є ознакою наявності гонок сигналів (класифікація гонок приведена в [1]).

При виконанні умови $t_{COM} > t_{CK}$, де t_{COM} – затримка спрацьовування комутатора; t_{CK} – затримка спрацьовування схеми керування, сигнали з виходів комутатора надходять на інформаційні входи базових тригерів вже після появи активного фронту сигналу синхронізації. Таким чином, тригери регістра будуть спрацьовувати некоректно, в зв'язку з тим, що в момент появи активного фронту сигналу синхронізації на виходах комутатора ще не сформувалися коректні значення вихідних сигналів, які надходять на інформаційні входи базових тригерів (звичайно, що достатньо наявності хоча б одного такого виходу).

Для забезпечення коректного функціонування регістра необхідно дотримуватися певних часових вимог між сигналами G і u_i. Часова діаграма взаємодії цих сигналів приведена на рис.2.2.

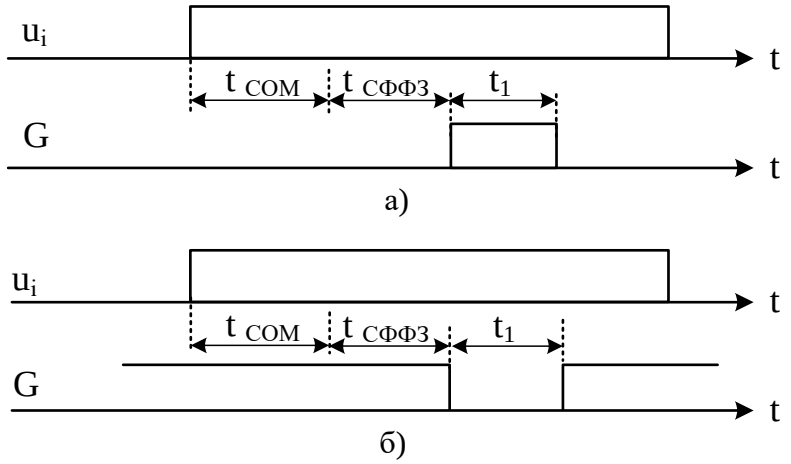


Рисунок 2.2 – Часова діаграма надходження сигналів u_i і G для забезпечення коректного функціонування регістра

На рис.2.2 використовуються такі позначення:

- t_{COM} – час затримки спрацьовування комутатора;

- $t_{СФФЗ}$ – час затримки спрацьовування схеми формування функцій збудження базових тригерів;
- t_1 – час затримки спрацьовування першого ступеня двотактового тригера.

З рис.2.2 випливає, що сигнал G в часі вкладений в сигнал u_i, що забезпечує появу активного фронту сигналу синхронізації вже після спрацьовування комутатора регістра. Крім того, на рис.2.2,а показані значення динамічних параметрів сигналів G і u_i, яких необхідно дотримуватися для забезпечення коректної роботи регістра.

Часова діаграма на рис.2.2,а ілюструє організацію процесу завантаження даних до регістра при використанні двоступеневих тригерів зі спрацьовуванням за заднім фронтом синхросигналу в якості базових тригерів регістра. В цьому випадку спочатку спрацьовує комутатор за час t_{COM}, далі – *СФФЗ* ($t_{СФФЗ}$), а після цього вже можна переводити сигнал G до одиничного рівня, за яким відбувається завантаження інформацій в перший ступінь базового тригера. Таким чином, в момент появи заднього фронту сигналу G дані на інформаційних входах базових тригерів вже відповідають правильнимним значенням та завантажені до першого ступеня базового тригера, тобто регістр спрацьовує коректно.

В результаті можна отримати, що для правильного спрацьовування регістра необхідно забезпечити виконання наступних умов:

$$t_{uG} \geq t_{COM} + t_{СФФЗ} + t_1 , \qquad (2.6)$$

де t_{uG} – проміжок часу між появою активного значення керуючого сигналу u_i і активного фронту сигналу синхронізації базового тригера G.

Часова діаграма на рис.2.2,б ілюструє організацію процесу завантаження даних до регістра при використанні двоступеневих тригерів зі спрацьовуванням за переднім фронтом синхросигналу в якості базових тригерів регістра. При цьому для підвищення завадостійкості тригера доцільно формувати сигнал G таким чином, щоб активне значення цього сигналу відповідало низькому рівню. З рис.2.2,б випливає, що для

правильного спрацьовування регістра необхідно також забезпечити виконання умови (2.6).

Якщо в якості базових тригерів регістра використовуються тригери з динамічним керуванням, то вираз (2.6) може бути представлений у такому вигляді:

$$t_{uG} \geq t_{COM} + t_{СФФЗ} + t_{ny}\ , \qquad (2.7)$$

де t_{ny} – час попередньої установки базового тригера [1].

Час затримки спрацьовування регістра t_{RG}, який побудований на основі двоступеневого тригера, визначається за виразим $t_{RG} = t_2$, де t_2 – час спрацьовування другого ступеня базового тригера регістра.

Таким чином, мінімальний період слідування сигналу G (T_{Gmin}), а, отже, і максимальна частота надходження цього сигналу (f_{Gmax}) визначається за виразом:

$$T_{Gmin} = t_{uG\,min} + t_2 = t_{COM} + t_{СФФЗ} + t_1 + t_2\ ;\ f_{Gmax} = 1/T_{Gmin}, \qquad (2.8)$$

У випадку використання тригерів з динамічним керуванням в якості базових тригерів регістра отримаємо вираз:

$$T_{Gmin} = t_{uG\,min} + t_{mp} = t_{COM} + t_{СФФЗ} + t_{ny} + t_{mp};\ f_{Gmax} = 1/T_{Gmin}, \qquad (2.9)$$

де t_{mp} – час затримки спрацьовування базового тригера.

Далі розглянемо другий спосіб заповнення таблиці істинності для визначення логічного виразу для реалізації сигналу $u_{БТ}$.

Використання другого способу пов’язано з можливістю одночасної появи на входах регістра кількох керуючих сигналів, що може бути пов’язано з існуванням гонок сигналів в керуючих автоматах. В цьому випадку для підвищення завадостійкості та надійності функціонування цифрових пристроїв з метою уникнення впливу гонок вхідні набори, які відповідають активності двох або більшої кількості керуючих сигналів, відмічають нулем (табл.2.5). Перша половина табл.2.5 збігається з відповідною частиною табл.2.4, тому в табл.2.5 показана тільки частина, що відповідає активному значенню сигналу G.

Таблиця 2.5 – Таблиця істинності формування $u_{БТ}$ другим способом

Номери наборів	G	u_k	…	u_i	…	u_2	u_1	$u_{БТ}$
2^k	1	0	…	0	…	0	0	0
2^k+1	1	0	…	0	…	0	1	1
2^k+2	1	0	…	0	…	1	0	1
…	…	…	…	…	…	…	…	1
2^k+i	1	0	…	1	…	0	0	1
…	…	…	…	…	…	…	…	1
2^k+k	1	1	…	0	…	0	0	1
Решта наборів при $G = 1$	…	…	…	…	…	…	…	0

В результаті отримаємо логічний вираз для реалізації сигналу $u_{БТ}$:

$$u_{БТ} = G\cdot(u_1\cdot\overline{u_2}\cdot\ldots\cdot\overline{u_k}\vee\ldots\vee\overline{u_1}\cdot\overline{u_2}\cdot u_i\cdot\overline{u_{i+1}}\cdot\ldots\cdot\overline{u_k}\vee\overline{u_1}\cdot\overline{u_2}\cdot\ldots\cdot\overline{u_{k-1}}\cdot u_k). \quad (2.10)$$

Звичайно, що реалізація виразу (2.10) потребує набагато більше апаратних витрат, ніж вираз (2.5), але він дозволяє отримати більш велику надійність роботи регістра під час взаємодії з керуючим автоматом. Як проміжний варіант в логічному виразі для реалізації сигналу $u_{БТ}$ частина термів може відповідати виразу (2.10), а решта – виразу (2.5).

2.1.1.3 Синтез схеми формування функцій збудження

Синтез схеми формування функцій збудження *СФФЗ* на відміну від процедури синтезу комутатора і схеми керування передбачає врахування типу базових тригерів регістру.

Нагадаємо, що *СФФЗ* використовується для формування відповідних функцій збудження з метою забезпечення приймання інформації з виходів комутатора в тригери регістра згідно з типом цих тригерів.

Синтез виконується для кожного з типів базових тригерів, які входять до складу регістра. Як правило, для побудови регістра використовується один

з типів базових тригерів. Процедура синтезу проводиться для одного базового тригера (Q_i) кожного типу і містить наступні кроки:

1. Розробка таблиці переходів для завантаження значення стану вихідного сигналу *i*-того розряду комутатора до базового тригера.

2. Визначення функцій збудження для кожного типу базового тригера регістра.

3. Мінімізація функцій збудження.

Функції збудження залежать тільки від типу базових тригерів і не залежать від мікрооперацій, що виконує регістр, тому для синтезу будь- якого регістра достатньо розглянути синтез функцій збудження для всіх стандартних тригерів: *D*-, *JK*-, *RCS*- і *TC*-тригерів.

Таким чином, синтез достатньо проводити для одного *i*-того розряду регістра.

Об'єднаємо кроки 1 і 2 процедури синтезу та розглянемо сумісну таблицю переходів *СФФЗ* і функцій збудження базових тригерів (табл.2.6).

Таблиця 2.6 – Таблиця переходів для синтезу *СФФЗ*

Номери наборів	$u_{БТ}$	$КОМ_i$	Q_i^t	Q_i^{t+1}	C_i	D_i	R_i	S_i	J_i	K_i	T_i
	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	*	*	*	*	*	*
1	0	0	1	1	0	*	*	*	*	*	*
2	0	1	0	0	0	*	*	*	*	*	*
3	0	1	1	1	0	*	*	*	*	*	*
4	1	0	0	0	1	0	*	0	0	*	0
5	1	0	1	0	1	0	1	0	*	1	1
6	1	1	0	1	1	1	0	1	1	*	1
7	1	1	1	1	1	1	0	*	*	0	0

В колонках 1 - 4 міститься таблиця переходів схеми формування функцій збудження (крок 1 процедури синтезу).

Перша половина таблиці переходів (рядки 0 - 3) відповідає неактивному значенню сигналу $u_{БТ}$ (нагадаємо, що неактивне значення позначаємо нулем), відповідно до якого регістр перебуває в стані збереження

інформації. Це означає, що новий стан тригерів регістра Q_i^{t+1} (стовпчик 4) не відрізняється від попереднього стану Q_i^t (стовпчик 3), тобто $Q_i^{t+1} = Q_i^t$.

Друга половина таблиці переходів (рядки 4 - 7) відповідає активному значенню сигналу $u_{БТ}$, яке позначаємо логічною одиницею. За активним значенням сигналу $u_{БТ}$ в регістрі відбувається запис станів виходів комутатора $КОМ_i$ (стовпчик 2) до базових тригерів регістра (стовпчик 4), тобто виконується мікрооперація $Q_i^{t+1} = КОМ_i$.

Далі здійснюється визначення функцій збудження для кожного типу базового тригера регістра (крок 2 процедури синтезу). Спочатку визначаються функція збудження для входу синхронізації тригера (стовпчик 5), а далі – функції збудження інформаційних входів тих тригерів, які використовуються в регістрі (стовпчики 6, 7-8, 9-10 або 11).

Детальний опис визначення функцій збудження приведений в [1] та розділі 1 цього посібника.

Після цього виконується мінімізація функцій збудження базових тригерів (третій крок процедури синтезу).

В зв'язку з ідентичністю стовпців 1 і 5 можна отримати функцію збудження для входу синхронізації базових тригерів: $C_i = u_{БТ}$.

Карта Карно для мінімізації функцій збудження *D*-тригера приведена на рис.2.3.

З карти Карно на рис.2.3 отримаємо функцію збудження для приймання інформації до *D*-тригера: $D_i = КОМ_i$.

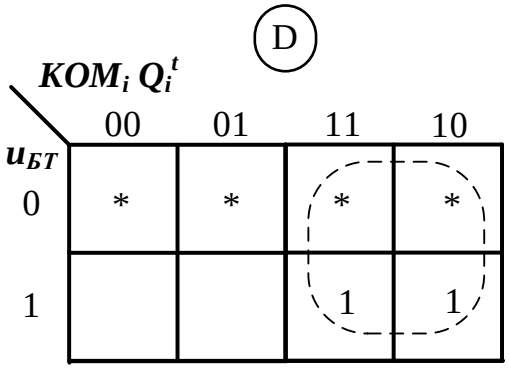


Рисунок 2.3 – Мінімізація функції збудження *D*-тригера

Карта Карно для мінімізації функцій збудження *RCS*-тригера приведена на рис.2.4.

З карти Карно на рис.2.4 отримаємо функції збудження для приймання інформації до *RCS*-тригера: $S_i = KOM_i$; $R_i = \overline{KOM_i}$.

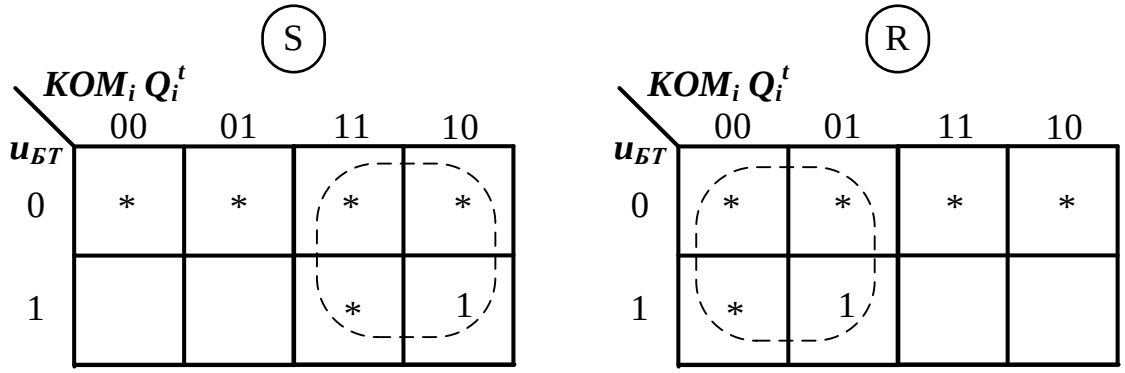


Рисунок 2.4 – Мінімізація функцій збудження *RCS*-тригера

Карта Карно для мінімізації функцій збудження *JK*-тригера приведена на рис.2.5.

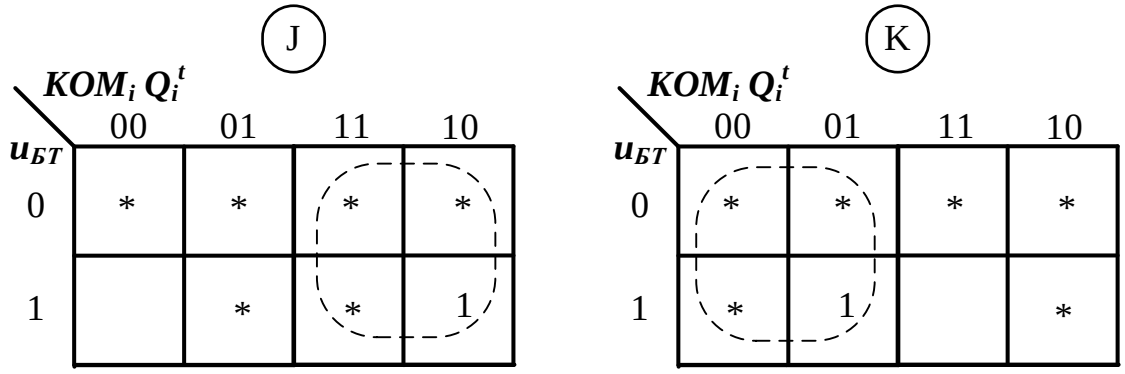


Рисунок 2.5 – Мінімізація функцій збудження *JK*-тригера

З карти Карно на рис.2.5 отримаємо функції збудження для приймання інформації до *JK*-тригера: $J_i = KOM_i$; $K_i = \overline{KOM_i}$.

Карта Карно для мінімізації функції збудження *TC*-тригера приведена на рис.2.6.

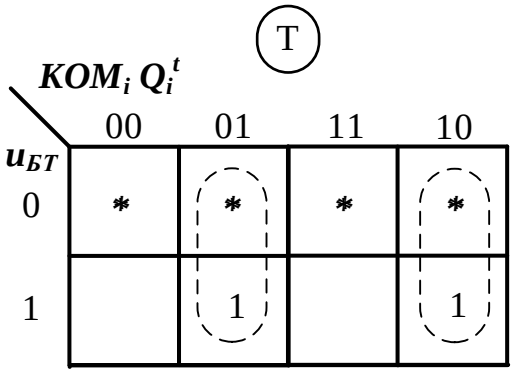


Рисунок 2.6 – Мінімізація функції збудження *TC*-тригера

З карти Карно на рис.2.6 отримаємо функцію збудження для приймання інформації до *TC*-тригера:

$$T_i = КОМ_i \cdot \overline{Q_i^t} \vee \overline{КОМ_i} \cdot Q_i^t = КОМ_i \oplus Q_i^t.$$

З цього виразу можна зробити висновок, що *T*-тригер виконує логічну функцію додавання за модулем два (*XOR*) вхідного сигналу на вході *T* та попереднього стану цього тригера.

Таким чином, розглянуто процедуру синтезу всіх комбінаційних частин регістра.

2.1.1.4 Виконання операції зсуву в регістрах

Як було відзначено раніше однією з мікрооперацій, які можуть виконувати регістри, є послідовне завантаження інформації. Виконання цієї мікрооперації здійснюється за допомогою використання зсувів вмісту регістра. Регістри, які виконують хоча б одну мікрооперацію зсуву, називаються регістрами зсуву.

Розглянемо класифікацію зсувів вмісту регістра, які використовуються в комп’ютерах.

Зсуви розрізняються відповідно до наступних класифікаційних ознак:

- за напрямом зсуву розрізняють зсуви вправо і вліво:

- за способом виконання зсувів розрізняють логічні, циклічні і арифметичні зсуви.

Під час виконання зсуву вміст регістра може зсуватися в напряму від старших розрядів до молодших, що відповідає зсуву вправо, або навпаки – від молодших розрядів до старших, що відповідає зсуву вліво.

Як відзначалося раніше, будемо нумерувати розряди регістра, починаючи з нуля. При цьому розряд з більшим номером вважається більш старшим. Таким чином, при зсуві вправо інформація в *n*-розрядному регістрі зсувається від (*n-1)*-ого розряду до нульового, а при зсуві вліво, навпаки, – від нульового розряду до (*n-1)*-ого розряду.

На рис.2.7 приведений приклад зсуву вмісту однобайтного регістра на один розряд вправо (рис.2.7,а) і вліво (рис.2.7,б).

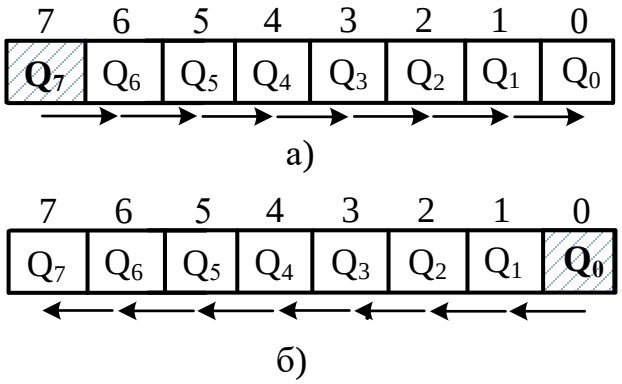


Рисунок 2.7 – Зсув вмісту регістра на один розряд

На рис.2.8 приведений приклад зсуву вмісту однобайтного регістра на два розряди вправо.

З рисунків 2.7 і 2.8 можна побачити, що в загальному випадку при виконанні зсуву на *n* розрядів вправо (вліво) звільнюються *n* старших (молодших) розрядів регістра (на рис.2.7, 2.8 такі розряди для заданих зсувів показані заштрихованою ділянкою та шрифтом ***Bold***).

В залежності від того, яким чином заповнюються розряди, що звільнюються під час зсуву, розрізняють 3 різновиди зсувів:

- логічний зсув;

- циклічний зсув;

- арифметичний зсув.

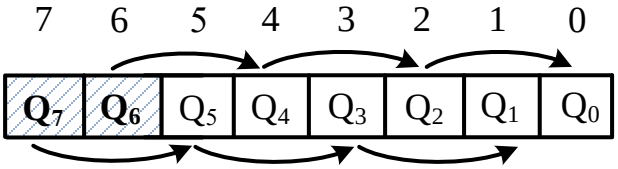


Рисунок 2.8 – Зсув вмісту регістра на два розряди вправо

Логічний зсув використовується для обробки беззнакових операндів. При виконанні логічного зсуву розряди, що звільнюються, заповнюються із зовнішніх джерел інформації або константними значеннями нуля або одиниці.

Під час виконання циклічного зсуву на *n* розрядів вправо (вліво) вміст регістра зсувається вправо (вліво) на *n* позицій, за винятком молодших (старших) біт, які записуються на місце відповідних старших (молодших) розрядів.

Арифметичний зсув використовується для зсуву знакових чисел. Під час виконання цього зсуву знаковий або знакові (при використанні, наприклад, модифікованих доповняльних чи обернених кодів [2, 3]) розряди не змінюються, але в зсуві приймають участь.

Далі розглянемо позначення різних типів і різновидів зсувів, які будемо використовувати в цьому посібнику.

Типи зсувів вліво або вправо будемо відповідно позначати літерами *L* (***left***) або *R* (***right***) , після яких вказується число розрядів *k*, на яке здійснюється зсув, наприклад, *R1*, *R2*, *R4*, *L2*, *L1* тощо.

Для використання тих чи інших різновидів зсувів будемо використовувати такі правила.

Циклічний зсув будемо позначати літерою *C* (***cyclic***) перед типом зсуву. Наприклад, *CLk* означає циклічний зсув вліво на *k* двійкових розрядів, а *CRk* – циклічний зсув вправо на *k* двійкових розрядів.

На рис.2.9 приведений приклад виконання циклічного зсуву вправо на один двійковий розряд *CR1* (рис.2.9,а) і на два розряди *CR2* (рис.2.9,б). На рис.2.9 показані як вміст регістра до початку виконання зсуву, так і після виконання зсуву.

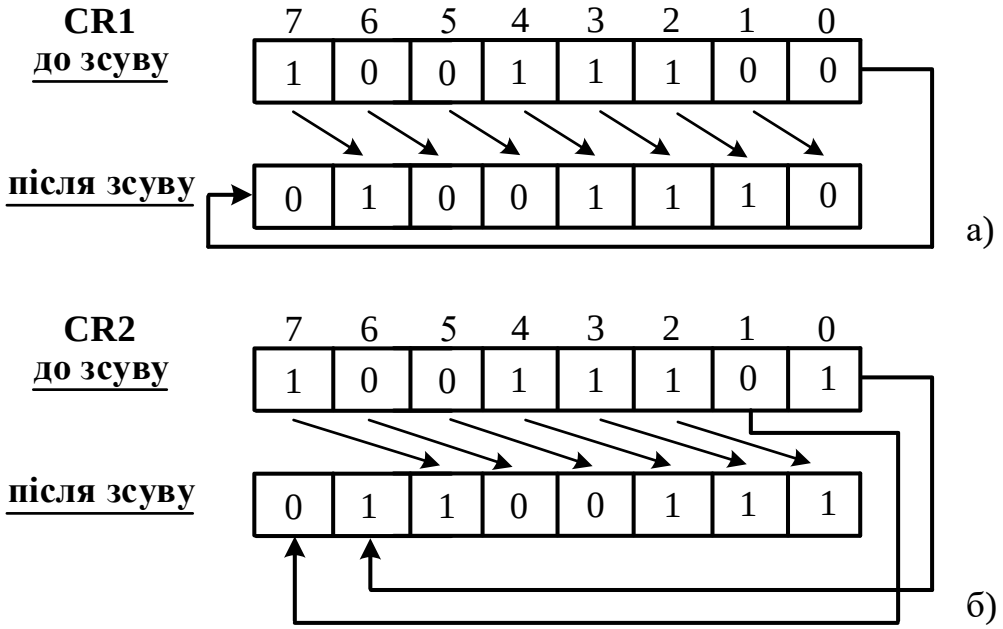


Рисунок 2.9 – Приклад виконання циклічного зсуву вправо на 1 і 2 біти

На рис.2.10 приведений приклад виконання циклічного зсуву вліво на один двійковий розряд *CL1*.

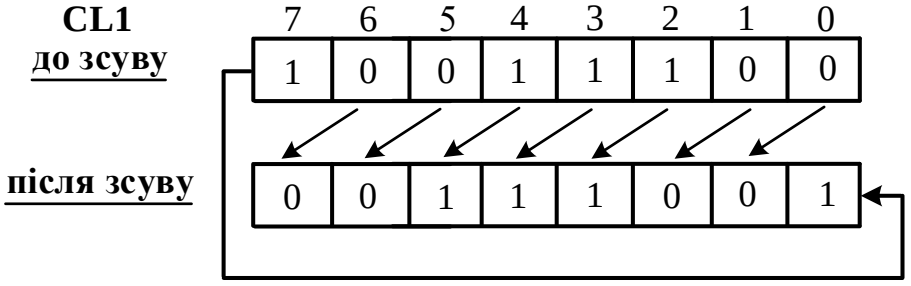


Рисунок 2.10 – Приклад виконання циклічного зсуву вліво на 1розряд

При позначенні логічного зсуву будемо вказувати значення бітів, які записуються у розряди, що звільнюються, безпосередньо до позначення зсуву при виконанні зсуву вправо (звільняються старші біти) або після позначення зсуву при зсуві вліво (звільняються молодші біти). Значення цих бітів будемо відокремлювати крапкою від позначення зсуву.

Наприклад,

L1.1 – логічний зсув на один біт вліво з записом одиниці до молодшого розряду, що звільнився;

L1.0 – логічний зсув на один біт вліво з записом нуля до молодшого розряду, що звільнився;

1.R1 (*0.R1*) – логічний зсув на один біт вправо з записом одиниці (нуля) до старшого розряду, що звільнився;

L2.XX (*XX.R2*) – логічний зсув на 2 біти вліво (вправо) з записом значень *XX*, де *XX* – двійкові цифри, в розряди, що звільнюються під час зсуву.

Як вже відзначалося раніше, в звільнені під час логічного зсуву розряди можуть бути записані не тільки константні значення нулів або одиниць, але й інформація із зовнішніх джерел. Так, в сучасних процесорах під час виконання арифметичних операцій використовується ознака переносу (***carry flag: CF***), а в системі команд існують команди зсувів з участю цієї ознаки. Серед таких команд можна відзначити команди циклічного зсуву з участю ознаки переносу, наприклад, команди *RCR* (***Rotate through Carry Right***), *RCL* ***Rotate through Carry Left***). Наприклад, при виконанні команди *RCR* (циклічний зсув вправо з ознакою переносу) в старший біт регістра записується вміст ознаки *CF*, а до *CF* записується молодший біт регістра (рис.2.11), тобто з точки зору регістра виконується логічний зсув вправо з заповненням вільного розряду вмістом ознаки *CF* (*CF.R1*).

Логічний зсув виконується також при виконанні послідовного завантаження інформації, тобто при перетворенні послідовного коду в паралельний. При цьому використовується однобітна шина даних *SB* (***serial***

bus), по якій послідовно, біт за бітом, надходять дані, починаючи з молодших розрядів. При цьому регістр виконує логічний зсув вмісту вправо з заповненням звільненого розряду з шини *SB* (*SB.R1*). Для отримання, наприклад, одного байту інформації мікрооперація зсуву виконується 8 разів.

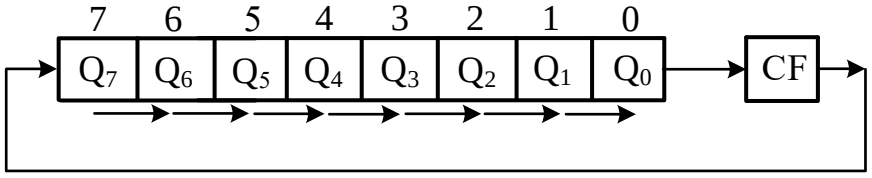


Рисунок 2.11 – Приклад виконання команди *RCR*

На рис.2.12 приведений приклад виконання логічного зсуву на один розряд вправо з заповненням вільного розряду сигналом *X*, тобто при *X* = 0 виконується зсув *0.R1*, а при *X* = 1 – *1.R1*.

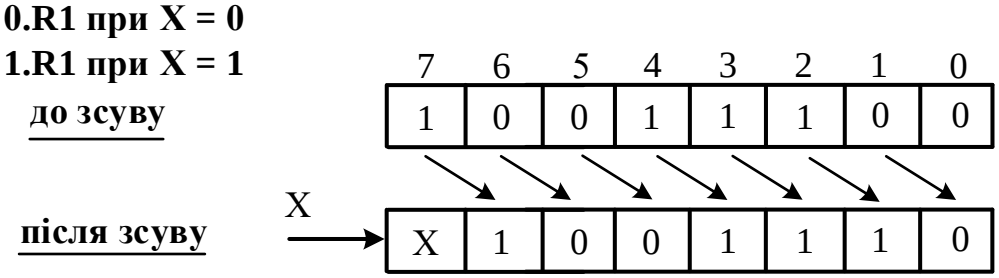


Рисунок 2.12 – Виконання логічного зсуву вправо на один біт

На рис.2.13 приведений приклад виконання логічного зсуву на два розряди вправо *01.R2* з заповненням двох старших вільних розрядів значеннями сигналів $X_1 = 0$, і $X_2 = 1$.

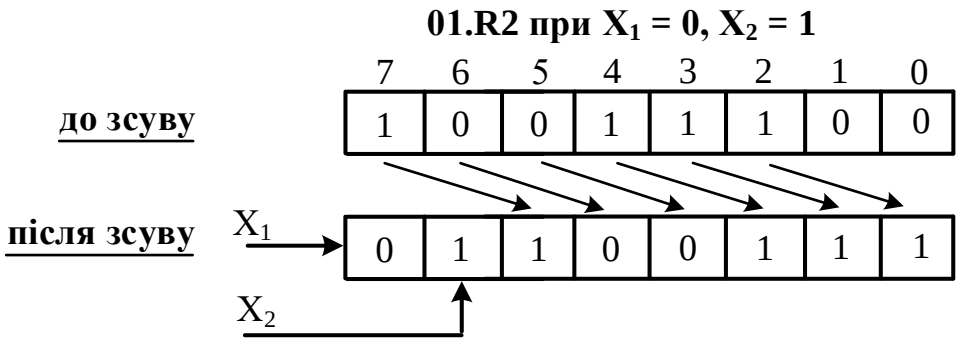


Рисунок 2.13 – Виконання логічного зсуву вправо на два біти

Аналогічно виконується логічний зсув на два розряди вліво *L2.10* з заповненням двох молодших вільних розрядів значеннями сигналів $X_1 = 1$, і $X_2 = 0$. (рис.2.14).

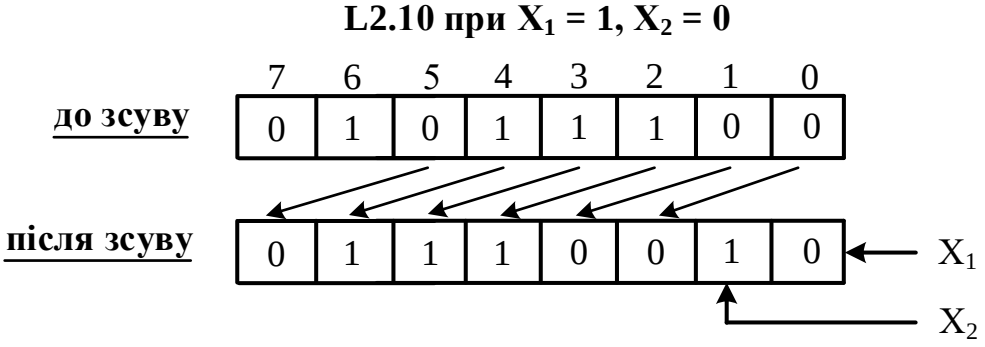


Рисунок 2.14 – Виконання логічного зсуву вліво на два біти

Арифметичний зсув використовується для обробки операндів зі знаками під час виконання арифметичних операцій множення і ділення. Розглянемо окремо виконання арифметичного зсуву вправо і вліво.

Арифметичний зсув вправо будемо позначати літерою *A* (***arithmetic***) перед типом зсуву. Наприклад, *ARk* означає арифметичний зсув вправо на *k* двійкових розрядів.

Як вже було відзначено раніше під час виконання арифметичного зсуву вміст найстаршого (знакового) розряду не змінюється, але в зсуві цей розряд приймає участь. На рис.2.15 приведений приклад виконання арифметичного зсуву на один (рис.2.15,а) і два розряди (рис.2.15,а) вправо *AR1* і *AR2* відповідно.

На цьому рисунку можна побачити, що значення знакового розряду (біт 7) не змінюється та копіюється в сусідній розряд (біт 7) при зсуві на один біт або в два сусідні розряди (біти 6,5) при зсуві на два біти.

Розглянемо виконання арифметичного зсуву вліво. В загальному випадку при виконанні цього зсуву на *k* розрядів знаковий розряд теж не змінюється, але в *k* молодших розрядів записується *k*-розрядний код із зовнішніх джерел інформації, тобто цей різновид зсуву має ознаки як

арифметичного, так і логічного зсуву. Таким чином, арифметичний зсув вліво також будемо позначати літерою *A* (***arithmetic***) перед типом зсуву, а після позначення типу зсуву через крапку будемо вказувати *k*-розрядний код, який записується в молодші розряди. Наприклад, *AL1.0* означає арифметичний зсув вліво на один двійковий розряд з записом нуля в молодший розряд.

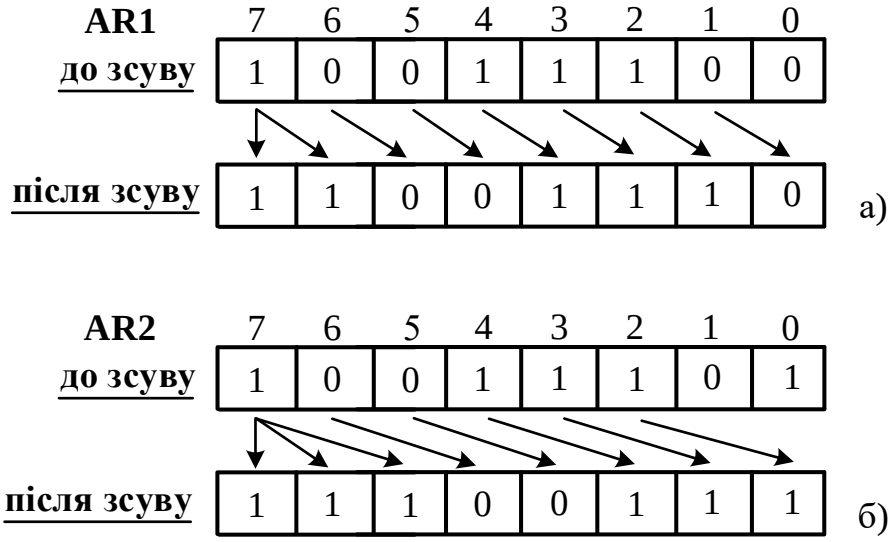


Рисунок 2.15 – Виконання арифметичного зсуву вправо

На рис.2.16 приведений приклад виконання арифметичного зсуву вліво *AL1.0* (рис.2.16,а) і *AL2.11* (рис.2.16,б).

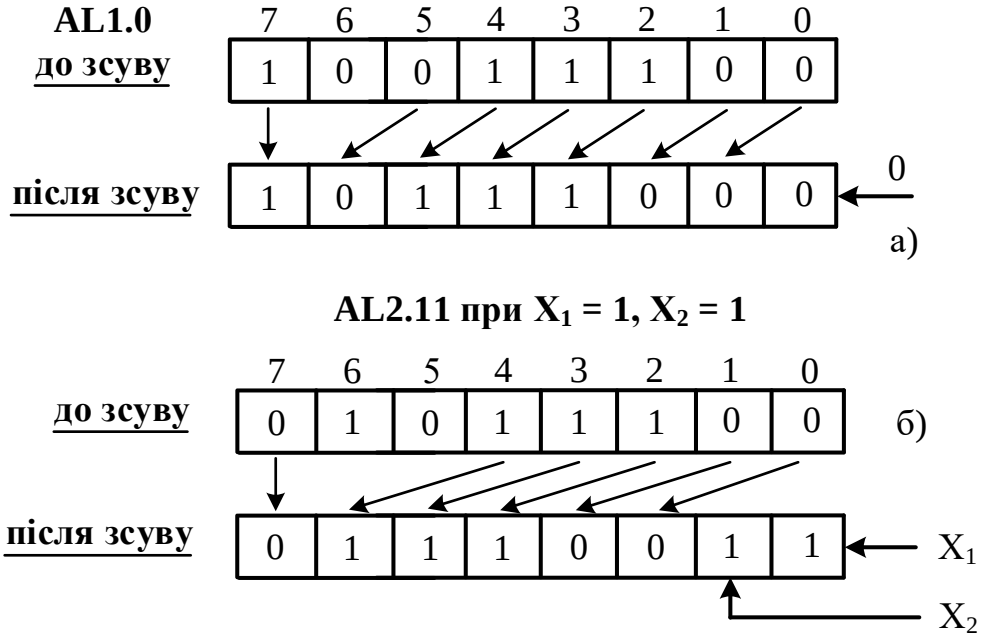


Рисунок 2.16 – Виконання арифметичного зсуву вліво

Розглянемо виконання арифметичного зсуву вліво в процесорах сімейства Pentium. Для виконання цього зсуву використовується команда процесора *SAL* (***Shift Arithmetic Left***), причому ця команда може зсувати інформацію на *k* біт (значення *k* змінюється від 1 до 31 та задається у другому операнді команди). Фактично процесор виконує команду *SAL k* разів, кожного разу виконуючи арифметичний зсув на один біт, тому далі розглянемо виконання цієї команди при здійснені зсуву на один біт.

Під час виконання цієї команди виконується зсув вліво вмісту першого операнду цієї команди, причому в молодший біт цього операнду записується завжди нуль.

На відміну від опису арифметичного зсуву вліво, розглянутого вище, при виконанні команди *SAL*, в знаковий розряд першого операнду записується старший розряд числової частини цього операнду, а сам знак записується в ознаку переносу *CF*. Таким чином, знаковий розряд при виконанні команди *SAL* змінюється. Але факт зміни знакового розряду відображується в стані ознаки переповнення *OF* (***overflow flag***), що свідчить про порушення незмінності знакового розряду при виконанні арифметичного зсуву вліво, тобто прийняття рішення про реакцію на результат зсуву покладається на програміста.

Нижче розглянемо текст невеликої програми на мові Асемблер для більш детального пояснення виконання арифметичного зсуву вліво в процесорі (після крапки з комою курсивом приведені коментарі до команд програми):

```
.model tiny
.data
.code
org 100h
start:
xor ax,ax          ; обнуління ax
```

```
mov al,0C3h      ; запис коду C3₁₆ в al
mov ah,al        ; зберігання коду C3₁₆ в ah
sal al,1         ; арифметичний зсув al на один біт вліво
mov bl,al        ; запис результату зсуву в bl
sal al,1         ; арифметичний зсув al на один біт вліво
mov bh,al        ; запис результату зсуву в bh
sal al,1         ; арифметичний зсув al на один біт вліво
mov cl,al        ; запис результату зсуву в cl
sal al,1         ; арифметичний зсув al на один біт вліво
mov ch,al        ; запис результату зсуву в ch
sal al,1         ; арифметичний зсув al на один біт вліво
mov dl,al        ; запис результату зсуву в dl
sal al,1         ; арифметичний зсув al на один біт вліво
mov dh,al        ; запис результату зсуву в dh
sal al,1         ; арифметичний зсув al на один біт вліво
mov ax,4c00h     ; підготовка до завершення програми
int 21h          ; завершення програми
end start
```

Програма завантажує в регістри ***ah*** і ***al*** знакове число $C3_{16} = 1.1000011_2 = -61_{10}$, де нижній індекс означає систему числення (в числі, представленому в двійковій системі числення, знак відділяється від цифрової частини крапкою). Після цього виконується 7 арифметичних зсувів регістра ***al*** вліво на один розряд за допомогою команди ***SAL al,1***. Результати зсувів послідовно записуються в регістри ***bl***, ***bh***, ***cl***, ***ch***, ***dl***, ***dh***, а результат останнього зсуву залишається в ***al***. В регістрі ***ah*** впродовж всього виконання програми зберігається початково задане число.

Результати виконання програми в середовищі ***TurboDebugger*** приведені на рис.2.17, на якому використовуються такі позначення:

ВПК – вказівник поточної команди, що виконується (***instruction pointer***);

CF – ознака переносу зі знакового розряду;

OF – ознака переповнення;

ah, ***al*** – старша і молодша частини регістра ***ax***.

На рис.2.17 можна побачити вміст регістрів загального призначення (РЗП) ***ax, bx, cx, dx***, показаний в шістнадцятковій системі числення (перші дві шістнадцяткові цифри відповідають значенню старшого байту цих регістрів ***ah, bh, ch, dh***, а дві молодші цифри – молодшому байту цих регістрів ***al, bl, cl, dl***). Крім того, на рисунку 2.17 показані також значення ознак стану процесора після закінчення виконання останнього зсуву (літера «*c*» відповідає стану ознаки переносу *CF*, а літера «*o*» – стану ознаки переповнення *OF*). Вказівник поточної команди ВПК вказує на те, що виконання програми було зупинено після останнього зсуву.

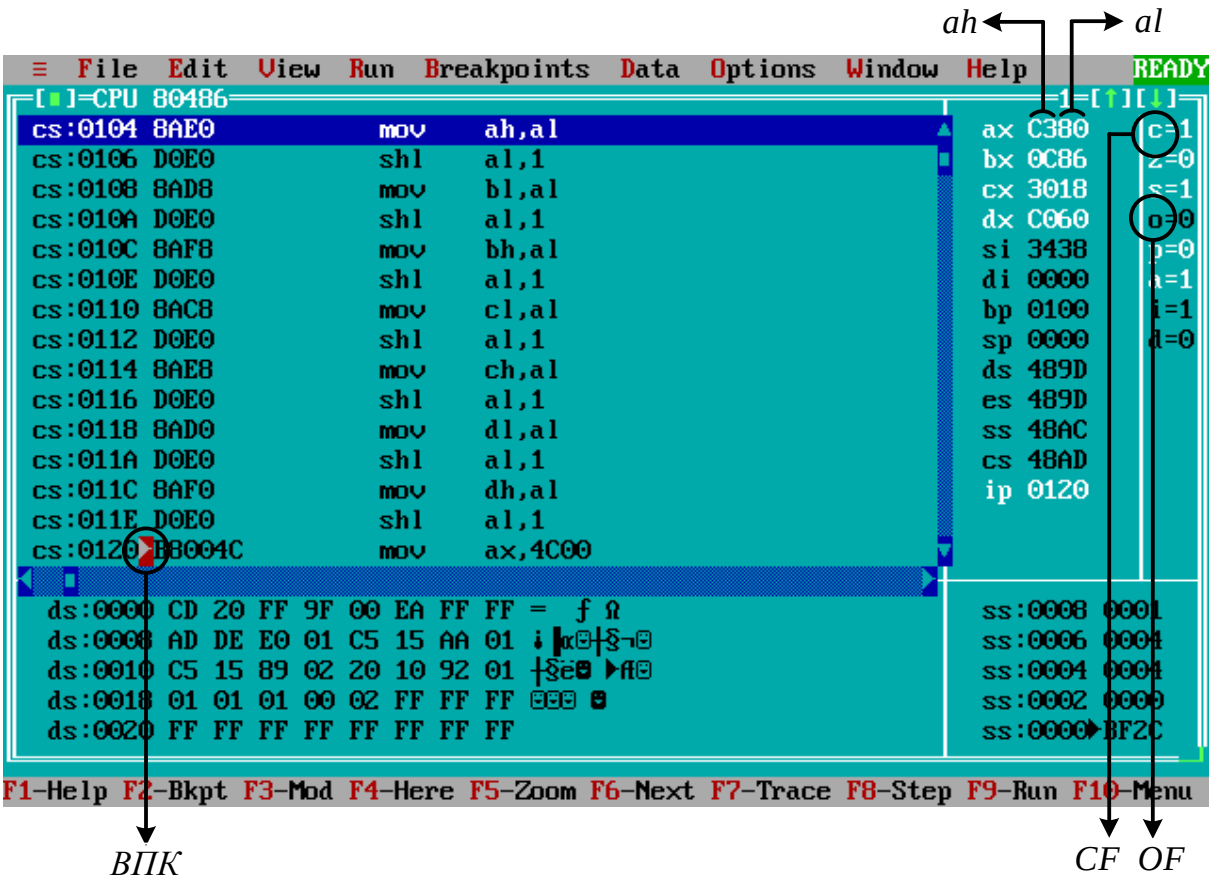


Рисунок 2.17 – Результати виконання програми ілюстрації виконання арифметичного зсуву вліво в середовищі ***TurboDebugger***

В таблиці 2.7 приведені результати покрокового виконання команди *SAL* в процесорі. В стовпчику 1 вказується адреса команди (відповідно до рис.2.17), яка буде виконана на поточному кроці виконання програми. В стовпчику 2 вказана назва регістра загального призначення РЗП, в який буде записаний результат останнього поточного зсуву. В стовпчиках 3,4 приведений вміст ознак переповнення *OF* і переносу *CF*. В стовпцях 5-8 представлений результат виконання поточного зсуву в різних системах числення (стовпчик 5 – шістнадцяткова, стовпчик 8 – десяткова, а в стовпчиках 6,7 дані представлені в двійковій системі числення в доповняльному і прямому кодах відповідно).

З таблиці 2.7 можна побачити, що при виконанні зсуву в *CF* записується стан знакового розряду попереднього вмісту регістра, а одиничне значення ознаки *OF* формується при нерівності вмісту знакових розрядів до і після виконання зсуву, тобто в даному випадку $OF = CF \oplus sign$, де *sign* – значення знакового розряду після зсуву.

Таблиця 2.7 – Таблиця з ілюстрацією виконання команди *SAL*

Адреса команди	РЗП	OF	CF	Вміст РЗП			
				16	2(ДК)	2(ПК)	10
1	2	3	4	5	6	7	8
cs:104	*ah*			*C3*	*1.100 0011*	*1.011 1101*	*-61*
cs:108	*bl*	*0*	*1*	*86*	*1.000 0110*	*1.111 1010*	*-122*
cs:10C	*bh*	*1*	*1*	*0C*	*0.000 1100*	*0.000 1100*	*12*
cs:110	*cl*	*0*	*0*	*18*	*0.001 1000*	*0.001 1000*	*24*
cs:114	*ch*	*0*	*0*	*30*	*0.011 0000*	*0.011 0000*	*48*
cs:118	*dl*	*0*	*0*	*60*	*0.110 0000*	*0.110 0000*	*96*
cs:11C	*dh*	*1*	*0*	*C0*	*1.100 0000*	*1.100 0000*	*-64*
cs:11E	*al*	*0*	*1*	*80*	*1.000 0000*	*1.000 0000*	*-128*

Зверніть увагу, що програма ***TurboDebugger*** при дизасемблюванні виконавчого файлу використовує замість мнемоніки команди *SAL* мнемоніку *SHL* (***Shift Logical Left***). Це пов’язано з тим, що виконання цих команд

практично однаково, але транслятор в об’єктному файлі (*.*obj*) підставляє код команди *SHL* замість *SAL* (рис.2.17).

Якщо значення другого операнду команди *SAL* більше одиниці, тобто виконується зсув більше, ніж на один розряд, то значення ознаки *OF* не визначено і рішення про правильність виконання зсуву покладається на програміста.

З таблиці 2.7 також випливає, що при використанні команди *SAL* для зсуву на 1 розряд відбувається збільшення вмісту операнду з будь-яким знаком вдвічі за умови нульового значення *OF*. При *OF = 1* результат такої арифметичної операції є некоректним (в таблиці 2.7 такі випадки помічені сірим кольором).

Розглянемо також виконання арифметичного зсуву вправо в процесорі (команда *SAR*). Для цього будемо використовувати текст програми на Асемблері, приведений вище, але замість команди *SAL* будемо використовувати команду *SAR* (***Shift Arithmetic Right***).

Як вже було відзначене раніше за цієї команди відбувається арифметичний зсув вправо, але спадаючий при зсуві молодший розряд не втрачається, а записується в ознаку *CF*. Результати роботи цієї програми в середовищі ***TurboDebugger*** приведені на рис.2.18.

В таблиці 2.8 приведені результати покрокового виконання команди *SAR* в процесорі.

З цієї таблиці випливає, що, арифметичний зсув вправо на один розряд забезпечує виконання операції ділення на два з округленням результату до найменшого числа.

Регістри, які виконують зсуви як вліво, так і вправо, називаються реверсивними.

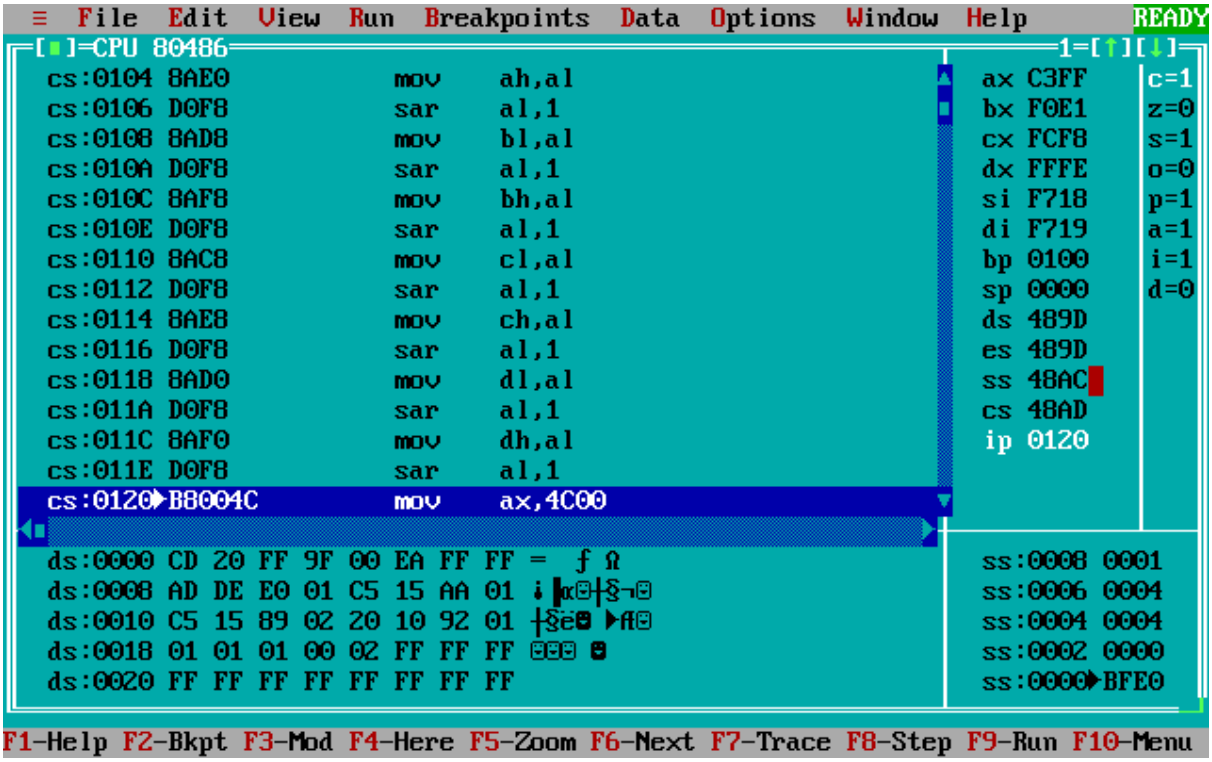


Рисунок 2.18 – Результати виконання програми ілюстрації виконання арифметичного зсуву вправо в середовищі ***TurboDebugger***

Таблиця 2.8 – Таблиця з ілюстрацією виконання команди *SAR*

Адреса команди	РЗП	OF	CF	Вміст РЗП			
				16	2(ДК)	2(ПК)	10
1	2	3	4	5	6	7	8
cs:104	*ah*			*C3*	*1.100 0011*	*1.011 1101*	*-61*
cs:108	*bl*	*0*	*1*	*E1*	*1.110 0001*	*1.001 1111*	*-31*
cs:10C	*bh*	*0*	*1*	*0C*	*1.111 0000*	*1.001 0000*	*-16*
cs:110	*cl*	*0*	*0*	*F8*	*1.111 1000*	*1.000 1000*	*-8*
cs:114	*ch*	*0*	*0*	*FC*	*1.111 1100*	*1.000 0100*	*-4*
cs:118	*dl*	*0*	*0*	*FE*	*1.111 1110*	*1.000 0010*	*-2*
cs:11C	*dh*	*0*	*0*	*FF*	*1.111 1111*	*1.000 0001*	*-1*
cs:11E	*al*	*0*	*1*	*FF*	*1.111 1111*	*1.000 0001*	*-1*

2.1.1.5 Приклади синтезу регістрів з керованою синхронізацією

Спочатку виконаємо синтез регістра, в якому схема комутатора буде реалізована класичним методом.

Приклад 2.1. На базі *D*-тригера виконати синтез 4-бітного регістра з керованою синхронізацією в будь-якому базисі логічних елементів. Регістр виконує такі мікрооперації: *uA*: *Rg* := *A*; *uB*: *Rg* := *B*. Структурна схема регістра приведена на рис.2.19.

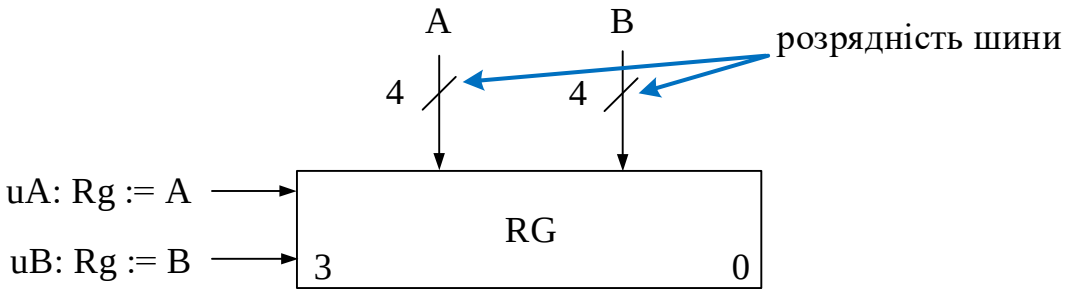


Рисунок 2.19 – Структурна схема регістра (приклад 2.1)

Розв'язок

Синтез регістра будемо виконувати відповідно до порядку, приведеному вище в підрозділі 2.1.1.

На першому кроці синтезу проведемо аналіз функціонування регістра під час виконання заданих мікрооперацій.

З умови завдання можна побачити, що регістр виконує завантаження інформації з шин *A* і *B*, тобто відбувається приймання інформації з двох напрямів. Зі структурної схеми випливає, що розрядність вхідних шин *A* і *B* (на структурній схемі показано стрілками) збігається з розрядністю регістра, тобто регістр виконує паралельне завантаження даних.

Таким чином, регістр виконує дві мікрооперації з приймання даних. За активного значення керуючого сигналу *uA* виконується завантаження в регістр вмісту шини *A* (*Rg* := *A*), а за активного значення керуючого сигналу *uB* – завантаження в регістр вмісту шини *B* (*Rg* := *B*). Звичайно, що за відсутності активних значень керуючих сигналів регістр знаходиться в режимі збереження інформації.

Далі виконаємо синтез комутатора регістра, що розглядається в підрозділі 2.1.1.1 (крок 2 процедури синтезу). В цьому прикладі проілюструємо виконання синтезу комутатора класичним методом, тобто за

допомогою таблиці істинності (табл.2.9). В зв'язку з тим, що регістр виконує тільки мікрооперації паралельного завантаження даних, то синтез можна проводити для *i*-того розряду комутатора $КОМ_i$.

Таким чином, таблиця істинності описує логічну функцію чотирьох змінних $КОМ_i = f(uA, uB, a_i, b_i)$, де uA, uB – відповідні керуючі сигнали; a_i, b_i – значення *i*-тих ліній вхідних шин даних *A* і *B* відповідно.

За неактивних значень керуючих сигналів uA і uB регістр буде знаходитися в стані збереження інформації за рахунок формування схемою керування неактивного значення сигналу $u_{БТ}$, який надходить на входи синхронізації базових тригерів регістра, забезпечуючи налаштування цих тригерів на режим збереження, тобто значення вихідних сигналів комутатора можуть бути будь-якими, що позначено в табл.2.9 на вхідних наборах 0-3 символом «*».

Одночасна поява активних значень керуючих сигналів заборонена, що також позначено на вхідних наборах 12-15 символом «*».

Табл.2.9. Таблиця істинності функціонування $КОМ_i$ (приклад 2.1)

Номери наборів	uA	uB	a_i	b_i	$КОМ_i$
	1	2	3	4	5
0	0	0	0	0	*
1	0	0	0	1	*
2	0	0	1	0	*
3	0	0	1	1	*
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	*
13	1	1	0	1	*
14	1	1	1	0	*
15	1	1	1	1	*

За активного значення сигналу *uA* (вхідні набори 8-11) на виході комутатора формується стан шини *A*, тобто сигнал на *i*-тому виході комутатора дорівнює стану сигналу на *i*-тій лінії вхідної шини *A* (стовпчик 3). Аналогічно при активному значенні сигналу *uB* (вхідні набори 4-7) на виході комутатора формується стан шини *B* (стовпчик 4).

Карти Карно для мінімізації логічної функції KOM_i приведені на рис.2.20. З карт Карно випливає, що існує кілька способів проведення операцій склеювання змінних. На карті Карно існує 4 можливих термів для забезпечення мінімізації (ці терми позначенні відповідно *T1*-*T4*). При представленні логічної функції KOM_i у вигляді диз'юнктивної нормальної форми (ДНФ) отримаємо такі логічні вирази для термів:

$$T1 = uB \cdot b_i \; ; \; T2 = uA \cdot a_i \; ; \; T3 = \overline{uA} \cdot b_i \; ; \; T4 = \overline{uB} \cdot a_i .$$

Таким чином, можливі 4 варіанти побудови логічних виразів для апаратної реалізації KOM_i:

$$KOM_i = T1 \vee T2 = T1 \vee T4 = T2 \vee T3 = T3 \vee T4 .$$

Різниця між цими виразами полягає в формуванні вихідного сигналу комутатора за двох активних значеннях керуючого сигналу, поява яких, як правило, означає наявність гонок сигналів у керуючому автоматі.

Реакція комутатора на наявність двох неактивних значень керуючих сигналів для регістра з керованою синхронізацією не є важливою, в зв'язку з тим, що тригери регістра будуть знаходитися в режимі збереження інформації за рахунок схеми керування регістром.

На рис.2.20 показані карти Карно для реалізації функції KOM_i за виразами $KOM_i = T1 \vee T2$ (рис.2.20,а) і $KOM_i = T3 \vee T4$ (рис.2.20,б).

В результаті розглянемо такі логічні вирази для KOM_i:

$$KOM_i = T1 \vee T2 = uA \cdot a_i \vee uB \cdot b_i \; ; \qquad (2.11)$$

$$KOM_i = T3 \vee T4 = \overline{uA} \cdot b_i \vee \overline{uB} \cdot a_i \; ; \qquad (2.12)$$

Відповідно до (2.11) на виході комутатора за двох активних значень керуючих сигналів формується порозрядна диз'юнкція відповідних ліній вхідних шин даних A і B: $КОМ_i = a_i \vee b_i$, що в принципі можна використовувати при проєктуванні регістра при необхідності завантаження результату порозрядної диз'юнкції вмісту вхідних шин даних.

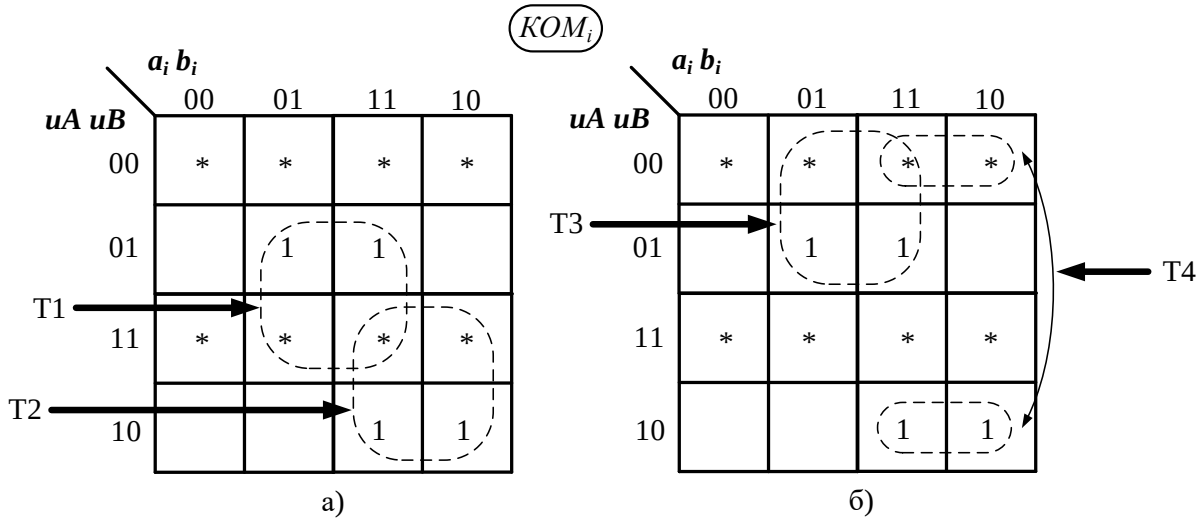


Рисунок 2.20 – Мінімізація функції вихідного стану $КОМ_i$ (приклад 2.1)

Відповідно до (2.12) за двох активних значень керуючих сигналів на кожному виході комутатора формується нульовий сигнал, що можна використовувати для виконання синхронного скиду регістра за відсутності асинхронних входів базових тригерів регістра.

Далі виконаємо синтез схеми керування регістра, що розглядається в підрозділі 2.1.1.2 (крок 3 процедури синтезу). Як вже було відзначено в підрозділі 2.1.1.2 схема керування даного регістра формує сигнал $u_{БТ} = f(uA, uB, G)$, логічний вираз для якого отримаємо з таблиці істинності, приведеної в табл.2.10. Для заповнення таблиці істинності будемо вважати, що на регістр не може одночасно надходити два активних керуючих сигнали.

Відповідно до (2.5) отримаємо: $u_{БТ} = G \cdot (uA \vee uB)$.

Виконаємо четвертий крок процедури синтезу регістра, який полягає в проведенні синтезу схеми формування функцій збудження *СФФЗ* базових тригерів (див. підрозділ 2.1.1.3).

Як вже було зазначено в підрозділі 2.1.1.3, синтез СФФЗ достатньо проводити для одного розряду базового тригера регістра. Відповідно до умови завдання необхідно проводити синтез регістра на основі *D*-тригера, тобто для синтезу *СФФЗ* будемо використовувати колонки 1-6 таблиці 2.6, в результаті чого, використовуючи карту Карно (рис.2.3), отримаємо, що $D_i = KOM_i$.

Таблиця 2.10 – Таблиця істинності формування $u_{БТ}$ (приклад 2.1)

Номери наборів	***G***	*uA*	*uB*	$\boldsymbol{u_{БТ}}$
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	*

На п’ятому кроці синтезу відповідно до умови завдання для побудови регістра використовуємо будь-який базис логічних елементів, в зв’язку з чим всі отримані на попередніх кроках логічні вирази залишимо без перетворень.

В результаті виконання синтезу всіх комбінаційних складових частин регістра побудуємо логічну схему регістра (крок 6 процедури синтезу), яка приведена на рис.2.21.

Складові комбінаційні частини регістра показанні на рис.2.21 пунктирними лініями. В зв’язку з тим, що регістр будується на базі *D*- тригерів, то в схемі відсутня *СФФЗ*, тобто виходи комутатора безпосередньо від’єднуються до входів *D*-тригерів.

На рис.2.22 приведена логічна схема для моделювання функціонування синтезованого регістра.

Комутатор регістра побудований на базі логічних елементів *U33-U35* (KOM_3), *U77-U79* (KOM_2), *U74-U76* (KOM_1), *U71-U73* (KOM_0). Схема

керування реалізована на елементах *U80*, *U65*. Генератори прямокутних імпульсів, побудовані на елементах *U81* і *U82*, імітують вхідні шини даних *A* і *B* відповідно. Інверсний сигнал ***nReset*** призначений для встановлення регістра в початковий нульовий стан.

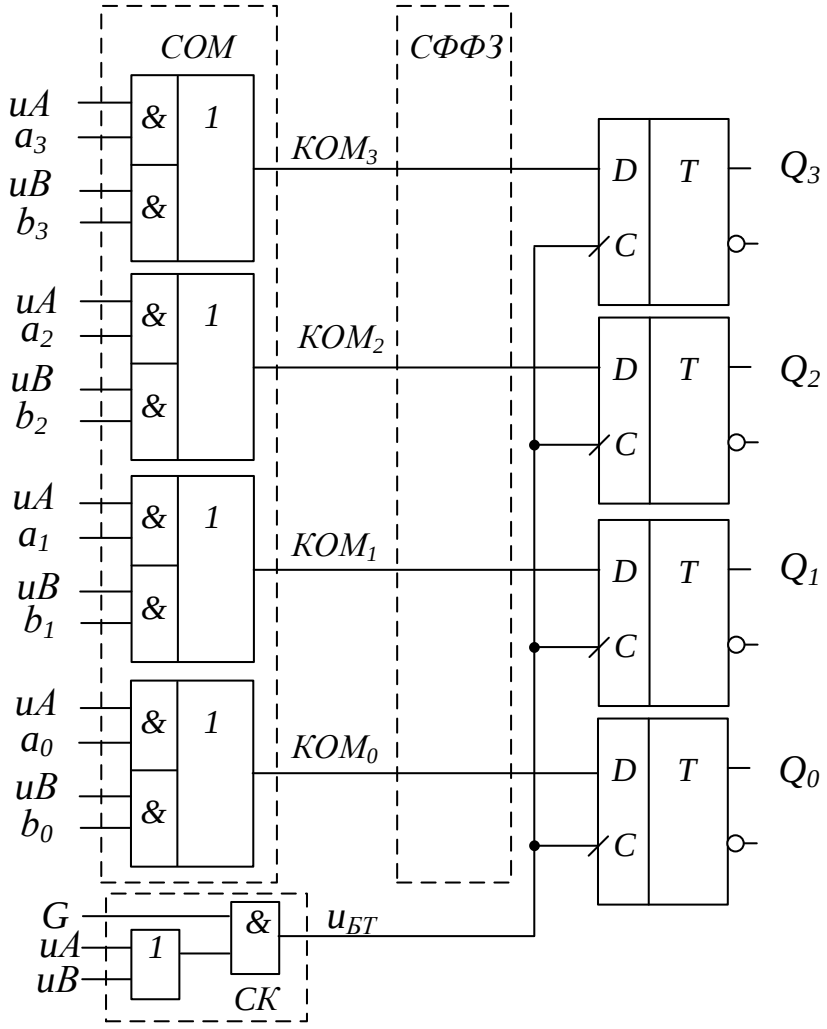


Рисунок 2.21 – Логічна схема регістра (приклад 2.1)

Результати перевірки коректності функціонування регістра (крок 7 процедури синтезу) за допомогою моделювання приведені на рис.2.23.

В моменти часу t_{A0111}, t_{A1010} відбувається завантаження інформації з шини даних *A* (відповідно кодам *0111* і *1010* на лініях цієї шини) за переднім фронтом сигналу *G*. Аналогічно за переднім фронтом сигналу *G* в моменти часу t_{B0101}, t_{B1011} відбувається завантаження інформації з шини *B* (відповідно кодам *0101* і *1011* на лініях цієї шини).

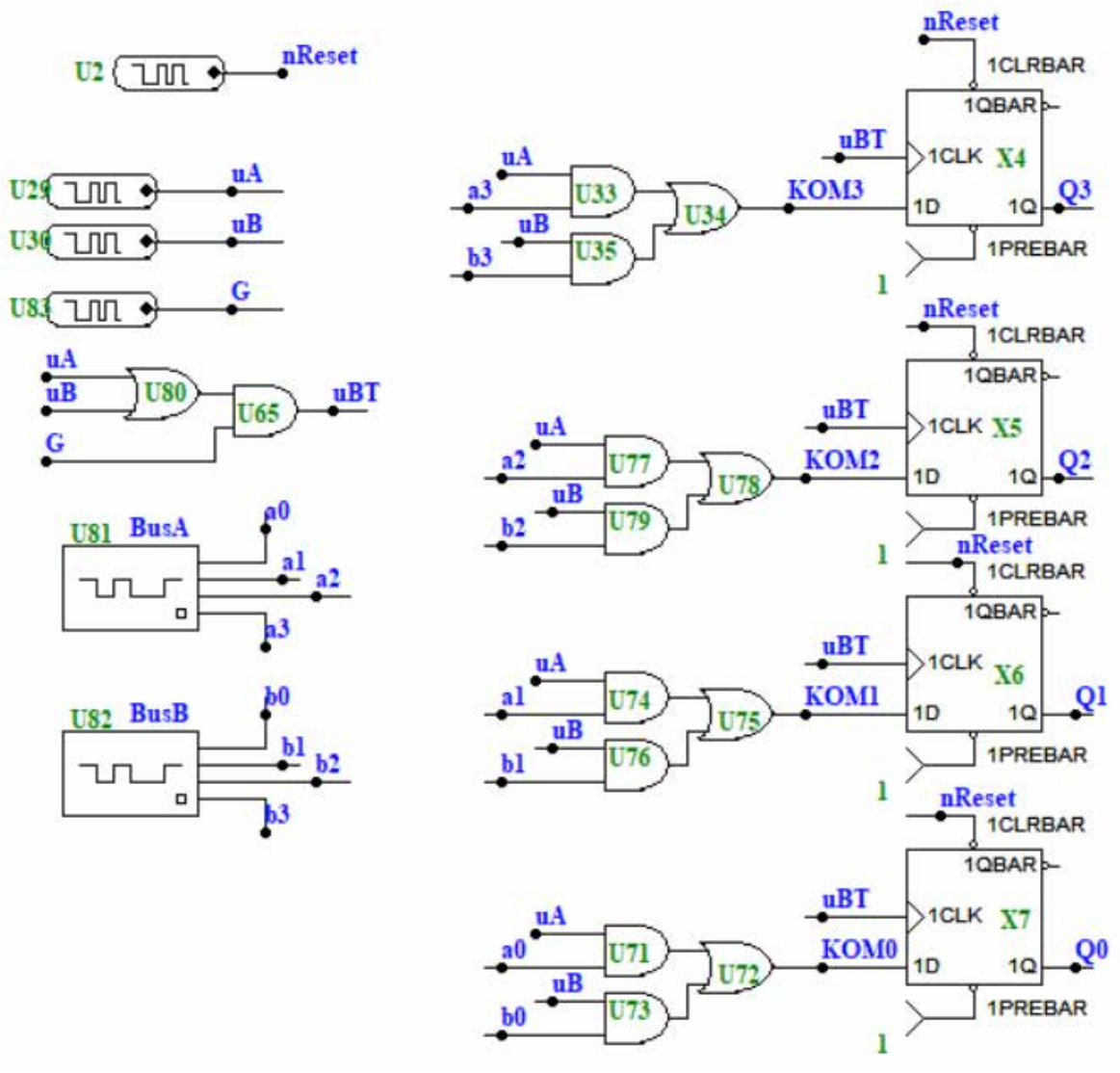


Рисунок 2.22 – Логічна схема для моделювання регістра (приклад 2.1)

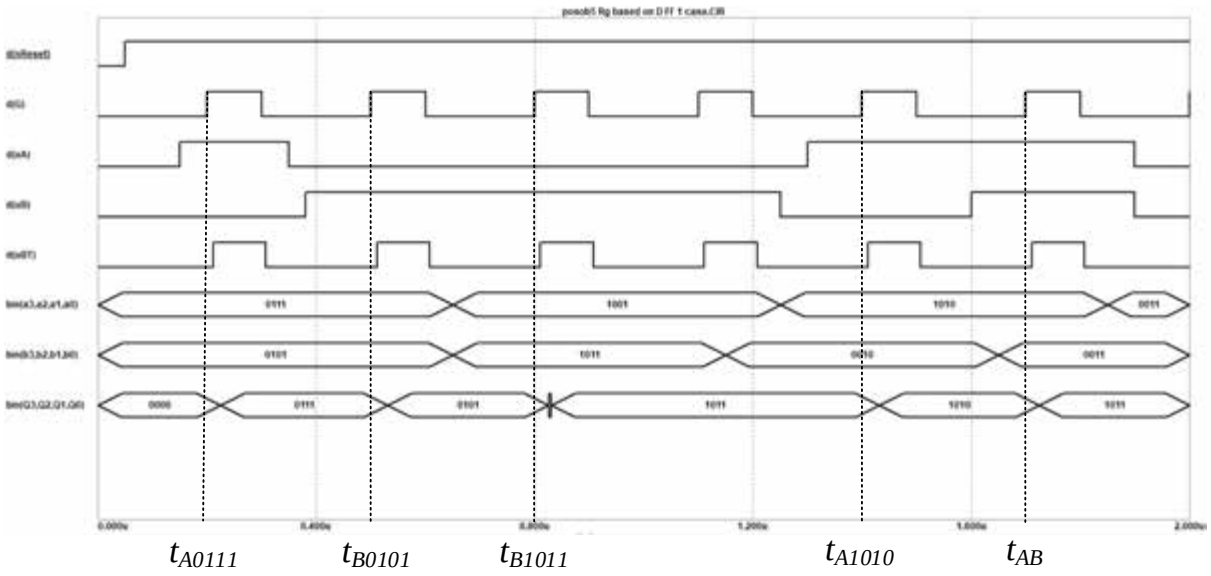


Рисунок 2.23 – Результати моделювання регістра (приклад 2.1)

В момент часу t_{AB} показана реакція регістра на одночасну активізацію керуючих сигналів *uA* і *uB*. Як вже було відзначено раніше, одночасна поява керуючих сигналів, що призначені для завантаження інформації в регістр, як правило, заборонена. Розглянемо, що відбувається в регістрі в даному випадку, тобто при $uA = uB = 1$.

Відповідно до (2.11) $KOM_i = uA \cdot a_i \vee uB \cdot b_i = a_i \vee b_i$. Таким чином, при реалізації комутатора відповідно до (2.11), за рахунок одночасної активізації сигналів *uA* і *uB* регістр може виконувати додаткову мікрооперацію порозрядної диз'юнкції вмісту шин *A* і *B* без використання додаткових апаратних витрат (на рис.2.23 можна побачити, що за переднім фронтом сигналу *G* відбувається завантаження результату диз'юнкції вмісту шини $A = 0111$ і $B = 1010$ до регістра: $0111 \vee 1010 = 1011$).

Далі розглянемо реалізацію комутатора відповідно до (2.12). Схема для моделювання і результати перевірки функціонування регістра приведені відповідно на рис.2.24 і 2.25.

На рис.2.25 в момент часу t_{AB} реакція регістра на одночасну активізацію керуючих сигналів *uA* і *uB* при реалізації комутатора за виразом (2.12) відповідає виконанню мікрооперації обнуління регістра. Відповідно до (2.12) $KOM_i = \overline{uA} \cdot b_i \vee \overline{uB} \cdot a_i$, тобто при $uA = uB = 1$ на виході комутатора формуються нульові сигнали ($KOM_i = 0$), що можна побачити на рис.2.25. При активізації тільки одного з керуючих сигналів регістр, схема якого приведена на рис.2.24, працює аналогічно регістру, показаному на рис.2.22.

Далі виконаємо визначення динамічних параметрів регістра (крок 8 процедури синтезу).

Для побудови регістра використовується *D*-тригер *SN7474* (*K155TM2*), який є тригером з динамічним керуванням. Відповідно до (2.7) і (2.9), враховуючи відсутність комутатора, можна отримати, що $T_{Gmin} = t_{uG\ min} + t_{mp} = t_{COM} + t_{ny} + t_{mp}$.

На рис.2.26 приведені результати моделювання для визначення динамічних параметрів регістра. На часовій діаграмі показані величини затримок спрацьовування комутатора t_{COM}, базових тригерів t_{mp} при переключенні цих тригерів з 0 в 1 і навпаки.

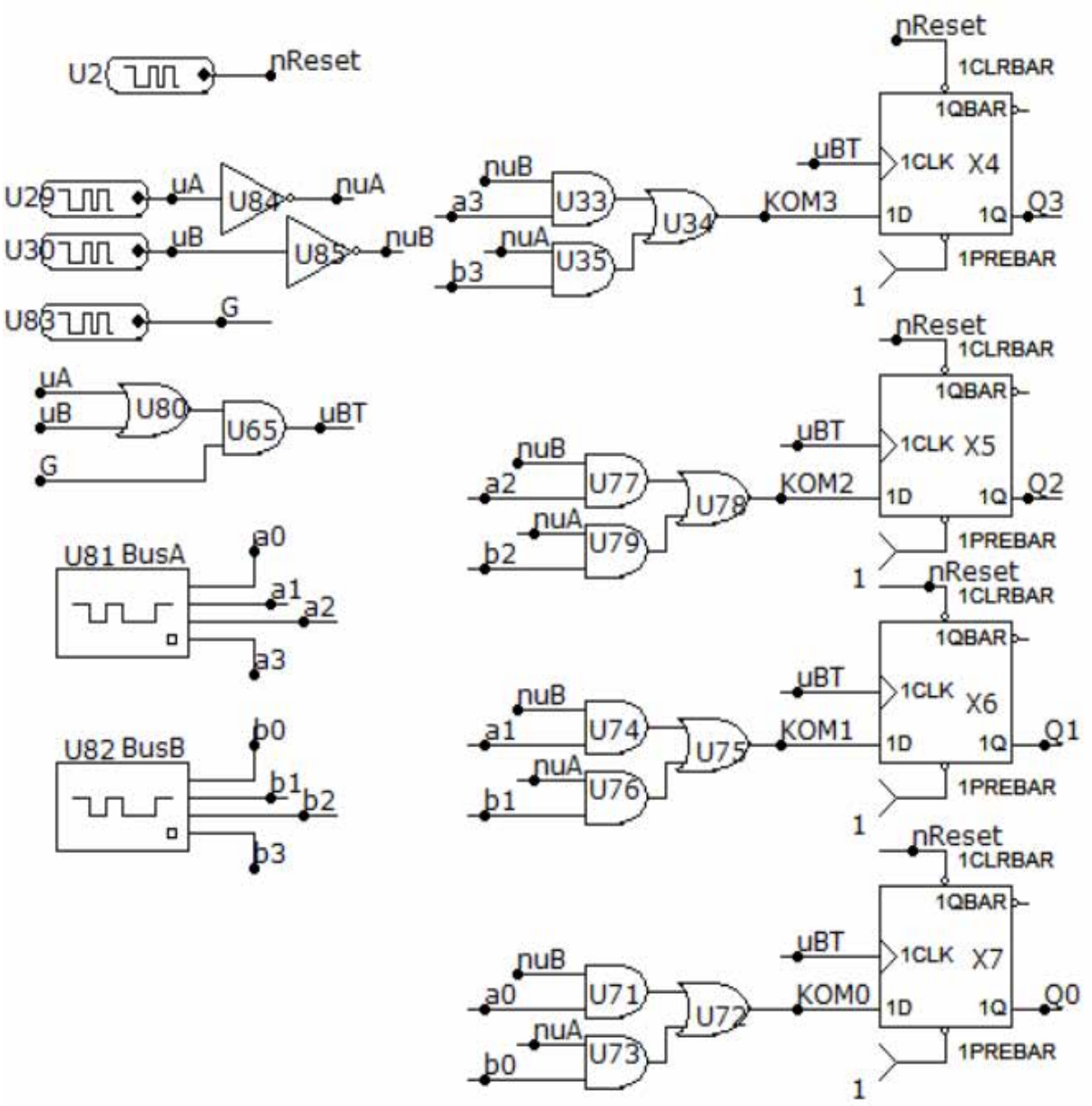


Рисунок 2.24 – Логічна схема для моделювання регістра (приклад 2.1) при реалізації комутатора за виразом (2.12)

Далі розглянемо приклад синтезу регістра зсуву.

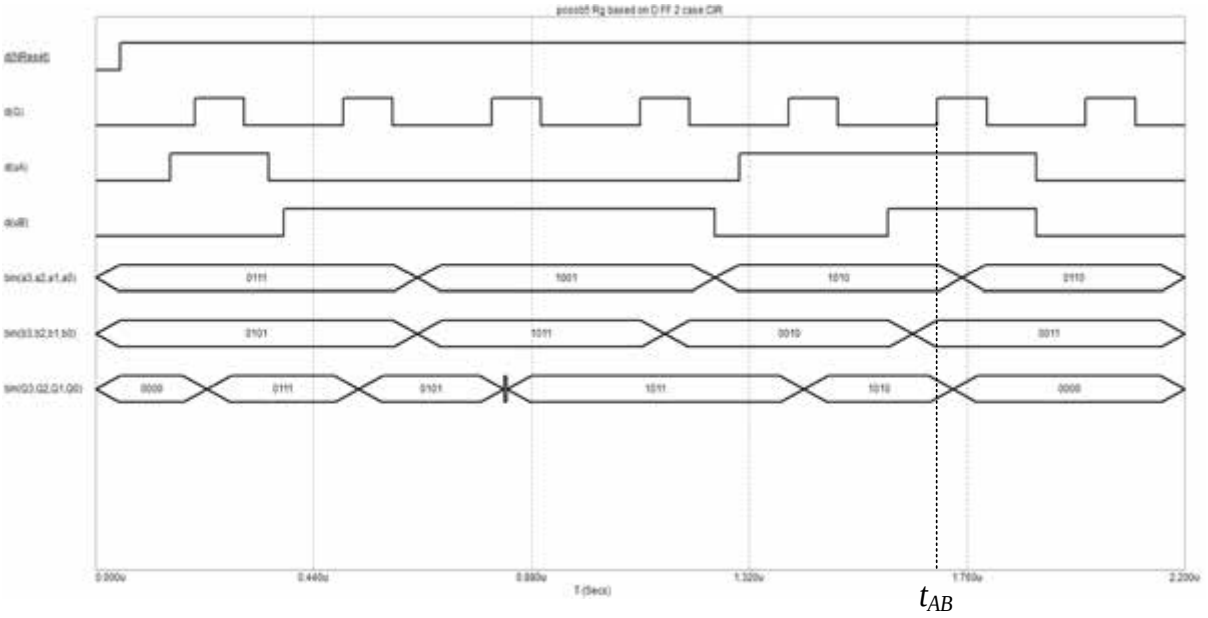


Рисунок 2.25 – Результати моделювання регістра (приклад 2.1) при реалізації комутатора за виразом (2.12)

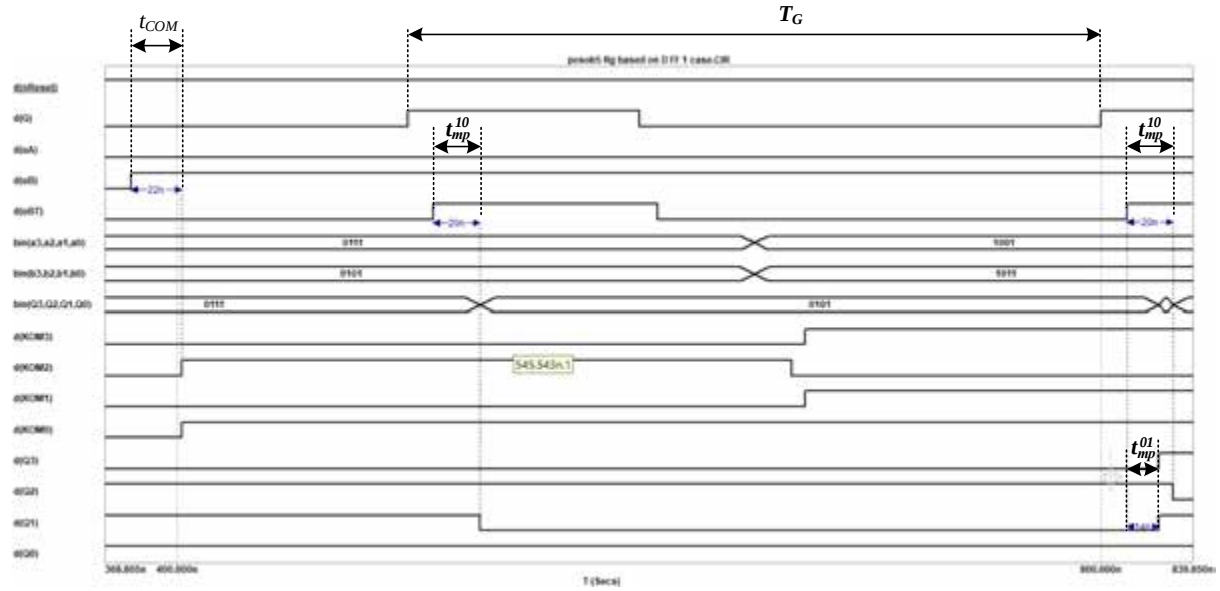


Рисунок 2.26 – Визначення динамічних параметрів регістра (приклад 2.1)

Приклад 2.2. На базі *D*-тригера виконати синтез 4-бітного реверсивного регістра з керованою синхронізацією в будь-якому базисі логічних елементів. Регістр виконує такі мікрооперації: uA: $Rg := A$;

uR: *Rg* := *AR1*(*Rg*) ; *uL*: *Rg* := *CL1*(*Rg*). Структурна схема регістра приведена на рис.2.27.

Розв'язок

Синтез регістра також будемо виконувати відповідно до порядку, приведеному вище в підрозділі 2.1.1.

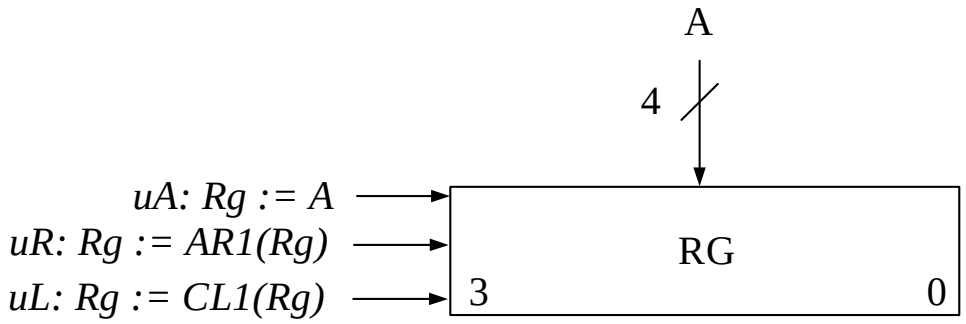


Рисунок 2.27 – Структурна схема регістра (приклад 2.2)

На першому кроці синтезу проведемо аналіз функціонування регістра під час виконання заданих мікрооперацій.

З умови завдання випливає, що регістр виконує мікрооперацію паралельного завантаження інформації за сигналом *uA* (*Rg* := *A*), арифметичний зсув вмісту регістра на один розряд вправо за сигналом *uR* (*Rg* := *AR1*(*Rg*)), а також циклічний зсув вмісту регістра вліво за сигналом *uL* (*Rg* := *CL1*(*Rg*)). Звичайно, що за відсутності активних значень керуючих сигналів регістр перебуває в режимі збереження інформації.

На відміну від попереднього прикладу виконаємо синтез комутатора регістра за допомогою таблиці мікрооперацій, що розглядається в підрозділі 2.1.1.1 (крок 2 процедури синтезу).

Відповідно до умови завдання до складу таблиці мікрооперацій входять 3 рядки згідно з кількістю мікрооперацій, що виконує регістр, та 4 стовпця згідно з розрядністю регістра.

Нагадаємо, що кожний стовпчик таблиці мікрооперацій відображує стан відповідної вихідної лінії комутатора в залежності від активного значення керуючого сигналу *uA*, *uR* або *uL*.

Для заданого регістра таблиця мікрооперацій приведена в табл.2.11.

Таблиця 2.11 – Таблиця МО для синтезу комутатора (приклад 2.2)

МО	KOM_3	KOM_2	KOM_1	KOM_0
1	2	3	4	5
uA	a_3	a_2	a_1	a_0
uR	Q_3	Q_3	Q_2	Q_1
uL	Q_2	Q_1	Q_0	Q_3

При надходженні активного значення керуючого сигналу *uA* на виходах комутатора KOM_3 - KOM_0 з'являється вміст ліній (a_3 - a_0) вхідної шини даних *A*, що відзначено у відповідних клітинах таблиці МО (стовпчики 2-4) для рядку, що відповідає сигналу *uA*.

Під час активізації керуючого сигналу *uR* регістр здійснює арифметичний зсув свого вмісту вправо на 1 біт. Нагадаємо, що при виконанні арифметичного зсуву вправо значення знакового (старшого розряду Q_3) не змінюється, але в зсуві цей розряд приймає участь (приклад виконання арифметичного зсуву вправо приведено на рис.2.15). В зв'язку з цим на виході KOM_3 (стовпчик 2) для сигналу *uR* формується значення знакового розряду Q_3, яке також зсувається в сусідній розряд і з'являється на виході KOM_2 (стовпчик 3). Аналогічно на виходах KOM_1-KOM_0 з'являються стани виходів регістра Q_2 - Q_1 (стовпчики 4-5).

По сигналу *uL* регістр виконує циклічний зсув вліво на 1 розряд (приклад такого зсуву приведений на рис.2.10). В стовпчиках 2-4 для сигналу *uL* виконується зсув інформації в поточний розряд з більш молодшого сусіднього розряду ($KOM_i = Q_{i-1}$, $i = 1,2,3$), а на молодшому виході комутатора формується стан старшого розряду регістра Q_2.

Після заповнення клітин таблиці мікрооперацій можна отримати логічні вирази для кожного виходу комутатора:

$$
\begin{aligned}
КОМ_3 &= uA \cdot a_3 \vee uR \cdot Q_3 \vee \ uL \cdot Q_2 ; \\
КОМ_2 &= uA \cdot a_2 \vee uR \cdot Q_3 \vee \ uL \cdot Q_1 ; \\
КОМ_1 &= uA \cdot a_1 \vee uR \cdot Q_2 \vee \ uL \cdot Q_0 ; \\
КОМ_0 &= uA \cdot a_0 \vee uR \cdot Q_1 \vee \ uL \cdot Q_3 ;
\end{aligned}
\qquad (2.13)
$$

Вирази для $КОМ_2$ та $КОМ_1$ можна записати в загальному вигляді:

$$КОМ_i = uA \cdot a_i \vee uR \cdot Q_{i+1} \vee \ uL \cdot Q_{i-1}, \ i = 1,2;$$

Таким чином, синтез комутатора за допомогою таблиці мікрооперацій є більш простішим і наочним по зрівнянню з класичним методом синтезу та не має обмежень по кількості логічних змінних, що використовуються при синтезі регістра.

Проєктування схеми керування виконується за аналогією з синтезом такої схеми, розглянутої в прикладі 2.1 та підрозділі 2.1.1.2 (крок 3 процедури синтезу). Схема керування даного регістра формує сигнал $u_{БТ} = f(uA, uR, uL, G)$, логічний вираз для якого можна отримати з таблиці істинності, що приведена в табл.2.12.

Відповідно до (2.5) отримаємо: $u_{БТ} = G \cdot (uA \vee uR \vee uL)$.

Синтез схеми формування функцій збудження *СФФЗ* (четвертий крок процедури синтезу) проводиться таким же чином, як в прикладі 2.1 та підрозділі 2.1.1.3. Використовуючи табл.2.6, отримаємо, що $D_i = КОМ_i$.

Далі на п’ятому кроці синтезу відповідно до умови завдання для побудови регістра використовуємо будь-який базис логічних елементів, в зв’язку з чим всі отримані на попередніх кроках логічні вирази залишимо без перетворень.

Таким чином, виконано синтез всіх комбінаційних складових частин регістра, що дає змогу побудувати логічну схему заданого регістра (крок 6 процедури синтезу), яка приведена на рис.2.28.

Як і в прикладі 2.1. складові комбінаційні частини регістра показанні на рис.2.28 пунктирними лініями.

Таблиця 2.12 – Таблиця істинності формування $u_{БТ}$ (приклад 2.2)

Номери наборів	***G***	*uA*	*uR*	*uL*	$u_{БТ}$
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	*
12	1	1	0	0	1
13	1	1	0	1	*
14	1	1	1	0	*
15	1	1	1	1	*

На рис.2.29 приведена логічна схема для моделювання функціонування синтезованого регістра, а на рис.2.30 – результати моделювання функціонування синтезованого регістра.

Розглянемо детально результати моделювання (рис.2.30).

В момент часу t_{uA1}, де цифра в кінці позначення керуючого сигналу означає номер цього сигналу на часовій діаграмі, задається активне значення сигналу *G* при активному *uA*, тобто регістр налаштовується на завантаження інформації з шини *A*. На часовій діаграмі можна побачити, що після спрацьовування регістра в цей момент часу на виході регістра з'являється стан *1000* (рядок діаграми $bin(Q_3, Q_2, Q_1, Q_0)$), що відповідає стану шини (рядок $bin(a_3, a_2, a_1, a_0)$).

Далі в моменти часу t_{uR1}, t_{uR2}, t_{uR3} задаються три керуючі сигнали *G* при активному *uR*, кожний з яких налаштовує регістр на виконання мікрооперації зсуву $Rg := AR1(Rg)$, тобто виконується арифметичний зсув вправо вмісту регістра (*1000*), яке було встановлено сигналом *uA* раніше в момент t_{uA1}.

На часовій діаграмі можна побачити, що після спрацьовування регістра в ці моменти часу на виході регістра код *1000* тричі зсувається з врахуванням значення старшого розряду і на виході поступово з'являються коди *1100*, *1110*, *1111* (див. рядок *bin*(Q_3, Q_2, Q_1, Q_0)).

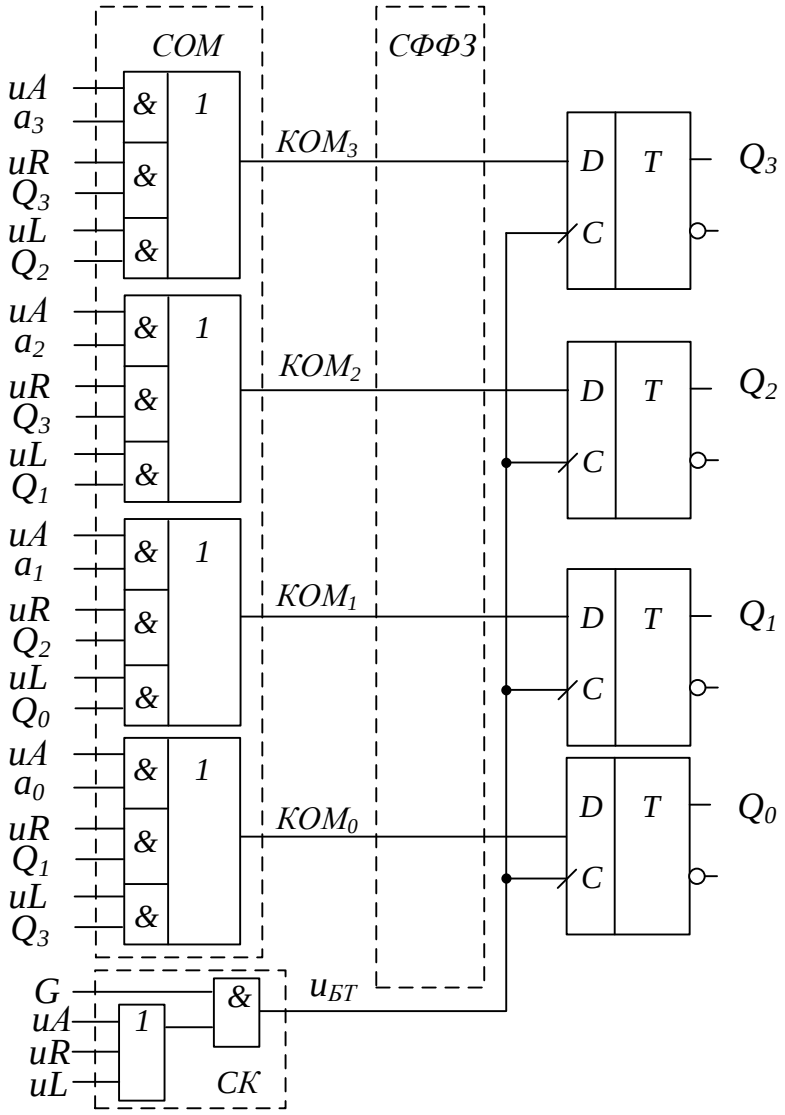


Рисунок 2.28 – Логічна схема регістра (приклад 2.2)

Після цього задається другий сигнал t_{uA2} для завантаження інформації з шини *A*. В цей момент часу вміст шини *A* дорівнює коду *1011*, який по сигналу *G* формується на виході регістра.

