МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

С.Ю. Леонов, Г.В. Гейко

ТЕХНОЛОГІЯ АВТОМАТИЗОВАНОГО ПРОЄКТУВАННЯ КОМП'ЮТЕРНИХ СИСТЕМ ТА ЇХ СКЛАДОВИХ

Навчальний посібник для студентів
спеціальності «Комп'ютерна інженерія»

Затверджено
редакційно-видавничою
радою НТУ «ХПІ»,
протокол № 2 від 27.06.2024

Харків
НТУ «ХПІ»
2024

УДК 004.896
Л47

Рецензенти:

Аврунін О.Г., д-р техн. наук, професор, завідувач кафедри біомедичної інженерії, Харківський національний університет радіоелектроніки

Трубчанінова К.А., д-р техн. наук, професор, професор кафедри транспортного зв'язку, Український державний університет залізничного транспорту

Леонов С.Ю.
Л47 Технологія автоматизованого проєктування комп'ютерних систем та їх складових. Навчальний посібник для студентів спеціальності «Комп'ютерна інженерія» / С. Ю. Леонов, Г. В. Гейко. – Харків : НТУ «ХПІ», 2024. – 256 с.

Навчальний посібник включає теоретичний і практичний матеріал, пов'язаний із забезпеченням систем автоматизованого проєктування. Наведено основні відомості про склад і принципи функціонування систем автоматизованого проєктування обчислювальної техніки. Описано повний цикл проєктування складних ієрархічних пристроїв – від функціонального моделювання та верифікації до конструкторського проєктування та отримання друкованих плат.

Для студентів спеціальності «Комп'ютерна інженерія» та інших технічних спеціальностей.

Іл. 272. Табл. 32. Бібліогр. 6 назв.

УДК 004.896
ISBN

ВСТУП

Системи автоматизованого проєктування (САПР) полегшують створення, зміну, аналіз та оптимізацію проєктів із використанням комп'ютерних систем. САПР поділяються на системи автоматизованого проєктування (computer-aided design, CAD), автоматизованого виробництва (computer-aided manufacturing, CAM) і автоматизованого конструювання (computer-aided engineering, CAE).

Найголовніша функція автоматизованого проєктування – це визначення геометрії конструкції та моделювання функціональних та електричних схем. Для цієї мети зазвичай використовуються системи розробки робочих креслень, моделювання та функціонального чи топологічного проєктування.

Автоматизоване конструювання – це технологія, що полягає у використанні комп'ютерних систем для аналізу геометрії CAD, моделювання, удосконалення та оптимізації конструкції.

OrCAD – це система, що дозволяє розробнику моделювати електронні схеми і розробляти друковані плати. Комплект програм складається з OrCAD Capture (графічний редактор електричних принципових схем); OrCAD Capture CIS (редактор принципових схем з можливістю ведення баз даних компонентів); OrCAD Layout Plus (редактор друкованих плат); OrCAD PSpice A/D (використовується для змішаного моделювання аналогових і цифрових пристроїв).

У навчальному посібнику висвітлюються питання методологічних основ побудови САПР та аналізу працездатності проєктованих пристроїв. Розглядається практичне використання САПР OrCAD при розробці складних ієрархічних обчислювальних пристроїв із застосуванням різних методів висхідного та спадного проєктування, аналізом їхньої працездатності на основі моделювання, а також топологічне проєктування цих пристроїв.

Розділ 1
ОСНОВИ ПОБУДОВИ САПР

1.1 Методологія побудови систем проєктування

Проєктування технічного об'єкта пов'язане зі створенням, перетворенням та поданням у прийнятій формі образу цього об'єкта. Проєктування починається зі складання завдання на проєктування, яке подається у вигляді технічних або інших документів і є вихідним описом об'єкта. Результат проєктування – це повний комплект документації, який містить відомості, достатні для виготовлення об'єкта. Така документація є остаточним описом об'єкта. Процес перетворення вихідного опису в остаточний опис об'єкта являє собою сукупність проміжних описів або проєктних рішень.