Далі в моменти часу t_{uL1}, t_{uL2}, t_{uL3} задаються три керуючі сигнали *G* при активному *uL*, кожний з яких налаштовує регістр на виконання мікрооперації зсуву *Rg* := *CL1*(*Rg*), тобто виконується циклічний зсув вліво вмісту регістра

(*1011*), яке було встановлено сигналом *uA* раніше в момент t_{uA2}. На часовій діаграмі можна побачити, що після спрацьовування регістра в ці моменти часу на виході регістра код *1011* тричі циклічно зсувається вліво і на виході послідовно з'являються коди *0111*, *1110*, *1101* (див. рядок *bin*(Q_3, Q_2, Q_1, Q_0)).

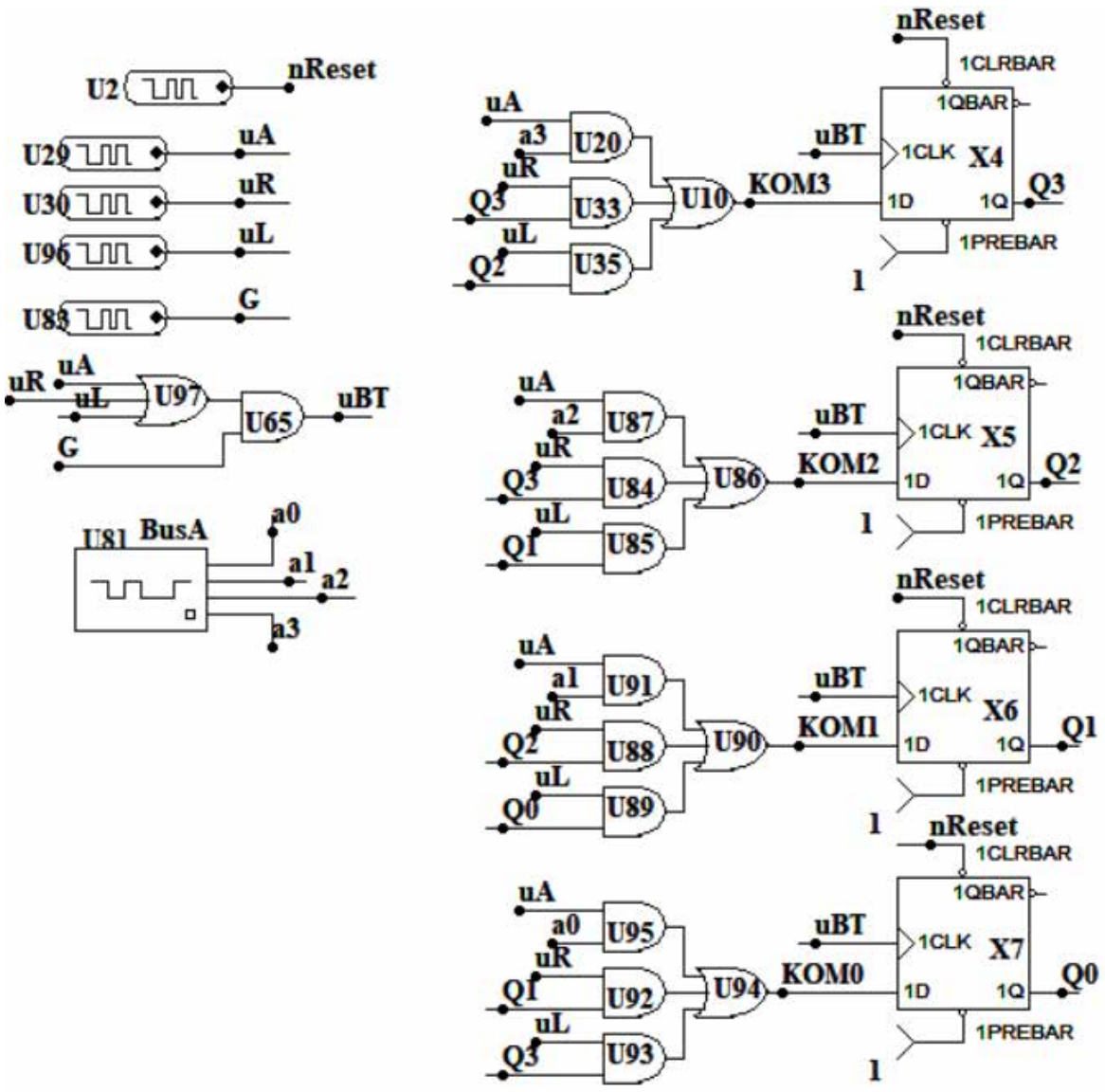


Рисунок 2.29 – Логічна схема для моделювання регістра (приклад 2.2)

На рис.2.31 приведені результати моделювання для визначення динамічних параметрів регістра. За аналогією з попереднім прикладом на часовій діаграмі показані величини затримок спрацьовування комутатора і базових тригерів.

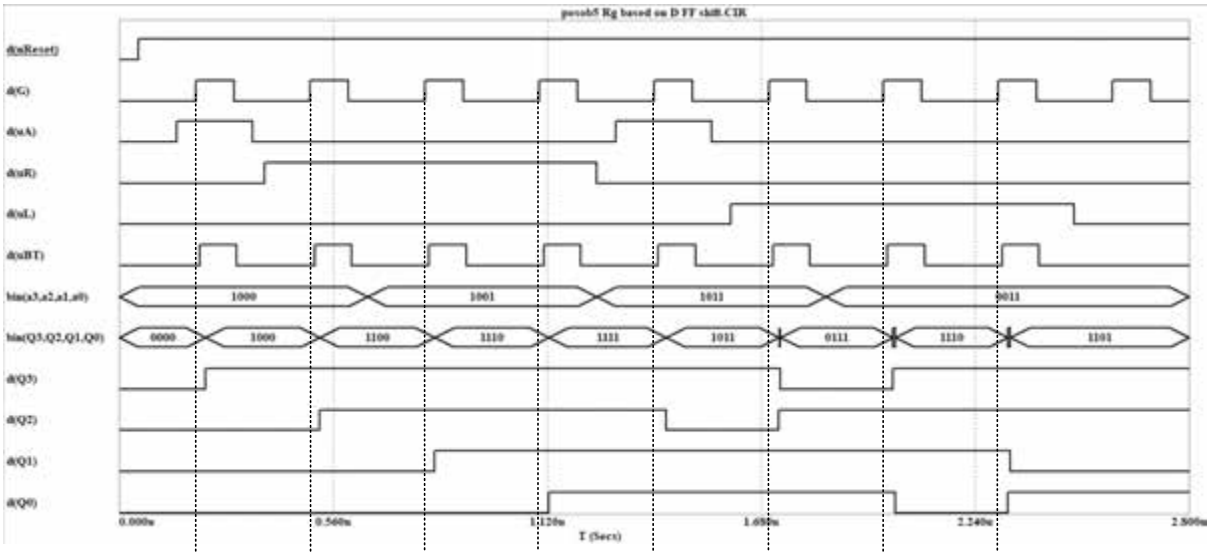


Рисунок 2.30 – Результати моделювання регістра (приклад 2.2)

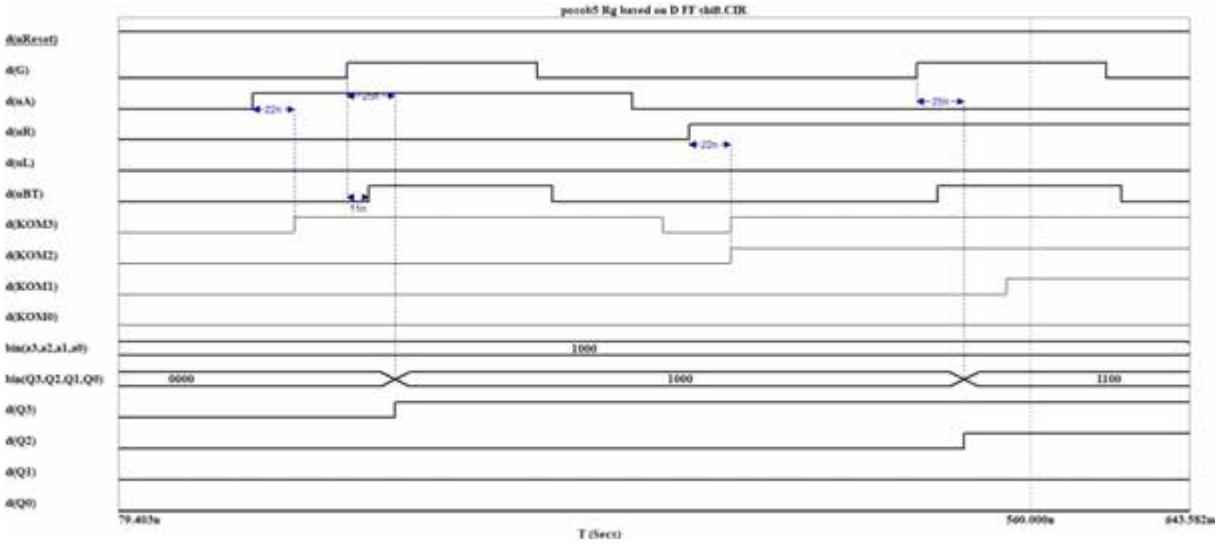


Рисунок 2.31 – Визначення динамічних параметрів регістра (приклад 2.2)

Контрольні завдання та запитання

1. Який цифровий пристрій називається регістром?
2. В чому полягає різниці між визначенням регістрів і тригерів?
3. Назвіть типи регістрів відповідно до способу синхронізації.
4. В чому полягає призначення регістрів?
5. Які функції виконує регістр?

6. Що називається мікрооперацією?
7. Які мікрооперації виконує регістр?
8. Скільки тригерів в складі *n*-розрядного регістра?
9. Яка нумерація розрядів регістра використовується в сучасних комп'ютерах?
10. Яке співвідношення існує між номером розряду в регістрі і його двійковою вагою?
11. Чи можуть в складі регістра використовуватися тригери різних типів за функціональним призначенням?
12. Чи є тригер комбінаційним пристроєм? Обґрунтуйте відповідь.
13. Чи є тригер пристроєм з пам'яттю? Обґрунтуйте відповідь.
14. Які цифрові вузли використовуються в якості пам'яті регістра?
15. В чому полякає ознака того, що регістр є пристроєм з пам'яттю?
16. За яким виразом можна описати функціонування регістра з точки зору теорії автоматів?
17. Як здійснюється доступ до виходів регістра?
18. Назвіть способи, за допомогою яких може бути організоване завантаження інформації в регістр.
19. В чому полягає суть паралельного завантаження інформації в регістр?
20. В чому полягає суть послідовного завантаження інформації в регістр?
21. В чому полягає суть послідовно-паралельного завантаження інформації в регістр?
22. В яких пристроях комп'ютерних систем використовується послідовний спосіб обміну інформацією?
23. Яке співвідношення між розрядністю регістра і кількістю розрядів вхідної шини при використанні паралельного способу завантаження даних?

24. Яке співвідношення між розрядністю регістра і кількістю розрядів вхідної шини при використанні послідовного способу завантаження даних?
25. Яке співвідношення між розрядністю регістра і кількістю розрядів вхідної шини при використанні послідовно-паралельного способу завантаження даних?
26. Як розрізняються регістри за способом керування завантаженням даних?
27. Приведіть структурну схему регістра з керованою синхронізацією.
28. Які складові частини входять до складу регістра з керованою синхронізацією?
29. Поясніть призначення комутатора в складі регістра.
30. Поясніть призначення схеми формування функцій збудження в складі регістра.
31. Поясніть призначення схеми керування в складі регістра.
32. Поясніть призначення базових тригерів в складі регістра.
33. Поясніть взаємодію складових частин під час функціонування регістра.
34. Які складові вузли регістра є комбінаційними?
35. Які складові вузли регістра виконують мікрооперацію збереження інформації?
36. Які складові вузли регістра забезпечують запис даних до базового тригера?
37. Поясніть термін «функція збудження».
38. Які дії виконує комутатор в складі регістра?
39. Які дії виконує схема формування функцій збудження в складі регістра?
40. Поясніть призначення сигналу u_i?
41. Поясніть призначення сигналу In_i?
42. Поясніть призначення сигналу G?

43. Поясніть призначення сигналу $u_{БТ}$?
44. В яких випадках в схемі регістра може бути відсутнім комутатор?
45. В яких випадках в схемі регістра може бути відсутньою схема формування функцій збудження?
46. Поясніть наявність ліній зворотного зв'язку в складі регістра.
47. В яких випадках в схемі регістра може бути відсутніми лінії зворотного зв'язку з виходів базових тригерів на вхід комутатора?
48. Що називається однофазною системою синхронізації регістра?
49. Що називається двофазною системою синхронізації регістра?
50. За яких умов може використовуватися двофазна система синхронізації регістра?
51. Приведіть послідовність кроків для виконання синтезу багатофункціонального регістра з керованою синхронізацією.
52. Схеми яких складових регістра необхідно синтезувати?
53. Якими способами можна виконати синтез комутатора?
54. Для чого використовуються асинхронні входи базових тригерів у складі регістра?
55. Як виконати синтез комутатора за допомогою таблиці істинності?
56. Який розмір таблиці істинності необхідно використовувати, якщо регістр може завантажувати інформацію тільки із зовнішніх джерел?
57. Чи достатньо виконувати синтез тільки одного розряду комутатора при використанні таблиці істинності? Обґрунтуйте відповідь.
58. Які недоліки використання способу синтезу комутатора за допомогою таблиці істинності?
59. Яким обсягом обмежується таблиця істинності при виконанні «ручного» метода синтезу?
60. Як виконати синтез комутатора за допомогою таблиці мікрооперацій?
61. Як визначити кількість рядків в таблиці мікрооперацій?
62. Як визначити кількість стовпців в таблиці мікрооперацій?

63. Поясніть, як визначити вміст поточної клітини таблиці мікрооперацій?
64. Як визначити логічні вирази для кожного виходу комутатора у вигляді ДНФ?
65. Яким чином при заповненні клітин для поточного керуючого сигналу необхідно використовувати інші керуючі сигнали?
66. Поясніть вирази (2.1).
67. Які переваги використання метода синтезу регістра за допомогою таблиці мікрооперацій?
68. Яким чином виконувати синтез схеми керування регістра?
69. Для чого під час синтезу схеми керування виникає необхідність проводити декомпозицію логічної функції $u_{БТ}$?
70. Що називається суперпозицією логічних функцій?
71. Як виконувати декомпозицію логічної функції?
72. В яких випадках необхідно виконувати декомпозицію логічної функції?
73. В чому полягає розкладання Шеннона?
74. Поясніть вирази (2.3), (2.4).
75. Поясніть термін «часткова логічна функція».
76. Як буде працювати регістр з прикладу 2.1, якщо в схемі керування не використовувати сигнал G?
77. За яких умов існує можливість не використовувати сигнал G?
78. Яким способом виконується синтез схеми керування регістра?
79. Поясніть, яким чином заповнювати таблицю істинності для формування сигналу $u_{БТ}$?
80. Поясніть термін «активне значення сигналу синхронізації».
81. Які способи заповнення таблиці істинності для сигналу $u_{БТ}$ використовуються в разі надходження відразу двох або більшої кількості активних значень керуючих сигналів?
82. За яких умов логічна функцію $u_{БТ}$ можна вважати недовизначеною?

83. Поясніть вираз (2.5).
84. Як виконати декомпозицію функції $u_{БТ}$ по аргументу G?
85. До чого призводить наявність розгалужень сигналів u_i, що сходяться, в схемі регістра?
86. Поясніть термін «гонки сигналів».
87. Поясніть, яким чином виникають гонки сигналів в регістрі?
88. Як буде спрацьовувати регістр, при виконанні умови $t_{COM} > t_{CK}$, де t_{COM} – затримка спрацьовування комутатора; t_{CK} – затримка спрацьовування схеми керування?
89. Які вимоги висуваються до часової діаграми сигналів G і u_i?
90. Які вимоги висуваються до проміжку часу t_{uG} між активними фронтами сигналів G і u_i?
91. Поясніть вираз (2.6).
92. Поясніть вираз (2.7).
93. В чому полягає різниця при визначенні значення t_{uG} для двоступеневих тригерів і тригерів з динамічним керуванням?
94. Як визначити значення мінімального періоду надходження сигналу G?
95. Поясніть вираз (2.8).
96. Поясніть вираз (2.9).
97. Як необхідно заповнювати таблицю істинності сигналу $u_{БТ}$ при можливості одночасної появи на входах регістра кількох керуючих сигналів?
98. Поясніть вираз (2.10).
99. В чому полягає різниця при реалізації схеми керування регістра за виразами (2.5) і (2.10)?
100. Які недоліки реалізації схеми керування регістра за виразом (2.10)?
101. Яку функцію виконує схема формування функцій збудження в регістрі?

102. В чому полягає процедура синтезу схеми формування функцій збудження?
103. В чому полягає різниця в синтезі схеми формування функцій збудження 8- і 64-розрядного регістра при використанні однакового типу базового тригера?
104. Яким чином функціонування схеми формування функцій збудження залежить від мікрооперацій, які виконує регістр?
105. Поясніть, яким чином заповнювати таблицю переходів для синтезу схеми формування функцій збудження?
106. Як визначати функції збудження для заданого базового тригера?
107. Поясніть заповнення колонки 4 в таблиці 2.6.
108. Для якого входу базового тригера спочатку визначається функція збудження?
109. Виконати синтез *СФФЗ* для забезпечення прийому інформації з комутатора регістра в *D*-тригер.
110. Виконати синтез *СФФЗ* для забезпечення прийому інформації з комутатора регістра в *RCS*-тригер.
111. Виконати синтез *СФФЗ* для забезпечення прийому інформації з комутатора регістра в *JK*-тригер.
112. Виконати синтез *СФФЗ* для забезпечення прийому інформації з комутатора регістра в *TC*-тригер.
113. Яку логічну функцію можна виконувати за допомогою *T*-тригера?
114. Який спосіб завантаження даних за допомогою зсувів використовується в сучасних комп'ютерних системах?
115. Які регістри називаються регістрами зсуву?
116. Назвіть класифікаційні ознаки, за якими розрізняються зсуви.
117. Які типи зсувів використовуються в регістрах відповідно до напряму зсувів?
118. Який тип зсуву відповідає передачі даних від старших розрядів до молодших?

119. Який тип зсуву відповідає передачі даних від молодших розрядів до старших?
120. В чому полягає різниця між зсувами вправо і вліво?
121. Як відрізнити більш старший розряд від більш молодшого в сучасних комп'ютерних системах?
122. Який розряд звільнюється при виконанні зсуву вмісту регістра вправо на один розряд?
123. Які розряди звільнюється при виконанні зсуву вмісту регістра вправо на *n* розрядів?
124. Які розряди звільнюється при виконанні зсуву вмісту регістра вліво на *n* розрядів?
125. Які існують типи зсувів в залежності від способу заповнення звільнених під час зсуву біт?
126. Яким чином виконується логічний зсув вправо на *n* розрядів?
127. Яким чином виконується логічний зсув вліво на *n* розрядів?
128. Яким чином виконується циклічний зсув вправо на *n* розрядів?
129. Яким чином виконується циклічний зсув вліво на *n* розрядів?
130. Чи використовується логічний зсув для виконання зсуву чисел зі знаком? Обґрунтуйте відповідь.
131. Який тип зсуву використовується для виконання зсуву чисел зі знаком?
132. Поясніть принцип виконання арифметичного зсуву вправо.
133. Поясніть принцип виконання арифметичного зсуву вліво.
134. Який тип зсуву виконується, якщо цей зсув позначений як *CL2*?
135. Який тип зсуву виконується, якщо цей зсув позначений як *CR3*?
136. Визначити результат виконання зсуву *CL1* вмісту 8-бітного регістра, якщо регістр до зсуву містив код 00110011.
137. Визначити результат виконання зсуву *CL2* вмісту 8-бітного регістра, якщо регістр до зсуву містив код 00110011.

138. Визначити результат виконання зсуву *CL3* вмісту 8-бітного регістра, якщо регістр до зсуву містив код 00110011.
139. Визначити результат виконання зсуву *CR1* вмісту 8-бітного регістра, якщо регістр до зсуву містив код 00110011.
140. Визначити результат виконання зсуву *CR2* вмісту 8-бітного регістра, якщо регістр до зсуву містив код 00110011.
141. Як зміниться вміст 8-бітного регістра при виконанні зсуву *CL8*?
142. Як зміниться вміст 8-бітного регістра при виконанні зсуву *CR8*?
143. Як зміниться вміст 8-бітного регістра при виконанні зсуву *CL4*?
144. Як зміниться вміст 8-бітного регістра при виконанні зсуву *CR4*?
145. Поясніть, як виконується зсув *L1.1*?
146. Поясніть, як виконується зсув *L2.01*?
147. Поясніть, як виконується зсув *0.R1*?
148. Поясніть, як виконується зсув *100.R3*?
149. Визначити результат виконання зсуву *L1.1* вмісту 8-бітного регістра, якщо регістр до зсуву містив код 00110011.
150. Визначити результат виконання зсуву *1.R1* вмісту 8-бітного регістра, якщо регістр до зсуву містив код 00110011.
151. Визначити результат виконання зсуву *01.R2* вмісту 8-бітного регістра, якщо регістр до зсуву містив код 00110011.
152. Поясніть, як виконується зсув *CF.R1*, де *CF* – ознака переносу в процесорі?
153. Поясніть, як виконується зсув *L1.CF*, де *CF* – ознака переносу в процесорі?
154. Визначити результат виконання зсуву *CF.R1* вмісту 8-бітного регістра, якщо регістр до зсуву містив код 00110011, а *CF* = 0.
155. Визначити результат виконання зсуву *L1.CF* вмісту 8-бітного регістра, якщо регістр до зсуву містив код 00110011, а *CF* = 1.
156. Як виконується в процесорі циклічний зсув вправо на один розряд з ознакою переносу (команда *RCR*)?

157. Як виконується в процесорі циклічний зсув вліво на один розряд з ознакою переносу (команда *RCL*)?
158. Який тип зсуву необхідно використовувати при послідовному завантаженні інформації в регістр?
159. Поясніть принцип виконання арифметичного зсуву вправо.
160. Поясніть принцип виконання арифметичного зсуву вліво.
161. Визначити результат виконання зсуву *AR1* вмісту 8-бітного регістра, якщо регістр до зсуву містив код 10110011.
162. Визначити результат виконання зсуву *AR1* вмісту 8-бітного регістра, якщо регістр до зсуву містив код 00110011.
163. Визначити результат виконання зсуву *AR2* вмісту 8-бітного регістра, якщо регістр до зсуву містив код 10110011.
164. Який тип зсуву має ознаки як арифметичного, так і логічного зсуву.
165. Визначити результат виконання зсуву *AL1.0* вмісту 8-бітного регістра, якщо регістр до зсуву містив код 10110011.
166. Визначити результат виконання зсуву *AL2.00* вмісту 8-бітного регістра, якщо регістр до зсуву містив код 10110011.
167. Як виконується арифметичний зсув вліво в процесорах сімейства Pentium (команда *SAL*)?
168. Яке значення записується в молодший розряд при виконанні команди *SAL*?
169. Як виконується обробка знакового розряду при виконанні команди *SAL*?
170. Як використовується ознака переповнення процесора при виконанні команди *SAL*?
171. Поясніть текст програми для пояснення виконання команди *SAL*?
172. Поясніть текст програми і результати її виконання, приведені на рис.2.17.
173. Прокоментуйте дані, приведені в табл.2.7.

174. Як зміниться значення числа, зсунутого на один розряд вправо за допомогою арифметичного зсуву?
175. Як зміниться значення числа, зсунутого на один розряд вліво за допомогою арифметичного зсуву?
176. Як виконується команда *SAR*?
177. Поясніть текст програми і результати її виконання, приведені на рис.2.18.
178. Прокоментуйте дані, приведені в табл.2.8.
179. Які регістри називаються реверсивними?
180. Поясніть різницю між виразами (2.11) і (2.12).
181. Як визначити динамічні параметри регістра?
182. На базі *D*-тригера виконати синтез 4-бітного регістра з керованою синхронізацією в будь-якому базисі логічних елементів. Регістр виконує такі мікрооперації: *uA*: *Rg* := *A*; *uR*: *Rg* := *11.R2*(*Rg*) ; *uL*: *Rg* := *L1.1*(*Rg*).
183. На базі *RCS*-тригера виконати синтез 4-бітного регістра з керованою синхронізацією в будь-якому базисі логічних елементів. Регістр виконує такі мікрооперації: *uA*: *Rg* := $\overline{A}$; *uR*: *Rg* := *CR2*(*Rg*) ; *uL*: *Rg* := *AL1.0*(*Rg*).
184. На базі *JK*-тригера виконати синтез 4-бітного регістра з керованою синхронізацією в базисі Шефера. Регістр виконує такі мікрооперації: *uA*: *Rg* := $\overline{A}$; *uR*: *Rg* := *CL2*(*Rg*) ; *uL*: *Rg* := *AR2*(*Rg*).
185. На базі *TC*-тригера виконати синтез 4-бітного регістра з керованою синхронізацією в будь-якому базисі логічних елементів. Регістр виконує такі мікрооперації: *uA*: *Rg* := *A*; *uR*: *Rg* := *L2.00*(*Rg*) ; *uL*: *Rg* := *CR4*(*Rg*).
186. Чи можна використовувати прозорі тригери в зсувних регістрах? Обґрунтуйте відповідь.

187. Чи можна використовувати асинхронні входи базових тригерів для організації паралельного завантаження інформації в регістр? Обґрунтуйте відповідь.
188. Поясніть принцип роботи пристрою, що перетворює паралельний код в послідовний.
189. Виконати синтез пристрою для перетворення паралельного коду в послідовний.
190. Поясніть принцип роботи пристрою, що перетворює послідовний код в паралельний.
191. Виконати синтез пристрою для перетворення послідовного коду в паралельний.
192. Чому при реалізації регістрів не доцільно використовувати *TC*- тригери?
193. В яких регістрах не використовується схема формування функцій збудження?

2.1.2 Регістри з некерованою синхронізацією

Структурна схема *n*-розрядного регістра з некерованою синхронізацією приведена на рис.2.32. В основному на структурній схемі використовуються такі ж складові частини, що і на рис.2.1, які мають те ж саме призначення.

З рисунка 2.31 можна побачити, що в регістрі з некерованою синхронізацією використовується дві схеми керування:

- *СКnU* – схема керування для формування сигналу *nU*;
- *СКБТ* – схема керування станом базових тригерів.

Розглянемо особливості синтезу регістрів з некерованою синхронізацією.

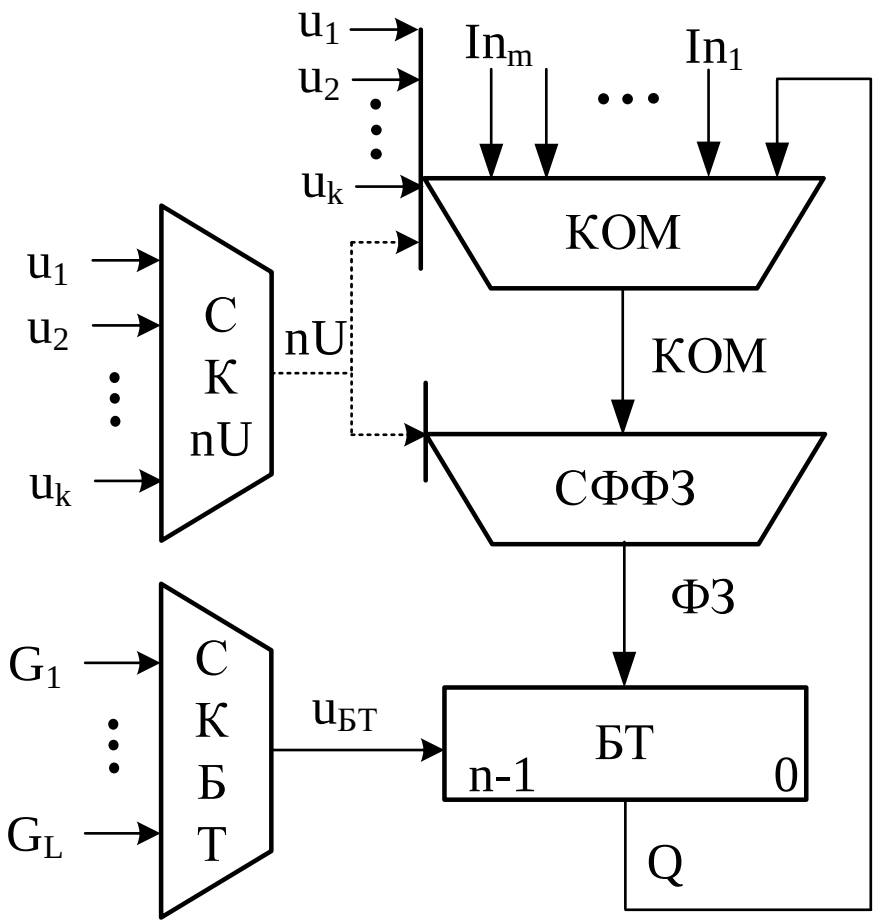


Рисунок 2.32 – Структурна схема регістра з некерованою синхронізацією

Зі структурної схеми можна побачити, що в якості вхідних сигналів схема керування *СКБТ* використовує тільки сигнали з генератора імпульсів G_i, а підсумковий сигнал $u_{БТ}$ вже безпосередньо підключається до входів синхронізації тригерів. Як правило, для побудови регістрів застосовується однофазна синхронізація. Це означає, що використовується тільки один сигнал G, тобто $u_{БТ} = G$. Таким чином, в регістрах з некерованою синхронізацією на входи синхронізації тригерів завжди надходять прямокутні імпульси, поки включена напруга живлення незалежно від наявності або відсутності керуючих сигналів регістра u_i.

Розглянемо функціонування регістрів, представлених на рис.2.21 і 2.28, якщо сигнал G безпосередньо надходить до входів синхронізації тригерів. За наявності керуючих сигналів регістр спрацьовує коректно, як показано на рис.2.23 і 2.30.

Але за відсутності активних значень керуючих сигналів відповідно до (2.2) на виходах комутатора будуть формуватися нульові значення, тобто по сигналу *G* регістр буде переключатися в нульовий стан замість виконання збереження інформації. Схема такого регістра приведена на рис.2.33 (по зрівнянню зі схемою на рис.2.21 видалена схема керування, а сигнал *G* підключений до входів *C* базових тригерів). Результати моделювання цього регістра приведені на рис.2.34.

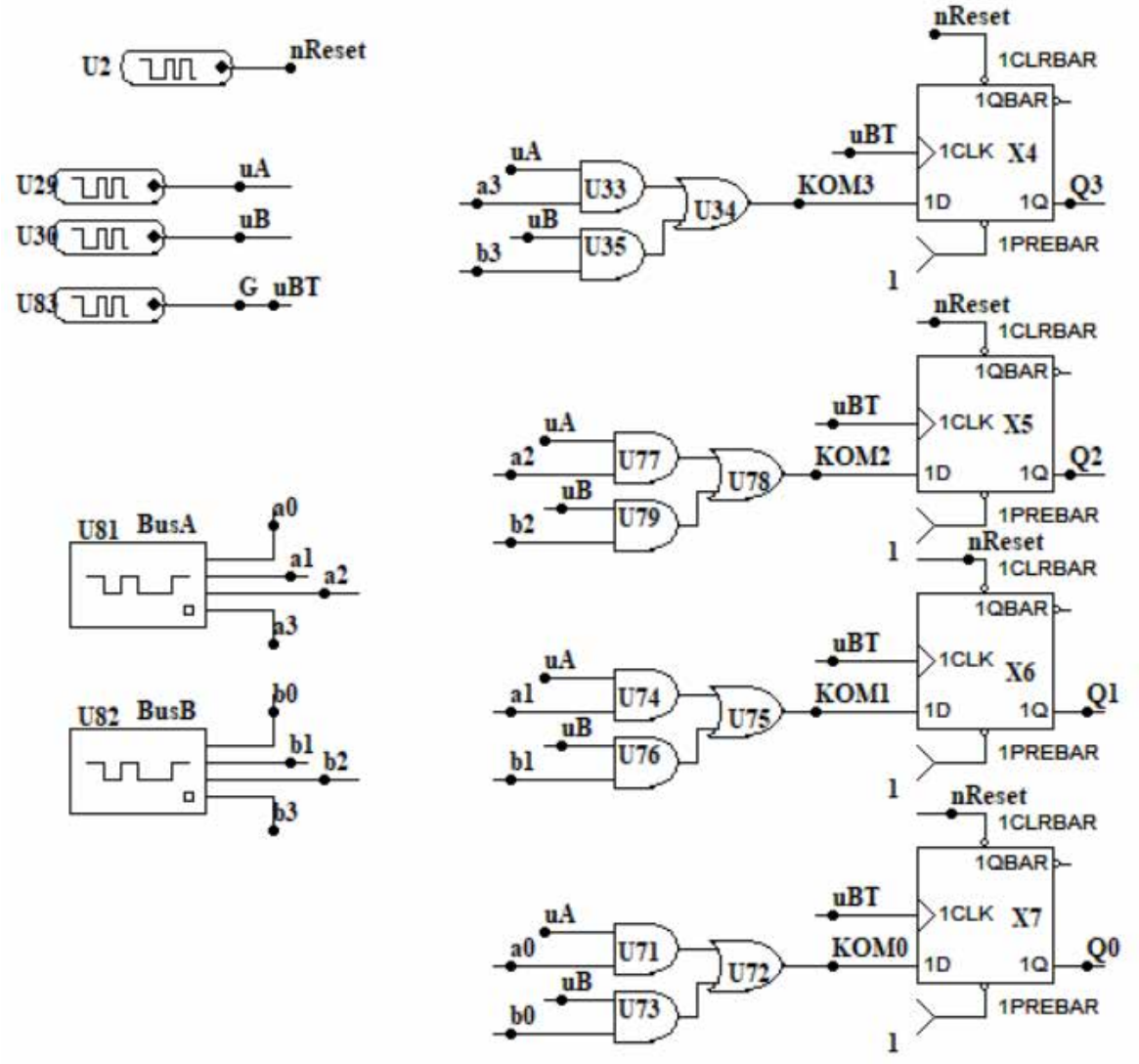


Рисунок 2.33 – Схема регістра з некерованою синхронізацією з некоректною роботою при відсутності керуючих сигналів

Відповідно до принципу роботи за відсутності активних значень керуючих сигналів регістр повинен зберігати свій стан, але результати

моделювання свідчать, що в схемі спостерігається переключення тригерів регістра в нульовий стан (на часовій діаграмі виділено пунктиром).

Таким чином, для організації регістра з некерованою синхронізацією необхідно забезпечити стан збереження інформації за відсутності активних значень керуючих сигналів.

Послідовність кроків для виконання синтезу регістра з некерованою синхронізацією збігається з процедурою синтезу регістра з керованою синхронізацією, розглянутою в п.2.1.1.

Крім того, зі структурної схеми можна побачити, що схема керування *CKnU* формує сигнал *nU*, який надходить або на комутатор, або на схему формування функцій збудження, що показано на структурній схемі пунктиром, тобто врахування особливостей використання некерованої синхронізації відбувається на етапі синтезу комутатора або схеми формування функції збудження.

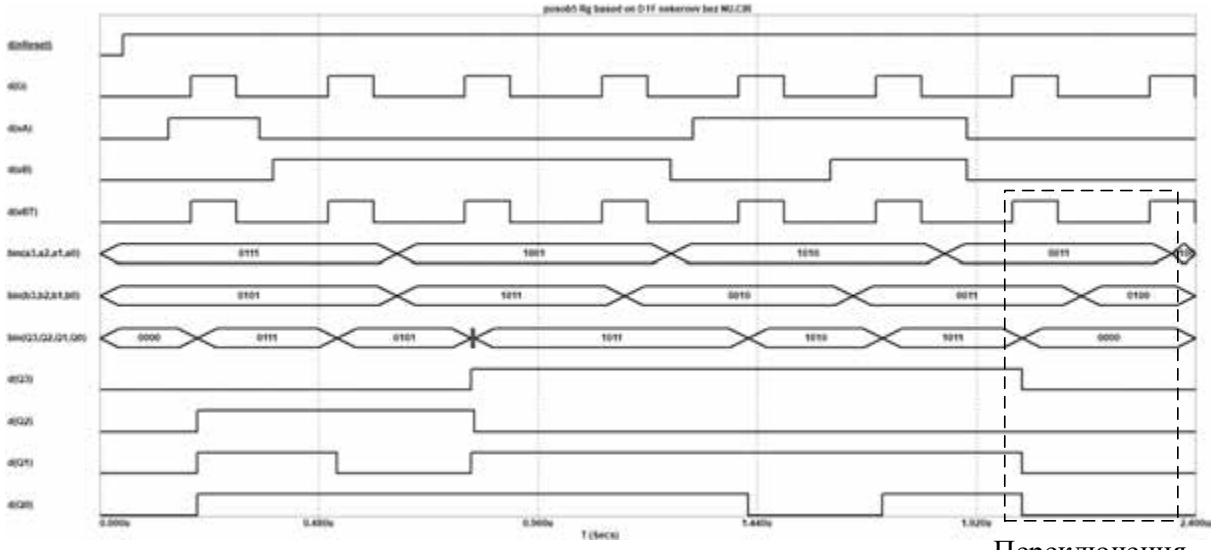


Рисунок 2.34 – Результати моделювання регістра з некерованою синхронізацією з некоректною роботою при *uA = 0* і *uB = 0*

Таким чином, апаратна реалізація регістра з некерованою синхронізацією можлива двома способами:

- на основі комутатора;

- на основі схеми формування функцій збудження.

Розглянемо спочатку формування сигналу *nU* за допомогою схеми керування *СКnU*, а далі виконаємо синтез комутатора і *СФФЗ* для побудови регістра з некерованою синхронізацією.

2.1.2.1 Синтез схеми керування *СКnU*

Як вже було зазначено вище, для коректного функціонування регістра з некерованою синхронізацією необхідно забезпечити налаштування цього регістра на виконання мікрооперації збереження інформації за відсутності активних значень керуючих сигналів. Для цього будемо використовувати сигнал *nU*, який формується схемою керування *СКnU*, тобто синтез схеми керування *СКnU* полягає у формуванні сигналу *nU*. Вказаний сигнал приймає активне значення за відсутності керуючих сигналів, тобто логічний вираз для цього сигналу виглядає наступним чином:

$$nU = \overline{u_1} \cdot \overline{u_2} \cdot \ldots \cdot \overline{u_k}. \qquad (2.14)$$

2.1.2.2 Синтез комутатора

Синтез комутатора, як правило, доцільно виконувати за допомогою використання таблиці мікрооперацій, який, як вже було відзначено раніше, базується на принципах роботи керуючих автоматів і передбачає, що сукупність активних значень керуючих сигналів, що надходять до регістру, являє собою або унітарний двійковий код, або сукупність неактивних значень цих сигналів. При цьому за наявності активності керуючого сигналу використання таблиці мікрооперацій для регістрі з керованою і некерованою синхронізацією є однаковим. Тому розглянемо, яким чином необхідно використовувати таблицю мікрооперацій за відсутності активних значень керуючих сигналів. Нагадаємо, що в цьому випадку схема керування регістра формує сигнал *nU*, активне значення якого є ознакою відсутності активних керуючих сигналів на входах регістра.

У випадку синтезу регістра з некерованою синхронізацією таблиця мікрооперацій містить на один рядок більше, ніж для регістрів з керованою синхронізацією, тобто містить $k+1$ рядків, де k – кількість мікрооперацій (МО), які виконує регістр. Кількість стовпців (n , де n – кількість розрядів регістра) є однаковою для обох типів регістрів. Узагальнений вид таблиці мікрооперацій приведений в табл.2.13.

Таблиця 2.13 – Узагальнена таблиця МО для синтезу комутатора регістра з некерованою синхронізацією

МО	$КОМ_{n-1}$	$КОМ_{n-2}$	…	$КОМ_1$	$КОМ_0$
u_1					
u_2					
…					
u_k					
nU					

Як і раніше кожний стовпчик таблиці мікрооперацій відображує стан відповідної вихідної лінії комутатора в залежності від певного активного значення керуючого сигналу u_i або сигналу nU.

Нагадаємо, що для виконання синтезу комутатора за допомогою таблиці МО необхідно виконати послідовно заповнення клітин таблиці мікрооперацій, а після цього – визначити логічні вирази для кожного виходу комутатора.

Заповнення клітин цієї таблиці за наявності керуючого сигналу виконується згідно з розглянутими в п.2.1.1 правилами.

Для забезпечення збереження інформації в регістрі за відсутності активних керуючих сигналів необхідно здійснити перезапис стану i-того тригера регістра на цей же тригер.

Таким чином, для регістра з некерованою синхронізацією (наприклад, за умови, що регістр виконує тільки паралельне завантаження інформації з

зовнішніх джерел) таблиця мікрооперацій буде виглядати таким чином (табл.2.14):

Таблиця 2.14 – Узагальнений вигляд таблиці мікрооперацій регістра з некерованою синхронізацією

MO	KOM_{n-1}	KOM_{n-2}	…	KOM_1	KOM_0
u_1	$In_{1\ n-1}$	$In_{1\ n-2}$	…	$In_{1\ 1}$	$In_{1\ 0}$
…	…	…	…	…	…
u_i	$In_{i\ n-1}$	$In_{i\ n-2}$	…	$In_{i\ 1}$	$In_{i\ 0}$
…	…	…	…	…	…
u_k	$In_{k\ n-1}$	$In_{k\ n-2}$	…	$In_{k\ 1}$	$In_{k\ 0}$
nU	Q_{n-1}	Q_{n-2}	…	Q_1	Q_0

Після заповнення таблиці мікрооперацій визначимо логічні вирази, що описують функціонування комутатора (див. п.2.1.1). В результаті логічні вирази для реалізації комутатора виглядають наступним чином:

$$KOM_{n-1} = In_{1\ n-1} \cdot u_1 \vee \ldots \vee In_{i\ n-1} \cdot u_i \vee \ldots \vee In_{k\ n-1} \cdot u_k \vee Q_{n-1} \cdot nU;\ i = 1, k;$$

$$\ldots \qquad (2.15)$$

$$KOM_0 = In_{1\ 0} \cdot u_1 \vee \ldots \vee In_{i\ 0} \cdot u_i \vee \ldots \vee In_{k\ 0} \cdot u_k \vee Q_0 \cdot nU\ ;\ i = 1, k.$$

Вирази (2.15) можна записати в загальному вигляді:

$$KOM_j = In_{1\ j} \cdot u_1 \vee \ldots \vee In_{i\ j} \cdot u_i \vee \ldots \vee In_{k\ j} \cdot u_k \vee Q_j \cdot nU,\ j = 0, n\text{-}1;\ i = 1, k. \qquad (2.16)$$

Таким чином, під час синтезу регістра з некерованою синхронізацією логічна схема комутатора ускладнюється за рахунок появи додаткового терму $Q_j \cdot nU$.

Аналогічні логічні вирази можна отримати, використовуючи класичний метод синтезу за допомогою таблиці істинності.

Реалізація некерованої синхронізації за допомогою комутатора може бути використана для базових тригерів будь-якого типу.

2.1.2.3 Синтез *СФФЗ*

Нагадаємо, що *СФФЗ* використовується для формування функцій збудження базових тригерів для забезпечення завантаження інформації в ці тригери з виходів комутатора.

На відміну від попереднього способу, реалізація некерованої синхронізації за допомогою *СФФЗ* неможлива для тригерів, які не виконують операцію збереження інформації, використовуючи інформаційні входи (наприклад, *D*-, *DV*-тригери), тобто цей спосіб може бути використаний тільки в тому разі, якщо в якості базових тригерів використовуються *RCS*-, *JK*-, *TC*-тригери.

Процедура синтезу, як і раніше, виконується для одного *i*-того розряду регістра та складається з розробки таблиці переходів для завантаження значення стану вихідного сигналу *i*-того розряду комутатора до базового тригера; визначення і мінімізації функцій збудження базових тригерів тригера регістра та реалізації схеми.

Розглянемо таблицю переходів для реалізації некерованої синхронізації (табл.2.15).

Відповідно до структурної схеми новий стан *i*- того базового тригера визначається за допомогою функції чотирьох змінних $f(G, nU, KOM_i, Q_i^t)$, в результаті чого таблиця переходів буде виглядати таким чином:

В колонках 1-4 містяться вхідні змінні, в тому числі і попередній стан *i*-того базового тригера.

Перша половина таблиці переходів (рядки 0-7) відповідає неактивному значенню сигналу *G* (нагадаємо, що неактивне значення сигналів *G* і $u_{БТ}$ позначаємо нулем), в результаті чого регістр перебуває в стані збереження інформації, тобто $Q_i^{t+1} = Q_i^t$ (стовпчик 5).

Рядки 12-15 відповідають активному значенню сигналу *nU*, що сигналізує про відсутність активних керуючих сигналів на входах регістра. Але в зв’язку з тим, що при цьому сигнал *G* також приймає активне значення

($G = 1$), то регістр також налаштовується на збереження інформації ($Q_i^{t+1} = Q_i^t$).

Таблиця 2.15 – Таблиця переходів для синтезу *СФФЗ* в регістрі з некерованою синхронізацією

Номери наборів	$u_{БТ} = G$	nU	$КОМ_i$	Q_i^t	Q_i^{t+1}
	1	2	3	4	5
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

Рядки 8-11 відповідають неактивному значенню сигналу nU, що сигналізує про наявність певного активного керуючого сигналу, тобто регістр налаштовується на виконання мікрооперації завантаження інформації з виходу комутатора по сигналу G ($Q_i^{t+1} = КОМ_i$).

Далі виконується визначення функцій збудження базових тригерів за аналогією з п.2.1.1.3 (в табл.2.15 не показано). Детальний опис визначення функцій збудження базових тригерів приведений в [1] та розділі 1 цього посібника.

Після цього виконується мінімізація функцій збудження базових тригерів.

2.1.2.4 Приклади синтезу регістрів з некерованою синхронізацією

В якості прикладу розглянемо синтез регістра з некерованою синхронізацією, який виконує мікрооперації, задані в прикладі 2.2.

Приклад 2.3. На базі *D*-тригера виконати синтез 4-бітного реверсивного регістра з некерованою синхронізацією в будь-якому базисі логічних елементів. Регістр виконує такі мікрооперації: *uA*: *Rg* := *A*; *uR*: *Rg* := *AR1*(*Rg*) ; *uL*: *Rg* := *CL1*(*Rg*). Структурна схема регістра приведена на рис.2.27.

Розв'язок

Синтез регістра також будемо виконувати відповідно до порядку, приведеному в підрозділах 2.1.1 і 2.1.2.

Аналіз функціонування регістра під час виконання заданих мікрооперацій приведений в підрозділі 2.1.1.5.

Реалізацію некерованої синхронізації виконаємо за допомогою схеми формування функцій збудження.

Відповідно до умови завдання до складу таблиці мікрооперацій входять 4 рядки (3 мікрооперації та сигнал формування некерованої синхронізації *nU*) та 4 стовпця згідно з розрядністю регістра.

Нагадаємо, що кожний стовпчик таблиці мікрооперацій відображує стан відповідної вихідної лінії комутатора в залежності від активного значення сигналу *uA*, *uR*, *uL* або *nU*.

Таблиця мікрооперацій для синтезу комутатора приведена в табл.2.16.

Таблиця 2.16 – Таблиця МО для синтезу комутатора (приклад 2.3)

MO	KOM_3	KOM_2	KOM_1	KOM_0
1	2	3	4	5
uA	a_3	a_2	a_1	a_0
uR	Q_3	Q_3	Q_2	Q_1
uL	Q_2	Q_1	Q_0	Q_3
nU	Q_3	Q_2	Q_1	Q_0

В результаті заповнення клітин таблиці мікрооперацій можна отримати логічні вирази для кожного виходу комутатора:

$$\begin{aligned} КОМ_3 &= uA \cdot a_3 \vee uR \cdot Q_3 \vee uL \cdot Q_2 \vee nU \cdot Q_3; \\ КОМ_2 &= uA \cdot a_2 \vee uR \cdot Q_3 \vee uL \cdot Q_1 \vee nU \cdot Q_2; \\ КОМ_1 &= uA \cdot a_1 \vee uR \cdot Q_2 \vee uL \cdot Q_0 \vee nU \cdot Q_1; \\ КОМ_0 &= uA \cdot a_0 \vee uR \cdot Q_1 \vee uL \cdot Q_3 \vee nU \cdot Q_0; \end{aligned} \qquad (2.17)$$

Вирази для $КОМ_2$ та $КОМ_1$ можна записати в загальному вигляді:

$$КОМ_i = uA \cdot a_i \vee uR \cdot Q_{i+1} \vee uL \cdot Q_{i-1} \vee nU \cdot Q_1,\ i = 1,2;$$

Проєктування схеми керування nU виконується відповідно до п. 2.1.2.1, тобто $nU = \overline{uA} \cdot \overline{uR} \cdot \overline{uL}$.

Відповідно до принципу організації однофазної некерованої синхронізації $u_{БТ} = G$.

Згідно з табл.2.6 і рис.2.3 функція збудження для D-тригера має вигляд: $D_i = КОМ_i$.

Далі відповідно до умови завдання для побудови регістра використовуємо будь-який базис логічних елементів, в зв’язку з чим всі отримані на попередніх кроках логічні вирази залишимо без перетворень.

Таким чином, виконано синтез всіх комбінаційних складових частин регістра з некерованою синхронізацією, що дає змогу побудувати логічну схему цього регістра, яка приведена на рис.2.35.

На рис.2.36 приведена логічна схема для моделювання функціонування синтезованого регістра, а на рис.2.37 – результати моделювання цього регістра.

Аналізуючи результати моделювання можна побачити, що за наявності неактивних значень керуючих сигналів регістр, незважаючи на активність сигналу G, перебуває в стані збереження інформації. На часовій діаграмі цей інтервал часу виділений пунктиром.

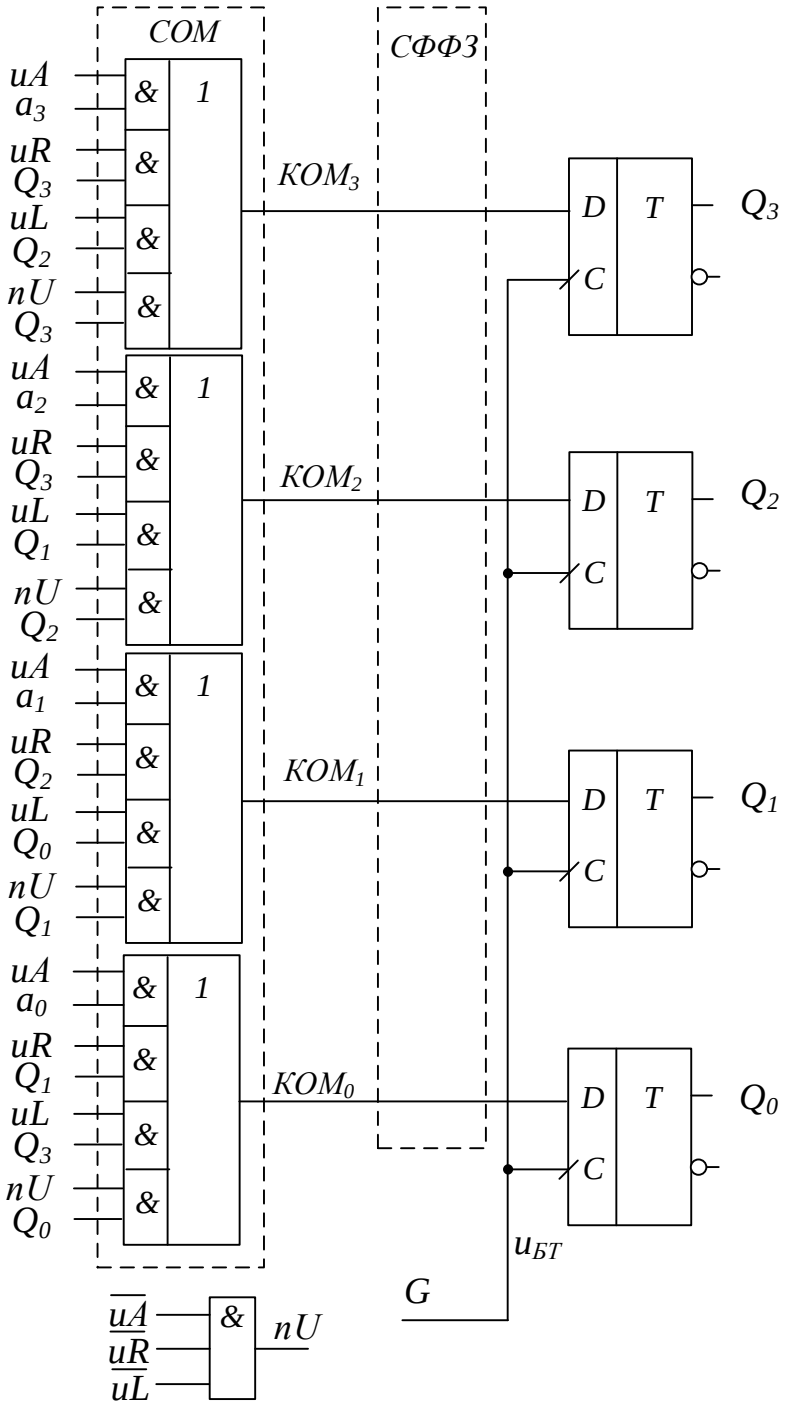


Рисунок 2.35 – Логічна схема регістра з некерованою синхронізацією (приклад 2.3)

Далі розглянемо приклад синтезу регістра з некерованою синхронізацією, де наявність сигналу *nU* враховується на етапі синтезу *СФФЗ*. Як вже було відзначено раніше, врахування наявності сигналу *nU* під час синтезу *СФФЗ* можливо, якщо базовий тригер, наприклад, не є

D- тригером. Тому розглянемо синтез регістра з некерованою синхронізацією, побудованого на базі *JK*-тригера.

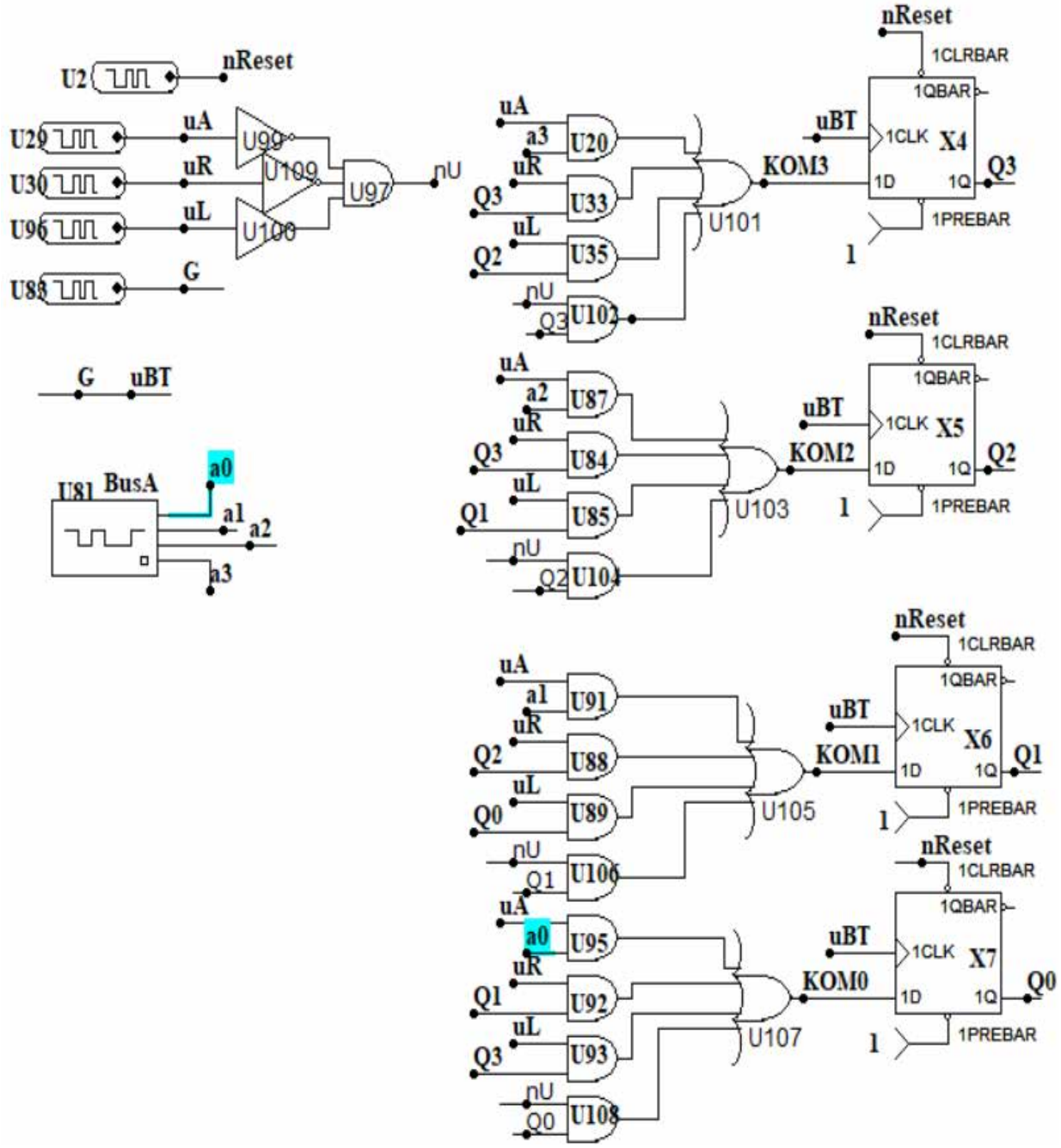


Рисунок 2.36 – Схема регістра з некерованою синхронізацією (приклад 2.3)

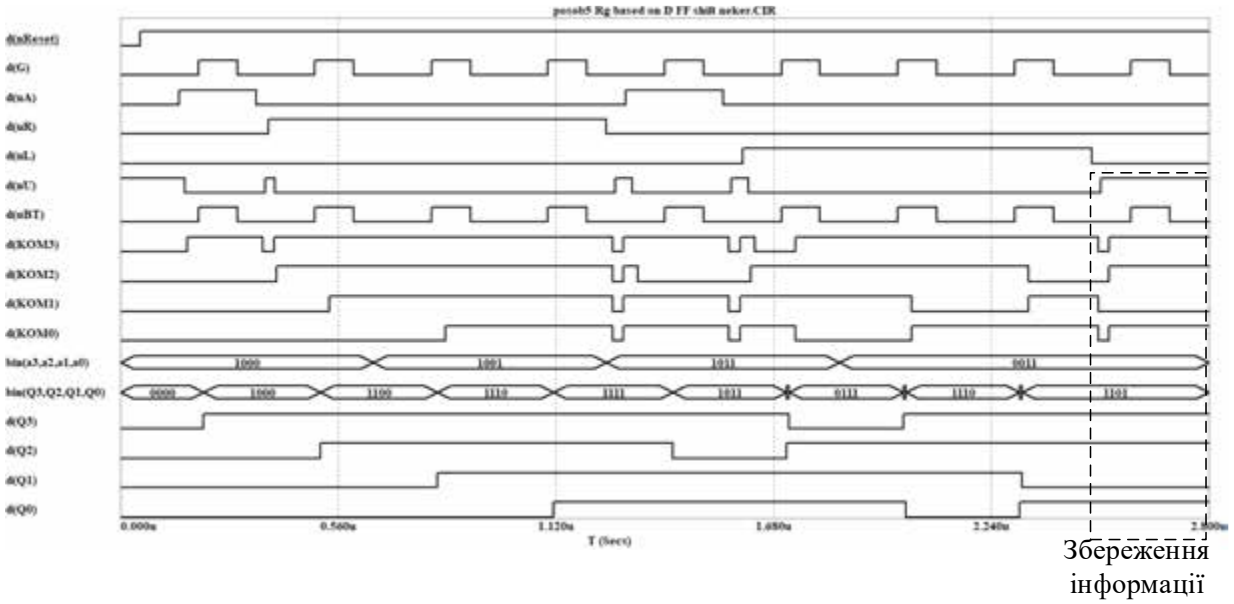


Рисунок 2.37 – Результати моделювання регістра з некерованою синхронізацією (приклад 2.3)

Приклад 2.4. На базі *JK*-тригера виконати синтез 4-бітного реверсивного регістра з некерованою синхронізацією в будь-якому базисі логічних елементів. Регістр виконує такі мікрооперації: *uA*: $Rg := A$; *uR*: $Rg := AR1(Rg)$; *uL*: $Rg := CL1(Rg)$. Структурна схема регістра приведена на рис.2.27.

Розв'язок

Синтез регістра також будемо виконувати відповідно до порядку, приведеному в підрозділах 2.1.1 і 2.1.2.

Аналіз функціонування регістра під час виконання заданих мікрооперацій приведений в підрозділі 2.1.1.5.

Синтез комутатора виконується таким же чином, як в прикладі 2.2 (табл.2.11).

Проєктування схем керування виконується відповідно до п.2.1.2.1, як в прикладі 2.3, тобто $nU = \overline{uA} \cdot \overline{uR} \cdot \overline{uL}$, а $u_{БТ} = G$.

Розглянемо синтез *СФФЗ* з врахуванням сигналу *nU*.

Таблиця переходів для реалізації некерованої синхронізації відповідно до табл.2.15 з функціями збудження для *JK*-тригера приведена в табл.2.17.

Детальний опис визначення функцій збудження базових тригерів приведений в [1], а приклад визначення функцій збудження *JK*-тригера приведений в табл.1.3.

Таблиця 2.17 – Таблиця переходів для синтезу *СФФЗ* (приклад 2.4)

Номери наборів	$u_{БТ} = G$	nU	$КОМ_i$	Q_i^t	Q_i^{t+1}	C	J_i	K_i
	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	*	*
1	0	0	0	1	1	0	*	*
2	0	0	1	0	0	0	*	*
3	0	0	1	1	1	0	*	*
4	0	1	0	0	0	0	*	*
5	0	1	0	1	1	0	*	*
6	0	1	1	0	0	0	*	*
7	0	1	1	1	1	0	*	*
8	1	0	0	0	0	1	0	*
9	1	0	0	1	0	1	*	1
10	1	0	1	0	1	1	1	*
11	1	0	1	1	1	1	*	0
12	1	1	0	0	0	1	0	*
13	1	1	0	1	1	1	*	0
14	1	1	1	0	0	1	0	*
15	1	1	1	1	1	1	*	0

В таблиці 2.17 додані стовпчики 6-8, в яких вказуються функції збудження базового тригера.

Після цього виконується мінімізація функцій збудження базових тригерів.

Визначимо логічні вирази для функцій збудження *JK*-тригера. Враховуючи однакові значення в колонках 1 і 6 можна записати, що $C = u_{БТ} = G$. Логічні вирази для входів *J* і *K* визначимо за допомогою карт Карно (рис.2.38).

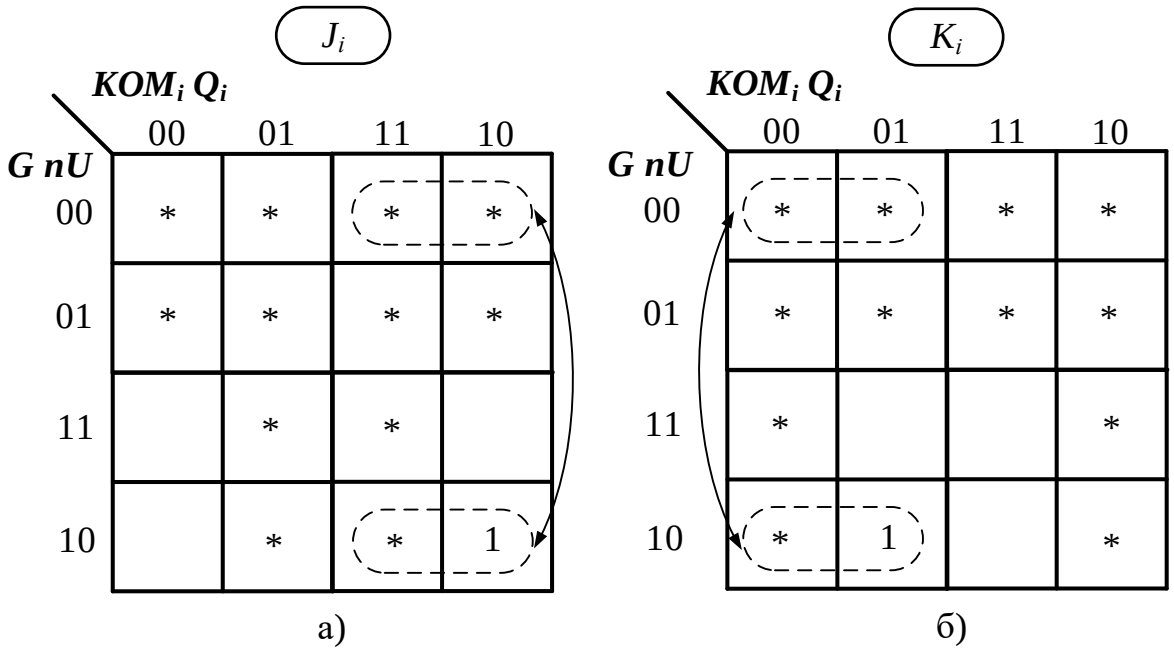


Рисунок 2.38– Мінімізація функцій збудження (приклад 2.4)

Відповідно до карт Карно отримаємо:

$$J_i = \overline{nU} \cdot KOM_i; \quad K_i = \overline{nU} \cdot \overline{KOM_i}.$$

Для того, щоб не використовувати парафазний код з виходу комутатора (KOM_i і $\overline{KOM_i}$), перетворимо вираз K_i за законом де Моргана:

$$K_i = \overline{nU} \cdot \overline{KOM_i} = \overline{nU \vee KOM_i}.$$

Крім того, для того, щоб не використовувати інверсні значення керуючих сигналів, перетворимо вираз для сигналу nU в інший базис:

$$nU = \overline{uA} \cdot \overline{uR} \cdot \overline{uL} = \overline{uA \vee uR \vee uL}\ .$$

Таким чином, $\overline{nU} = uA \vee uR \vee uL$.

В результаті будемо використовувати таку сукупність формул:

$$J_i = \overline{nU} \cdot KOM_i; \quad K_i = \overline{nU \vee KOM_i}; \quad nU = \overline{uA \vee uR \vee uL};$$

Логічні вирази для реалізації комутатора будемо реалізовувати в булевському базисі, тобто ці вирази залишимо без перетворень.

Таким чином, виконано синтез всіх комбінаційних складових частин регістра з некерованою синхронізацією, що дає змогу побудувати логічну схему цього регістра, яка приведена на рис.2.39.

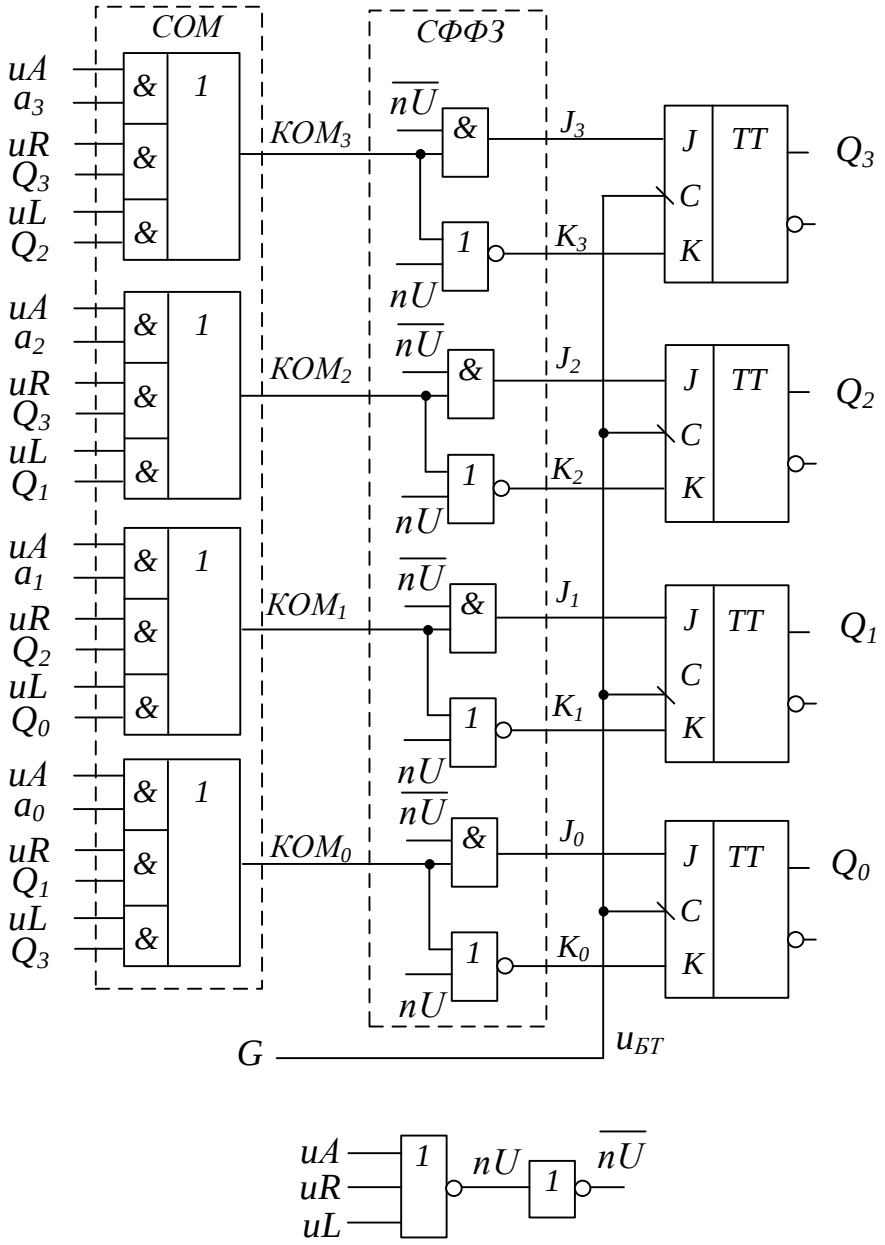


Рисунок 2.39 – Логічна схема регістра з некерованою синхронізацією (приклад 2.4)

На рис.2.40 приведена логічна схема, яка використовується для моделювання функціонування синтезованого регістра. В якості базових тригерів використовується *JK*-тригер ***SN74107***, який спрацьовує за заднім фронтом сигналу синхронізації (для даного прикладу сигнал *G*) та має асинхронний вхід скиду тригера.

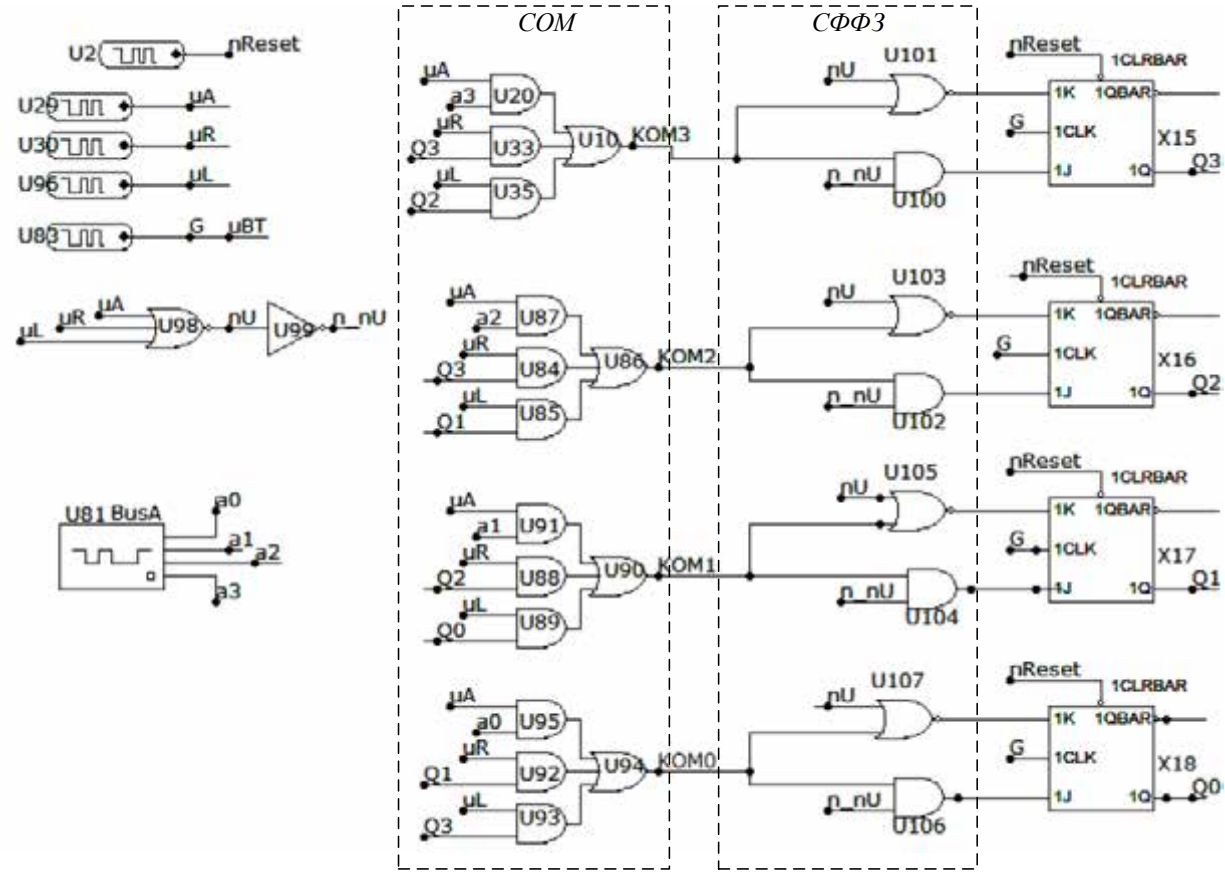


Рисунок 2.40 – Логічна схема для моделювання регістра з некерованою синхронізацією (приклад 2.4)

Результати моделювання цього регістра приведені на рис.2.41.

На часовій діаграмі можна побачити, що за наявності неактивних значень керуючих сигналів регістр перебуває в стані збереження інформації, що забезпечується *СФФ3*.

На останок відзначимо, що регістри як з керованою, так і з некерованою синхронізацією випускаються у вигляді інтегральних схем.

Так, наприклад, регістри ***SN74173***, ***SN74199***, ***SN74299***, ***SN74377***, ***SN74LS323*** та інші являють собою регістри з некерованою синхронізацією.

2.1.3 Регістри з двотактовим завантаженням інформації

За наявності двофазної синхронізації, тобто, наприклад, використовуються сигнали G_1 і G_2, завантаження в регістр може бути

виконано за два мікротакти, тобто такт синхронізації розділяється на дві частини. Перший мікротакт пов'язаний з надходженням активного значення сигналу G_1 , а другий – G_2.

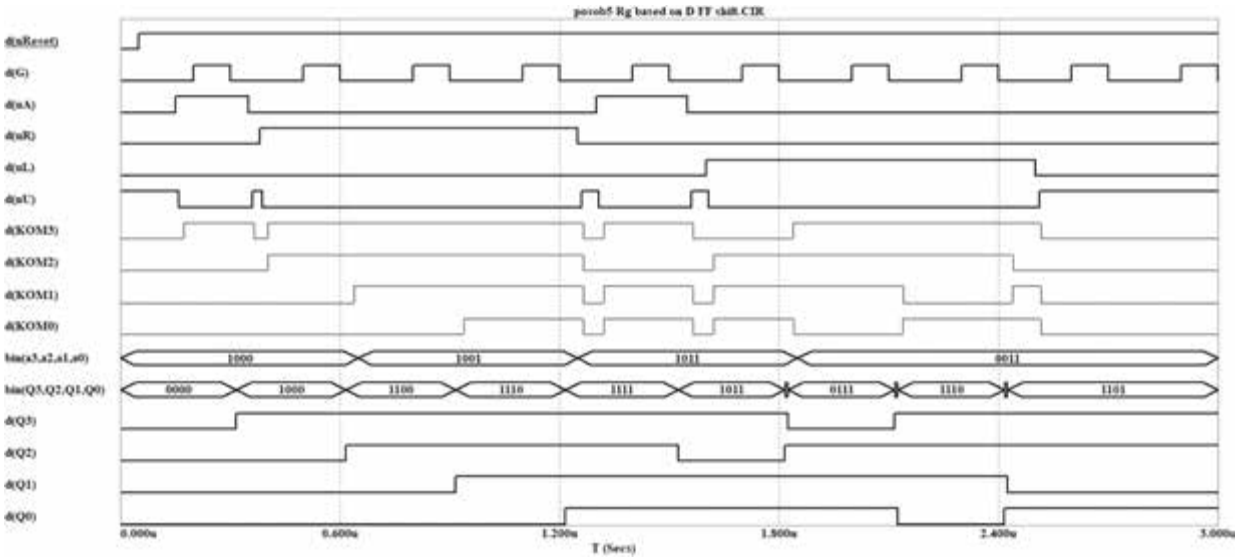


Рисунок 2.41 – Результати моделювання регістра з некерованою синхронізацією (приклад 2.4)

Для коректної роботи для сигналів G_1 і G_2 необхідно дотримуватися умови $G_1 \cdot G_2 = 0$, тобто активні значення сигналів в часі не збігаються (рис.2.42).

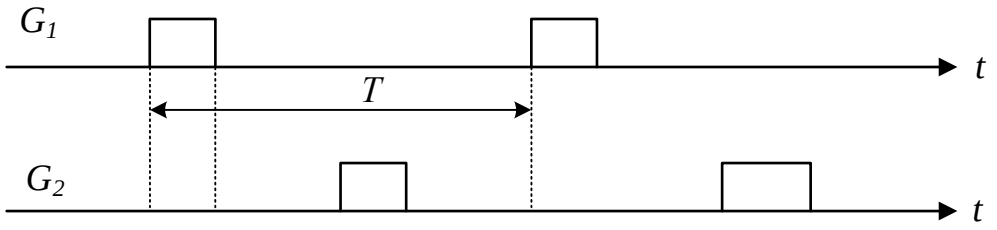


Рисунок 2.42 – Часова діаграма керуючих сигналів G_1 і G_2 регістра

Під час першого мікротакту здійснюється скид вмісту регістра, а під час другого мікротакту виконується завантаження інформації в базові тригери. Скид регістра, як правило, здійснюється за допомогою асинхронного входу R базового тригера, а сигнал G_2 вже безпосередньо підключається на вхід синхронізації базового тригера.

Використання такого способу завантаження інформації спрощує функції збудження базового тригера в разі використання в якості цих тригерів *RCS*-, *JK*-, *TC*-тригерів. Функції збудження *i*-того тригера в цьому випадку будуть мати вигляд:

$$S_i = J_i = T_i = KOM_i;\ R_i = K_i = 0.$$

Логічна схема для реалізації двотактового завантаження інформації в регістр на базі *JK*-тригера (розряд *i*) приведена на рис.2.43. Часова діаграма сигналів G_1 і G_2 приведена на рис.2.44.

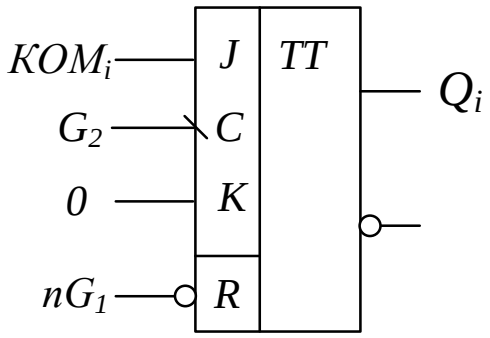


Рисунок 2.43 – Фрагмент логічної схеми і-того розряду з двотактовим завантаженням інформації

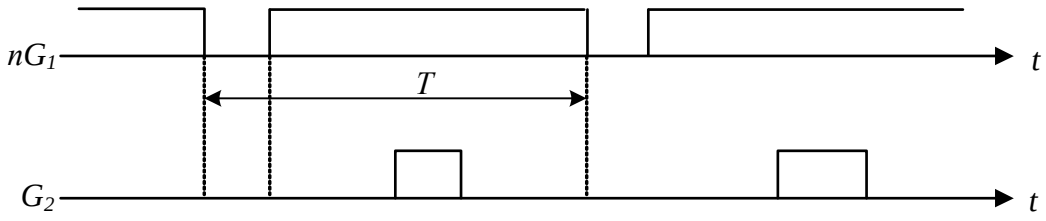


Рисунок 2.44 – Часова діаграма керуючих сигналів G_1 і G_2 для регістра, фрагмент схеми якого приведений на рис.2.43

Сигнал G_1 є сигналом з інверсним керуванням [1] і позначений nG_1. За низьким рівнем цього сигналу, який надходить на асинхронний вхід *nR*, тригер переключається в нульовий стан, а за активним фронтом сигналу G_2 переключається в одиничний стан, якщо $KOM_i = 1$, або залишається в нульовому стані при $KOM_i = 0$.

Недоліком такого способу завантаження інформації в регістр є обмеження його використання тільки для регістрів, які приймають

інформацію із зовнішніх джерел, тобто такий спосіб не можна використовувати для апаратної реалізації зсувних регістрів.

Контрольні завдання та запитання

1. Назвіть типи регістрів відповідно до способу синхронізації.
2. В чому полягає різниця між регістрами з керованою і некерованою синхронізацією?
3. Які функції виконує регістр?
4. Що називається мікрооперацією?
5. Чи можуть в складі регістра використовуватися тригери різних типів за функціональним призначенням?
6. Яким чином здійснюється організація некерованої синхронізації в регістрах?
7. Поясніть призначення сигналу *nU*.
8. Звідки надходить сигнал *nU*?
9. Яким чином реалізується некерована синхронізація за допомогою комутатора?
10. Яким чином реалізується некерована синхронізація за допомогою *СФФЗ*?
11. На яких типах тригерів реалізується некерована синхронізація за допомогою комутатора?
12. На яких типах тригерів реалізується некерована синхронізація за допомогою *СФФЗ*?
13. Приведіть структурну схему регістра з некерованою синхронізацією.
14. Які складові частини входять до складу регістра з некерованою синхронізацією?
15. Поясніть призначення комутатора в складі регістра.
16. Поясніть призначення схеми формування функцій збудження в складі регістра.

17. Поясніть призначення схеми керування *CKnU* в складі регістра.
18. Поясніть призначення базових тригерів в складі регістра.
19. Поясніть взаємодію складових частин під час функціонування регістра.
20. Які складові вузли регістра є комбінаційними?
21. Які складові вузли регістра виконують мікрооперацію збереження інформації?
22. Які складові вузли регістра забезпечують запис даних до базового тригера?
23. Поясніть термін «функція збудження».
24. Яким чином використовується сигнал *G* в регістрах з некерованою синхронізацією?
25. Поясніть термін «однофазна система синхронізації».
26. Які дії виконує схема формування функцій збудження в складі регістра з некерованою синхронізацією?
27. Поясніть призначення сигналу $u_{БТ}$?
28. В яких випадках в схемі регістра може бути відсутнім комутатор?
29. В яких випадках в схемі регістра може бути відсутньою схема формування функцій збудження?
30. Поясніть наявність ліній зворотного зв'язку у складі регістра.
31. В яких випадках в схемі регістра може бути відсутніми лінії зворотного зв'язку з виходів базових тригерів на вхід комутатора?
32. Що називається двофазною системою синхронізації регістра?
33. За яких умов може використовуватися двофазна система синхронізації регістра?
34. Приведіть послідовність кроків для виконання синтезу багатофункціонального регістра з некерованою синхронізацією.
35. Схеми яких складових регістра необхідно синтезувати?
36. Якими способами можна виконати синтез комутатора?
37. Як виконати синтез комутатора за допомогою таблиці істинності?

38. Який розмір таблиці істинності необхідно використовувати, якщо регістр є пристроєм з некерованою синхронізацією?
39. Чи достатньо виконувати синтез тільки одного розряду комутатора при використанні таблиці істинності? Обґрунтуйте відповідь.
40. Які недоліки використання способу синтезу комутатора за допомогою таблиці істинності?
41. Яким обсягом обмежується таблиця істинності при виконанні «ручного» метода синтезу?
42. Як виконати синтез комутатора за допомогою таблиці мікрооперацій, враховуючи використання некерованої синхронізації?
43. Як визначити кількість рядків в таблиці мікрооперацій регістра з некерованою синхронізацією?
44. Як визначити кількість стовпців в таблиці мікрооперацій?
45. Поясніть, як визначити вміст поточної клітини таблиці мікрооперацій?
46. Яка мікрооперація виконується за наявності активного значення сигналу *nU*, якщо цей сигнал використовується при синтезі комутатора?
47. Яка мікрооперація виконується в регістрі за наявності активного значення сигналу *nU*, якщо цей сигнал використовується при синтезі *СФФЗ*?
48. Як визначити логічні вирази для кожного виходу комутатора у вигляді ДНФ?
49. Яким чином реалізується схема керування *СКБТ*?
50. Яким чином буде функціонувати схема регістра, представлена на рис.2.21, якщо сигнал *G* безпосередньо надходить до входів синхронізації тригерів?

51. Яким чином буде функціонувати схема регістра, представлена на рис.2.28, якщо сигнал *G* безпосередньо надходить до входів синхронізації тригерів?
52. Поясніть вираз (2.14).
53. Поясніть вирази (2.15), (2.16).
54. Поясніть термін «активне значення сигналу синхронізації».
55. Яким способом виконується реалізація некерованої синхронізації, якщо базовим тригером регістра є *RCS*-тригер? Обґрунтуйте відповідь.
56. Яким способом виконується реалізація некерованої синхронізації, якщо в якості базового тригера регістра використовується *TC*- тригер? Обґрунтуйте відповідь.
57. Яким способом виконується реалізація некерованої синхронізації, якщо в якості базового тригера регістра використовується *DV*- тригер? Обґрунтуйте відповідь.
58. Поясніть, як заповнюється таблиця 2.15?
59. Чи можуть виникати гонки сигналів в регістрах з некерованою синхронізацією? Обґрунтуйте відповідь.
60. Які вимоги висуваються до часової діаграми сигналів *G* і u_i?
61. Розробіть таблицю переходів для врахування сигналу *nU* схемою *СФФЗ* в регістрі з некерованою синхронізацією, якщо базовим тригером регістра є *RCS*-тригер.
62. Розробіть таблицю переходів для врахування сигналу *nU* схемою *СФФЗ* в регістрі з некерованою синхронізацією, якщо базовим тригером регістра є *D*-тригер.
63. Розробіть таблицю переходів для врахування сигналу *nU* схемою *СФФЗ* в регістрі з некерованою синхронізацією, якщо базовим тригером регістра є *TC*-тригер.

64. Розробіть таблицю переходів для врахування сигналу *nU* схемою *СФФ3* в регістрі з некерованою синхронізацією, якщо базовим тригером регістра є *DV*-тригер.
65. Як визначити динамічні параметри регістра з некерованою синхронізацією?
66. Як визначити значення мінімального періоду надходження сигналу *G*?
67. Яким чином функціонування схеми формування функцій збудження залежить від мікрооперацій, які виконує регістр?
68. Поясніть заповнення таблиці 2.16.
69. Поясніть заповнення колонок 6-8 в таблиці 2.17.
70. Для якого входу базового тригера спочатку визначається функція збудження?
71. Який спосіб завантаження даних за допомогою зсувів використовується в сучасних комп'ютерних системах?
72. Як зміниться значення числа, зсунутого на один розряд вправо за допомогою арифметичного зсуву?
73. Як зміниться значення числа, зсунутого на один розряд вліво за допомогою арифметичного зсуву?
74. Поясніть вираз (2.17).
75. Які регістри називаються реверсивними?
76. На базі *D*-тригера виконати синтез 4-бітного регістра з некерованою синхронізацією в будь-якому базисі логічних елементів. Регістр виконує такі мікрооперації: *uA*: *Rg* := *A*; *uR*: *Rg* := *11.R2*(*Rg*) ; *uL*: *Rg* := *L1.1*(*Rg*). Наявність сигналу *nU* врахувати в комутаторі.
77. На базі *D*-тригера виконати синтез 4-бітного регістра з некерованою синхронізацією в будь-якому базисі логічних елементів. Регістр виконує такі мікрооперації: *uA*: *Rg* := *A*; *uR*: *Rg* := *10.R2*(*Rg*) ; *uL*: *Rg* := *L1.0*(*Rg*). Наявність сигналу *nU* врахувати в *СФФ3*.

78. На базі *RCS*-тригера виконати синтез 4-бітного регістра з некерованою синхронізацією в будь-якому базисі логічних елементів. Регістр виконує такі мікрооперації: *uA*: $Rg := \overline{A}$; *uR*: $Rg := CR2(Rg)$; *uL*: $Rg := AL1.0(Rg)$.
79. На базі *JK*-тригера виконати синтез 4-бітного регістра з некерованою синхронізацією в базисі Шефера. Регістр виконує такі мікрооперації: *uA*: $Rg := \overline{A}$; *uR*: $Rg := CL1(Rg)$; *uL*: $Rg := AR3(Rg)$.
80. На базі *TC*-тригера виконати синтез 4-бітного регістра з некерованою синхронізацією в будь-якому базисі логічних елементів. Регістр виконує такі мікрооперації: *uA*: $Rg := A$; *uR*: $Rg := L2.10(Rg)$; *uL*: $Rg := CR4(Rg)$.
81. Чи можна використовувати прозорі тригери в зсувних регістрах? Обґрунтуйте відповідь.
82. Чи можна використовувати асинхронні входи базових тригерів для організації паралельного завантаження інформації в регістр? Обґрунтуйте відповідь.
83. Поясніть принцип роботи пристрою, що перетворює паралельний код в послідовний.
84. Виконати синтез пристрою для перетворення паралельного коду в послідовний.
85. Поясніть принцип роботи пристрою, що перетворює послідовний код в паралельний.
86. Виконати синтез пристрою для перетворення послідовного коду в паралельний.
87. Чому при реалізації регістрів не доцільно використовувати *TC*- тригери?
88. Яким чином виконується завантаження даних в регістр при використанні двотактового приймання інформації?
89. Які дії виконуються під час першого мікротакту?
90. Які дії виконуються під час другого мікротакту?

91. Переваги двотактового способу завантаження інформації.
92. Недоліки двотактового способу завантаження інформації.
93. В яких випадках не можна використовувати двотактовий спосіб завантаження інформації?
94. Поясніть принцип роботи схеми, приведеної на рис.2.43.
95. Поясніть, чому в схемі, приведеній на рис.2.43, $K = 0$.
96. Приведіть схему реалізації двотактового приймання інформації на базі *RCS*-тригера.
97. Приведіть схему реалізації двотактового приймання інформації на базі *TC*-тригера.
98. Чи можна реалізувати двотактове завантаження інформації в регістр на базі *D*-тригера? Обґрунтуйте відповідь.
99. В яких інтегральних схемах регістрів використовується некерована синхронізація?
100. Яким чином по схемі визначити, до якого типу керування синхронізацією відноситься регістр?
101. Розробіть 4-розрядний регістр з некерованою синхронізацією та двотактовим завантаженням інформації на базі *D*-тригера, який виконує мікрооперації *uA*: *Rg* := *A*; *uB*: *Rg* := *B*.
102. Розробіть 4-розрядний регістр з керованою синхронізацією та двотактовим завантаженням інформації на базі *D*-тригера, який виконує мікрооперації *uA*: *Rg* := *A*; *uB*: *Rg* := *B*.
103. Розробіть 4-розрядний регістр з некерованою синхронізацією та двотактовим завантаженням інформації на базі *SCR*-тригера, який виконує мікрооперації *uA*: *Rg* := *A*; *uB*: *Rg* := *B*.
104. Розробіть 4-розрядний регістр з керованою синхронізацією та двотактовим завантаженням інформації на базі *SCR*-тригера, який виконує мікрооперації *uA*: *Rg* := *A*; *uB*: *Rg* := *B*.

105. Розробіть 4-розрядний регістр з некерованою синхронізацією та двотактовим завантаженням інформації на базі *TC*-тригера, який виконує мікрооперації *uA*: *Rg* := *A*; *uB*: *Rg* := *B*.
106. Розробіть 4-розрядний регістр з керованою синхронізацією та двотактовим завантаженням інформації на базі *TC*-тригера, який виконує мікрооперації *uA*: *Rg* := *A*; *uB*: *Rg* := *B*.

2.2 Регістри на базі регістрів

Під час проєктування цифрових пристроїв часто виникає потреба виконувати синтез регістрів не на основі окремих базових тригерів, а з використанням вже готових регістрів, виконаних у вигляді інтегральних схем (ІС), або, застосовуючи програмні моделі регістрів у складі бібліотек цифрових компонентів систем автоматизованого проєктування (САПР), що підтримують *VHDL*- і *FPGA*-технології.

2.2.1 Регістри в інтегральному виконанні

Для того, щоб виконувати синтез регістрів на базі інтегральних схем розглянемо різні способи реалізації регістрів у вигляді ІС на прикладі сімейства елементів транзисторно-транзисторної логіки (ТТЛ, TTL) на базі біполярних транзисторів.

Серія ІС за TTL-технологією широко представлена серією цифрових інтегральних схем *SN54/74* (*74xx*, *54xx*). Фірма *Texas Instruments* розробила цю серію цифрових мікросхем, а зараз кілька виробників виготовляють ці ІС, використовуючи однакову систему позначень.

Серії *74xx* і *54xx* у функціональному сенсі є однаковими, але розрізняються застосуванням в різних температурних діапазонах. Для комерційної серії *74xx* припустимим діапазоном є температурний інтервал від 0ºС до +70ºС, а для серії *54xx*, яка вважається для військового

використання: від -55ºС до +125ºС (наприклад, [7]). Таким чином, далі достатньо буде розглядати ІС серії *SN74*.

Окрім ІС ТТЛ у складі серії *SN54/74* випускаються також мікросхеми по *КМОН* (*CMOS*)-технології (*КМОН* – комплементарні транзистори за структурою метал-оксид-напівпровідник; *CMOS – Complementary-metal-oxide-semiconductor*), що означає використання електронної схеми, в якій застосовуються *МОН*-транзистори обох типів провідності каналу.

Використовуються також *BiCMOS*-технологія (*bipolar complementary metal-oxide-semiconductor*) – технологія, за якою в ІС використовуються одночасно біполярні та КМОН-транзистори.

Для позначення інтегральних схем серії *SNxx* використовуються шість складових елементів, які приведені нижче:

- назва фірми: *SN* (в деяких випадках може бути відсутня);
- номер сімейства: 54 або 74;
- код серії: до трьох символів (в тому числі жодного);
- ідентифікатор спеціального типу (може бути відсутнім);
- тип мікросхеми (від двох до шести символів);
- код типу корпусу (може бути відсутнім).

В якості коду серії використовуються такі позначення:

- відсутність різновиду означає стандартні ТТЛ, які свого часу були виготовлені першими;
- *L* – ІС з малим (*Low*) значенням споживаної потужності: для ТТЛ;
- *S* – ІС з діодами Шоттки (*Schottky*): для ТТЛ;
- *LS* – ІС з діодами Шоттки та малим значенням споживаної потужності (*Low Power Schottky*) : для ТТЛ;
- *ALS* – поліпшені (*Advanced Low Power Schottky*) ІС: для ТТЛ;
- *F* – швидкісні (*Fast*) ІС: для ТТЛ;
- *HC* – швидкісні (*High Speed*): для КМОН;
- *HCT* – швидкісні, сумісні з ТТЛ (*High Speed with TTL inputs*): для КМОН;

- *AC* – покращені швидкісні ІС (*Advanced High Speed*): для КМОН;
- *ACT* – покращені швидкісні ІС, сумісні з ТТЛ (*Advanced High Speed with TTL inputs*): для КМОН;
- *BCT – BiCMOS Technology*: для КМОН;
- *LVT* – ІС з низькою напругою живлення (*Low Voltage Technology*): для КМОН;
- *ABT* – покращенні ІС (*Advanced BiCMOS Technology)*: для КМОН;

Код типу корпусу позначається за допомогою одного або двох символів: *D*, *DB*, *PW*, *FK*, *W*, *N*, *J*, *T* тощо.

Наприклад, позначення інтегральних схем: *SN7400*, *SN54154*, *SN74S194*, *SN74LS194A*, *SN74ALS273*, *SN74ACT32*, *SN74ACT32N*, *SN74ACT32FK*.

Далі розглянемо загальні особливості побудови інтегральних схем регістрів.

До складу серій *SN54/74* входить достатньо велика кількість регістрів. Так, тільки на основі ТТЛ можна використовувати більше 20 типів різних регістрів. Далі для скорочення будемо вказувати позначення ІС *SN54/74* без коду серії.

Всі регістри в інтегральному виконанні можна умовно розділити на чотири групи:

1. Регістри, які виконують завантаження інформації тільки з зовнішніх джерел. Наприклад, ІС *SN74175*, *SN74174*, *SN74173*, *SN74373* та інші.
2. Регістри, які виконують завантаження інформації та зсув в одному напрямку. Наприклад, ІС *SN7495*, *SN74164*, *SN74165*, *SN74195* та інші.
3. Регістри, які виконують завантаження інформації та зсув в обох напрямках (реверсивні зсувні регістри). Наприклад, ІС *SN74194*, *SN74198*, *SN74299* та інші.
4. Регістри в складі інших цифрових пристроїв, наприклад, у складі лічильників. Наприклад, ІС *SN74192*, *SN74193*, *SN74160*, *SN74161*, *SN74168* та інші.

Синтез регістрів першої групи виконується таким же чином, як і регістри на базі окремих тригерів.

Далі розглянемо синтез регістрів другої і третьої груп.

2.2.2 Синтез зсувних регістрів на базі інтегральних схем регістрів

Процедура синтезу регістрів на основі регістрів виконується аналогічно порядку синтезу регістрів на основі окремих тригерів.

Як правило, в регістрах у вигляді інтегральних схем або у складі бібліотеки елементів САПР в якості базових тригерів використовуються *D*- тригери. Це означає, що виходи комутатора підключаються безпосередньо до входів *D* цих тригерів, тобто *СФФЗ* в таких регістрах відсутня.

Для синтезу регістрів в цьому випадку використовується об'єднана таблиця мікрооперацій, яка одночасно містить інформацію для синтезу схеми керування та комутатора.

Регістр, що синтезується, будемо називати підсумковим регістром (ПР), а готові регістри, які використовуються для проєктування підсумкового регістра, будемо називати базовими регістрами (БР).

Методика синтезу підсумкового регістра полягає у виконанні такої послідовності дій:

1. Аналіз принципів роботи базового регістра.
2. Розрахунок кількості БР.
3. Заповнення об'єднаної таблиці мікрооперацій.
4. Визначення функцій збудження БР.
5. Мінімізація функцій збудження.
6. Перетворення логічних виразів в заданий базис.
7. Розробка функціональної схеми ПР.
8. Перевірка коректності функціонування ПР.

Далі розглянемо зазначені вище кроки для проведення синтезу регістра більш детально.

До початку проєктування (крок 1 синтезу) необхідно визначитися з типом базового регістра. Опис базових регістрів можна знайти в будь-якому довіднику з цифрових інтегральних схем, наприклад, [6] або на сайтах фірм-виробників, наприклад, [5]. Таким чином, необхідно мати інформацію про функціональний опис регістра, його таблицю переходів, логічну схему, значення динамічних параметрів (затримки спрацьовування), електронні параметри (наприклад, споживану потужність) тощо.

Після вибору типу базового регістра необхідно визначити кількість цих регістрів k_{BR} у складі підсумкового регістра за формулою (крок 2 процедури синтезу):

$$k_{BR} = \left] \frac{n_R}{n_B} \right[, \qquad (2.18)$$

де $n_R(n_B)$ – розрядність відповідно підсумкового (базового) регістра;

] [– операція округлення до найближчого більшого цілого числа в разі отримання дробового результату.

На третьому кроці відбувається розробка об'єднаної таблиці мікрооперацій.

Об'єднана таблиця мікрооперацій (далі – таблиця мікрооперацій) може містити, як і раніше, *k* або *k+1* рядків для регістрів з керованою або некерованою синхронізацією відповідно, де *k* визначається кількістю мікрооперацій, що виконує регістр.

Кількість колонок таблиці мікрооперацій відповідає загальній кількості входів всіх базових регістрів, які використовуються в регістрі, що синтезується, тобто кількість стовпців в таблиці $k_{ст}$ визначається за виразом $k_{ст} = k_{вх} \cdot k_{BR}$, де $k_{вх}$ – кількість входів одного базового регістра, а k_{BR} – кількість базових регістрів відповідно до (2.18).

Заповнення таблиці починається з керуючих входів регістра і виконується аналогічно тому, як це робилося під час синтезу регістрів на базі тригерів. Більш детально заповнення таблиці мікрооперацій буде розглядатися далі під час виконання прикладів з синтезу регістрів.

Виконання кроків 4-7 нічим принципово не відрізняється від синтезу регістрів на базі тригерів.

2.2.3 Приклади синтезу регістрів з керованою синхронізацією на базі ІС регістрів

Приклад 2.5. На базі ІС *SN74195* виконати синтез 8-бітного реверсивного регістра з керованою синхронізацією в будь-якому базисі логічних елементів. Регістр виконує такі мікрооперації: *uR*: *Rg* := *AR1*(*Rg*) ; *uL*: *Rg* := *CL1*(*Rg*); *uA*: *Rg* := *A*. Структурна схема регістра приведена на рис.2.27.

Розв'язок

Синтез регістра також будемо виконувати відповідно до порядку, приведеному вище в підрозділі 2.2.2.

На першому кроці синтезу проведемо аналіз функціонування підсумкового регістра під час виконання заданих мікрооперацій.

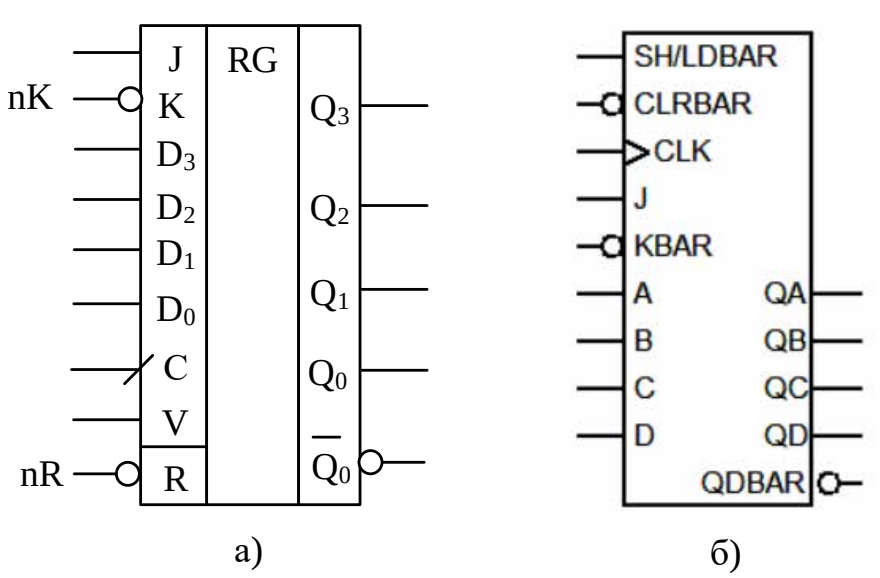
З умови завдання можна побачити, що регістр виконує три мікрооперації:

- арифметичний зсув вмісту регістра на один розряд вправо за сигналом *uR* (*Rg* := *AR1*(*Rg*));
- циклічний зсув вмісту регістра вліво на один розряд за сигналом *uL* (*Rg* := *CL1*(*Rg*));
- паралельне завантаження інформації за сигналом *uA* (*Rg* := *A*).

За відсутності активних значень керуючих сигналів вміст регістра не повинен змінюватися, тобто необхідно забезпечити режим збереження інформації.

Крім того, необхідно провести аналіз функціонування заданого базового регістра *SN74195*. Опис цього регістра можна отримати в будь-якому довіднику з цифрових схем або на будь-якому сайті з інформацією від виробника, наприклад [8].

Умовне графічне позначення регістра *SN74195* приведено на рис.2.45, де УГП на рис.2.45,а приведено за вітчизняними стандартами, а на рис.2.45,б – відповідно до позначення цього регістра в середовищі MicroCap.



а) б)

Рисунок 2.45 – УГП регістра *SN74195*

З УГП можна побачити, що базовий регістр є 4-розрядним.

Призначення виводів регістра *SN74195* приведено в табл.2.18 [8].

Таблиця 2.18 – Призначення виводів ІС *SN74195*

Виводи на рис.2.45,а	Виводи на рис.2.45,б	Призначення
V	SH/LDBAR	Вхід вибору режиму
D_3, D_2, D_1, D_0	A,B,C,D	Входи паралельного завантаження
C	CLK	Загальний вхід синхронізації
R	CLRBAR	Асинхронний вхід для скиду регістра
J, K	J, KBAR	Входи послідовного завантаження
Q_3, Q_2, Q_1, Q_0	QA, QB, QC, QD	Виходи регістра
$\overline{Q_0}$	QDBAR	Інверсний вихід молодшого розряду

Розглянемо більш детально призначення виводів регістра.

Виводи Q_3, Q_2, Q_1, Q_0 (*QA*, *QB, QC, QD*) є виходами регістра і визначають його стан в будь-який момент часу.

Вхід *V* (*SH/LDBAR*) називається входом вибору режиму (***Mode Select Input***). Позначення цього входу в різних джерелах може відрізнятися від зазначених. Наприклад, в [8] цей вхід називається $\overline{PE}$ (***Parallel Enable***). Вхід вибору режиму використовується для налаштування регістра на той чи інший режим функціонування, тобто значення сигналу на цьому вході забезпечує виконання однієї з мікрооперацій, які може виконувати регістр. В зв'язку з тим, що регістр має один такий вхід, то цей регістр може виконувати одну з двох мікрооперацій.

Відповідно до [8] низький рівень сигналу (***Low Voltage Level***) на цьому вході налаштовує регістр на виконання паралельного завантаження інформації з входів D_3, D_2, D_1, D_0 (*A*, *B*, *C*, *D*) в базові тригери регістра, в результаті чого виходи регістра Q_3, Q_2, Q_1, Q_0 приймають стан сигналів з інформаційних входів, тобто $Q_i = D_i$, $i = 0, \ldots, 3$; (*QA* = *A*; … *QD* = *D*).

Нагадаємо, що низький рівень сигналу при використанні позитивної логіки (***Positive Logic***) відповідає логічному нулю, а високий рівень сигналу – логічній одиниці, що далі будемо використовувати за замовчуванням.

Поява високого рівня сигналу (***High Voltage Level***) на вході *V* налаштовує регістр на виконання мікрооперації зсуву вправо на один розряд. Для забезпечення реалізації будь-якого типу зсуву (логічний, циклічний або арифметичний) при виконанні цієї мікрооперації в регістрі використовуються входи послідовного завантаження *J* і *K*, інформація з яких записується в старший розряд регістра. При цьому вхід *J* є прямим [1], а вхід *K* – інверсним [1], тому сигнал на вході *K* будемо позначати *nK*. Таким чином, при виконанні мікрооперації зсуву вправо вміст регістра зсувається в напрямку $Q_3 \rightarrow Q_2 \rightarrow Q_1 \rightarrow Q_0$ (*QA*→ *QB*→ *QC*→*QD*). При цьому в старший розряд регістра Q_3(*QA*), що звільнився, записується інформація відповідно до стану сигналів *J* і *nK*. Інформація про переключення старшого розряду при зсуві в залежності від сигналів *J* і *nK* приведена в табл.2.19.

Таблиця 2.19 – Вплив сигналів *J*, *nK* на стан Q_3(*QA*)

J	*nK* (*KBAR*)	Стан Q_3 (*QA*)
L (0)	*L (0)*	Нуль
L (0)	*H (1)*	Збереження стану
H (1)	*L (0)*	Переключення в протилежний стан
H (1)	*H (1)*	Одиниця

Вхід *C*(*CLK*) є загальним для всіх базових тригерів входом синхронізації, а переключення регістра відбувається за переднім фронтом цього сигналу.

Вхід *R* (*CLRBAR*) застосовується для встановлення всіх базових тригерів регістра в нульовий стан, що використовується, наприклад, для ініціалізації регістра на початку його роботи в складі певного цифрового пристрою. Вхід *R* є асинхронним та інверсним (активне значення – нуль), тому будемо позначати сигнал на цьому вході *nR*.

Вихід $\overline{Q}_0$ (*QDBAR*) – інверсний вихід молодшого розряду регістра. Цей вихід може застосовуватися для передачі стану Q_0 парафазним кодом або при використанні кількох базових регістрів для налаштування старшого розряду наступного регістра на роботу в якості Т-тригера (вихід Q_0 підключається до входу *J*, а вихід $\overline{Q}_0$ – до інверсного входу *K* наступного регістра).

На рис.2.46 приведена функціональна схема ІС регістра *SN74195* [8]. На схемі видно, що в мікросхемі в якості базових тригерів використовуються *RCS*-тригери, які в регістрі перетворюються на *D*-тригери (виходи $Q_1 – Q_3$), а тригер Q_0 перетворений на *JK*-тригер. Процедура синтезу одного тригера на базі іншого приведена в розділі 1 цього посібника. Крім того, в схемі регістра з [8] використовується зворотна нумерація розрядів (Q_0 є найстаршим, а Q_3 – наймолодшим), хоча користувач може використовувати будь-яку нумерацію розрядів: важливо, щоб при зсуві вправо тригер, до якого підключаються входи *J* і *K*, був найстаршим, а тригер з інверсним виходом – наймолодшим).

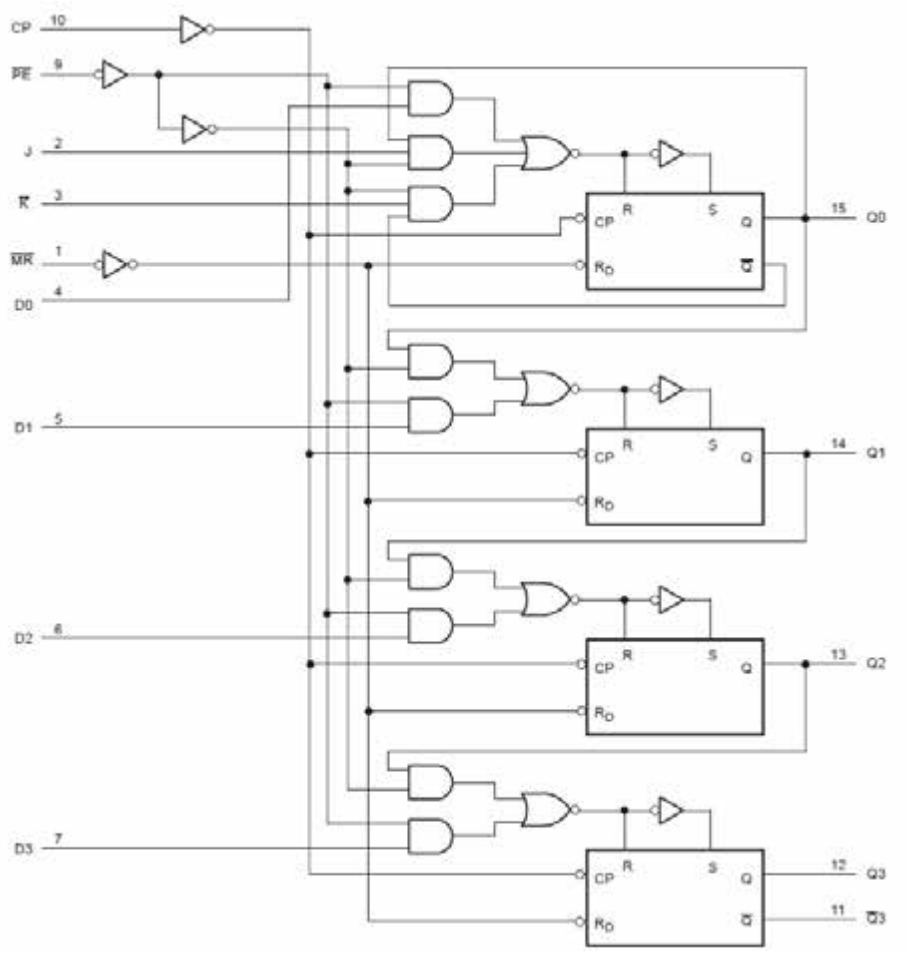


Рисунок 2.46 – Функціональна схема базового регістра *SN74195*

Таким чином, виконано аналіз функціонування заданого базового регістра, в результаті якого визначена розрядність цього регістра, призначення його входів і виходів, режими роботи, тобто перший крок процедури синтезу підсумкового регістра завершено.

На другому кроці процедури синтезу визначимо кількість базових регістрів, необхідних для реалізації підсумкового регістра. Відповідно до (2.18), враховуючи, що згідно з умовою завдання $n_R = 8$, а $n_B = 4$, отримаємо, що кількість базових регістрів $k_{BR} = 2$.

Далі (крок 3) виконаємо розробку об’єднаної таблиці мікрооперацій. З умови завдання випливає, що регістр виконує три мікрооперації, тобто кількість рядків таблиці дорівнює трьом.

Відповідно до рис.2.45 кількість входів одного базового регістра складає 9, а кількість цих регістрів дорівнює двом, тобто кількість стовпців

таблиці мікрооперації дорівнює 18. Підготовлена до заповнення таблиця мікрооперацій для синтезу регістра приведена в табл.2.20.

Таблиця 2.20 – Загальний вигляд таблиці мікрооперацій (приклад 2.5)

МО	Старша тетрада									Молодша тетрада								
	J_c	nK_c	D_{3c}	D_{2c}	D_{1c}	D_{0c}	C_c	V_c	nR_c	$J_м$	$nK_м$	$D_{3м}$	$D_{2м}$	$D_{1м}$	$D_{0м}$	$C_м$	$V_м$	$nR_м$
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
uA																		
uR																		
uL																		
Номер виходу			7	6	5	4						3	2	1	0			

Відповідно до попереднього кроку синтезу отримано, що підсумковий регістр складається з двох базових 4-бітних регістрів, тобто кожний базовий регістр є тетрадою підсумкового регістра. Для того, щоб відрізняти ці регістри, один з них будемо вважати старшою тетрадою (в табл.2.20 входи цього регістра позначений індексом «*с*»), а інший – молодшою тетрадою входи цього регістра позначений індексом «*м*». Крім того, будемо також використовувати наскрізну нумерацію розрядів підсумкового регістру (рядок «Номер виходу» в табл.2.20), тобто старша тетрада відповідає розрядам 7-4 підсумкового регістра (колонки 4-7 в таблиці 2.20), а молодша – розрядам 3- 0 (колонки 13-16 в таблиці 2.20).

Далі необхідно виконати заповнення відповідного рядку таблиці мікрооперацій для кожної мікрооперації (колонка 1) окремо.

Заповнення клітин таблиці необхідно починати з входів керування регістром.

Розглянемо заповнення таблиці для мікрооперації *uA*. Відповідно до умови завдання за сигналом *uA* відбувається паралельне завантаження інформації з шини *A*. Для забезпечення виконання завантаження необхідно сформувати сигнал низького рівня (0) на входи V_c і $V_м$ базових регістрів (колонки 9, 18). При цьому на входах nR_c і $nR_м$ (колонки 10, 19) сформуємо неактивний сигнал (1). На входах C_c і $C_м$ (колонки 8, 17) необхідно

забезпечити надходження сигналу *uA*, що відмічається у відповідних клітинах. Таким чином, заповнення таблиці для керуючих входів завершено, а далі заповнюємо клітини для інформаційних входів. В зв'язку з тим, що для виконання мікрооперації *uA*, регістр налаштовується на паралельне завантаження ($V_c = V_м = 0$), входи J_c, nK_c, $J_м$, $nK_м$ (колонки 2, 3, 11, 12) в цьому режимі не впливають на роботу регістра, тому у відповідних клітинах записується символ «*». І нарешті, заповнюємо клітини для входів паралельного завантаження старшої (колонки 4-7) і молодшої (колонки 13- 16) тетрад регістра. В старшу тетраду завантажується вміст старших чотирьох розрядів шини даних *A* (a_7-a_4), а в молодшу – вміст молодших чотирьох розрядів цієї шини (a_3-a_0).

В результаті таблиця для виконання мікрооперації *uA* заповнена (табл.2.21), що виділено сірим кольором.

Таблиця 2.21 – Таблиця мікрооперацій для сигналу *uA* (приклад 2.5)

МО	Старша тетрада									Молодша тетрада								
	Jc	nK_c	D_{3c}	D_{2c}	D_{1c}	D_{0c}	C_c	V_c	nR_c	$J_м$	$nK_м$	$D_{3м}$	$D_{2м}$	$D_{1м}$	$D_{0м}$	$C_м$	$V_м$	$nR_м$
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
uA	*	*	a_7	a_6	a_5	a_4	*uA*	*0*	*1*	*	*	a_3	a_2	a_1	a_0	*uA*	*0*	*1*
uR																		
uL																		
Номер виходу			7	6	5	4						3	2	1	0			

Далі заповнимо таблицю мікрооперацій для сигналу *uR*.

Знову ж таки починаємо заповняти таблицю, починаючи з входів керування регістром.

Відповідно до умови завдання за сигналом *uR* відбувається арифметичний зсув вмісту регістра вправо на один розряд. Для забезпечення виконання цієї мікрооперації необхідно сформувати сигнал високого рівня (1) на входи V_c і $V_м$ базових регістрів. В цьому випадку в базовому регістрі активізуються кола зсуву, а входи паралельного завантаження відключаються від активних кіл та на процес зсуву не мають впливу.

Як і для попередньої мікрооперації, на входах nR_c і $nR_м$ формується неактивний сигнал, а на входи C_c і $C_м$ – забезпечується надходження сигналу *uR*. Далі заповнюємо клітини таблиці для інформаційних входів. Як вже було зазначено вище, при налаштуванні базового регістра на виконання зсуву вправо на 1 біт входи паралельного завантаження відключаються і не використовуються регістром в цьому режимі, тобто у відповідних клітинах записується символ «*».

Розглянемо реалізацію заданого зсуву в підсумковому регістрі. При налаштуванні на зсув ($V_c = V_м = 1$) в базових регістрах відбувається зсув вправо за допомогою внутрішніх кіл цих регістрів ($Q_7 \rightarrow Q_6 \rightarrow Q_5 \rightarrow Q_4$, $Q_3 \rightarrow Q_2 \rightarrow Q_1 \rightarrow Q_0$), тобто вміст регістру зсувається в розряди 6-4 і 2-0. Таким чином, для виконання зсуву ще необхідно записувати певну інформацію в старші біти кожної тетради (Q_7 і Q_3), що виконується за допомогою входів J_c, nK_c, $J_м$, $nK_м$. Вміст старшого розряду регістра (Q_7) визначається типом зсуву. Відповідно до умови завдання, виконується арифметичний зсув вправо на один розряд, тобто значення старшого розряду регістра не змінюється. Це можна реалізувати, наприклад, за рахунок перезапису стану старшого розряду в цей же розряд або налаштування цього розряду на збереження стану. В старший же розряд молодшої тетради (Q_3) записується вміст молодшого розряду попередньої тетради (Q_4). Таким чином, входи J_c, nK_c, використовуються для реалізації заданого типу зсуву, а входи $J_м$, $nK_м$ – для передачі інформації між тетрадами.

Передача інформації між розрядами регістра при виконанні арифметичного зсуву приведена на рис.2.47 (зовнішні з’єднання показані пунктиром).

Заповнена таблиця мікрооперацій для виконання арифметичного зсуву вправо на один розряд приведена в табл.2.22.

Далі розглянемо заповнення клітин таблиці МО для виконання мікрооперації *uL*.

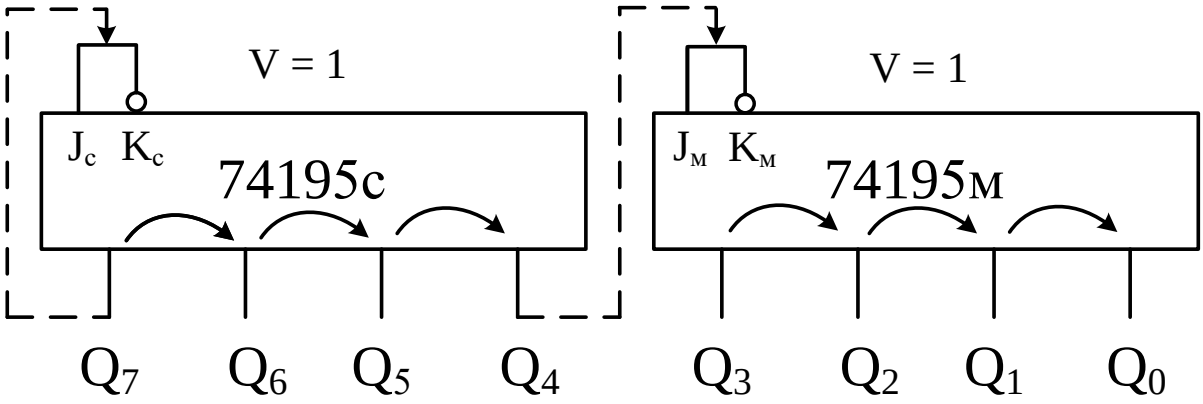


Рисунок 2.47 – Виконання арифметичного зсуву вправо на один розряд на базі регістрів *SN74195*

Таблиця 2.22 – Таблиця мікрооперацій для сигналу *uR* (приклад 2.5)

МО	Старша тетрада									Молодша тетрада								
	Jc	nK_c	D_{3c}	D_{2c}	D_{1c}	D_{0c}	C_c	V_c	nR_c	$J_м$	$nK_м$	$D_{3м}$	$D_{2м}$	$D_{1м}$	$D_{0м}$	$C_м$	$V_м$	$nR_м$
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
uA																		
uR	Q_7	Q_7	*	*	*	*	*uR*	1	1	Q_4	Q_4	*	*	*	*	*uR*	1	1
uL																		
Номер виходу			7	6	5	4						3	2	1	0			

Відповідно до завдання третя мікрооперація відповідає виконанню циклічного зсуву вліво на один розряд. Однак згідно опису функціонування базовий регістр не може виконувати зсув вліво за рахунок внутрішніх кіл. В цьому випадку реалізація зсуву виконується за допомогою зовнішніх кіл, тобто базові регістри налаштовуються на виконання мікрооперації паралельного завантаження ($V_c = V_м = 0$). При цьому заповнення таблиці мікрооперацій для інформаційних входів виконується за аналогією з синтезом комутатора регістра на базі тригерів (див., наприклад, табл.2.11). Заповнення клітин для керуючих входів виконується аналогічно попереднім мікрооперацям.

Заповнена таблиця мікрооперацій для виконання циклічного зсуву вліво на один розряд приведена в табл.2.23.

Таблиця 2.23 – Таблиця мікрооперацій для сигналу *uL* (приклад 2.5)

МО	Старша тетрада									Молодша тетрада								
	Jc	nK_c	D_{3c}	D_{2c}	D_{1c}	D_{0c}	C_c	V_c	nR_c	$J_м$	$nK_м$	$D_{3м}$	$D_{2м}$	$D_{1м}$	$D_{0м}$	$C_м$	$V_м$	$nR_м$
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
uA																		
uR																		
uL	*	*	Q_6	Q_5	Q_4	Q_3	*uL*	0	1	*	*	Q_2	Q_1	Q_0	Q_7	*uL*	*0*	*1*
Номер виходу			7	6	5	4						3	2	1	0			

В результаті повна таблиця мікрооперацій підсумкового регістра приведена в табл.2.24.

Таблиця 2.24 – Таблиця мікрооперацій для синтезу регістра (приклад 2.5)

МО	Старша тетрада									Молодша тетрада								
	Jc	nK_c	D_{3c}	D_{2c}	D_{1c}	D_{0c}	C_c	V_c	nR_c	$J_м$	$nK_м$	$D_{3м}$	$D_{2м}$	$D_{1м}$	$D_{0м}$	$C_м$	$V_м$	$nR_м$
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
uA	*	*	a_7	a_6	a_5	a_4	*uA*	*0*	*1*	*	*	a_3	a_2	a_1	a_0	*uA*	*0*	*1*
uR	Q_7	Q_7	*	*	*	*	*uR*	*1*	*1*	Q_4	Q_4	*	*	*	*	*uR*	*1*	*1*
uL	*	*	Q_6	Q_5	Q_4	Q_3	*uL*	*0*	*1*	*	*	Q_2	Q_1	Q_0	Q_7	*uL*	*0*	*1*
Номер виходу			7	6	5	4						3	2	1	0			

Таким чином, третій крок процедури синтезу регістра закінчено.

Після цього визначимо функції збудження для кожного входу базового регістра (четвертий крок синтезу регістра) аналогічно правилам, приведеним в п. 2.1.1.1.

В результаті отримаємо наступні функції збудження:

- для входів керування:

$$nR_c = nR_м = 1; \quad C_c = C_м = (uA \vee uR \vee uL)\cdot G; \quad V_c = V_м = uR; \tag{2.19}$$

- для інформаційних входів:

$$D_{ic} = a_{i+4} \cdot uA \vee Q_{i+3} \cdot uL \ (i = 0, \dots, 3);$$

$$D_{iм} = a_i \cdot uA \vee Q_{i-1} \cdot uL \ (i = 1, \dots, 3); \ D_{0м} = a_0 \cdot uA \vee Q_7 \cdot uL; \tag{2.20}$$

$$J_c = nK_c = Q_7 \cdot uR; \ J_м = nK_м = Q_4 \cdot uR;$$

Фактично функції збудження для інформаційних входів D_{3c} - D_{0c} і $D_{3м}$ - $D_{0м}$ відповідно до виразів (2.20) реалізують комутатор підсумкового регістра, а вирази (2.19) – схему керування.

Для входів, які приймають участь тільки в одній мікрооперації, функцію збудження можна спростити, видаливши з логічного виразу керуючий сигнал (п'ятий крок процедури синтезу).

З таблиці мікрооперацій випливає, що входи J_c, nK_c, $J_м$, $nK_м$ використовуються тільки в мікрооперації *uR*, тому логічні вирази для цих входів можна спростити:

$$J_c = nK_c = Q_7 \; ; \; J_м = nK_м = Q_4. \qquad (2.21)$$

Відповідно до умови завдання логічні вирази реалізуються в будь- якому базисі, тому залишимо їх без перетворень.

Логічна схема підсумкового регістра приведена на рис.2.48.

Схема для моделювання підсумкового регістра на основі базових регістрів *SN74195* в середовищі MicroCap приведена на рис.2.49.

Для імітації вмісту вхідної шини *A* в схемі на рис.2.49 використовуються вісім елементів ***Fixed Digital*** (наприклад, другі входи логічних елементів *U33*, *U38* та інших), які задають постійне двійкове значення певного сигналу на весь період моделювання. Зі схеми можна побачити, що елементи ***Fixed Digital*** утворюють двійковий код *10001101* за допомогою формування функцій збудження D_{3c}, …, D_{0c} і $D_{3м}$, …, $D_{0м}$. В результаті цей код за сигналом *uA* завантажується в базові регістри.

На вхід *R* базових регістрів підключається сигнал ***nReset***, який на початку моделювання приймає значення низького рівня, забезпечуючи встановлення на виходах базових регістрів нульового стану, а потім переключається до верхнього (неактивного) рівня.

Результати моделювання функціонування підсумкового регістра приведені на рис.2.50.

В функції збудження до керуючих входів C_c, $C_м$ доданий сигнал *G* відповідно до (2.5).

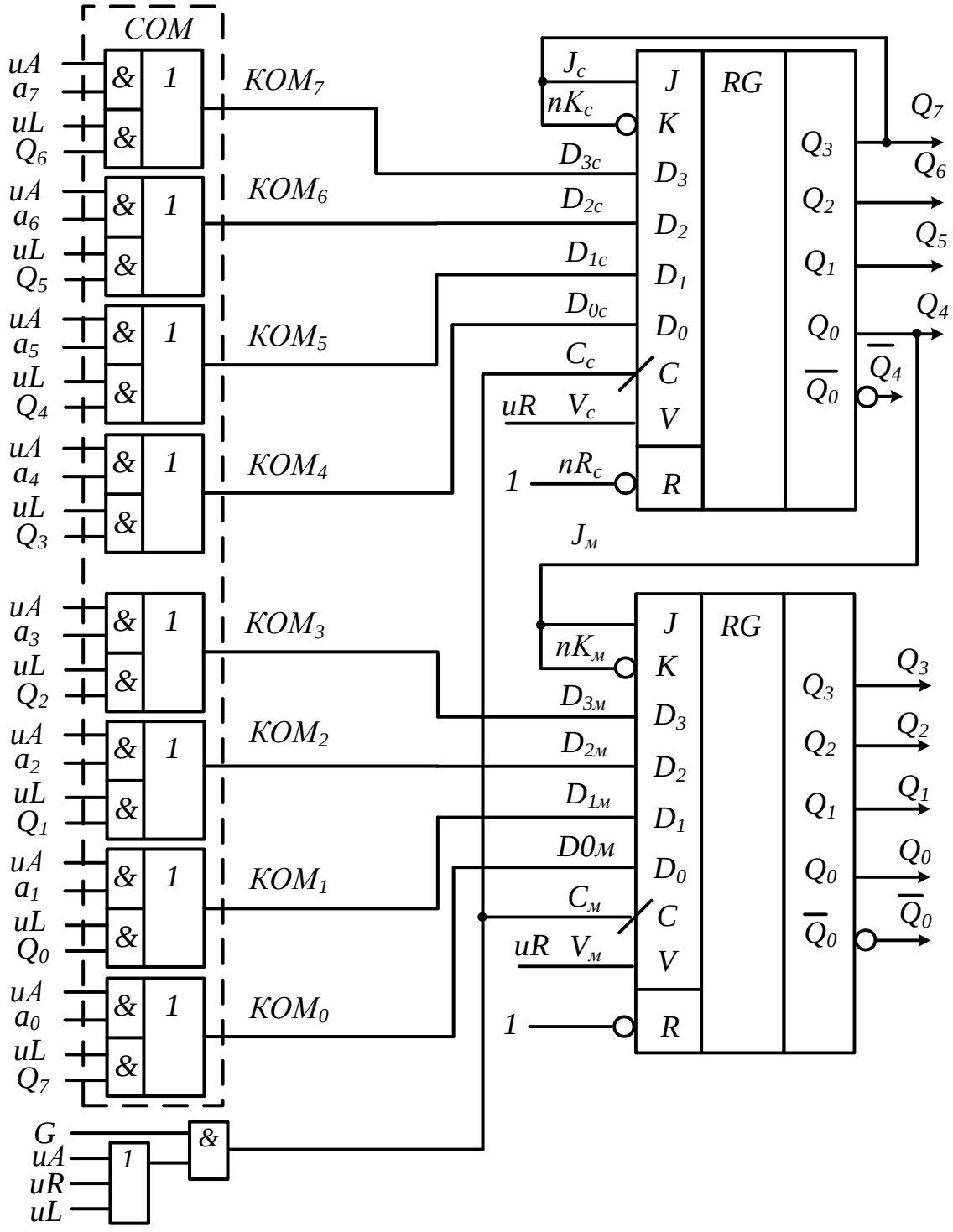


Рисунок 2.48 – Логічна схема підсумкового регістра (приклад 2.5)

Час спрацьовування підсумкового регістра визначається часом затримки базових регістрів як при паралельному завантаженні, так і під час зсуву (відповідно рис.2.51, 2.52), тобто динамічні параметри регістра визначаються практично таким же чином, як і в регістрах на базі тригерів.

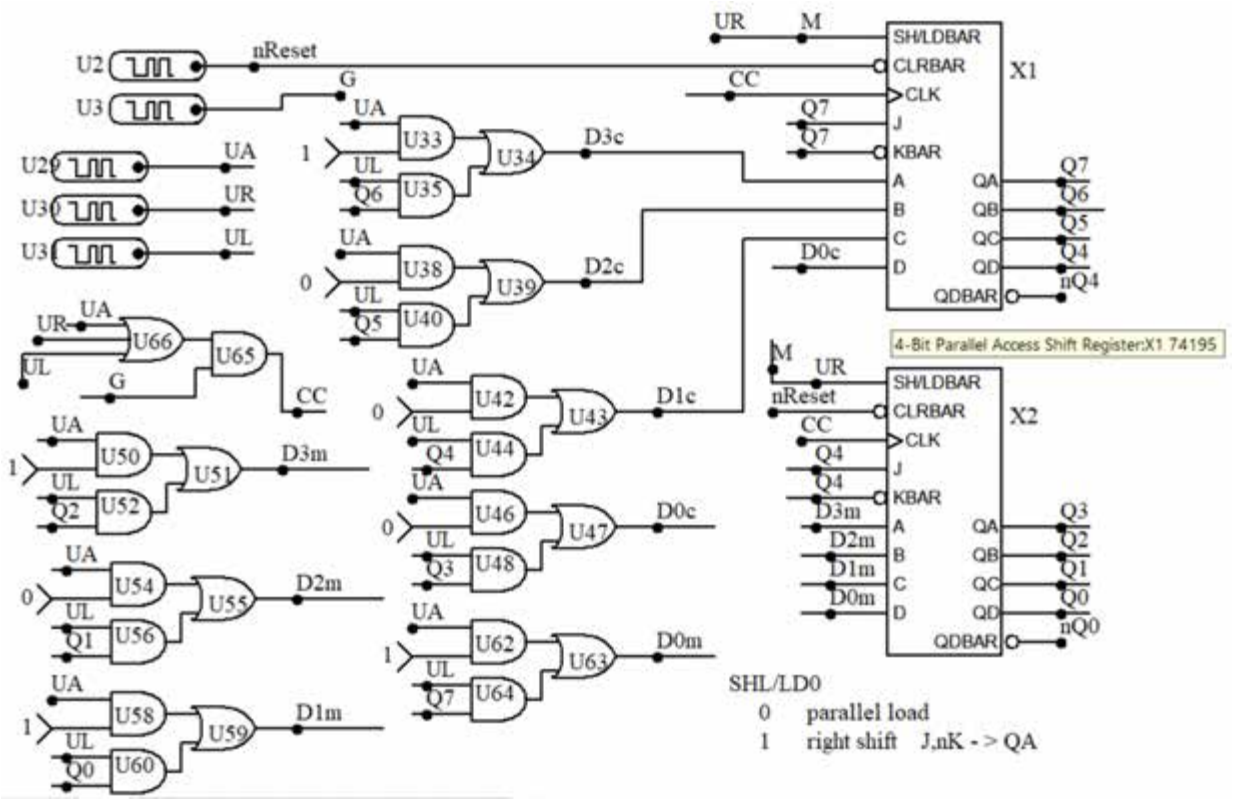


Рисунок 2.49 – Логічна схема для моделювання підсумкового регістра на базі *SN74195* (приклад 2.5)

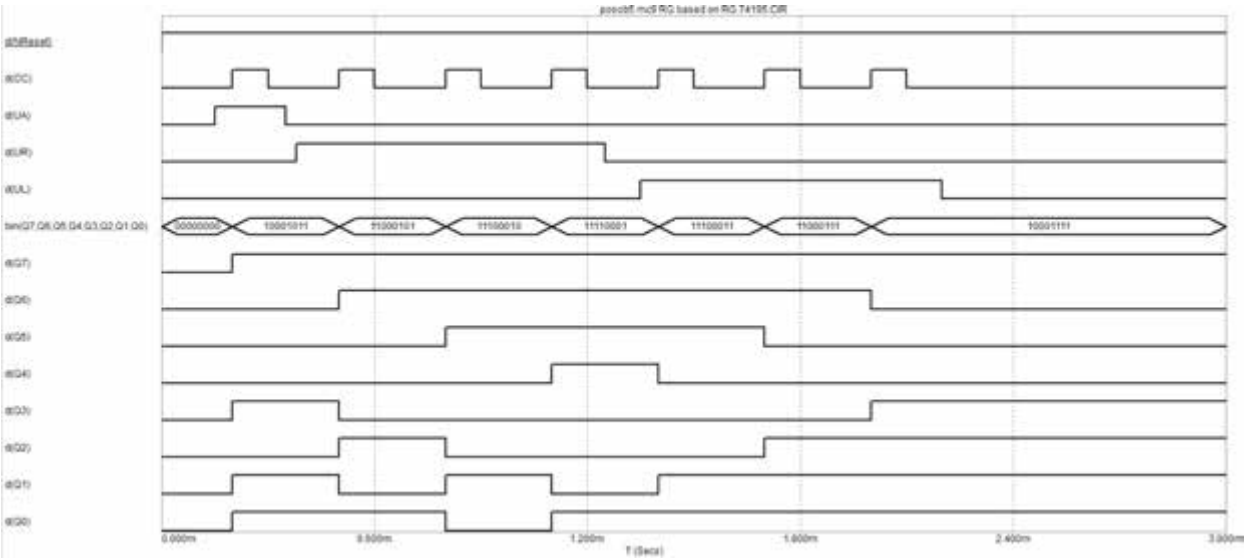


Рисунок 2.50 – Результати моделювання підсумкового регістра на базі *SN74195* (приклад 2.5)

Але при цьому необхідно пам’ятати, що перед переключенням базового регістра необхідно забезпечити спрацьовування комутатора, що впливає на максимальну частоту надходження сигналу синхронізації.

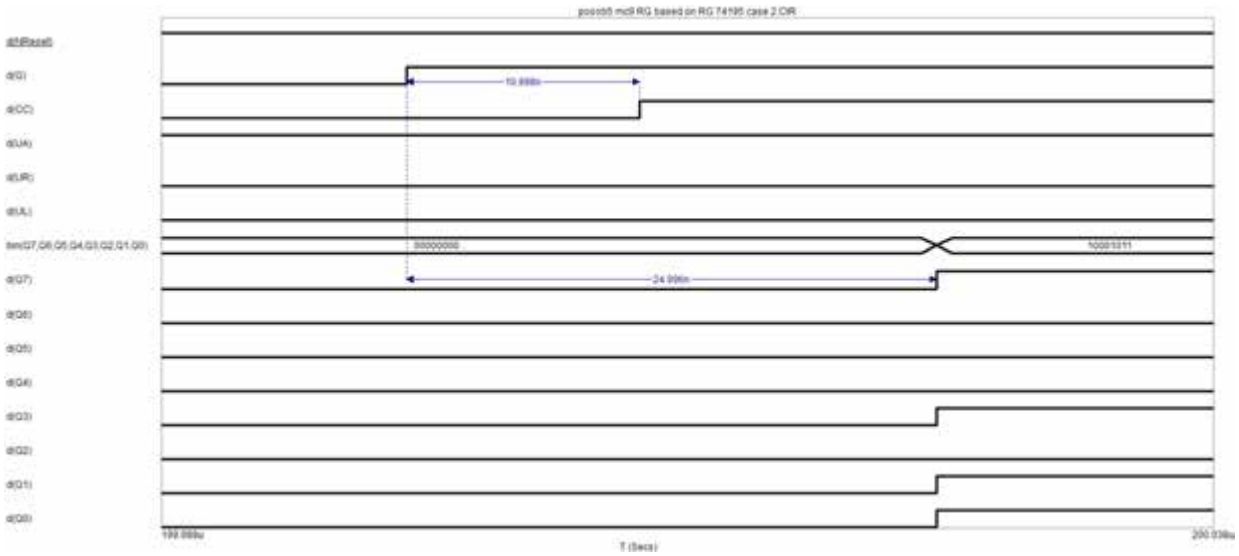


Рисунок 2.51 – Час спрацьовування регістра на базі *SN74195* при виконанні паралельного завантаження (приклад 2.5)

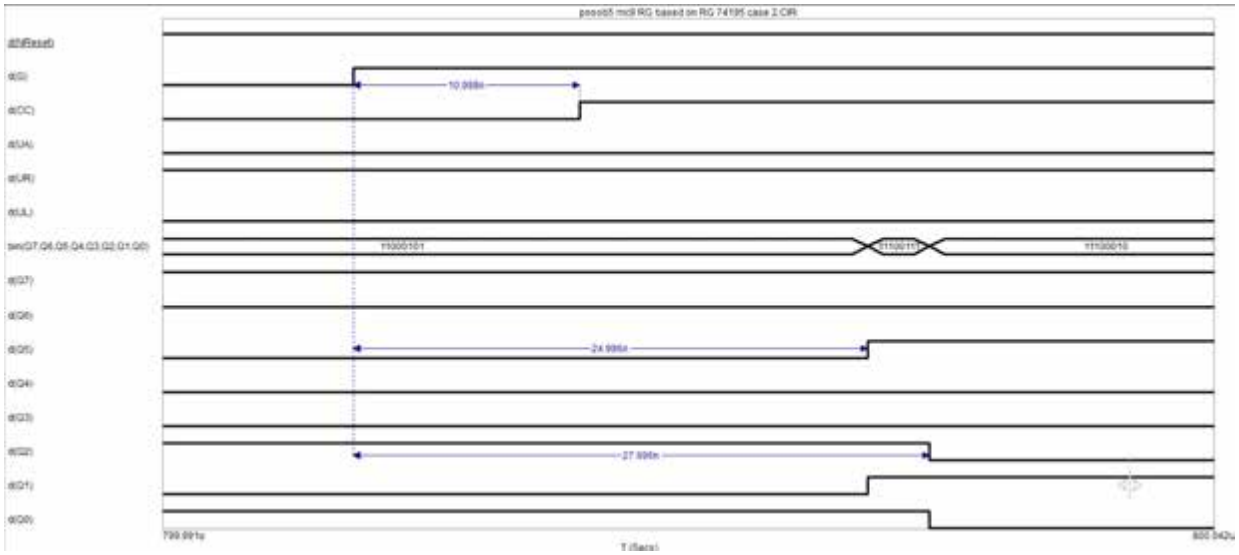


Рисунок 2.52 – Час спрацьовування регістра на базі *SN74195* при виконанні зсуву вправо (приклад 2.5)

На цьому синтез регістра на базі *SN74195* виконано.

На завершення можна відзначити, що в цьому прикладі реалізацію арифметичного зсуву можна отримати не за рахунок перезапису інформації з виходу Q_7 на входи J_c, nK_c , а за допомогою організації збереження інформації в старшому розряді старшої тетради $J_c = 0$; $nK_c = 1$ (відповідно до табл.2.19). Фрагмент схеми регістра приведений на рис.2.53 (кола входів J_c, nK_c обведені пунктиром). Результати моделювання регістра з цією схемою приведені на рис.2.54.

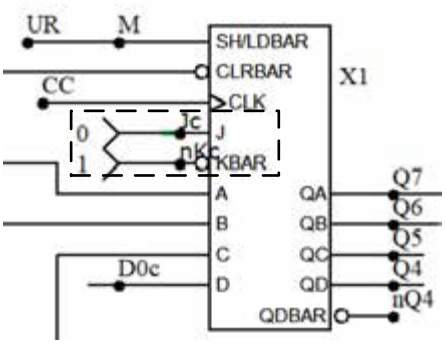


Рисунок 2.53 – Фрагмент схеми регістра з реалізацією збереження старшого розряду при виконанні арифметичного зсуву (приклад 2.5)

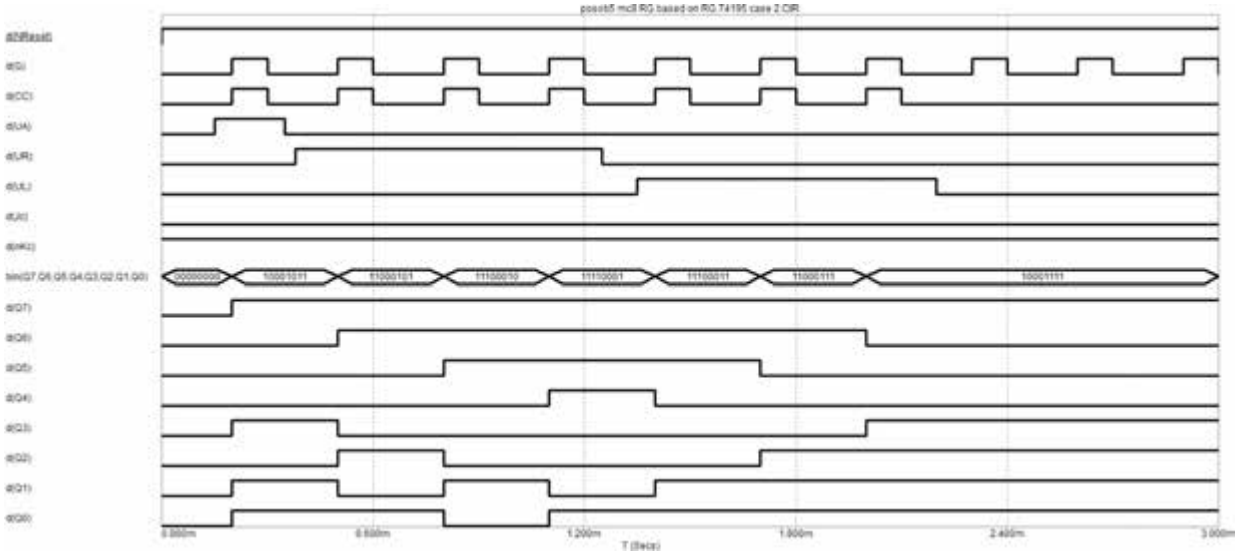


Рисунок 2.54 – Результати моделювання регістра на базі *SN74195* при $J_c = 0$; $nK_c = 1$ (приклад 2.5)

Розглянемо ще один спосіб використання ІС регістрів другої групи. Наприклад, якщо необхідно виконати зсув вмісту регістра *SN74195* вліво на

один розряд (нагадаємо, що згідно з описом цей регістр виконує тільки зсув вправо) за умови, що зсув вправо виконувати непотрібно, то цей зсув також можна зробити за рахунок внутрішніх кіл зсуву цього регістра. Для цього необхідно змінити нумерацію виходів на зворотну, фактично змінивши старшинство розрядів, тобто вихід регістра Q_3 позначити як Q_0 , Q_2 – як Q_1 і так далі. (рис.2.55).

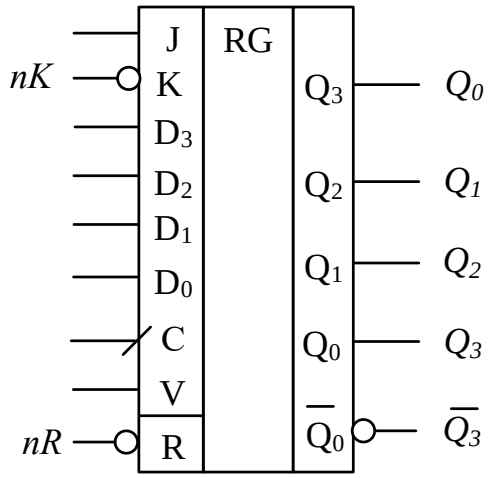


Рисунок 2.55 – Використання регістра на базі *SN74195* для виконання зсуву вліво на один розряд

Далі розглянемо синтез підсумкового регістра на базі ІС регістрів третьої групи, які є реверсивними зсувними регістрами.

Приклад 2.6. На базі ІС *SN74194* виконати синтез 8-бітного реверсивного регістра з керованою синхронізацією в будь-якому базисі логічних елементів. Регістр виконує такі мікрооперації: *uR*: *Rg* := *AR1*(*Rg*) ; *uL*: *Rg* := *CL1*(*Rg*); *uA*: *Rg* := *A*. Структурна схема регістра приведена на рис.2.27.

Розв'язок

Синтез регістра також будемо виконувати відповідно до порядку, приведеному в підрозділі 2.2.2.

На першому кроці синтезу проведемо аналіз функціонування підсумкового регістра під час виконання заданих мікрооперацій.

З умови завдання можна побачити, що регістр, як і в попередньому прикладі, виконує ті ж самі мікрооперації, тому аналіз функціонування підсумкового регістра див. в прикладі 2.5.

Крім того, необхідно провести аналіз функціонування заданого базового регістра *SN74194*. Опис цього регістра можна отримати в будь-якому довіднику з цифрових схем або на будь-якому сайті з інформацією від виробника, наприклад [7].

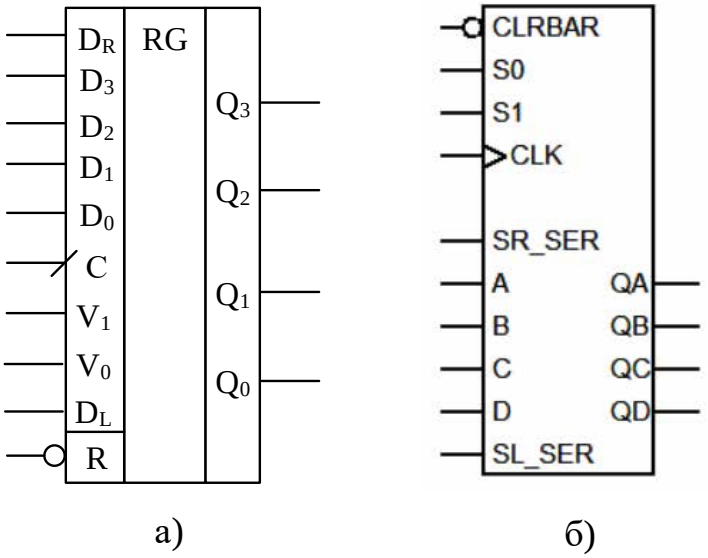
Умовне графічне позначення базового регістра *SN74194* приведено на рис.2.56, де УГП на рис.2.56,а приведено за вітчизняними стандартами, а на рис.2.56,б – відповідно до позначення цього регістра в середовищі MicroCap.

З УГП можна побачити, що базовий регістр, як і в попередньому прикладі, є 4-розрядним.

Призначення виводів регістра *SN74194* приведено в табл.2.25 [7].

Виводи Q_3, Q_2, Q_1, Q_0 (*QA*, *QB, QC, QD*) є виходами регістра і визначають його стан в будь-який момент часу. Входи *C* (*CLK*), *R* (*CLRBAR*) та $D_3 – D_0$ (*A*, *B*, *C*, *D*) мають те ж саме призначення, що і в ІС *SN74195*.

На відміну від регістра з прикладу 2.5, регістр *SN74194* має два входи вибору режиму V_1, V_0 (S_1, S_0), тобто регістр може виконувати чотири мікрооперації в залежності від значень сигналів на цих входах.



а) б)

Рисунок 2.56 – УГП регістра *SN74194*

Таблиця 2.25 – Призначення виводів ІС *SN74194*

Виводи на рис.2.56,а	Виводи на рис.2.56,б	Призначення
V_1, V_0	S_1, S_0	Входи вибору режиму
D_3,D_2,D_1,D_0	*A, B, C, D*	Входи паралельного завантаження
C	*CLK*	Загальний вхід синхронізації
R	*CLRBAR*	Асинхронний вхід для скиду регістра
D_R	*SR_SER*	Вхід послідовного завантаження (зсув вправо)
D_L	*SL_SER*	Вхід послідовного завантаження (зсув вліво)
Q_3,Q_2,Q_1,Q_0	*QA,QB,QC,QD*	Виходи регістра

Перелік мікрооперацій, що виконує базовий регістр *SN74194* приведений в табл.2.26.

Таблиця 2.26 – Режими роботи базового регістра *SN74194*

V_1	V_0	Режим
L(0)	*L(0)*	Збереження стану регістра
L(0)	*H(1)*	Зсув вправо ($D_R \Rightarrow Q_3$)
H(1)	*L(0)*	Зсув вліво ($D_L \Rightarrow Q_0$)
H(1)	*H(1)*	Паралельне завантаження

Значення входів вибору режиму V_1, V_0 (S_1, S_0) задані за допомогою рівнів напруги *L* і *H* та відповідних значень двійкових змінних для позитивної логіки у дужках (*1* або *0*).

Як видно з таблиці базовий регістр може виконувати зсув в обох напрямках, тобто є реверсивним. Для забезпечення виконання будь-якого типу зсуву в регістрі використовуються два входи для послідовного завантаження даних D_R (D_R – ***Data Right***, *SR_SER* – ***Shift Right Serial***) при організації зсуву вправо і D_L (D_L – ***Data Left***, *SL_SER* – ***Shift Left Serial***) при зсуві вліво.

Під час виконання зсуву вправо інформація з входу D_R записується в

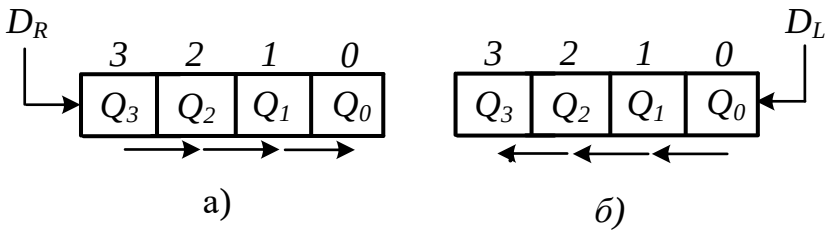
с

т

а

р

ш



а) *б)*

Рисунок 2.57 – Використання входів послідовного завантаження

При виконанні зсувів входи паралельного завантаження є відключеними і не впливають на роботу регістра.

Для організації паралельного завантаження необхідно забезпечити високі рівні напруги на входах V_1, V_0 (S_1, S_0), в результаті чого інформація з входів D_3, D_2, D_1, D_0 (*A, B, C, D*) буде записуватися у відповідні тригери регістра Q_3, Q_2, Q_1, Q_0 (*QA, QB, QC, QD*).

На рис.2.58 приведена функціональна схема ІС регістра *SN74194* [7].

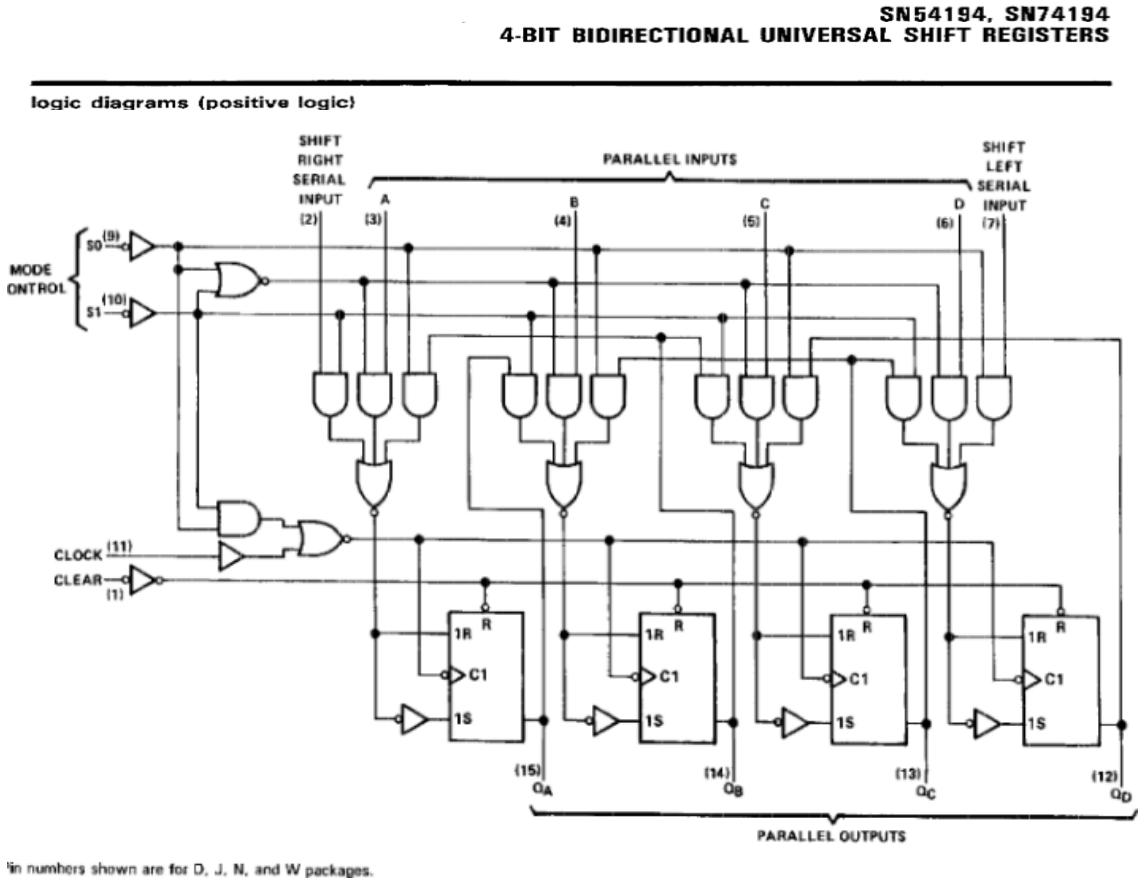


Рисунок 2.58 – Функціональна схема базового регістра *SN74194*

З функціональної схеми можна побачити, що в регістрі *SN74194* в якості базових тригерів використовуються *RCS*-тригери, які перетворені на *D*-тригери. Процедура синтезу одного тригера на базі іншого приведена в розділі 1 цього посібника.

Таким чином, здійснено аналіз функціонування заданого базового регістра, в результаті якого визначена розрядність цього регістра, призначення його входів і виходів, режими роботи, тобто перший крок процедури синтезу підсумкового регістра завершено.

На другому кроці процедури синтезу визначимо кількість базових регістрів, необхідних для реалізації підсумкового регістра. Відповідно до (2.18), враховуючи, що згідно з умовою завдання $n_R = 8$, а $n_B = 4$, отримаємо, що кількість базових регістрів $k_{BR} = 2$.

Далі (крок 3) виконаємо розробку об'єднаної таблиці мікрооперацій. З умови завдання випливає, що регістр виконує три мікрооперації, тобто кількість рядків таблиці дорівнює трьом. Відповідно до рис.2.56 кількість входів одного базового регістра складає 10, а кількість цих регістрів дорівнює двом, тобто кількість стовпців в таблиці мікрооперації дорівнює 20. У зв'язку з великою кількістю стовпців будемо представляти таблицю мікрооперацій для кожної тетради окремо.

Підготовлена до заповнення таблиця мікрооперацій для синтезу регістра приведена в табл.2.27.

За аналогією з попереднім прикладом входи старшої і молодшої тетрад будемо позначати відповідно індексами «*с*» і «*м*», а також використовувати наскрізну нумерацію розрядів регістра (рядки «Номер виходу»).

Далі виконаємо заповнення відповідного рядку таблиці мікрооперацій для кожної мікрооперації окремо (колонка 1). Як і в попередньому прикладі, заповнення клітин таблиці необхідно починати з входів керування регістром.

Таблиця 2.27 – Загальний вигляд таблиці мікрооперацій (приклад 2.6)

МО	Старша тетрада									
	D_{Rc}	D_{3c}	D_{2c}	D_{1c}	D_{0c}	D_{Lc}	C_c	V_{1c}	V_{0c}	nR_c
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
uA										
uR										
uL										
Номер виходу		*7*	*6*	*5*	*4*					
МО	Молодша тетрада									
	$D_{Rм}$	$D_{3м}$	$D_{2м}$	$D_{1м}$	$D_{0м}$	$D_{Lм}$	$C_м$	$V_{1м}$	$V_{0м}$	$nR_м$
uA										
uR										
uL										
Номер виходу		*3*	*2*	*1*	*0*					

Розглянемо заповнення таблиці для мікрооперації *uA*, за яким відбувається паралельне завантаження інформації з шини *A*. Для цього необхідно сформувати сигнали високого рівня (*1*) на входах V_{1c}, V_{0c}, $V_{1м}$, $V_{0м}$ базових регістрів (колонки 9, 10). При цьому на входах nR_c і $nR_м$ (колонка 11) сформуємо неактивний сигнал (*1*). На входах C_c і $C_м$ (колонка 8) необхідно забезпечити надходження сигналу *uA*, що відмічається у відповідних клітинах. Таким чином, заповнення таблиці для керуючих входів завершено, а далі заповнюємо клітини для інформаційних входів. В зв'язку з тим, що для виконання мікрооперації *uA*, регістр налаштовується на паралельне завантаження, входи послідовного завантаження D_{Rc}, $D_{Rм}$, D_{Lc}, $D_{Lм}$ (колонки 2, 7) в цьому режимі не впливають на роботу регістра, тому у відповідних клітинах записується символ «*». І, нарешті, заповнюємо клітини для входів паралельного завантаження старшої і молодшої тетрад регістра (колонки 3- 6). В старшу тетраду завантажується вміст старших чотирьох ліній шини даних *A* (a_7-a_4), а в молодшу – вміст молодших чотирьох ліній цієї шини (a_3-a_0). ТМО для *uA* приведена в табл.2.28.

Таблиця 2.28 – Таблиця мікрооперацій для сигналу *uA* (приклад 2.6)

МО	Старша тетрада									
	D_{Rc}	D_{3c}	D_{2c}	D_{1c}	D_{0c}	D_{Lc}	C_c	V_{1c}	V_{0c}	nR_c
1	2	3	4	5	6	7	8	9	10	11
uA	*	a_7	a_6	a_5	a_4	*	*uA*	1	1	1
uR										
uL										
Номер виходу		7	6	5	4					
МО	Молодша тетрада									
	$D_{Rм}$	$D_{3м}$	$D_{2м}$	$D_{1м}$	$D_{0м}$	$D_{Lм}$	$C_м$	$V_{1м}$	$V_{0м}$	$nR_м$
uA	*	a_3	a_2	a_1	a_0	*	*uA*	1	1	1
uR										
uL										
Номер виходу		3	2	1	0					

Після цього заповнимо таблицю мікрооперацій для сигналу *uR*.

Для забезпечення виконання мікрооперації налаштовуємо регістр на здійснення зсуву вправо, для чого на входах V_1 обох тетрад формуємо низький рівень сигналу, а на входах V_0 – високий рівень. В цьому випадку в базових регістрах активізуються внутрішні кола зсуву, а входи паралельного завантаження відключаються і при цьому не використовуються (в таблиці мікрооперацій в клітинах, що відповідають цим входам, записується символ «*»).

Як і для попередньої мікрооперації, на входах nR_c і $nR_м$ формується неактивний сигнал (*1*), а на входи C_c і $C_м$ забезпечується надходження сигналу *uR*.

Розглянемо формування функцій збудження для входів послідовного завантаження D_R і D_L. Відповідно до умови завдання за сигналом *uR* відбувається арифметичний зсув вмісту регістра вправо на один розряд, тобто входи D_L не використовуються, що позначається «*». В розряди Q_6 - Q_4 , Q_2 - Q_0 інформація записується за допомогою внутрішніх кіл цих регістрів. Для забезпечення реалізації арифметичного зсуву та зсувів між тетрадами розглянемо використання входів D_R.

Передача інформації між розрядами регістра при виконанні арифметичного зсуву приведена на рис.2.59 (зовнішні з'єднання показані пунктиром).

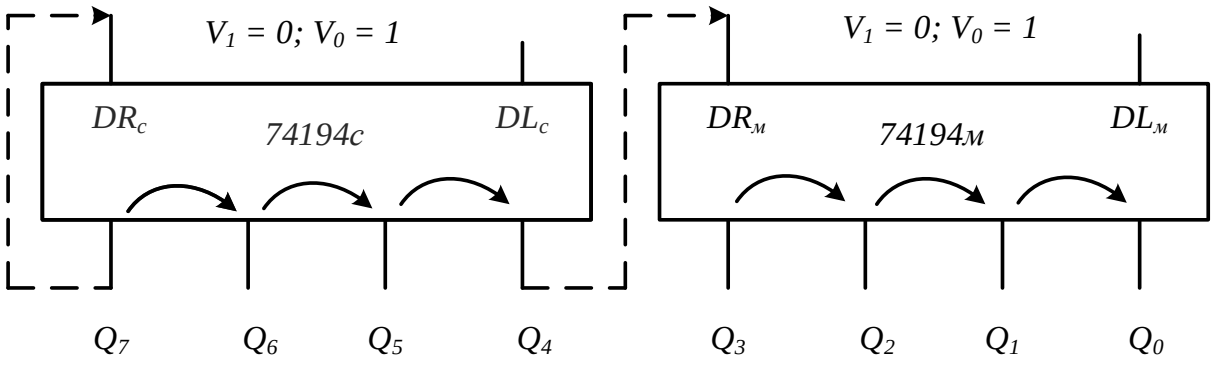


Рисунок 2.59 – Виконання арифметичного зсуву вправо на один розряд на базі регістрів *SN74194*

Вміст старшого розряду регістра (Q_7) визначається типом зсуву і в даному випадку його вміст не змінюється, що можна реалізувати за допомогою перезапису стану старшого розряду в цей же розряд. В старший розряд молодшої тетради (Q_3) записується вміст молодшого розряду попередньої тетради (Q_4). Таким чином, вхід D_{Rc} використовується для реалізації заданого типу зсуву, а вхід $D_{Rм}$ – для передачі інформації між тетрадами. Заповнена таблиця мікрооперацій для виконання арифметичного зсуву вправо на один розряд приведена в табл.2.29.

Відповідно до рис.2.59 в старшому розряді старшої тетради відбувається перезапис стану Q_7 зі входу D_{Rc} в цей же розряд, а стан молодшого розряду цієї тетради Q_4 зі входу $D_{Rм}$ – в старший розряд молодшої тетради.

Далі розглянемо заповнення клітин таблиці мікрооперацій для забезпечення виконання мікрооперації *uL*, яка полягає у виконанні циклічного зсуву вмісту регістра вліво на один розряд.

Таблиця 2.29 – Таблиця мікрооперацій для сигналу *uR* (приклад 2.6)

МО	Старша тетрада									
	D_{Rc}	D_{3c}	D_{2c}	D_{1c}	D_{0c}	D_{Lc}	C_c	V_{1c}	V_{0c}	nR_c
1	2	3	4	5	6	7	8	9	10	11
uA										
uR	Q_7	*	*	*	*	*	*uR*	*0*	*1*	*1*
uL										
Номер виходу		7	6	5	4					
МО	Молодша тетрада									
	$D_{Rм}$	$D_{3м}$	$D_{2м}$	$D_{1м}$	$D_{0м}$	$D_{Lм}$	$C_м$	$V_{1м}$	$V_{0м}$	$nR_м$
uA										
uR	Q_4	*	*	*	*	*	*uR*	*0*	*1*	*1*
uL										
Номер виходу		3	2	1	0					

На відміну від попереднього прикладу заданий базовий регістр може виконувати зсув вліво за рахунок внутрішніх кіл. Для забезпечення виконання мікрооперації налаштовуємо регістр на здійснення зсуву вліво, для чого на входах V_1 обох тетрад формуємо високий рівень сигналу, а на входах V_0 – низький рівень. В цьому випадку в базових регістрах активізуються внутрішні кола зсуву вліво, а входи паралельного завантаження D_3-D_0 та входи послідовного завантаження D_R відключаються і не використовуються (в таблиці мікрооперацій у відповідних клітинах записується символ «*»).

Функції збудження для входів nR_c, $nR_м$ та C_c, $C_м$ формуються таким же чином, як і для попередньої мікрооперації.

Передача інформації між розрядами регістра при виконанні циклічного зсуву приведена на рис.2.60 (зовнішні з’єднання між тетрадами показані пунктиром). Для виконання зсуву використовуються входи D_L базових регістрів, при чому в молодший розряд старшої тетради зі входу D_{Lc} записується стан старшого розряду молодшої тетради Q_3, а в молодший розряд молодшої тетради зі входу $D_{Lм}$ записується стан старшого розряду старшої тетради Q_7.

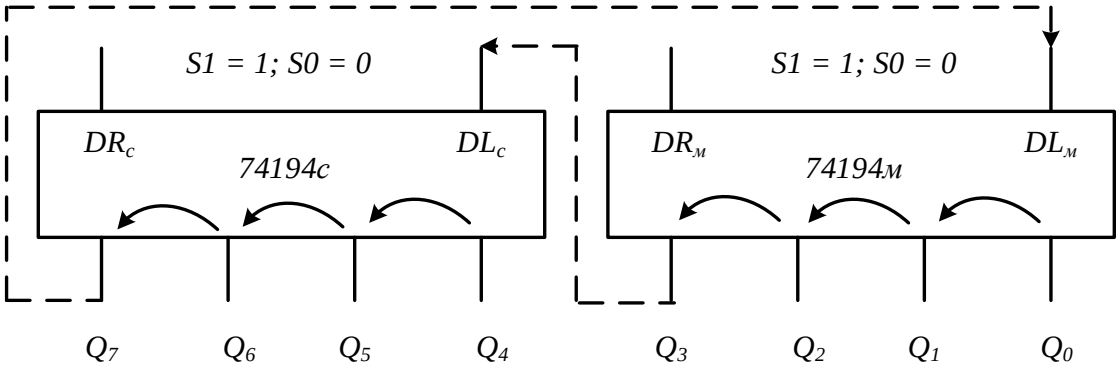


Рисунок 2.60 – Виконання циклічного зсуву вліво на один розряд на базі регістрів *SN74194*

Заповнена таблиця мікрооперацій для виконання циклічного зсуву вліво на один розряд приведена в табл.2.30. В результаті повна таблиця мікрооперацій для синтезу підсумкового регістра приведена в табл.2.31 та третій крок процедури синтезу регістра закінчено.

Виконаємо четвертий крок процедури синтезу регістра. Для цього визначимо функції збудження для кожного входу базового регістра аналогічно правилам, приведеним в п. 2.1.1.1.

Таблиця 2.30 – Таблиця мікрооперацій для сигналу *uL* (приклад 2.6)

МО	Старша тетрада									
	D_{Rc}	D_{3c}	D_{2c}	D_{1c}	D_{0c}	D_{Lc}	C_c	V_{1c}	V_{0c}	nR_c
1	2	3	4	5	6	7	8	9	10	11
uA										
uR										
uL	*	*	*	*	*	Q_3	*uL*	1	0	1
Номер виходу		7	6	5	4					
МО	Молодша тетрада									
	$D_{Rм}$	$D_{3м}$	$D_{2м}$	$D_{1м}$	$D_{0м}$	$D_{Lм}$	$C_м$	$V_{1м}$	$V_{0м}$	$nR_м$
uA										
uR										
uL	*	*	*	*	*	Q_7	*uL*	1	0	1
Номер виходу		3	2	1	0					

Таблиця 2.31 – Таблиця мікрооперацій для синтезу регістра (приклад 2.6)

МО	Старша тетрада									
	D_{Rc}	D_{3c}	D_{2c}	D_{1c}	D_{0c}	D_{Lc}	C_c	V_{1c}	V_{0c}	nR_c
1	2	3	4	5	6	7	8	9	10	11
uA	*	a_7	a_6	a_5	a_4	*	uA	1	1	1
uR	Q_7	*	*	*	*	*	uR	0	1	1
uL	*	*	*	*	*	Q_3	uL	1	0	1
Номер виходу		7	6	5	4					
МО	Молодша тетрада									
	$D_{Rм}$	$D_{3м}$	$D_{2м}$	$D_{1м}$	$D_{0м}$	$D_{Lм}$	$C_м$	$V_{1м}$	$V_{0м}$	$nR_м$
uA	*	a_3	a_2	a_1	a_0	*	uA	1	1	1
uR	Q_4	*	*	*	*	*	uR	0	1	1
uL	*	*	*	*	*	Q_7	uL	1	0	1
Номер виходу		3	2	1	0					

В результаті отримаємо наступні функції збудження:

- для входів керування:

$$nR_c = nR_м = 1; \quad C_c = C_м = (uA \vee uR \vee uL)\cdot G;$$
$$V_{1c} = V_{1м} = uA \vee uL ; \qquad V_{0c} = V_{0м} = uA \vee uR; \tag{2.22}$$

- для інформаційних входів:

$$D_{ic} = a_{i+4} \cdot uA \ (i = 0, \ldots, 3); \qquad D_{iм} = a_i \cdot uA \ (i = 0, \ldots, 3);$$
$$D_{Rc} = Q_7 \cdot uR; \ D_{Rм} = Q_4 \cdot uR; \ D_{Lc} = Q_3 \cdot uL; \ D_{Lм} = Q_7 \cdot uL. \tag{2.23}$$

Для входів, які приймають участь тільки в одній мікрооперації, функцію збудження можна спростити, видаливши з логічного виразу керуючий сигнал (п’ятий крок процедури синтезу). З таблиці мікрооперацій випливає, що всі інформаційні входи регістрів використовуються тільки в одній із заданих мікрооперацій, тому логічні вирази для цих входів можна спростити таким чином:

$$D_{ic} = a_{i+4} (i = 0, \ldots, 3); \quad D_{iм} = a_i (i = 0, \ldots, 3);$$
$$D_{Rc} = Q_7 ; \quad D_{Rм} = Q_4; \quad D_{Lc} = Q_3; \quad D_{Lм} = Q_7. \tag{2.24}$$

Відповідно до умови завдання логічні вирази реалізуються в будь- якому базисі, тому залишимо їх без перетворень.

Логічна схема підсумкового регістра приведена на рис.2.61.

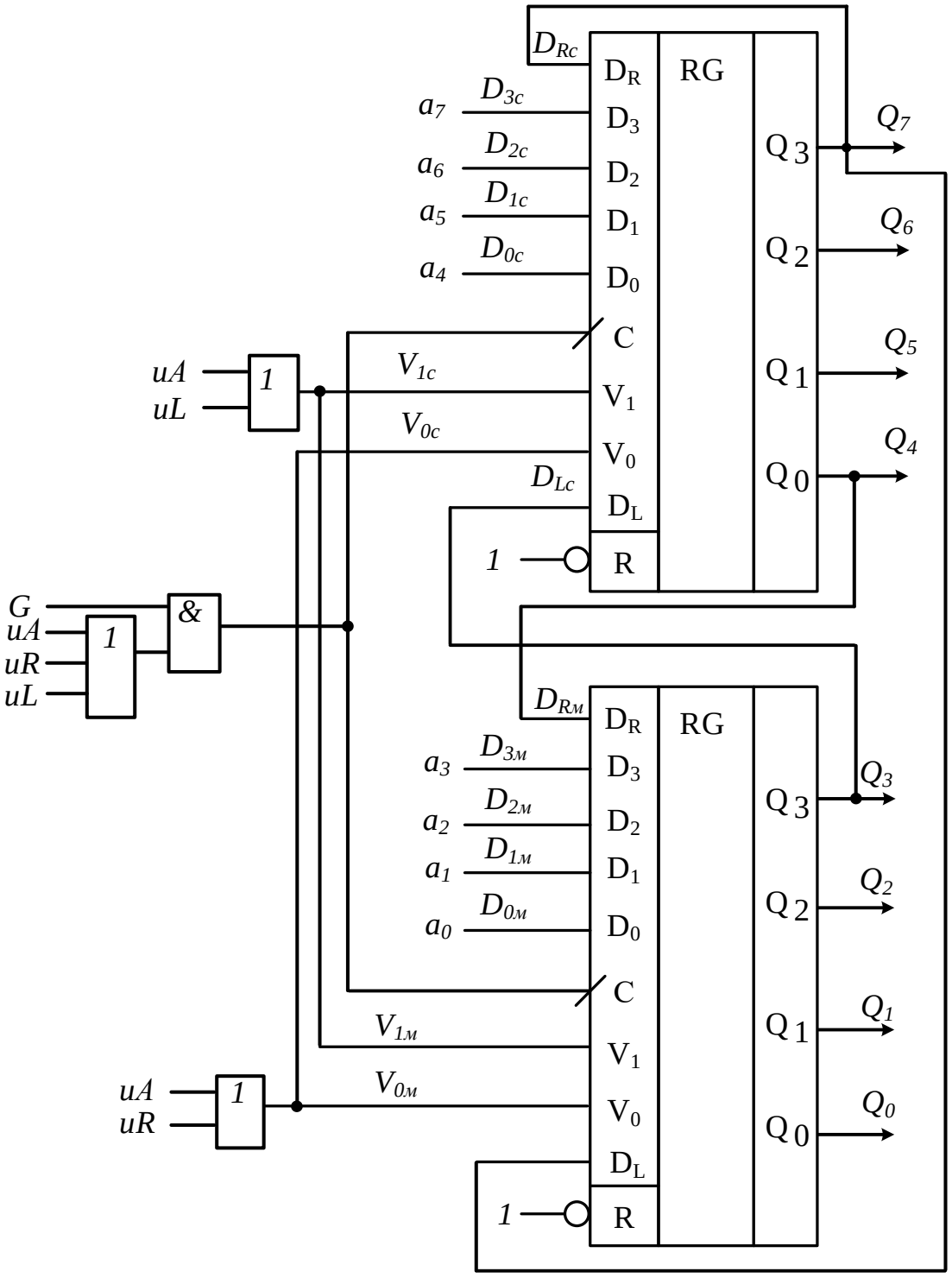


Рисунок 2.61 – Логічна схема підсумкового регістра (приклад 2.6)

Схема для моделювання підсумкового регістра на основі базових регістрів *SN74194* в середовищі MicroCap приведена на рис.2.62.

Для імітації вмісту вхідної шини *A* використовуються елементи ***Fixed Digital***, які утворюють двійковий код *01111101* для формування функцій

збудження D_{3c}, …, D_{0c} і $D_{3м}$, …, $D_{0м}$. В результаті цей код за сигналом *uA* завантажується в базові регістри.

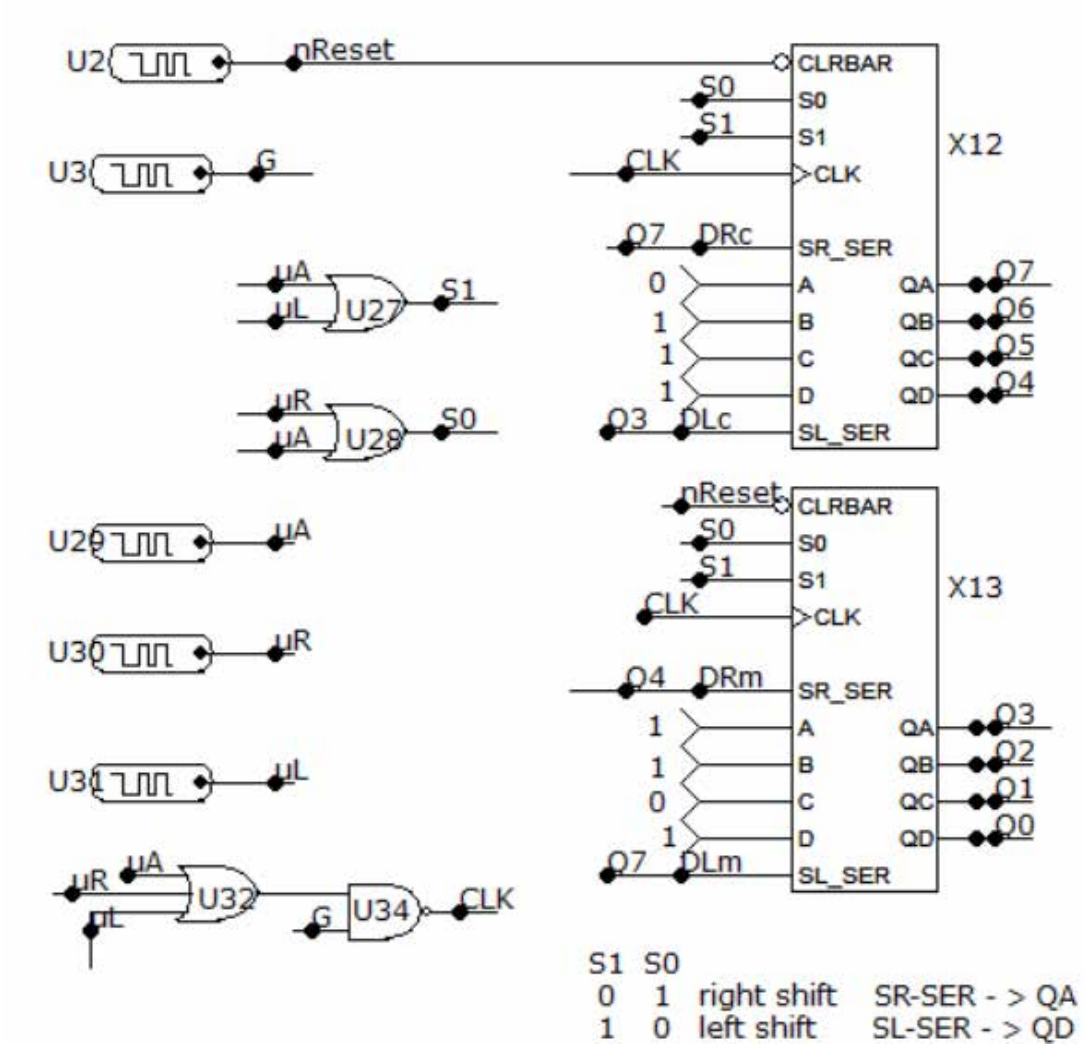


Рисунок 2.62 – Логічна схема для моделювання підсумкового регістра на базі *SN74194* (приклад 2.6)

Вхід *R* базових регістрів використовується таким же чином, як і в прикладі 2.5.

В функції збудження до керуючих входів C_c, $C_м$ доданий сигнал *G* відповідно до (2.5).

Моделювання виконується таким чином, що спочатку задається сигнал *uA*, забезпечуючи паралельне завантаження коду *01111101*. Далі тричі задається сигнали *uR*, тобто вміст регістра тричі зсувається арифметично вправо (*00111110*, *00011111*, *00001111*), а потім задаються три сигнали *uL*, за

якими вміст регістра зсувається вліво циклічно (*00011110*, *00111100*, *01111000*).

Результати моделювання функціонування підсумкового регістра приведені на рис.2.63.

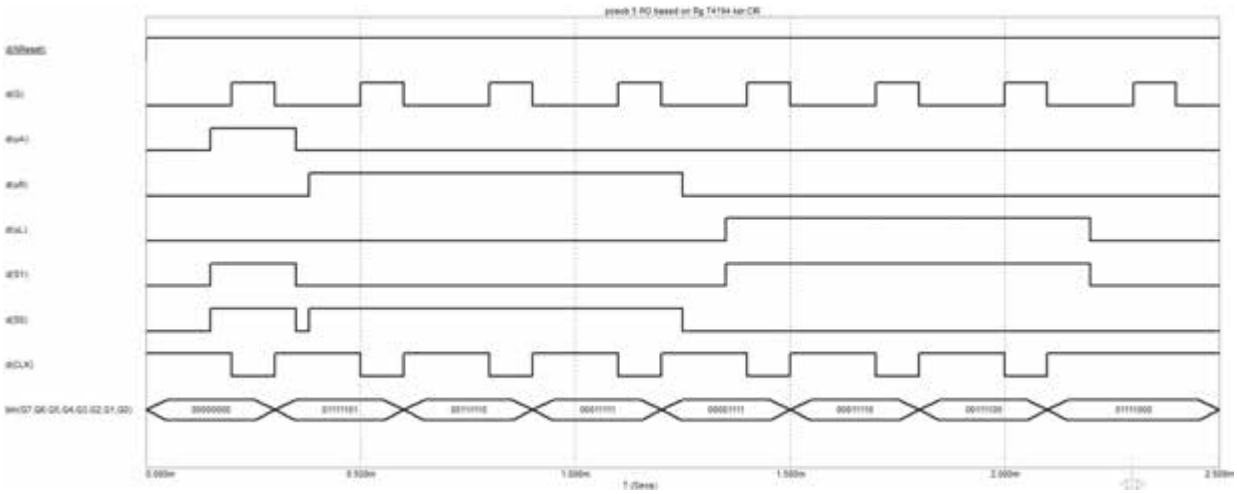


Рисунок 2.63 – Результати моделювання підсумкового регістра на базі *SN74194* (приклад 2.6)

Динамічні параметри підсумкового регістра визначаються за аналогією з прикладом 2.5.

Таким чином, синтез регістра на базі ІС *SN74194* виконано.

Синтез регістрів четвертої групи (в складі лічильників) буде розглянуто пізніше.

2.2.4 Приклади синтезу регістрів з некерованою синхронізацією на базі ІС регістрів

Синтез регістрів першої групи виконується таким же чином, як і регістри на базі окремих тригерів.

Розглянемо синтез підсумкового регістра з некерованою синхронізацією на базі ІС регістрів другої і третьої груп.

Приклад 2.7. На базі ІС *SN74195* виконати синтез 8-бітного реверсивного регістра з некерованою синхронізацією в будь-якому базисі

логічних елементів. Регістр виконує такі мікрооперації: uR: $Rg := AR1(Rg)$; uL: $Rg := CL1(Rg)$; uA: $Rg := A$. Структурна схема регістра приведена на рис.2.27.

Розв'язок

Синтез регістра також будемо виконувати відповідно до порядку, приведеному вище в підрозділі 2.2.2.

Аналіз функціонування підсумкового регістра, а також УГП, призначення виводів, таблиця режимів роботи базового регістра проведено в прикладі 2.5.

Кількість базових регістрів $k_{BR} = 2$ (див. в прикладі 2.5).

Розглянемо розробку об'єднаної таблиці мікрооперацій. Реалізація виконання мікрооперацій uA, uR, uL практично нічим не відрізняється від таблиці 2.24, але в колонках 8, 17 записується сигнал G, в зв'язку з тим, що при використанні некерованої синхронізації на синхровходи базових регістрів безпосередньо підключається тільки сигнал G. Крім того, в таблиці мікрооперацій додається ще один рядок, який відповідає мікрооперації nU.

Нагадаємо, що мікрооперація nU відповідає випадку відсутності активних значень керуючих сигналів на входах регістра.

Відповідно до (2.14) логічний вираз для реалізації сигналу nU виглядає наступним чином: $nU = \overline{uA} \cdot \overline{uR} \cdot \overline{uL}$ (див. приклад 2.3). При цьому врахування наявності сигналу nU виконується за рахунок налаштування режиму паралельного завантаження, при якому відбувається перезапис стану i-того розряду регістра на входи цього ж розряду.

В результаті таблиця мікрооперацій регістра набуває вигляду, як показано в табл.2.32.

Після цього визначимо функції збудження для кожного входу базових регістрів аналогічно правилам, приведеним в п. 2.1.1.1:

- для входів керування:

$$nR_c = nR_м = 1; \quad C_c = C_м = G; \quad V_c = V_м = uR; \qquad (2.25)$$

- для інформаційних входів:

$$D_{ic} = a_{i+4} \cdot uA \vee Q_{i+3} \cdot uL \vee Q_{i+4} \cdot nU \ (i = 0, \dots, 3);$$
$$D_{iм} = a_i \cdot uA \vee Q_{i-1} \cdot uL \vee Q_i \cdot nU \ (i = 1, \dots, 3);$$
$$D_{0м} = a_0 \cdot uA \vee Q_7 \cdot uL \vee Q_0 \cdot nU \ ; \qquad (2.26)$$
$$J_c = nK_c = Q_7 \cdot uR; \ J_м = nK_м = Q_4 \cdot uR;$$

Таблиця 2.32 – Таблиця мікрооперацій для синтезу регістра (приклад 2.7)

МО	Старша тетрада									Молодша тетрада								
	Jc	nK_c	D_{3c}	D_{2c}	D_{1c}	D_{0c}	C_c	V_c	nR_c	$J_м$	$nK_м$	$D_{3м}$	$D_{2м}$	$D_{1м}$	$D_{0м}$	$C_м$	$V_м$	$nR_м$
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
uA	*	*	a_7	a_6	a_5	a_4	*G*	*0*	*1*	*	*	a_3	a_2	a_1	a_0	*G*	*0*	*1*
uR	Q_7	Q_7	*	*	*	*	*G*	*1*	*1*	Q_4	Q_4	*	*	*	*	*G*	*1*	*1*
uL	*	*	Q_6	Q_5	Q_4	Q_3	*G*	*0*	*1*	*	*	Q_2	Q_1	Q_0	Q_7	*G*	*0*	*1*
nU	*	*	Q_7	Q_6	Q_5	Q_4	Q_3	*0*	*1*	*	*	Q_3	Q_2	Q_1	Q_0	*G*	*0*	*1*
Номер виходу			*7*	*6*	*5*	*4*						*3*	*2*	*1*	*0*			

Після мінімізації можна отримати, що $J_c = nK_c = Q_7$; $J_м = nK_м = Q_4$.

Логічна схема підсумкового регістра приведена на рис.2.64.

Схема для моделювання ПР з некерованою синхронізацією на базі ІС *SN74195* в середовищі MicroCap приведена на рис.2.65.

Результати моделювання функціонування підсумкового регістра приведені на рис.2.66.

Під час моделювання спочатку задається сигнал *uA*, забезпечуючи паралельне завантаження коду *10101001*. Далі тричі задається сигнали *uR*, тобто вміст регістра тричі зсувається арифметично вправо (*11010100*, *11101010*, *11110101*), а потім задаються три сигнали *uL*, за якими вміст регістра зсувається вліво циклічно (*11101011*, *11010111*, *10101111*).

Після цього керуючі сигнали для виконання мікрооперацій відсутні, в цьому випадку формується сигнал *nU*, який налаштовує регістр на збереження інформації, що можна побачити на рис.2.66.