Автоматизоване проєктування – це процес або сукупність заходів, спрямованих на виконання проєктних рішень за допомогою електронних обчислювальних машин (ЕОМ). При цьому має бути передбачений раціональний розподіл функцій між людиною (проєктувальником) та ЕОМ. Неавтоматизоване проєктування – це проєктування, у якому ЕОМ немає. Автоматичне проєктування – це вид проєктування, у якому проєктування виконується за допомогою ЕОМ без участі людини, тобто проєктувальник повністю «виключений» зі сфери проєктування і бере участь тільки у процесі прийняття рішень.

Мета автоматизації проєктування:

– підвищення якості;

– зниження матеріальних витрат;

– скорочення засобів проєктування;

– зменшення кількості проєктувальників;

– підвищення продуктивності праці проєктувальників.

САПР – це комплекс засобів автоматизованого проєктування, взаємопов'язаний із підрозділами проєктної організації. Тісна взаємодія людини та ЕОМ у процесі проєктування э одним із принципів побудови та експлуатації САПР. Розрізняють зовнішнє та внутрішнє проєктування. До зовнішнього проєктування належать передпроєктні дослідження, розробка, затвердження технічного завдання. До внутрішнього проєктування відносять стадії розробки технічної пропозиції, ескізного

проєкту, технічного проєкту, робочого проєкту, виготовлення, налагодження, випробування та введення в дію. Передпроєктні дослідження – це коли проводяться обстеження проєктної організації, оформлення технічного звіту та затвердження.

Обстеження включає такі заходи:

– оцінка можливості створення САПР;

– збір даних, опис та аналіз САПР;

– збір пропозицій щодо створення САПР;

– склад підсистем та компонентів САПР;

– формування технічних вимог до функцій та структури САПР;

– види забезпечення та принципи створення САПР.

Класифікацію САПР здійснюють за різними ознаками. У галузі застосування широко використовуваними є САПР для машинобудування (MCAD), САПР для радіоелектроніки (ECAD, EDA), САПР в галузі архітектури та будівництва (ArCAD). За цільовим призначенням розрізняють САПР, які реалізують різні аспекти проєктування. Наприклад, у складі MCAD виділяють САПР функціонального проєктування, які називають системами розрахунків та інженерного аналізу або системами CAE (Computer Aided Engineering); конструкторські САПР (CAD – Computer Aided Design), технологічні САПР (CAM – Computer Aided Manufacturing). Також САПР розрізняють за масштабами і за характером базової підсистеми.

Необхідність конструювання складних систем привела до активного використання системного підходу в техніці, який являє собою сукупність методологічних принципів і положень, що дозволяють розглядати систему як єдине ціле. Такий підхід передбачає вивчення кожного елемента системи у його взаємодії з іншими елементами і враховує такі особливості: процес проєктування повинен мати ієрархічну структуру; процес проєктування повинен враховувати аналіз зв’язків, які існують в об’єкті та у керуючій частині системи; також при проєктуванні слід передбачати безперервний розвиток системи.

В процесі проєктування виділяють задачі структурного та параметричного синтезу. У задачах синтезу найчастіше використовуються методи штучного інтелекту. Особливістю таких задач є необхідність багаторазового прийняття рішень в процесі їх реалізації.

Основні типи синтезу: оптимізаційний (розробник сам створює початковий варіант проєктованого об'єкта, а потім, якщо характеристики об'єкта відрізняються від необхідних, змінюються параметри або структура цього об'єкта; генераційний (полягає у створенні моделі об'єкта).

Існує безліч класифікацій задач синтезу. Наприклад, в якості класифікаційної ознаки можуть бути обрані стадії, рівні, етапи, характеристики математичних моделей, способи вирішення та ін. На рис. 1.1 наведена одна з таких класифікацій.