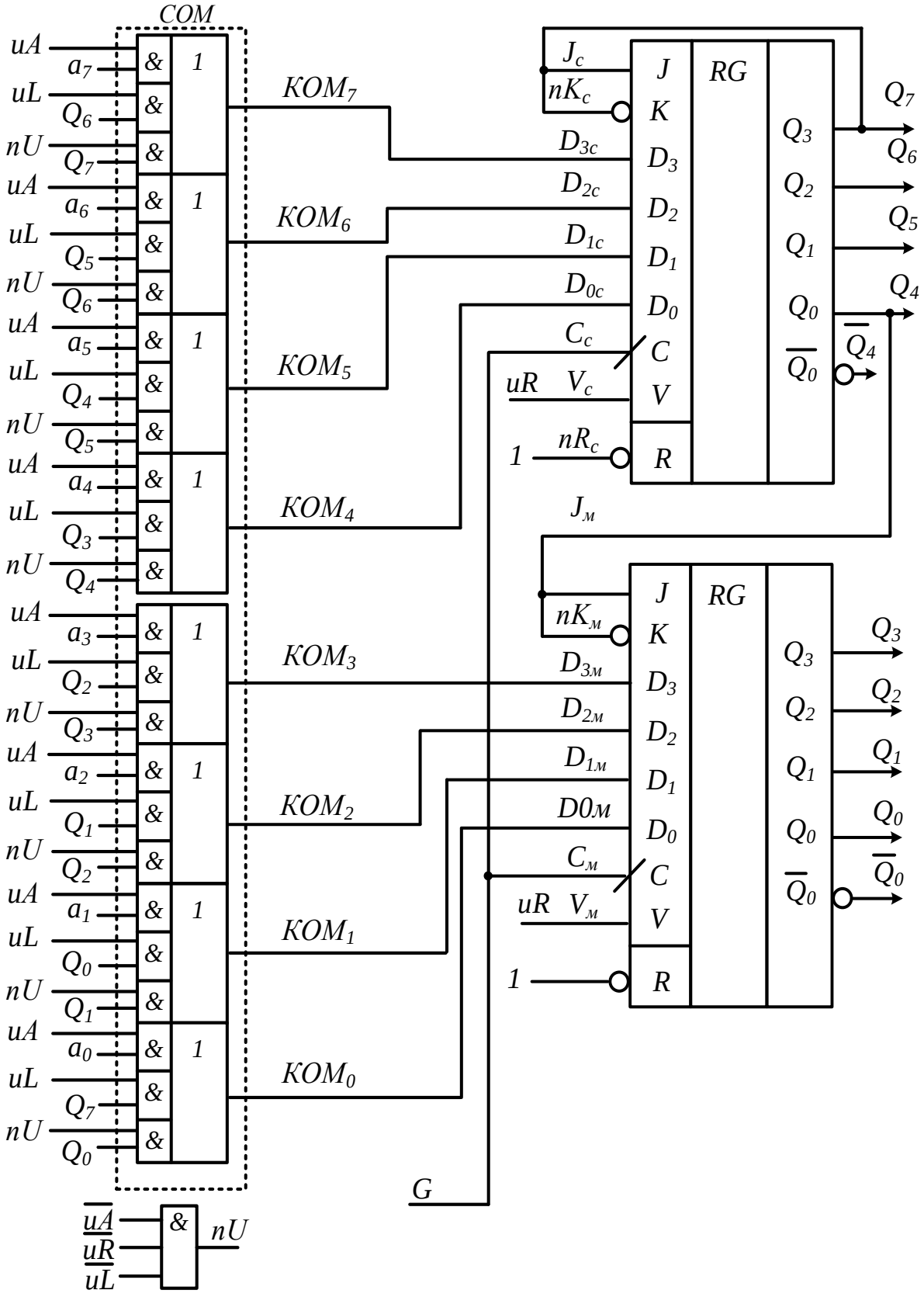


Рисунок 2.64 – Логічна схема підсумкового регістра (приклад 2.7)

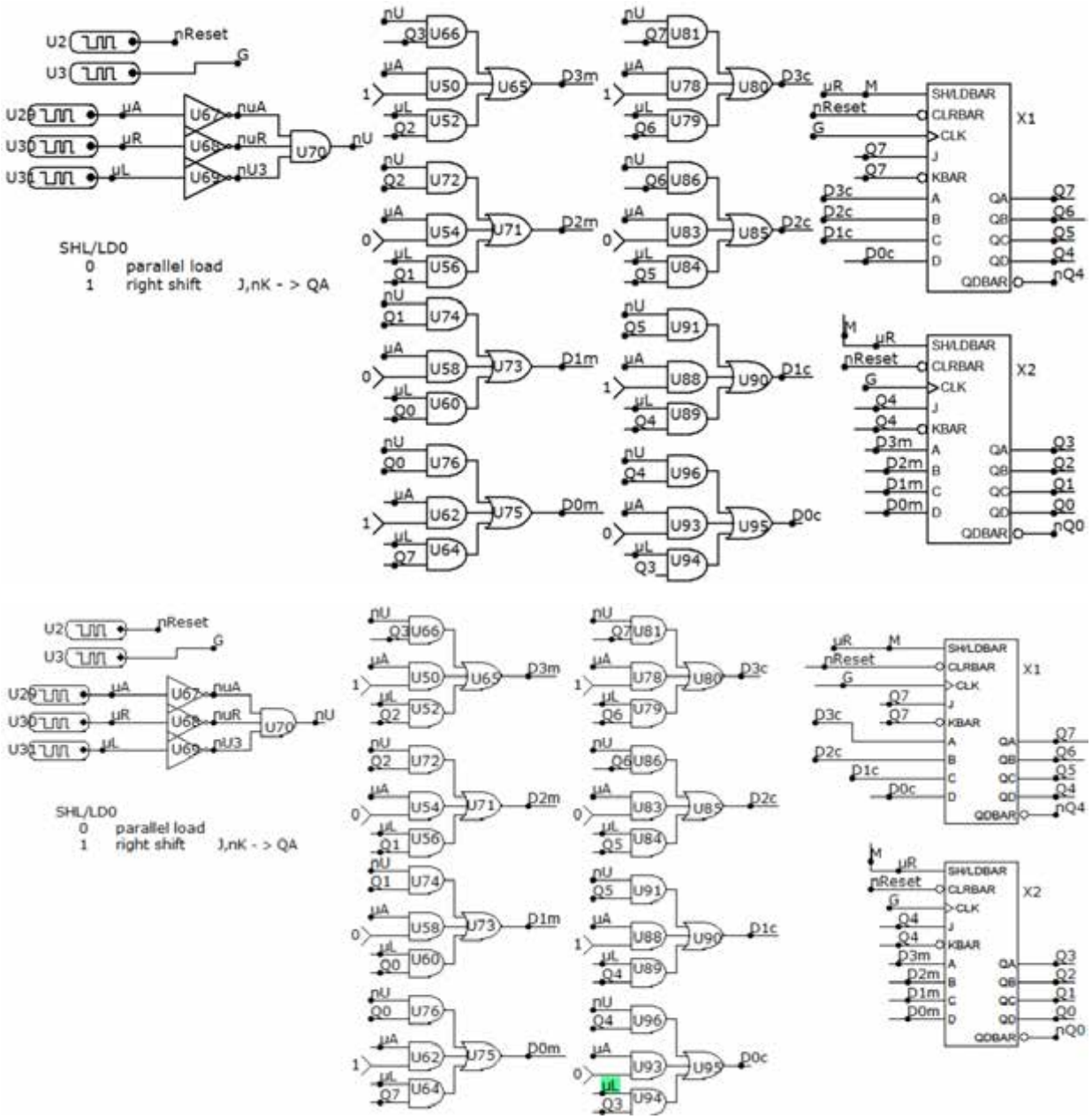


Рисунок 2.65 – Логічна схема для моделювання підсумкового регістра на базі *SN74194* (приклад 2.7)

Динамічні параметри визначаються за аналогією з попередніми прикладами.

На цьому синтез регістра на базі *SN74195* завершено.

Порівнюючи логічні схеми регістрів на рис.2.48 і рис.2.64, можна зробити висновок, що в другій схемі (рис.2.64) використовується більш проста реалізація керування регістром, але при цьому достатньо сильно ускладнилася схема комутатора. Таким чином, при використанні базового

регістра другої групи доцільно виконувати синтез регістра з керованою синхронізацією.

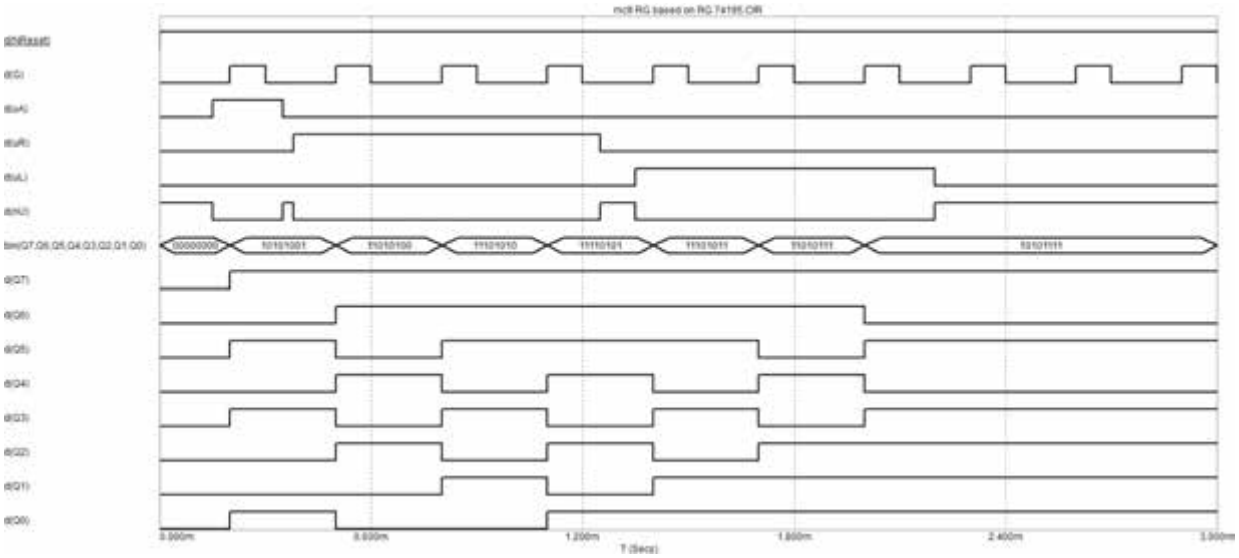


Рисунок 2.66 – Результати моделювання підсумкового регістра на базі *SN74194* (приклад 2.7)

Далі розглянемо синтез підсумкового регістра з некерованою синхронізацією на базі ІС регістрів третьої групи, які є реверсивними зсувними регістрами.

Приклад 2.8. На базі ІС *SN74194* виконати синтез 8-бітного реверсивного регістра з некерованою синхронізацією в будь-якому базисі логічних елементів. Регістр виконує такі мікрооперації: *uR*: *Rg* := *AR1*(*Rg*) ; *uL*: *Rg* := *CL1*(*Rg*); *uA*: *Rg* := *A*. Структурна схема регістра приведена на рис.2.27.

Розв’язок

Синтез регістра також будемо виконувати відповідно до порядку, приведеному в підрозділі 2.2.2.

Аналіз функціонування підсумкового регістра, а також УГП, призначення виводів, таблиця режимів роботи базового регістра проведено в прикладі 2.6.

Кількість базових регістрів $k_{BR} = 2$ (див. в прикладі 2.6).

Розглянемо розробку об'єднаної таблиці мікрооперацій. Реалізація виконання мікрооперацій *uA*, *uR*, *uL* практично нічим не відрізняється від таблиці 2.31, але в стовпчику 8 записується сигнал *G*. Як і в попередньому прикладі в таблиці мікрооперацій додається ще один рядок, який відповідає мікрооперації *nU*, який формується в регістрі за виразом:

$$nU = \overline{uA} \cdot \overline{uR} \cdot \overline{uL} = \overline{uA \vee uR \vee uL}.$$

При цьому врахування наявності сигналу *nU* доцільно застосовувати за рахунок налаштування мікрооперації збереження інформації не за допомогою перезапису стану *i*-того розряду регістра на входи цього ж розряду, а за рахунок використання входів вибору режиму $V_1 = V_0 = 0$. В результаті таблиця мікрооперацій регістра набуває вигляду, як показано в табл.2.33.

Таблиця 2.33 – Таблиця мікрооперацій для синтезу регістра (приклад 2.8)

МО	Старша тетрада									
	D_{Rc}	D_{3c}	D_{2c}	D_{1c}	D_{0c}	D_{Lc}	C_c	V_{1c}	V_{0c}	nR_c
1	2	3	4	5	6	7	8	9	10	11
uA	*	a_7	a_6	a_5	a_4	*	*G*	1	1	1
uR	Q_7	*	*	*	*	*	*G*	0	1	1
uL	*	*	*	*	*	Q_3	*G*	1	0	1
nU	*	*	*	*	*	*	*G*	0	0	1
Номер виходу		7	6	5	4					
МО	Молодша тетрада									
	$D_{Rм}$	$D_{3м}$	$D_{2м}$	$D_{1м}$	$D_{0м}$	$D_{Lм}$	$C_м$	$V_{1м}$	$V_{0м}$	$nR_м$
uA	*	a_3	a_2	a_1	a_0	*	*G*	1	1	1
uR	Q_4	*	*	*	*	*	*G*	0	1	1
uL	*	*	*	*	*	Q_7	*G*	1	0	1
nU	*	*	*	*	*	*	*G*	0	0	1
Номер виходу		3	2	1	0					

З таблиці 2.33 можна отримати такі функції збудження:

- для входів керування:

$$nR_c = nR_м = 1;\ C_c = C_м = G;\ V_{1c} = V_{1м} = uA \vee uL\ ;\ V_{0c} = V_{0м} = uA \vee uR; \quad (2.27)$$

- для інформаційних входів з врахуванням мінімізації:

$$D_{ic} = a_{i+4}\,(i = 0, \ldots, 3); \quad D_{iм} = a_i\,(i = 0, \ldots, 3);$$
$$D_{Rc} = Q_7;\ D_{Rм} = Q_4;\ D_{Lc} = Q_3;\ D_{Lм} = Q_7. \qquad (2.28)$$

Зверніть увагу, що до виразів (2.27), (2.28) не входить змінна nU (хоча під час синтезу використовувалася), тобто в схемі регістра її можна не реалізовувати.

Логічна схема підсумкового регістра приведена на рис.2.67.

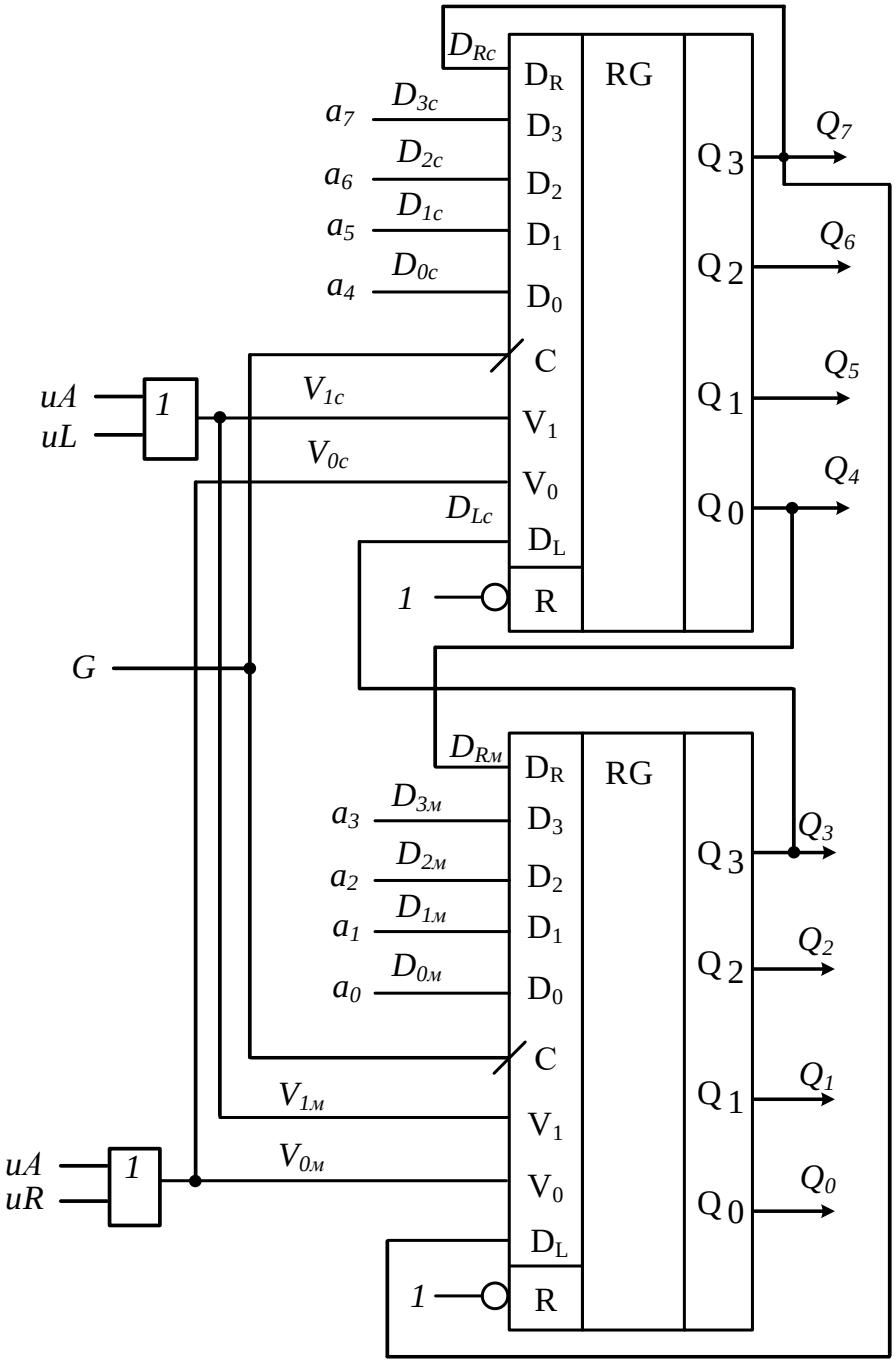


Рисунок 2.67 – Логічна схема підсумкового регістра (приклад 2.8)

Схема для моделювання підсумкового регістра на основі базових регістрів *SN74194* в середовищі MicroCap приведена на рис.2.68.

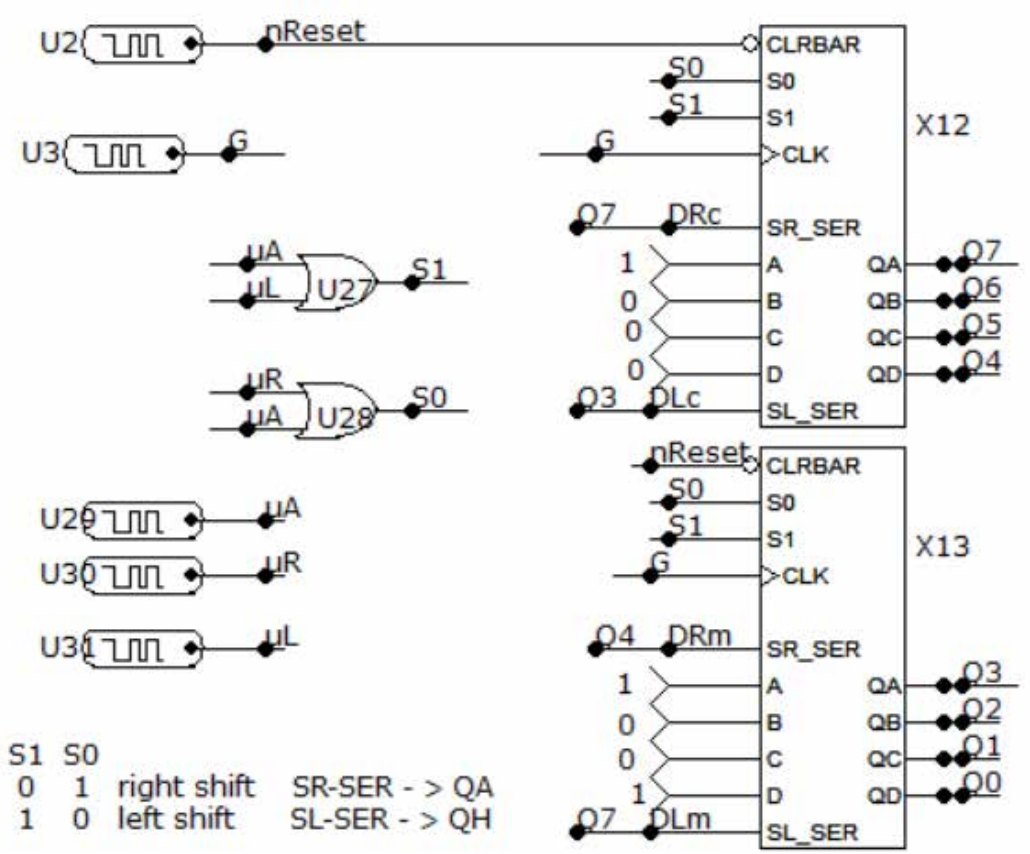


Рисунок 2.68 – Логічна схема для моделювання підсумкового регістра на базі *SN74194* (приклад 2.8)

Як і раніше для імітації вмісту вхідної шини *A* використовуються елементи ***Fixed Digital***, які утворюють двійковий код *10001001*, завантажується за сигналом *uA*.

Далі після трьох керуючих сигналів *uR* вміст регістра кожного сигналу зсувається арифметично вправо (*11000100*, *11100010*, *11110001*), а потім за трьома сигналами *uL* вміст регістра зсувається вліво циклічно (*11100011*, *11000111*, *10001111*).

Результати моделювання підсумкового регістра на базі *SN74194* приведені на рис.2.69.

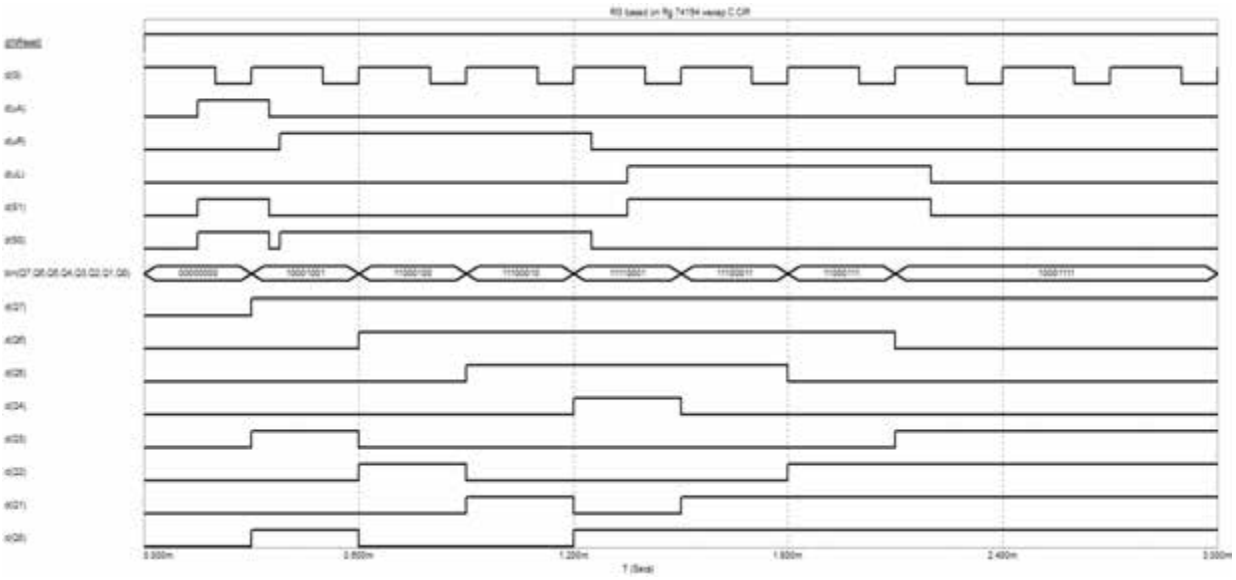


Рисунок 2.69 – Результати моделювання підсумкового регістра на базі *SN74194* (приклад 2.8)

Динамічні параметри підсумкового регістра визначаються за аналогією з попередніми прикладами.

Порівнюючи логічні схеми регістрів на рис.2.61 і рис.2.67, можна зробити висновок, що друга схема (рис.2.67) є більш простішою за рахунок відсутності схеми керування.

Таким чином, при використанні базових регістрів третьої групи доцільно виконувати синтез підсумкового регістра з некерованою синхронізацією.

Таким чином, синтез регістра з некерованою синхронізацією на базі ІС *SN74194* виконано.

Далі розглянемо ще один приклад, в якому ілюструється випадок налаштування базових регістрів в ПР на різні режими роботи.

Приклад 2.9. На базі ІС *SN74194* виконати синтез 8-бітного реверсивного регістра з керованою синхронізацією в будь-якому базисі логічних елементів. Регістр виконує такі мікрооперації: *uR*: *Rg* := *AR1*(*Rg*) ; *uL*: *Rg* := *AL1.0*(*Rg*); *uA*: *Rg* := *A*.

Розв’язок

На відміну від попередніх прикладів за мікрооперацією *uL* виконується арифметичний зсув вліво на один розряд із заповненням розряду, що звільнюється, нулем.

Синтез аналогічного регістра було детально розглянуто в прикладі 2.6. В зв’язку з цим розглянемо налаштування регістра на виконання мікрооперації *uL*.

Умовне графічне позначення базового регістра *SN74194* приведено на рис.2.56.

Далі розглянемо заповнення клітин таблиці мікрооперацій для забезпечення виконання мікрооперації *uL*.

Налаштування молодшої тетради підсумкового регістра не відрізняється від налаштування цієї тетради в прикладі 2.6. Для реалізації арифметичного зсуву вліво на один розряд в старшій тетраді необхідно забезпечити як завантаження інформації в молодший розряд цієї тетради зі старшого розряду молодшої тетради (Q_3), так і збереження вмісту старшого розряду старшої тетради (Q_7). Це означає, що старший базовий регістр не можна налаштовувати на зсув вліво в зв’язку з тим, що при цьому не буде виконуватися збереження вмісту розряду Q_7. Таким чином, старший базовий регістр необхідно налаштовувати на виконання мікрооперації паралельного завантаження інформації за аналогією виконання зсуву вліво для ІС регістрів другої групи (приклад 2.5).

Таким чином, для виконання синтезу заданого підсумкового регістра при реалізації арифметичного зсуву вліво на один розряд *AL1.0* необхідно налаштувати базовий регістр молодшої тетради на зсув вліво ($V_{1м} = 1$; $V_{0м} = 0$), а базовий регістр старшої тетради – на паралельне завантаження ($V_{1c} = 1$; $V_{0c} = 1$).

Передача інформації між розрядами регістра при виконанні арифметичного зсуву вліво *AL1.0* приведена на рис.2.70 (зовнішні з’єднання між тетрадами показані пунктиром).

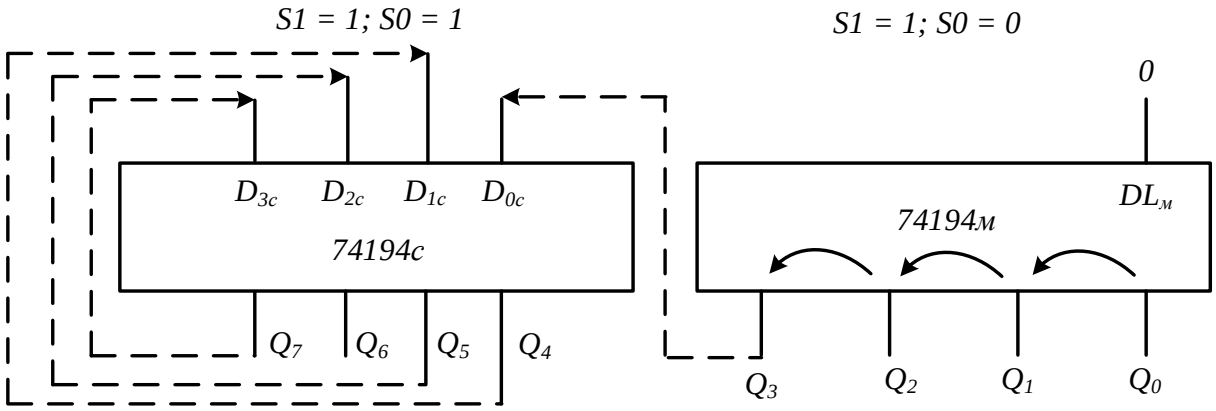


Рисунок 2.70 – Виконання зсуву *AL1.0* на базі регістрів *SN74194*

Заповнена таблиця мікрооперацій для синтезу заданого регістра приведена в табл.2.34.

Таблиця 2.34 – Таблиця мікрооперацій для синтезу регістра (приклад 2.9)

МО	Старша тетрада									
	D_{Rc}	D_{3c}	D_{2c}	D_{1c}	D_{0c}	D_{Lc}	C_c	V_{1c}	V_{0c}	nR_c
1	2	3	4	5	6	7	8	9	10	11
uA	*	a_7	a_6	a_5	a_4	*	*uA*	1	1	1
uR	Q_7	*	*	*	*	*	*uR*	0	1	1
uL	*	Q_7	Q_5	Q_4	Q_3	*	*uL*	1	1	1
Номер виходу		7	6	5	4					
МО	Молодша тетрада									
	$D_{Rм}$	$D_{3м}$	$D_{2м}$	$D_{1м}$	$D_{0м}$	$D_{Lм}$	$C_м$	$V_{1м}$	$V_{0м}$	$nR_м$
uA	*	a_3	a_2	a_1	a_0	*	*uA*	1	1	1
uR	Q_4	*	*	*	*	*	*uR*	0	1	1
uL	*	*	*	*	*	0	*uL*	1	0	1
Номер виходу		3	2	1	0					

Функції збудження для входів nR_c, $nR_м$ та C_c, $C_м$ формуються таким же чином, як і для попередніх прикладів синтезу регістра з керованою синхронізацією.

Функції збудження (з врахуванням мінімізації) для кожного входу базового регістра визначаємо аналогічно правилам, приведеним в п. 2.1.1.1.

В результаті отримаємо наступні функції збудження:

- для входів керування:

$$nR_c = nR_м = 1; \qquad C_c = C_м = (uA \vee uR \vee uL)\cdot G;$$
$$V_{1c} = V_{1м} = uA \vee uL\ ; \quad V_{0м} = uA \vee uR; \qquad V_{0c} = 1; \tag{2.29}$$

- для інформаційних входів:

$$D_{ic} = a_{i+4}\cdot uA \vee Q_{i+3}\cdot uR\ (i = 0, \ldots, 2);\ D_{3c} = a_7\cdot uA \vee Q_7\cdot uR;$$
$$D_{iм} = a_i\ (i = 0, \ldots, 3);\ D_{Rc} = Q_7;\ D_{Rм} = Q_4;\ D_{Lc} = D_{Lм} = 0. \tag{2.30}$$

Логічна схема підсумкового регістра приведена на рис.2.71.

Схема для моделювання підсумкового регістра на основі базових регістрів *SN74194* в середовищі MicroCap приведена на рис.2.72.

Для імітації вмісту вхідної шини *A* використовуються елементи ***Fixed Digital***, які утворюють двійковий код *01111101* для формування функцій збудження D_{3c}, …, D_{0c} і $D_{3м}$, …, $D_{0м}$. В результаті цей код за сигналом *uA* (старша тетрада завантажується через комутатор) завантажується в базові регістри.

До функцій збудження керуючих входів C_c, $C_м$ додано сигнал *G* відповідно до (2.5).

Моделювання виконується таким чином, що спочатку задається сигнал *uA*, забезпечуючи паралельне завантаження коду *01111101*. Далі тричі задаються сигнали *uL*, тобто вміст регістра тричі зсувається арифметично вліво із заповненням нулем в молодшому розряді (*01111010*, *01110100*, *01101000*), а потім задаються три сигнали *uR*, за якими вміст регістра зсувається арифметично вправо (*00110100*, *00011010*, *00001101*).

Результати моделювання функціонування підсумкового регістра приведені на рис.2.73.

Динамічні параметри підсумкового регістра визначаються за аналогією з попередніми прикладами.

Таким чином, синтез регістра, що виконує зсув *AL1.0* на базі ІС *SN74194*, виконано.

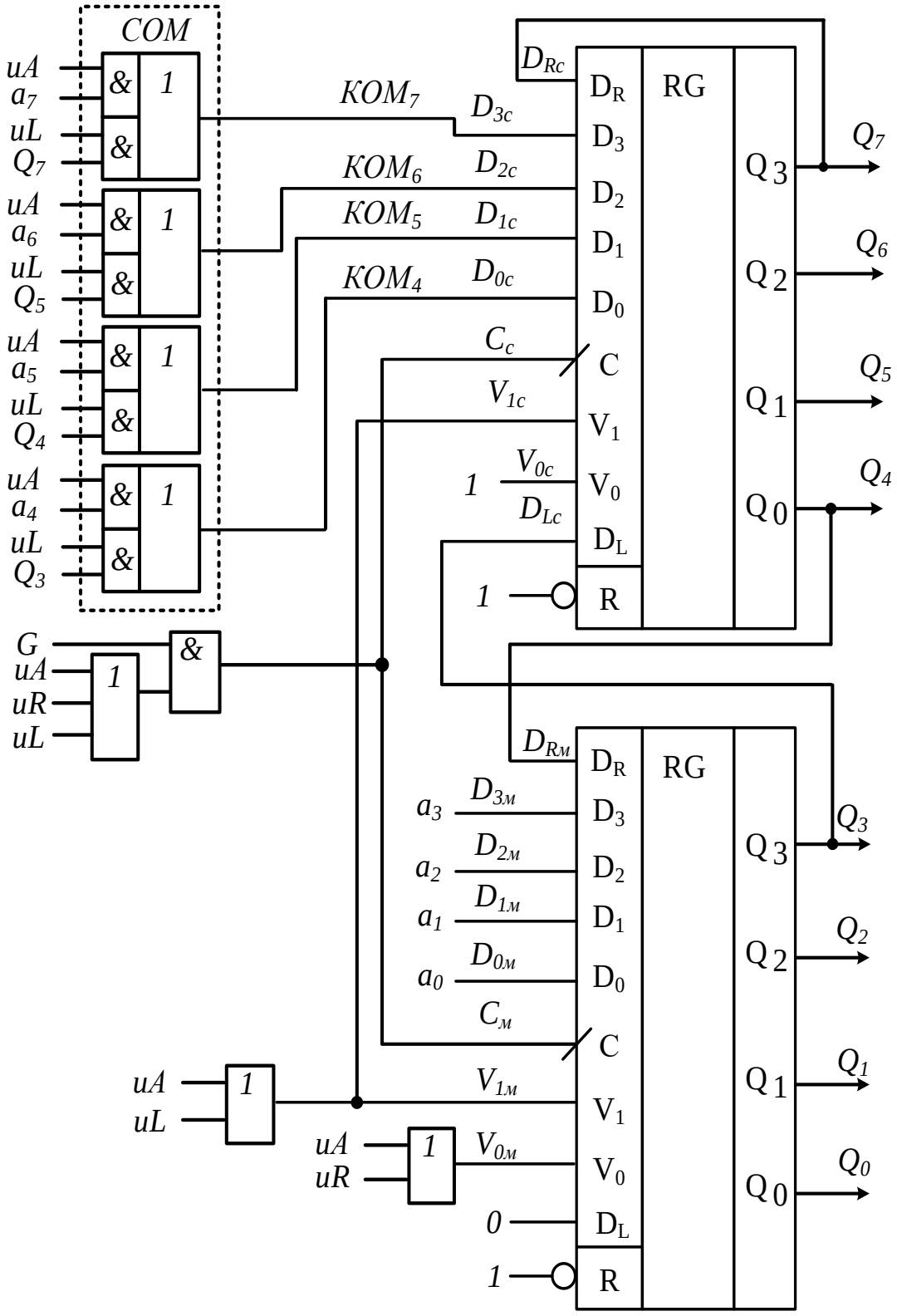


Рисунок 2.71 – Логічна схема підсумкового регістра (приклад 2.9)

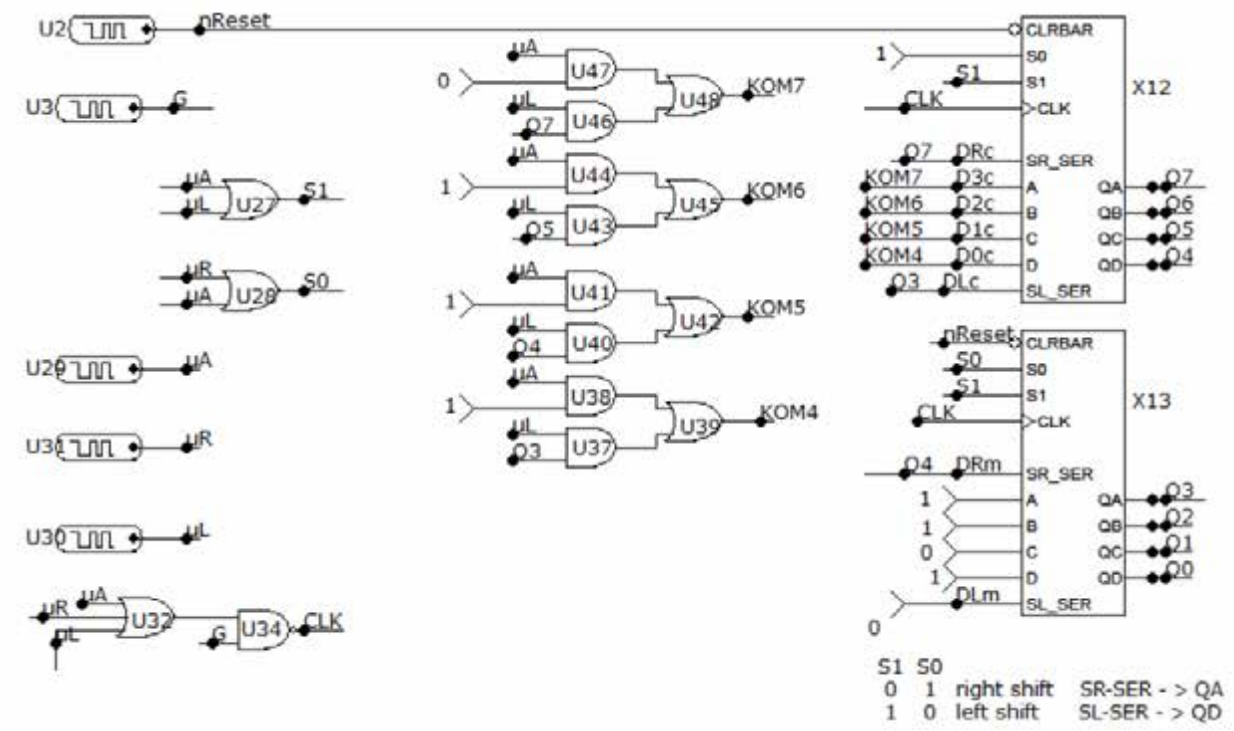


Рисунок 2.72 – Логічна схема для моделювання підсумкового регістра на базі *SN74194* (приклад 2.9)

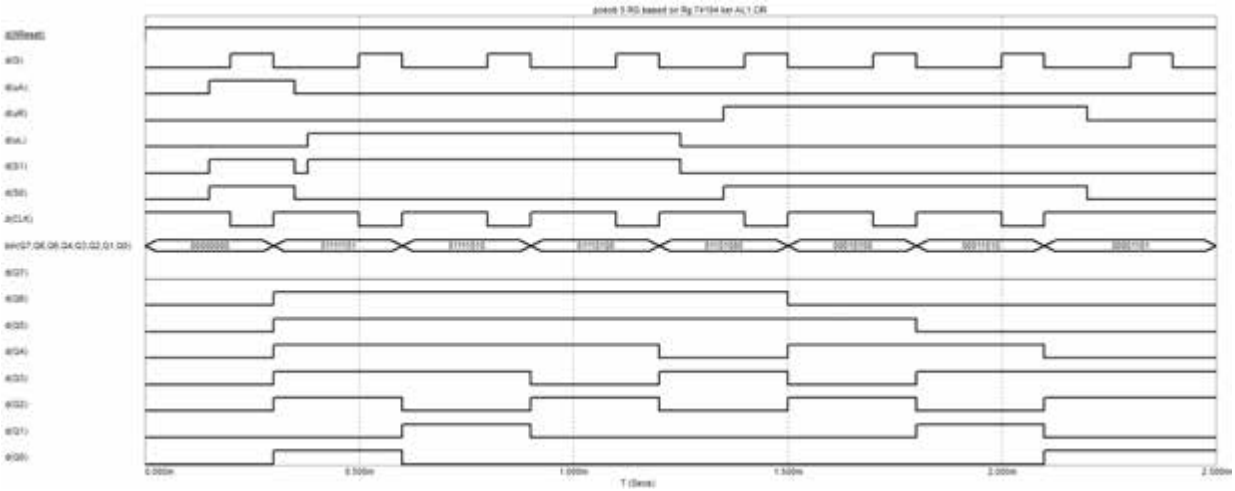


Рисунок 2.73 – Результати моделювання підсумкового регістра на базі *SN74194* (приклад 2.9)

Синтез регістрів четвертої групи (в складі лічильників) буде розглянуто пізніше.

На завершення зазначимо, що в разі виконання регістром хоча б однієї мікрооперації зсуву, в якості базових тригерів обов'язково необхідно використовувати тільки **непрозорі** [1] тригери.

Контрольні завдання та запитання

1. З якою метою виконується синтез регістрів на основі ІС регістрів?
2. В чому полягає різниця між синтезом регістрів на базі окремих тригерів і на базі ІС регістрів?
3. Приведіть приклад серії ІС на базі TTL-технології.
4. В чому полягає різниця між серіями *74xx* і *54xx*?
5. В чому полягає особливість ІС за *CMOS*-технологією?
6. В чому полягає особливість ІС за *BiCMOS* -технологією?
7. Яким чином позначаються ІС серії *SNxx*?
8. Які позначення використовуються в якості коду серії?
9. Якими властивостями характеризуються ІС в коді серії яких використовується літера «*L*»?
10. Якими властивостями характеризуються ІС в коді серії яких використовується літера «*S*»?
11. Якими властивостями характеризуються ІС в коді серії яких використовується літери «*LS*»?
12. Якими властивостями характеризуються ІС в коді серії яких використовується літера «*L*»?
13. Якими властивостями характеризуються ІС в коді серії яких використовується літери «*ALS*»?
14. Якими властивостями характеризуються ІС в коді серії яких використовується літера «*F*»?
15. Якими властивостями характеризуються ІС в коді серії яких використовується літери «*HC*»?

16. Якими властивостями характеризуються ІС в коді серії яких використовується літери «*HCT*»?

17. Якими властивостями характеризуються ІС в коді серії яких використовується літери «*AC*»?

18. Якими властивостями характеризуються ІС в коді серії яких використовується літери «*ABT*»?

19. Якими властивостями характеризуються ІС в коді серії яких використовується літери «*ACT*»?

20. Якими властивостями характеризуються ІС в коді серії яких використовується літери «*BCT*»?

21. Якими властивостями характеризуються ІС в коді серії яких використовується літери «*LVT*»?

22. Назвіть типи регістрів відповідно до способу синхронізації.

23. Назвіть складові частини позначення ІС *SN74ALS273*.

24. На скільки функціональних груп можна умовно розділити регістри в інтегральному виконанні?

25. Які мікрооперації виконують регістри першої групи?

26. Які мікрооперації виконують регістри другої групи?

27. Які мікрооперації виконують регістри третьої групи?

28. Як використовуються регістри четвертої групи?

29. До якої умовної групи належать реверсивні регістри?

30. Поясніть термін «реверсивний регістр».

31. Яким чином виконується синтез регістрів першої групи?

32. Назвіть етапи синтезу регістрів першої групи?

33. В чому полягає сенс синтезу регістрів на основі ІС регістрів?

34. Чим відрізняється процедура синтезу регістрів на основі окремих тригерів і регістрів другої і третьої груп на основі ІС регістрів?

35. Які базові тригери з точки зору користувача, як правило, використовуються у складі базових регістрів?

36. Яким чином будується таблиця мікрооперацій для регістрів з керованою синхронізацією?

37. Яким чином будується таблиця мікрооперацій для регістрів з некерованою синхронізацією?

38. Як визначити кількість рядків в таблиці мікрооперацій для регістрів з некерованою синхронізацією?

39. Як визначити кількість рядків в таблиці мікрооперацій для регістрів з керованою синхронізацією?

40. Як визначити кількість стовпчиків в таблиці мікрооперацій?

41. Чому при синтезі регістрів на базі ІС регістрів використовується об’єднана таблиця мікрооперацій?

42. В чому полягає методика синтезу підсумкового регістра на основі ІС регістрів?

43. Що необхідно визначити при виконанні аналізу принципів функціонування базового регістра?

44. Як визначити кількість базових регістрів?

45. Як виконувати заповнення об’єднаної таблиці мікрооперацій?

46. З яких входів необхідно починати заповнення об’єднаної таблиці мікрооперацій?

47. Як визначити функцію збудження базового регістра?

48. За якими правилами виконується мінімізація функцій збудження?

49. Які дії необхідно здійснити під час виконання першого кроку процедури синтезу підсумкового регістра?

50. Які дії необхідно здійснити під час виконання другого кроку процедури синтезу підсумкового регістра?

51. Які дії необхідно здійснити під час виконання третього кроку процедури синтезу підсумкового регістра?

52. Які дії необхідно здійснити під час виконання четвертого кроку процедури синтезу підсумкового регістра?

53. Які дії необхідно здійснити під час виконання п’ятого кроку процедури синтезу підсумкового регістра?

54. Яким чином виконується розробка логічної схеми пілсумкового регістра?

55. Поясніть вираз (2.18).

56. Що означає операція]... [?

57. Як кількість входів одного базового регістра?

58. Поясніть принцип функціонування ІС *SN74195.*

59. Поясніть принцип функціонування ІС *SN74194.*

60. Поясніть призначення входу «*R*» ІС *SN74195* і *SN74194*.

61. Поясніть призначення входу «*C*» ІС *SN74195* і *SN74194*.

62. Поясніть призначення входу «*V*» ІС *SN74195*.

63. Як спрацьовує регістр *SN74194*, якщо на вхід *V* надходить високий рівень напруги?

64. Як спрацьовує регістр *SN74194*, якщо на вхід *V* надходить низький рівень напруги?

65. Для чого використовуються входи «*J, K*» ІС *SN74195*?

66. Чого сигнал, який надходить на вхід *K* називається *nK*?

67. Для чого використовуються вихід $\overline{Q}_0$?

68. Які входи в регістрах *SN74195*, *SN74194* використовуються для паралельного завантаження даних?

69. Який вихід базового регістра вважається найстаршим?

70. Яка логічна змінна відповідає напрузі високого рівня при використанні позитивної логіки?

71. Яка логічна змінна відповідає напрузі низького рівня при використанні позитивної логіки?

72. Як організувати виконання арифметичного зсуву вправо на один розряд на базі регістра *SN74195*?

73. Як організувати виконання арифметичного зсуву вправо на два розряди на базі регістра *SN74195*?

74. Як організувати виконання циклічного зсуву вправо на два розряди на базі регістра *SN74195*?

75. Як організувати виконання циклічного зсуву вправо на один розряд на базі регістра *SN74195*?

76. Як організувати виконання логічного зсуву вправо на два розряди на базі регістра *SN74195* із заповненням розрядів, що звільнилися, нулями?

77. Як організувати виконання логічного зсуву вправо на один розряд на базі регістра *SN74195* із заповненням розряду, що звільнився, нулем або одиницею?

78. Яким чином використовувати входи *J* і *nK* в регістрі *SN74195*?

79. Для чого використовується вхід *R* в регістрах *SN74195* і *SN74194*?

80. Поясніть принцип роботи регістра *SN74195* по функціональній схемі на рис.2.46.

81. Поясніть, чому входи паралельного завантаження регістра *SN74195* позначені літерою «*D*», хоча відповідно до функціональної схеми (рис.2.46) базовими тригерами є *RCS*-тригери?

82. Яка розрядність регістра *SN74195*?

83. Яка розрядність регістра *SN74194*?

84. Яким чином визначена кількість рядків і стовпців таблиці мікрооперацій в прикладі 2.5?

85. Поясніть заповнення таблиці мікрооперацій для керуючого сигналу *uA* в прикладі 2.5.

86. Поясніть заповнення таблиці мікрооперацій для керуючого сигналу *uR* в прикладі 2.5.

87. Поясніть заповнення таблиці мікрооперацій для керуючого сигналу *uL* в прикладі 2.5.

88. Які клітини таблиці мікрооперацій в прикладі 2.5 для керуючого сигналу *uA* заповнюються символом «*»? Обґрунтуйте відповідь.

89. Які клітини таблиці мікрооперацій в прикладі 2.5 для керуючого сигналу *uR* заповнюються символом «*»? Обґрунтуйте відповідь.

90. Які клітини таблиці мікрооперацій в прикладі 2.5 для керуючого сигналу *uL* заповнюються символом «*»? Обґрунтуйте відповідь.

91. Поясніть рис.2.47.

92. Яким чином отримано вирази (2.19), (2.20)?

93. Які вирази описують комутатор в регістрі *SN74195*?

94. Як виконувати мінімізацію функцій збудження, використовуючи таблицю мікрооперацій?

95. Яким чином отримано вирази (2.21)?

96. Як побудована логічна схема підсумкового регістра на рис.2.48?

97. Як під час моделювання імітується вміст вхідної шини даних?

98. Поясніть використання сигналу ***nReset***?

99. Для чого використовуються елементи ***Fixed Digital*** в прикладі 2.5?

100. Прокоментуйте результати моделювання, приведені на рис.2.50.

101. Для чого використовується сигнал *G*?

102. Як визначити динамічні параметри підсумкового регістра на базі ІС *SN74195*?

103. Поясніть реалізацію арифметичного зсуву на рис.2.53.

104. Як реалізувати зсув вліво на один розряд на базі регістрів *SN74195*, якщо підсумковому регістру не потрібно виконувати зсув вправо?

105. Яким чином визначена кількість рядків і стовпців таблиці мікрооперацій в прикладі 2.6?

106. Поясніть заповнення таблиці мікрооперацій для керуючого сигналу *uA* в прикладі 2.6.

107. Поясніть заповнення таблиці мікрооперацій для керуючого сигналу *uR* в прикладі 2.6.

108. Поясніть заповнення таблиці мікрооперацій для керуючого сигналу *uL* в прикладі 2.6.

109. Які клітини таблиці мікрооперацій в прикладі 2.6 для керуючого сигналу *uA* заповнюються символом «*»? Обґрунтуйте відповідь.

110. Які клітини таблиці мікрооперацій в прикладі 2.6 для керуючого сигналу *uR* заповнюються символом «*»? Обґрунтуйте відповідь.

111. Які клітини таблиці мікрооперацій в прикладі 2.6 для керуючого сигналу *uL* заповнюються символом «*»? Обґрунтуйте відповідь.

112. Поясніть призначенні виводів регістра *SN74194*?

113. Як використовуються входи D_R і D_L регістра *SN74194*?

114. Як використовуються входи V_1 і V_0 в регістрі *SN74194*?

115. Прокоментуйте інформацію, приведену в табл.2.26?

116. Поясніть рис.2.57.

117. Поясніть функціонування базового регістра *SN74194* на рис.2.58.

118. Поясніть, чому входи паралельного завантаження регістра *SN74194* позначені літерою «*D*», хоча відповідно до функціональної схеми (рис.2.58) базовими тригерами є *RCS*-тригери?

119. Поясніть рис.2.59.

120. Поясніть рис.2.60.

121. Яким чином отримано вирази (2.22), (2.23)?

122. Які вирази описують комутатор в регістрі *SN74194* в прикладі 2.6?

123. Яким чином отримано вирази (2.24)?

124. Як побудована логічна схема підсумкового регістра на рис.2.61?

125. Поясніть результати моделювання на рис.2.63.

126. Як визначити динамічні параметри підсумкового регістра на базі ІС *SN74194*?

127. Яким чином визначена кількість рядків і стовпців таблиці мікрооперацій в прикладі 2.7?

128. Поясніть заповнення таблиці мікрооперацій для керуючого *uA* в прикладі 2.7.

129. Поясніть заповнення таблиці мікрооперацій для керуючого сигналу *uR* в прикладі 2.7.

130. Поясніть заповнення таблиці мікрооперацій для керуючого сигналу *uL* в прикладі 2.7.

131. Які клітини таблиці мікрооперацій в прикладі 2.7 для керуючого сигналу *uA* заповнюються символом «*»? Обґрунтуйте відповідь.

132. Які клітини таблиці мікрооперацій в прикладі 2.7 для керуючого сигналу *uR* заповнюються символом «*»? Обґрунтуйте відповідь.

133. Які клітини таблиці мікрооперацій в прикладі 2.7 для керуючого сигналу *uL* заповнюються символом «*»? Обґрунтуйте відповідь.

134. В чому полягає різниця в синтезі регістрів з керованою і некерованою синхронізацією?

135. Яким чином формується сигнал *nU*?

136. Поясніть вирази (2.25), (2.26).

137. Поясніть формування таблиці 2.32.

138. Як побудована логічна схема підсумкового регістра на рис.2.64?

139. Поясніть результати моделювання на рис.2.66.

140. Виконайте порівняльний аналіз логічних схем підсумкових регістрів на рис.2.48 і рис.2.64.

141. Який тип синхронізації доцільно використовувати для синтезу підсумкового регістра на основі базових регістрів другої групи?

142. Яким чином визначена кількість рядків і стовпців таблиці мікрооперацій в прикладі 2.8?

143. Поясніть заповнення таблиці мікрооперацій для керуючого сигналу *uA* в прикладі 2.8.

144. Поясніть заповнення таблиці мікрооперацій для сигналу *uR* в прикладі 2.8.

145. Поясніть заповнення таблиці мікрооперацій для сигналу *uL* в прикладі 2.8.

146. Які клітини таблиці мікрооперацій в прикладі 2.8 для керуючого сигналу *uA* заповнюються символом «*»? Обґрунтуйте відповідь.

147. Які клітини таблиці мікрооперацій в прикладі 2.8 для керуючого сигналу *uR* заповнюються символом «*»? Обґрунтуйте відповідь.

148. Які клітини таблиці мікрооперацій в прикладі 2.8 для керуючого сигналу *uL* заповнюються символом «*»? Обґрунтуйте відповідь.

149. Поясніть формування таблиці 2.33.

150. Поясніть вирази (2.27), (2.28).

151. Як побудована логічна схема підсумкового регістра на рис.2.67?

152. Поясніть результати моделювання на рис.2.69.

153. Чому в схемі регістра на рис.2.67 відсутня схема формування сигналу *nU*?

154. Виконайте порівняльний аналіз логічних схем підсумкових регістрів на рис.2.61 і рис.2.67.

155. Який тип синхронізації доцільно використовувати для синтезу підсумкового регістра на основі базових регістрів третьої групи?

156. В чому полягає особливість регістра, який синтезовано в прикладі 2.9?

157. Поясніть заповнення таблиці мікрооперацій для сигналу *uL* в прикладі 2.9.

158. Чому необхідно налаштовувати базові регістри в прикладі 2.9 на виконання різних мікрооперацій під час реалізації арифметичного зсуву вліво?

159. Поясніть рис.2.70.

160. Поясніть формування таблиці 2.34.

161. Поясніть вирази (2.29), (2.30).

162. Як побудована логічна схема підсумкового регістра на рис.2.71?

163. Поясніть результати моделювання на рис.2.73.

164. В чому полягає різниця між регістрами з керованою і некерованою синхронізацією?

165. Чи можуть в складі регістра використовуватися різні типи базових регістрів?

166. Як відбувається організація некерованої синхронізації в регістрах?

167. Вкажіть на джерело формування сигналу *nU*?

168. Які складові вузли підсумкового регістра є комбінаційними?

169. Поясніть термін «функція збудження».

170. В яких випадках в схемі підсумкового регістра не використовується комутатор?

171. Яку мікрооперацію необхідно виконувати за появи активного значення сигналу *nU*?

172. Поясніть наявність ліній зворотного зв'язку у складі регістра з некерованою синхронізацією.

173. Який спосіб завантаження даних за допомогою зсувів використовується в сучасних комп'ютерних системах?

174. На базі регістрів *SN74195* виконати синтез 8-бітного регістра з некерованою синхронізацією в будь-якому базисі логічних елементів. Регістр виконує такі мікрооперації: *uA*: *Rg* := *A*; *uR*: *Rg* := *11.R2*(*Rg*) ; *uL*: *Rg* := *L1.1*(*Rg*).

175. На базі регістрів *SN74195* виконати синтез 8-бітного регістра з керованою синхронізацією в будь-якому базисі логічних елементів. Регістр виконує такі мікрооперації: *uA*: *Rg* := *A*; *uR*: *Rg* := *11.R2*(*Rg*) ; *uL*: *Rg* := *L1.1*(*Rg*).

176. На базі регістрів *SN74194* виконати синтез 8-бітного регістра з некерованою синхронізацією в будь-якому базисі логічних елементів. Регістр виконує такі мікрооперації: *uA*: *Rg* := *A*; *uR*: *Rg* := *11.R2*(*Rg*) ; *uL*: *Rg* := *L1.1*(*Rg*).

177. На базі регістрів *SN74194* виконати синтез 8-бітного регістра з керованою синхронізацією в будь-якому базисі логічних елементів. Регістр виконує такі мікрооперації: *uA*: *Rg* := *A*; *uR*: *Rg* := *11.R2*(*Rg*) ; *uL*: *Rg* := *L1.1*(*Rg*).

178. В яких інтегральних схемах регістрів використовується некерована синхронізація?

3 ЛІЧИЛЬНИКИ

Лічильник (***Counter***) – цифровий пристрій, який призначений для виконання підрахунку кількості вхідних сигналів.

Принцип роботи будь-якого лічильника випливає з вищезазначеного визначення, тобто при надходженні активного значення вхідного сигналу код вихідного стану лічильника змінюється, збільшуючись або зменшуючись в залежності від способу використання, як правило, на одиницю (вихідний код лічильника визначає кількість підрахованих вхідних сигналів). Таким чином, як правило, лічильник виконує мікрооперацію «інкремент» або «декремент». При цьому вхідні сигнали можуть бути розподілені в часі як завгодно, тобто можуть бути як періодичними, так і аперіодичними. Параметри цих сигналів визначаються вимогами апаратних засобів, за допомогою яких побудований лічильник.

В комп'ютерних системах лічильники використовуються для таких застосувань:

- при виконанні ітераційних арифметичних операцій множення та ділення для підрахунку кількості виконаних ітерацій;

- при організації обчислювальних процесів в якості лічильника адреси команди, що буде виконуватися (***Program Counter, Instruction Pointer***);

- в мікроконтролерних системах в якості різноманітних годинників, таймерів реального часу тощо.

В будь-якому лічильнику одним з основних параметрів є так званий модуль ліку (модуль лічби, модуль рахування тощо), який будемо позначати літерою «*M*».

За допомогою модулю ліку визначається максимальна можлива кількість вхідних імпульсів, яку лічильник може підрахувати.

Прикладом лічильника є будь-який годинник, до складу якого входять кілька лічильників: лічильники секунд, хвилин, годин тощо. В якості ілюстрації розглянемо, наприклад, лічильник секунд. Максимальна кількість

секунд, яку може підрахувати цей лічильник, вочевидь, дорівнює *60*. Таким чином, модуль ліку цього лічильника теж складає *60* (*M = 60*). Крім того, це означає, що лічильник секунд може перебувати в одному з *60* станів.

В загальному випадку можна сказати, що будь-який лічильник, який має модуль ліку *M* (далі будемо говорити: лічильник за модулем *M*) може підрахувати *M* вхідних сигналів і має *M* станів. Причому знову ж таки лічильник секунд формує значення секунд в діапазоні від *0* до *59*, тобто в загальному випадку будь-який лічильник формує вихідний код від *0* до *M-1*, тобто *M* різних значень. Після того, як вихідний код лічильника досягне значення *M-1*, за наступним вхідним сигналом лічильник повертається в стан *0* (в загальному випадку – в початковий стан).

Таким чином, маючи на увазі циклічність роботи лічильника, достатньо далі розглядати тільки один період його роботи.

В проміжку між активними значеннями вхідних сигналів лічильник перебуває в стані збереження інформації. Це означає, що до складу будь- якого лічильника входять тригери. Однак, на відміну від регістрів, в яких для синтезу достатньо розглядати один або кілька сусідніх розрядів, лічильники характеризуються більш складними логічними зв’язками між розрядами.

Як вже було зазначено вище, до складу лічильників входять тригери, причому, як і в регістрах зсуву, в лічильниках **обов’язково** необхідно використовувати тільки **непрозорі** (двоступеневі або з динамічним керуванням) тригери [1], тобто лічильники спрацьовують за активним фронтом вхідного сигналу.

Таким чином, активний фронт вхідного сигналу $\boldsymbol{U_c}$ (такий сигнал ще називається лічильним сигналом) здійснює переключення базових тригерів лічильника, змінюючи його стан, тобто стани базових тригерів формують стан лічильника, який відповідає двійковому коду на виходах базових тригерів.

Крім того, в зв'язку з тим, що, лічильники, як правило, виконують мікрооперацію «інкремент» або «декремент», то між розрядами лічильників можуть виникати сигнали переносу (для мікрооперації «інкремент») або позики (для мікрооперації «декремент») між розрядами.

При цьому, як і під час виконання звичайних арифметичних операцій, обробка операндів при додаванні або відніманні одиниці починається з молодших розрядів, тобто вхідний лічильний сигнал U_c надходить в базовий тригер молодшого розряду лічильника в асинхронних лічильниках або відразу на всі тригери в синхронних лічильниках.

Класифікацію лічильників будемо розглядати на основі таких класифікаційних ознак:

за напрямом ліку розрізняють:

- підсумовувальні лічильники (***Up Counter***);
- віднімальні лічильники (***Down Counter***);

за способом синхронізації розрізняють:

- синхронні лічильники (***Synchronous Counter***);
- асинхронні лічильники (***Asynchronous Counter***);

за способом організації переносу між розрядами розрізняють:

- лічильники з безпосереднім переносом (***Ripple Counter***);
- з паралельним трактом розповсюдження переносу (паралельний перенос);
- лічильники з послідовним трактом розповсюдження переносу (наскрізний перенос, транзитний перенос);
- лічильники з груповим переносом;

за модулем ліку розрізняють:

- двійкові лічильники (***Binary Counter***), в яких модуль ліку є кратним ступеню 2);
- десяткові лічильники (***Decimal Counter***), в яких модуль ліку є кратним ступеню *10*, як правило, для таких лічильників *M = 10*);

- лічильники з іншими модулями ліку (наприклад, в годинниках $M = 60$, $M = 24$).

Відповідно до класифікації за напрямом ліку переважна більшість підсумовувальних лічильників виконують мікрооперацію «інкремент», а віднімальні лічильники – мікрооперацію «декремент».

Крім того, до лічильників, які розрізняються за напрямом ліку, іноді відносять також так звані реверсивні лічильники (***Up-Down Counter***), які відповідно до керуючого сигналу можуть функціонувати як підсумовувальними, так і віднімальними лічильниками. Але ми не будемо вважати реверсивні лічильники окремим видом лічильників в зв'язку з тим, що фактично будь-який реверсивний лічильник є поєднанням підсумовувального і віднімального лічильників, які синтезуються окремо, а потім об'єднуються в загальну схему.

Як і для будь-яких цифрових пристроїв, функціонуванню лічильників притаманні певні динамічні параметри, серед яких будемо використовувати параметри, приведені нижче.

Час переключення лічильника $t_{\text{п}}$ (час спрацьовування, час встановлення, час затримки) – проміжок часу, який вимірюється від моменту появи активного фронту лічильного сигналу на вході до моменту завершення перехідних процесів на виходах базових тригерів лічильника.

Час формування переносу $t_{\text{фп}}$ – величина проміжку часу від моменту появи активного фронту лічильного сигналу на вході до моменту формування активного значення вихідного сигналу переносу зі старшого розряду лічильника.

Максимальна частота надходження вхідних сигналів f_{cmax} визначається мінімальним періодом надходженням цих сигналів, впродовж якого зберігається коректне функціонування лічильника.

3.1 Способи організації переносів в двійкових лічильниках

Як вже було зазначено вище, двійкові лічильники мають модуль ліку, який може бути представлений у вигляді ступеню двійки. Розглянемо різні способи організації переносів між розрядами двійкових лічильників.

3.1.1 Двійкові лічильники з паралельним трактом розповсюдження переносу

3.1.1.1 Асинхронні двійкові лічильники з паралельним трактом розповсюдження переносу

Синтез асинхронних лічильників з паралельним трактом розповсюдження переносу (далі будемо використовувати термін «лічильник з паралельним переносом») необхідно виконувати в такій послідовності:

1. Розрахунок кількості базових тригерів у складі лічильника.
2. Розробка таблиці переходів, що описує функціонування лічильника.
3. Визначення функцій збудження базових тригерів.
4. Мінімізація функцій збудження базових тригерів.
5. Побудова логічної схеми лічильника.
6. Перевірка коректності функціонування лічильника.
7. Визначення динамічних параметрів лічильника.

Кроки процедури синтезу *2 - 5* відносяться до класичного способу структурного синтезу цифрових автоматів.

Розглянемо більш детально кроки синтезу.

Кількість базових тригерів лічильника (перший крок) розраховується за виразом:

$$N_{\text{тр}} =] \log_2 M [, \qquad (3.1)$$

де M – модуль ліку;

]…[– операція округлення до найближчого більшого цілого числа в разі отримання дробового результату.

Загальний вигляд таблиці переходів лічильника приведений в табл.3.1.

В якості вхідних сигналів в таблиці переходів використовуються лічильний сигнал U_c та n станів базових тригерів, де $n = n_{mp}$, тобто загальна кількість вхідних змінних дорівнює $n + 1$, тобто в таблиці буде 2^{n+1} рядків.

В таблиці використовуються такі скорочення:

- ФЗ БТ – функції збудження базових тригерів лічильника;

- С – сигнал синхронізації базових тригерів;

- ІВ БТ – інформаційні входи базових тригерів.

Кількість інформаційних входів базових тригерів відповідає типу цих тригерів, а при використанні асинхронних непрозорих тригерів колонка «*С*» в таблиці переходів відсутня.

Таблиця 3.1 – Загальний вигляд таблиці переходів лічильника

Номери наборів	U_c	Попередній стан					Наступний стан					ФЗ БТ		
		Q^t_{n-1}	…	Q^t_i	…	Q^t_0	Q^{t+1}_{n-1}	…	Q^{t+1}_i	…	Q^{t+1}_0	С	ІВ БТ	
0	*0*	*0*	*0*	*0*	*0*	*0*								
…	*0*	…	…	…	…	…								
i	*0*	…	…	…	…	…								
…	*0*	…	…	…	…	…								
2^n-1	*0*	*1*	*1*	*1*	*1*	*1*								
2^n	*1*	*0*	*0*	*0*	*0*	*0*								
…	*1*	…	…	…	…	…								
2^n+i	*1*	…	…	…	…	…								
…	*1*	…	…	…	…	…								
2^{n+1}-1	*1*	*1*	*1*	*1*	*1*	*1*								

Розглянемо правила заповнення таблиці переходів, в результаті виконання яких визначається наступний стан лічильника Q_i^{t+1}, тобто визначаються стани базових тригерів в наступному такті роботи лічильника (другий крок процедури синтезу).

Таблиця заповнюється по рядках без врахування інших рядків. В першій половині таблиці 3.1 лічильний сигнал U_c приймає неактивне значення, яке позначається нулем ($U_c = 0$). Це означає, що лічильник

перебуває в стані збереження інформації ($Q_i^{t+1} = Q_i^t$), тобто кожний рядок таблиці для стовпців «Наступний стан» повторює значення відповідних стовпців «Попередній стан».

В другій половині таблиці лічильний сигнал U_c приймає активне значення, яке позначається одиницею ($U_c = 1$). Це означає, що лічильник виконує підрахунок вхідних сигналів. Для підсумовувальних (віднімальних) лічильників двійковий код, що відповідає наступному стану, як правило, на одиницю більше (менше) коду попереднього стану. Виняток становить випадок, коли підсумовувальний лічильник досягає стану *M-1*, а віднімальний – нульового стану. В цьому разі наступним станом підсумовувального лічильника буде нульовий стан, а віднімального – стан *M- 1*.

Далі виконаємо третій крок процедури синтезу, який полягає в заповненні стовпців функції збудження базових тригерів таблиці переходів. Визначення функцій збудження тригерів розглядалося в [1], розділі 1 та підрозділі 2.1.1 цього посібника.

Виконання наступних кроків процедури синтезу за методикою нічим не відрізняються від аналогічних кроків під час синтезу тригерних схем на основі тригерів (розділ 1) або синтезу регістрів на основі тригерів (підрозділ 2.1.1).

Розглянемо процедуру синтезу асинхронних двійкових лічильників з паралельним переносом на прикладі.

Приклад 3.1. Виконати синтез асинхронного підсумовувального лічильника за модулем *16* з паралельним трактом розповсюдження переносу на основі асинхронних *T*- тригерів.

Розв’язок

На першому кроці розрахуємо кількість базових тригерів лічильника. Відповідно до умови завдання $M = 16$, тобто відповідно до виразу (3.1) $n_{mp} =] \log_2 16 [= 4$. Це означає, що в складі лічильника будуть використовуватися чотири *T*-тригери.

Результати виконання другого і третього кроків синтезу лічильника представлені в табл.3.2.

Таблиця 3.2 – Таблиця переходів лічильника (приклад 3.1)

Номери наборів	U_c	Попередній стан				Наступний стан				ФЗ БТ			
		Q_3^t	Q_2^t	Q_1^t	Q_0^t	Q_3^{t+1}	Q_2^{t+1}	Q_1^{t+1}	Q_0^{t+1}	T_3	T_2	T_1	T_0
1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	***0***	***0***	***0***	***0***
1	*0*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*1*	***0***	***0***	***0***	***0***
2	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*1*	*0*	***0***	***0***	***0***	***0***
3	*0*	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	***0***	***0***	***0***	***0***
4	*0*	*0*	*1*	*0*	*0*	*0*	*1*	*0*	*0*	***0***	***0***	***0***	***0***
5	*0*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	***0***	***0***	***0***	***0***
6	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	***0***	***0***	***0***	***0***
7	*0*	*0*	*1*	*1*	*1*	*0*	*1*	*1*	*1*	***0***	***0***	***0***	***0***
8	*0*	*1*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	***0***	***0***	***0***	***0***
9	*0*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*1*	***0***	***0***	***0***	***0***
10	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	***0***	***0***	***0***	***0***
11	*0*	*1*	*0*	*1*	*1*	*1*	*0*	*1*	*1*	***0***	***0***	***0***	***0***
12	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	***0***	***0***	***0***	***0***
13	*0*	*1*	*1*	*0*	*1*	*1*	*1*	*0*	*1*	***0***	***0***	***0***	***0***
14	*0*	*1*	*1*	*1*	*0*	*1*	*1*	*1*	*0*	***0***	***0***	***0***	***0***
15	*0*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	***0***	***0***	***0***	***0***
16	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	***0***	***0***	***0***	***1***
17	*1*	*0*	*0*	*0*	*1*	*0*	*0*	*1*	*0*	***0***	***0***	***1***	***1***
18	*1*	*0*	*0*	*1*	*0*	*0*	*0*	*1*	*1*	***0***	***0***	***0***	***1***
19	*1*	*0*	*0*	*1*	*1*	*0*	*1*	*0*	*0*	***0***	***1***	***1***	***1***
20	*1*	*0*	*1*	*0*	*0*	*0*	*1*	*0*	*1*	***0***	***0***	***0***	***1***
21	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*1*	*0*	***0***	***0***	***1***	***1***
22	*1*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*1*	***0***	***0***	***0***	***1***
23	*1*	*0*	*1*	*1*	*1*	*1*	*0*	*0*	*0*	***1***	***1***	***1***	***1***
24	*1*	*1*	*0*	*0*	*0*	*1*	*0*	*0*	*1*	***0***	***0***	***0***	***1***
25	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*1*	*0*	***0***	***0***	***1***	***1***
26	*1*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*1*	***0***	***0***	***0***	***1***
27	*1*	*1*	*0*	*1*	*1*	*1*	*1*	*0*	*0*	***0***	***1***	***1***	***1***
28	*1*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*1*	***0***	***0***	***0***	***1***
29	*1*	*1*	*1*	*0*	*1*	*1*	*1*	*1*	*0*	***0***	***0***	***1***	***1***
30	*1*	*1*	*1*	*1*	*0*	*1*	*1*	*1*	*1*	***0***	***0***	***0***	***1***
31	*1*	*1*	*1*	*1*	*1*	*0*	*0*	*0*	*0*	***1***	***1***	***1***	***1***

Під час виконання другого кроку синтезу необхідно відповідно до інформації про попередній стан лічильника (стовпці 3-6 таблиці 3.2: Q_3^t, Q_2^t, Q_1^t, Q_0^t) визначити нові стани базових тригерів лічильника, сукупність значень яких утворюють наступний стан лічильника (стовпці 7-10: Q_3^{t+1}, Q_2^{t+1}, Q_1^{t+1}, Q_0^{t+1}).

Як вже відзначалося раніше, перша половина таблиці (рядки 0-15) відповідає відсутності активного значення лічильного сигналу, що позначається логічним нулем ($U_c = 0$). В цьому випадку лічильник перебуває в стані збереження інформації, тобто стани базових тригерів не змінюються (в кожному рядку таблиці новий стан лічильника не відрізняється від попереднього стану). Наприклад, для вхідного набору 5 попередній і наступний стани лічильника складають двійковий код 0101_2 ($Q_3^{t+1} = Q_3^t = 0$; $Q_2^{t+1} = Q_2^t = 1$; $Q_1^{t+1} = Q_1^t = 0$; $Q_0^{t+1} = Q_0^t = 1$).

За активного значення лічильного сигналу U_c (друга половина таблиці: рядки 16-31) двійковий код нового стану на одиницю більше коду попереднього стану. Наприклад, для рядка 27 код попереднього стану відповідає значенню вихідних сигналів базових тригерів Q_3^t, Q_2^t Q_1^t Q_0^t і дорівнює 1011_2 (колонки 3-6), що відповідає десятковому еквіваленту 11_{10}. Таким чином, в цьому випадку код наступного стану лічильника становить 12_{10}, що відповідає значенню вихідних сигналів базових тригерів Q_3^{t+1} Q_2^{t+1} Q_1^{t+1} $Q_0^{t+1} = 1100_2$ (колонки 7-10).

Окремо розглянемо останній рядок таблиці 3.2. Для цього випадку код стану лічильника відповідає найбільшому можливому значенню 15_{10} (M-1), тобто Q_3^t, Q_2^t Q_1^t $Q_0^t = 1111_2$ (колонки 3-6). Це означає, що за активним значенням сигналу U_c лічильник повертається в нульовий стан Q_3^{t+1} Q_2^{t+1} Q_1^{t+1} $Q_0^{t+1} = 0000_2$ (колонки 7-10).

Далі виконаємо визначення функцій збудження (стовпчики 11-14) базових тригерів (третій крок процедури синтезу). Визначення функцій

збудження тригерів детально розглянуто в [1]. В даному завданні базовими тригерами є асинхронні *T*-тригери. Для визначення функцій збудження *i*-того тригера необхідно проаналізувати стан цього тригера в попередній і наступний моменти часу Q_i^t і Q_i^{t+1}. Відповідно до принципу функціонування асинхронного *T*-тригера [1], збереження стану тригера відбувається за нульового значення сигналу на вході *T*, тобто, якщо *i*-тий тригер не переключається, то $T_i = 0$. За наявності активного фронту сигналу на вході *T,* тригер переключається в протилежний стан, тобто, у випадку, якщо $Q_i^t \neq Q_i^{t+1}$, то $T_i = 1$ (в таблиці 3.2 цей випадок показаний сірим кольором в рядку 27 для виходу Q_1).

Після цього за допомогою карт Карно виконаємо мінімізацію функцій збудження базових тригерів (четвертий крок процедури синтезу). При цьому функція збудження T_0 для тригера Q_0 може бути отримана без використання карти Карно в зв'язку з тим, що колонки 2 і 14 є однаковими, тобто можна записати $T_0 = U_c$. Карти Карно для мінімізації функцій збудження T_1, T_2 і T_3 приведені відповідно на рис.3.1, 3.2, 3.3.

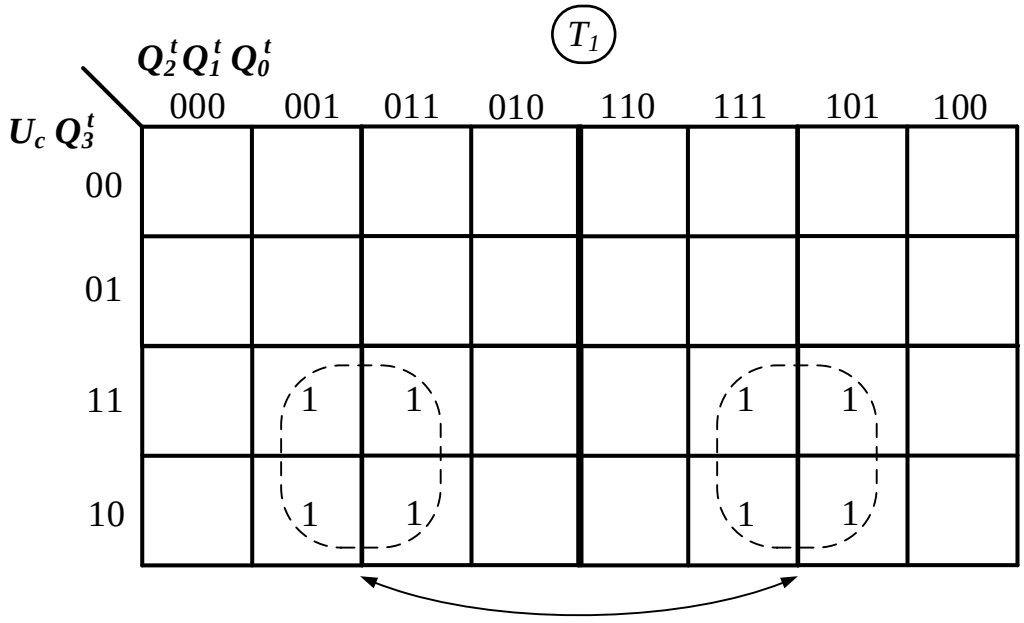


Рисунок 3.1 – Карта Карно для визначення ФЗ тригера Q_1 (приклад 3.1)

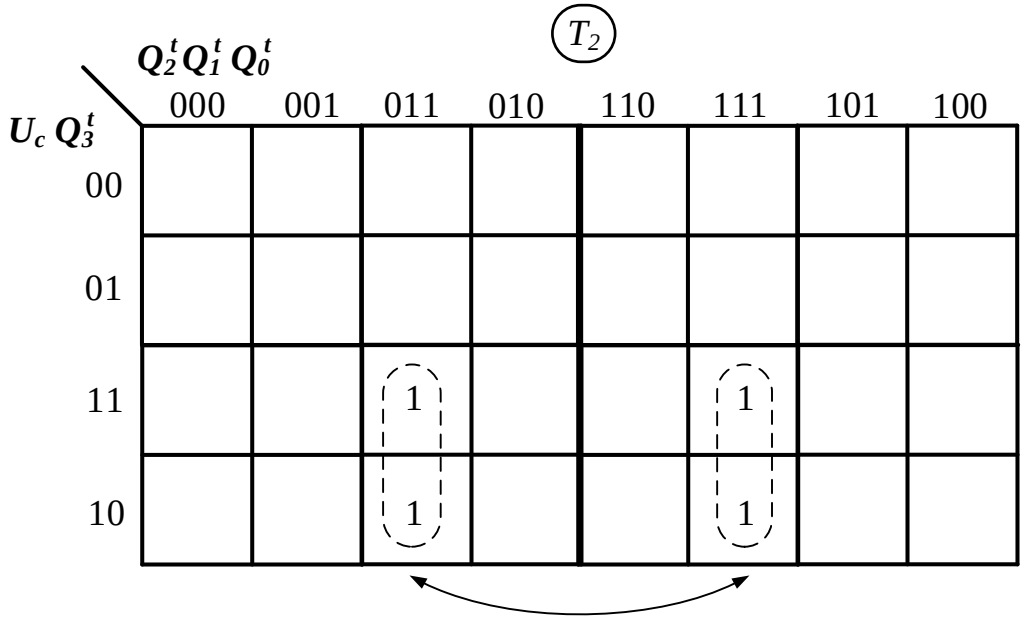


Рисунок 3.2 – Карта Карно для визначення ФЗ тригера Q_2 (приклад 3.1)

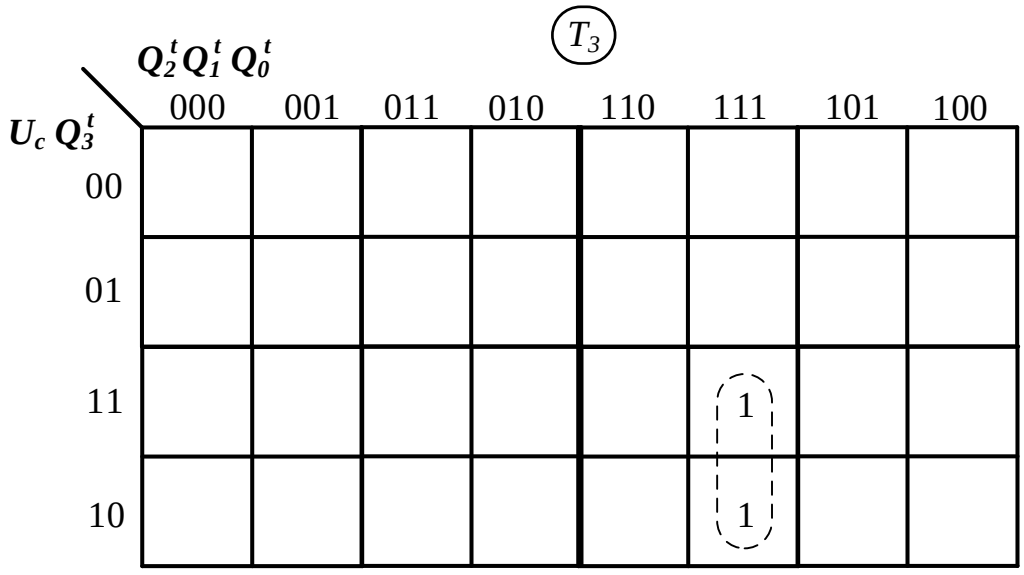


Рисунок 3.3 – Карта Карно для визначення ФЗ тригера Q_3 (приклад 3.1)

В результаті мінімізації отримаємо:

$$T_1 = U_c \cdot Q_0; \qquad T_2 = U_c \cdot Q_0 \cdot Q_1; \qquad T_3 = U_c \cdot Q_0 \cdot Q_1 \cdot Q_2; \quad (3.2)$$

Відповідно до виразу $T_0 = U_c$ та виразів (3.2) побудуємо функціональну схему заданого лічильника (п’ятий крок процедури синтезу), яка представлена на рис.3.4.

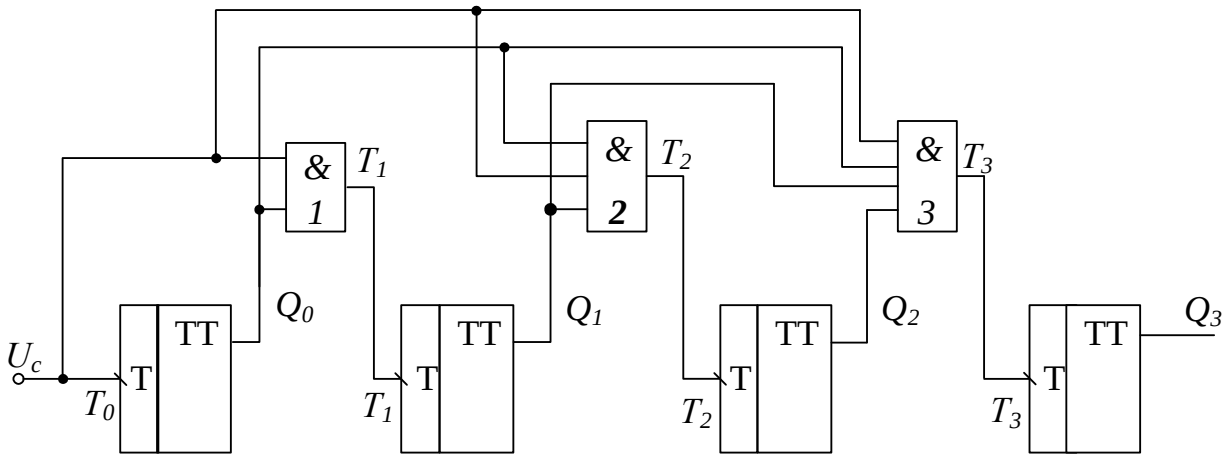


Рисунок 3.4 – Функціональна схема підсумовувального асинхронного лічильника за модулем 16 з паралельним трактом розповсюдження переносу (приклад 3.1)

На цьому місці перервемо поки виконання прикладу 3.1 та розглянемо деякі властивості лічильника, які можна застосовувати після отримання і аналізу функціональної схеми лічильника.

У функціональній схемі на рис.3.4 можна виділити 2 структурні складові:

- базові тригери Q_3, Q_2, Q_1, Q_0, де відповідно до таблиці 3.2 розряд Q_3 є найстаршим (***the most significant bit***);

- тракт розповсюдження переносу, реалізований на елементах I_1, I_2, I_3, на виходах який формуються функції збудження T_1, T_2, T_3 для базових тригерів Q_1, Q_2, Q_3 відповідно (фактично сигнали T_i є сигналами переносу в i-тий розряд тригера Q_i).

В загальному випадку схема лічильника, як правило, ще доповнюється схемою формування переносу зі старшого розряду T_4, який не було враховано під час синтезу лічильника, і який фактично є функцією збудження для четвертого (Q_4) неіснуючого розряду. Це було пов’язано з відносно великим обсягом таблиці переходів, з якої можна зрозуміти, що для синтезу лічильників з більш великими модулями ліку необхідно збільшувати обсяг таблиці переходів вдвічі для додавання кожного додаткового базового тригера (так для синтезу лічильника за модулем $M = 32$ вже необхідно

використовувати таблицю переходів на 64 рядки, для $M = 64$ – на 128 рядків і т.д.).

Аналізуючи функції збудження T_0, T_1, T_2, T_3 з виразів (3.2), можна зробити припущення, що логічний вираз для T_4 буде виглядати таким чином: $T_4 = U_c \cdot Q_0 \cdot Q_1 \cdot Q_2 \cdot Q_3$; для T_5 : $T_5 = U_c \cdot Q_0 \cdot Q_1 \cdot Q_2 \cdot Q_3 \cdot Q_4$, тобто в загальному випадку для будь-якого модулю ліку можна отримати:

$$T_i = U_c \cdot Q_0 \cdot Q_1 \cdot \ldots \cdot Q_{i-1} = U_c \cdot \&_{j=0}^{i-1} Q_j. \qquad (3.3)$$

Виконаємо розрахунок динамічних параметрів лічильника, використовуючи схему на рис.3.4.

Відповідно до виразів (3.3) всі сигнали переносу залежать від станів тригерів лічильника, які були встановлені під час попереднього спрацьовування лічильника, та сигналу U_c, тобто всі сигнали переносу формуються паралельно, в результаті чого можна визначити час переключення лічильника $t_п$:

$$t_п = max(t_{I1},\ t_{I2},\ \ldots,\ t_{In}) + max(t_{mp1},\ t_{mp2},\ \ldots,\ t_{mpn}), \qquad (3.4)$$

де t_{Ii} – час спрацьовування i-того логічного елемента I;

t_{mpi} – час спрацьовування i-того базового тригера;

n – кількість тригерів.

Далі будемо вважати, що час спрацьовування всіх елементів I (t_I), а також базових тригерів (t_{mp}) є однаковим в межах кожної групи цих елементів, тому час переключення лічильника відповідно до (3.4) можна визначити за виразом $t_п = t_I + t_{mp}$.

В зв’язку з тим, що всі переноси формуються паралельно, то час формування переносу зі старшого розряду $t_{фп} = t_I$.

Таким чином, час спрацьовування лічильника та час формування переносу визначаються за виразами:

$$t_п = t_I + t_{mp}; \qquad t_{фп} = t_I. \qquad (3.5)$$

Для визначення максимальної частоти надходження сигналів синхронізації f_{Ucmax} необхідно визначити мінімальний період надходження цього сигналу T_{Ucmin}, які пов'язані співвідношенням $f_{Ucmax} = 1/T_{Ucmin}$.

Часова діаграма для визначення T_{Ucmin} приведена на рис.3.5.

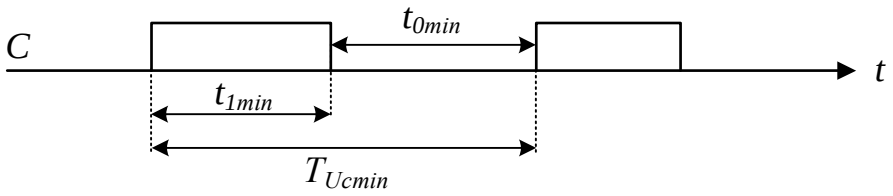


Рисунок 3.5 – Часова діаграма для визначення T_{Ucmin}

З рис.3.5 можна побачити, що період сигналу U_c складається з суми мінімальних значень тривалості одиничного (t_{1min}) і нульового (t_{0min}) рівнів цього сигналу, тобто $T_{Ucmin} = t_{1min} + t_{0min}$. Якщо, наприклад, лічильник побудований на базі двоступеневих тригерів зі спрацьовуванням за заднім фронтом сигналу синхронізації, то величина t_{1min} визначається часом спрацьовування першого ступеня базового тригера t_1 [1], а величина t_{0min} складається з часів спрацьовування інвертора t_{HI} між ступенями базового тригера, другого ступеня t_2 цього тригера (нагадаємо, що $t_{HI} + t_2$ складає час спрацьовування двоступеневого тригера зі спрацьовуванням за заднім фронтом сигналу синхронізації [1]) та часу спрацьовування елементів тракту розповсюдження переносу $t_{\text{фп}}$, тобто:

$$T_{Ucmin} = t_{1min} + t_{0min} = t_1 + t_{HI} + t_2 + t_{\text{фп}} = t_1 + t_{mp} + t_{\text{фп}}.$$

Далі продовжимо виконувати приклад 3.1.

Для перевірки коректності отриманої функціональної схеми виконаємо моделювання функціонування лічильника (шостий крок процедури синтезу) з урахуванням наявності переносу зі старшого розряду відповідно до (3.3). Схема для моделювання лічильника приведена на рис.3.6.

Для реалізації лічильника використовуються *JK*-тригери *SN74H106*, які відповідно до рис.1.14,б перетворені на *T*-тригери. На асинхронні входи *R* (*1CLRBAR*) тригерів підключено сигнал *nReset*, який на початку моделювання

приймає активне (нульове) значення, встановлюючи базові тригери лічильника в нульовий стан, а далі переключається до неактивного стану. На асинхронні входи *S* (*1PREBAR*) базових тригерів підключено неактивне (одиничне) значення сигналу.

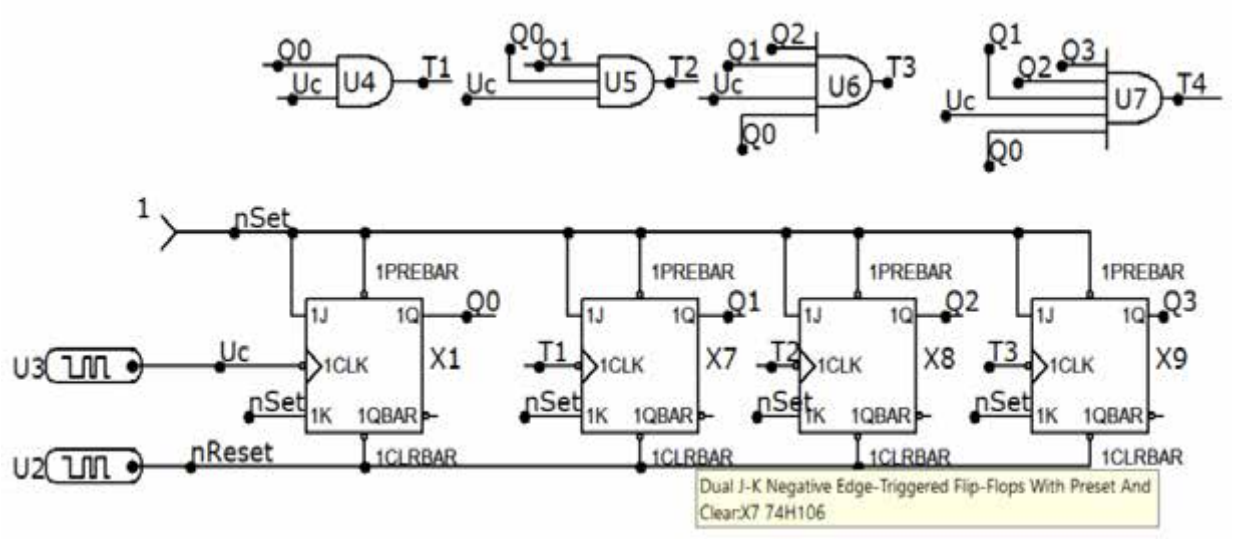


Рисунок 3.6 – Схема для моделювання підсумовувального асинхронного лічильника за модулем 16 з паралельним трактом розповсюдження переносу (приклад 3.1)

Результати моделювання лічильника приведені на рис.3.7.

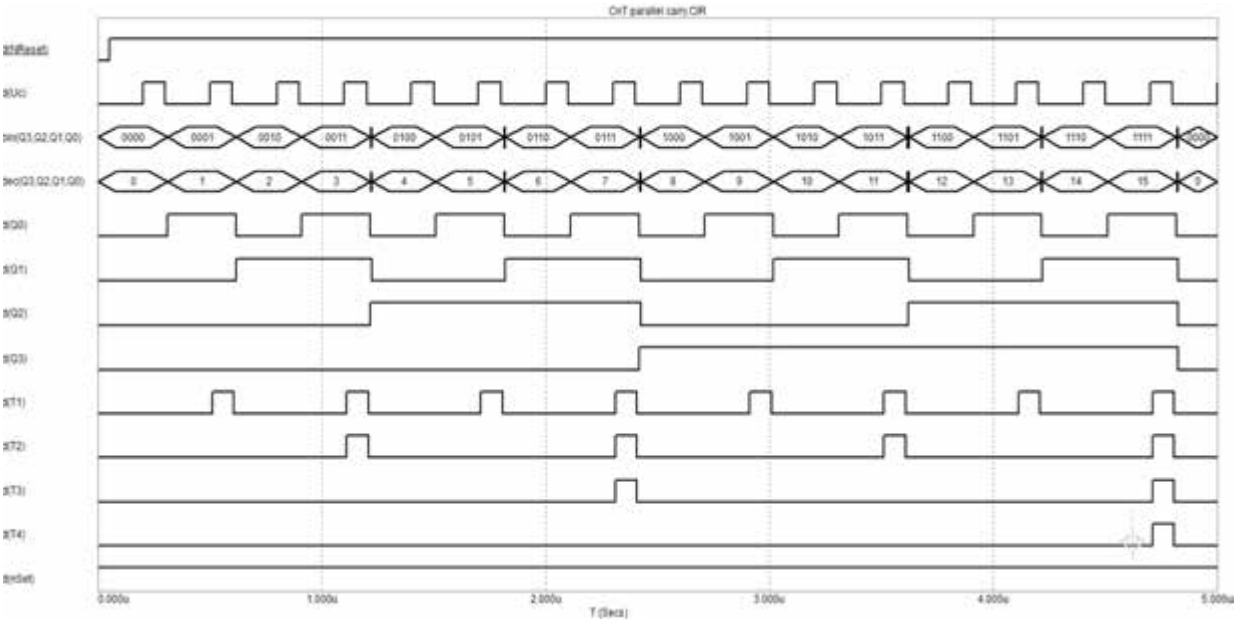


Рисунок 3.7 – Результати моделювання лічильника (приклад 3.1)

З часової діаграми (рис.3.7) можна побачити, що сигнали переносів T_1, T_2, T_3, T_4 формуються одночасно, тобто приведена схема є лічильником з паралельним переносом.

Значення динамічних параметрів визначаються за виразами (3.5). Крім того, значення цих параметрів можна визначити за результатами моделювання.

Розглянемо часову діаграму функціонування лічильника під час формування переносу зі старшого розряду (рис.3.8), на якій показані величини затримок спрацьовування елементів схеми: I (t_I), тригера ($t_{тр}$) та лічильника ($t_п$).

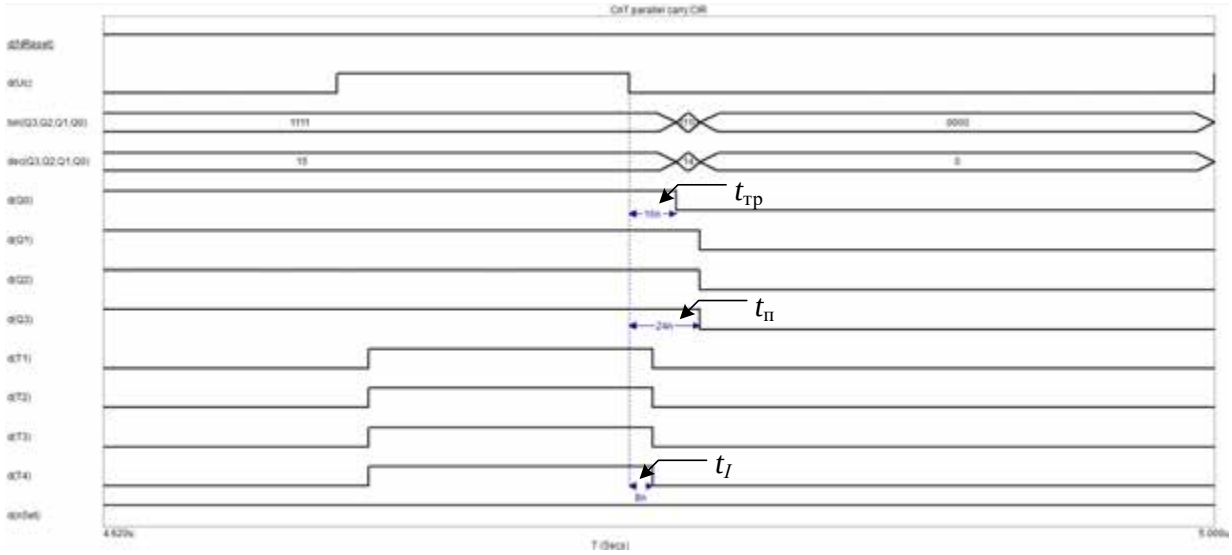


Рисунок 3.8 – Результати моделювання лічильника під час формування переносу зі старшого розряду

З часової діаграми можна побачити, що сигнал переносу зі старшого розряду T_4 формується після спрацьовування елемента U_7 (рис.3.6) через *8нс* (час затримки спрацьовування логічного елемента I в системі моделювання) після появи активного фронту лічильного сигналу U_c, тобто відповідно до виразу $t_{фп} = t_I$. Крім того, з результатів моделювання видно, що тригери Q_3, Q_2, Q_1 також спрацьовують паралельно через *24нс* після появи активного фронту лічильного сигналу U_c, що відповідає виразу $t_п = t_I + t_{тр}$, де $t_I = 8нс$, а $t_{тр} = 16нс$.

На цьому виконання прикладу 3.1 завершено.

Перевагою лічильників з паралельним трактом розповсюдження переносу є висока швидкодія в зв'язку з паралельним спрацьовуванням тригерів лічильника.

Однак, аналізуючи вирази (3.3) та функціональну схему (рис.3.4, рис.3.6), можна відзначити, що при збільшенні кількості розрядів лічильника з кожним новим розрядом збільшується кількість входів елементів *I*, які формують перенос до наступного розряду. Це означає, що при досягненні максимально можливої кількості входів логічного елемента для подальшої реалізації лічильника необхідно виконувати каскадування схем формування переносу, що знижує швидкодію лічильника.

Далі розглянемо приклад синтезу віднімального лічильника з паралельним трактом розповсюдження переносу.

Приклад 3.2. Виконати синтез синхронного віднімального лічильника за модулем *16* з паралельним трактом розповсюдження переносу на основі асинхронних *T*- тригерів.

Розв'язок

Відповідно до (3.1) кількість базових тригерів дорівнює *4*.

Таблиця переходів і функції збудження базових тригерів приведені в табл.3.3, яка заповнюється за аналогією з табл.3.2.

Перша половина таблиці відповідає перебуванню лічильника в стані збереження інформації, тобто стани базових тригерів не змінюються.

В другій половині таблиці лічильний сигнал U_c приймає активне значення і виконує мікрооперацію «декремент», тобто двійковий код нового стану лічильника на одиницю менше коду попереднього стану. Наприклад, для рядка 24 код попереднього стану відповідає значенню вихідних сигналів базових тригерів Q_3^t, Q_2^t Q_1^t Q_0^t і дорівнює 1000_2, що відповідає десятковому еквіваленту 8_{10}. В цьому випадку десятковий код наступного стану лічильника становить 7_{10}, що відповідає значенню вихідних сигналів базових тригерів Q_3^{t+1} Q_2^{t+1} Q_1^{t+1} $Q_0^{t+1} = 0111_2$.

Таблиця 3.3 – Таблиця переходів віднімального лічильника (приклад 3.2)

Номери наборів	U_c	Попередній стан				Наступний стан				ФЗ БТ			
		Q_3^t	Q_2^t	Q_1^t	Q_0^t	Q_3^{t+1}	Q_2^{t+1}	Q_1^{t+1}	Q_0^{t+1}	T_3	T_2	T_1	T_0
1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	***0***	***0***	***0***	***0***
1	*0*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*1*	***0***	***0***	***0***	***0***
2	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*1*	*0*	***0***	***0***	***0***	***0***
3	*0*	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	***0***	***0***	***0***	***0***
4	*0*	*0*	*1*	*0*	*0*	*0*	*1*	*0*	*0*	***0***	***0***	***0***	***0***
5	*0*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	***0***	***0***	***0***	***0***
6	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	***0***	***0***	***0***	***0***
7	*0*	*0*	*1*	*1*	*1*	*0*	*1*	*1*	*1*	***0***	***0***	***0***	***0***
8	*0*	*1*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	***0***	***0***	***0***	***0***
9	*0*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*1*	***0***	***0***	***0***	***0***
10	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	***0***	***0***	***0***	***0***
11	*0*	*1*	*0*	*1*	*1*	*1*	*0*	*1*	*1*	***0***	***0***	***0***	***0***
12	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	***0***	***0***	***0***	***0***
13	*0*	*1*	*1*	*0*	*1*	*1*	*1*	*0*	*1*	***0***	***0***	***0***	***0***
14	*0*	*1*	*1*	*1*	*0*	*1*	*1*	*1*	*0*	***0***	***0***	***0***	***0***
15	*0*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	***0***	***0***	***0***	***0***
16	*1*	*0*	*0*	*0*	*0*	*1*	*1*	*1*	*1*	***1***	***1***	***1***	***1***
17	*1*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*0*	***0***	***0***	***0***	***1***
18	*1*	*0*	*0*	*1*	*0*	*0*	*0*	*0*	*1*	***0***	***0***	***1***	***1***
19	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*0*	***0***	***0***	***0***	***1***
20	*1*	*0*	*1*	*0*	*0*	*0*	*0*	*1*	*1*	***0***	***1***	***1***	***1***
21	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*0*	***0***	***0***	***0***	***1***
22	*1*	*0*	*1*	*1*	*0*	*0*	*1*	*0*	*1*	***0***	***0***	***1***	***1***
23	*1*	*0*	*1*	*1*	*1*	*0*	*1*	*1*	*0*	***0***	***0***	***0***	***1***
24	*1*	*1*	*0*	*0*	*0*	*0*	*1*	*1*	*1*	***1***	***1***	***1***	***1***
25	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*0*	***0***	***0***	***0***	***1***
26	*1*	*1*	*0*	*1*	*0*	*1*	*0*	*0*	*1*	***0***	***0***	***1***	***1***
27	*1*	*1*	*0*	*1*	*1*	*1*	*0*	*1*	*0*	***0***	***0***	***0***	***1***
28	*1*	*1*	*1*	*0*	*0*	*1*	*0*	*1*	*1*	***0***	***1***	***1***	***1***
29	*1*	*1*	*1*	*0*	*1*	*1*	*1*	*0*	*0*	***0***	***0***	***0***	***1***
30	*1*	*1*	*1*	*1*	*0*	*1*	*1*	*0*	*1*	***0***	***0***	***1***	***1***
31	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*0*	***0***	***0***	***0***	***1***

Окремо розглянемо рядок 16 таблиці 3.3. Для цього випадку код попереднього стану лічильника відповідає нульовому значенню, тобто

Q_3^t, Q_2^t Q_1^t $Q_0^t = 0000_2$. Це означає, що за активним значенням фронту лічильного сигналу U_c лічильник переключається в стан 15_{10} (*M-1*), тобто Q_3^{t+1} Q_2^{t+1} Q_1^{t+1} $Q_0^{t+1} = 1111_2$.

Функції збудження базових тригерів визначаються аналогічно тому, як це було виконано в прикладі 3.1.

Далі за допомогою карт Карно виконаємо мінімізацію функцій збудження базових тригерів. При цьому функція збудження T_0 наймолодшого тригера Q_0 також може бути отримана без використання карти Карно в зв'язку з тим, що колонки 2 і 14 є однаковими, тобто можна записати $T_0 = U_c$, а карти Карно для мінімізації функцій збудження T_1, T_2 і T_3 приведені відповідно на рис.3.9, 3.10, 3.11.

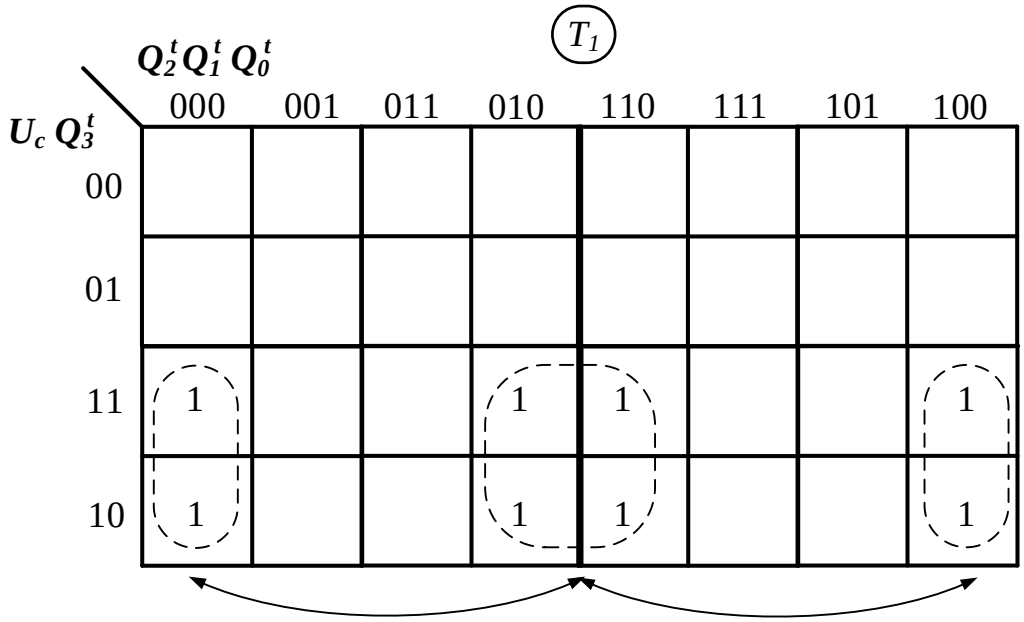


Рисунок 3.9 – Карта Карно для визначення ФЗ тригера Q_1 (приклад 3.2)

В результаті мінімізації можна отримати:

$$T_1 = U_c \cdot \overline{Q}_0; \qquad T_2 = U_c \cdot \overline{Q}_0 \cdot \overline{Q}_1; \qquad T_3 = U_c \cdot \overline{Q}_0 \cdot \overline{Q}_1 \cdot \overline{Q}_2; \qquad (3.6)$$

Аналізуючи функції збудження T_0, T_1, T_2, T_3 з виразів (3.6), можна визначити загальні формули ФЗ для будь-якої кількості розрядів віднімального лічильника:

$$T_i = U_c \cdot \overline{Q_0} \cdot \overline{Q_1} \cdot \ldots \cdot \overline{Q_{i-1}} = U_c \cdot \&_{j=0}^{i-1} \overline{Q_j}. \qquad (3.7)$$

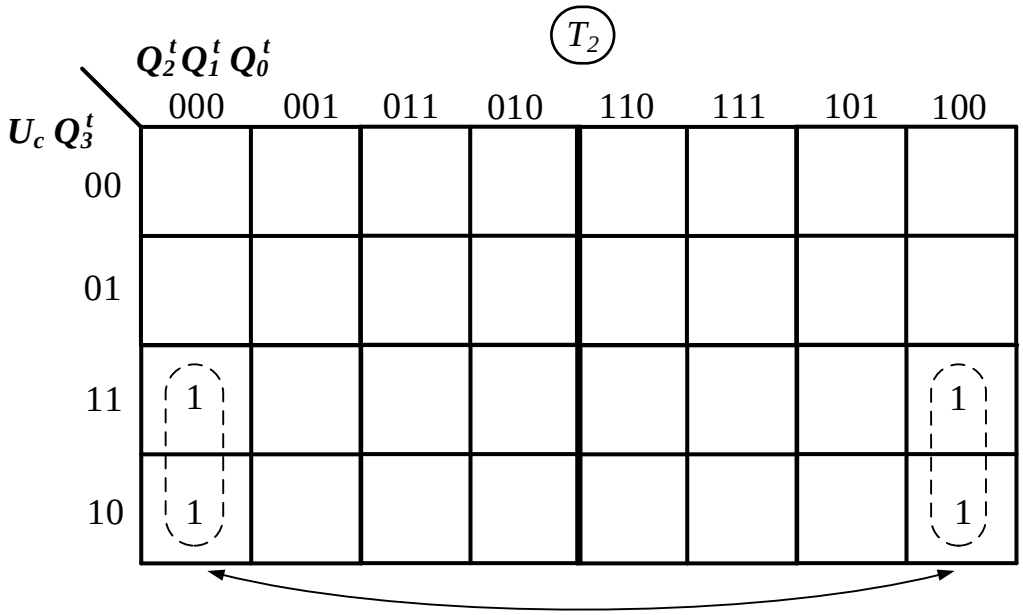


Рисунок 3.10 – Карта Карно для визначення ФЗ тригера Q_2 (приклад 3.2)

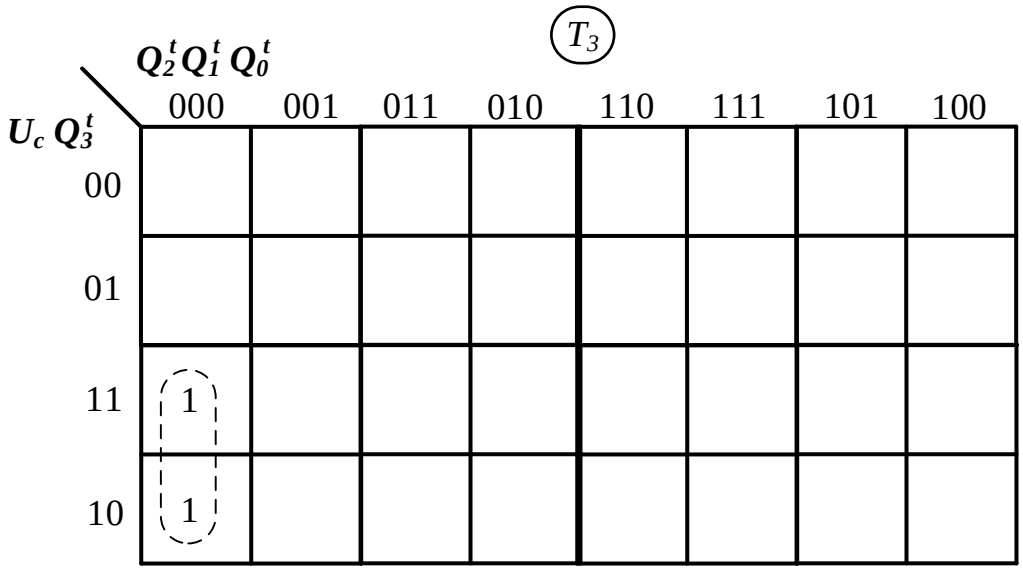


Рисунок 3.11 – Карта Карно для визначення ФЗ тригера Q_3 (приклад 3.2)

На основі отриманих функцій збудження побудуємо функціональну схему заданого лічильника, яка представлена на рис.3.12.

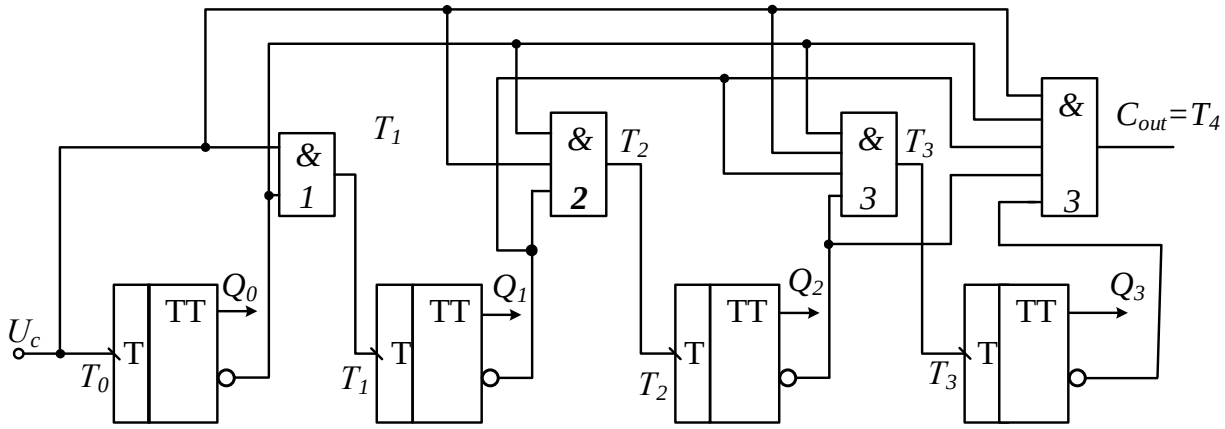


Рисунок 3.12 – Функціональна схема віднімального асинхронного лічильника за модулем 16 з паралельним трактом розповсюдження переносу (приклад 3.2)

Схема для моделювання функціонування лічильника приведена на рис.3.13, а результати моделювання у вигляді часової діаграми цього пристрою – на рис.3.14.

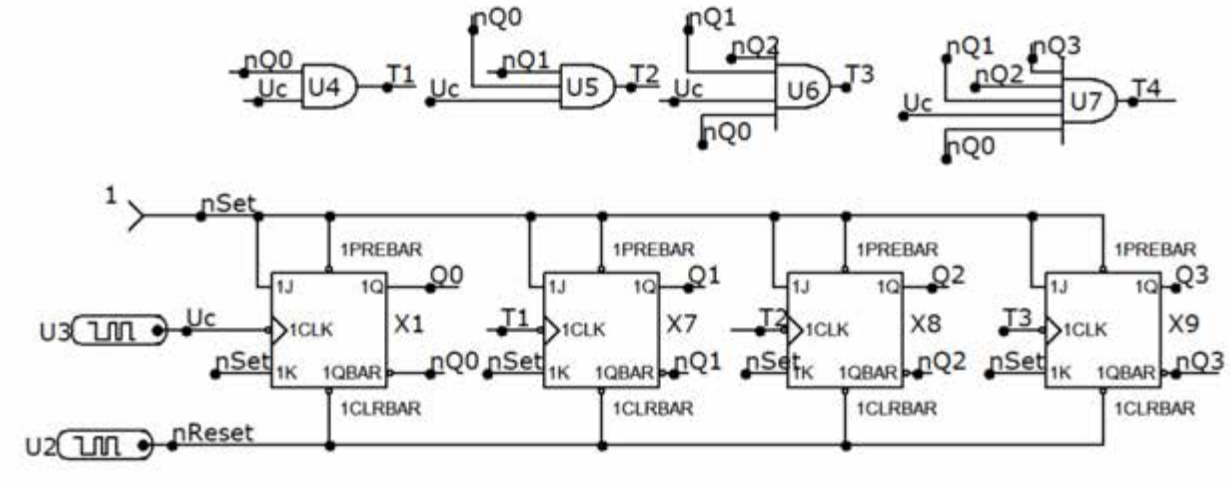


Рисунок 3.13 – Схема для моделювання віднімального асинхронного лічильника за модулем 16 з паралельним трактом розповсюдження переносу (приклад 3.2)

Визначення динамічних параметрів віднімального лічильника виконується аналогічно тому, к розглядалося в прикладі 3.1.

На цьому виконання прикладу 3.2 завершено.

На завершення можна відмітити, що у випадку наявності доступу як до прямих, так і до інверсних виходів базових тригерів віднімальний лічильник може бути побудований на основі підсумовувального лічильника, але в якості

виходів в цьому випадку необхідно використовувати інверсні виходи базових тригерів.

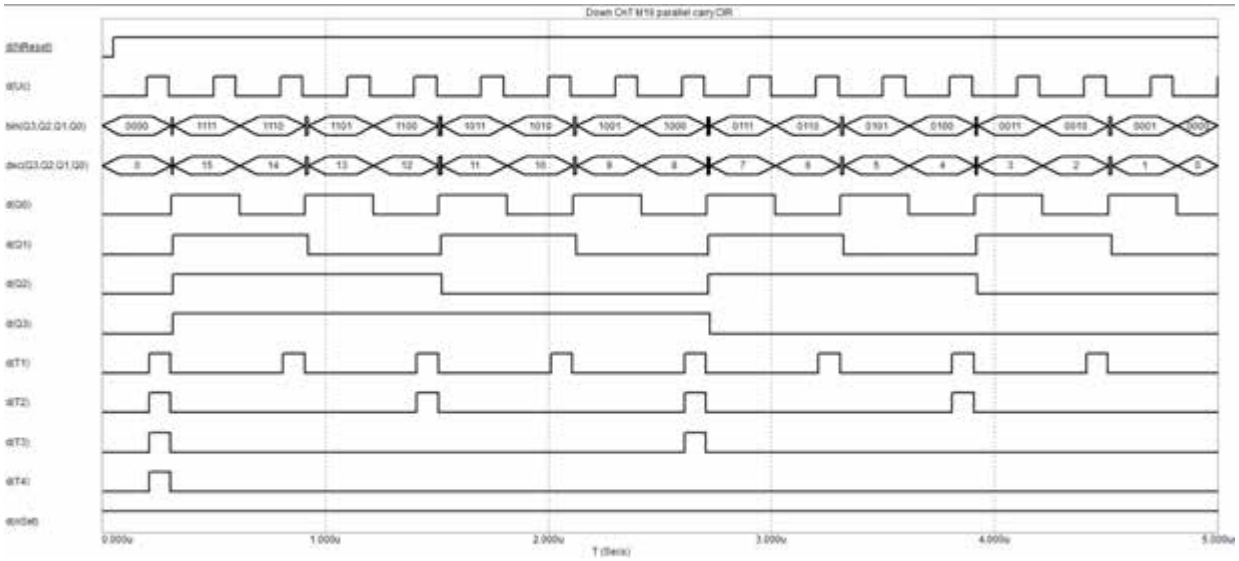


Рисунок 3.14 – Результати моделювання віднімального лічильника (приклад 3.2)

На рис.3.15 приведені результати моделювання лічильника, схема якого приведена на рис.3.6. На часовій діаграмі можна побачити, що прямі виходи базових тригерів (Q_3, Q_2, Q_1, Q_0) формують послідовність станів підсумовувального лічильника, а інверсні виходи цих тригерів ($\overline{Q_3}$, $\overline{Q_2}$, $\overline{Q_1}$, $\overline{Q_0}$) – послідовність станів віднімального лічильника.

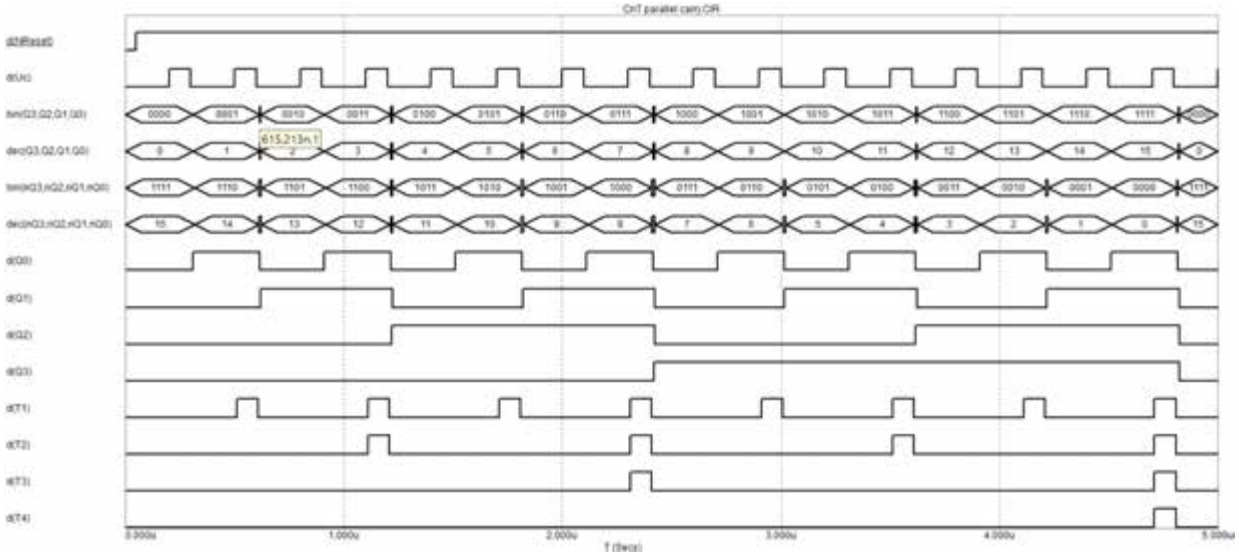


Рисунок 3.15 – Результати зміни станів на прямих та інверсних виходах лічильника

3.1.1.2 Синхронні двійкові лічильники з паралельним трактом розповсюдження переносу

Синтез синхронних лічильників з паралельним трактом розповсюдження переносу виконується в тій же послідовності, що і асинхронні лічильники (п.3.1.1), тому розглянемо синтез цього лічильника відразу на прикладі.

Приклад 3.3. Виконати синтез синхронного підсумовувального лічильника за модулем *16* з паралельним трактом розповсюдження переносу на основі синхронних *TC*- тригерів.

Розв'язок

Відповідно до (3.1) кількість базових тригерів дорівнює *4*.

Таблиця переходів і функції збудження базових тригерів приведені в табл.3.4.

Опис функціонування лічильника (стовпчики 2-10) виконується таким же чином, як в прикладі 3.1, різниця з яким полягає тільки у визначенні функцій збудження базових тригерів (стовпчики 11-15).

Далі виконаємо визначення функцій збудження базових тригерів. В даному завданні базовими тригерами є синхронні *TC*-тригери, тобто спочатку необхідно визначити ФЗ для керуючих входів базових тригерів. Для цього достатньо визначити ФЗ тільки для одного такого входу C_i (стовпчик 11 таблиці 3.4).

Для організації виконання мікрооперації збереження інформації на керуючий вхід *TC*-тригера достатньо подати неактивне значення сигналу, яке, як і раніше будемо позначати логічним нулем. Звичайно, що в цьому випадку тригер буде перебувати в стані збереження інформації незалежно від значення сигналу на вході *T*, що позначається символом «*». Таким чином, в першій половині таблиці (рядки 0 - 15) проставляється $C_i = 0$ за невизначеного значення на вході *T*.

Таблиця 3.4 – Таблиця переходів синхронного лічильника (приклад 3.3)

Номери наборів	U_c	Попередній стан				Наступний стан				ФЗ БТ				
		Q_3^t	Q_2^t	Q_1^t	Q_0^t	Q_3^{t+1}	Q_2^{t+1}	Q_1^{t+1}	Q_0^{t+1}	C_i	T_3	T_2	T_1	T_0
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	***0***	*	*	*	*
1	*0*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*1*	***0***	*	*	*	*
2	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*1*	*0*	***0***	*	*	*	*
3	*0*	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	***0***	*	*	*	*
4	*0*	*0*	*1*	*0*	*0*	*0*	*1*	*0*	*0*	***0***	*	*	*	*
5	*0*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	***0***	*	*	*	*
6	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	***0***	*	*	*	*
7	*0*	*0*	*1*	*1*	*1*	*0*	*1*	*1*	*1*	***0***	*	*	*	*
8	*0*	*1*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	***0***	*	*	*	*
9	*0*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*1*	***0***	*	*	*	*
10	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	***0***	*	*	*	*
11	*0*	*1*	*0*	*1*	*1*	*1*	*0*	*1*	*1*	***0***	*	*	*	*
12	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	***0***	*	*	*	*
13	*0*	*1*	*1*	*0*	*1*	*1*	*1*	*0*	*1*	***0***	*	*	*	*
14	*0*	*1*	*1*	*1*	*0*	*1*	*1*	*1*	*0*	***0***	*	*	*	*
15	*0*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	***0***	*	*	*	*
16	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	***1***	***0***	***0***	***0***	***1***
17	*1*	*0*	*0*	*0*	*1*	*0*	*0*	*1*	*0*	***1***	***0***	***0***	***1***	***1***
18	*1*	*0*	*0*	*1*	*0*	*0*	*0*	*1*	*1*	***1***	***0***	***0***	***0***	***1***
19	*1*	*0*	*0*	*1*	*1*	*0*	*1*	*0*	*0*	***1***	***0***	***1***	***1***	***1***
20	*1*	*0*	*1*	*0*	*0*	*0*	*1*	*0*	*1*	***1***	***0***	***0***	***0***	***1***
21	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*1*	*0*	***1***	***0***	***0***	***1***	***1***
22	*1*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*1*	***1***	***0***	***0***	***0***	***1***
23	*1*	*0*	*1*	*1*	*1*	*1*	*0*	*0*	*0*	***1***	***1***	***1***	***1***	***1***
24	*1*	*1*	*0*	*0*	*0*	*1*	*0*	*0*	*1*	***1***	***0***	***0***	***0***	***1***
25	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*1*	*0*	***1***	***0***	***0***	***1***	***1***
26	*1*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*1*	***1***	***0***	***0***	***0***	***1***
27	*1*	*1*	*0*	*1*	*1*	*1*	*1*	*0*	*0*	***1***	***0***	***1***	***1***	***1***
28	*1*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*1*	***1***	***0***	***0***	***0***	***1***
29	*1*	*1*	*1*	*0*	*1*	*1*	*1*	*1*	*0*	***1***	***0***	***0***	***1***	***1***
30	*1*	*1*	*1*	*1*	*0*	*1*	*1*	*1*	*1*	***1***	***0***	***0***	***0***	***1***
31	*1*	*1*	*1*	*1*	*1*	*0*	*0*	*0*	*0*	***1***	***1***	***1***	***1***	***1***

В другій половині таблиці (рядки 16-31) для забезпечення можливості переключення тригерів використовується активне значення фронту керуючого сигналу, яке позначаємо логічною одиницею ($C_i = 1$), а функції

збудження для відповідних входів T_i визначаються таким же чином, як в прикладах 3.1 і 3.2.

Після цього за допомогою карт Карно виконаємо мінімізацію функцій збудження базових тригерів. При цьому функція збудження для керуючих входів базових тригерів C_i може бути отримана без використання карти Карно в зв'язку з тим, що колонки 2 і 11 є однаковими, тобто можна записати $C_i = U_c$. Враховуючи, що для наймолодшого тригера функція збудження (колонка 15) містить тільки символи «*» і «1», то довизначаючи символи «*» одиницями, можна отримати, що $T_0 = 1$.

Карти Карно для мінімізації функцій збудження інформаційних входів *TC*-тригерів (T_1 - T_3) приведені відповідно на рис.3.16 - 3.18.

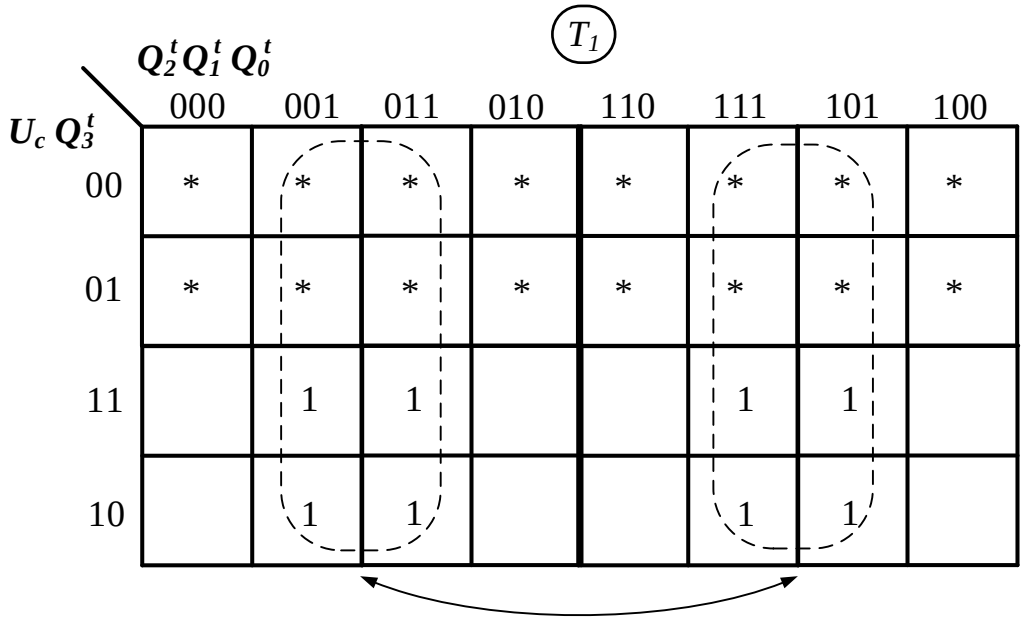


Рисунок 3.16 – Карта Карно для визначення ФЗ тригера Q_1 (приклад 3.3)

В результаті мінімізації можна отримати:

$$C_i = U_c\ (i = 0, \ldots, 3);\ \ T_0 = 1;\ \ T_1 = Q_0;\ \ T_2 = Q_0 \cdot Q_1;\ T_3 = Q_0 \cdot Q_1 \cdot Q_2; \quad (3.8)$$

Відповідно до виразів (3.8) побудуємо функціональну схему заданого лічильника, яка представлена на рис.3.19.

В загальному випадку функції збудження для синхронних лічильників на базі *TC*-тригерів для будь-якого модулю ліку можуть бути представлені таким чином:

$$C_i = U_c; \qquad T_0 = 1; \qquad T_1 = Q_0; \qquad T_i = \&_{j=0}^{i-1} Q_j,\ i = 2, n-1, \qquad (3.9)$$

де n – кількість розрядів лічильника.

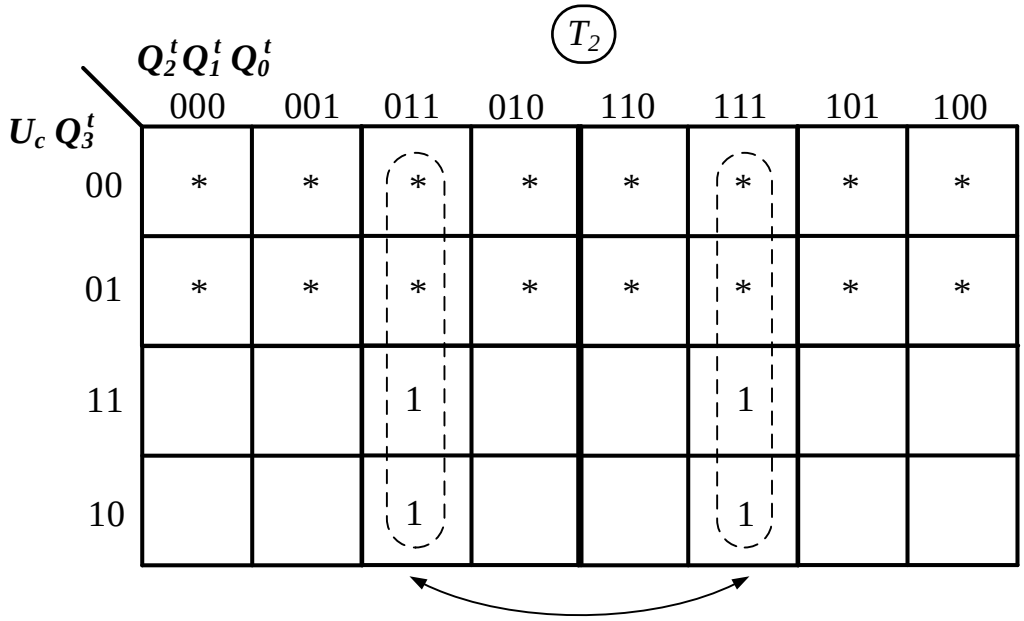


Рисунок 3.17 – Карта Карно для визначення ФЗ тригера Q_2 (приклад 3.3)

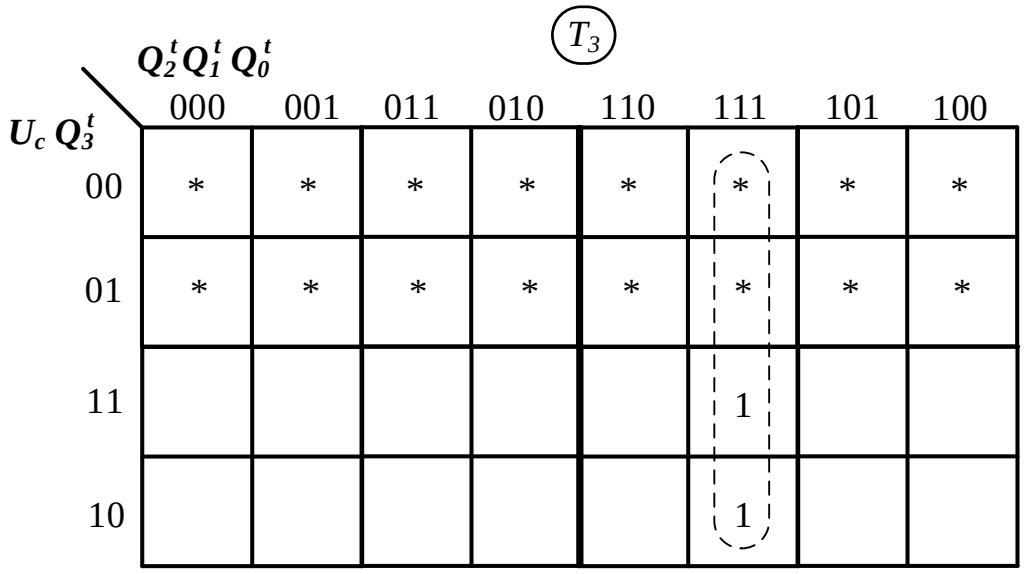


Рисунок 3.18 – Карта Карно для визначення ФЗ тригера Q_3 (приклад 3.3)

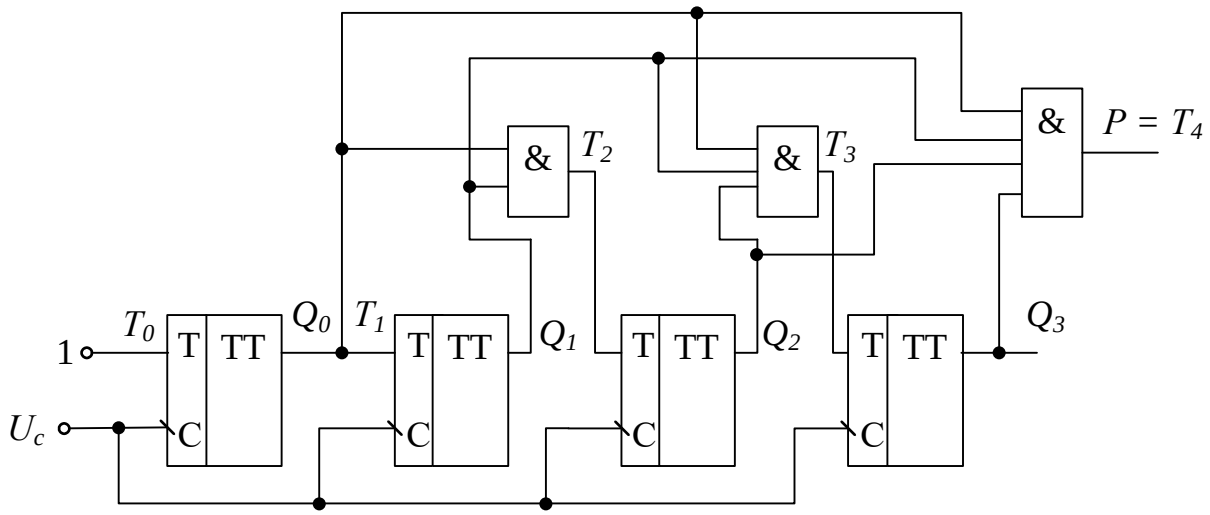


Рисунок 3.19 – Функціональна схема підсумовувального синхронного лічильника за модулем 16 з паралельним трактом розповсюдження переносу (приклад 3.3)

Для перевірки коректності отриманої функціональної схеми виконаємо моделювання функціонування лічильника. Схема для моделювання лічильника приведена на рис.3.20.

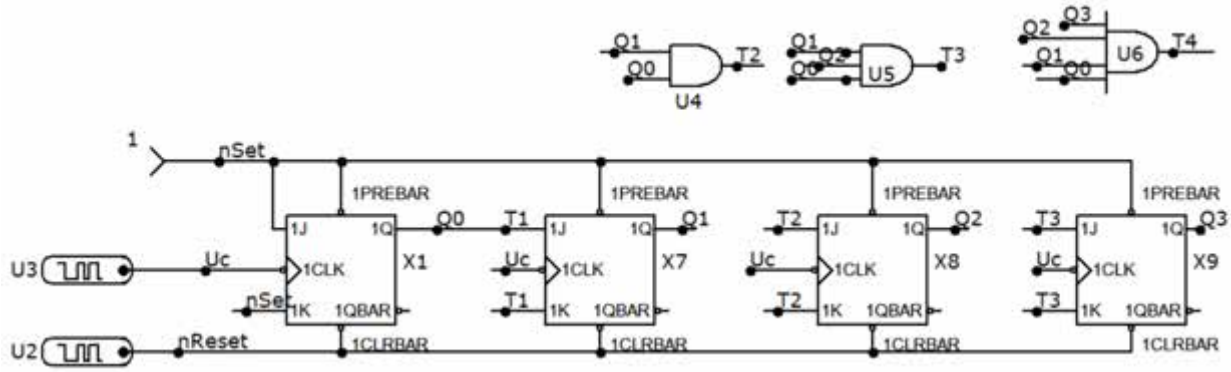


Рисунок 3.20 – Схема для моделювання асинхронного лічильника за модулем 16 з паралельним трактом розповсюдження переносу (приклад 3.3)

Результати моделювання заданого лічильника приведені на рис.3.21.

Для визначення динамічних параметрів заданого лічильника скористуємося результатами моделювання під час формування переносу з найстаршого розряду T_4, які приведені на рис.3.22.

Аналізуючи структуру функціональної схеми (рис.3.19, 3.20), можна відзначити, що формування переносу T_4 з виходу елемента *U6* (рис.3.20) відбувається після переключення базових тригерів, тобто $t_{фп} = t_{тр} + t_I = 24нс$,

де $t_{тр} = 16нс$, а $t_I = 8нс$, що підтверджується результатами моделювання. В зв'язку з тим, що лічильник є синхронним, то всі тригери цього пристрою спрацьовують одночасно, тобто $t_п = t_{тр} = 16нс$ (рис.3.22).

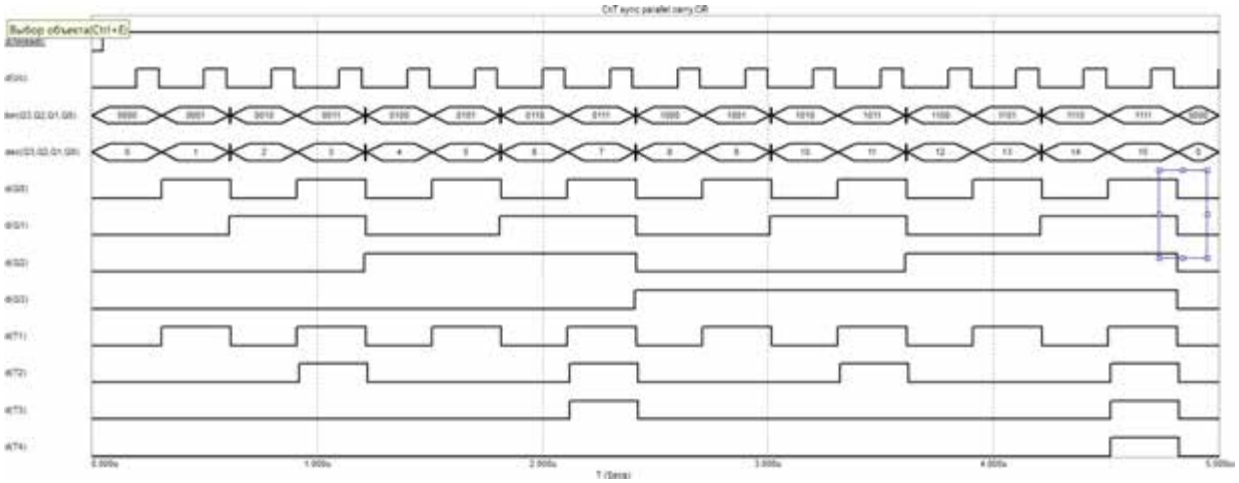


Рисунок 3.21 – Результати моделювання синхронного лічильника (приклад 3.3)

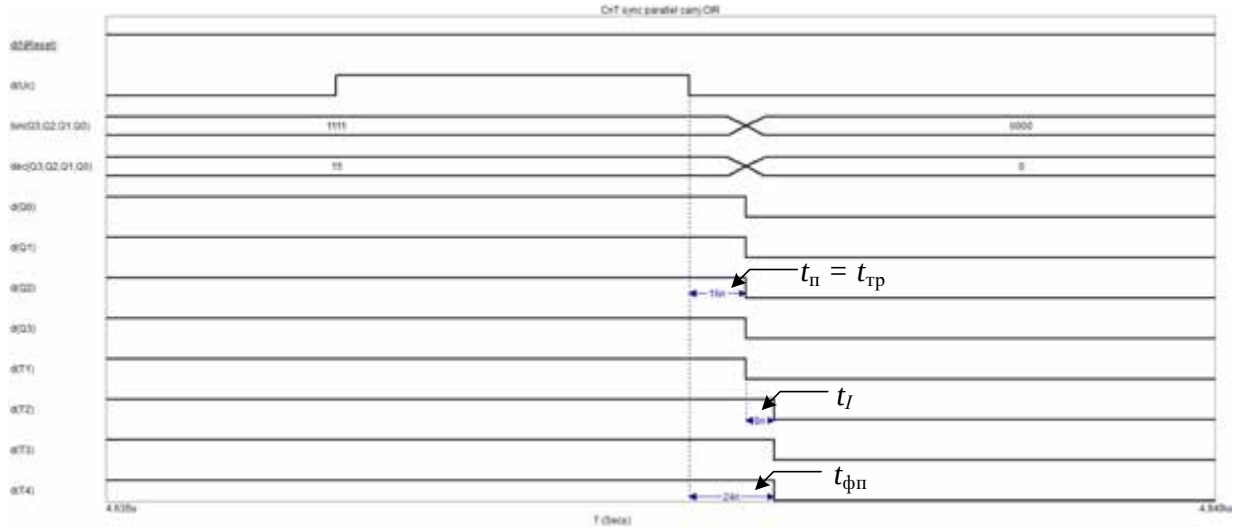


Рисунок 3.22 – Результати моделювання для визначення динамічних параметрів синхронного лічильника (приклад 3.3)

На цьому виконання прикладу 3.3 завершено.

На завершення слід відзначити, що синтез синхронного лічильника можна проводити за результатами синтезу асинхронного лічильника без розробки таблиці переходів і мінімізації функцій збудження базових тригерів. Враховуючи, що, наприклад, в двоступеневих тригерах вхід синхронізації кон'юнктивно об'єднується з відповідними інформаційними

входами (рис.3.23) [1], то тоді у виразах (3.3) змінну U_c можна використовувати для підключення на керуючий вхід *TC*-тригера, а кон'юнкцію решти змінних підключити до входу *T*, в результаті чого можна отримати вирази (3.9).

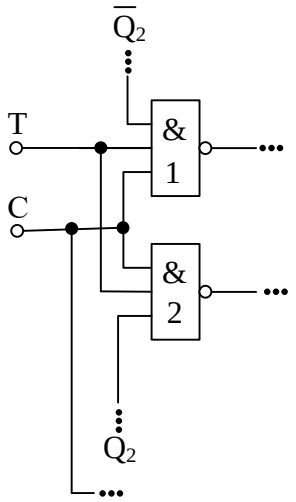


Рисунок 3.22 – Організація вхідних кіл двоступеневого *TC*-тригера

Аналізуючи часові діаграми функціонування лічильників (рис.3.7, 3.14, 3.21), можна зробити висновок, що частота появи сигналу переносу зі старшого розряду в *16* разів менше (нагадаємо, що *16* – модуль ліку *M* лічильника в прикладах, що розглядалися), ніж частота вхідного сигналу. Таким чином, будь-який лічильник виконує функцію **ділення частоти** вхідного сигналу в *M* разів.

В зв'язку з цим розрізняють два режими роботи лічильників:

- режим підрахунку вхідних сигналів;

- режим ділення частоти.

В режимі підрахунку вхідних сигналів користувач має доступ до стану лічильника в будь-який момент часу, а послідовність зміни кодів станів лічильника відповідає природній послідовності (наприклад, показання годинника, лічильника води тощо).

В режимі ділення частоти користувач, як правило, не має доступу до стану лічильника, але при цьому користувача цікавить момент формування

переносу зі старшого розряду, наприклад, поява сигналу T_4 в лічильниках, що розглядалися в прикладах 3.1 - 3.3. В цьому режимі може використовуватися будь-яка послідовність зміни станів лічильника (наприклад, пральна машина, при роботі якої користувача, як правило, цікавить момент формування сигналу про завершення процесу прання).

Контрольні завдання та запитання

1. Який цифровий пристрій називається лічильником?
2. Які мікрооперації виконує лічильник?
3. З якою метою використовуються лічильники?
4. Поясніть принцип функціонування лічильників в комп'ютерних системах.
5. Для чого використовуються лічильники при виконанні арифметичних операцій множення та ділення?
6. Як використовуються лічильники, що зберігають адресу команди, що буде виконуватися?
7. Який статичний параметр характеризує функціонування лічильників в комп'ютерних системах?
8. Для чого використовується термін «модуль ліку»?
9. Як буде реагувати лічильник, якщо кількість вхідних сигналів перевищила максимальну кількість, яку може підрахувати лічильник?
10. Вкажіть, як визначити кількість станів, в яких може перебувати лічильник?
11. Який модуль ліку у лічильника хвилин в годиннику?
12. Який модуль ліку у лічильника годин в годиннику?
13. Який модуль ліку у лічильника електроенергії, якщо він має 4 десяткові позиції?
14. Скільки станів у лічильника, у якого $M = 24$?
15. В чому полягає циклічність роботи лічильника?

16. Чи достатньо розглядати тільки один цикл роботи лічильника? Обґрунтуйте відповідь.

17. Яку мікрооперацію виконує лічильник, якщо на вході немає активного значення вхідного сигналу?

18. Які вузли обов’язково входять до складу лічильників?

19. Чи можна використовувати у складі лічильника прозорі тригери? Обґрунтуйте відповідь.

20. Які способи структурної організації тригерів використовуються для побудови непрозорих тригерів?

21. Стани яких вузлів лічильника формують його стан?

22. Поясніть термін «перенос» в лічильниках.

23. Поясніть термін «позика» в лічильниках.

24. До якого тригера підключається вхідний сигнал в асинхронних лічильниках?

25. Приведіть класифікаційні ознаки, за якими здійснюється класифікація лічильників.

26. Як розрізняються лічильники за способом синхронізації?

27. В чому полягає відмінність між синхронними і асинхронними лічильниками?

28. Як розрізняються лічильники за напрямом ліку?

29. До якого типу (за напрямом ліку) відноситься лічильник, що виконує мікрооперацію «інкремент»?

30. До якого типу (за напрямом ліку) відноситься лічильник, що виконує мікрооперацію «декремент»?

31. Яким чином функціонують підсумовувальні лічильники?

32. Яким чином функціонують віднімальні лічильники?

33. Поясніть термін «реверсивний лічильник».

34. Яким чином функціонують реверсивні лічильники?

35. Як розрізняються лічильники за способом організації міжрозрядного переносу?

36. Як розрізняються лічильники за модулем ліку?

37. До якого типу за модулем ліку належить лічильник, у якого $M = 100$?

38. До якого типу за модулем ліку належить лічильник, у якого $M = 128$?

39. До якого типу за модулем ліку належить лічильник, у якого $M = 127$?

40. До якого типу за модулем ліку належить лічильник, у якого $M = 60$?

41. До якого типу за напрямом ліку належить лічильник, якщо код попереднього стану дорівнює 10111_2, а код наступного – 11000_2?

42. До якого типу за напрямом ліку належить лічильник, якщо код попереднього стану дорівнює 10111_2, а код наступного – 10110_2?

43. До якого типу за напрямом ліку належить лічильник, якщо код попереднього стану дорівнює 11111_2, а код наступного – 00000_2?

44. До якого типу за напрямом ліку належить лічильник, якщо код попереднього стану дорівнює 00000_2, а код наступного – 11111_2?

45. Заданий підсумовувальний лічильник з $M = 16$. Поточний стан лічильника 0111_2. Визначити наступний стан лічильника, якщо він виконує мікрооперацію «інкремент», а на вхід надходить активне значення лічильного сигналу.

46. Заданий підсумовувальний лічильник з $M = 16$. Поточний стан лічильника 1111_2. Визначити наступний стан лічильника, якщо він виконує мікрооперацію «інкремент», а на вхід надходить активне значення лічильного сигналу.

47. Заданий підсумовувальний лічильник з $M = 10$. Поточний стан лічильника 1001_2. Визначити наступний стан лічильника, якщо він виконує мікрооперацію «інкремент», а на вхід надходить активне значення лічильного сигналу.

48. Заданий підсумовувальний лічильник з $M = 10$. Поточний стан лічильника 1000_2. Визначити наступний стан лічильника, якщо він виконує мікрооперацію «інкремент», а на вхід надходить активне значення лічильного сигналу.

49. Заданий віднімальний лічильник з $M = 16$. Поточний стан лічильника 0111_2. Визначити наступний стан лічильника, якщо він виконує мікрооперацію «декремент», а на вхід надходить активне значення лічильного сигналу.

50. Заданий віднімальний лічильник з $M = 16$. Поточний стан лічильника 1111_2. Визначити наступний стан лічильника, якщо він виконує мікрооперацію «декремент», а на вхід надходить активне значення лічильного сигналу.

51. Заданий віднімальний лічильник з $M = 16$. Поточний стан лічильника 0000_2. Визначити наступний стан лічильника, якщо він виконує мікрооперацію «декремент», а на вхід надходить активне значення лічильного сигналу.

52. Заданий віднімальний лічильник з $M = 10$. Поточний стан лічильника 0000_2. Визначити наступний стан лічильника, якщо він виконує мікрооперацію «декремент», а на вхід надходить активне значення лічильного сигналу.

53. Заданий віднімальний лічильник з $M = 10$. Поточний стан лічильника 1001_2. Визначити наступний стан лічильника, якщо він виконує мікрооперацію «декремент», а на вхід надходить активне значення лічильного сигналу.

54. Заданий підсумовувальний лічильник з $M = 10$. Поточний стан лічильника 1000_2. Визначити наступний стан лічильника, якщо він виконує мікрооперацію «інкремент», а на вхід надходить неактивне значення лічильного сигналу.

55. Заданий віднімальний лічильник з $M = 10$. Поточний стан лічильника 0000_2. Визначити наступний стан лічильника, якщо він виконує мікрооперацію «декремент», а на вхід надходить неактивне значення лічильного сигналу.

56. Якими динамічними параметрами характеризується функціонування лічильників?

57. Дайте визначення терміну «час переключення лічильника».

58. Дайте визначення терміну «час формування переносу зі старшого розряду».

59. З якою метою використовується сигнал переносу зі старшого розряду?

60. Дайте визначення терміну «максимальна частота надходження вхідних сигналів».

61. Як визначити максимальну частоту надходження вхідного сигналу лічильника?

62. Приведіть методику синтезу асинхронних двійкових лічильників з паралельним трактом розповсюдження переносу.

63. Як визначити кількість базових тригерів у складі лічильника?

64. Опишіть загальний вигляд таблиці переходів лічильника.

65. Для чого використовується таблиця переходів лічильника?

66. Яким чином заповнюється таблиця переходів лічильника?

67. Яким чином визначаються функції збудження базових тригерів?

68. Дайте пояснення терміну «функція збудження тригера».

69. Яким чином виконується мінімізація функцій збудження базових тригерів?

70. Яким чином виконується перевірка коректності проведення процедури синтезу лічильника?

71. Скільки тригерів необхідно використовувати в лічильнику з $M = 64$?

72. Скільки тригерів необхідно використовувати в лічильнику з $M = 60$?

73. Скільки тригерів необхідно використовувати в лічильнику з $M = 24$?

74. Скільки тригерів необхідно використовувати в лічильнику з $M = 99$?

75. Як працює лічильник, якщо на вхід підключено неактивне значення лічильного сигналу?

76. Як працює лічильник, якщо на вхід підключено активне значення лічильного сигналу?

77. Як визначати функції збудження *D*-тригера?

78. Як визначати функції збудження *RCS*-тригера?

79. Як визначати функції збудження *JK*-тригера?

80. Як визначати функції збудження *TC*-тригера?

81. Виконати синтез асинхронного підсумовувального лічильника за модулем *8* з паралельним трактом розповсюдження переносу на базі асинхронних *T*-тригерів. Побудувати часові діаграми лічильника з використанням затримок логічних елементів.

82. Виконати синтез асинхронного підсумовувального лічильника за модулем *32* з паралельним трактом розповсюдження переносу на базі асинхронних *T*-тригерів. Побудувати часові діаграми функціонування лічильника з використанням затримок логічних елементів.

83. Виконати синтез асинхронного віднімального лічильника за модулем *8* з паралельним трактом розповсюдження переносу на базі асинхронних *T*-тригерів. Побудувати часові діаграми функціонування лічильника з використанням затримок логічних елементів.

84. Виконати синтез асинхронного віднімального лічильника за модулем *32* з паралельним трактом розповсюдження переносу на базі асинхронних *T*-тригерів. Побудувати часові діаграми функціонування лічильника з використанням затримок логічних елементів.

85. Виконати синтез синхронного підсумовувального лічильника за модулем *32* з паралельним трактом розповсюдження переносу на базі синхронних *TC*-тригерів. Побудувати часові діаграми функціонування лічильника з використанням затримок логічних елементів.

86. Виконати синтез синхронного віднімального лічильника за модулем *32* з паралельним трактом розповсюдження переносу на базі синхронних *TC*-тригерів. Побудувати часові діаграми функціонування лічильника з використанням затримок логічних елементів.

87. Виконати синтез синхронного підсумовувального лічильника за модулем *8* з паралельним трактом розповсюдження переносу на базі

синхронних *RCS*-тригерів. Побудувати часові діаграми функціонування лічильника з використанням затримок логічних елементів.

88. Виконати синтез синхронного віднімального лічильника за модулем *8* з паралельним трактом розповсюдження переносу на базі синхронних *RCS*- тригерів. Побудувати часові діаграми функціонування лічильника з використанням затримок логічних елементів.

89. Виконати синтез синхронного підсумовувального лічильника за модулем *8* з паралельним трактом розповсюдження переносу на базі *JK*- тригерів. Побудувати часові діаграми функціонування лічильника з використанням затримок логічних елементів.

90. Виконати синтез синхронного віднімального лічильника за модулем *8* з паралельним трактом розповсюдження переносу на базі *JK*- тригерів. Побудувати часові діаграми функціонування лічильника з використанням затримок логічних елементів.

91. Виконати синтез синхронного підсумовувального лічильника за модулем *32* з паралельним трактом розповсюдження переносу на базі *D*- тригерів. Побудувати часові діаграми функціонування лічильника з використанням затримок логічних елементів.

92. Виконати синтез синхронного віднімального лічильника за модулем *32* з паралельним трактом розповсюдження переносу на базі *D*- тригерів. Побудувати часові діаграми функціонування лічильника з використанням затримок логічних елементів.

93. Визначити динамічні параметри асинхронного лічильника з паралельним трактом розповсюдження переносу.

94. Визначити динамічні параметри синхронного лічильника з паралельним трактом розповсюдження переносу.

95. Як отримати вирази (3.2)?

96. Доведіть вирази (3.3).

97. Для чого використовуються вирази (3.3)?

98. Прокоментуйте вираз (3.4).

99. Доведіть вирази (3.5).

100. Як визначити час спрацьовування асинхронного лічильника з паралельним трактом розповсюдження переносу?

101. Як визначити час формування переносу зі старшого розряду асинхронного лічильника з паралельним трактом розповсюдження переносу?

102. Як визначити час спрацьовування синхронного лічильника з паралельним трактом розповсюдження переносу?

103. Як визначити час формування переносу зі старшого розряду синхронного лічильника з паралельним трактом розповсюдження переносу?

104. Як розрахувати мінімальний період надходження лічильного сигналу?

105. Прокоментуйте часову діаграму на рис.3.5.

106. Для чого використовується сигнал ***nReset*** в схемі на рис.3.6.?

107. Розрахуйте $t_{\text{п}}$ асинхронного лічильника з паралельним трактом розповсюдження переносу, якщо $t_I = 10нс$; $t_{тр} = 20нс$.

108. Розрахуйте $t_{\text{п}}$ синхронного лічильника з паралельним трактом розповсюдження переносу, якщо $t_I = 10нс$; $t_{тр} = 20нс$.

109. Розрахуйте $t_{\text{фп}}$ асинхронного лічильника з паралельним трактом розповсюдження переносу, якщо $t_I = 10нс$; $t_{тр} = 20нс$.

110. Розрахуйте $t_{\text{фп}}$ синхронного лічильника з паралельним трактом розповсюдження переносу, якщо $t_I = 10нс$; $t_{тр} = 20нс$.

111. Які недоліки асинхронних лічильників з паралельним трактом розповсюдження переносу?

112. Яка перевага асинхронних лічильників з паралельним трактом розповсюдження переносу?

113. Доведіть вирази (3.9).

114. Для чого використовуються вирази (3.9)?

115. Як побудувати синхронний лічильник за результатами синтезу асинхронного лічильника?

116. Прокоментуйте рис.3.22.

117. Поясніть, чому лічильник називають дільником частоти?

118. Заданий лічильник за модулем *M = 10*. На скільки ділить частоту вхідного сигналу цей лічильник?

119. Частота лічильного сигналу *10МГц*. Вкажіть частоту сигналу переносу зі старшого розряду, якщо лічильник має модуль ліку *M = 100*.

120. Вкажіть, чи є асинхронний *T*-тригер дільником частоти? Якщо це так, то на скільки цей тригер ділить частоту вхідного сигналу? Обґрунтуйте відповідь.

121. Чому асинхронний *T*-тригер називають лічильним тригером?

122. На скільки ділить частоту вхідного сигналу лічильник хвилин в годиннику?

123. На скільки ділить частоту вхідного сигналу лічильник годин в годиннику?

124. В яких режимах функціонування можуть використовуватися лічильники?

125. Поясніть функціонування лічильника в режимі підрахунку вхідних сигналів.

126. Поясніть функціонування лічильника в режимі ділення частоти.

127. В чому полягає різниця між режимами підрахунку вхідних сигналів і ділення частоти?

3.1.2 Двійкові лічильники з послідовним трактом розповсюдження переносу

3.1.2.1 Асинхронні двійкові лічильники з послідовним трактом розповсюдження переносу

Виконаємо аналіз функцій збудження асинхронних *T*-тригерів, які використовувалися для побудови лічильників з паралельним трактом розповсюдження переносу відповідно до виразів (3.2) і (3.3). Аналізуючи ці вирази, можна помітити, що кожна нова функція збудження для більш

старших розрядів містить в своєму складі функцію збудження для попереднього розряду, тобто кожна функція збудження поточного розряду залежить від попередньої. Наприклад, функція T_2 містить вираз $U_c \cdot Q_0$, який є функцією збудження для більш молодшого тригера T_1. В результаті можна записати:

$$
\begin{aligned}
&T_1 = U_c \cdot Q_0;\\
&T_2 = U_c \cdot Q_0 \cdot Q_1 = T_1 \cdot Q_1;\\
&T_3 = U_c \cdot Q_0 \cdot Q_1 \cdot Q_2 = T_2 \cdot Q_2;\\
&T_4 = U_c \cdot Q_0 \cdot Q_1 \cdot Q_2 \cdot Q_3 = T_3 \cdot Q_3;\\
&\dots\\
&T_i = U_c \cdot Q_0 \cdot Q_1 \cdot \ldots \cdot Q_{i-1} = T_{i-1} \cdot Q_{i-1}.
\end{aligned}
\qquad (3.10)
$$

Вирази (3.10) дозволяють реалізувати лічильники з послідовним трактом розповсюдження переносу. Функціональна схема такого лічильника приведена на рис.3.23.

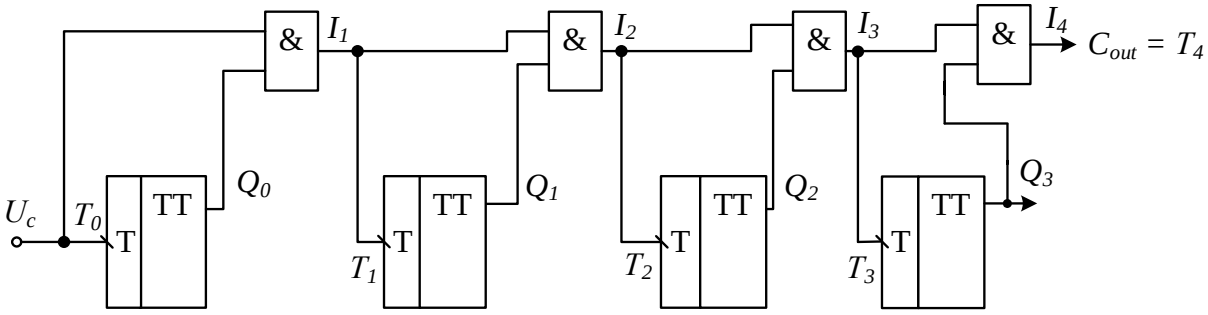


Рисунок 3.23 – Функціональна схема підсумовувального асинхронного лічильника за модулем 16 з послідовним трактом розповсюдження переносу

Схема для моделювання лічильника приведена на рис.3.24.

Результати моделювання синтезованого лічильника приведені на рис.3.25. З часової діаграми можна побачити, що послідовність зміни станів лічильника відповідає функціонуванню підсумовувального лічильника за модулем *16*.

Визначимо динамічні параметри лічильника.

Відповідно до структурної організації лічильника елементи $I_1 - I_4$ утворюють тракт розповсюдження переносу між розрядами, причому ці

логічні елементи включені до схеми за допомогою каскадного з'єднання, отже спрацьовують один після іншого, тобто послідовно.

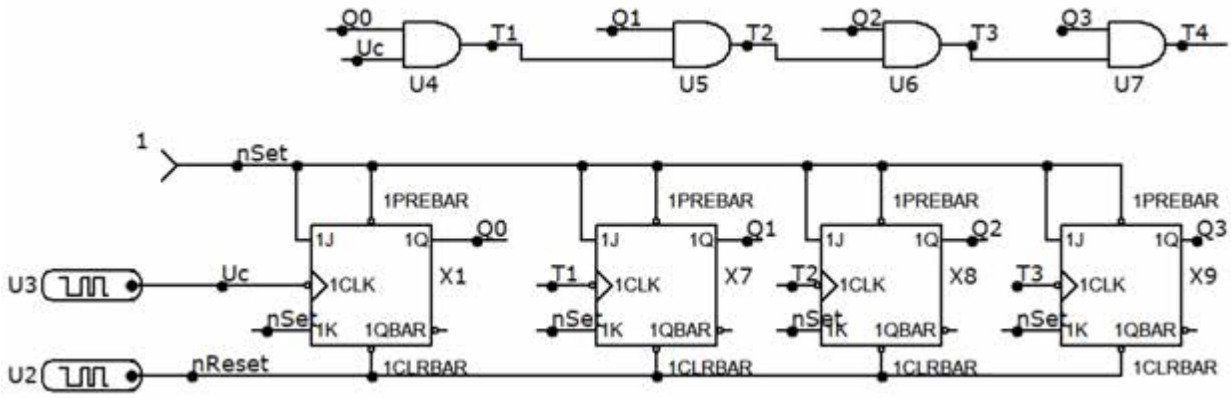


Рисунок 3.24 – Схема для моделювання асинхронного лічильника за модулем 16 з послідовним трактом розповсюдження переносу

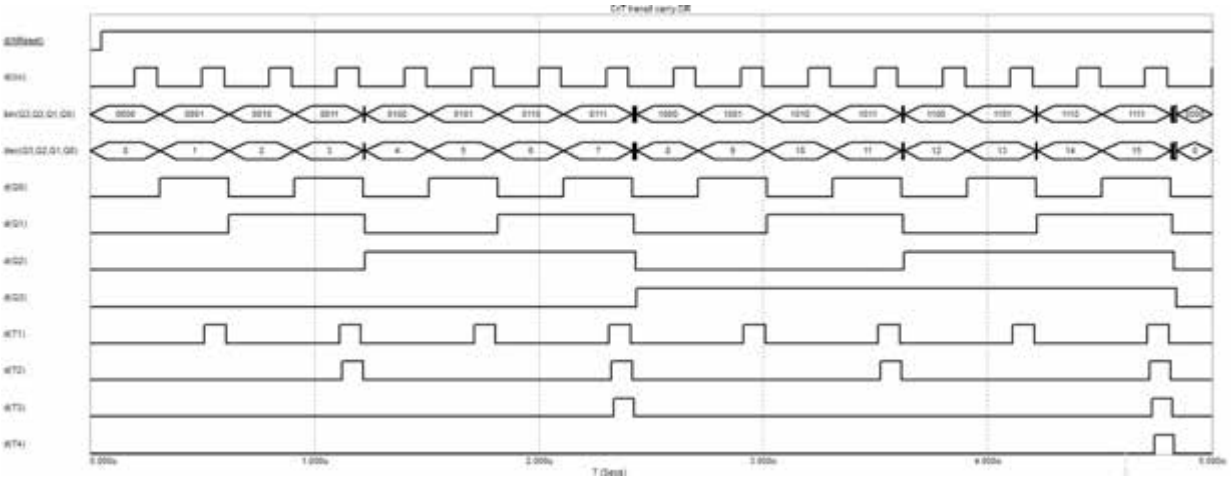


Рисунок 3.25 – Результати моделювання асинхронного лічильника за модулем 16 з послідовним трактом розповсюдження переносу

Таким чином, зі структурної організації лічильника випливає, що найстарший тригер пристрою спрацьовує тільки після спрацювання всіх попередніх елементів *I*, тобто час переключення лічильника можна визначити за виразом:

$$t_{\text{п}} = (n_{\text{тр}}\text{-}1)\cdot t_I + t_{\text{тр}}, \qquad (3.11)$$

де $n_{\text{тр}}$ – кількість тригерів в лічильнику.

Аналогічно вихідний сигнал переносу зі старшого розряду $C_{out} = T_4$ формується після послідовного спрацьовування всіх елементів *I*, тобто час формування переносу може бути визначений таким чином:

$$t_{фп} = n_{тр} \cdot t_I. \quad (3.12)$$

Результати моделювання для моменту формування вихідного переносу приведені на рис.3.26.

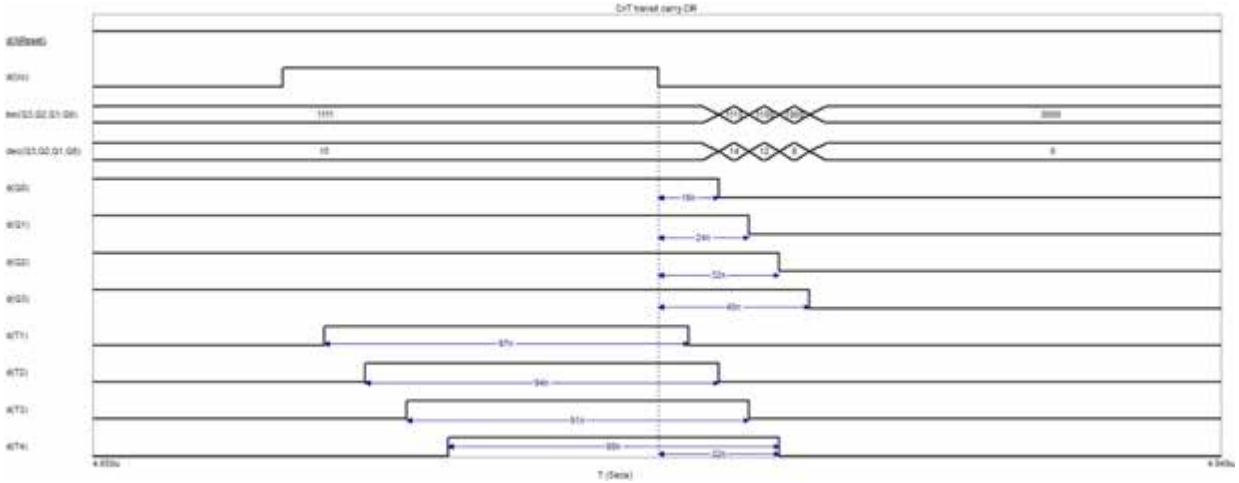


Рисунок 3.26 – Результати моделювання для визначення динамічних параметрів асинхронного лічильника за модулем 16 з послідовним трактом розповсюдження переносу

На часовій діаграмі можна побачити, що базові тригери спрацьовують по черзі та час переключення лічильника відповідає виразу (3.11)

$$t_п = (n_{тр}\text{-}1) \cdot t_I + t_{тр} = 3 \cdot 8 + 16 = 40нс,$$

де $n_{тр} = 4$; $t_I = 8нс$; $t_{тр} = 16нс$.

Час формування переносу відповідає виразу (3.12) $t_{фп} = n_{тр} \cdot t_I = 32нс$.

Розглянемо ще одну особливість роботи такого лічильника. Відповідно до моделі елемента *I* за замовченням час затримки елемента *I* при переключенні з високого рівня до низького складає *8нс*, а навпаки – *11нс*. В цьому випадку тривалість сигналу переносу від молодших розрядів до старших зменшується, що можна побачити на рис.3.26. Це означає, що за достатньо короткої тривалості сигналу U_c та великої кількості тригерів тривалість сигналу переносу в старших розрядах може скоротитися до критичної величини, коли тригери не будуть реагувати на ці сигнали. Все це необхідно мати на увазі під час проєктування лічильників з послідовним трактом розповсюдження переносу.

3.1.2.2 Синхронні двійкові лічильники з послідовним трактом розповсюдження переносу

Виконуючи аналіз виразів (3.8), (3.9), які є функціями збудження *TC*- тригера синхронного лічильника, аналогічно п.3.1.2.1, можна також використовувати каскадне з'єднання логічних елементів, що формують сигнали переносу між розрядами:

$$
\begin{aligned}
&C_i = U_c \ (i = 0, \ldots, n_{\text{тр}}\text{-}1);\\
&T_0 = 1;\ \ T_1 = Q_0;\\
&T_2 = Q_0 \cdot Q_1 = T_1 \cdot Q_1;\\
&T_3 = Q_0 \cdot Q_1 \cdot Q_2 = T_2 \cdot Q_2; \qquad (3.13)\\
&T_4 = Q_0 \cdot Q_1 \cdot Q_2 \cdot Q_3 = T_3 \cdot Q_3;\\
&\ldots\\
&T_i = Q_0 \cdot Q_1 \cdot \ldots \cdot Q_{i-1} = T_{i-1} \cdot Q_{i-1}.
\end{aligned}
$$

Вирази (3.13) дозволяють реалізувати синхронні лічильники з послідовним трактом розповсюдження переносу. Функціональна схема такого лічильника приведена на рис.3.27.

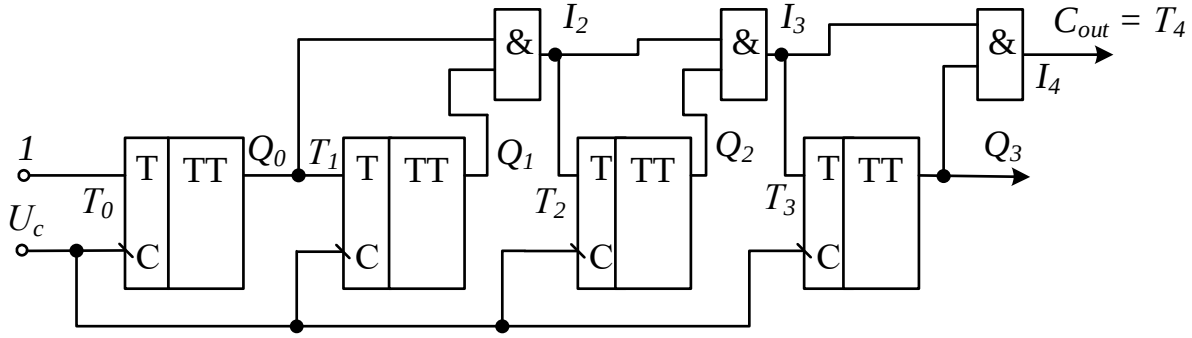


Рисунок 3.27 – Функціональна схема підсумовувального синхронного лічильника за модулем 16 з послідовним трактом розповсюдження переносу

Схема для моделювання синхронного лічильника приведена на рис.3.28.

Результати моделювання синтезованого лічильника приведені на рис.3.29. З часової діаграми можна побачити, що послідовність зміни станів

лічильника відповідає функціонуванню підсумовувального лічильника за модулем *16*.

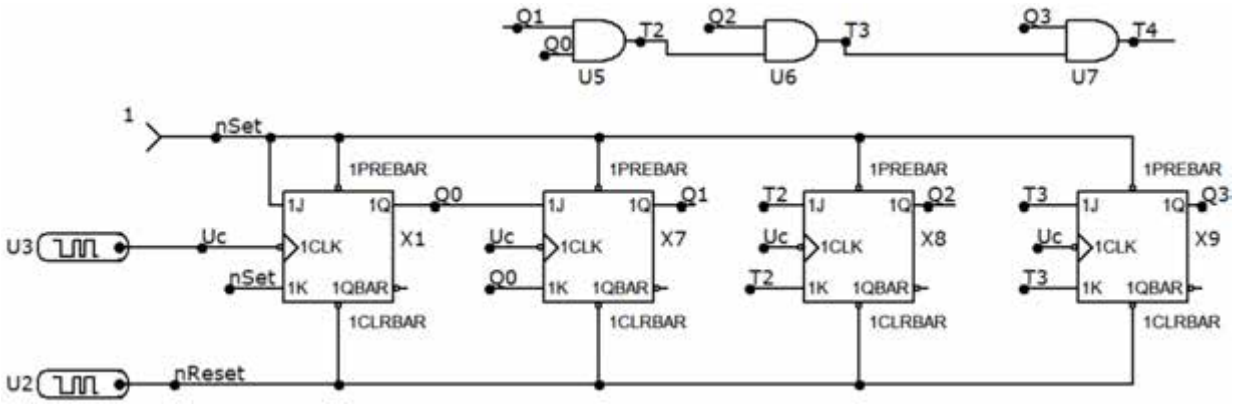


Рисунок 3.28 – Схема для моделювання підсумовувального синхронного лічильника за модулем 16 з послідовним трактом розповсюдження переносу

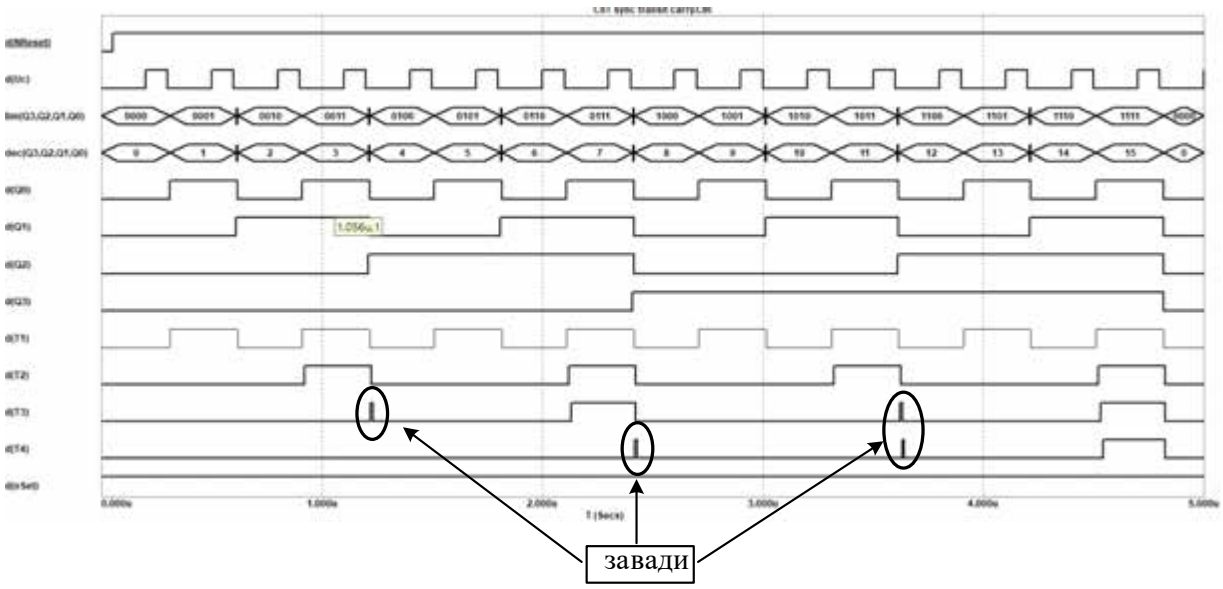


Рисунок 3.29 – Результати моделювання синхронного лічильника за модулем 16 з послідовним трактом розповсюдження переносу

Аналіз часової діаграми на рис.3.29 дозволяє отримати такі висновки про динамічні властивості лічильника:

1. На часовій діаграмі спостерігаються завади на виходах елементів *I*, що формують переноси T_3 і T_4, що свідчить про можливість несанкціонованого (відповідно до таблиці переходів) спрацювання базових тригерів лічильника.

Розглянемо більш детально причину появи першої завади (рис.3.30), яка з'являється під час переключення лічильника зі стану 3_{10} (0011_2) в стан 4_{10} (0100_2). Послідовність переключення елементів, що формують сигнал переносу T_3 на рис.3.30 показані пунктиром і пронумеровані.

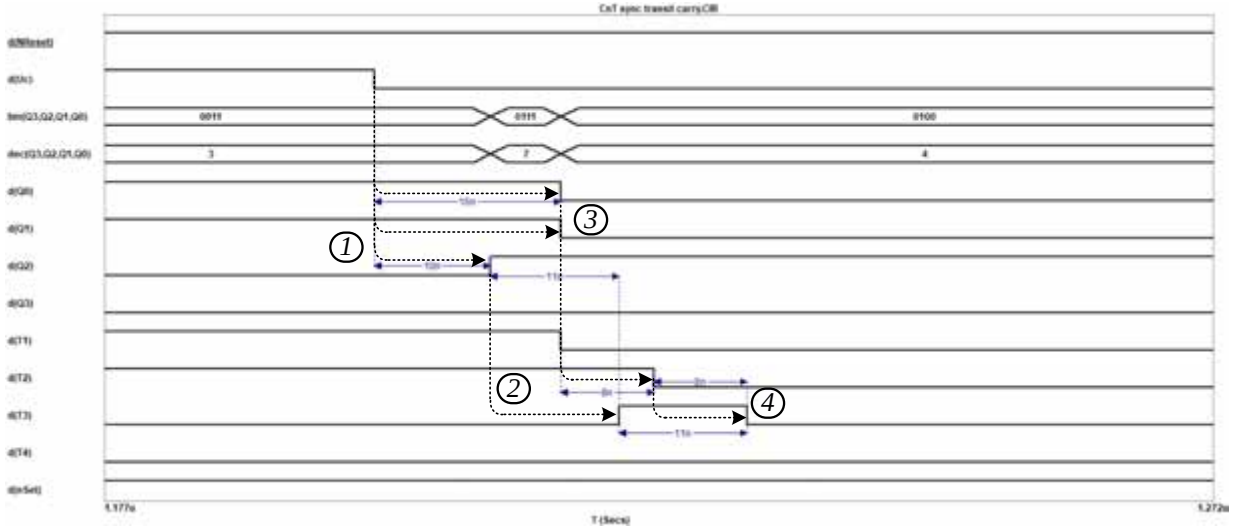


Рисунок 3.30 – Результати моделювання лічильника при появі завади на виході переносу

В момент заднього фронту сигналу U_c базові тригери лічильника Q_2, Q_1, Q_0 переключаються відповідно до таблиці переходів зі стану 0011_2 в стан 0100_2, тобто тригер Q_2 переключається $0 \rightarrow 1$, а Q_1, Q_0 – $1 \rightarrow 0$. Модель *JK*- тригера в ***MicroCap*** забезпечує за замовчуванням час переключення вихідного сигналу $0 \rightarrow 1$ за *10нс*, а $1 \rightarrow 0$ за *16нс*, що відзначено на рис.3.30 пунктиром і цифрою «1». Через *10нс* після заднього фронту сигналу U_c тригер Q_2 переключається в одиничний стан, а тригери Q_1, Q_0 поки ще перебувають в одиничному стані (вони будуть переключатися ще тільки через *6нс* після переключення Q_2). Таким чином, на виходах лічильника короткочасно формується стан 0111_2, який визиває переключення в одиничний стан елемента I_3 (рис.3.27), тобто появу завади на виході елемента, що формує сигнал переносу T_3 (на рис.3.30 позначено цифрою «2»). Але після переключення тригерів Q_1, Q_0 в нульовий стан сигнал T_2 через *8нс* переключається в нуль (цифра «3» на рис.3.30), а ще через *8нс* формується задній фронт сигналу T_3, тобто завершується формування завади. Тривалість

завади складає *11нс*. Необхідно відзначити, що поява завади відбулася вже після появи активного фронту сигналу U_c, крім того, невелика тривалість завади може привести до того, що цю заваду буде відфільтровувано на вході базового тригера в зв'язку наявності інерційної затримки. Таким чином, завади, що з'явилися, не впливають на коректне функціонування лічильника, тобто ці завади є некритичними до функціонування лічильника. Але вважаючи на те, що завада сформувалася через *21нс* після активного фронту сигналу U_c, то в разі затягнення тривалості цього фронту можливо некоректне спрацьовування тригерів лічильника.

2. На часовій діаграмі можна також спостерігати поступове скорочення тривалості сигналів переносу (рис.3.31), яке пов'язане з тим, що одиничне значення переносу з'являється в поточному розряді після спрацьовування переносів попередніх розрядів, а закінчення сигналів переносів відбувається одночасно по сигналу U_c. Таким чином, при використанні великої частоти сигналу U_c тривалість переносів буде достатньо маленькою, в результаті чого тригери в старших розрядах можуть ці сигнали відфільтровувати.

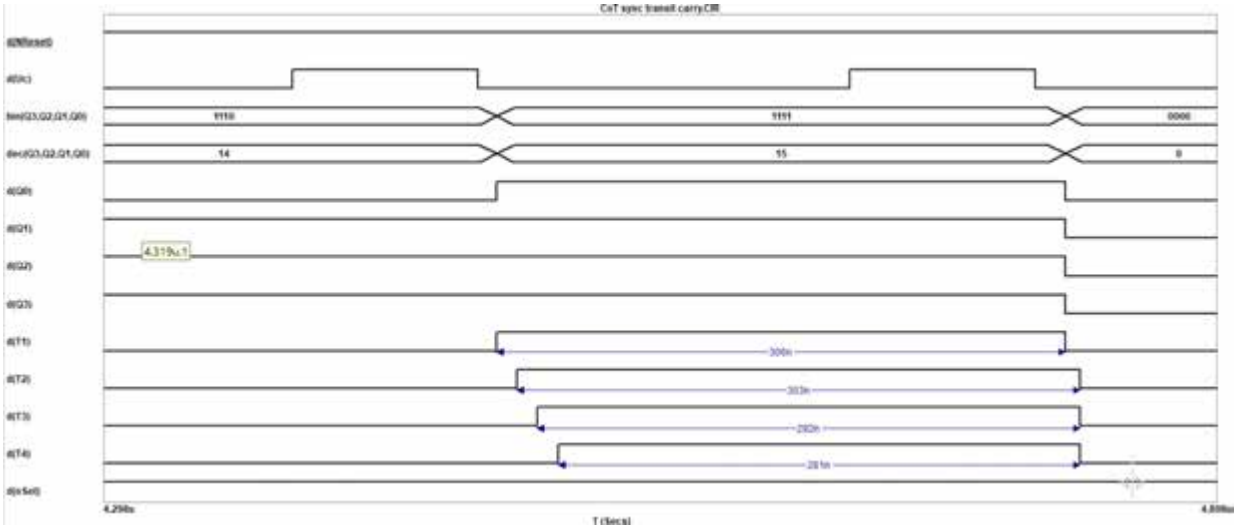


Рисунок 3.31 – Ілюстрація зменшення тривалості сигналів переносів

Визначимо динамічні параметри лічильника.

Відповідно до функціональної схеми (рис.3.27, 3.28), можна відзначити, що формування переносу T_4 з виходу елемента *U7* (рис.3.28)

відбувається після переключення базових тригерів в нульовий стан, тобто $t_{фп} = t_{тр} + t_I = 24нс$, де $t_{тр} = 16нс$, а $t_I = 8нс$, що підтверджується результатами моделювання. В зв'язку з тим, що лічильник є синхронним, то всі тригери цього пристрою спрацьовують одночасно, тобто $t_п = t_{тр} = 16нс$ (рис.3.32).

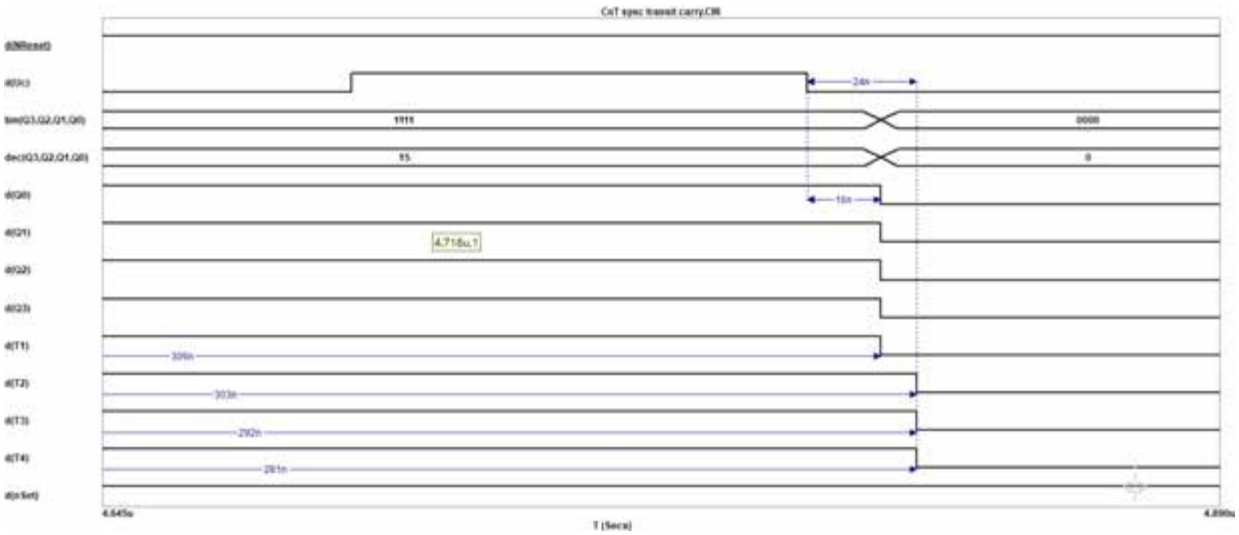


Рисунок 3.32 – Визначення динамічних параметрів синхронного лічильника за модулем 16 з послідовним трактом розповсюдження переносу

Порівнюючи структурну організацію лічильників з послідовним і паралельним трактами розповсюдження переносу, можна відзначити, що в лічильниках з паралельним трактом спостерігається ускладнення логічної схеми за рахунок збільшення кількості входів логічних схем, що формують сигнали переносу, по мірі збільшення розрядності лічильника. Лічильники з послідовним трактом розповсюдження переносу мають регулярну структуру, відповідно до якої кожний розряд лічильника складається з тригера і двовходового елемента *І*.

Контрольні завдання та запитання

1. Яким чином будуються синхронні лічильники?

2. В чому полягає різниця між асинхронними та синхронними лічильниками?

3. В чому полягає різниця між послідовним і паралельним трактом розповсюдження переносу ?

4. Як отримати логічні вирази для реалізації послідовного тракту розповсюдження переносу ?

5. Доведіть вирази (3.10).

6. Виконайте порівняння характеристик лічильників з послідовним і паралельним трактом розповсюдження переносу?

7. Які недоліки характерні для асинхронних лічильників з послідовним трактом розповсюдження переносу?

8. Які недоліки характерні для синхронних лічильників з послідовним трактом розповсюдження переносу?

9. Які переваги характерні для асинхронних лічильників з послідовним трактом розповсюдження переносу?

10. Які переваги характерні для синхронних лічильників з послідовним трактом розповсюдження переносу?

11. Яку функцію виконують логічні елементи *І* в асинхронних лічильниках з послідовним трактом розповсюдження переносу?

12. Яку функцію виконують логічні елементи *І* в синхронних лічильниках з послідовним трактом розповсюдження переносу?

13. Якими динамічними параметрами характеризується функціонування лічильників?

14. Як визначити час переключення в асинхронних лічильниках з послідовним трактом розповсюдження переносу?

15. Доведіть вираз (3.11).

16. Як визначити величину часу формування переносу зі старшого розряду в асинхронних лічильниках з послідовним трактом розповсюдження переносу?

17. Доведіть вираз (3.12).

18. Прокоментуйте результати моделювання на рис.3.26.

19. Визначити час переключення асинхронного підсумовувального лічильника з послідовним трактом розповсюдження переносу за модулем 32, якщо $t_I = 10нс$; $t_{тр} = 20нс$.

20. Визначити час переключення асинхронного віднімального лічильника з послідовним трактом розповсюдження переносу за модулем 64, якщо $t_I = 10нс$; $t_{тр} = 20нс$.

21. Визначити час формування переносу зі старшого розряду асинхронного підсумовувального лічильника з послідовним трактом розповсюдження переносу за модулем 32, якщо $t_I = 10нс$; $t_{тр} = 20нс$.

22. Визначити час формування переносу зі старшого розряду асинхронного віднімального лічильника з послідовним трактом розповсюдження переносу за модулем 64, якщо $t_I = 10нс$; $t_{тр} = 20нс$.

23. Чи можна використовувати у складі лічильника прозорі тригери? Обґрунтуйте відповідь.

24. Як розрізняються лічильники за способом синхронізації?

25. З якою метою використовується сигнал переносу зі старшого розряду?

26. Яким чином виконується синтез асинхронних лічильниках з послідовним трактом розповсюдження переносу?

27. Виконати синтез асинхронного підсумовувального лічильника за модулем *8* з послідовним трактом розповсюдження переносу на базі асинхронних *T*-тригерів. Побудувати часові діаграми лічильника з використанням затримок логічних елементів.