Рисунок 1.1 – Класифікація задач синтезу

Щоб об'єкт синтезувати, необхідно спочатку визначити його структуру, параметри елементів і конструкцію. При проєктуванні виділяють такі стадії:

– на стадії науково-дослідних робіт вивчаються потреби в отриманні нових виробів із заданим цільовим призначенням, досліджуються принципи побудови виробів і можливості реалізації цих принципів, прогнозуються значення характеристик та параметрів об'єктів (результатом є формулювання технічного завдання на розробку об'єкта);

– на стадії дослідно-конструкторських робіт створюється ескізний проєкт виробу (при його розробці перевіряються, конкретизуються і коригуються принципи і положення, встановлені на стадії науково-дослідницьких робіт);

– на стадії технічного проєкту розробляється більш деталізована

графічна і текстова документація, яка дає повне уявлення про пристрій (в технічний проєкт при цьому включають всі необхідні розрахунки);

– на стадії робочого проєкту створюється повний комплект конструкторсько-технологічної документації, достатній для виготовлення об'єкта;

– на стадії випробувань отримують результати, що дозволяють виявити можливі помилки і недоробки проєкту, а також вживаються заходи щодо їх усунення.

На будь-якому етапі проєктування може бути виявлена помилковість або раніше прийняті не оптимальні рішення. Зокрема, може бути виявлена необхідність коригування технічного завдання.

З науково-технічної точки зору проблема створення САПР зводиться до вирішення таких задач:

– розробка методики проєктування виробів електронної техніки;

– розробка прикладного математичного забезпечення;

– створення технічних засобів САПР;

– створення організаційно-технічного забезпечення.

Методологія створення САПР в загальному вигляді може бути сформульована таким чином: підготовка кадрів; організація процесу розробки САПР; реалізація розподіленої системи управління; використання апарату штучного інтелекту. Більшість всіх сучасних САПР основані на вичерпній внутрішній логіці, що вимагає від користувача явного опису всіх дій, які визначають процес рішення. Така організація процесу проєктування має недоліки, які полягають в тому, що САПР не є еволюційними та область їх застосування обмежується рішенням типових завдань. У той же час введення методів штучного інтелекту (ШІ) дозволяє усунути ці недоліки.

Умовно можна виділити такі рівні інтелектуалізації САПР:

1) рівень, на якому засоби ШІ використовуються тільки як компоненти окремих підсистем САПР;

2) рівень, який передбачає наявність у САПР підсистем, які повністю організовані відповідно до методології ШІ;

3) рівень, який досягається при побудові САПР цілком на організаційних принципах систем ШІ.

Основним об'єктом вивчення в ШІ є знання. Це поняття, яке не

визначається формально і має такі особливості: інтерпретованість, структурованість, зв'язність, активність. Знання про об'єкти та явища ділять на синтаксичні (описують структуру об'єктів та явищ), семантичні (інформація, пов'язана з їх значенням і сенсом) і прагматичні (описують об'єкти і явища з точки зору розв'язуваної задачі).

У процесі роботи інтелектуалізованої САПР (ІСАПР) вирішуються наступні задачі:

– складається технічне завдання (ТЗ) і виконується його аналіз;

– вибирається конструктивно-компонувальна схема і аналізується вартість проєкту;

– проводиться структурний синтез і оптимізація;

– ведеться пошукове конструювання та планується проєкт;

– здійснюється перепроєктування та доопрацювання конструкції;

– підвищується ефективність та якість інженерного аналізу, проводиться імітаційне моделювання;

– перевіряється відповідність галузевим стандартам;

– готується документація і робочі креслення.