28. Виконати синтез асинхронного віднімального лічильника за модулем *8* з послідовним трактом розповсюдження переносу на базі асинхронних *T*-тригерів. Побудувати часові діаграми лічильника з використанням затримок логічних елементів.

29. Виконати синтез синхронного підсумовувального лічильника за модулем *32* з послідовним трактом розповсюдження переносу на базі

синхронних *TC*-тригерів. Побудувати часові діаграми функціонування лічильника з використанням затримок логічних елементів.

30. Виконати синтез синхронного віднімального лічильника за модулем *32* з послідовним трактом розповсюдження переносу на базі синхронних *TC*- тригерів. Побудувати часові діаграми функціонування лічильника з використанням затримок логічних елементів.

31. Виконати синтез синхронного підсумовувального лічильника за модулем *8* з послідовним трактом розповсюдження переносу на базі *JK*- тригерів. Побудувати часові діаграми функціонування лічильника з використанням затримок логічних елементів.

32. Виконати синтез синхронного віднімального лічильника за модулем *8* з послідовним трактом розповсюдження переносу на базі *JK*- тригерів. Побудувати часові діаграми функціонування лічильника з використанням затримок логічних елементів.

33. Як розрахувати мінімальний період надходження лічильного сигналу?

34. Для чого використовується сигнал ***nReset*** в лічильниках?

35. Чому може зменшуватися тривалість сигналів переносів в старших розрядах в асинхронних лічильниках з послідовним трактом розповсюдження переносу?

36. Як може вплинути зменшення тривалості сигналів переносів на функціонування лічильника?

37. Доведіть вираз (3.13).

38. Виконайте порівняння характеристик синхронних і асинхронних лічильників з послідовним трактом розповсюдження переносу?

39. Як визначити час переключення в синхронних лічильниках з послідовним трактом розповсюдження переносу?

40. Прокоментуйте результати моделювання на рис.3.29.

41. Як визначити величину часу формування переносу зі старшого розряду в синхронних лічильниках з послідовним трактом розповсюдження переносу?

42. Визначити час переключення синхронного підсумовувального лічильника з послідовним трактом розповсюдження переносу за модулем 32, якщо $t_I = 10нс$; $t_{тр} = 20нс$.

43. Визначити час переключення синхронного віднімального лічильника з послідовним трактом розповсюдження переносу за модулем 64, якщо $t_I = 10нс$; $t_{тр} = 20нс$.

44. Визначити час формування переносу зі старшого розряду синхронного підсумовувального лічильника з послідовним трактом розповсюдження переносу за модулем 32, якщо $t_I = 10нс$; $t_{тр} = 20нс$.

45. Визначити час формування переносу зі старшого розряду синхронного віднімального лічильника з послідовним трактом розповсюдження переносу за модулем 64, якщо $t_I = 10нс$; $t_{тр} = 20нс$.

46. Як розрізняються лічильники за способом організації переносу між розрядами ?

47. Яким чином виконується синтез синхронних лічильниках з послідовним трактом розповсюдження переносу?

48. Виконати синтез синхронного підсумовувального лічильника за модулем *8* з послідовним трактом розповсюдження переносу на базі синхронних *TC*-тригерів. Побудувати часові діаграми лічильника з використанням затримок логічних елементів.

49. Виконати синтез синхронного віднімального лічильника за модулем *8* з послідовним трактом розповсюдження переносу на базі асинхронних *TC*- тригерів. Побудувати часові діаграми лічильника з використанням затримок логічних елементів.

50. Виконати синтез синхронного підсумовувального лічильника за модулем *32* з послідовним трактом розповсюдження переносу на базі

синхронних *TC*-тригерів. Побудувати часові діаграми функціонування лічильника з використанням затримок логічних елементів.

51. Виконати синтез синхронного віднімального лічильника за модулем *32* з послідовним трактом розповсюдження переносу на базі синхронних *TC*- тригерів. Побудувати часові діаграми функціонування лічильника з використанням затримок логічних елементів.

52. Виконати синтез синхронного підсумовувального лічильника за модулем *8* з послідовним трактом розповсюдження переносу на базі *JK*- тригерів. Побудувати часові діаграми функціонування лічильника з використанням затримок логічних елементів.

53. Виконати синтез синхронного віднімального лічильника за модулем *8* з послідовним трактом розповсюдження переносу на базі *JK*- тригерів. Побудувати часові діаграми функціонування лічильника з використанням затримок логічних елементів.

54. Як розрахувати мінімальний період надходження лічильного сигналу в синхронних лічильниках з послідовним трактом розповсюдження переносу?

55. Поясніть появу завад під час функціонування синхронних лічильників з послідовним трактом розповсюдження переносу.

56. Прокоментуйте результати моделювання на рис.3.30.

57. Поясніть причину скорочення сигналів переносу в синхронних лічильниках з послідовним трактом розповсюдження переносу.

58. Прокоментуйте результати моделювання на рис.3.31.

59. Яким чином впливає зменшення тривалості сигналів переносів на функціонування лічильника?

60. Прокоментуйте результати моделювання на рис.3.32.

61. Як побудувати синхронний лічильник з послідовним трактом розповсюдження переносу за результатами синтезу асинхронного лічильника?

62. Виконайте порівняльний аналіз динамічних параметрів синхронних і асинхронних лічильників з послідовним трактом розповсюдження переносу.

63. Виконайте порівняльний аналіз динамічних параметрів асинхронних лічильників з послідовним і паралельним трактами розповсюдження сигналу переносу.

64. Виконайте порівняльний аналіз динамічних параметрів синхронних лічильників з послідовним і паралельним трактами розповсюдження сигналу переносу.

65. Виконайте порівняльний аналіз структурної організації асинхронних лічильників з послідовним і паралельним трактами розповсюдження сигналу переносу.

66. Виконайте порівняльний аналіз структурної організації синхронних лічильників з послідовним і паралельним трактами розповсюдження сигналу переносу.

3.1.3 Двійкові лічильники з безпосереднім переносом

3.1.3.1 Асинхронні двійкові лічильники з безпосереднім переносом

Для виконання синтезу будемо використовувати таке позначення: $X^t \rightarrow X^{t+1}$, де X^t (X^{t+1}) – відповідно стани логічного сигналу в попередній (наступний) такт будь-якого цифрового пристрою.

Далі виконаємо розробку таблиці переходів на прикладі синтезу підсумовувального лічильника за модулем *8* на базі асинхронного *T*-тригера зі спрацьовуванням за заднім фронтом сигналу синхронізації.

Відповідно до (3.1) для реалізації лічильника необхідно використовувати] $log_2 8$ [= *3* тригери.

Таблицю переходів лічильника, яка приведена в табл.3.5, будемо складати тільки для активного фронту вхідного сигналу U_c, тобто для

переключення сигналу на вході тригера з одиничного рівня в нульовий (*1 → 0*).

Таблиця 3.5 – Таблиця переходів для синтезу підсумовувального лічильника з безпосереднім переносом

Номери наборів	U_c	$Q_2^t \to Q_2^{t+1}$	$Q_1^t \to Q_1^{t+1}$	$Q_0^t \to Q_0^{t+1}$
1	2	3	4	5
0	*1 → 0*	*0 → 0*	*0 → 0*	*0 → 1*
1	*1 → 0*	*0 → 0*	*0 → 1*	*1 → 0*
2	*1 → 0*	*0 → 0*	*1 → 1*	*0 → 1*
3	*1 → 0*	*0 → 1*	*1 → 0*	*1 → 0*
4	*1 → 0*	*1 → 1*	*0 → 0*	*0 → 1*
5	*1 → 0*	*1 → 1*	*0 → 1*	*1 → 0*
6	*1 → 0*	*1 → 1*	*1 → 1*	*0 → 1*
7	*1 → 0*	*1 → 0*	*1 → 0*	*1 → 0*

В таблиці показано переключення тригерів лічильника по задньому фронту сигналу U_c. Наприклад, в рядку 5 по сигналу U_c лічильник переключається зі стану Q_2^t, Q_1^t, $Q_0^t = 101_2 = 5_{10}$ в новий стан $Q_2^{t+1}, Q_1^{t+1}, Q_0^{t+1} = 110_2 = 6_{10}$. З таблиці можна помітити, що переключення наймолодшого тригера Q_0 (колонка *5*) відбувається по кожному задньому фронту сигналу U_c, тобто достатньо під'єднати цей сигнал до входу T_0 цього тригера. Аналізуючи стовпчик *4*, можна побачити, що переключення розряду Q_1 відбувається за заднім фронтом Q_0 (ці випадки позначені в таблиці сірим кольором), а в інших випадках тригер Q_1 перебуває в стані збереження інформації. Це означає, що для забезпечення апаратної реалізації лічильника вхід T тригера Q_1 достатньо з'єднати з виходом Q_0. Аналогічно для тригера Q_2 отримаємо, що цей тригер переключається тільки за заднім фронтом Q_1 (це теж позначено сірим кольором), тобто вхід T тригера Q_2 достатньо з'єднати з виходом Q_1.

Для того, щоб отримати лічильник за модулем *16* за аналогією додаємо ще один тригер Q_3, вхід якого підключимо до виходу Q_2. В результаті отримаємо підсумовувальний лічильник за модулем *16* з безпосереднім переносом, схема якого приведена на рис.3.33.

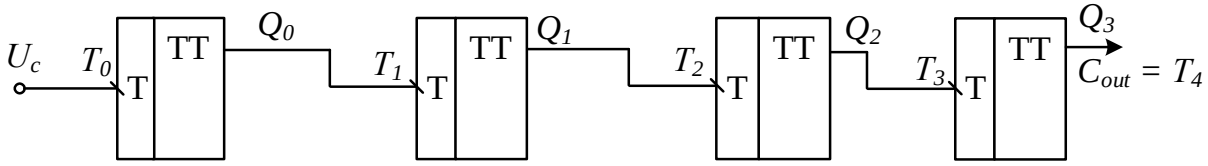


Рисунок 3.33 – Функціональна схема підсумовувального асинхронного лічильника за модулем *16* з безпосереднім переносом

Для того, щоб отримати логічні вирази для функцій збудження базових тригерів будемо використовувати такі позначення: $dX\,(d\overline{X})$, де $dX\,(d\overline{X})$ відповідають передньому (задньому) фронту сигналу *X*. Таблиця переходів розглянутого вище лічильника приведена в табл.3.6.

Таблиця 3.6 – Таблиця переходів підсумовувального лічильника з безпосереднім переносом та позначенням фронтів сигналів

Номери наборів	U_c	Попередній стан			Наступний стан			Функції збудження		
		Q_2^t	Q_1^t	Q_0^t	Q_2^{t+1}	Q_1^{t+1}	Q_0^{t+1}	T_2	T_1	T_0
1	2	3	4	5	6	7	8	9	10	11
0	$d\overline{U_c}$	0	0	0	0	0	dQ_0	0	0	$d\overline{T_0}$
1	$d\overline{U_c}$	0	0	1	0	dQ_1	$d\overline{Q_0}$	0	$d\overline{T_1}$	$d\overline{T_0}$
2	$d\overline{U_c}$	0	1	0	0	1	dQ_0	0	0	$d\overline{T_0}$
3	$d\overline{U_c}$	0	1	1	dQ_2	$d\overline{Q_1}$	$d\overline{Q_0}$	$d\overline{T_2}$	$d\overline{T_1}$	$d\overline{T_0}$
4	$d\overline{U_c}$	1	0	0	1	0	dQ_0	0	0	$d\overline{T_0}$
5	$d\overline{U_c}$	1	0	1	1	dQ_1	$d\overline{Q_0}$	0	$d\overline{T_1}$	$d\overline{T_0}$
6	$d\overline{U_c}$	1	1	0	1	1	dQ_0	0	0	$d\overline{T_0}$
7	$d\overline{U_c}$	1	1	1	$d\overline{Q_2}$	$d\overline{Q_1}$	$d\overline{Q_0}$	$d\overline{T_2}$	$d\overline{T_1}$	$d\overline{T_0}$

Далі можна отримати функції збудження базових тригерів:

$$T_0 = d\overline{U_c};\ T_1 = d\overline{Q_0};\ T_2 = d\overline{Q_1}. \qquad (3.14)$$

Наприклад, з таблиці 3.6 можна побачити, що на вході T_2 (стовпчик 9) виникає задній фронт ($d\overline{T_2}$) тільки в тому разі, якщо цей фронт з'являється на якомусь виході, тобто в даному випадку тільки з виходу Q_1 ($d\overline{Q_1}$.), що записано у відповідних клітинах (стовпчик 7, рядки 3 і 7). Функціональна схема лічильника на рис.3.33 відповідає виразам (3.14).

Ці функції збудження можна записати у вигляді:

$$T_0 = U_c; \qquad T_1 = Q_0; \qquad T_2 = Q_1. \tag{3.15}$$

Аналізуючи вирази (3.15) в загальному випадку для будь-якої кількості розрядів можна отримати:

$$T_0 = U_c; \qquad T_i = Q_{i-1}; \qquad i = 1, n_{\text{тр}} - 1. \tag{3.16}$$

Схема для моделювання функціонування підсумовувального лічильника за модулем *16* з безпосереднім переносом відповідно до (3.16) приведена на рис.3.34.

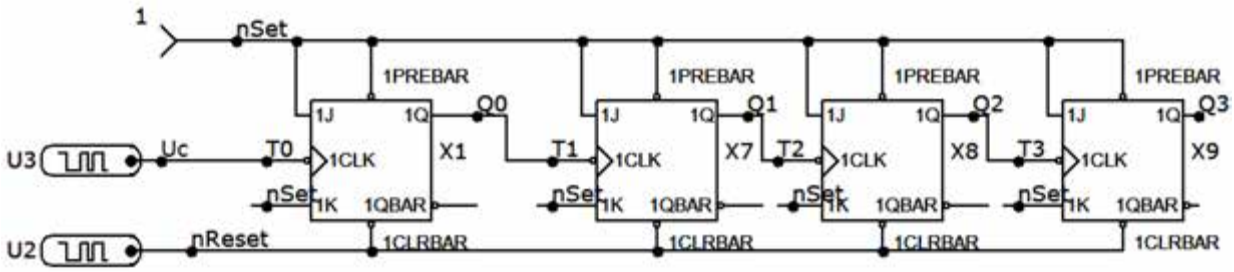


Рисунок 3.34 – Схема для моделювання підсумовувального асинхронного лічильника за модулем *16* з безпосереднім переносом

Результати моделювання цього лічильника приведені на рис.3.35 та свідчать про коректний синтез лічильника.

Визначимо динамічні параметри лічильника.

Відповідно до структурної організації функціональної схеми, що приведена на рис.3.33, 3.34, можна відзначити, що задній фронт сигналу на виході тригера визиває переключення наступного тригера, тобто тригери лічильника спрацьовують послідовно. В результаті час переключення лічильника визначається за виразом: $t_{\text{п}} = n_{\text{тр}} \cdot t_{\text{тр}}$. За цим же виразом

визначається час формування переносу зі старшого розряду $t_{фп}$, який формується на виході найстаршого тригера.

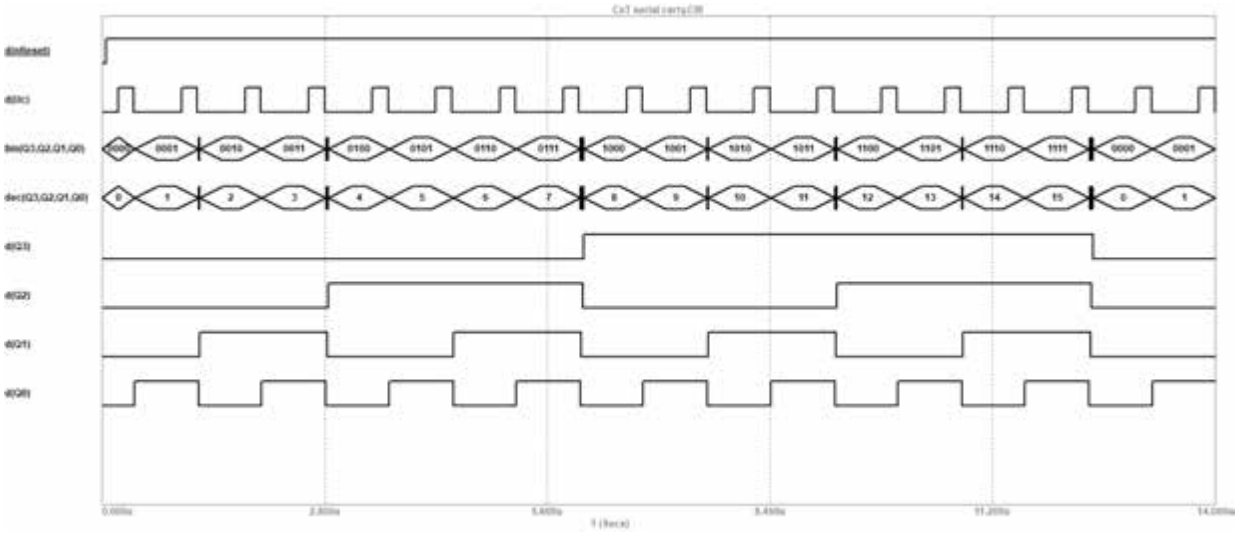


Рисунок 3.35 – Результати моделювання підсумовувального асинхронного лічильника за модулем *16* з безпосереднім переносом

Результати вимірів $t_п$ і $t_{фп}$ приведені на рис.3.36 для переключення лічильника зі стану 1111_2 в стан *0000*.

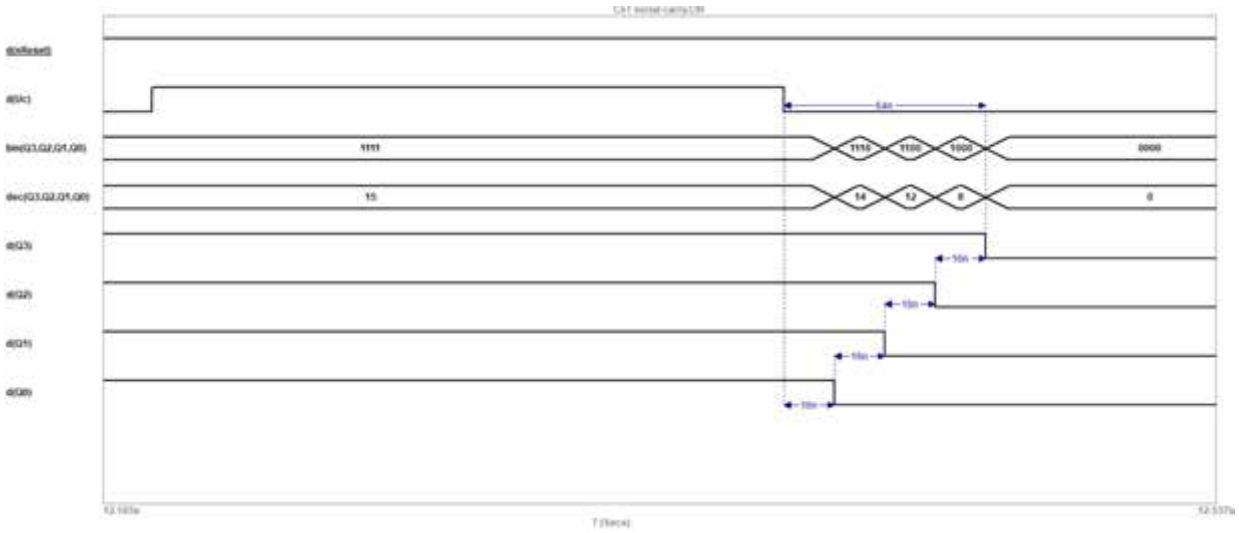


Рисунок 3.36 – Визначення динамічних параметрів підсумовувального асинхронного лічильника за модулем *16* з безпосереднім переносом

На часовій діаграмі чітко видно послідовне спрацьовування тригерів, починаючи з наймолодшого, тобто з урахуванням, що час спрацьовування тригера $t_{тр}$ при переключенні з одиничного стану до нульового складає *16нс*,

то можна отримати $t_п = t_{фп} = n_{тр} \cdot t_{тр} = 4 \cdot 16 = 64нс$, що підтверджується результатами моделювання.

Далі розглянемо реалізацію лічильників з безпосереднім переносом на базі тригерів зі спрацьовуванням за переднім фронтом синхросигналу.

В зв'язку з тим, що функціонування цього лічильника не змінюється, то таблиця переходів такого лічильника аналогічна таблиці 3.5 і відрізняється тільки стовпчиком *2*, в якому необхідно позначити передній фронт сигналу U_c: $0 \rightarrow 1$. Приведемо змінену таблицю переходів в табл.3.7.

Таблиця 3.7 – Таблиця переходів для синтезу лічильника з безпосереднім переносом зі спрацьовуванням базових тригерів за переднім фронтом

Номери наборів	U_c	$Q_2^t \rightarrow Q_2^{t+1}$	$Q_1^t \rightarrow Q_1^{t+1}$	$Q_0^t \rightarrow Q_0^{t+1}$
1	2	3	4	5
0	$0 \rightarrow 1$	$0 \rightarrow 0$	$0 \rightarrow 0$	$0 \rightarrow 1$
1	$0 \rightarrow 1$	$0 \rightarrow 0$	$0 \rightarrow 1$	$1 \rightarrow 0$
2	$0 \rightarrow 1$	$0 \rightarrow 0$	$1 \rightarrow 1$	$0 \rightarrow 1$
3	$0 \rightarrow 1$	$0 \rightarrow 1$	$1 \rightarrow 0$	$1 \rightarrow 0$
4	$0 \rightarrow 1$	$1 \rightarrow 1$	$0 \rightarrow 0$	$0 \rightarrow 1$
5	$0 \rightarrow 1$	$1 \rightarrow 1$	$0 \rightarrow 1$	$1 \rightarrow 0$
6	$0 \rightarrow 1$	$1 \rightarrow 1$	$1 \rightarrow 1$	$0 \rightarrow 1$
7	$0 \rightarrow 1$	$1 \rightarrow 0$	$1 \rightarrow 0$	$1 \rightarrow 0$

З таблиці видно, що наймолодший тригер Q_0 (стовпчик *5*) переключається за переднім фронтом сигналу U_c (стовпчик *2*), тобто сигнал U_c необхідно підключити до входу *T* тригера Q_0. Однак з таблиці витікає, що тригер Q_1 (стовпчик *4*) переключається тільки при появі заднього фронту на виході тригера Q_0 (стовпчик *5*). Аналогічно тригер Q_2 (стовпчик *3*) переключається тільки при появі заднього фронту на виході тригера Q_1 (стовпчик *4*). Таким чином, для забезпечення функціонування лічильника необхідно сигнали заднього фронту на виходах Q_0, Q_1, а в загальному випадку при будь-якій розрядності лічильника і на інших тригерах, перетворити в передній фронт сигналу. Звичайно, що це робиться за допомогою

інвертування заднього фронту сигналу. Враховуючи наявність інверсних виходів тригерів, реалізація лічильника полягає у з'єднанні виходів $\overline{Q_0}$, $\overline{Q_1}$ та інших інверсних виходів тригерів зі входом *T* наступних тригерів. В результаті в загальному випадку отримаємо такі функції збудження:

$$T_0 = U_c; \qquad T_i = \overline{Q_{i-1}}, ; \quad i = 1, n_{\text{тр}} - 1. \tag{3.17}$$

Функціональна схема такого лічильника відповідно до (3.17) приведена на рис.3.37.

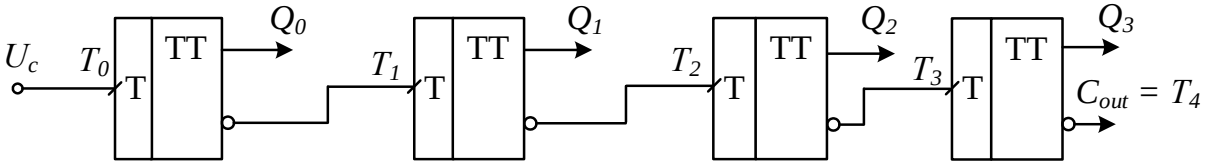


Рисунок 3.37 – Функціональна схема підсумовувального асинхронного лічильника за модулем *16* з безпосереднім переносом на базі тригерів зі спрацьовуванням за переднім фронтом

Схема для моделювання функціонування підсумовувального лічильника за модулем *16* з безпосереднім переносом на базі тригерів зі спрацьовуванням за переднім фронтом відповідно до (3.17) приведена на рис.3.38. В якості базових тригерів використовуються *JK*-тригери *SN74109* зі спрацьовуванням за переднім фронтом синхросигналу [9]. У цих тригерів вхід *K* є інверсним (*nK*), тому для перетворення цього тригера в асинхронний *T*- тригер необхідно забезпечити *J = 1*, *nK = 0*.

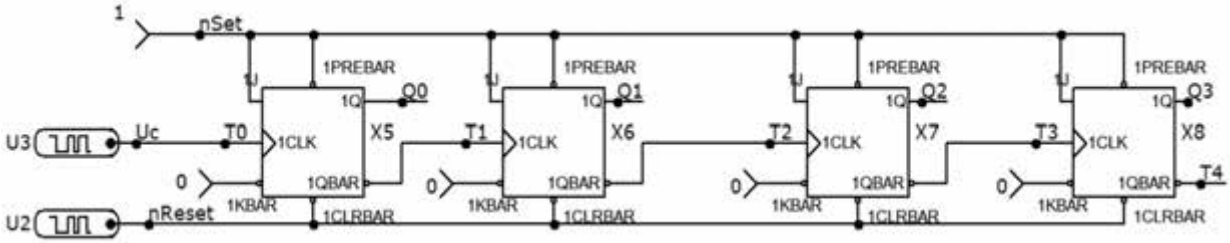


Рисунок 3.38 – Схема для моделювання підсумовувального асинхронного лічильника за модулем *16* з безпосереднім переносом на базі тригерів зі спрацьовуванням за переднім фронтом

Результати моделювання цього лічильника приведені на рис.3.39 та свідчать про коректну реалізацію схеми лічильника.

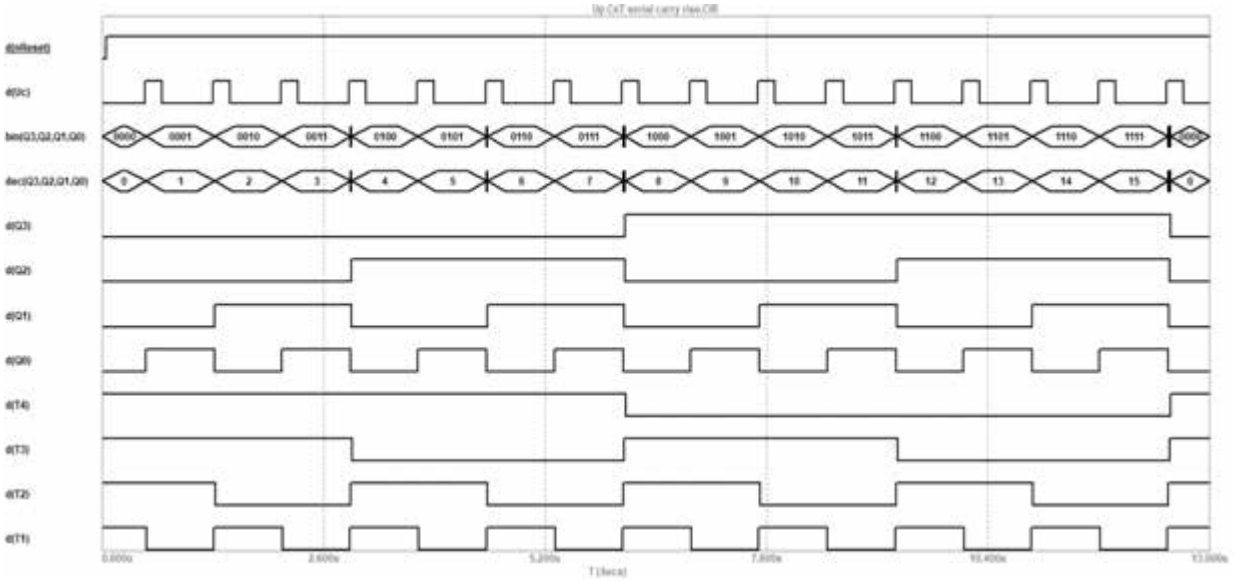


Рисунок 3.39 – Результати моделювання підсумовувального асинхронного лічильника за модулем *16* з безпосереднім переносом на базі тригерів зі спрацьовуванням за переднім фронтом

Визначимо динамічні параметри лічильника.

Відповідно до структурної організації функціональної схеми, що приведена на рис.3.37, 3.38, можна відзначити, що передній фронт сигналу на інверсному виході тригера визиває переключення наступного тригера, тобто тригери лічильника спрацьовують послідовно.

В результаті час переключення лічильника визначається за виразом: $t_{\text{п}} = n_{\text{тр}} \cdot t_{\text{тр}}^{10}$, де $t_{\text{тр}}^{10}$ – час спрацьовування тригера при переключенні з *1* в *0*. Перенос зі старшого розряду формується на інверсному виході найстаршого тригера $\overline{Q_3}$, причому цей сигнал переключається з *0* в *1*. Тому час формування переносу зі старшого розряду $t_{\text{фп}}$ обчислюється за виразом $t_{\text{фп}} = n_{\text{тр}} \cdot t_{\text{тр}}^{01}$, де $t_{\text{тр}}^{01}$ – час спрацьовування тригера при переключенні з *0* в *1*.

Результати вимірів $t_{\text{п}}$ і $t_{\text{фп}}$ приведені на рис.3.40 для переключення лічильника зі стану 1111_2 в стан *0000*. На відміну від попереднього лічильника (рис.3.34) параметри $t_{\text{п}}$ і $t_{\text{фп}}$ не співпадають за значенням в зв'язку з різним часом спрацьовування тригерів $0 \rightarrow 1$ і $1 \rightarrow 0$. За замовчуванням в ***MicroCap*** час переключення тригера в одиничний стан (для інверсного виходу) складає *10нс*, тобто $t_{\text{фп}} = n_{\text{тр}} \cdot t_{\text{тр}}^{01} = 40$*нс*.

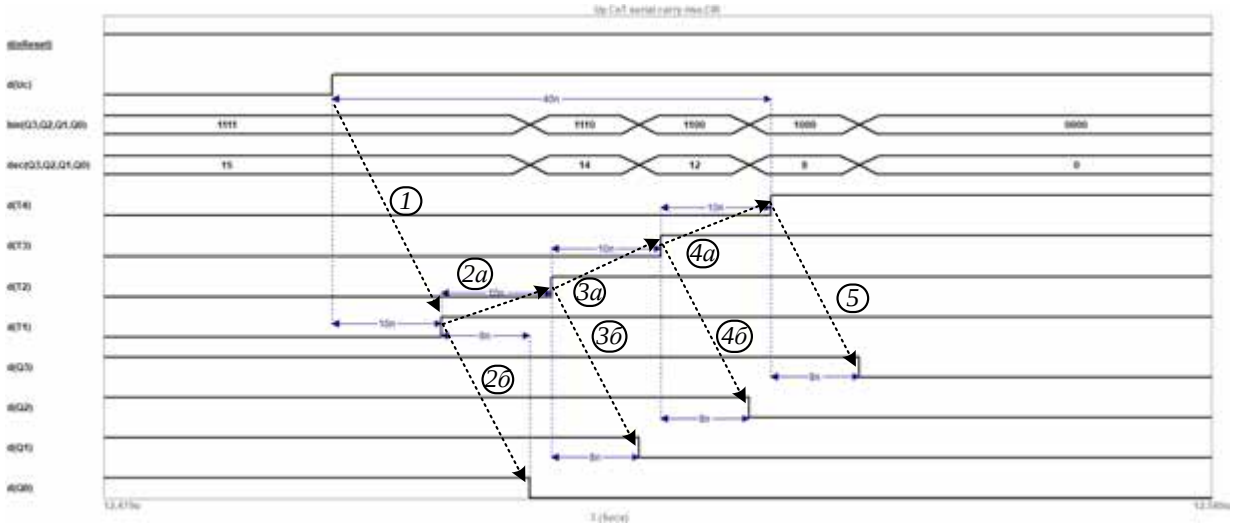


Рисунок 3.40 – Визначення динамічних параметрів підсумовувального асинхронного лічильника за модулем *16* з безпосереднім переносом на базі тригерів зі спрацьовуванням за переднім фронтом

Далі визначимо час переключення лічильника. Послідовність переключення складових елементів показана на рис.3.40 за допомогою пронумерованих стрілок. Передній фронт сигналу U_c спричиняє переключення наймолодшого тригера Q_0 і через *10нс* відбувається переключення інверсного виходу цього тригера, який формує сигнал переносу в тригер Q_1 (це позначено цифрою «*1*»). Після цього починається спрацьовування тригера Q_1 (позначено «*2а*»), а також продовжується переключення в нуль прямого виходу тригера Q_0, яке завершується ще через *8нс* («*2б*»), тобто формується новий вихідний сигнал наймолодшого тригера лічильника. Ще через *10нс* відбувається переключення інверсного виходу тригера Q_1(«*2а*»), який формує сигнал переносу T_2 в тригер Q_2.

Після цього починається спрацьовування інверсного виходу тригера Q_2 («*3а*»), яке завершується через *10нс*, а також продовжується переключення в нуль прямого виходу тригера Q_1, яке завершується через *8нс* («*3б*») після появи T_2, тобто формується новий вихідний сигнал тригера Q_1.

Далі за аналогією починається спрацьовування інверсного виходу тригера Q_3 («*4а*»), яке завершується через *10нс*, а також продовжується

переключення в нуль прямого виходу тригера Q_2, яке завершується через *8нс* («*4б*») після появи T_2, тобто формується новий вихідний сигнал тригера Q_2.

Після завершення переключення інверсного виходу тригера Q_3 продовжується переключення в нуль прямого виходу цього тригера, яке завершується через *8нс* («5») після появи T_3, тобто формується новий вихідний сигнал тригера Q_3.

Таким чином, формування остаточного коду на виході лічильника завершується після спрацювання по черзі інверсних виходів всіх тригерів («*1*», «*2а*», «*3а*», «*4а*») і формування вихідного сигналу найстаршого тригера Q_3 («5»), що складає *48нс*. В загальному випадку час спрацьовування лічильника визначається за виразом:

$$t_{\text{п}} = \sum_{i=1}^{n_{\text{тр}}-1} (t_{\text{фп}_i}) + t_{\text{тр}}^{10}, \qquad (3.18)$$

де $t_{\text{фп}_i} = 10\text{нс}$ – час формування переносу в *i*-тий розряд тригера;

$t_{\text{тр}}^{10} = 18\text{нс}$ відповідно до результатів моделювання.

Далі розглянемо синтез віднімальних лічильників з безпосереднім переносом.

Як вже було відзначено раніше в п.3.1.1.1 (рис.3.15), в підсумовувальних лічильниках прямі (інверсні) виходи базових тригерів реалізують послідовність кодів станів, що відповідають виконанню мікрооперації «інкремент» («декремент»), а у віднімальних лічильників – навпаки. Але, як правило, в лічильниках користувач не має доступу до інверсних виходів лічильників, тому розглянемо синтез віднімальних лічильників з безпосереднім переносом.

Синтез будемо виконувати аналогічно тому, як це описано вище для підсумовувальних лічильників.

В табл.3.8 приведена таблиця переходів віднімального лічильника за модулем *8* на основі базових тригерів зі спрацьовуванням за переднім фронтом сигналу синхронізації.

Таблиця 3.8 – Таблиця переходів для синтезу віднімального лічильника з безпосереднім переносом зі спрацьовуванням тригерів за переднім фронтом

Номери наборів	U_c	$Q_2^t \to Q_2^{t+1}$	$Q_1^t \to Q_1^{t+1}$	$Q_0^t \to Q_0^{t+1}$
1	2	3	4	5
0	*0 → 1*	*0 → 1*	*0 → 1*	*0 → 1*
1	*0 → 1*	*0 → 0*	*0 → 0*	*1 → 0*
2	*0 → 1*	*0 → 0*	*1 → 0*	*0 → 1*
3	*0 → 1*	*0 → 0*	*1 → 1*	*1 → 0*
4	*0 → 1*	*1 → 0*	*0 → 1*	*0 → 1*
5	*0 → 1*	*1 → 1*	*0 → 0*	*1 → 0*
6	*0 → 1*	*1 → 1*	*1 → 0*	*0 → 1*
7	*0 → 1*	*1 → 1*	*1 → 1*	*1 → 0*

З цієї таблиці можна побачити, що наймолодший тригер Q_0 (стовпчик *5*) спрацьовує за переднім фронтом сигналу U_c (стовпчик *2*), тобто сигнал U_c необхідно підключити до входу *T* цього тригера. Тригер Q_1 (стовпчик *4*) спрацьовує тільки за переднім фронтом сигналу Q_0 (виділене сірим кольором в таблиці 3.8), тобто сигнал з виходу Q_0 необхідно підключити до входу *T* тригера Q_1. За аналогією тригер Q_2 (стовпчик *3*) спрацьовує тільки за переднім фронтом сигналу Q_1 (виділене сірим кольором), тобто сигнал з виходу Q_1 необхідно підключити до входу *T* тригера Q_2.

Таким же чином для реалізації лічильника з будь-яким модулем ліку необхідно вихід найстаршого тригера існуючого лічильника з'єднати зі входом *T* тригера, що додається. Цю операцію можна виконувати кілька разів, щоб отримати заданий лічильник. Наприклад, для реалізації лічильника за модулем *16* сигнал з виходу Q_2 від'єднуємо до входу *T* додаткового тригера Q_3. Функціональна схема підсумовувального лічильника за модулем *16* з безпосереднім переносом на базі тригерів зі спрацьовуванням за переднім фронтом приведена на рис.3.41.

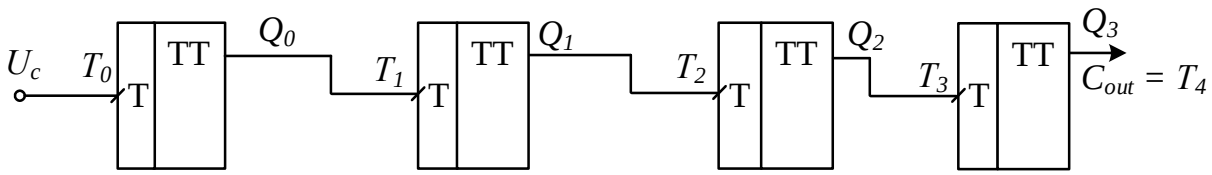


Рисунок 3.41 – Функціональна схема віднімального асинхронного лічильника за модулем *16* з безпосереднім переносом на базі тригерів зі спрацьовуванням за переднім фронтом

Схема для моделювання функціонування віднімального лічильника за модулем *16* з безпосереднім переносом на базі тригерів зі спрацьовуванням за переднім фронтом приведена на рис.3.42. Як і в попередньому лічильнику в якості базових тригерів будемо використовувати *JK*-тригери *SN74109*.

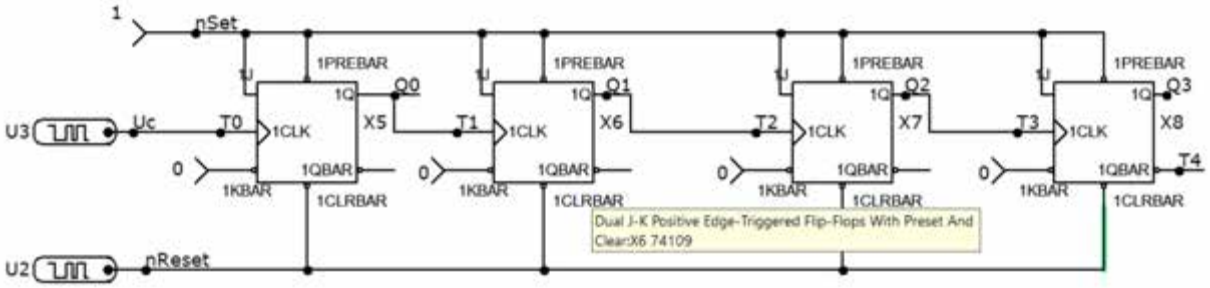


Рисунок 3.42 – Схема для моделювання віднімального асинхронного лічильника за модулем *16* з безпосереднім переносом на базі тригерів зі спрацьовуванням за переднім фронтом

Результати моделювання цього лічильника приведені на рис.3.43 та свідчать про коректну реалізацію схеми лічильника.

Динамічні параметри визначаються аналогічно тому, як це було виконано для підсумовувальних лічильників з безпосереднім переносом.

Розглянемо синтез віднімальних лічильників на основі базових тригерів зі спрацьовуванням за заднім фронтом синхросигналу. Нагадаємо, що під час розгляду синтезу підсумовувальних лічильників з безпосереднім переносом в схемі лічильника на базі тригерів зі спрацьовуванням за заднім фронтом сигналу синхронізації для реалізації лічильників зі спрацьовуванням за переднім фронтом виконувалася заміна базових тригерів на тригери з протилежним активним фронтом, а сигнали переносу формувалися на інверсних виходів базових тригерів.

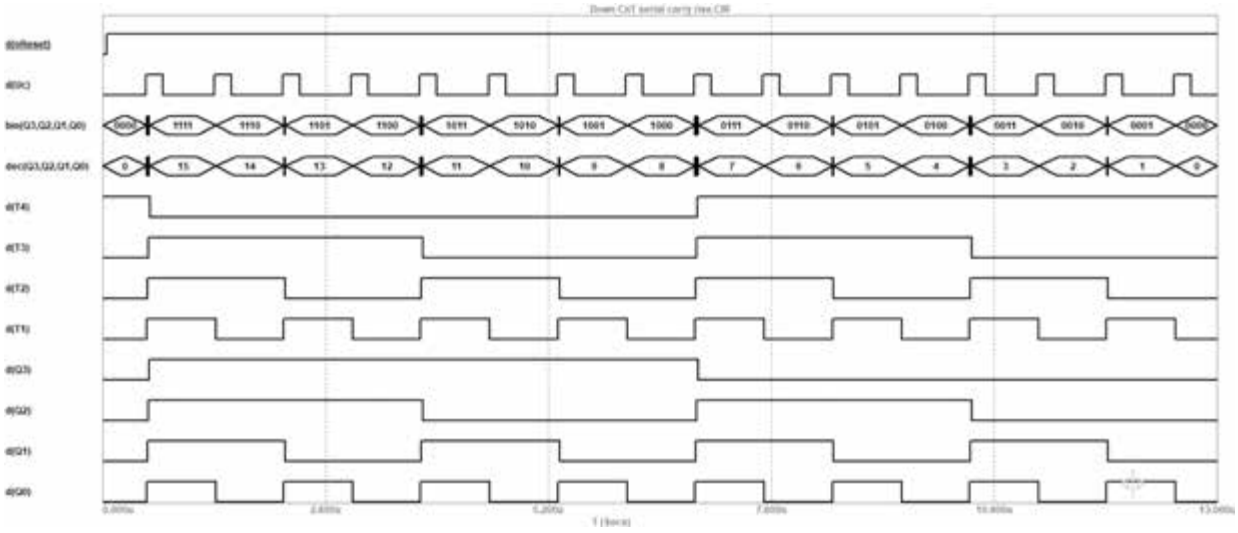


Рисунок 3.43 – Результати моделювання віднімального асинхронного лічильника за модулем *16* з безпосереднім переносом на базі тригерів зі спрацьовуванням за переднім фронтом

Виконуючи такі ж самі дії для реалізації віднімальних лічильників, отримаємо функціональну схему (рис.3.44) такого лічильника з використанням базових тригерів зі спрацьовуванням за заднім фронтом сигналу синхронізації.

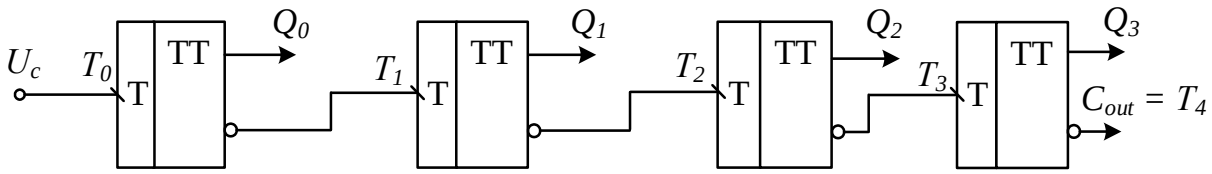


Рисунок 3.44 – Функціональна схема віднімального асинхронного лічильника за модулем *16* з безпосереднім переносом на базі тригерів зі спрацьовуванням за заднім фронтом

Схема для моделювання цього лічильника приведена на рис.3.45.

Результати моделювання цього лічильника приведені на рис.3.46 та свідчать про коректну реалізацію схеми лічильника.

Таким чином, можна відзначити, що використовуються *4* базові схеми лічильників з безпосереднім переносом на основі *T*- або *JK*-тригерів. Дві з цих схем (рис.3.33, 3.37) реалізують підсумовувальний лічильник, а дві схеми на рис.3.41, 3.44 – віднімальний лічильник.

Таким чином, аналізуючи властивості двійкових лічильників з безпосереднім переносом, можна відзначити, що ці лічильники мають найменшу швидкодію по зрівнянню з лічильниками з іншими типами переносів між розрядами, але при цьому характеризуються найбільшою простотою апаратної реалізації.

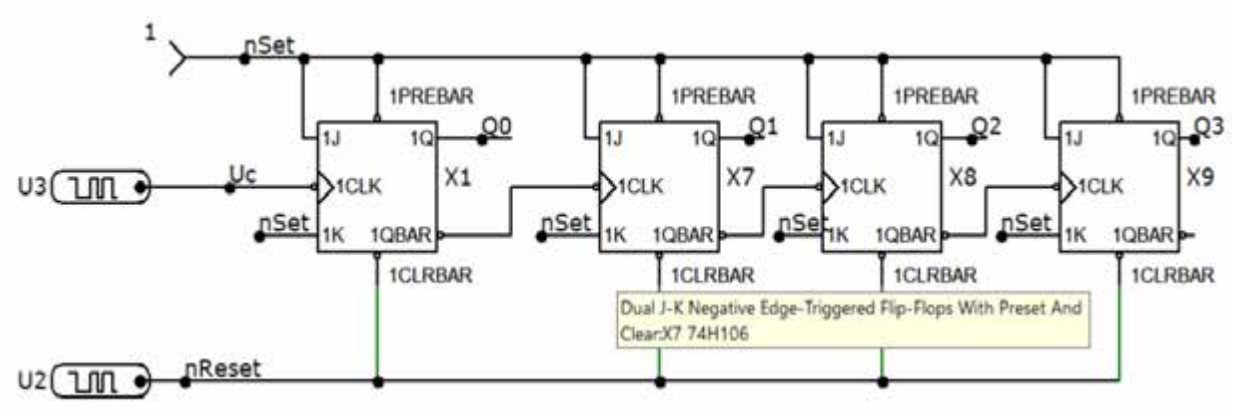


Рисунок 3.45 – Схема для моделювання віднімального асинхронного лічильника за модулем *16* з безпосереднім переносом на базі тригерів зі спрацьовуванням за заднім фронтом

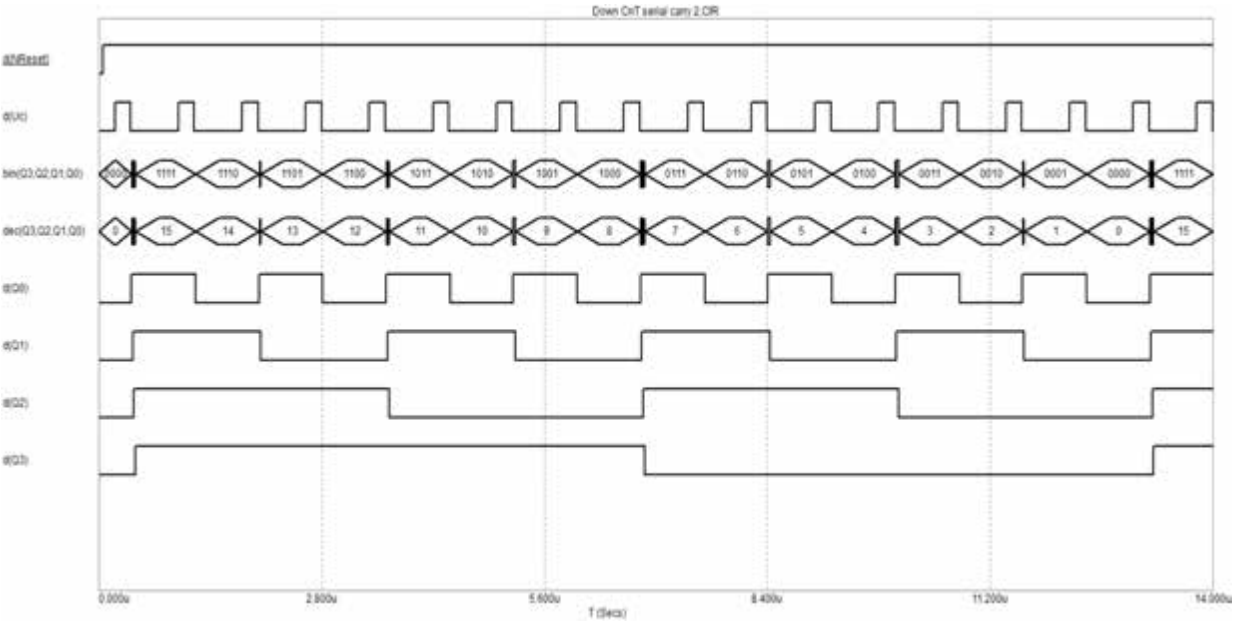


Рисунок 3.46 – Результати моделювання віднімального асинхронного лічильника за модулем *16* з безпосереднім переносом на базі тригерів зі спрацьовуванням за заднім фронтом

3.1.3.2 Синхронні двійкові лічильники з безпосереднім переносом

В попередніх підрозділах схеми синхронних лічильників були отримані зі схем асинхронних лічильників за рахунок перенесення лічильного сигналу з функцій збудження інформаційних сигналів до кіл синхронних входів базових тригерів. Виконаємо аналогічні дії для реалізації синхронного лічильника з безпосереднім переносом. Функціональна схема цього лічильника приведена на рис.3.47, а схема для моделювання – на рис.3.48.

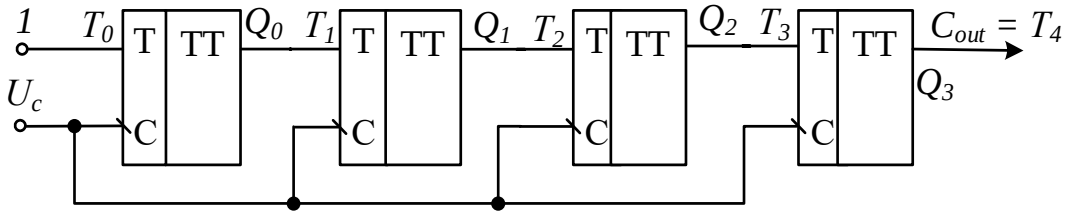


Рисунок 3.47 – Функціональна схема синхронного лічильника з безпосереднім переносом

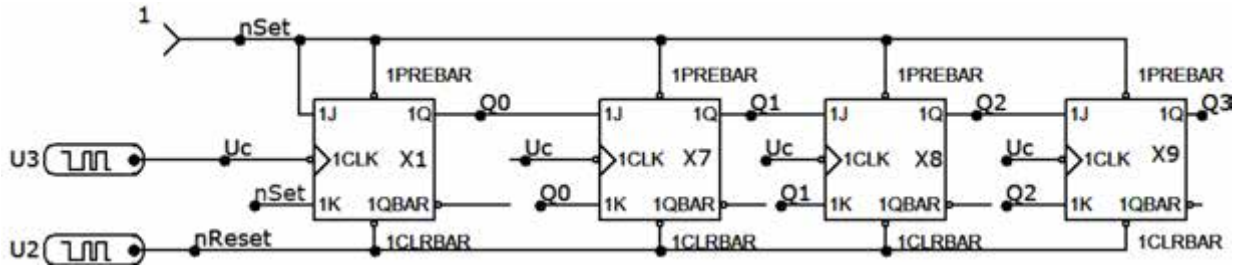


Рисунок 3.48 – Схема для моделювання синхронного лічильника з безпосереднім переносом

Результати моделювання синхронного лічильника з безпосереднім переносом приведені на рис.3.49.

Аналізуючи часову діаграму функціонування лічильника, можна зробити такі висновки:

- на виході лічильника відсутня природня послідовність зміни кодів станів ($0 \rightarrow 1 \rightarrow 2_{10} \rightarrow 7_{10} \rightarrow 8_{10} \rightarrow 9_{10} \rightarrow 10_{10} \rightarrow 15_{10} \rightarrow 0$);

- лічильник не виконує мікрооперацію підрахунку вхідних сигналів за модулем *16* відповідно до кількості тригерів (відповідно до часової діаграми цей лічильник має модуль ліку $M = 8$).

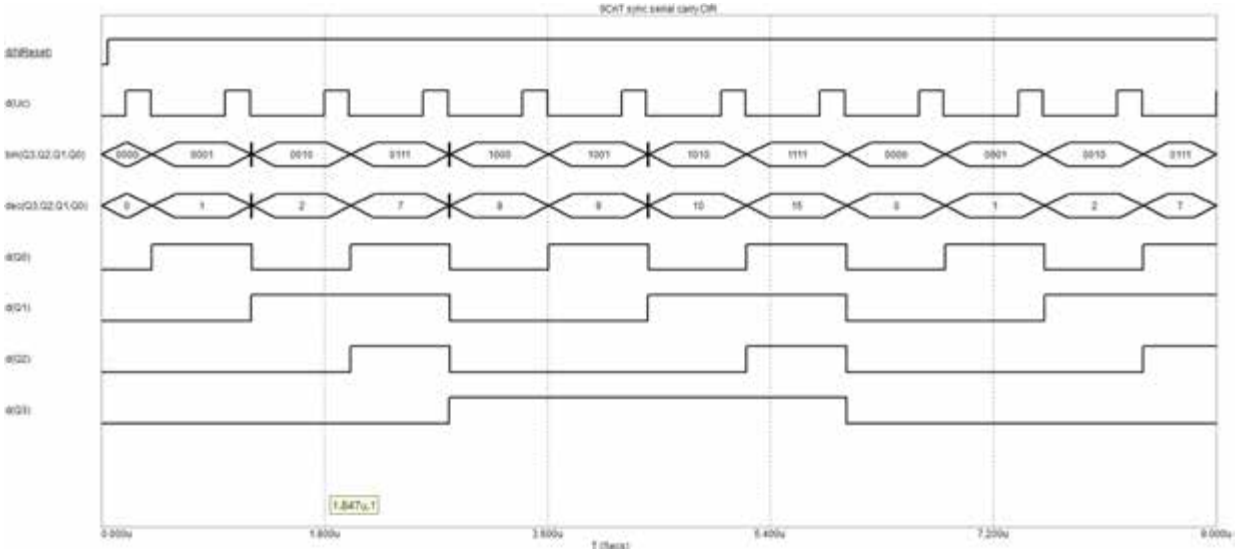


Рисунок 3.49 – Результати моделювання синхронного лічильника з безпосереднім переносом

Таким чином, такий лічильник може використовуватися тільки в режимі ділення частоти, але враховуючи перебільшену кількість тригерів, синхронні лічильники з безпосереднім переносом ***недоцільно*** використовувати під час проєктування комп'ютерних систем.

Контрольні завдання та запитання

1. Яким чином будуються асинхронні лічильники з безпосереднім переносом?
2. Яким чином будуються синхронні лічильники з безпосереднім переносом?
3. В чому полягає різниця між асинхронними та синхронними лічильниками?
4. В чому полягає різниця між послідовним, паралельним трактом розповсюдження переносу та безпосереднім переносом?
5. Як побудувати таблицю 3.5.
6. Поясніть принцип функціонування підсумовувального лічильника на прикладі таблиці 3.5.

7. Як виконати синтез схеми підсумовувального лічильника з безпосереднім переносом, використовуючи таблицю 3.5?

8. Як реалізувати схему двійкового підсумовувального лічильника з безпосереднім переносом з будь-яким модулем ліку?

9. Яким чином отримати схему лічильника, приведену на рис.3.33 за допомогою табл.3.5?

10. Поясніть сенс позначення dX.

11. Поясніть сенс позначення $d\overline{X}$.

12. Як побудувати таблицю 3.6.

13. Поясніть принцип функціонування підсумовувального лічильника на прикладі таблиці 3.6.

14. Як виконати синтез схеми підсумовувального лічильника з безпосереднім переносом, використовуючи таблицю 3.6?

15. Яким чином отримати схему лічильника, приведену на рис.3.33 за допомогою табл.3.6?

16. Як отримати вирази (3.14)?

17. Доведіть вирази (3.15).

18. Доведіть вирази (3.16).

19. Прокоментуйте призначення сигналів і функціональну схему, приведену на рис.3.34.

20. Як реалізувати *T*-тригер на базі *JK*-тригера?

21. Як визначити динамічні параметри асинхронних лічильників з безпосереднім переносом?

22. Як визначити час переключення асинхронних лічильників з безпосереднім переносом на базі *T*-тригерів зі спрацьовуванням за заднім фронтом сигналу синхронізації?

23. Як визначити час формування переносу зі старшого розряду асинхронних лічильників з безпосереднім переносом на базі *T*-тригерів зі спрацьовуванням за заднім фронтом сигналу синхронізації?

24. Прокоментуйте часові діаграми на рис.3.35.

25. Як по часовій діаграмі визначити модуль ліку лічильника?

26. Доведіть, що часова діаграма на рис.3.35 відповідає лічильнику з модулем ліку *16*.

27. Як по часовій діаграмі визначити напрям ліку лічильника?

28. Прокоментуйте часові діаграми на рис.3.36.

29. Визначити значення часу переключення асинхронного підсумовувального лічильника з безпосереднім переносом за модулем 32 на базі *T*-тригерів зі спрацьовуванням за заднім фронтом сигналу синхронізації, якщо $t_{тр} = 20нс$.

30. Визначити значення часу формування переносу зі старшого розряду асинхронного підсумовувального лічильника з безпосереднім переносом за модулем 32 на базі *T*-тригерів зі спрацьовуванням за заднім фронтом сигналу синхронізації, якщо $t_{тр} = 20нс$.

31. Яким чином отримати схему лічильника, приведену на рис.3.37 за допомогою табл.3.7?

32. Як побудувати таблицю 3.6.

33. Поясніть принцип функціонування підсумовувального лічильника на прикладі таблиці 3.7.

34. Як виконати синтез схеми підсумовувального лічильника з безпосереднім переносом, використовуючи таблицю 3.7?

35. Як отримати вирази (3.17)?

36. Доведіть вирази (3.17).

37. Прокоментуйте призначення сигналів і функціональну схему, приведену на рис.3.38.

38. Як визначити час переключення асинхронних лічильників з безпосереднім переносом на базі *T*-тригерів зі спрацьовуванням за переднім фронтом сигналу синхронізації?

39. Як визначити час формування переносу зі старшого розряду асинхронних лічильників з безпосереднім переносом на базі *T*-тригерів зі спрацьовуванням за переднім фронтом сигналу синхронізації?

40. Як по таблиці 3.7 визначити, що на вхід T_i необхідно підключати сигнал з інверсного виходу базового тригера Q_{i-1}?

41. Чому на входи *nK* базових тригерів підключається нульовий рівень сигналу?

42. Прокоментуйте часові діаграми на рис.3.39.

43. Поясніть часову діаграму на вході ***nReset***.

44. Доведіть, що часова діаграма на рис.3.39 відповідає лічильнику з модулем ліку *16*.

45. Як по часовій діаграмі визначити напрям ліку лічильника на рис.3.39?

46. Прокоментуйте часові діаграми на рис.3.40.

47. Які процеси, що відмічені символами «*1*», «*2а*», «*2б*» на рис.3.40, відбуваються в лічильнику схема якого приведена на рис.3.38?

48. Які процеси, що відмічені символами «*2а*», «*3а*», «*3б*» на рис.3.40, відбуваються в лічильнику, схема якого приведена на рис.3.38?

49. Які процеси, що відмічені символами «*3а*», «*4а*», «*4б*» на рис.3.40, відбуваються в лічильнику, схема якого приведена на рис.3.38?

50. Які процеси, що відмічені символами «*4а*», «*5*» на рис.3.40, відбуваються в лічильнику, схема якого приведена на рис.3.38?

51. Визначити значення часу формування переносу зі старшого розряду асинхронного підсумовувального лічильника з безпосереднім переносом за модулем 32 на базі *T*-тригерів зі спрацьовуванням за переднім фронтом сигналу синхронізації, якщо $t_{\text{тр}}^{01} = 20нс$.

52. Які динамічні процеси в лічильнику на рис.3.38 формують час переключення лічильника?

53. Прокоментуйте вираз (3.18).

54. Доведіть вираз (3.18).

55. Визначити значення часу переключення асинхронного підсумовувального лічильника з безпосереднім переносом за модулем 32 на

базі *Т*-тригерів зі спрацьовуванням за переднім фронтом сигналу синхронізації, якщо $t_{\text{тр}}^{10} = 20нс$; $t_{\text{фп}_i} = 10нс$.

56. Виконайте порівняння динамічних параметрів лічильників з безпосереднім переносом на базі тригерів зі спрацьовуванням за переднім та заднім фронтами сигналу синхронізації?

57. Яким чином не змінюючи схему лічильника перетворити підсумовувальний лічильник у віднімальний?

58. В яких випадках неможливе перетворення, задане в попередньому завданні?

59. Яким чином не змінюючи схему лічильника перетворити віднімальний лічильник в підсумовувальний?

60. В яких випадках неможливе перетворення, задане в попередньому завданні?

61. Як побудувати таблицю 3.8.

62. Поясніть принцип функціонування віднімального лічильника на прикладі таблиці 3.8.

63. Як виконати синтез схеми віднімального лічильника з безпосереднім переносом на базі тригерів зі спрацьовуванням за переднім фронтом сигналу синхронізації, використовуючи таблицю 3.8?

64. Як реалізувати схему двійкового віднімального лічильника з безпосереднім переносом з будь-яким модулем ліку?

65. Яким чином отримати схему лічильника, приведену на рис.3.41 за допомогою табл.3.8?

66. Прокоментуйте призначення сигналів і функціональну схему, приведену на рис.3.42.

67. Як визначити час переключення асинхронних віднімальних лічильників з безпосереднім переносом на базі *Т*-тригерів зі спрацьовуванням за переднім фронтом сигналу синхронізації?

68. Як визначити час формування переносу зі старшого розряду асинхронних віднімальних лічильників з безпосереднім переносом на базі *Т*- тригерів зі спрацьовуванням за переднім фронтом сигналу синхронізації?

69. Прокоментуйте часові діаграми на рис.3.43.

70. Як по часовій діаграмі визначити модуль ліку віднімального лічильника?

71. Доведіть, що часова діаграма на рис.3.43 відповідає віднімальному лічильнику з модулем ліку *16*.

72. Визначити значення часу переключення асинхронного віднімального лічильника з безпосереднім переносом за модулем 32 на базі *Т*-тригерів зі спрацьовуванням за переднім фронтом сигналу синхронізації, якщо $t_{тр} = 20нс$.

73. Визначити значення часу формування переносу зі старшого розряду асинхронного віднімального лічильника з безпосереднім переносом за модулем 32 на базі *Т*-тригерів зі спрацьовуванням за переднім фронтом сигналу синхронізації, якщо $t_{тр} = 20нс$.

74. Яким чином отримати схему лічильника, приведену на рис.3.42 за допомогою табл.3.8?

75. Яким чином отримана схема лічильника на рис.3.44?

76. Приведіть таблицю переходів віднімального лічильника з безпосереднім переносом тригерів зі спрацьовуванням за заднім фронтом сигналу синхронізації

77. Поясніть принцип функціонування віднімального лічильника на прикладі таблиці, розробленій в попередньому завданні.

78. Прокоментуйте призначення сигналів і функціональну схему, приведену на рис.3.45.

79. Прокоментуйте часові діаграми на рис.3.46.

80. Як визначити час переключення віднімальних асинхронних лічильників з безпосереднім переносом на базі *Т*-тригерів зі спрацьовуванням за заднім фронтом сигналу синхронізації?

81. Як визначити час формування переносу зі старшого розряду віднімальних асинхронних лічильників з безпосереднім переносом на базі *T*- тригерів зі спрацьовуванням за заднім фронтом сигналу синхронізації

82. Доведіть, що часова діаграма на рис.3.46 відповідає віднімальному лічильнику з модулем ліку *16*.

83. Визначити значення часу формування переносу зі старшого розряду асинхронного віднімальному лічильника з безпосереднім переносом за модулем 32 на базі *T*-тригерів зі спрацьовуванням за заднім фронтом сигналу синхронізації, якщо $t_{\text{тр}}^{01} = 20нс$.

84. Які динамічні процеси в лічильнику на рис.3.45 формують час переключення лічильника?

85. Визначити значення часу переключення асинхронного віднімальному лічильника з безпосереднім переносом за модулем 32 на базі *T*-тригерів зі спрацьовуванням за заднім фронтом сигналу синхронізації, якщо $t_{\text{тр}}^{10} = 20нс$; $t_{\text{фп}_i} = 10нс$.

86. Які недоліки характерні для асинхронних лічильників з безпосереднім переносом?

87. Які переваги характерні для асинхронних лічильників з безпосереднім переносом?

88. Виконати синтез асинхронного підсумовувального лічильника за модулем *16* з безпосереднім переносом на базі асинхронних *T*-тригерів зі спрацьовуванням за заднім фронтом сигналу синхронізації. Побудувати часові діаграми лічильника з використанням затримок логічних елементів.

89. Виконати синтез асинхронного підсумовувального лічильника за модулем *16* з безпосереднім переносом на базі асинхронних *T*-тригерів зі спрацьовуванням за переднім фронтом сигналу синхронізації. Побудувати часові діаграми лічильника з використанням затримок логічних елементів.

90. Виконати синтез асинхронного віднімального лічильника за модулем *16* з безпосереднім переносом на базі асинхронних *T*-тригерів зі

спрацьовуванням за заднім фронтом сигналу синхронізації. Побудувати часові діаграми лічильника з використанням затримок логічних елементів.

91. Виконати синтез асинхронного віднімального лічильника за модулем *16* з безпосереднім переносом на базі асинхронних *T*-тригерів зі спрацьовуванням за переднім фронтом сигналу синхронізації. Побудувати часові діаграми лічильника з використанням затримок логічних елементів.

92. Яким чином отримано схему лічильника на рис.3.47?

93. Прокоментуйте призначення сигналів і функціональну схему, приведену на рис.3.48.

94. Прокоментуйте часові діаграми на рис.3.49.

95. За яким модулем ліку працює синхронний лічильник, схема якого приведена на рис.3.48? Обґрунтуйте відповідь.

96. Чи можна використовувати синхронні лічильники з безпосереднім переносом в режимі ділення частоти? Обґрунтуйте відповідь.

97. Чи можна використовувати синхронні лічильники з безпосереднім переносом в режимі підрахунку вхідних сигналів? Обґрунтуйте відповідь.

98. Чи спостерігається природня послідовність зміни кодів вихідних станів синхронного лічильника з безпосереднім переносом? Обґрунтуйте відповідь.

99. Чому не доцільно використовувати синхронні лічильники з безпосереднім переносом?

100. Якими недоліками характеризується функціонування синхронних лічильників з безпосереднім переносом?

101. Виконайте порівняльний аналіз швидкодії синхронних і асинхронних лічильників з різними способами організації переносу.

102. Поясніть термін «безпосередній перенос» при організації лічильників.

103. Виконати синтез асинхронного підсумовувального лічильника за модулем *32* з безпосереднім переносом на базі асинхронних *T*-тригерів зі

спрацьовуванням за заднім фронтом сигналу синхронізації. Побудувати часові діаграми лічильника з використанням затримок логічних елементів.

104. Виконати синтез асинхронного підсумовувального лічильника за модулем *32* з безпосереднім переносом на базі асинхронних *T*-тригерів зі спрацьовуванням за переднім фронтом сигналу синхронізації. Побудувати часові діаграми лічильника з використанням затримок логічних елементів.

105. Чи можна в складі лічильника використовувати прозорі тригери? Обґрунтуйте відповідь.

3.2 Лічильники з довільним модулем ліку

Попередній розділ був присвячений розгляданню процедури синтезу **двійкових** лічильників з різними способами організації кіл переносу між розрядами. Нагадаємо, що до двійкових лічильників відносяться лічильники, у яких модуль ліку є кратним степеню двійки ($M = 2^n$). Далі розглянемо методи синтезу лічильників з іншими модулями ліку.

Для апаратної реалізації лічильників з довільним модулем ліку використовуються такі методи:

- синтез за допомогою таблиці переходів (класичний метод);
- реалізація лічильників за модулем M = $2^n + 1$;
- послідовне з’єднання лічильників;
- лічильники з асинхронним встановленням початкового стану;
- лічильники з синхронним встановленням початкового стану.

3.2.1 Синтез лічильників за допомогою таблиці переходів

Процедура синтезу лічильників за допомогою таблиці переходів детально розглянута в підрозділі 3.1.1.1 на прикладах синтезу двійкових лічильників, тобто загальний вигляд таблиці переходів і правила її заповнення аналогічні тому, як відзначено в п.3.1.1.1.

Розглянемо процедуру синтезу недвійкових лічильників за допомогою таблиці переходів на прикладі.

Приклад 3.4. Виконати синтез підсумовувального лічильника за модулем *9* на основі *JK*-тригерів.

Розв'язок

На першому кроці визначимо кількість базових тригерів лічильника. Відповідно до умови завдання $M = 9$, тобто згідно з виразом (3.1) $n_{mp} =] \log_2 9 [=] 3,17.. [= 4$. Таким чином, в складі лічильника будуть використовуватися чотири *JK*-тригери.

Результати виконання другого і третього кроків процедури синтезу лічильника (розробка таблиці переходів і визначення функцій збудження базових тригерів) представлені в табл.3.9.

Таблиця переходів заповнюється аналогічно табл.3.4.

Відповідно до завдання необхідно побудувати підсумовувальний лічильник за модулем *9*, тобто послідовність станів на виході лічильника відповідає зміні кодів $0 \to 1 \to 2_{10} \to \ldots \to 7_{10} \to 8_{10} \to 0 \to \ldots$ і так далі. Це означає, що стани з кодами $9 - 15$ в цьому лічильнику не існують, що відмічено у відповідних стовпчиках станів Q_i^{t+1} (стовпчики 7-10) символами «*» (рядки 9- 15, 25 - 31). Крім того, як було відзначено вище, після стану 8_{10} лічильник переключається в початковий стан *0*, що було позначено в рядку 24 сірим кольором.

Після опису функціонування лічильника (заповнення стовпчиків 7 - 10) виконується визначення функцій збудження базових *JK*-тригерів (заповнення стовпчиків 11 - 19). Визначення функцій збудження для входу *C* (колонка 11) описано в підрозділі 3.1.1.2, а детально визначення функцій збудження будь-яких тригерів розглянуто в [1].

Для неактивного значення сигналу на вході синхронізації *C* (позначається нулем) значення інформаційних сигналів на входах *J* і *K* можуть бути будь-якими і функцій збудження позначаються символом «*» (рядки 0 - 15 табл.3.9).

Для активного фронту сигналу на вході синхронізації функції збудження *JK*-тригерів приведені в табл.3.10.

Таблиця 3.9 – Таблиця переходів лічильника (приклад 3.4)

Номери наборів	U_c	Попередній стан				Наступний стан				ФЗ БТ								
		Q_3^t	Q_2^t	Q_1^t	Q_0^t	Q_3^{t+1}	Q_2^{t+1}	Q_1^{t+1}	Q_0^{t+1}	C	J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*	*	*	*	*	*	*	*
1	*0*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*1*	*0*	*	*	*	*	*	*	*	*
2	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*1*	*0*	*0*	*	*	*	*	*	*	*	*
3	*0*	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*	*	*	*	*	*	*	*
4	*0*	*0*	*1*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*	*	*	*	*	*	*	*
5	*0*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*	*	*	*	*	*	*	*
6	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*	*	*	*	*	*	*	*
7	*0*	*0*	*1*	*1*	*1*	*0*	*1*	*1*	*1*	*0*	*	*	*	*	*	*	*	*
8	*0*	*1*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*0*	*	*	*	*	*	*	*	*
9	*0*	*1*	*0*	*0*	*1*	*	*	*	*	*0*	*	*	*	*	*	*	*	*
10	*0*	*1*	*0*	*1*	*0*	*	*	*	*	*0*	*	*	*	*	*	*	*	*
11	*0*	*1*	*0*	*1*	*1*	*	*	*	*	*0*	*	*	*	*	*	*	*	*
12	*0*	*1*	*1*	*0*	*0*	*	*	*	*	*0*	*	*	*	*	*	*	*	*
13	*0*	*1*	*1*	*0*	*1*	*	*	*	*	*0*	*	*	*	*	*	*	*	*
14	*0*	*1*	*1*	*1*	*0*	*	*	*	*	*0*	*	*	*	*	*	*	*	*
15	*0*	*1*	*1*	*1*	*1*	*	*	*	*	*0*	*	*	*	*	*	*	*	*
16	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*1*	*0*	*	*0*	*	*0*	*	*1*	*
17	*1*	*0*	*0*	*0*	*1*	*0*	*0*	*1*	*0*	*1*	*0*	*	*0*	*	*1*	*	*	*1*
18	*1*	*0*	*0*	*1*	*0*	*0*	*0*	*1*	*1*	*1*	*0*	*	*0*	*	*	*0*	*1*	*
19	*1*	*0*	*0*	*1*	*1*	*0*	*1*	*0*	*0*	*1*	*0*	*	*1*	*	*	*1*	*	*1*
20	*1*	*0*	*1*	*0*	*0*	*0*	*1*	*0*	*1*	*1*	*0*	*	*	*0*	*0*	*	*1*	*
21	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*1*	*0*	*1*	*0*	*	*	*0*	*1*	*	*	*1*
22	*1*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*1*	*1*	*0*	*	*	*0*	*	*0*	*1*	*
23	*1*	*0*	*1*	*1*	*1*	*1*	*0*	*0*	*0*	*1*	*1*	*	*	*1*	*	*1*	*	*1*
24	*1*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*	*1*	*0*	*	*0*	*	*0*	*
25	*1*	*1*	*0*	*0*	*1*	*	*	*	*	*	*	*	*	*	*	*	*	*
26	*1*	*1*	*0*	*1*	*0*	*	*	*	*	*1*	*	*	*	*	*	*	*	*
27	*1*	*1*	*0*	*1*	*1*	*	*	*	*	*1*	*	*	*	*	*	*	*	*
28	*1*	*1*	*1*	*0*	*0*	*	*	*	*	*1*	*	*	*	*	*	*	*	*
29	*1*	*1*	*1*	*0*	*1*	*	*	*	*	*1*	*	*	*	*	*	*	*	*
30	*1*	*1*	*1*	*1*	*0*	*	*	*	*	*1*	*	*	*	*	*	*	*	*
31	*1*	*1*	*1*	*1*	*1*	*	*	*	*	*1*	*	*	*	*	*	*	*	*

Наприклад, для виконання мікрооперації збереження нульового стану ($Q^t = 0 \rightarrow Q^{t+1} = 0$) тригера Q_3 (рядок 19, стовпчики 3, 7) за рахунок інформаційних сигналів J і K необхідно, з одного боку, підключити на входи комбінацію збереження $J_3 = K_3 = 0$, а з іншого боку – використати комбінацію ***Reset***, за якої $J_3 = 0$; $K_3 = 1$, тобто в загальному випадку $J_3 = 0$, а сигнал на вході K_3 може бути будь-яким, що позначено символом «*» (рядок 19, стовпчики 12, 13 в таблиці 3.9 або рядок 1 в таблиці 3.10).

Таблиця 3.10 – Функції збудження *JK*-тригера для активного фронту сигналу синхронізації

№ з/п	Попередній стан Q^t	Наступний стан Q^{t+1}	J	K	Мікрооперація
1	*0*	*0*	*0*	*	Збереження нуля
2	*0*	*1*	*1*	*	Переключення в 1
3	*1*	*0*	*	*1*	Переключення в 0
4	*1*	*1*	*	*0*	Збереження одиниці

Для виконання мікрооперації переключення з нульового стану в одиничний ($Q^t = 0 \rightarrow Q^{t+1} = 1$), наприклад, тригера Q_0 (рядок 16, стовпчики 6, 10) необхідно, з одного боку, підключити на входи комбінацію ***Set*** $J_0 = 1$; $K_0 = 0$, а з іншого боку – використати комбінацію переключення в протилежний стан, за якої $J_0 = 1$; $K_0 = 1$, тобто в загальному випадку $J_0 = 1$, а сигнал на вході K_0 може бути будь-яким, що позначено символом «*» (рядок 16, стовпчики 18, 19 в таблиці 3.9 або рядок 2 в таблиці 3.10).

Аналогічним чином визначаються функції збудження для рядків 3, 4 таблиці 3.10.

Після заповнення таблиці 3.9 виконаємо мінімізацію функцій збудження базових тригерів лічильника за допомогою карт Карно.

З таблиці 3.9 можна побачити, що в стовпчиках 13, 19 для входів K_3 і K_0 містяться тільки одиниці і символи «*», тобто без використання карт Карно можна записати: $K_3 = K_0 = 1$.

Карти Карно для мінімізації функцій збудження інформаційних входів *JK*-тригерів приведені відповідно на рис.3.50 - 3.55.

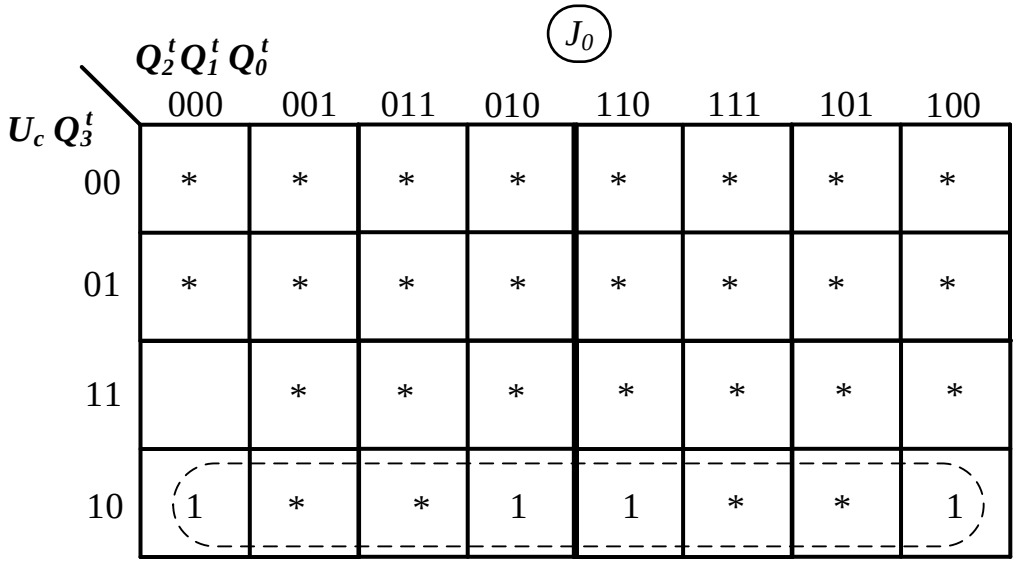


Рисунок 3.50 – Карта Карно для визначення ФЗ входу J_0 тригера Q_1 (приклад 3.4)

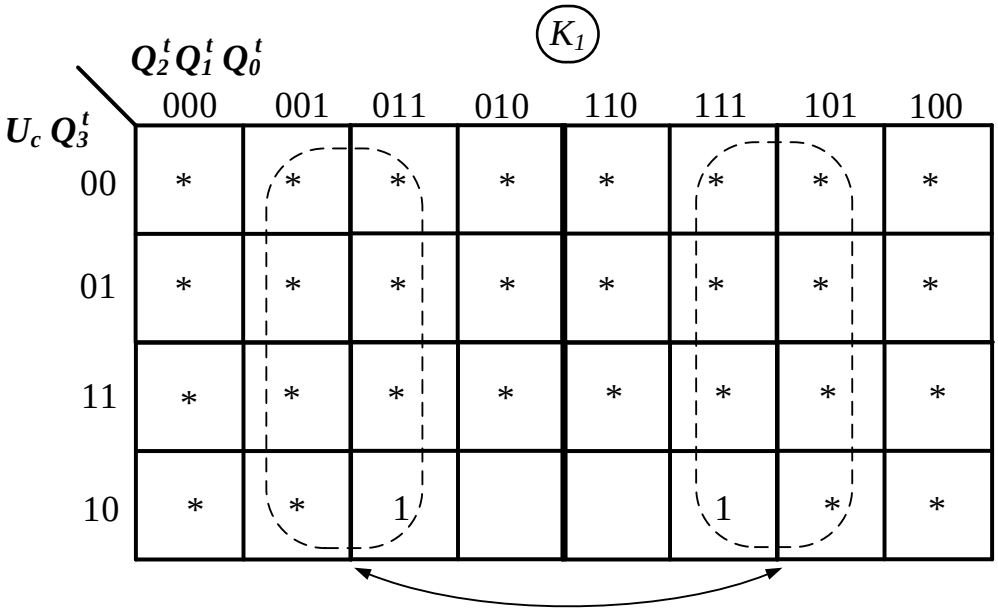


Рисунок 3.51 – Карта Карно для визначення ФЗ входу K_1 тригера Q_1 (приклад 3.4)

В результаті мінімізації отримуємо логічні вирази для функцій збудження базових тригерів:

$$C_i = U_c\ (i = 0, \ldots, 3);\quad K_3 = K_0 = 1;\quad J_0 = \overline{Q_3}$$
$$J_1 = K_1 = Q_0;\quad J_2 = K_2 = Q_0 \cdot Q_1;\quad J_3 = Q_0 \cdot Q_1 \cdot Q_2; \qquad (3.19)$$

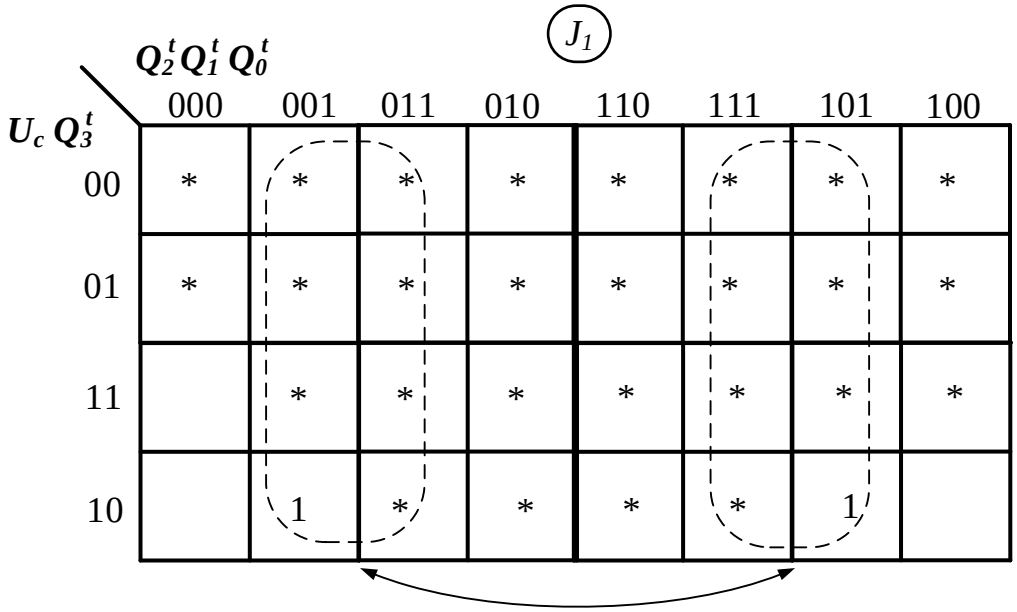


Рисунок 3.52 – Карта Карно для визначення ФЗ входу J_1 тригера Q_1 (приклад 3.4)

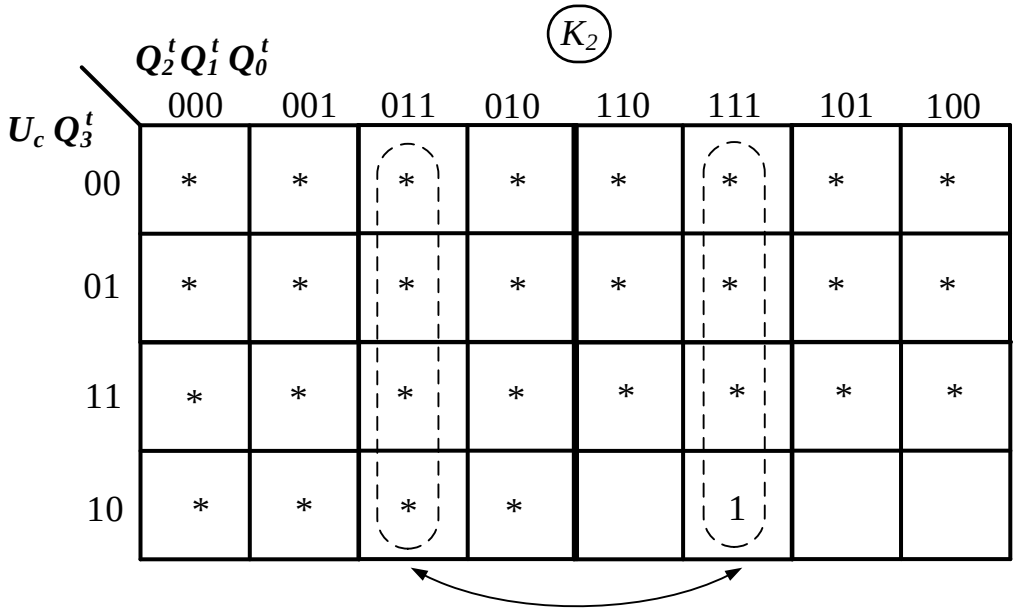


Рисунок 3.53 – Карта Карно для визначення ФЗ входу K_2 тригера Q_2 (приклад 3.4)

Функціональна схема лічильника відповідно до (3.19) приведена на рис.3.56.

Схема для моделювання підсумовувального лічильника за модулем *9* приведена на рис.3.57.

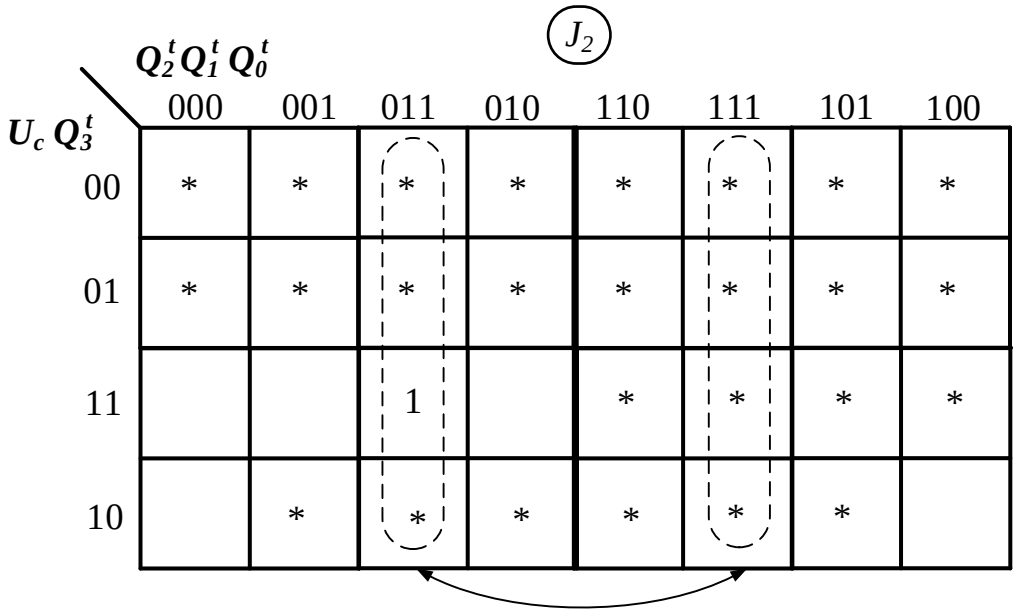


Рисунок 3.54 – Карта Карно для визначення ФЗ входу J_2 тригера Q_2 (приклад 3.4)

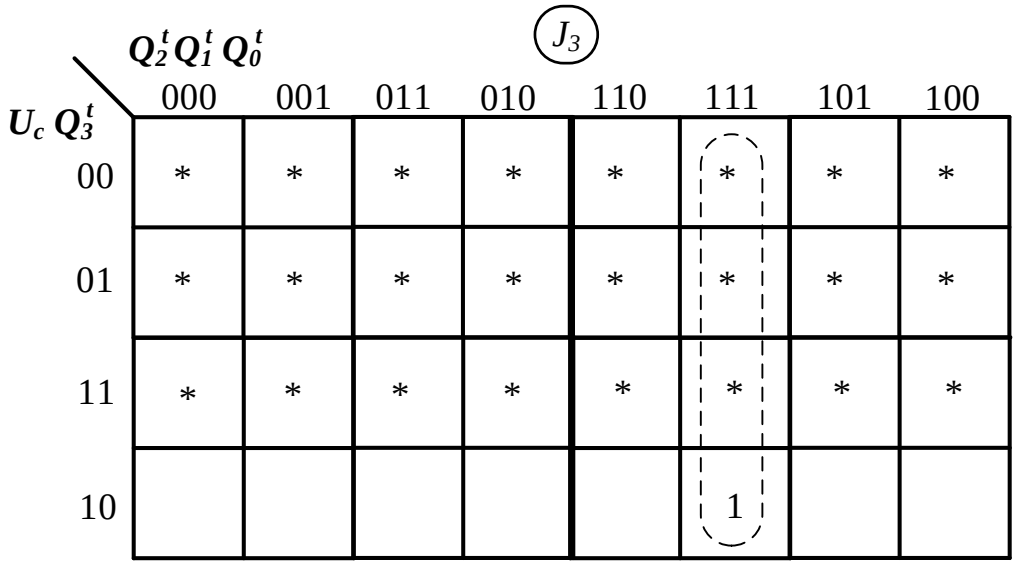


Рисунок 3.55 – Карта Карно для визначення ФЗ входу J_3 тригера Q_3 (приклад 3.4)

Результати моделювання підсумовувального лічильника за модулем *9* приведені на рис.3.58 та свідчать про коректність проведеної процедури синтезу.

Результати моделювання для визначення динамічних параметрів приведені на рис.3.59.

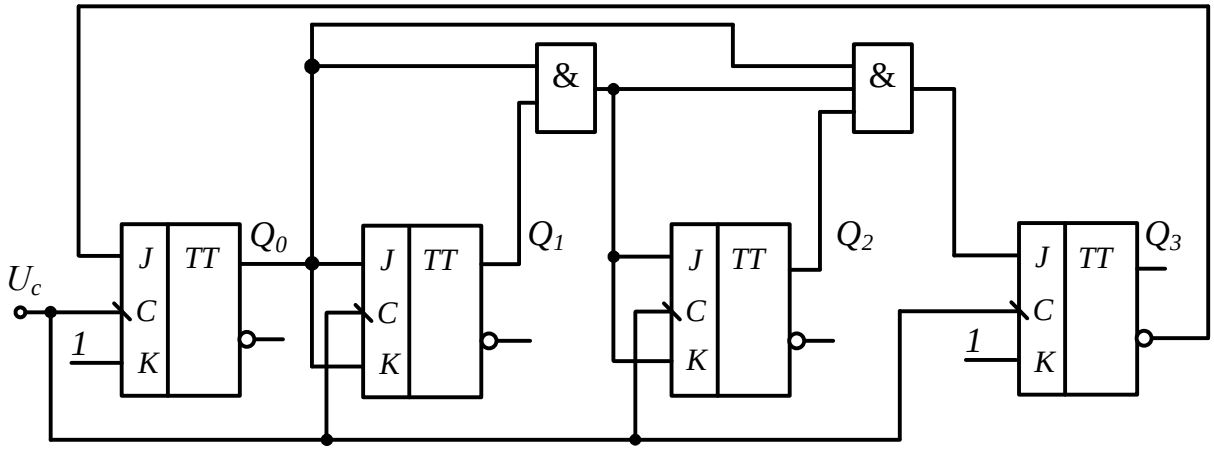


Рисунок 3.56 – Функціональна схема підсумовувального лічильника за модулем *9*

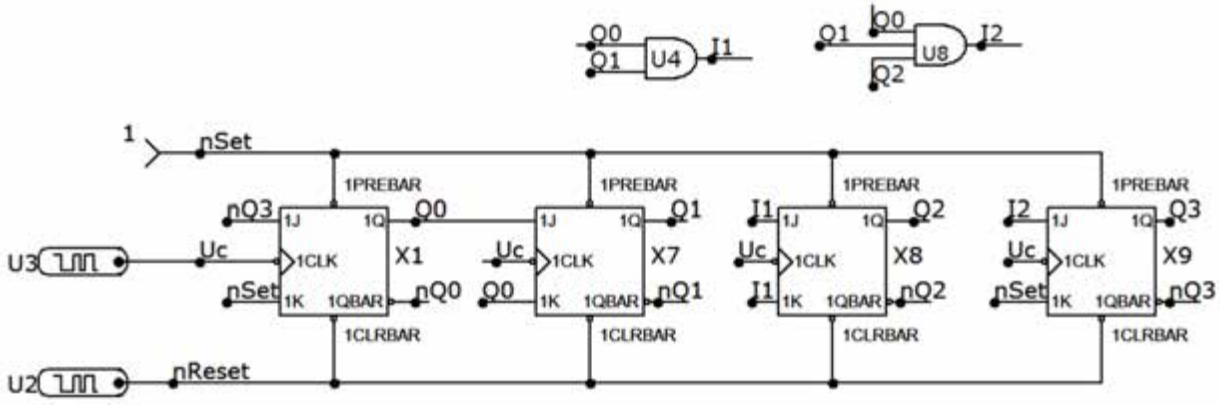


Рисунок 3.57 – Схема для моделювання підсумовувального лічильника за модулем *9* (приклад 3.4)

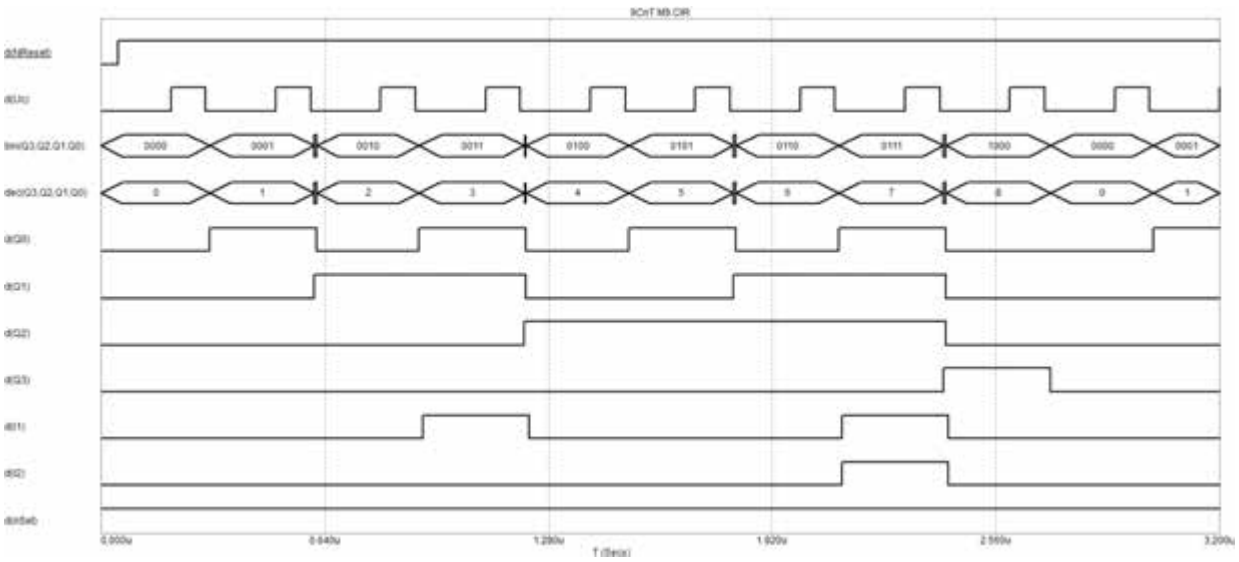


Рисунок 3.58 – Результати моделювання підсумовувального лічильника за модулем *9* (приклад 3.4)

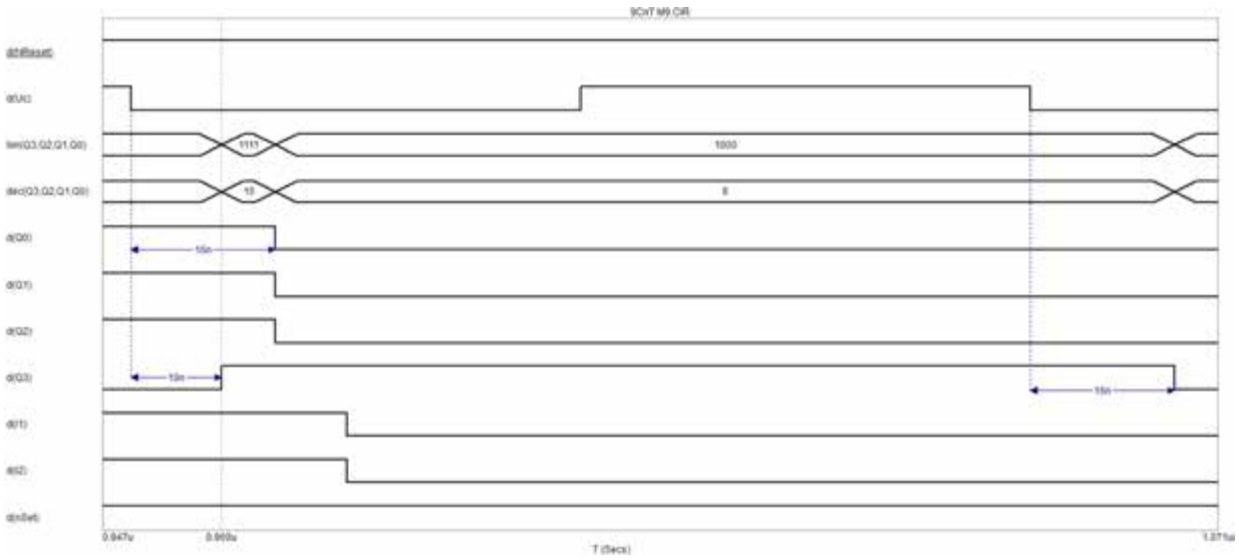


Рисунок 3.59 – Визначення динамічних параметрів підсумовувального лічильника за модулем *9* (приклад 3.4)

На результатах моделювання (рис.3.59) в більш детальному масштабі показано переключення лічильника зі стану 7_{10} в стан 8_{10} ($7_{10} \rightarrow 8_{10}$) та $8_{10} \rightarrow 0_{10}$. При переключенні $7_{10} \rightarrow 8_{10}$ перенос пробігає через всі розряди лічильника, але в зв'язку з тим, що реалізовано синхронний лічильник, то всі тригери починають переключатися одночасно. В результаті час переключення лічильника $t_{п} = t_{тр}$ (на рис.3.59 $t_{тр} = 16нс$ при переключенні тригерів в нульовий стан).