Досвід проєктування великих інтегральних схем (ВІС) дозволяє сформулювати вимоги до САПР таких схем:

– інваріантність структури до зміни засобів, методів проєктування і технології виготовлення ВІС;

– сумісність ручного, автоматизованого і автоматичного проєктування;

– різноманітність режимів взаємодії проєктувальника з САПР;

– продуктивність;

– адаптивність;

– одночасність проєктування декількох виробів;

– незалежність введення в дію та експлуатацію окремих підсистем;

– швидка налаштованість системи на обраний технологічний маршрут проєктування;

– інформаційна узгодженість підсистем і пакетів прикладних програм для різних етапів проєктування ВІС;

– різноманітність форм представлення вихідної документації;

– висока надійність та прийнятна вартість.

Ці вимоги будуть задоволені, якщо при створенні САПР ВІС

використовувати такі принципи:

– принцип комплексності (можливості розв'язання завдань по всьому циклу проєктування виробу);

– сумісність автоматичного, автоматизованого і традиційного проєктування;

– наявність загального банку даних;

– модульність (це дозволить реалізовувати окремі підсистеми у вигляді функціонально самостійних модулів з подальшою їх заміною іншими модулями);

– мультидоступність (САПР ВІС повинна бути системою колективного використання);

– відкритість (додавання нових підсистем або окремих програм не призводить до необхідності змінювати способи організації управління, опису та обробки інформації).

При проєктуванні ВІС використовується, перш за все, принцип декомпозиції, який дозволяє звести задачу великої розмірності до кількох задач меншої розмірності. Декомпозиція проєктованого об'єкта призводить до ієрархічного принципу проєктування, і чим складніший об'єкт, що проєктується, тим більше рівнів ієрархії. При проєктуванні необхідно забезпечити перевірку правильності проєктних рішень ще на стадії проєктування. Суть ітераційного принципу проєктування полягає в послідовному наближенні до виконання заданих вимог за результатами моделювання та оптимізації на кожному етапі. Необхідно зазначити, що контрольованість кожного етапу є важливим принципом проєктування. При розміщенні елементів і трасування усередині схемних з'єднань на кристалі інтегральної схеми (ІС) або друкованої плати застосовується контроль, який поєднано з процесом проєктування. Контроль правильності виконання проєктних робіт на різних етапах ВІС називають верифікацією.

При виготовленні ВІС слід ураховувати також можливу появу дефектів, обумовлених помилками проєктування, через що до стадії виготовлення зразка всі помилки слід ліквідувати, скориставшись системою верифікації топології. Також слід зазначити, що при проєктуванні надвеликих інтегральних схем (НВІС), коли процес проєктування топології не завершений (тобто не закінчено трасування

частини міжз'єднань), може бути проведена корекція топології ручними методами, в ході якої також виявляються помилки.

Види перевірки, необхідні на останньому етапі проєктування, діляться на три групи:

1) перевірка геометричних параметрів, виявлення відхилень від правил проєктування;

2) перевірка міжз'єднань, виявлення дефектів і похибок у виготовленні схемних елементів;

3) перевірка електричних характеристик.

Результатом роботи САПР ВІС повинен бути випуск проєктної документації, що включає в себе:

– текстові та тексто-графічні документи (опис алгоритму функціонування, функціонально-логічну схему, часові діаграми, принципові електричні схеми функціональних блоків, електричні характеристики функціональних блоків і компонентів, таблиці параметрів компонентів);

– графічні документи (креслення топології функціональних блоків ВІС);

– документи на машинних носіях (контролюючі тести для установок функціонального контролю, інформація на машинних носіях для програмно-керованого технологічного устаткування ВІС).

Програмне забезпечення (ПЗ) САПР ВІС розділяється на загальносистемне та прикладне. Загальносистемне ПЗ являє собою сукупність системних програм, яка організує виконання прикладних програм на технічних засобах САПР. Прикладне ПЗ характеризується сукупністю пакетів прикладних програм (ППП) для всіх етапів проєктування ВІС та інструкцій щодо їх використання. Прикладне програмне забезпечення САПР ВІС має складатися з набору ППП, який для кожного етапу проєктування повинен бути свій, та в межах кожного з цих наборів повинні використовуватися єдині правила для опису вихідних даних. ППП повинні мати в своєму складі програми, які повинні виконувати: структурний синтез складання математичної моделі; рішення математичної моделі; оптимізацію; статистичний аналіз. У методичне забезпечення входить сукупність інструкцій та інших документів, що регламентують порядок використання засобів

автоматизованого проєктування.