При переключенні $8_{10} \rightarrow 0_{10}$ формується перенос зі старшого розряду, який є виходом тригера Q_3. Таким чином, час формування переносу визначається за виразом $t_{фп} = t_{тр}$, що теж складає $16нс$.

На цьому виконання прикладу 3.4 завершено, але розглянемо ще реалізацію заданого лічильника на базі *JK*-тригерів зі спрацьовуванням за переднім фронтом сигналу синхронізації *SN74109*, які вже використовувалися в п.3.1.3.1. Синтез лічильника на базі цього тригера нічим не відрізняється від процедури синтезу, розглянутої вище. Але в зв'язку з наявністю у цього тригера інверсного входу *K* функції збудження для цього входу отримаємо у вигляді:

$$nK_3 = nK_0 = 0; \qquad nK_1 = \overline{Q_0}; \quad nK_2 = \overline{Q_0 \cdot Q_1}. \tag{3.20}$$

Інші функції збудження для входів *C* та *J* тригерів лічильника відповідають виразам (3.19).

Схема для моделювання підсумовувального лічильника за модулем *9* на базі тригерів зі спрацьовуванням за переднім фронтом приведена на рис.3.60, а результати моделювання цього лічильника – на рис.3.61.

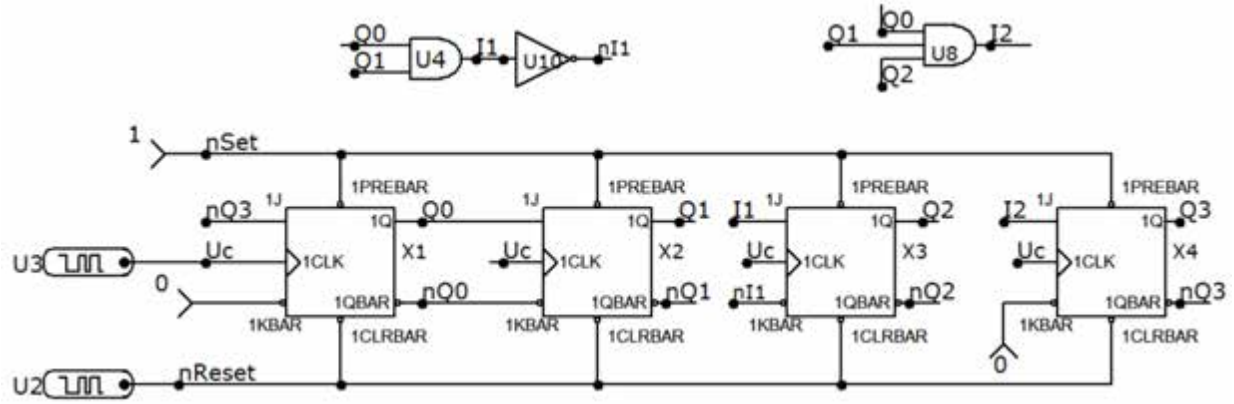


Рисунок 3.60 – Схема для моделювання підсумовувального лічильника за модулем *9* на базі *SN74109*

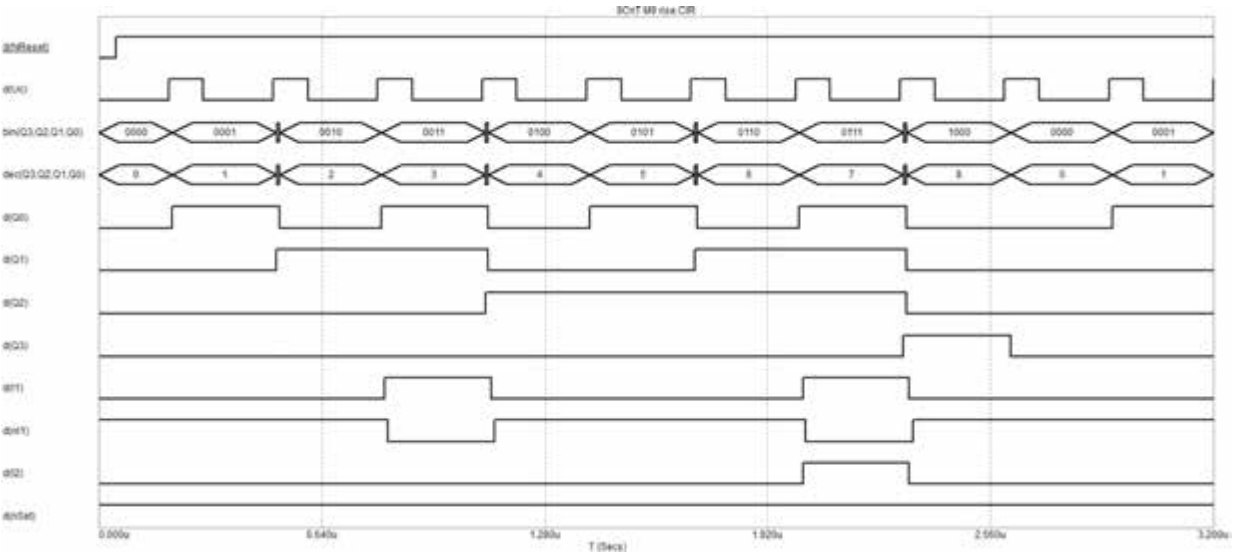


Рисунок 3.61 – Результати моделювання підсумовувального лічильника за модулем *9* на базі *SN74109*

На завершення треба відмітити, що при використанні синтезу лічильників за допомогою таблиці переходів виконується реалізація лічильників з паралельним трактом розповсюдження переносу. При цьому,

якщо базовими тригерами є *T*-тригер, то лічильник є асинхронним, а для інших типів тригерів – синхронним.

Перевагою цього методу є його універсальність, але використання цього методу обмежується кількістю змінних таблиці переходів в разі виконання ручної мінімізації за допомогою карт Карно. Як правило, ручна мінімізація на картах Карно обмежена наявністю шести змінних, тобто сигналами U_c, Q_4 - Q_0, що обмежує лічильник модулем ліку $M = 32$. Враховуючи, що в синхронних лічильниках сигнал U_c не входить до складу функцій збудження таблицю переходів можна представляти тільки для активного значення сигналу U_c. В цьому випадку синтез може бути проведений для реалізації лічильника з модулем ліку до *64*. Якщо необхідно реалізовувати недвійковий лічильник з великим модулем ліку, то необхідно використовувати комп'ютерні методи мінімізації або застосовувати інші методи синтезу.

3.2.2 Підсумовувальні лічильники за модулем $M = 2^k + 1$

Лічильники з вищезазначеним модулем ліку мають схоже структурне побудування.

Відповідно до (3.1) кількість базових тригерів ($n_{тр}$, далі n) для лічильників з модулем ліку 2^k+1 складає $n = k+1$, тобто для $k = 3$ ($M = 9$) кількість тригерів дорівнює $n = 4$.

Згідно з принципом функціонування лічильників підсумовувальний лічильник з модулем ліку 2^k+1 для будь-якого додатного значення k має в загальному випадку послідовність зміни кодів станів (далі – «станів»), яка показана в табл.3.11.

Таким чином, наприклад, для підсумовувального лічильника з $M = 9$ використовується *4* тригери Q_3, Q_2, Q_1, Q_0 з такою послідовністю зміни станів: $0 \rightarrow 1 \rightarrow 2_{10} \rightarrow 3_{10} \rightarrow 4_{10} \rightarrow 5_{10} \rightarrow 6_{10} \rightarrow 7_{10} \rightarrow 8_{10} \rightarrow 0 \rightarrow \ldots$ або в двійковому коді:

0000→0001→0010→0011→0100→0101→0110→0111→1000→0000→… ,

тобто будь-який підсумовувальний лічильник з модулем ліку $M = 2^k+1$ починає переключатися з початкового стану $00…00_2$ до кінцевого стану $100..00_2$ і далі знову переключається до стану $00…00_2$.

Таблиця 3.11 – Послідовність станів підсумовувального лічильника за модулем 2^k+1

Десятковий код стану	Q_{n-1}	Q_{n-2}	…	Q_i	…	Q_1	Q_0
0	*0*	*0*	*0*	*0*	*0*	*0*	*0*
1	*0*	*0*	*0*	*0*	*0*	*0*	*1*
2	*0*	*0*	*0*	*0*	*0*	*1*	*0*
…	…		…	…	…		…
i	…		…	…	…		…
…	…		…	…	…		…
2^k-1	*0*	*1*	*1*	*1*	*1*	*1*	*1*
2^k	*1*	*0*	*0*	*0*	*0*	*0*	*0*

Структурно-функціональна схема підсумовувальних лічильників за модулем 2^k+1 приведена на рис.3.62.

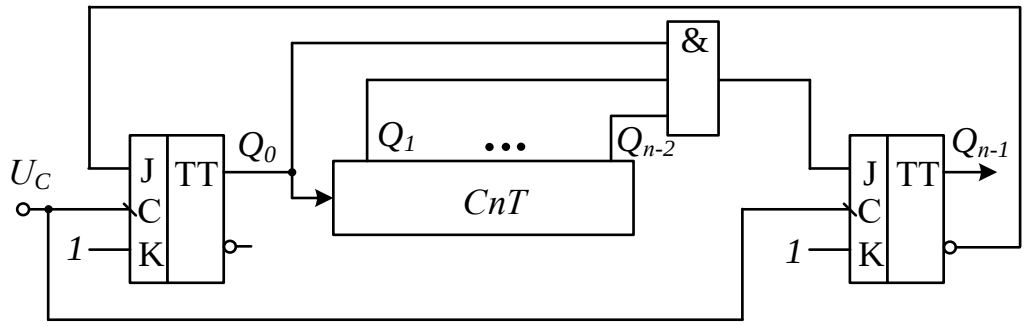


Рисунок 3.62 – Структурно-функціональна схема підсумовувального лічильника за модулем $2^k + 1$

Аналізуючи порядок переключення станів лічильника, можна зробити висновок, що послідовність зміни станів всіх базових тригерів, окрім найстаршого, в межах одного циклу роботи від нульового стану до стану $M - 1$, відповідає принципам функціонування звичайного двійкового

лічильника з будь-яким типом переносу між розрядами. При цьому відповідно до схеми на рис.3.62 в цих розрядах наймолодший тригер обов'язково є *JK*-тригером, а решта тригерів в цій частині лічильника можуть бути будь-якими, як правило *JK*-, *TC*- або *T*- тригером. Звичайно, що найстарший тригер всього лічильника теж є обов'язково *JK*-тригером.

Розглянемо принцип функціонування цього лічильника. Починаючи зі стану $00\ldots00_2$ до стану $01\ldots11_2$ (тригери Q_{n-2}, …, Q_0) лічильник працює, як звичайний двійковий. Коли всі тригери, окрім найстаршого, переключаються в одиничний стан, то спрацьовує логічний елемент *I*, на його виході формується одиниця, яка надходить на вхід *J* найстаршого тригера Q_{n-1}. За активним фронтом сигналу U_c тригер Q_{n-1} переключається в одиничний стан, а решта тригерів – в нульовий, тобто новий стан лічильника відповідає коду $10\ldots00_2$. В результаті на інверсному виході найстаршого тригера з'являється сигнал низького рівня (логічний 0), тобто на входах тригера Q_0 формується комбінація ***Reset*** ($J_0 = 0$; $K_0 = 1$). Крім того, на виході елемента *I* також з'являється нуль, тобто на входах тригера Q_{n-1} також формується комбінація ***Reset*** ($J_{n-1} = 0$; $K_{n-1} = 1$). Це означає, що за новим активним фронтом сигналу U_c тригер Q_{n-1} переключається в нульовий стан, а тригер Q_0 залишається ц нульовому стані, в результаті чого лічильник переключається зі стану $10\ldots00_2$ в стан $00\ldots00_2$, що відповідає таблиці переходів лічильника. Далі цикл роботи лічильника повторюється.

Нагадаємо, що при виконанні прикладу 3.4 було виконано синтез підсумовувального лічильника з модулем ліку $M = 9$ (рис.3.56, 3.57), схема якого повністю узгоджується зі структурно-функціональною схемою на рис.3.62 (в схемі лічильника на рис.3.56 тригери $Q_2 - Q_0$ реалізують синхронний двійковий лічильник за модулем *8* з паралельним трактом розповсюдження переносу).

На рис.3.63 приведена схема підсумовувального лічильника з модулем ліку $M = 9$, у якого в якості двійкового лічильник використовується

лічильник за модулем *8* з безпосереднім переносом, а на рис.3.64 – результати моделювання цього лічильника зі значеннями динамічних параметрів.

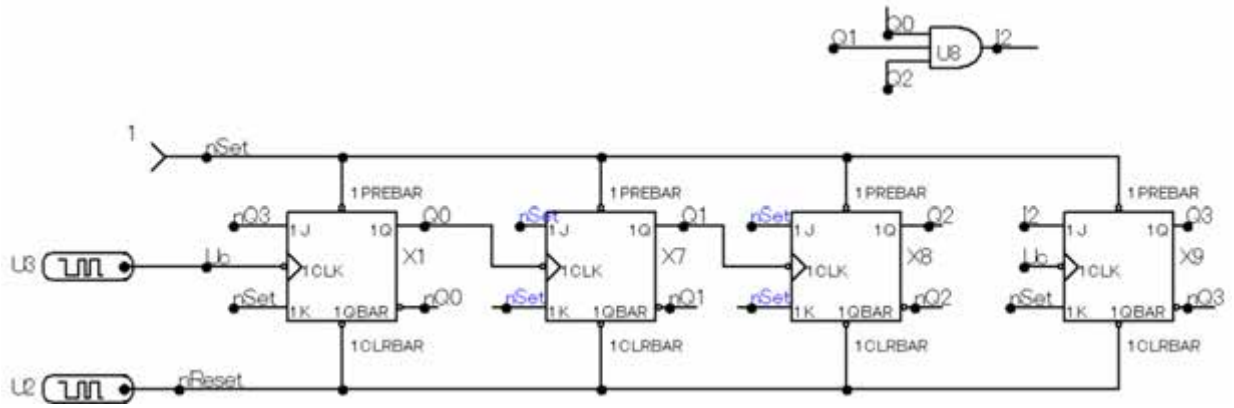


Рисунок 3.63 – Схема для моделювання підсумовувального лічильника за модулем *9* з двійковим лічильником з безпосереднім переносом

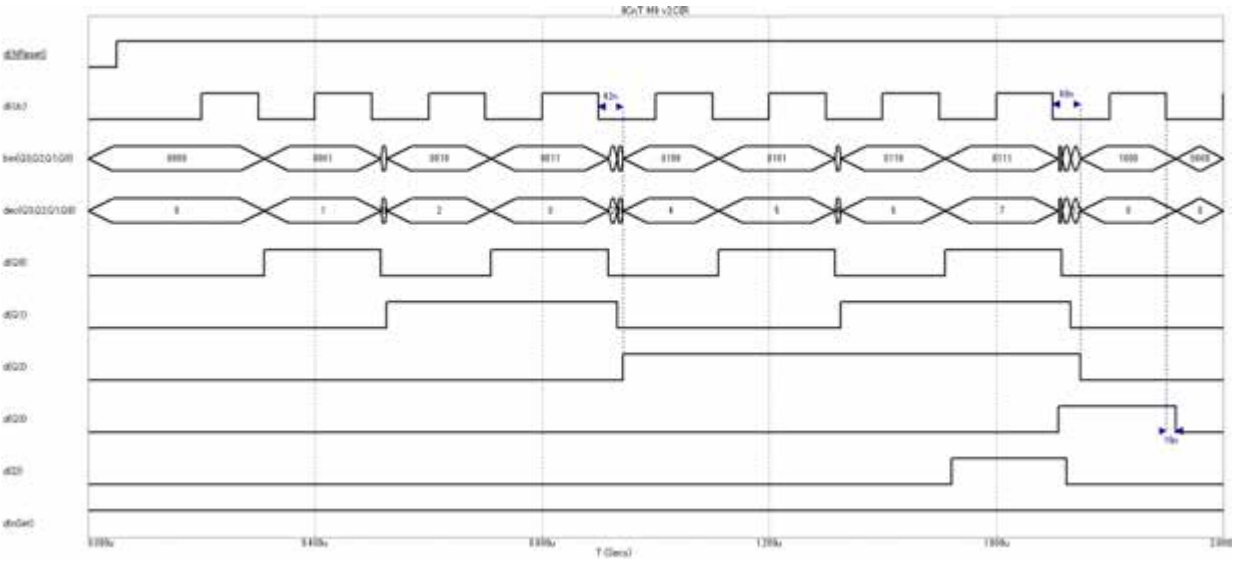


Рисунок 3.64 – Результати моделювання підсумовувального лічильника за модулем *9* з двійковим лічильником з безпосереднім переносом

3.2.3 Послідовне з’єднання лічильників

Послідовне з’єднання двох лічильників здійснюється за рахунок підключення виходу переносу зі старшого розряду одного лічильника до входу U_c другого лічильника. Звичайно, що таким чином можна з’єднувати будь-яку кількість лічильників, тому далі буде достатньо розглядати послідовне з’єднання двох лічильників.

При послідовному з’єднанні двох лічильників з модулями ліку M_1 і M_2 загальний модуль ліку M_0 підсумкового лічильника дорівнює добутку модулів з’єднаних лічильників $M_0 = M_1 \cdot M_2$. Аналогічно, при послідовному з’єднанні n лічильників загальний модуль ліку визначається за виразом $M_0 = M_1 \cdot M_2 \cdot \ldots \cdot M_n$.

Лічильник, до виходу переносу зі старшого розряду якого підключається інший лічильник, далі будемо називати молодшим лічильником, і, навпаки, лічильник, що підключається до виходу переносу зі старшого розряду попереднього лічильника, будемо називати старшим лічильником.

Очевидно, що існує два способи послідовного з’єднання двох лічильників M_1 і M_2, де, наприклад, лічильник M_1 може бути як молодшим ($M_{мол}$), так і, навпаки, старшим ($M_{ст}$). Далі розглянемо різницю між використанням цих способів.

Спосіб послідовного з’єднання лічильників залежить від режиму використання підсумкового лічильника. Нагадаємо, що лічильники можуть використовуватися в режимі підрахунку вхідних сигналів або в режимі ділення частоти.

Для використання підсумкового лічильника в режимі підрахунку вхідних сигналів всі лічильники окрім найстаршого повинні бути тільки двійковими, а найстарший лічильник може мати будь-який модуль ліку. Тільки в цьому разі на виході лічильника буде спостерігатися природня послідовність зміни кодів станів. В протилежному випадку природня послідовність зміни кодів станів буде відсутня, тому до виходу лічильника ще необхідно буде під’єднувати комбінаційний перетворювач кодів на виході лічильника в природню послідовність кодів станів.

Розглянемо зазначені вище вимоги на прикладі. Нехай, маються в наявності два лічильника з модулями ліку $M_1 = 3$ та $M_2 = 4$, які будемо позначати відповідно *Cnt3* і *Cnt4*. Очевидно, що за виразом (3.1) обидва лічильники мають у своєму складі по *2* тригери. Нагадаємо, що зміна кодів

станів для лічильника *Cnt3* відповідає двійковій послідовності *00→01→10→00→…*, а для лічильника *Cnt4*: *00→01→10→11→00→…* .

Послідовне з'єднання лічильників *Cnt3* і *Cnt4* утворює підсумковий лічильник з модулем $M_0 = M_1 \cdot M_2 = 12$. Два способи послідовного з'єднання лічильників *Cnt3* і *Cnt4* приведені на рис.3.65.

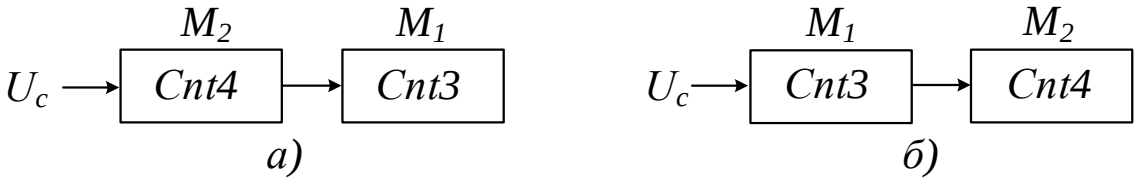


Рисунок 3.65 – Способи послідовного з'єднання лічильників

В табл.3.12 приведені послідовності зміни кодів станів для обох способів утворення підсумкового лічильника (в стовпчиках «$Стан_{10}$» вказані десяткові еквіваленти двійкового стану лічильника).

Таблиця 3.12 – Послідовність станів підсумкового лічильника на базі каскадного з'єднання базових лічильників

Лічильник за схемою на рис.3.65,а		
Cnt3	*Cnt4*	$Стан_{10}$
00	00	0
00	01	1
00	10	2
00	11	3
01	0	4
01	01	5
01	10	6
01	11	7
10	00	8
10	01	9
10	10	10
10	11	11
00	00	0

Лічильник за схемою на рис.3.65,б		
Cnt4	*Cnt3*	$Стан_{10}$
00	00	0
00	01	1
00	10	2
01	00	4
01	01	5
01	10	6
10	00	8
10	01	9
10	10	10
11	00	12
11	01	13
11	10	14
00	00	00

Для схеми з'єднання, що приведена на рис.3.65,а, молодшим лічильником є двійковий лічильник *Cnt4*. Після стану 11_2 лічильник *Cnt4*

формує сигнал переносу в старший лічильник (*Cnt3*), переключаючи його в наступний стан, а сам *Cnt4* переключається в нульовий стан (в табл.3.12 ці переключення показані стрілочками). При цьому з цієї таблиці видно, що послідовність зміни станів підсумкового лічильника відповідає природній нумерації чисел, тобто цей спосіб з'єднання може бути використаний для підрахунку вхідних сигналів.

Для схеми з'єднання, що приведена на рис.3.65,б, молодшим лічильником є лічильник *Cnt3*, який після стану 10_2 формує сигнал переносу в старший лічильник (*Cnt4*), переключаючи його в наступний стан, а сам *Cnt3* при цьому переключається в нульовий стан (в табл.3.12 ці переключення теж показані стрілочками). При цьому з цієї таблиці видно, що послідовність зміни станів підсумкового лічильника не відповідає природній нумерації чисел, тобто цей спосіб з'єднання може бути використаний тільки в режимі ділення частоти.

Схема для моделювання послідовного з'єднання лічильників *Cnt4* і *Cnt3*, де лічильник *Cnt4* є молодшим, приведена на рис.3.66.

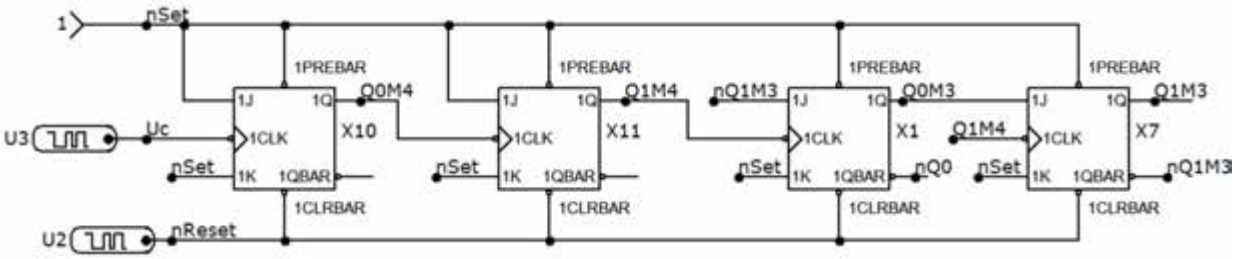


Рисунок 3.66 – Схема для моделювання послідовного з'єднання лічильників *Cnt4* і *Cnt3*

Виходи лічильників *Cnt4* (*Cnt3*) на рис.3.66 відповідно позначені *Q1M4*, *Q0M4* (*Q1M3*, *Q0M3*). Результати моделювання такого з'єднання лічильників приведені на рис.3.67. На часовій діаграмі видно, що послідовність кодів станів відповідає природній нумерації чисел $0 \rightarrow 1 \rightarrow 2_{10} \rightarrow 3_{10} \rightarrow \ldots \rightarrow 11_{10} \rightarrow 0 \rightarrow \ldots$.

Схема для моделювання послідовного з’єднання лічильників *Cnt3* і *Cnt4*, де лічильник *Cnt3* є молодшим, приведена на рис.3.68. Результати моделювання цього з’єднання лічильників приведені на рис.3.69.

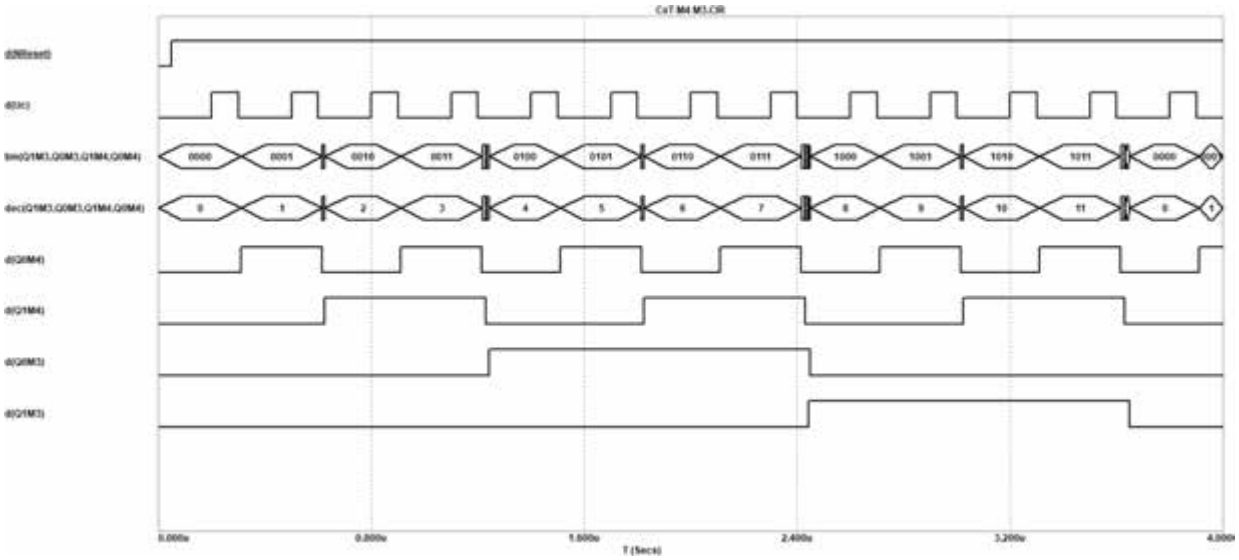


Рисунок 3.67 – Результати моделювання послідовного з’єднання лічильників *Cnt4* і *Cnt3*

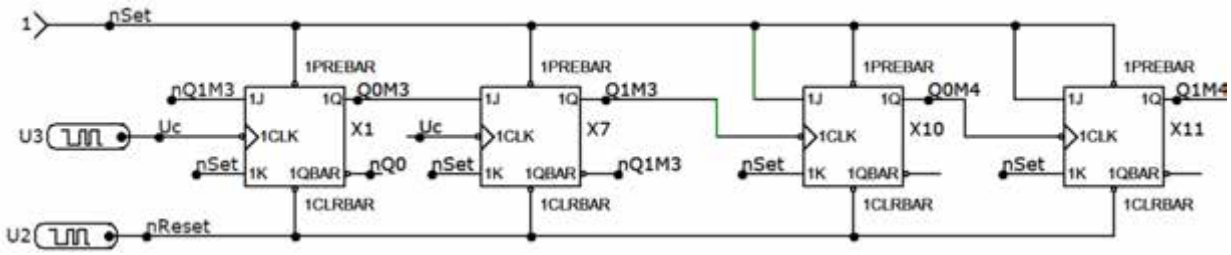


Рисунок 3.68 – Схема для моделювання послідовного з’єднання лічильників *Cnt3* і *Cnt4*

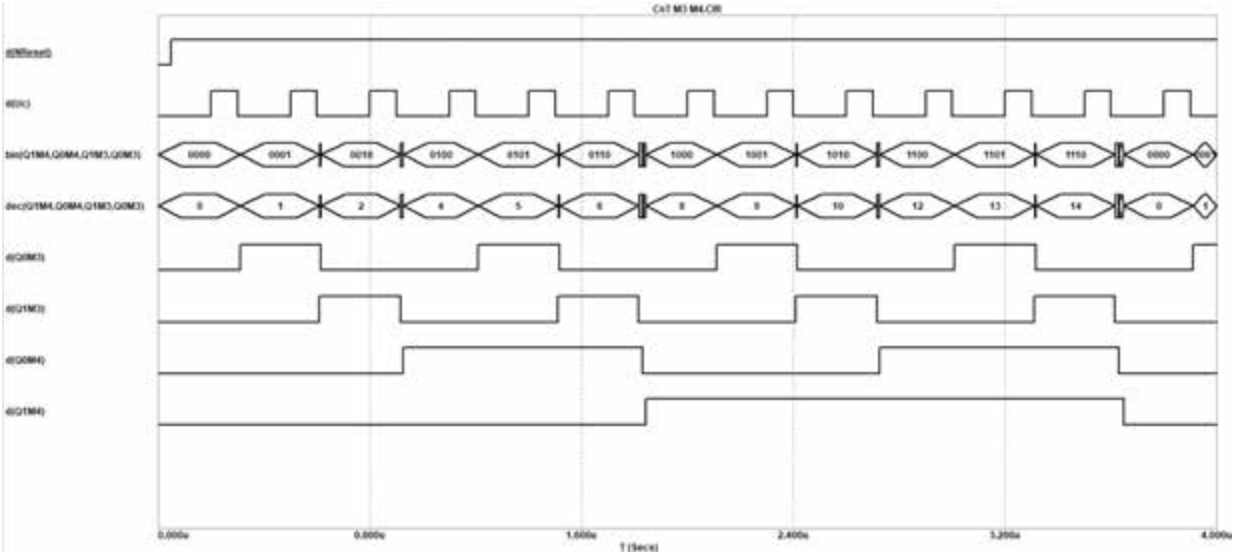


Рисунок 3.69 – Результати моделювання послідовного з'єднання лічильників *Cnt3* і *Cnt4*

На часовій діаграмі видно, що послідовність кодів станів не відповідає природній нумерації чисел:

$0 \rightarrow 1 \rightarrow 2_{10} \rightarrow 4_{10} \rightarrow 5_{10} \rightarrow 6_{10} \rightarrow 8_{10} \rightarrow 9_{10} \rightarrow 10_{10} \rightarrow 12_{10} \rightarrow 13_{10} \rightarrow 14_{10} \rightarrow 0 \rightarrow \ldots$ ..

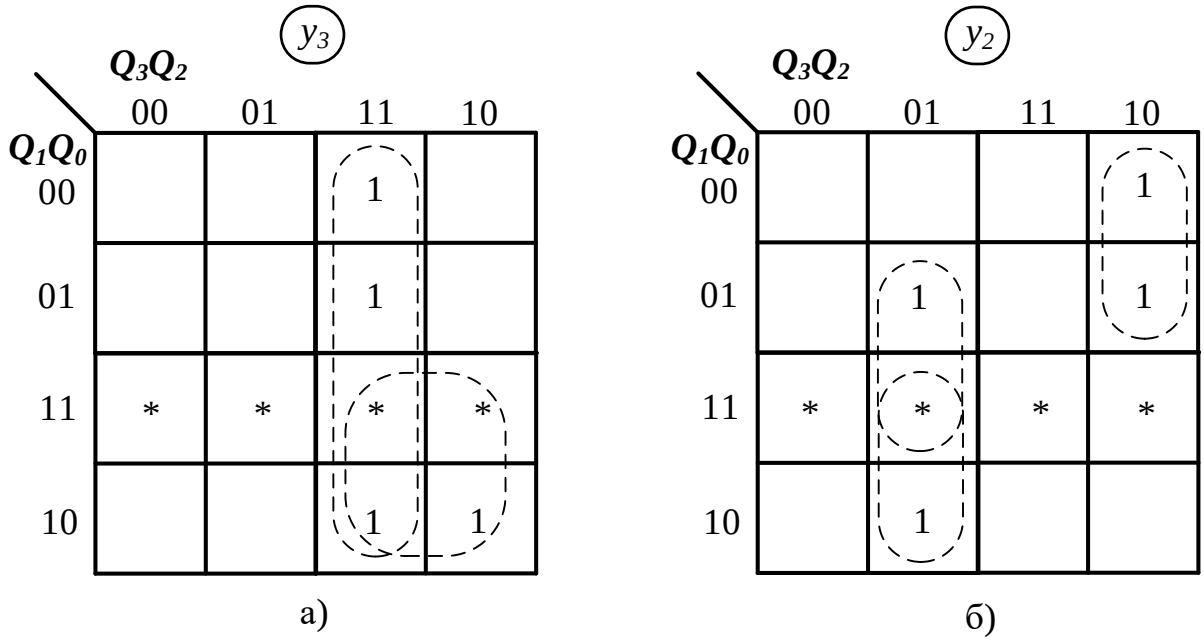
Таким чином, цей лічильник може бути використаний тільки в режимі ділення частоти (в даному випадку на *12*).

Для забезпечення можливості використання лічильника в режимі підрахунку вхідних сигналів до виходу лічильника необхідно підключити комбінаційний перетворювач кодів (ПК) станів лічильника в природню послідовність зміни кодів. Таблиця істинності перетворювача кодів для даного лічильника приведена в табл.3.13.

Таблиця 3.13 – Таблиця істинності ПК лічильника

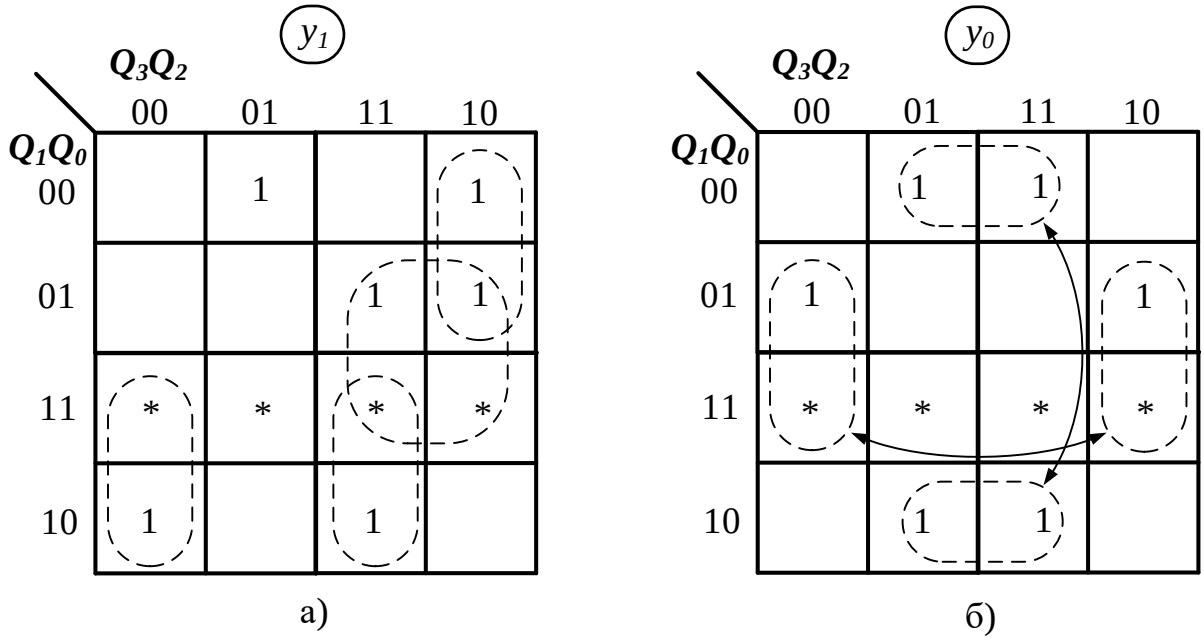
Номери наборів	Стан лічильника				Вихідний код перетворювача			
	Q_3^t	Q_2^t	Q_1^t	Q_0^t	y_3	y_2	y_1	y_0
1	2	3	4	5	6	7	8	9
0	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
1	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*1*
2	*0*	*0*	*1*	*0*	*0*	*0*	*1*	*0*
3	*0*	*0*	*1*	*1*	*	*	*	*
4	*0*	*1*	*0*	*0*	*0*	*0*	*1*	*1*
5	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*0*
6	*0*	*1*	*1*	*0*	*0*	*1*	*0*	*1*
7	*0*	*1*	*1*	*1*	*	*	*	*
8	*1*	*0*	*0*	*0*	*0*	*1*	*1*	*0*
9	*1*	*0*	*0*	*1*	*0*	*1*	*1*	*1*
10	*1*	*0*	*1*	*0*	*1*	*0*	*0*	*0*
11	*1*	*0*	*1*	*1*	*	*	*	*
12	*1*	*1*	*0*	*0*	*1*	*0*	*0*	*1*
13	*1*	*1*	*0*	*1*	*1*	*0*	*1*	*0*
14	*1*	*1*	*1*	*0*	*1*	*0*	*1*	*1*
15	*1*	*1*	*1*	*1*	*	*	*	*

В лічильнику, схема якого приведена на рис.3.68, відсутні стани з кодами 0011_2, 0111_2, 1011_2, 1111_2 (рядки 3, 7, 11, 15), тому в цих рядках стани ПК (колонки 6 - 9) є невизначеними і відмічені символом «*». В інших рядках вихідні коди ПК утворюють природню послідовність зміни станів $0 \rightarrow 1 \rightarrow 2_{10} \rightarrow 3_{10} \rightarrow \ldots \rightarrow 11_{10}$. Карти Карно для визначення мінімальних логічних виразів, що описують функціонування виходів ПК (*y3*, *y2*, *y1*, *y0*) приведені на рис.3.70, 3.71.



а) б)

Рисунок 3.70 – Карта Карно для виходів *y3*, *y2* ПК



а) б)

Рисунок 3.71 – Карта Карно для виходів *y1*, *y0* ПК

В результаті отримаємо логічні вирази для реалізації ПК:

$y_3 = Q_3 \cdot Q_2 \vee Q_3 \cdot Q_1 = Q_3 \cdot (Q_2 \vee Q_1)$;

$y_2 = Q_3 \cdot \overline{Q_2} \cdot \overline{Q_1} \vee \overline{Q_3} \cdot Q_2 \cdot Q_0 \vee \overline{Q_3} \cdot Q_2 \cdot Q_1$;

$y_1 = Q_3 \cdot Q_0 \vee Q_3 \cdot \overline{Q_2} \cdot \overline{Q_1} \vee Q_3 \cdot Q_2 \cdot Q_1 \vee \overline{Q_3} \cdot \overline{Q_2} \cdot Q_1 \cdot \overline{Q_0} \vee \overline{Q_3} \cdot Q_2 \cdot \overline{Q_1} \cdot \overline{Q_0} =$

$= Q_3 \cdot Q_0 \vee Q_3 \cdot (\overline{Q_2} \cdot \overline{Q_1} \vee Q_2 \cdot Q_1) \vee \overline{Q_3} \cdot \overline{Q_0} \cdot (\overline{Q_2} \cdot Q_1 \vee Q_2 \cdot \overline{Q_1}) =$

$= Q_3 \cdot Q_0 \vee Q_3 \cdot (\overline{Q_2 \oplus Q_1}) \vee \overline{Q_3} \cdot \overline{Q_0} \cdot (Q_1 \oplus Q_2)$;

$y_0 = \overline{Q_2} \cdot Q_0 \vee Q_2 \cdot \overline{Q_0} = Q_0 \oplus Q_2$.

Схема для моделювання послідовного з'єднання лічильників *Cnt3* і *Cnt4* з перетворювачем кодів приведена на рис.3.72, а результати моделювання – на рис.3.73.

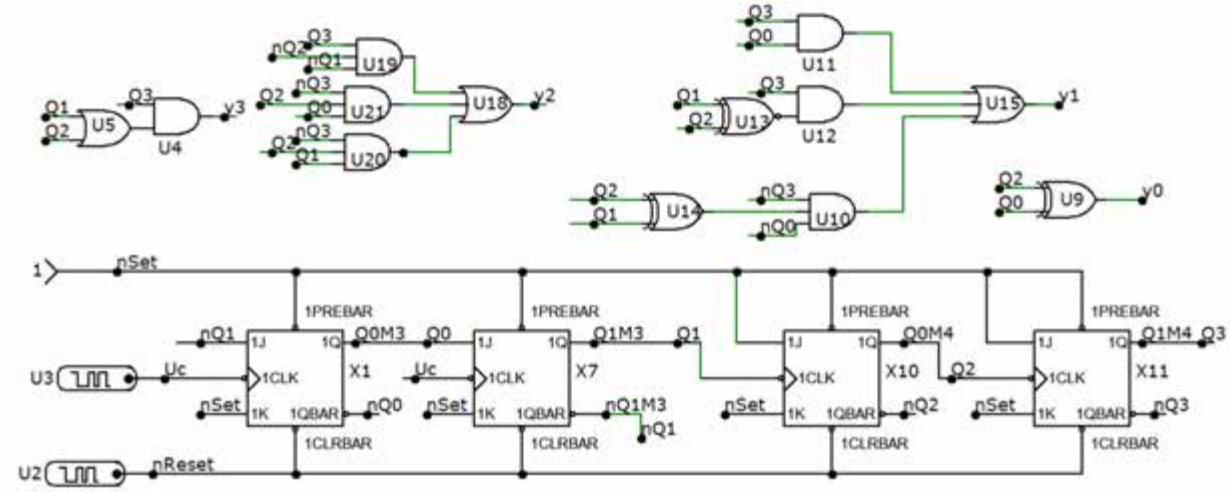


Рисунок 3.72 – Схема для моделювання послідовного з'єднання лічильників *Cnt3* і *Cnt4* з ПК

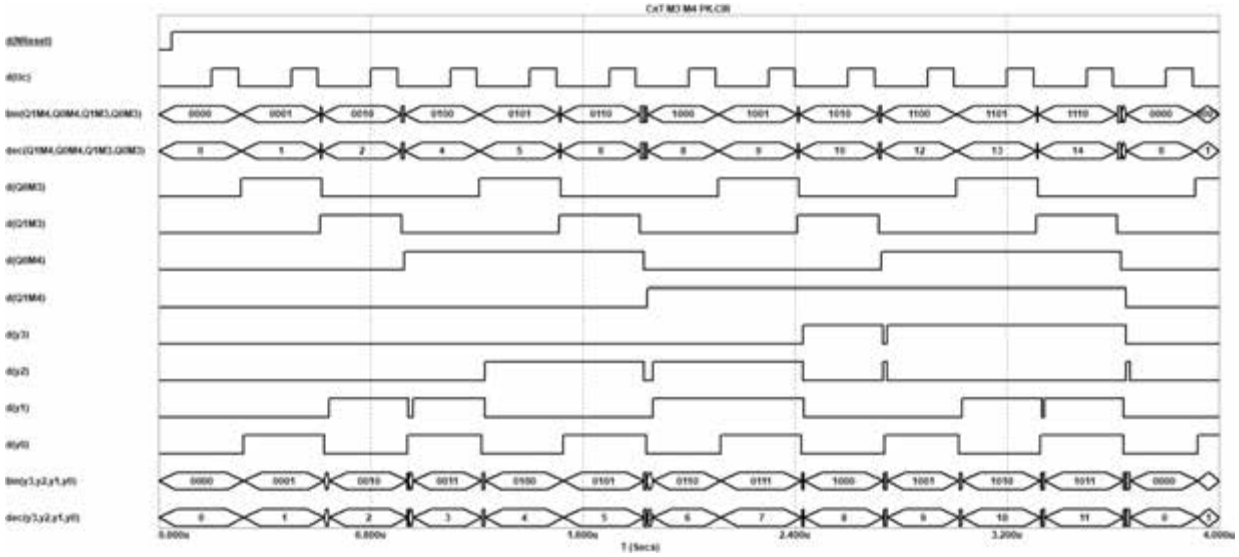


Рисунок 3.73 – Результати моделювання послідовного з'єднання лічильників *Cnt3* і *Cnt4* з ПК

На часовій діаграмі видно, що на виходах перетворювача кодів $y_3 - y_0$ формується природня послідовність зміни кодів станів лічильника, але недоліком такої організації лічильника є достатня складність перетворювача кодів.

3.2.4 Лічильники з асинхронним встановленням початкового стану

Структурна схема лічильників за модулем *M* з асинхронним встановленням початкового стану приведена на рис.3.74.

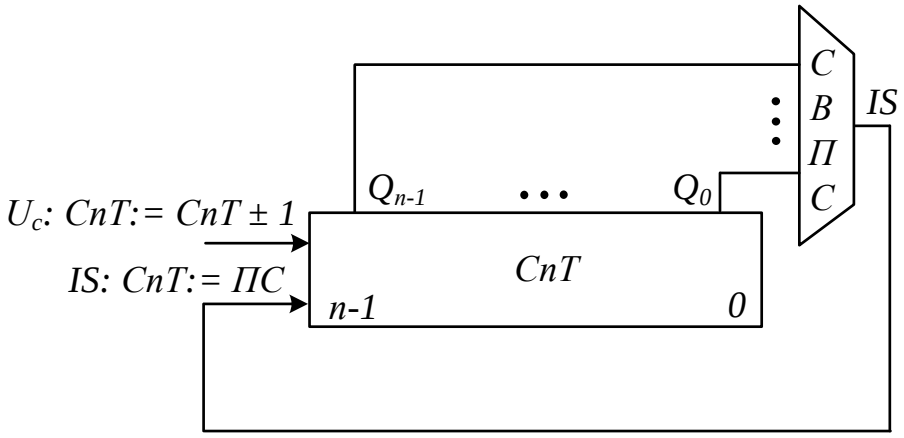


Рисунок 3.74 – Структурна схема лічильника з асинхронним встановленням початкового стану

На структурній схемі використовуються такі позначення:

- *CnT* – лічильник;

- *СВПС* – схема встановлення лічильника в початковий стан;

- *IS* – сигнал встановлення в початковий стан (***initial state***);

- *ПС* – код початкового стану.

Зі структурної схеми випливає, що на відміну від лічильників, що було розглянуто вище, наведений лічильник виконує дві мікрооперації:

- U_c: підрахунок вхідних сигналів (як правило, мікрооперації «інкремент» або «декремент»);

- *IS*: встановлення лічильника в заданий стан.

Таким чином, до складу лічильника входять два вузли:

- *n*-розрядний двійковий лічильник з модулем ліку $M_{дв} = 2^n$, який є базовим для підсумкового лічильника;

- комбінаційна схема встановлення лічильника в початковий стан.

Розглянемо принцип функціонування підсумкового підсумовувального лічильника за модулем M ($M_{дв} > M$). Спочатку лічильник працює як звичайний двійковий лічильник за модулем $M_{дв}$. Коли лічильник досягне стану $M - 1$, спрацьовує схема встановлення в початковий стан *СВПС*, яка формує активне значення сигналу *IS*. Цей сигнал надходить на асинхронні входи *R* базових тригерів, переключаючи ці тригери в нульовий стан, після чого цикл роботи лічильника повторюється.

Підсумковий віднімальний лічильник за модулем *M* також спочатку працює як звичайний двійковий лічильник. При цьому коли лічильник досягне нульового стану, спрацьовує *СВПС*, яка формує активне значення сигналу *IS*. Цей сигнал надходить на асинхронні входи *R* або *S* базових тригерів, переключаючи ці тригери в стан $M - 1$, після чого цикл роботи лічильника повторюється.

Таким чином, порядок синтезу лічильників за модулем *M* з асинхронним встановленням початкового стану полягає у виконанні таких кроків:

1. Визначення кількості базових тригерів *n* відповідно до (3.1).

2. Синтез двійкового лічильника з модулем $M_{дв} = 2^n$ з будь-яким способом організації переносу між розрядами.

3. Синтез схеми встановлення лічильника в початковий стан *СВПС*.

4. Розробка схеми підсумкового лічильника.

5. Перевірка коректності функціонування підсумкового лічильника.

6. Визначення параметрів підсумкового лічильника.

Кроки 1 і 2 детально розглядалися в підрозділі 3.1.

Синтез *СВПС*, яка є комбінаційною, відбувається за допомогою таблиці істинності та подальшої мінімізації логічного виразу.

Кроки 4 і 5 виконуються аналогічно тому, як це робилося в попередніх підрозділах. Для забезпечення розробки схеми лічильника необхідно, щоб базові тригери мали асинхронні входи *R* для підсумовувальних лічильників та асинхронні входи *R* і *S* для віднімальних лічильників.

Визначення параметрів підсумкового лічильника буде відбуватися за результатами виконання попередніх кроків процедури синтезу.

Розглянемо синтез лічильника з асинхронним встановленням початкового стану на прикладі.

Приклад 3.5. Виконати синтез підсумовувального лічильника з асинхронним встановленням початкового стану за модулем *6* на основі *T*- тригерів.

Розв'язок

На першому кроці визначимо кількість базових тригерів лічильника. Відповідно до умови завдання $M = 6$, тобто згідно з виразом (3.1) $n =] \log_2 6 [=] 2{,}58.. [= 3$, тобто до складу лічильника входять три *T*- тригери.

На другому кроці необхідно синтезувати базовий двійковий лічильник за модулем $M_{дв} = 2^n = 8$. Синтез двійкових лічильників розглядався в п.3.1. В якості двійкового лічильника будемо використовувати тригери Q_2- Q_0 асинхронного підсумовувального лічильника з безпосереднім переносом, схема якого приведена на рис.3.45.

Далі розглянемо синтез схеми встановлення лічильника в початковий стан *СВПС*. Ця схема є комбінаційною, тобто для її синтезу будемо використовувати таблицю істинності, яка приведена в табл.3.14.

Порядок зміни кодів станів заданого лічильника відповідає послідовності $000_2 \rightarrow 001_2 \rightarrow 010_2 \rightarrow 011_2 \rightarrow 100_2 \rightarrow 101_2 \rightarrow 000_2$, тобто при досягненні лічильником стану 101_2 необхідно активізувати сигнал встановлення лічильника в початковий стан (*IS*) для переключення в нульовий стан. В зв'язку з тим, що в базових тригерах асинхронний вхід *R* є інверсним, то в таблиці 3.14 для стану 101_2 (рядок *6*) задається нульове

значення сигналу *IS*, яке є активним для асинхронних входів тригерів і далі будемо позначати цей сигнал *nIS*.

Таблиця 3.14 – Таблиця істинності *СВПС*

Номер набору	Q_2	Q_1	Q_0	*nIS*
1	*0*	*0*	*0*	*1*
2	*0*	*0*	*1*	*1*
3	*0*	*1*	*0*	*1*
4	*0*	*1*	*1*	*1*
5	*1*	*0*	*0*	*1*
6	*1*	*0*	*1*	*0*
7	*1*	*1*	*0*	*
8	*1*	*1*	*1*	*

Стани 110_2 і 111_2 в цьому лічильнику не використовуються, тому в таблиці істинності позначаються символом «*». Для решти станів значення сигналу *nIS* приймає одиничне (неактивне) значення.

Карта Карно для мінімізації логічного виразу, що описує сигнал *IS*, приведена на рис.3.75.

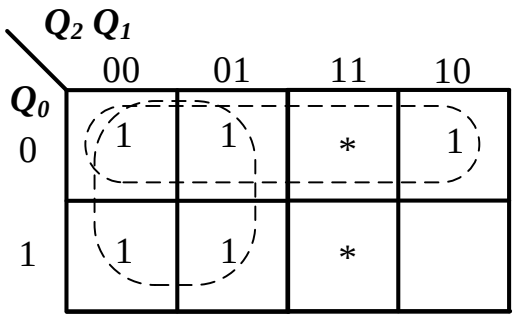


Рисунок 3.75 – Карта Карно для мінімізації *СВПС*

В результаті отримаємо:

$$nIS = \overline{Q_2} \vee \overline{Q_0} = \overline{Q_2 \cdot Q_0}\,.$$

Функціональна схема лічильника приведена на рис.3.76.

Схема для моделювання функціонування лічильника приведена на рис.3.77, а результати моделювання у вигляді часової діаграми функціонування цього пристрою – на рис.3.78.

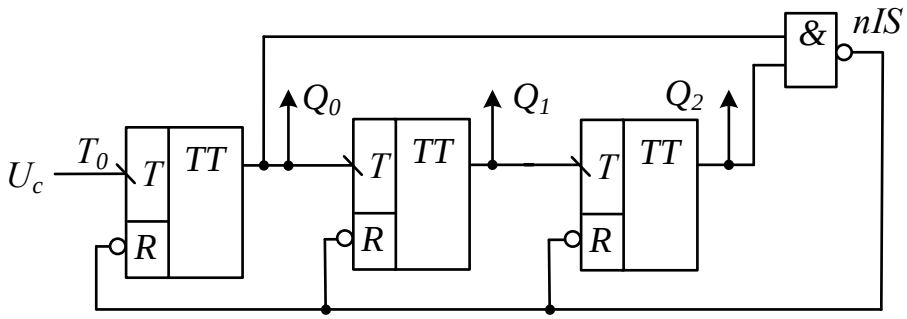


Рисунок 3.76 – Функціональна схема лічильника з асинхронним встановленням початкового стану

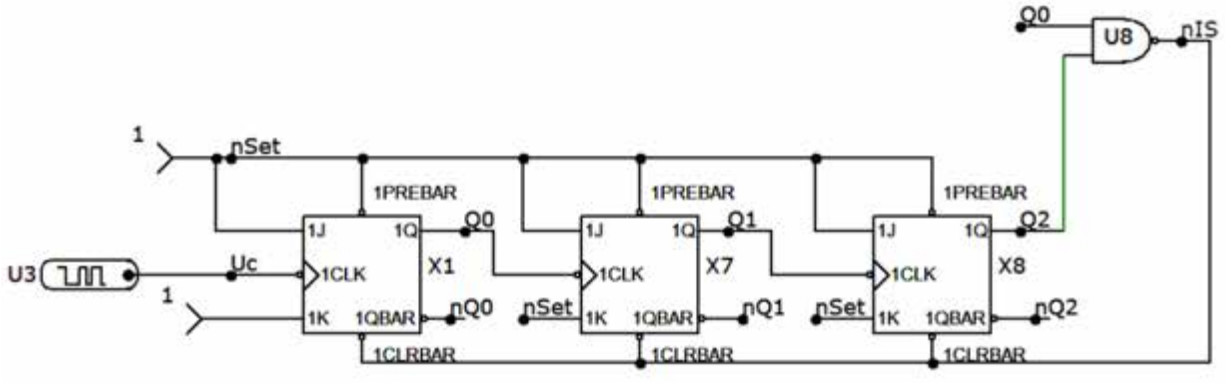


Рисунок 3.77 – Схема для моделювання лічильника з асинхронним встановленням початкового стану

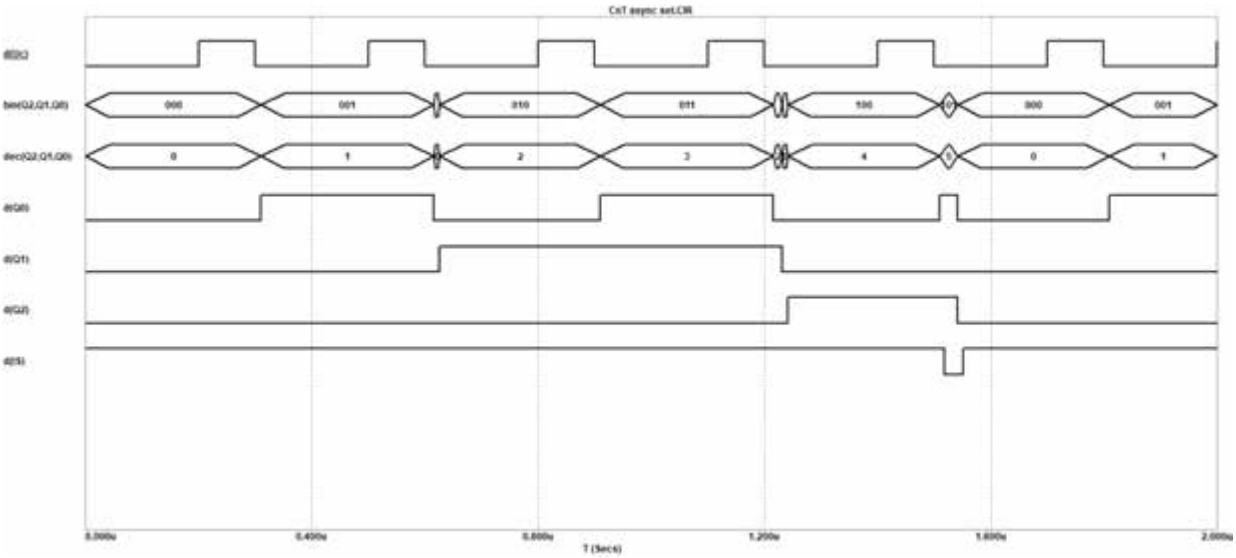


Рисунок 3.78 – Результати моделювання лічильника з асинхронним встановленням початкового стану

Виконуючи аналіз результатів моделювання, можна зробити такі висновки:

1. В зв’язку з асинхронним скидом стану лічильника тривалість останнього стану є дуже малою. Результати моделювання функціонування лічильника при спрацьовуванні *СВПС* приведені на рис.3.79.

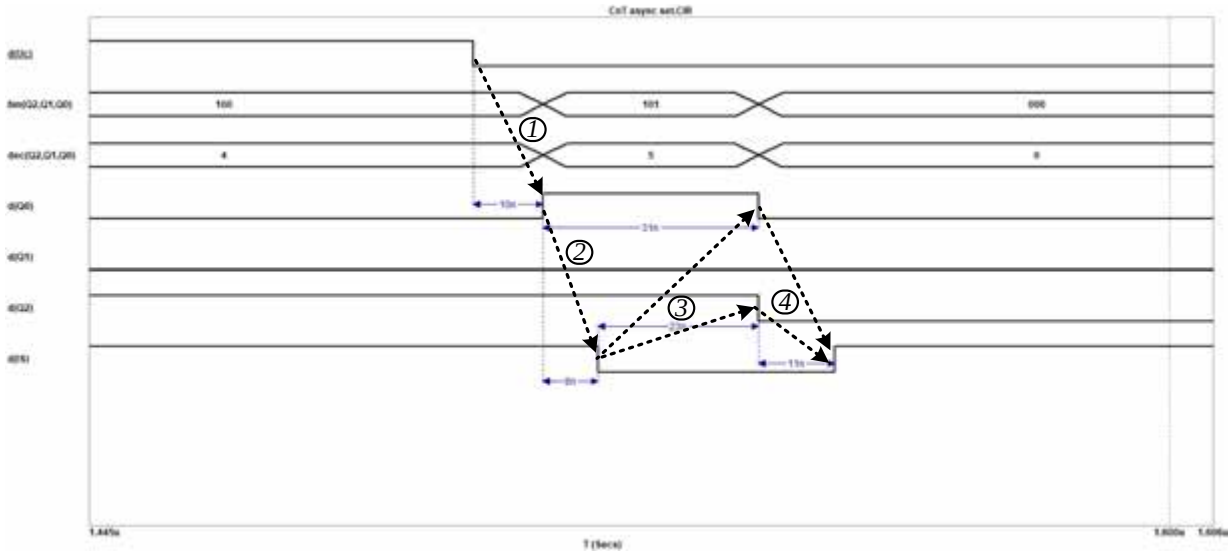


Рисунок 3.79 – Результати моделювання лічильника при спрацьовуванні *СВПС*

Пунктирними лініями на цьому рисунку показана послідовність переключення елементів лічильника при поточному стані 100_2 і надходженні лічильного сигналу U_c. Розглянемо ці процеси більш детально.

Через *10нс* після появи активного фронту сигналу U_c наймолодший тригер Q_0 переключається в одиничний стан (на часовій діаграмі позначено цифрою «1»), в результаті чого лічильник досягає стану *M - 1* (101_2). Це визиває спрацьовування *СВПС* і переключення сигналу *nIS* через *8нс* в активне нульове значення («2»). В свою чергу сигнал *nIS* ініціює переключення базових тригерів лічильника в нульовий стан через *23нс* («3»). Далі новий стан лічильника переводить *СВПС* в неактивний стан через *8нс* («4»). Таким чином, тривалість стану *M - 1* дорівнює *31нс*, що складається з інтервалів «2», «3».

В результаті розгляду переключень послідовності станів лічильника $100_2 \rightarrow 101_2 \rightarrow 000_2$ можна зробити висновок, що з точки зору ділення частоти синтезований лічильник фактично є лічильником за модулем *5* (частота вихідного сигналу зменшується в *5* разів), але за кількістю станів наявність,

хоч і короткого, стану 101_2 дає можливість вважати, що модуль цього лічильника дорівнює 6, тим більше, що цей останній стан може мати вплив на функціонування навантаження лічильника.

2. Розглянемо випадок, коли в якості двійкового лічильника використовується лічильник з паралельним трактом розповсюдження переносу. В цьому разі за наявності різниці між часом спрацьовування базових тригерів лічильника по сигналу U_c існує ймовірність некоректного функціонування пристрою. Наприклад, розглянемо випадок, при якому наймолодший тригер Q_0 переключається довше інших. Розглянемо переключення лічильника зі стану $Q_2 = 0$; $Q_1 = 1$; $Q_0 = 1$. Звичайно, що наступним станом повинен бути стан $Q_2 = 1$; $Q_1 = 0$; $Q_0 = 0$. Однак, при затримці спрацьовування Q_0 тригери Q_2 і Q_1 вже переключаться відповідно в стан 1 і 0, а наймолодший тригер ще залишається в одиничному стані, тобто на виходах лічильника формується код $Q_2 = 1$; $Q_1 = 0$; $Q_0 = 1$, що призводить до спрацьовування *СВПС* і передчасному скиду лічильника.

Схема для моделювання функціонування такого лічильника приведена на рис.3.80, а результати моделювання у вигляді часової діаграми цього пристрою – на рис.3.81.

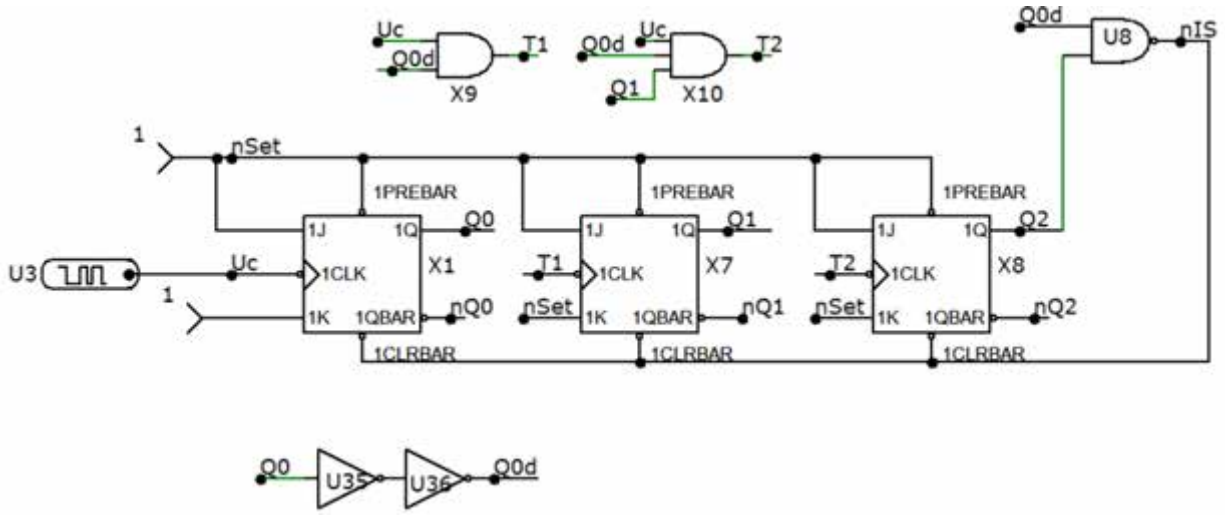


Рисунок 3.80 – Схема для моделювання лічильника з асинхронним встановленням початкового стану із затримкою переключення Q_0

На рис.3.80 моделювання затримки переключення наймолодшого тригера Q_0 здійснюється за рахунок додавання двох послідовно з'єднаних інверторів *U35*, *U36* і використанні сигналу Q_{0d} замість Q_0.

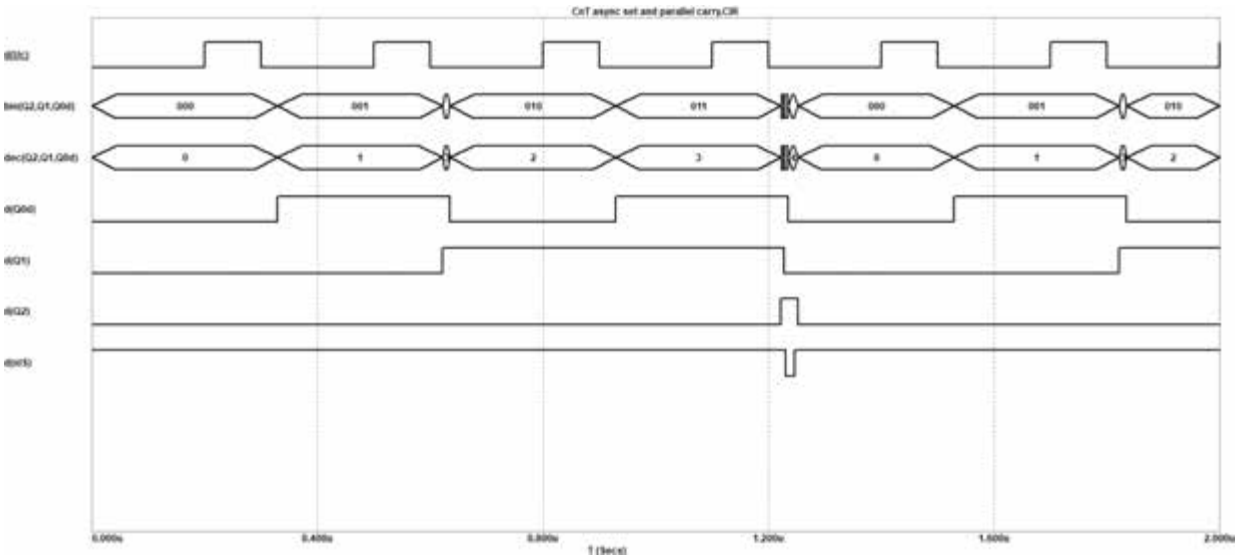


Рисунок 3.81 – Результати моделювання лічильника з асинхронним встановленням початкового стану із затримкою переключення Q_0

На рис.3.81 можна побачити, що в зв'язку із затримкою спрацьовування тригера Q_0 активне значення сигналу *nIS* формується передчасно і лічильник після стану 011_2 переключається в нульовий стан. Більш детальні результати моделювання роботи цього лічильника при переключенні зі стану 011_2 приведені на рис.3.82.

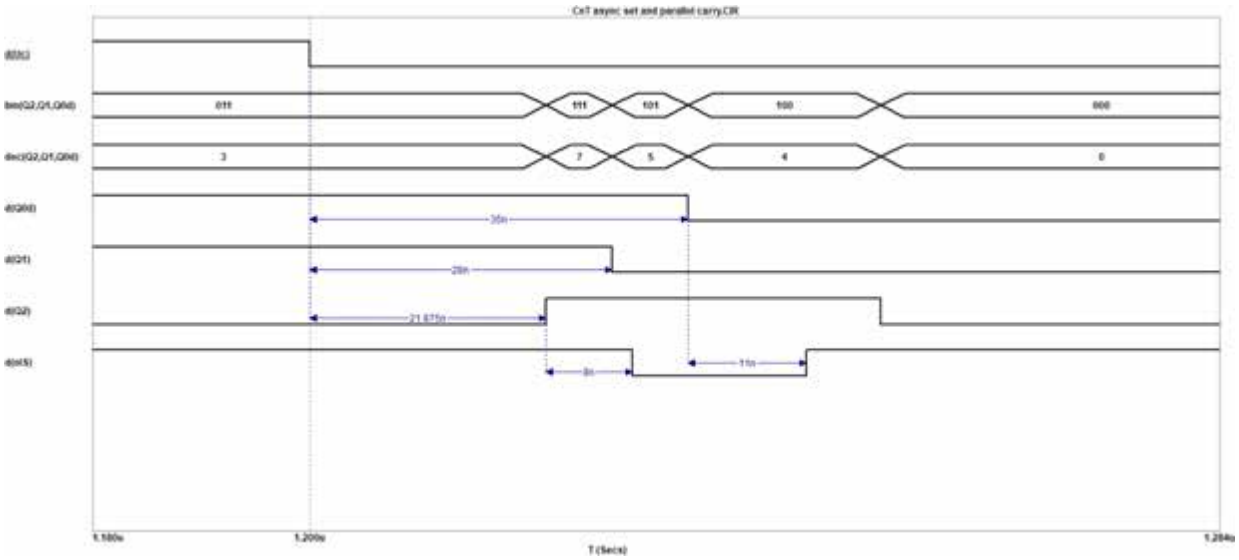


Рисунок 3.82 – Результати моделювання лічильника з асинхронним встановленням початкового стану при переключенні зі стану 011_2

3. Крім того, існує ймовірність неправильної роботи лічильника за наявності різниці у часі реакції тригерів на активне значення установчого сигналу *nIS*. В цьому випадку, якщо один з тригерів, наприклад Q_0, реагує на сигнал *nIS* довше інших, то всі тригери окрім Q_0 вже переключаться, переводячи схему формування *nIS* в неактивний стан, тобто сигнал скиду тригера Q_0 стає неактивним (його активне значення може бути доволі коротким і відфільтровуватися за рахунок наявності у логічних елементів інерційної затримки), в результаті чого тригер може не переключитися в нульовий стан і лічильник буде працювати некоректно.

На цьому виконання прикладу 3.5 завершено.

Для реалізації віднімальних лічильників процедура синтезу не відрізняється від розглянутої раніше, але при досягненні лічильником нульового стану його тригери необхідно переключати таким чином, щоб на виходах тригерів лічильника сформувати код $M - 1$, тобто в цьому випадку необхідно використовувати тригери з наявністю як входів *R* або *nR*, так і *S* або *nS*.

Схема для моделювання функціонування віднімального лічильника за модулем *6* приведена на рис.3.83.

Активне (нульове) значення сигналу встановлення початкового стану *nIS* формується при нульовому стані лічильника за виразами:

$$nIS = \overline{Q_2} \cdot \overline{Q_1} \cdot \overline{Q_0} = \overline{Q_2 \vee Q_1 \vee Q_0}\,.$$

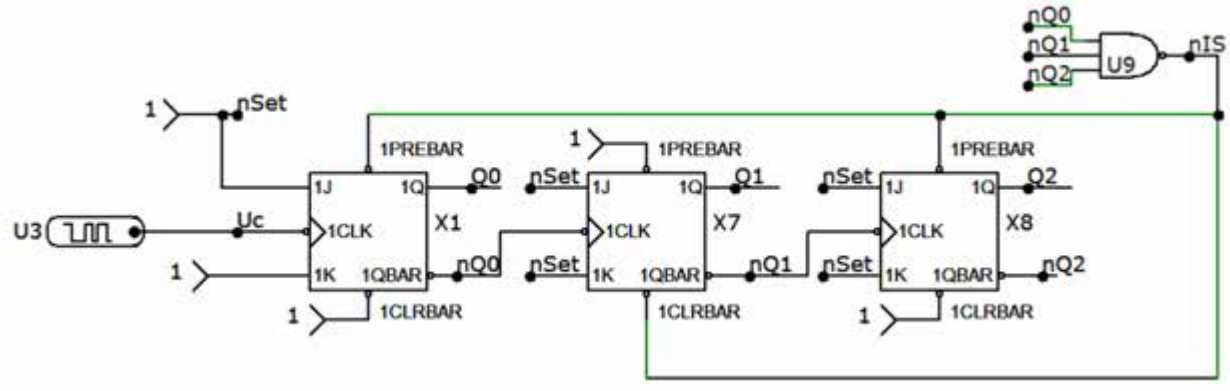


Рисунок 3.83 – Схема для моделювання віднімального лічильника з асинхронним встановленням початкового стану

Результати моделювання цього лічильника приведені рис.3.84.

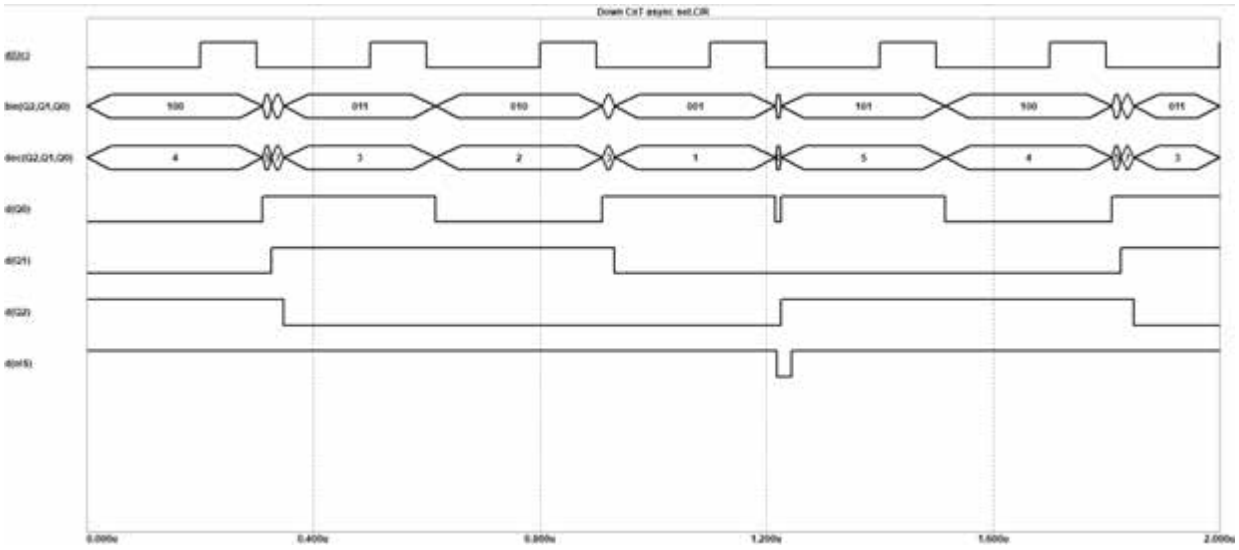


Рисунок 3.84 – Результати моделювання віднімального лічильника з асинхронним встановленням початкового стану

Більш детальна ілюстрація функціонування лічильника при досягненні нульового стану приведено на рис.3.85.

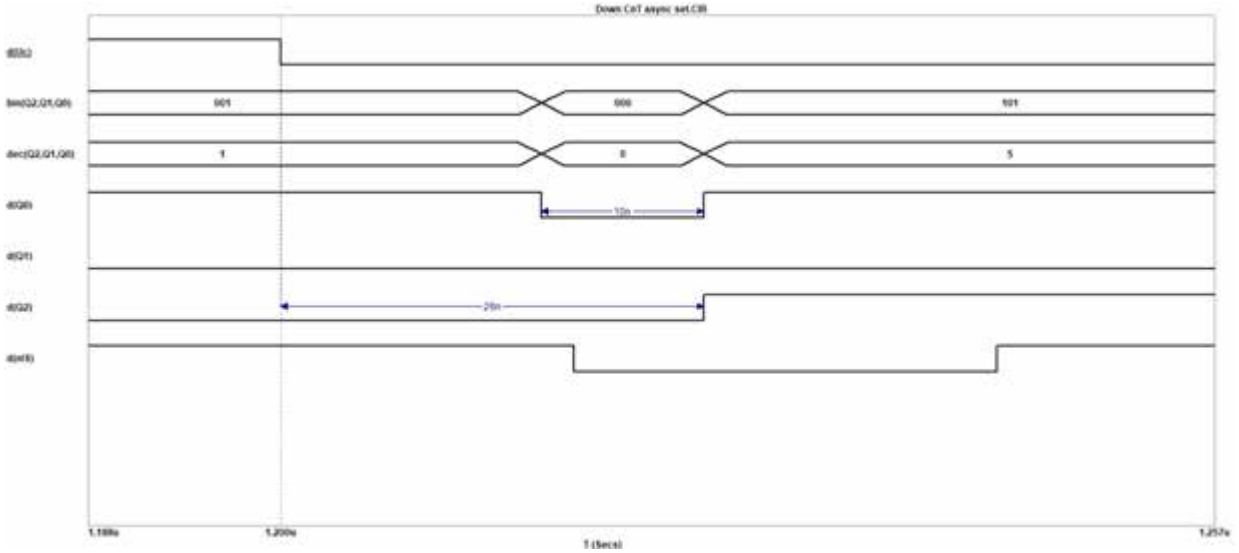


Рисунок 3.85 – Результати моделювання віднімального лічильника при досягненні нульового стану

3.2.5 Лічильники з синхронним встановленням початкового стану

Лічильники з синхронним встановленням початкового стану відрізняються більш надійним функціонуванням. Структурна схема лічильників за модулем *M* з синхронним встановленням початкового стану

аналогічна структурній схемі, що приведена на рис.3.74, але має більш складну організацію *СВПС*.

Структурно-функціональна схема *СВПС* приведена на рис.3.86.

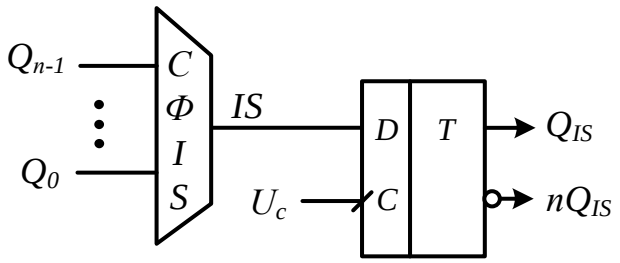


Рисунок 3.86 – Структурно-функціональна схема *СВПС*

Відповідно до цієї схеми, *СВПС* складається з комбінаційної схеми формування активного значення сигналу *IS* (*СФIS*) та додаткового *D*-тригера, який призначений для запам'ятовування факту появи активного значення сигналу *IS*. Виходи цього тригера Q_{IS} і/або nQ_{IS} далі підключаються до асинхронних входів базових тригерів лічильника. Звичайно, що вихід Q_{IS} (nQ_{IS}) використовується у випадку, якщо в тригерах використовуються прямі (інверсні) асинхронні установчі входи.

Синтез лічильника відбувається за тією же процедурою, що використовувалася в попередньому підрозділі.

Розглянемо приклад синтезу такого лічильника.

Приклад 3.6. Виконати синтез підсумовувального лічильника з синхронним встановленням початкового стану за модулем *6* на основі *T*- тригерів.

Розв'язок

За аналогією з прикладом 3.5 спочатку визначаємо кількість базових тригерів заданого лічильника відповідно до виразу (3.1): $n =] \log_2 M [=] \log_2 6 [=] 2{,}58.. [= 3$.

Далі використовуємо базовий двійковий лічильник за модулем $M_{дв} = 2^n = 8$ з будь-яким способом організації переносу між розрядами (підрозділ 3.1). В якості базового двійкового лічильника оберемо

асинхронний підсумовувальний лічильник з безпосереднім переносом (рис.3.45).

Після цього виконаємо синтез схеми формування сигналу *IS*, для чого використаємо таблицю істинності, аналогічну табл.3.14.

Таблиця істинності для синтезу *СФІS* заданого лічильника приведена в табл.3.15.

Таблиця 3.15 – Таблиця істинності *СФІS*

Номер набору	Q_2	Q_1	Q_0	*IS*
1	*0*	*0*	*0*	*0*
2	*0*	*0*	*1*	*0*
3	*0*	*1*	*0*	*0*
4	*0*	*1*	*1*	*0*
5	*1*	*0*	*0*	*0*
6	*1*	*0*	*1*	*1*
7	*1*	*1*	*0*	*
8	*1*	*1*	*1*	*

Карта Карно для мінімізації логічного виразу *СФІS* приведена на рис.3.75.

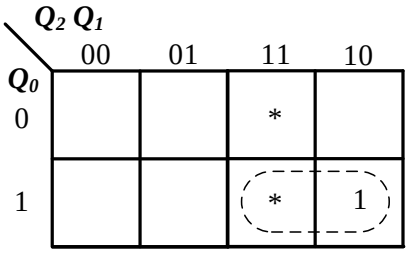


Рисунок 3.87 – Карта Карно для мінімізації *СФІS*

Таким чином, в результаті мінімізації отримаємо:

$$IS = Q_2 \cdot Q_0 = \overline{\overline{Q_2} \vee \overline{Q_0}}\,.$$

Функціональна схема лічильника приведена на рис.3.88. Розглянемо принцип функціонування цього лічильника.

Як було відзначено під час розглядання прикладу 3.5, порядок зміни кодів станів на виходах заданого лічильника відповідає послідовності $000_2 \rightarrow 001_2 \rightarrow \ldots \rightarrow 101_2 \rightarrow 000_2$.

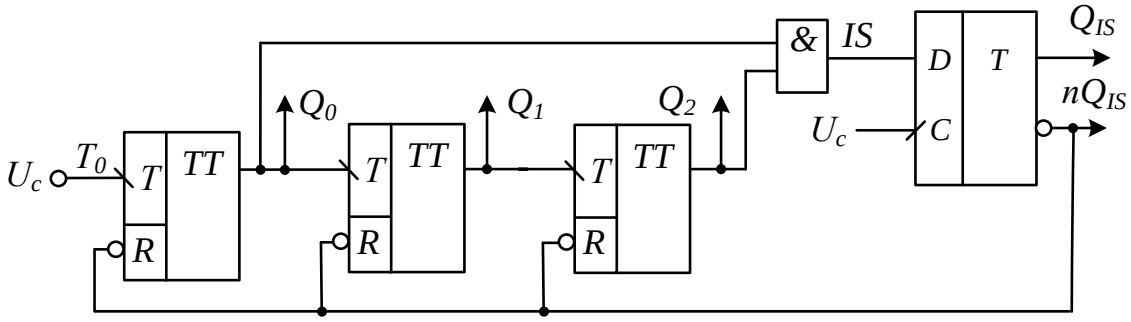


Рисунок 3.88 – Функціональна схема лічильника з синхронним встановленням початкового стану (приклад 3.6)

При досягненні лічильником стану 101_2 відбувається активізація схеми формування сигналу *IS*, який приймає активне (одиничне) значення ы надходить на вхід *D* додаткового тригера Q_{IS}. Далі за наступним лічильним сигналом U_c тригер Q_{IS} переключається в одиничний стан, і його інверсний вихід nQ_{IS} – в нульовий стан. Низький рівень сигналу nQ_{IS} надходить на асинхронні входи скиду базових тригерів, переключаючи їх в нульовий стан. Нульовий стан лічильника приводить до переключення сигналу *IS* в неактивний (нульовий) стан. За переднім фронтом наступного сигналу U_c тригер Q_{IS} переключається в нульовий стан і лічильник переходить до нового циклу підрахунку вхідних сигналів.

Функціонування лічильника фактично відбувається з використанням двофазної синхронізації, де базові тригери лічильника спрацьовують за заднім фронтом сигналу U_c, а додатковий тригер Q_{IS} – за переднім фронтом цього сигналу.

Схема для моделювання функціонування лічильника приведена на рис.3.89, а результати моделювання у вигляді часової діаграми функціонування цього пристрою – на рис.3.90.

На рис.3.91 приведені результати моделювання лічильника при спрацьовуванні *СФIS* , а послідовність переключення елементів лічильника показані стрілками.

Визначення динамічних параметрів виконується аналогічно тому, як було розглянуто в попередніх прикладах.

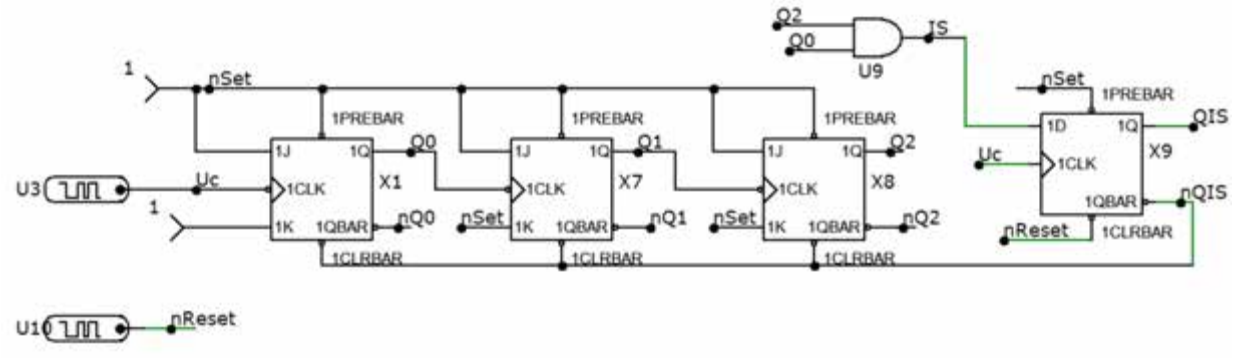


Рисунок 3.89 – Схема для моделювання лічильника з синхронним встановленням початкового стану (приклад 3.6)

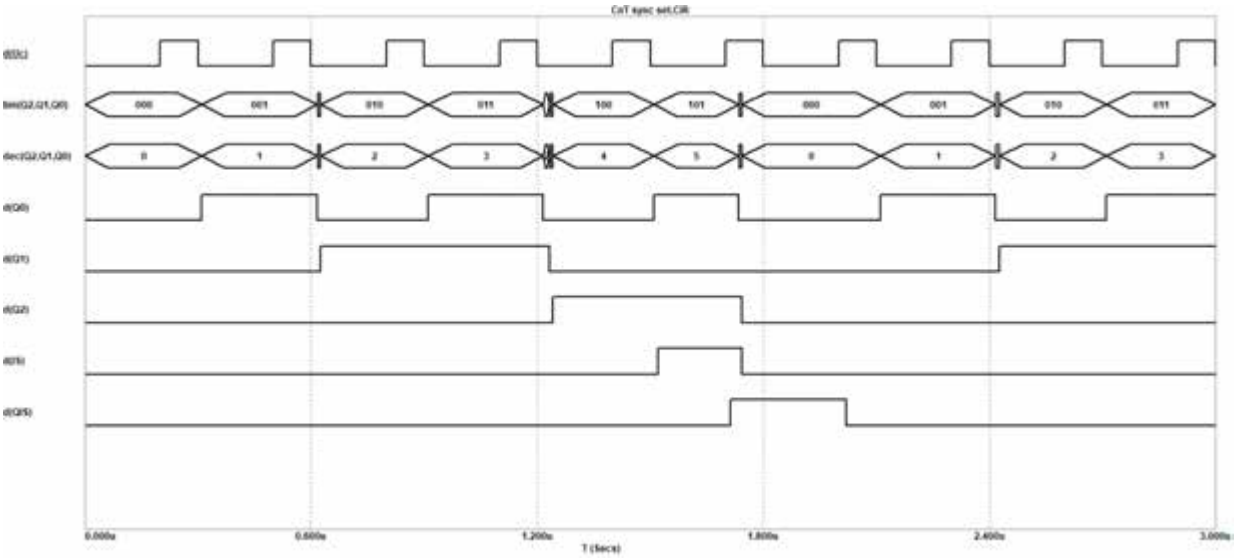


Рисунок 3.90 – Результати моделювання лічильника з синхронним встановленням початкового стану (приклад 3.6)

На цьому виконання прикладу 3.6 завершено.

Синтез віднімальних лічильників виконується аналогічно, але для встановлення лічильника в початковий стан необхідно використовувати асинхронні входи базових тригерів обох типів (як *S*, так і *R*).

Розглянутий спосіб організації лічильника з довільним модулем ліку дозволяє проєктувати більш надійні лічильники, ніж в попередньому підрозділі, але в схемах такого лічильника використовується додатковий тригер.

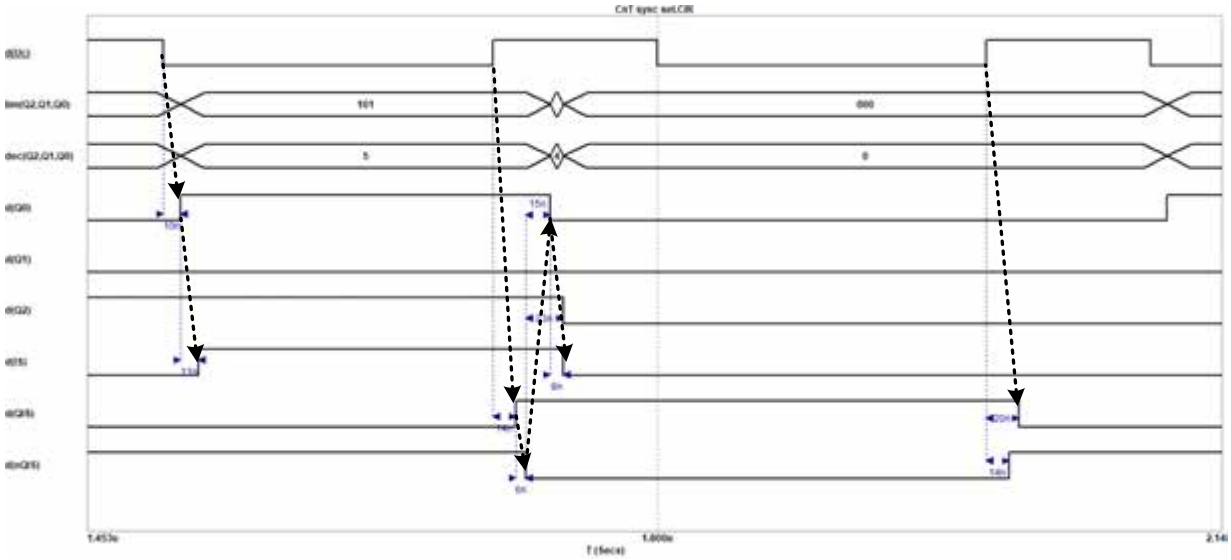


Рисунок 3.91 – Послідовність переключення логічних елементів лічильника при спрацьовуванні *СФІS* (приклад 3.6)

Під час розробки лічильників з довільним модулем ліку можна використовувати будь-який спосіб апаратної реалізації, розглянутий вище, або комбінацію цих способів. Наприклад, для побудови лічильника за модулем *10* можна застосувати класичний метод синтезу або використовувати послідовне з'єднання лічильників з модулями *2* і *5*, де лічильник з модулем *5* будується за структурою 2^k+1.

Контрольні завдання та запитання

1. Які лічильники відносяться до двійкових лічильників?

2. Які методи використовуються для синтезу лічильників з довільним модулем ліку?

3. Охарактеризуйте класичний метод синтезу лічильників.

4. В чому полягає різниця між синтезом двійкових лічильників і лічильників з довільним модулем ліку за допомогою класичного метода?

5. Як визначити кількість базових тригерів у складі лічильника з довільним модулем ліку?

6. Приведіть таблицю переходів для синтезу підсумовувального лічильника за модулем *5*.

7. Приведіть таблицю переходів для синтезу віднімального лічильника за модулем *5*.

8. Приведіть таблицю переходів для синтезу підсумовувального лічильника за модулем *10*.

9. Приведіть таблицю переходів для синтезу віднімального лічильника за модулем *10*.

10. Приведіть послідовність зміни станів підсумовувального десяткового лічильника.

11. Приведіть послідовність зміни станів віднімального десяткового лічильника.

12. Приведіть послідовність зміни станів підсумовувального лічильника за модулем *12*.

13. Приведіть послідовність зміни станів віднімального лічильника за модулем *12*.

14. Поясніть, яким чином заповнюється таблиця 3.9.

15. Як позначаються в таблиці переходів неіснуючі стани лічильника?

16. Поясніть термін «функції збудження тригерів».

17. Як визначити функції збудження асинхронного *T*-тригера?

18. Як визначити функції збудження синхронного *TC*-тригера?

19. Як визначити функції збудження *D*-тригера?

20. Як визначити функції збудження *JK*-тригера?

21. Як визначити функції збудження *RCS*-тригера?

22. Поясніть, яким чином заповнювалися стовпчики 11-19 в табл.3.9.

23. Поясніть, яким чином заповнюється таблиця 3.10.

24. Доведіть вирази (3.19).

25. Яким чином побудовано схему на рис.3.56?

26. З якою метою в схемі лічильника на рис.3.57 використовується сигнал ***nReset***?

27. Прокоментуйте часову діаграму на рис.3.58.

28. Визначте модуль ліку лічильника по часовій діаграмі на рис.3.58.

29. Як визначити динамічні параметри лічильника, схема якого приведена на рис.3.56?

30. Прокоментуйте часову діаграму на рис.3.59.

31. Як визначити час спрацьовування лічильника, схема якого приведена на рис.3.56?

32. Як визначити час формування переносу зі старшого розряду лічильника, схема якого приведена на рис.3.56?

33. Який сигнал є сигналом переносу зі старшого розряду лічильника, схема якого приведена на рис.3.56?

34. Як визначити максимальну частоту надходження лічильного сигналу?

35. Прокоментуйте вирази (3.20).

36. Яким чином побудовано схему на рис.3.60?

37. Прокоментуйте часову діаграму на рис.3.61.

38. Визначте модуль ліку лічильника по часовій діаграмі на рис.3.61.

39. Який тип переносу між розрядами лічильника можна отримати за допомогою класичного метода синтезу?

40. В чому полягає перевага використання класичного метода синтезу лічильників?

41. В чому полягають недоліки використання класичного метода синтезу лічильників?

42. В чому полягає особливість побудови лічильників за модулем $M = 2^k + 1$?

43. Який тип лічильників з точки зору напряму ліку можна побудувати, використовуючи структуру лічильників з модулем $2^k + 1$?

44. Як визначити кількість тригерів для побудови лічильників з модулем $2^k + 1$?

45. В чому полягає особливість послідовності зміни станів лічильників з модулем $2^k + 1$?

46. Який загальний вигляд мають числа 2^k в двійковій системі числення?

47. Поясніть, яким чином заповнюється таблиця 3.11.

48. Прокоментуйте приведену на рис.3.62 структурно-функціональну схему підсумовувального лічильника за модулем $2^k + 1$.

49. Поясніть принцип роботи лічильника по структурно-функціональній схемі лічильника на рис.3.62.

50. Який лічильник реалізовано на тригерах Q_0 - Q_{n-2} в схемі на рис.3.62?

51. Які типи переносів між розрядами можна використовувати в частині лічильника з модулем $2^k + 1$ на тригерах Q_0 - Q_{n-2}?

52. Які тригери можна використовувати в розрядах Q_1 - Q_{n-2} в лічильниках з модулем $2^k + 1$?

53. Які тригери можна використовувати в розрядах Q_0 і Q_{n-1} в лічильниках з модулем $2^k + 1$?

54. Яким чином працює частина лічильника з модулем 2^k+1, реалізована на тригерах Q_0 - Q_{n-2}?

55. Прокоментуйте схему лічильника на рис.3.63.

56. Прокоментуйте часову діаграму на рис.3.64.

57. Визначте модуль ліку лічильника по часовій діаграмі на рис.3.64.

58. Як визначити динамічні параметри лічильників з модулем 2^k+1?

59. Виконати побудову підсумовувального лічильника з модулем *3*, використовуючи структуру 2^k+1.

60. Виконати побудову підсумовувального лічильника з модулем *5*, використовуючи структуру 2^k+1.

61. Виконати побудову підсумовувального лічильника з модулем *17*, використовуючи структуру 2^k+1.

62. Як визначити загальний модуль ліку підсумкового лічильника при використанні послідовного з'єднання кількох лічильників?

63. Яким чином виконується послідовне з'єднання двох лічильників?

64. Який лічильник при послідовному з’єднанні двох лічильників називається молодшим?

65. Який лічильник при послідовному з’єднанні двох лічильників називається старшим?

66. В чому полягає різниця у способах послідовного з’єднання двох лічильників?

67. Яким чином необхідно підключати лічильники один до одного для використання в режимі підрахунку вхідних сигналів?

68. Яким чином необхідно підключати лічильники один до одного для використання в режимі ділення частоти?

69. В яких випадках необхідно підключати перетворювач кодів на виходах підсумкового лічильника?

70. Приведіть послідовність зміни станів підсумкового лічильника, якщо $M_{мол} = 2$; $M_{ст} = 5$, де $M_{мол}\,(M_{ст})$ – модуль ліку молодшого (старшого) лічильника.

71. Приведіть послідовність зміни станів підсумкового лічильника, якщо $M_{мол} = 5$; $M_{ст} = 2$, де $M_{мол}\,(M_{ст})$ – модуль ліку молодшого (старшого) лічильника.

72. Приведіть послідовність зміни станів підсумкового лічильника, якщо $M_{мол} = 3$; $M_{ст} = 5$, де $M_{мол}\,(M_{ст})$ – модуль ліку молодшого (старшого) лічильника.

73. Приведіть послідовність зміни станів підсумкового лічильника, якщо $M_{мол} = 5$; $M_{ст} = 3$, де $M_{мол}\,(M_{ст})$ – модуль ліку молодшого (старшого) лічильника.

74. Прокоментуйте способи з’єднання лічильників на рис.3.65.

75. В чому полягає різниці у способах з’єднання лічильників на рис.3.65,а і 3.65,б.

76. Яким чином утворена табл.3.12 для схеми на рис.3.65,а?

77. Яким чином утворена табл.3.12 для схеми на рис.3.65,б?

78. В якому режимі можна використовувати схему підсумкового лічильника на рис.3.66?

79. Прокоментуйте часову діаграму на рис.3.67.

80. Визначте модуль ліку лічильника по часовій діаграмі на рис.3.67.

81. В якому режимі можна використовувати схему підсумкового лічильника на рис.3.68?

82. Прокоментуйте часову діаграму на рис.3.69.

83. Визначте модуль ліку лічильника по часовій діаграмі на рис.3.69.

84. Яким чином створена табл.3.13?

85. Чому в деяких рядках виходи $y_3 - y_0$ позначені символом «*»?

86. Для синтезу якого пристрою призначена табл.3.13?

87. До якого класу пристроїв відноситься перетворювач кодів на виходах лічильника?

88. З якою метою використовується перетворювач кодів на виходах лічильника?

89. Виконайте синтез перетворювача кодів для підсумкового лічильника, який утворений базовими лічильниками $M_{мол} = 5$; $M_{ст} = 2$.

90. Виконайте синтез перетворювача кодів для підсумкового лічильника, який утворений базовими лічильниками $M_{мол} = 5$; $M_{ст} = 3$.

91. В якому режимі можна використовувати схему підсумкового лічильника на рис.3.72?

92. Назвіть призначення складових вузлів лічильника, схема якого приведена на рис.3.72?

93. Прокоментуйте часову діаграму на рис.3.73.

94. Визначте модуль ліку лічильника по часовій діаграмі на рис.3.73.

95. Яким чином не виконуючи синтез базових лічильників, можна побудувати схему підсумкового лічильника з $M = 10$ на основі послідовного з'єднання базових лічильників в режимі підрахунку вхідних сигналів?

96. Яким чином не виконуючи синтез базових лічильників, можна побудувати схему підсумкового лічильника з *M = 10* на основі послідовного з'єднання базових лічильників в режимі ділення частоти?

97. Яким чином не виконуючи синтез базових лічильників, можна побудувати схему підсумкового лічильника з *M = 24* на основі послідовного з'єднання базових лічильників в режимі підрахунку вхідних сигналів?

98. Яким чином не виконуючи синтез базових лічильників, можна побудувати схему підсумкового лічильника з *M = 24* на основі послідовного з'єднання базових лічильників в режимі ділення частоти?

99. Поясніть принцип функціонування підсумовувального лічильника з асинхронним встановленням початкового стану, структурна схема якого приведена на рис.3.74.

100. Назвіть призначення складових елементів лічильника на структурній схемі (рис.3.74).

101. Поясніть принцип функціонування віднімального лічильника з асинхронним встановленням початкового стану, структурна схема якого приведена на рис.3.74.

102. Які мікрооперації виконує лічильник з асинхронним встановленням початкового стану?

103. Поясніть призначення і принцип роботи *СВПС* в лічильниках з асинхронним встановленням початкового стану.

104. Який тип базового лічильника використовується в лічильниках з асинхронним встановленням початкового стану?

105. Як визначається код початкового стану в підсумовувальних лічильниках з асинхронним встановленням початкового стану?

106. Як визначається код початкового стану у віднімальних лічильниках з асинхронним встановленням початкового стану?

107. Чи є *СВПС* комбінаційною схемою? Обґрунтуйте відповідь.

108. Для чого використовуються асинхронні входи базових тригерів в лічильниках з асинхронним встановленням початкового стану?

109. Приведіть послідовність дій для виконання синтезу лічильників з асинхронним встановленням початкового стану.

110. Як визначити кількість тригерів в лічильниках з асинхронним встановленням початкового стану?

111. Яким чином виконується синтез *СВПС*?

112. Поясніть заповнення таблиці 3.14.

113. Чому в деяких рядках таблиці 3.14 в якості значення сигналу *nIS* використовуються символи «*»?

114. Поясніть призначення сигналу *nIS*.

115. В яких випадках активним значенням сигналу *nIS* (*IS*) є логічний нуль (одиниця)?

116. Виконайте синтез підсумовувального лічильника за модулем *10* з асинхронним встановленням початкового стану на асинхронних *T*-тригерах.

117. Виконайте синтез підсумовувального лічильника за модулем *10* з асинхронним встановленням початкового стану на синхронних *TC*- тригерах.

118. Виконайте синтез віднімального лічильника за модулем *10* з асинхронним встановленням початкового стану на асинхронних *T*-тригерах.

119. Виконайте синтез віднімального лічильника за модулем *10* з асинхронним встановленням початкового стану на синхронних *TC*- тригерах.

120. Прокоментуйте функціонування лічильника, схема якого приведена на рис.3.76.

121. Прокоментуйте часову діаграму на рис.3.78.

122. Визначте модуль ліку лічильника по часовій діаграмі на рис.3.78.

123. Чому в лічильнику на рис.3.77 використовуються *JK*-тригери?

124. Поясніть порядок переключення логічних елементів на часовій діаграмі (рис.3.79).

125. Чому тривалість останнього стану перед переключенням в початковий стан є дуже малою?

126. Як визначити тривалість останнього стану лічильника з асинхронним встановленням початкового стану?

127. В яких випадках лічильник з асинхронним встановленням початкового стану може працювати некоректно?

128. Поясніть ситуацію, при якій лічильник з асинхронним встановленням початкового стану передчасно переключається в цей початковий стан.

129. Поясніть ситуацію, при якій лічильник з асинхронним встановленням початкового стану може переключитися в некоректний початковий стан.

130. Чому лічильники з асинхронним встановленням початкового стану можуть бути ненадійними?

131. Прокоментуйте схему лічильника на рис.3.80.

132. Яким чином промодельована затримка переключення тригера Q_0 в схемі на рис.3.80?

133. Прокоментуйте часову діаграму на рис.3.81.

134. Визначте модуль ліку лічильника по часовій діаграмі на рис.3.81.

135. Поясніть, чому лічильник на рис.3.80 працює неправильно?

136. Вкажіть на часовій діаграмі (рис.3.81) місце, де лічильник спрацьовує неправильно.

137. Прокоментуйте часову діаграму на рис.3.82.

138. Яким чином виконується синтез віднімальних лічильників з асинхронним встановленням початкового стану?

139. Прокоментуйте схему лічильника на рис.3.83.

140. Прокоментуйте часову діаграму на рис.3.84.

141. Визначте модуль ліку лічильника по часовій діаграмі на рис.3.84.

142. Прокоментуйте послідовність переключення логічних елементів лічильника на рис.3.85.

143. Як визначити модуль ліку в лічильниках з асинхронним встановленням початкового стану відповідно до кількості станів цього лічильника.

144. Як визначити модуль ліку в лічильниках з асинхронним встановленням початкового стану з точки зору ділення частоти вихідного сигналу переносу зі старшого розряду?

145. Поясніть принцип функціонування підсумовувального лічильника з синхронним встановленням початкового стану, структурна схема якого приведена на рис.3.74.

146. В чому полягає різниця між лічильниками з синхронним і асинхронним встановленням початкового стану?

147. Поясніть принцип функціонування *СВПС* на рис.3.86.

148. Назвіть призначення складових елементів структурно-функціональної схеми на рис.3.86.

149. Яку функцію виконує *СФІS*?

150. Яку функцію виконує додатковий *D*-тригер в лічильнику з синхронним встановленням початкового стану?

151. Яким чином використовується сигнал Q_{IS}?

152. Приведіть порядок синтезу лічильника з синхронним встановленням початкового стану.

153. Поясніть принцип функціонування віднімального лічильника з синхронним встановленням початкового стану.

154. Поясніть принцип функціонування підсумовувального лічильника з синхронним встановленням початкового стану.

155. Які мікрооперації виконує лічильник з синхронним встановленням початкового стану?

156. Як визначається код початкового стану в підсумовувальних лічильниках з синхронним встановленням початкового стану?

157. Як визначається код початкового стану у віднімальних лічильниках з синхронним встановленням початкового стану?

158. Чи є СВПС комбінаційною схемою в лічильниках з синхронним встановленням початкового стан? Обґрунтуйте відповідь.

159. Для чого використовуються асинхронні входи базових тригерів в лічильниках з синхронним встановленням початкового стану?

160. Як визначити кількість тригерів в лічильниках з синхронним встановленням початкового стану?

161. Яким чином виконується синтез *СФІS*?

162. Поясніть заповнення таблиці 3.15.

163. Чому в деяких рядках таблиці 3.15 в якості значення сигналу *IS* використовуються символи «*»?

164. Поясніть призначення сигналу *IS*.

165. В яких випадках активним значенням сигналу *IS* є логічний нуль або одиниця?

166. Виконайте синтез підсумовувального лічильника за модулем 10 з синхронним встановленням початкового стану на асинхронних Т-тригерах.

167. Виконайте синтез підсумовувального лічильника за модулем 10 з синхронним встановленням початкового стану на синхронних ТС- тригерах.

168. Виконайте синтез віднімального лічильника за модулем 10 з синхронним встановленням початкового стану на асинхронних Т-тригерах.

169. Виконайте синтез віднімального лічильника за модулем 10 з синхронним встановленням початкового стану на синхронних ТС- тригерах.

170. Як отримати логічний вираз для формування сигналу *IS*?

171. Прокоментуйте функціонування лічильника, схема якого приведена на рис.3.88.

172. Поясніть функціонування лічильника на рис.3.89.

173. Прокоментуйте часову діаграму на рис.3.90.

174. Визначте модуль ліку лічильника по часовій діаграмі на рис.3.90.

175. Чому в лічильнику на рис.3.89 використовуються JK-тригери?

176. Поясніть порядок переключення логічних елементів на часовій діаграмі (рис.3.91).

177. Чому лічильники з синхронним встановленням початкового стану є більш надійними, ніж лічильники з асинхронним встановленням початкового стану?

178. Яким чином виконується синтез віднімальних лічильників з синхронним встановленням початкового стану?

179. Чи можна в складі лічильника використовувати прозорі тригери? Обґрунтуйте відповідь.

180. Чи можна в складі лічильника використовувати двоступеневі тригери? Обґрунтуйте відповідь.

181. Чи можна в складі лічильника використовувати тригери з динамічним керуванням? Обґрунтуйте відповідь.

3.3 Лічильники на базі лічильної схеми

Структурна схема лічильника на базі лічильної схеми приведена на рис.3.92.

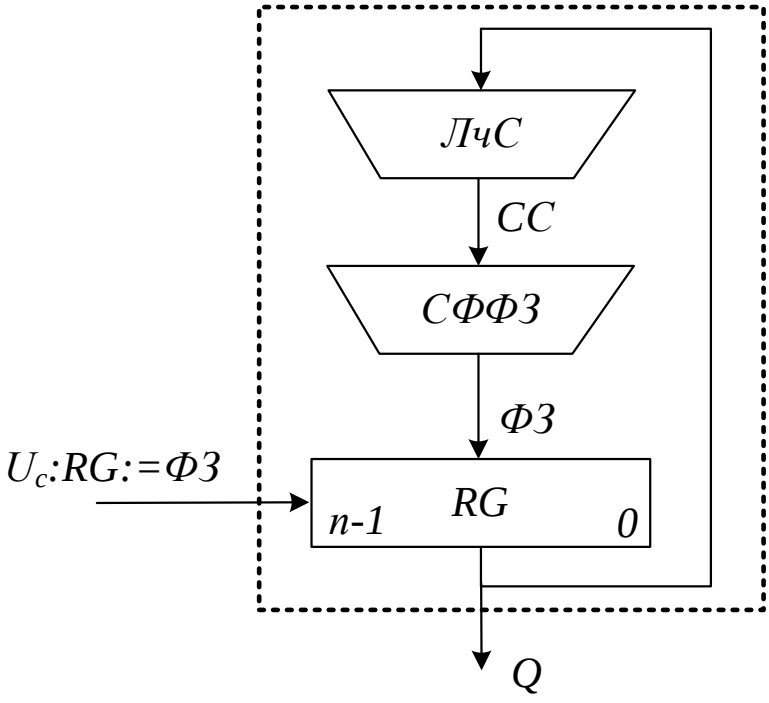


Рисунок 3.92 – Структурна схема лічильника на базі лічильної схеми

На структурній схемі використовуються такі позначення:

- U_c – лічильний сигнал;

- *ЛчС* – лічильна схема (***counting circuit***);

- *СС* – вихідні сигнали лічильної схеми;

- *СФФЗ* – схема формування функцій збудження тригерів регістра;

- *ФЗ* – функції збудження базових тригерів регістра;

- *RG* – регістр;

- *Q* – вихідні сигнали лічильника.

Відповідно до структурної схеми до складу цього лічильника входять звичайний *n*-розрядний регістр *RG*, схема формування функцій збудження базових тригерів регістра *СФФЗ* та комбінаційна лічильна схема *ЛчС*.

Розрядність регістра визначається розрядністю лічильника. Регістр за активним фронтом вхідного сигналу U_c виконує завантаження інформації з виходу лічильної схеми через *СФФЗ*, тобто виконує мікрооперацію *RG := ФЗ*.

СФФЗ забезпечує завантаження інформації з виходів лічильної схеми в базові тригери регістра в залежності від типу цих тригерів та детально описана в розділі 2 цього посібника.

Лічильники на базі лічильної схеми, як правило, входять до складу лічильників з паралельним завантаженням інформації або до складу регістра, який окрім звичайного завантаження інформації із зовнішніх джерел та зсувів, може виконувати функції підрахунку. В останньому випадку це може бути регістри арифметичних пристроїв, що перетворюють прямі коди представлення даних в доповняльний код або навпаки.

В лічильниках на базі лічильної схеми базовими тригерами, як правило, є *D*-, *RCS*- або *JK*-тригери. Нагадаємо, що при використанні *D*-тригерів схема формування функцій збудження базових тригерів регістра відсутня і виходи лічильної схеми безпосередньо під’єднані до інформаційних входів базових тригерів.

Лічильна схема *ЛчС*, як правило, виконує мікрооперацію «інкремент» стану лічильника (*CC := Q + 1*) для підсумовувальних лічильників або

мікрооперацію «декремент» стану лічильника (*СС* := *Q* - *1*) для віднімальних лічильників.

Принцип функціонування лічильника полягає в тому, що вихідний стан регістра *Q* надходить на входи лічильної схеми *ЛчС*, на виході якої формується двійковий код *СС* відповідно до виразу *СС* := *Q* + *1*. За активним фронтом лічильного сигналу U_c регістр приймає цей новий код і далі все повторюється.

Синтез лічильника полягає в проведенні синтезу складових вузлів: регістра і лічильної схеми. Синтез регістрів детально розглядався в розділі 2 посібника.

Для побудови лічильника використовується два типи лічильних схем:

- лічильні схеми з паралельним трактом розповсюдження переносу (паралельна лічильна схема);

- лічильні схеми з послідовним трактом розповсюдження переносу (послідовна лічильна схема).

Проєктування паралельної лічильної схеми виконується за допомогою класичного методу (з використанням таблиці істинності).

Розглянемо синтез лічильника на базі паралельної лічильної схеми на прикладі.

Приклад 3.7. Виконати синтез підсумовувального лічильника за модулем *8* на основі *D*-тригерів на базі лічильної схеми .

Розв’язок

За аналогією з прикладами 3.5 і 3.6 спочатку визначаємо, що кількість базових тригерів заданого лічильника відповідно до виразу (3.1) складає: $n = \lceil \log_2 M \rceil = \lceil \log_2 8 \rceil = 3$, отже будемо використовувати 3-розрядний регістр. В зв’язку з тим, що базовими тригерами є *D*-тригери, то *СФФЗ* відсутня, тобто виходи лічильної схеми безпосередньо підключаються до входів *D* базових тригерів регістра.

Розглянемо синтез лічильної схеми.

Таблиця істинності лічильної схеми приведена в табл.3.16. Двійковий код вихідного стану лічильної схеми (колонки 5-7) на одиницю більше коду вихідного стану регістра (колонки 2-4), тобто в рядках 0-6 таблиці вихідний стан *ЛчС* відповідає виразу $CC := Q + 1$. Рядок 7 в таблиці відповідає переключенню лічильника зі стану *M-1* (111_2) в код *000*.

Таблиця 3.16 – Таблиця істинності лічильної схеми

Номери наборів	Стан регістра			Вихідний стан лічильної схеми		
	Q_2^t	Q_1^t	Q_0^t	CC_2	CC_1	CC_0
1	2	3	4	5	6	7
0	*0*	*0*	*0*	*0*	*0*	*1*
1	*0*	*0*	*1*	*0*	*1*	*0*
2	*0*	*1*	*0*	*0*	*1*	*1*
3	*0*	*1*	*1*	*1*	*0*	*0*
4	*1*	*0*	*0*	*1*	*0*	*1*
5	*1*	*0*	*1*	*1*	*1*	*0*
6	*1*	*1*	*0*	*1*	*1*	*1*
7	*1*	*1*	*1*	*0*	*0*	*0*

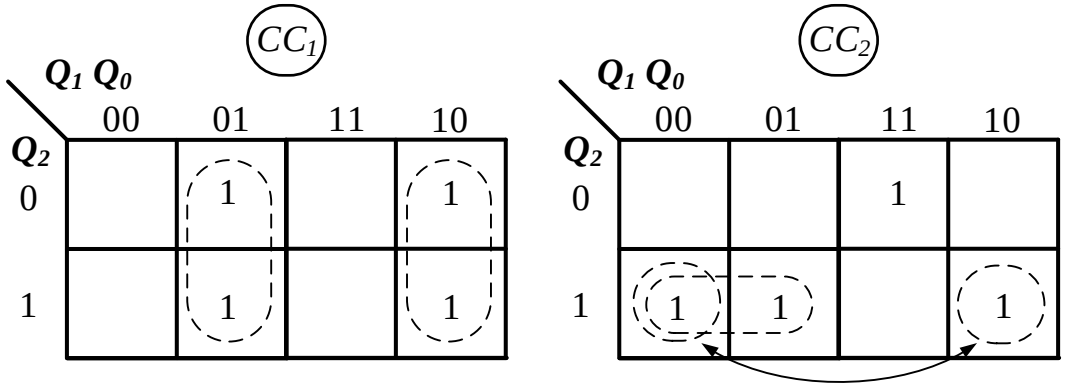


Рисунок 3.93 – Карти Карно для мінімізації CC_1 і CC_2

З таблиці 3.16 можна побачити, що значення наймолодшого виходу лічильної схеми CC_0 протилежне вхідному коду цієї схеми, що надходить з виходу регістра Q_0, тобто можна записати $CC_0 = \overline{Q_0}$.

Далі виконаємо мінімізацію логічних виразів для виходів лічильної схеми CC_1 і CC_2 з використанням карт Карно, які показані на рис.3.93.

В результаті мінімізації можна отримати:

$$CC_1 = Q_1 \cdot \overline{Q_0} \vee Q_0 \cdot \overline{Q_1} = Q_1 \oplus Q_0; \tag{3.21}$$

$$\begin{aligned} CC_2 &= Q_2 \cdot \overline{Q_1} \vee Q_2 \cdot \overline{Q_0} \vee \overline{Q_2} \cdot Q_1 \cdot Q_0 = Q_2 \cdot (\overline{Q_1} \vee \overline{Q_0}) \vee \overline{Q_2} \cdot Q_1 \cdot Q_0 = \\ &= Q_2 \cdot (\overline{Q_1 \cdot Q_0}) \vee \overline{Q_2} \cdot (Q_1 \cdot Q_0) = Q_2 \oplus Q_1 \cdot Q_0. \end{aligned} \tag{3.22}$$

Функціональна схема підсумовувального лічильника за модулем *6* на базі лічильної схеми приведена на рис.3.94.

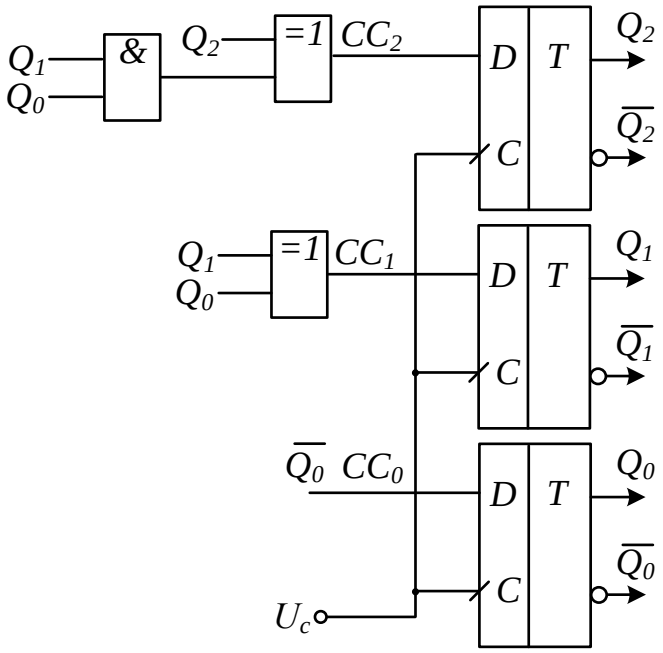


Рисунок 3.94 – Функціональна схема лічильника на базі лічильної схеми (приклад 3.7)

Схема для моделювання функціонування лічильника приведена на рис.3.95, а результати моделювання у вигляді часової діаграми функціонування цього пристрою – на рис.3.96.

Сигнал переносу зі старшого розряду лічильника формується на виході старшого тригера. Час переключення лічильника визначається часом спрацьовування регістра $t_{\text{п}} = t_{RG}$.

Для правильного завантаження інформації в регістр на момент активного фронту сигналу U_c перехідні процеси в лічильній схемі повинні закінчитися, тобто мінімальний період лічильного сигналу, який використовується для розрахунку максимальної частоти надходження

сигналу U_c, визначається за виразом $T_{Ucmin} = t_{CC} + t_{RG}$, де t_{CC}, t_{RG} – відповідно час затримки спрацьовування лічильної схеми та регістра.

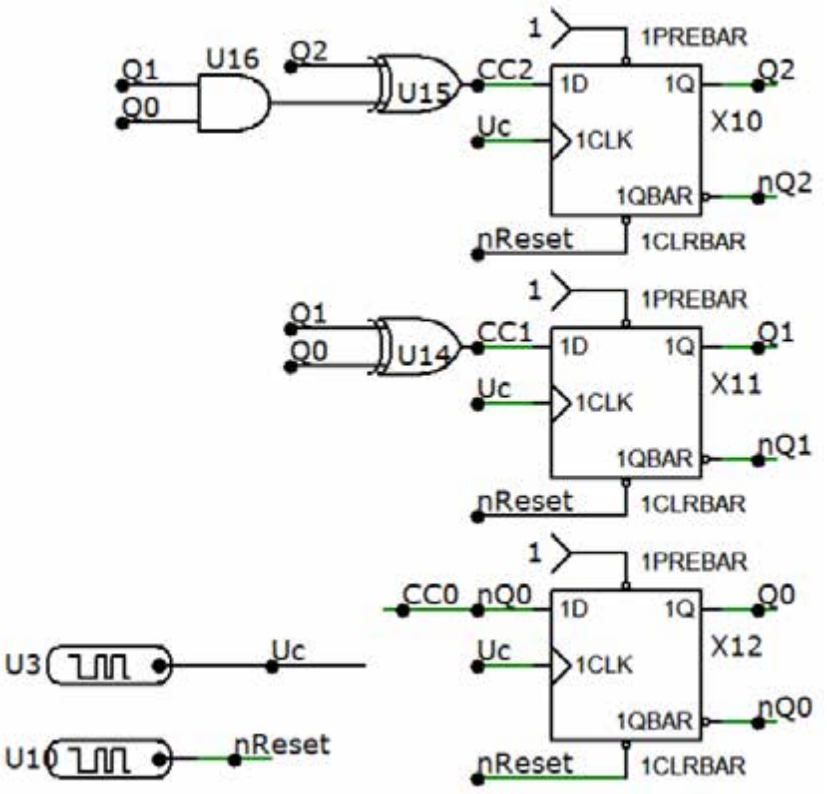


Рисунок 3.95 – Схема для моделювання лічильника на базі лічильної схеми (приклад 3.7)

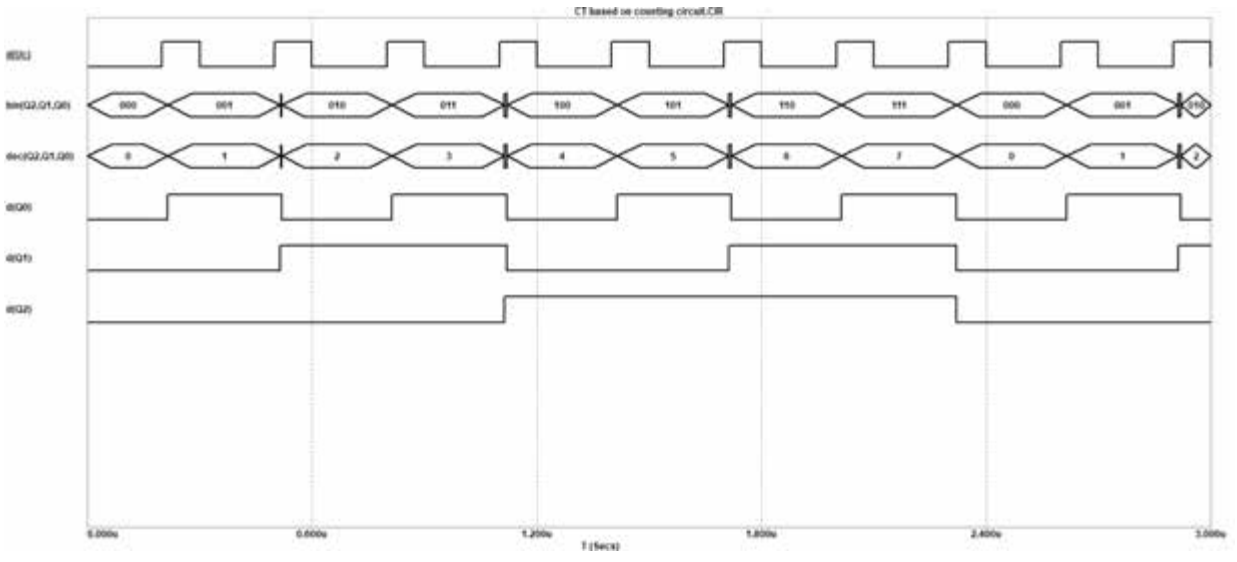


Рисунок 3.96 – Результати моделювання лічильника на базі лічильної схеми (приклад 3.7)

На цьому виконання прикладу 3.7 завершено.

В результаті синтезу реалізовано двійковий підсумовувальний лічильник на базі паралельної лічильної схеми, тобто сигнали CC_i формуються фактично паралельно. Розглянемо, яким чином можна збільшувати розрядність таких лічильників. Наприклад, синтез підсумовувального лічильника за модулем *16* також може проведений за допомогою таблиці істинності, аналогічній табл.3.16, але використання цього методу обмежено обсягом таблиці істинності та при використанні ручного способу мінімізації обмежується шістьма розрядами лічильника. Для спроби отримати загальний вираз для будь-якого розряду лічильника без проведення процедури синтезу, розглянемо логічні вирази (3.21), (3.22). Аналізуючи ці вирази, можна зробити припущення, що логічна формула для третього розряду лічильної схеми CC_3 відповідає такому виразу:

$$CC_3 = Q_3 \oplus Q_2 \cdot Q_1 \cdot Q_0. \qquad (3.23)$$

На рис.3.97 приведена схема для моделювання підсумовувального лічильника за модулем *16* на базі лічильної схеми, а на рис.3.98 – результати моделювання цього лічильника, на яких можна побачити, що лічильник дійсно має модуль ліку *16*.

Таким чином, отримаємо загальну формулу для реалізації вихідного сигналу будь-якого розряду лічильної схеми:

$$CC_i = Q_i \oplus Q_{i-1} \cdot Q_{i-2} \cdot \ldots \cdot Q_0. \qquad (3.24)$$

З виразу (3.24) можна побачити, що зі збільшенням розрядності лічильної схеми відбувається ускладнення логічної формули для кожного нового розряду.

Далі розглянемо синтез лічильника на базі послідовної лічильної схеми. Для цього представимо формули для реалізації лічильної схеми в такому вигляді:

$$\begin{aligned} CC_0 &= \overline{Q_0} = Q_0 \oplus 1; \\ CC_1 &= Q_1 \oplus Q_0 \cdot 1; \end{aligned} \qquad (3.25)$$

$CC_2 = Q_2 \oplus Q_1 \cdot Q_0 \cdot 1$;

$CC_3 = Q_3 \oplus Q_2 \cdot Q_1 \cdot Q_0 \cdot 1$;

…

$CC_i = Q_i \oplus Q_{i-1} \cdot Q_{i-2} \cdot \ldots \cdot Q_0 \cdot 1$.

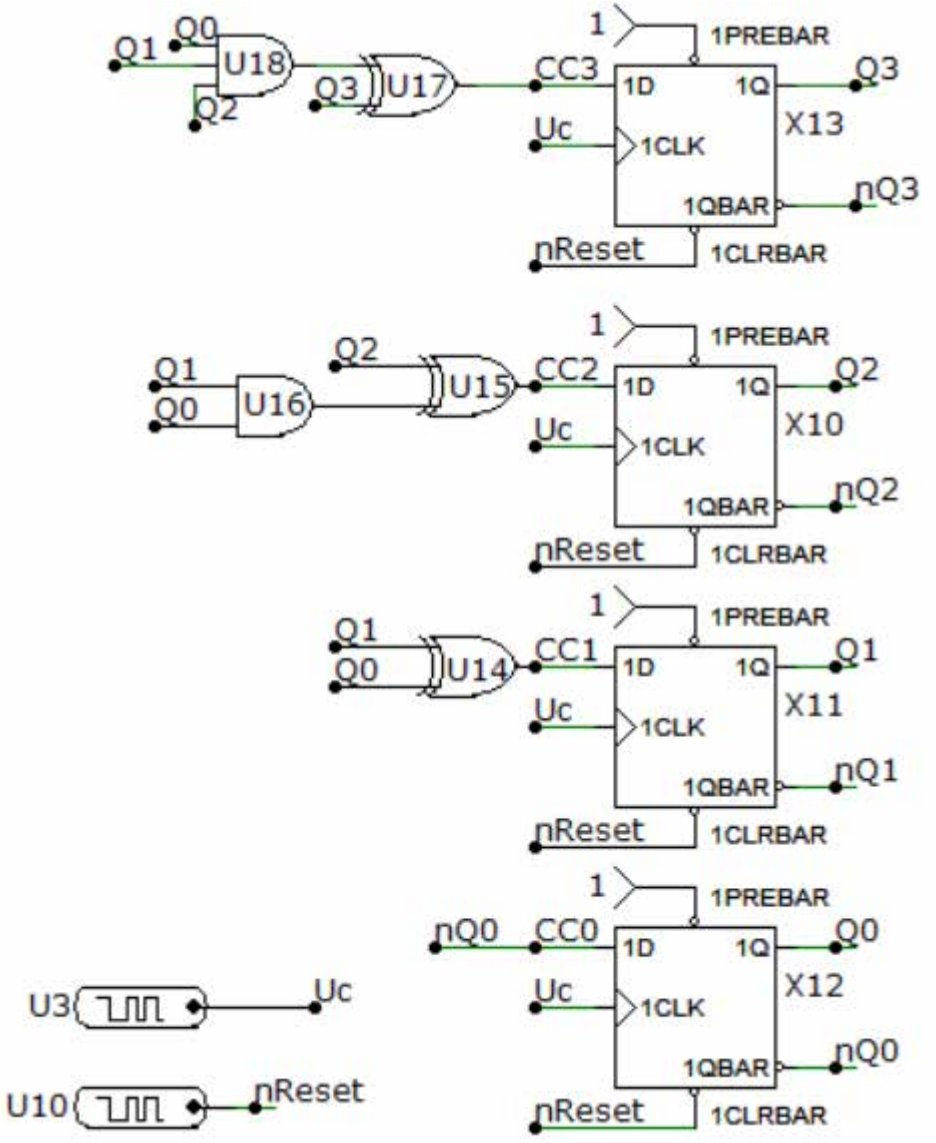


Рисунок 3.97 – Схема для моделювання підсумовувального лічильника за модулем *16* на базі лічильної схеми

З виразів (3.25) можна побачити, що до частини другого операнду функції додавання за модулем два входить другий операнд попередньої формули.

Введемо позначення $C_{in0} = 1$. Тоді вирази (3.25) можна записати таким чином:

$$C_{in0} = 1; \quad C_{in1} = Q_0 \cdot C_{in0}; \quad C_{in2} = Q_1 \cdot C_{in1};$$
$$C_{in3} = Q_2 \cdot C_{in2}; \quad \ldots \quad C_{in\,i} = Q_{i-1} \cdot C_{in\,i-1}. \qquad (3.26)$$

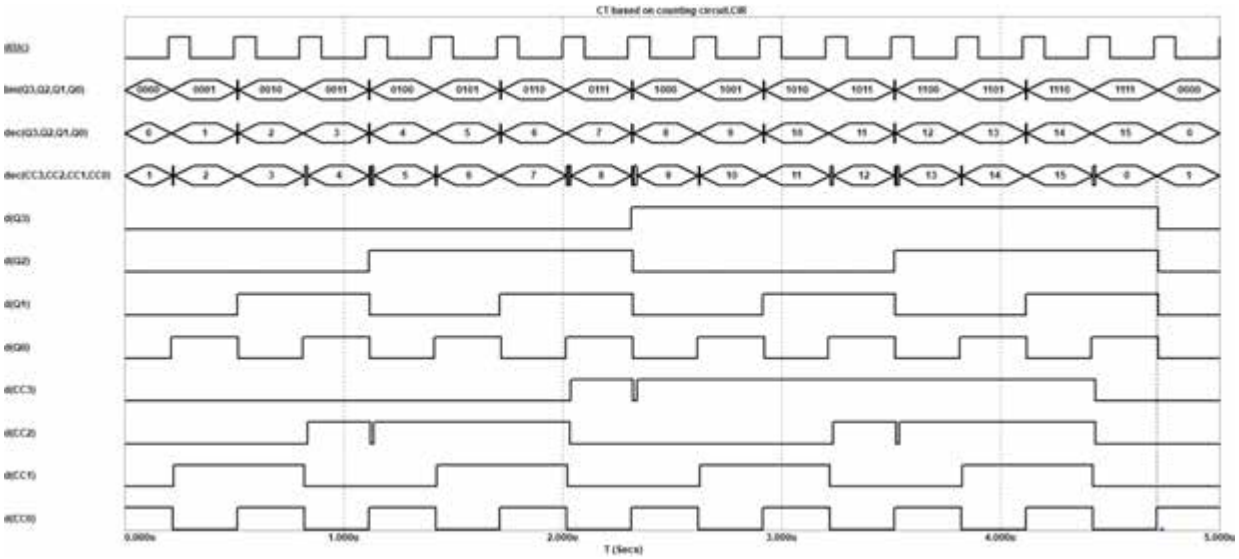


Рисунок 3.98 – Результати моделювання підсумовувального лічильника за модулем *16* на базі паралельної лічильної схеми

Отже, вирази (3.25) з урахуванням (3.26) приймають вигляд:

$$CC_0 = Q_0 \oplus C_{in0}; \quad CC_1 = Q_1 \oplus C_{in1}; \quad CC_2 = Q_2 \oplus C_{in2};$$
$$CC_3 = Q_3 \oplus C_{in1}; \quad \ldots \quad CC_i = Q_i \oplus C_{in\,i}. \qquad (3.27)$$

На рис.3.99 приведена схема для моделювання підсумовувального лічильника за модулем *16* на базі послідовної лічильної схеми, а на рис.3.100 – результати моделювання цього лічильника. Звичайно, що в реальних схемах сигнал CC_0 реалізується так, як показано на рис.3.97.

Послідовна лічильна схема може бути також реалізована за допомогою напівсуматорів (***half-adder***) для підсумовувальних лічильників або напіввіднімачів (***half-subtractor***) для віднімальних лічильників.

Напівсуматори і напіввіднімачі є комбінаційними пристроями з двома входами і виходами [2]. Таблиця істинності напівсуматора і напіввіднімача приведена в табл.3.17.

В табл.3.17 використовуються такі позначення:

- a – інформаційний вхід для підключення операнду;
- C_{in}, C_{out} – вхідний і вихідний переноси (***input and output carry***);

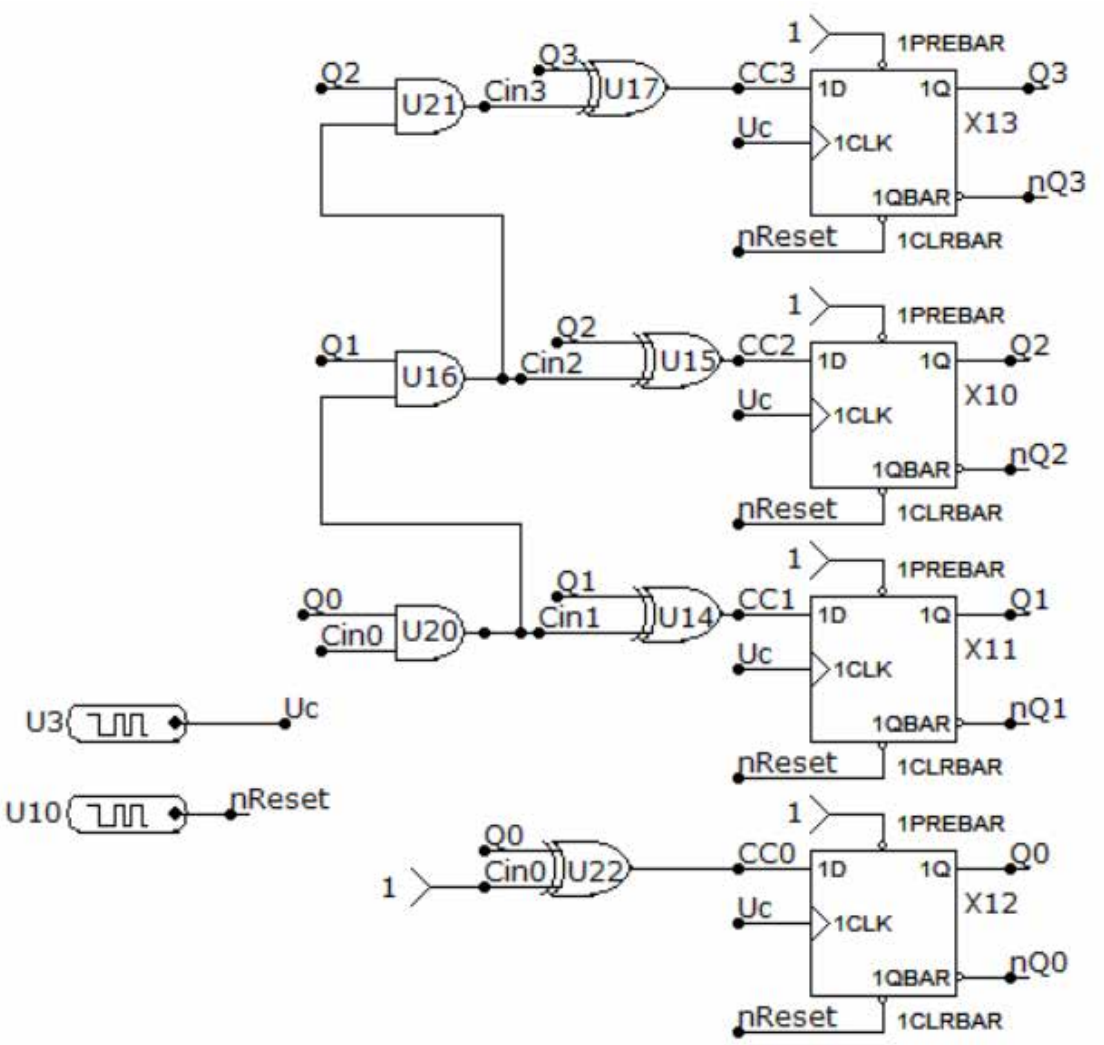


Рисунок 3.99 – Схема для моделювання підсумовувального лічильника за модулем *16* на базі послідовної лічильної схеми

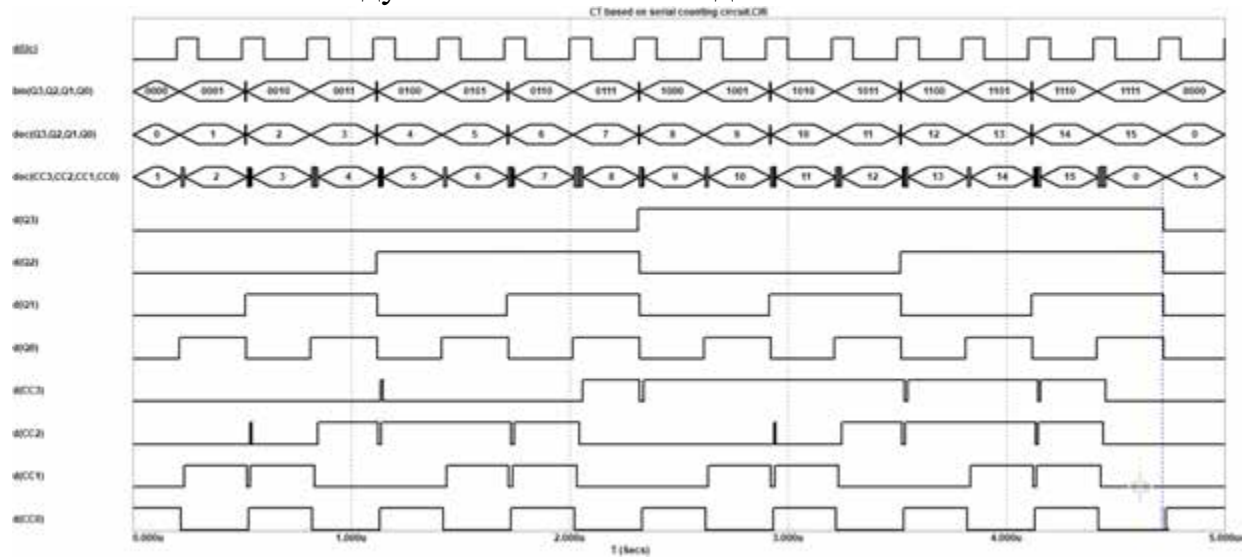


Рисунок 3.100 – Результати моделювання підсумовувального лічильника за модулем *16* на базі послідовної лічильної схеми

- B_{in}, B_{out} – вхідна і вихідна позики (***input and output borrow***);

HS – напівсума (***half-sum***);

HD – напіврізниця (***half-difference***).

Таблиця 3.17 – Таблиця істинності напівсуматора і напіввіднімача

Напівсуматор			
a	C_{in}	HS	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Напіввіднімач			
a	B_{in}	HD	B_{out}
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Функціонування цих пристроїв описується логічними виразами:

$$HS = a \cdot \overline{C_{in}} \vee \overline{a} \cdot C_{in} = \overline{a \cdot C_{in} \vee \overline{a} \cdot \overline{C_{in}}} = a \oplus C_{in};$$

$$HD = a \cdot \overline{B_{in}} \vee \overline{a} \cdot B_{in} = \overline{a \cdot B_{in} \vee \overline{a} \cdot \overline{B_{in}}} = a \oplus B_{in}; \qquad (3.28)$$

$$C_{out} = a \cdot C_{in}; \qquad B_{out} = \overline{a} \cdot B_{in}.$$

Відповідно до табл.3.17 та (3.28) логічні вирази для напівсуми і напіврізниці є однаковими, а вирази для вихідного переносу і позики відрізняються взаємно інверсними значеннями операнду, що може бути використано для апаратної реалізації комбінованого пристрою, який виконує як функції напівсуматора, так і напіввіднімача. Схема однорозрядного напівсуматора з використанням функції додавання за модулем два приведена на рис.3.101.

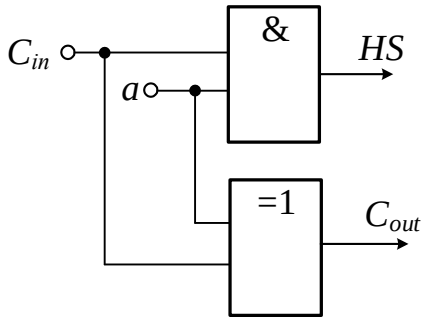


Рисунок 3.101 – Логічна схема однорозрядного напівсуматора

Розглянемо послідовне з'єднання двох напівсуматорів, де вихідний перенос одного напівсуматорів з'єднується зі вхідним переносом наступного напівсуматора (рис.3.102).

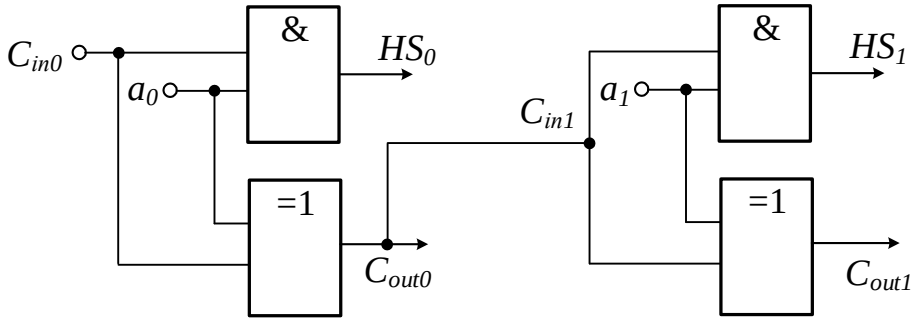


Рисунок 3.102 – Логічна схема дворозрядного напівсуматора

Відповідно до (3.28) отримаємо:

$$HS_0 = a_0 \oplus C_{in0};\quad C_{out0} = a_0 \cdot C_{in0};\qquad C_{in1} = C_{out0};$$
$$HS_1 = a_1 \oplus C_{in1} = a_1 \oplus C_{out0} = a_1 \oplus a_0 \cdot C_{in0}; \qquad (3.29)$$
$$C_{out1} = a_1 \cdot C_{in1} = a_1 \cdot C_{out0} = a_1 \cdot a_0 \cdot C_{in0}\ .$$

Аналізуючи вирази (3.29) та функціональну схему лічильника з послідовною лічильною схемою на рис.3.99, можна зробити висновок, що ця схема фактично побудована з використання напівсуматорів, де в якості операндів напівсуматора a_i використовуються виходи регістра Q_i. В схемі на рис.3.99 для побудови молодшого розряду використовується перетворення $C_{out0} = a_0 \cdot C_{in0} = a_0 \cdot 1 = a_0$, де $a_0 = Q_0$.

Таким чином, для при побудові лічильників на базі послідовної лічильної схеми в якості цієї схеми можна використовувати послідовне з'єднання напівсуматорів або за наявності в цифровому пристрої повного суматора можна використовувати цей суматор як послідовну лічильну схему, задаючи нульові коди на входи одного з доданків.

Порівнюючи розглянуті вище способи синтезу лічильників на базі лічильної схеми, необхідно зазначити наступне:

- на основі паралельної лічильної схеми можуть бути побудовані лічильники з будь-яким модулем ліку, але при цьому є обмеження на

кількість розрядів цієї схеми при використання ручного способу мінімізації логічних функції, що описують функціонування лічильної схеми;

- на основі послідовної лічильної схеми в основному будуються тільки двійкові лічильники.

Контрольні завдання та запитання

1. Поясніть принцип роботи лічильника на базі лічильної схеми.
2. Приведіть структурну схему лічильника на базі лічильної схеми.
3. Поясніть призначення складових елементів лічильника на базі лічильної схеми.
4. Для чого в складі лічильника на базі лічильної схеми використовується регістр?
5. До якого класу цифрових вузлів відноситься лічильна схема?
6. Яким чином виконується синтез регістра в складі лічильника на базі лічильної схеми?
7. Чи можна в складі лічильника використовувати прозорі тригери? Обґрунтуйте відповідь.
8. Як визначити розрядність регістра в складі лічильника на базі лічильної схеми?
9. Поясніть принцип роботи лічильної схеми.
10. Назвіть основні типи лічильних схем.
11. В чому полягає відмінність у функціонуванні паралельної і послідовної лічильної схеми?
12. Як виконується синтез паралельної лічильної схеми?
13. Охарактеризуйте класичний метод синтезу лічильної схеми.
14. Приведіть таблицю переходів для синтезу лічильної схеми підсумовувального лічильника за модулем *5*.
15. Приведіть таблицю переходів для синтезу лічильної схеми підсумовувального лічильника за модулем *10*.

16. Приведіть таблицю переходів для синтезу лічильної схеми віднімального лічильника за модулем *5*.

17. Приведіть таблицю переходів для синтезу лічильної схеми віднімального лічильника за модулем *8*.

18. Приведіть таблицю переходів для синтезу лічильної схеми віднімального лічильника за модулем *10*.

19. Приведіть послідовність зміни станів підсумовувального десяткового лічильника.

20. Приведіть послідовність зміни станів віднімального десяткового лічильника.

21. Приведіть послідовність зміни станів підсумовувального лічильника за модулем *6*.

22. Приведіть послідовність зміни станів віднімального лічильника за модулем *6*.

23. Поясніть, яким чином заповнюється таблиця 3.17.

24. Як позначаються в таблиці істинності неіснуючі стани лічильника?

25. Як отримано вираз (3.21)?

26. Як отримано вираз (3.22)?

27. Яким чином побудовано схему на рис.3.94?

28. Яким чином побудовано схему на рис.3.95?

29. З якою метою в схемі лічильника на рис.3.95 використовується сигнал ***nReset***?

30. Де формується сигнал переносу зі старшого розряду в лічильниках на базі лічильної схеми?

31. Прокоментуйте часову діаграму на рис.3.96.

32. Визначте модуль ліку лічильника по часовій діаграмі на рис.3.96.

33. Як визначити час переключення лічильника, схема якого приведена на рис.3.94?

34. Як визначити максимальну частоту надходження сигналу синхронізації лічильника, схема якого приведена на рис.3.94?

35. Прокоментуйте часову діаграму на рис.3.59.

36. Як збільшити розрядність двійкового лічильника на базі лічильної схеми?

37. Яким чином отримано вираз (3.23)?

38. Яким чином побудовано схему на рис.3.97?

39. Прокоментуйте часову діаграму на рис.3.98.

40. Яким чином отримано вираз (3.24)?

41. Визначте модуль ліку лічильника по часовій діаграмі на рис.3.98.

42. Прокоментуйте вираз (3.25).

43. Яким чином отримано вирази (3.26)?

44. Яким чином отримано вирази (3.27)?

45. Яким чином побудовано схему на рис.3.99?

46. Прокоментуйте часову діаграму на рис.3.100.

47. Поясніть принцип роботи напівсуматора.

48. Поясніть принцип роботи напіввіднімача.

49. До якого класу цифрових вузлів відноситься напівсуматор?

50. До якого класу цифрових вузлів відноситься напіввіднімач?

51. Поясніть таблицю істинності напівсуматора.

52. Поясніть таблицю істинності напіввіднімача.

53. Яким чином отримано вирази (3.28)?

54. Яку мікрооперацію виконує напівсуматор?

55. Яку мікрооперацію виконує напіввіднімач?

56. Як побудувати комбінований пристрій «напівсуматор-напіввіднімач»?

57. Синтезуйте напівсуматор в булевому базисі.

58. Синтезуйте напіввіднімач в булевому базисі.

59. Синтезуйте напівсуматор в базисі Шефера.

60. Синтезуйте напіввіднімач в базисі Шефера.

61. Синтезуйте напівсуматор в базисі Пірса.

62. Синтезуйте напіввіднімач в базисі Пірса.

63. Синтезуйте напівсуматор з використанням елементів *XOR*.

64. Синтезуйте напіввіднімач з використанням елементів *XOR*.

65. Для чого використовується послідовне з'єднання напівсуматорів?

66. Для чого використовується послідовне з'єднання напіввіднімачів?

67. Поясніть принцип роботи схеми на рис.3.101.

68. Прокоментуйте вирази (3.29).

69. Побудуйте підсумовувальний лічильник за модулем 32 на базі лічильної схеми з використанням напівсуматорів.

70. Побудуйте віднімальний лічильник за модулем 32 на базі лічильної схеми з використанням напіввіднімачів.

71. Які переваги паралельної лічильної схеми?

72. Які недоліки паралельної лічильної схеми?

73. Які переваги послідовної лічильної схеми?

74. Які недоліки послідовної лічильної схеми?

75. Який тип переносу між розрядами лічильника можна отримати за допомогою паралельної лічильної схеми?

76. Який тип переносу між розрядами лічильника можна отримати за допомогою послідовної лічильної схеми?

3.4 Багатофункціональні лічильники

3.4.1 Реверсивні лічильники

Реверсивні лічильники, як правило, виконують мікрооперації «інкремент» (***Up Counter***) і «декремент» (***Down Counter***) за активними значеннями відповідних керуючих сигналів U_+ і U_-. Синтез таких лічильників аналогічний синтезу багатофункціональних регістрів на базі окремих тригерів та полягає у виконанні наступних кроків:

1. Визначення кількості базових тригерів.

2. Визначення функцій збудження базових тригерів підсумовувального лічильника.

3. Визначення функцій збудження базових тригерів віднімального лічильника.

4. Розробка комутатора реверсивного лічильника.

5. Визначення функцій збудження базових тригерів реверсивного лічильника.

6. Перетворення логічних виразів функцій збудження в заданий базис логічних елементів.

7. Розробка функціональної схеми реверсивного лічильника.

8. Перевірка коректності функціонування реверсивного лічильника.

В загальному випадку модулі ліку підсумовувального M_+ і віднімального M_- лічильників реверсивного лічильника можуть відрізнятися один від іншого, але, як правило, на практиці використовуються реверсивні лічильники з однаковими модулями ліку $M_+ = M_-$.

Для визначення кількості базових тригерів використовується вираз (3.1) за умови $M_+ = M_-$. У випадку нерівності M_+ і M_- визначається кількість базових тригерів окремо для підсумовувального і віднімального лічильників, а в якості результату обирається максимальне з отриманих значень.

Визначення функцій збудження для базових тригерів підсумовувального і віднімального лічильників розглянуто в підрозділах 3.1 – 3.3 цього посібника.

Використання комутатора розглядалося в підрозділі 2.1.1.1.

Інші кроки процедури синтезу реверсивного лічильника нічим не відрізняється від аналогічних кроків при синтезі регістрів та інших лічильників.

Приклад 3.8. Виконати синтез реверсивного лічильника за модулем *16* на основі *JK*-тригерів.

Розв'язок

Відповідно до умови завдання $M_+ = M_-$, в зв'язку з чим згідно з (3.1) кількість базових тригерів заданого лічильника складає: $n = \lceil \log_2 M \rceil = \lceil \log_2 16 \rceil = 4$.

Визначення функцій збудження базових тригерів, наприклад *T*-, *TC*- тригерів, для лічильників з різними типами організації переносів між розрядами і напрямом ліку описується виразами (3.3), (3.7), (3.9), (3.10), (3.13), (3.16), (3.17).

Для синтезу реверсивного лічильника будемо використовувати функції збудження асинхронного підсумовувального і віднімального лічильників відповідно до (3.3) і (3.7).

Далі виконаємо розробку комутатора реверсивного лічильника, таблиця мікрооперацій якого приведена в табл.3.18. В кожну клітину таблиці вписуємо відповідну функцію збудження згідно з (3.3) і (3.7).

Таблиця 3.18 – Таблиця мікрооперацій реверсивного лічильника

Мікрооперація	Функції збудження			
	T_3	T_2	T_1	T_0
U_+	$U_c \cdot Q_0 \cdot Q_1 \cdot Q_2$	$U_c \cdot Q_0 \cdot Q_1$	$U_c \cdot Q_0$	U_c
U_-	$U_c \cdot \overline{Q_0} \cdot \overline{Q_1} \cdot \overline{Q_2}$	$U_c \cdot \overline{Q_0} \cdot \overline{Q_1}$	$U_c \cdot \overline{Q_0}$	U_c

В результаті отримаємо логічні вирази функцій збудження базових тригерів:

$$T_0 = U_+ \cdot U_c \vee U_- \cdot U_c = U_c \cdot (U_+ \vee U_-);$$

$$T_1 = U_+ \cdot U_c \cdot Q_0 \vee U_- \cdot U_c \cdot \overline{Q_0};$$

$$T_2 = U_+ \cdot U_c \cdot Q_0 \cdot Q_1 \vee U_- \cdot U_c \cdot \overline{Q_0} \cdot \overline{Q_1}$$

$$T_2 = U_+ \cdot U_c \cdot Q_0 \cdot Q_1 \cdot Q_2 \vee U_- \cdot U_c \cdot \overline{Q_0} \cdot \overline{Q_1} \cdot \overline{Q_2};$$

Схема для моделювання реверсивного лічильника за модулем *16* приведена на рис.3.103, а результати моделювання – на рис.3.104.

На часовій діаграмі (рис.3.104) спочатку задане активне значення керуючого сигналу U_+ (при неактивному U_-), який налаштовує лічильник на роботу в якості підсумовувального лічильника, а далі, на другій частині діаграми задається активне значення сигналу U_- (при неактивному U_+), в

результаті чого лічильник налаштовується на роботу в якості віднімального лічильника.

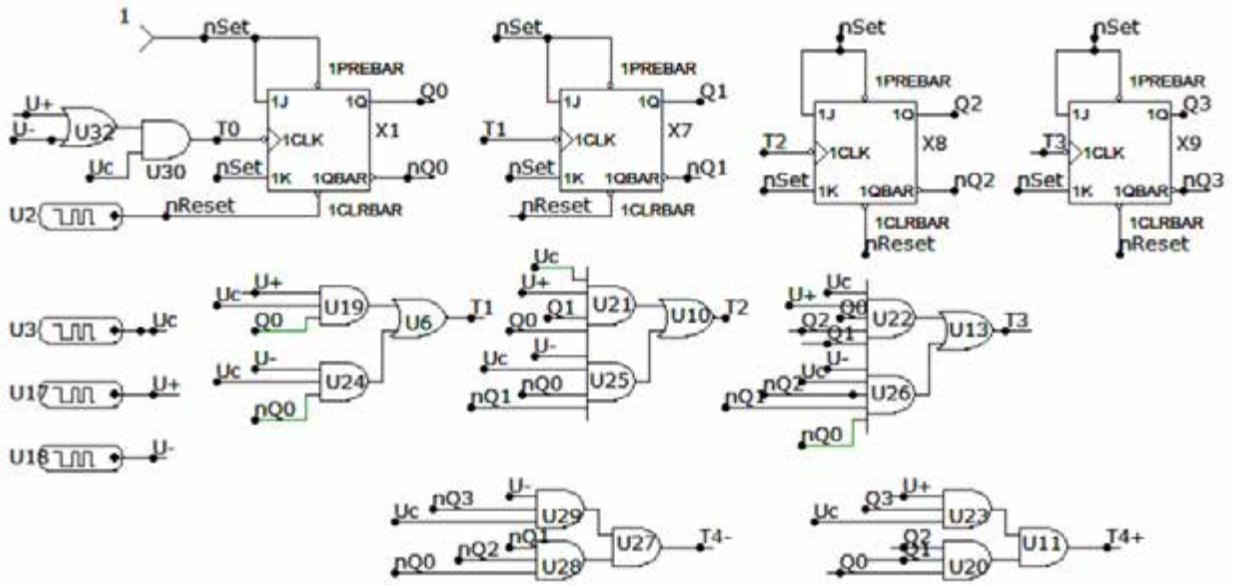


Рисунок 3.103 – Схема для моделювання реверсивного лічильника за модулем *16*

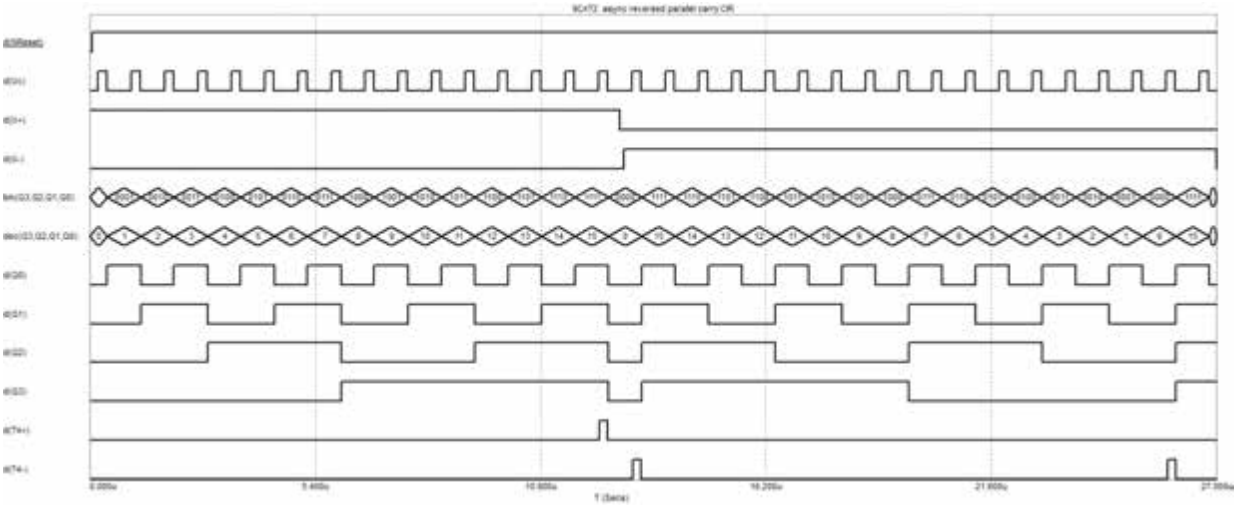


Рисунок 3.104 – Результати моделювання реверсивного лічильника за модулем *16*

Крім того, на часові діаграмі показані сигнали переносів зі старшого розряду лічильника T_+, який використовується як сигнал переносу підсумовувального лічильника (активізується при переключенні лічильника зі стану, що відповідає коду *M-1* в нульовий стан), та T_-, який використовується як сигнал переносу віднімального лічильника

(активізується при переключенні лічильника з нульового стану, в стан, що відповідає коду *M-1*). Ці сигнали використовуються для збільшення розрядності лічильника.

Звичайно, що за відсутності активних значень сигналів U_+ і U_- лічильник перебуває в стані збереження інформації.

У випадку наявності доступу користувача як до прямих, так і інверсних виходів базових тригерів реверсивний лічильник може бути отриманий за рахунок використання цих виходів. Наприклад, якщо в наявності є підсумовувальний лічильник за модулем 8, то на прямих виходах його базових тригерів формується послідовність кодів станів $0, 1, 2_{10}, \ldots, 7_{10}, 0$, що відповідає функціонуванню підсумовувального лічильника, а при цьому на інверсних виходах утворюється послідовність $7_{10}, 6_{10}, 5_{10}, \ldots, 1, 0, 7_{10}$, що відповідає функціонуванню віднімального лічильника. Таким чином, прямі виходи використовуються як виходи підсумовувального лічильника, а інверсні – віднімального.

Однак при цьому необхідно відзначити, що в режимі підрахунку вхідних сигналів таким чином можна виконувати реалізацію тільки двійкових лічильників. Якщо лічильник не є двійковим, то цей спосіб можна використовувати тільки в режимі ділення частоти. Наприклад, для підсумовувального лічильника за модулем 5 на прямих виходах базових тригерів маємо послідовність кодів станів $0, 1, 2_{10}, 3_{10}, 4_{10}, 0$, а на інверсних – $7_{10}, 6_{10}, 5_{10}, 4_{10}, 3_{10}, 7_{10}$.

3.4.2 Лічильники з можливістю завантаження інформації

Лічильники можуть також входити до складу багатофункціональних пристроїв, які окрім підрахунку вхідних сигналів виконують завантаження інформації із зовнішніх джерел, тобто лічильники і регістри інтегруються в такі пристрої.

Синтез таких пристроїв здійснюється окремо для кожної мікрооперації таким чином, як розглядалося в попередніх підрозділах, присвячених синтезу регістрів та лічильників, після чого отримані для кожної мікрооперації функції збудження базових тригерів об'єднуються за допомогою таблиці мікрооперацій.

В якості базових тригерів в таких пристроях, як правило, виступають *D*- або *JK*-тригери.

Якщо базовими тригерами є *D*-тригери, то синтез регістрової частини багатофункціонального пристрою (БП) виконується як синтез звичайного регістра, що було розглянуто в розділі 2 цього посібника. Синтез лічильної частини здійснюється, як правило, наступним чином:

1. Виконується синтез лічильника на базі паралельної або послідовної лічильної схеми, виходи якої через комутатор підключаються до базових тригерів.

2. Для кіл, що відносяться до лічильної частини БП, виконується перетворення базового *D*-тригера в асинхронний *T*-тригер відповідно до схеми на рис.1.15,в, після чого до рядків таблиці мікрооперацій, що стосуються лічильника, записуються функції збудження асинхронного *T*- тригера.

Розглянемо синтез БП на прикладі.

Приклад 3.9. Виконати синтез багатофункціонального пристрою на базі *D*-тригерів, що виконує такі мікрооперації:

- U_L: $Rg := A$ – паралельне завантаження інформації з шини *A*;

- U_c: $Rg := Rg + 1$ – підрахунок вхідних сигналів за модулем 16.

Розв'язок

Будемо вважати, що розрядність регістра визначається розрядністю лічильника, тобто відповідно до (3.1) кількість базових тригерів заданого лічильника складає: $n = \lceil \log_2 M \rceil = \lceil \log_2 16 \rceil = 4$.

Для синтезу лічильної частини БП будемо використовувати лічильник на базі паралельної лічильної схеми (підрозділ 3.3).

Визначення функцій збудження базових тригерів для обох частин БП вже розглядалося раніше, наприклад, для функцій збудження лічильної схеми будемо використовувати вирази (3.24).

В цьому разі таблиця мікрооперацій, яка описує комутатор пристрою, буде виглядати таким чином (табл.3.19):

Таблиця 3.19 – Таблиця мікрооперацій БП (приклад 3.9)

Мікрооперація	Функції збудження			
	D_3	D_2	D_1	D_0
U_L	a_3	a_2	a_1	a_0
U_c	CC_3	CC_2	CC_1	CC_0

В табл.3.19 для лічильної схеми використовуються вирази (3.21)-(3.23), отримані раніше:

$CC_3 = Q_3 \oplus Q_2 \cdot Q_1 \cdot Q_0$; $CC_2 = Q_2 \oplus Q_1 \cdot Q_0$; $CC_1 = Q_1 \oplus Q_0$; $CC_0 = \overline{Q_0}$.

В результаті отримаємо логічні вирази функцій збудження базових тригерів: $D_i = U_L \cdot a_i \vee U_c \cdot CC_i$.

Сигнал синхронізації базових тригерів $u_{БТ}$ визначається відповідно до виразу (2.5): $u_{БТ} = G \cdot (U_L \vee U_c)$.

Схема для моделювання заданого БП приведена на рис.3.105, а результати моделювання – на рис.3.106.

Для організації моделювання після встановлення базових тригерів в нульовий стан за допомогою сигналу ***nReset*** (на часовій діаграмі не показаний) задається активне значення сигналу U_L та по сигналу G пристрій приймає чотири біти інформації (1001_2) з шини A, яка імітується постійними сигналами ***Fixed Digital***. Після цього задається серія лічильних сигналів U_c, за якими відбувається підрахунок вхідних сигналів, починаючи з прийнятого по U_L коду, тобто за стану 1001_2.

Розглянемо також синтез БП на базі *JK*-тригерів.

Приклад 3.10. Виконати синтез багатофункціонального пристрою на базі *JK*-тригерів, що виконує ті ж самі мікрооперації, що і в прикладі 3.9.

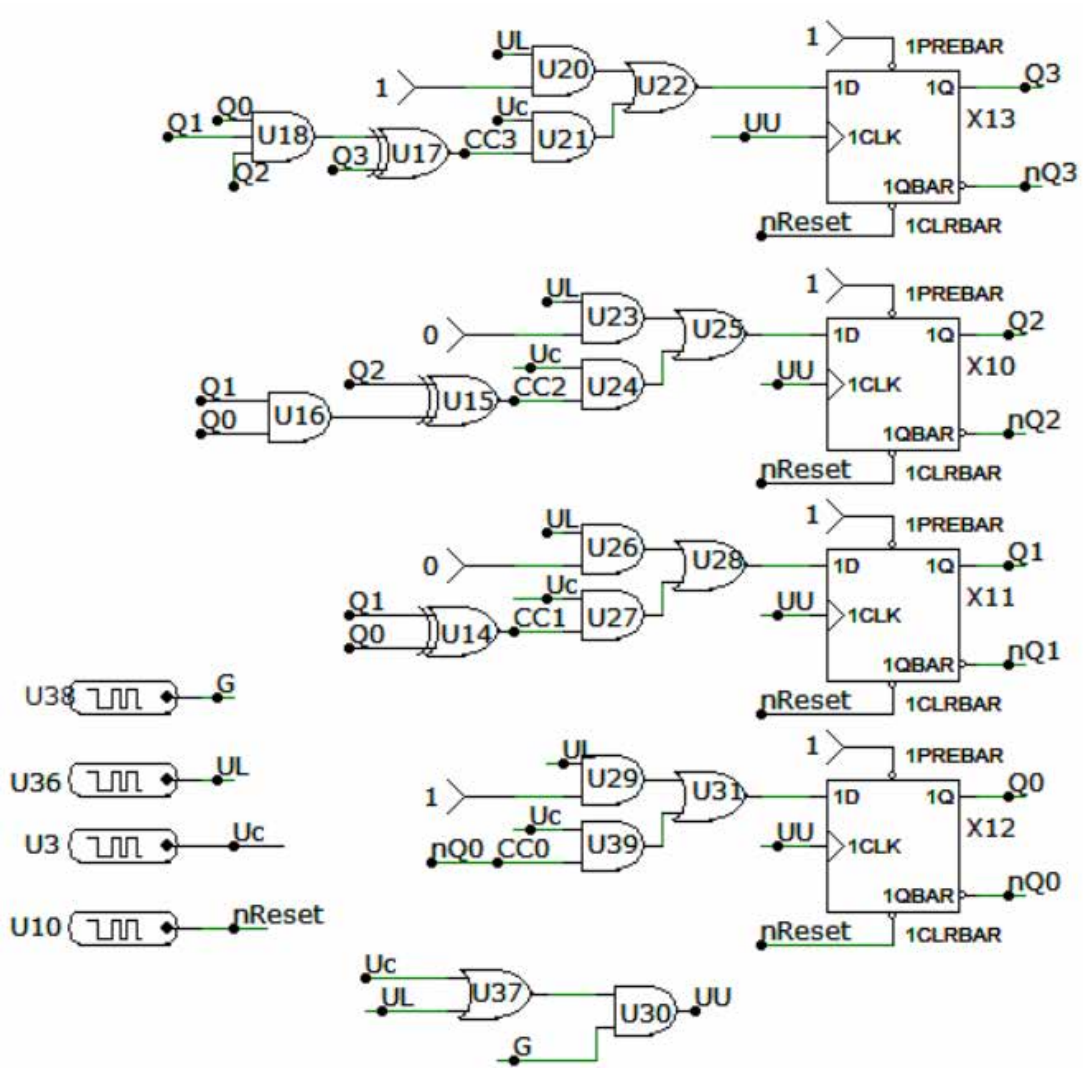


Рисунок 3.105 – Схема для моделювання БП (приклад 3.9)

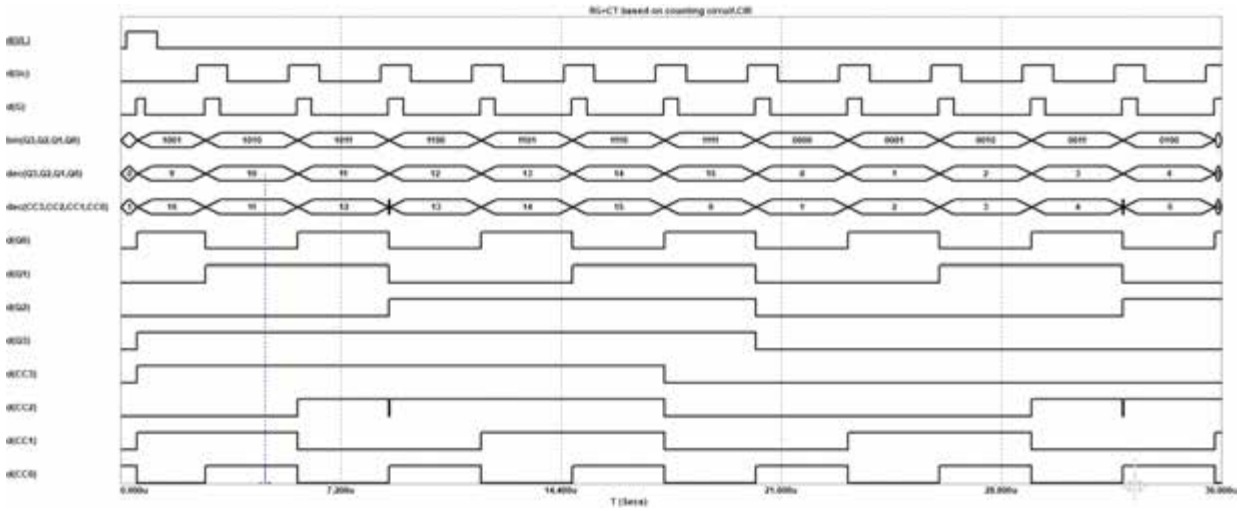


Рисунок 3.106 – Результати моделювання БП (приклад 3.9)

Розв'язок

Розрядність пристрою, як і в прикладі 3.9 дорівнює *4*.

Для синтезу лічильної частини БП будемо використовувати синхронний підсумовувальний лічильник з паралельним трактом розповсюдження переносу (підрозділ 3.1.1.2).

Визначення функцій збудження базових тригерів для обох частин БП вже розглядалося раніше, наприклад, для функцій збудження лічильної схеми будемо використовувати вирази (3.8).

В цьому разі таблиця мікрооперацій, яка описує комутатор пристрою, буде виглядати таким чином (табл.3.20):

Таблиця 3.20 – Таблиця мікрооперацій БП (приклад 3.10)

Мікрооперація	Функції збудження			
	KOM_3	KOM_2	KOM_1	KOM_0
U_L	a_3	a_2	a_1	a_0
U_c	$Q_2 \cdot Q_1 \cdot Q_0$	$Q_1 \cdot Q_0$	Q_0	$1c$

Функції збудження *JK*-тригера приведено в розділі 2 (рис.2.5), тобто $J_i = KOM_i$; $K_i = \overline{KOM_i}$

Таким чином, маємо:

$J_3 = U_L \cdot a_3 \vee U_c \cdot Q_2 \cdot Q_1 \cdot Q_0$; $K_3 = U_L \cdot \overline{a_3} \vee U_c \cdot Q_2 \cdot Q_1 \cdot Q_0$;

$J_2 = U_L \cdot a_2 \vee U_c \cdot Q_1 \cdot Q_0$; $K_2 = U_L \cdot \overline{a_2} \vee U_c \cdot Q_1 \cdot Q_0$;

$J_1 = U_L \cdot a_1 \vee U_c \cdot Q_0$; $K_1 = U_L \cdot \overline{a_1} \vee U_c \cdot Q_0$;

$J_0 = U_L \cdot a_0 \vee U_c$; $K_0 = U_L \cdot \overline{a_0} \vee U_c$.

Сигнал синхронізації базових тригерів $u_{БТ}$ визначається відповідно до виразу (2.5): $u_{БТ} = G \cdot (U_L \vee U_c)$.

Схема для моделювання заданого БП приведена на рис.3.107, а результати моделювання – на рис.3.108.

Процес моделювання організований таким же чином, як в прикладі 3.9, тобто спочатку приймається код 1001_2, а потім відбувається виконання мікрооперацій підрахунку вхідних сигналів.

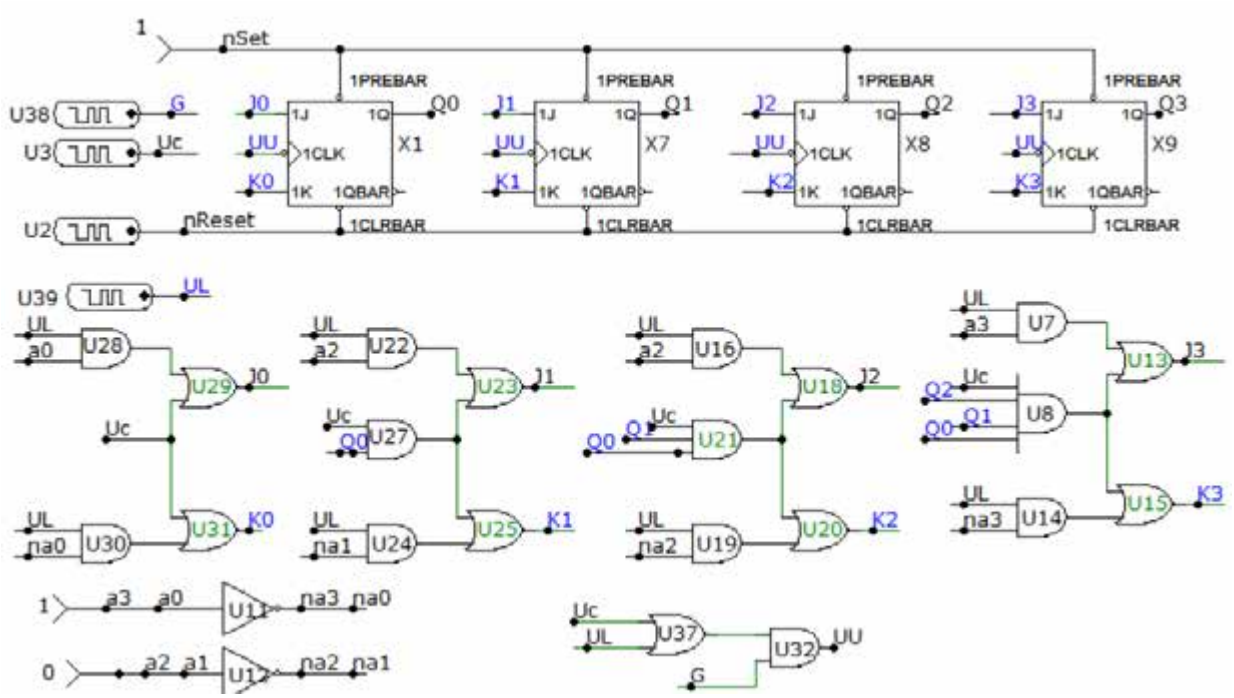


Рисунок 3.107 – Схема для моделювання БП (приклад 3.10)

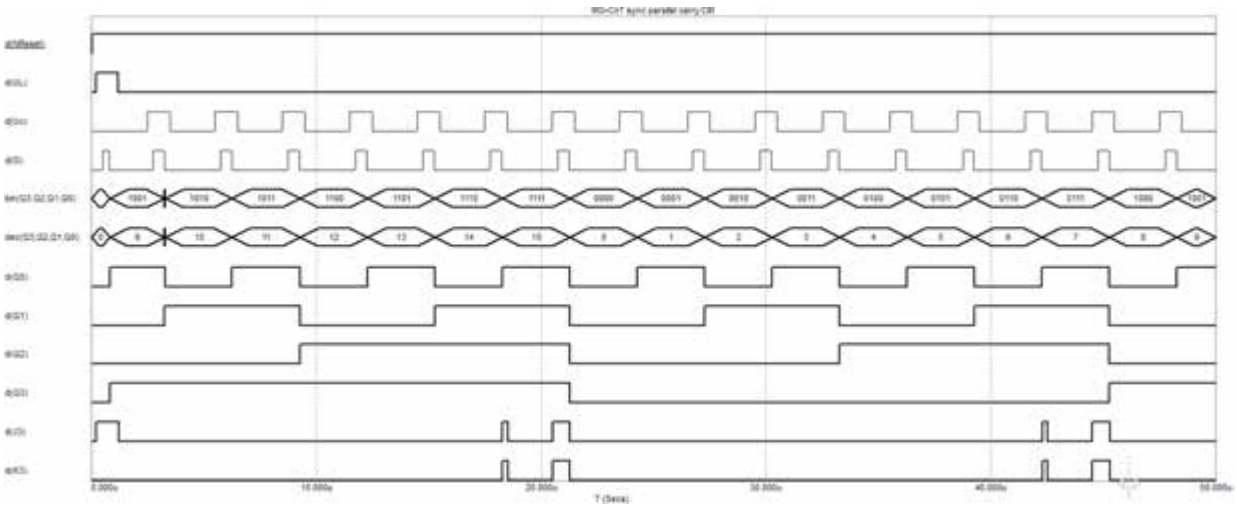


Рисунок 3.108 – Результати моделювання БП (приклад 3.10)

На цьому виконання прикладу 3.10 завершено.

Іноді синтез БП виконується таким чином, що лічильні кола проєктуються як у звичайного лічильника, а завантаження інформації із зовнішніх джерел здійснюється за допомогою асинхронних установчих входів базових тригерів.

Контрольні завдання та запитання

1. Які лічильники називаються реверсивними?

2. Яку мікрооперацію виконує підсумовувальний лічильник?

3. Яку мікрооперацію виконує віднімальний лічильник?

4. Яким чином відбувається синтез реверсивних лічильників?

5. Наведіть кроки процедури синтезу реверсивних лічильників.

6. Поясніть принцип функціонування реверсивних лічильників за умови, що $M_+ = M_-$.

7. Поясніть принцип функціонування реверсивних лічильників за умови, що $M_+ \neq M_-$.

8. Як визначається кількість базових тригерів, якщо $M_+ = M_-$?

9. Як визначається кількість базових тригерів, якщо $M_+ \neq M_-$?

10. Поясніть принцип заповнення табл.3.18.

11. Як отримати функції збудження з табл.3.18?

12. Як отримано функціональну схему на рис.3.103?

13. Прокоментуйте результати моделювання на рис.3.104.

14. Яким чином формуються сигнали переносів зі старшого розряду в схемі на рис.1.103?

15. Яким чином працює лічильник за відсутності активних значень сигналів U_+ і U_-?

16. Як можна реалізувати реверсивний лічильник за наявності доступу до прямих та інверсних виходів базових тригерів?

17. Який тип реверсивного лічильника з точки зору модулю ліку можна отримати за рахунок сумісного використання прямих та інверсних виходів базових тригерів?

18. В якому режимі може використовуватися не двійковий реверсивний лічильник при сумісному використанні прямих та інверсних виходів базових тригерів?

19. Чи можна використовувати в реверсивному лічильнику прозорі тригери? Обґрунтуйте відповідь.

20. Виконати синтез реверсивного лічильника за модулем 32 на базі *T*- тригерів.

21. Виконати синтез реверсивного лічильника за модулем 32 на базі *TC*-тригерів.

22. Виконати синтез реверсивного лічильника за модулем 10 на базі *JK*- тригерів.

23. Виконати синтез реверсивного лічильника за модулем 10 на базі *T*- тригерів.

24. Які мікрооперації виконують лічильники з можливістю завантаження інформації із зовнішніх джерел?

25. Риси яких цифрових вузлів включають лічильники з можливістю завантаження інформації із зовнішніх джерел?

26. Чи можуть лічильники з можливістю завантаження інформації із зовнішніх джерел виконувати мікрооперації зсуву інформації? Обґрунтуйте відповідь.

27. Яким чином здійснюється синтез лічильників з можливістю завантаження інформації?

28. Як побудувати таблицю мікрооперацій?

29. Як визначити функції збудження за допомогою таблиці мікрооперацій?

30. Яку функцію виконує комутатор в лічильниках з можливістю завантаження інформації?

31. Які типи базових тригерів з точки зору функціонального призначення в основному використовуються в лічильниках з можливістю завантаження інформації?

32. Вкажіть на особливості синтезу лічильників з можливістю завантаження інформації, якщо базовими тригерами є *D*-тригери?

33. Вкажіть на особливості синтезу лічильників з можливістю завантаження інформації, якщо базовими тригерами є *T*-тригери?

34. Як перетворити *D*-тригер в *T*-тригер ?

35. Виконати синтез багатофункціонального пристрою на базі *D*- тригерів. Пристрій виконує мікрооперації U_L: *Rg*:= *A*; U_c: *Rg*:= *Rg-1* – за модулем 16.

36. Виконати синтез багатофункціонального пристрою на базі *D*- тригерів. Пристрій виконує мікрооперації U_L: *Rg*:= *AR1*(*A*) – арифметичний зсув вправо на 1 розряд; U_c: *Rg*:= *Rg+1* – за модулем 16.

37. Виконати синтез багатофункціонального пристрою на базі *T*- тригерів. Пристрій виконує мікрооперації U_L: *Rg*:= *A*; U_c: *Rg*:= *Rg-1* – за модулем 16.

38. Виконати синтез багатофункціонального пристрою на базі *T*- тригерів. Пристрій виконує мікрооперації U_L: *Rg*:= *AR1*(*A*) – арифметичний зсув вправо на 1 розряд; U_c: *Rg*:= *Rg+1* – за модулем 16.

39. Виконати синтез багатофункціонального пристрою на базі *D*- тригерів. Пристрій виконує мікрооперації U_L: *Rg*:= *A*; U_c: *Rg*:= *Rg+1* – за модулем 10.

40. Виконати синтез багатофункціонального пристрою на базі *D*- тригерів. Пристрій виконує мікрооперації U_L: *Rg*:= *A*; U_c: *Rg*:= *Rg-1* – за модулем 10.

41. Синтезувати реверсивний лічильник з будь-яким модулем ліку з можливістю завантаження інформації.

42. Як заповнити таблицю 3.19?

43. Як реалізовано формування сигналу $u_{БТ}$?

44. Як визначені функції збудження *D*-тригерів в прикладі 3.9?

45. Прокоментуйте схему на рис.3.105.

46. Прокоментуйте результати моделювання на рис.3.106.

47. Для чого використовується сигнал ***nReset***?

48. Як визначається кількість базових тригерів в прикладі 3.9. Обґрунтуйте відповідь.

49. Як визначається кількість базових тригерів в прикладі 3.10. Обґрунтуйте відповідь.

50. Як заповнена таблиця 3.20?

51. Як побудувати *СФФЗ* для *JK*-тригерів?

52. Як визначені функції збудження *JK*-тригерів в прикладі 3.10?

53. Прокоментуйте схему на рис.3.107.

54. Прокоментуйте результати моделювання на рис.3.108.

55. Як реалізувати завантаження інформації в лічильниках за допомогою асинхронних установчих входів базових тригерів?

56. Виконати синтез багатофункціонального пристрою на базі *JK*- тригерів. Пристрій виконує мікрооперації U_L: *Rg*:= *A*; U_c: *Rg*:= *Rg-1* – за модулем 16.

57. Виконати синтез багатофункціонального пристрою на базі *JK*- тригерів. Пристрій виконує мікрооперації U_L: *Rg*:= *AR1*(*A*) – арифметичний зсув вправо на 1 розряд; U_c: *Rg*:= *Rg+1* – за модулем 16.

58. Виконати синтез багатофункціонального пристрою на базі *JK*- тригерів. Пристрій виконує мікрооперації U_L: *Rg*:= *A*; U_c: *Rg*:= *Rg+1* – за модулем 10.

59. Виконати синтез багатофункціонального пристрою на базі *JK*- тригерів. Пристрій виконує мікрооперації U_L: *Rg*:= *A*; U_c: *Rg*:= *Rg-1* – за модулем 10.

60. Виконати синтез багатофункціонального пристрою на базі *JK*- тригерів. Пристрій виконує мікрооперації U_L: *Rg*:= *A*; U_c: *Rg*:= *Rg+1* – за модулем 16 із завантаженням інформації за допомогою асинхронних установчих входів базових тригерів.

СПИСОК ЛІТЕРАТУРИ
для більш докладного самостійного вивчення математичних принципів і теорії побудування цифрових пристроїв

1. Комп'ютерна схемотехніка (частина 1) [навчальний посібник] / Б.С.Гусєв, Д.Ю. Касаткін, Т.Ю. Осипова // - К.: НУБіП України, 2022.- 264с.
2. Комп'ютерна логіка [навчальний посібник] / Лахно В.А., Гусєв Б.С., Касаткін Д.Ю. // - Київ, ЦП «Компрінт», 2018, - 418с.
3. Комп'ютерна логіка. Конспект лекцій / Укладач Б.С.Гусєв // - К.: НУБіП України, 2020.- 161с.
4. Методичні вказівки до виконання лабораторних робіт з курсу «Комп'ютерна схемотехніка» з використанням навчально-лабораторних стендів TRIGGER і LOGIC (частина 1) / Укладач Б.С.Гусєв. – Київ, НУБіП України, 2022, 114с.
5. Методичні вказівки до виконання лабораторних робіт з курсу «Комп'ютерна схемотехніка» з використанням навчально-лабораторних стендів TRIGGER і LOGIC (частина 2) / Укл. Б.С.Гусєв. – Київ, НУБіП України, 2022, 115с.
6. https://www.ti.com
7. https://datasheetspdf.com
8. https://pdf1.alldatasheetru.com/datasheet-pdf/view/27384/TI/SN74194.html
9. https://pdf1.alldatasheet.com/datasheet-pdf/view/27391/TI/SN74195N.html
10. https://pdf1.alldatasheet.com/datasheet-pdf/view/1526559/TI1/SN74109.html
11. Лапко В.В., Гусєв Б.С., Касаткін Д.Ю., Смолій В.В., Блозва А.І., Осипова Т.Ю., Матус Ю.В., Савицька Я.А. Комп'ютерна схемотехніка та логіка. Частина І. «Математичні засади і схемотехніка арифметичних засобів комп'ютерних пристроїв»: Навчальний посібник // - К.: НУБіП України, 2017.- 291с.
12. Жабін В.І., Жуков І.А., Клименко І.А., Ткаченко В.В. Прикладна теорія цифрових автоматів. Навчальний посібник. Київ, Національний авіаційний університет, 2007р., 363с.
13. Бабич М.П., Жуков І. А. Комп'ютерна схемотехніка: Навчальний посібник. – К: «МК-Прес», 2004. – 412с.
14. Матвієнко М. П. Комп'ютерна логіка: Навчальний посібник. – К: «Ліра- К», 2012. – 289с.
15. Anil K. Maini. Digital Electronics - Principles, Devices and Applications, Wiley, 2007, 741p.
16. Матвієнко М. П., Розен В.П. Комп'ютерна схемотехніка: Навчальний посібник. – К: «Ліра- К», 2014. – 192с.
17. Матвієнко М. П., Проєктування цифрових пристроїв: Навчальний посібник. – К: «Ліра- К», 2018. – 362с.
18. Соколовський Я.І. Комп'ютерна схемотехніка: навч. посіб. / Я. І. Соколовський, І. І. Пірко, І. Р. Кенс, М. В. Дендюк, С. І. Яцишин. – Львів: Магнолія – 2006, 2018. – 313 с.

19. Raymond E. Frey. Lecture Notes for Digital Electronics. University of Oregon Eugene, 2000, 43c.
20. Mano M. Morris. Digital Logic And Computer Design. 525c.