Організаційне забезпечення являє собою перелік документів, що регламентують взаємодію і функції користувачів, які беруть участь у процесі автоматизованого проєктування, а також матеріально-технічне забезпечення САПР ВІС.

Технічне забезпечення – це сукупність ЕОМ і периферійних засобів для введення, зберігання, переробки інформації, передачі програм і даних. Ці технічні засоби повинні забезпечувати інтерактивний режим роботи, виконання всіх необхідних проєктних процедур, взаємодію між усіма розробниками, які працюють над спільним проєктом. Під маршрутом проєктування ВІС розуміють послідовність проєктних процедур, які необхідно виконати при розробці конкретної схеми. На даний час внаслідок різноманітності технологічних процесів виготовлення ВІС та методик проєктування існує безліч маршрутів проєктування. Необхідно зазначити, що в наш час у підготовці виробництва виробів центральне місце займають питання проєктування технологічних процесів, які містять інформацію про трудові і матеріальні нормативи, без яких неможливе планування і управління виробничими ресурсами. Якщо є засоби автоматизації оформлення технічної документації та засоби інформаційної підтримки проєктування, то можна досягти великих успіхів в економії праці технологів, що є основною метою створення САПР технологічних процесів.

1.2 Блочно-ієрархічний підхід при проєктуванні

Практика показує, що переважна більшість складних систем як у природі так і в техніці має ієрархічну внутрішню структуру. Кожну підсистему можна розділити на підсистеми аж до нижнього «елементарного» рівня. При цьому вибір рівня, компоненти якого слід вважати елементарними, залишається за дослідником. На елементарному рівні система зазвичай складається з небагатьох типів підсистем, по-різному скомбінованих і організованих.

Процес розбиття складного об’єкта на порівняно незалежні частини має назву декомпозиції. При декомпозиції враховують, що зв’язки між окремими частинами мають бути слабшими, ніж зв’язки елементів усередині частин. Крім того, щоб з отриманих частин можна було зібрати

об’єкт, в процесі декомпозиції необхідно визначити всі види зв’язків частин між собою.

При створенні дуже складних об’єктів процес декомпозиції виконується багаторазово: кожен блок, у свою чергу, декомпонують на частини, доки не отримають блоки, які порівняно легко розробити. Цей метод розробки отримав назву покрокової деталізації.

Результат декомпозиції зазвичай представляють у вигляді схеми ієрархії, на нижньому рівні якої мають порівняно прості блоки, а на верхньому – об’єкт, що підлягає розробці. На кожному ієрархічному рівні опис блоків виконують із певним ступенем деталізації, абстрагуючись від несуттєвих деталей. Для кожного рівня використовують свої форми документації та свої моделі, що відображають сутність процесів, які виконуються кожним блоком. Для об’єкта в цілому, як правило, вдається сформулювати лише найзагальніші вимоги, а блоки нижнього рівня мають бути специфіковані так, щоб із них можна було зібрати працюючий об’єкт. Іншими словами, чим більше блок, тим абстрактнішим має бути його опис. За дотримання цього принципу розробник зберігає можливість осмислення проєкту і, отже, може приймати найбільш правильні рішення на кожному етапі, що називають локальною оптимізацією. Слід пам’ятати, що поняття складного об’єкта у міру вдосконалення технологій змінюється, і те, що було складним вчора, не обов’язково залишиться складним завтра.

Отже, основою блочно-ієрархічного підходу є декомпозиція та ієрархічне впорядкування. Важливу роль відіграють такі принципи:

– несуперечність (контроль узгодженості елементів між собою);

– повнота (контроль на присутність зайвих елементів);

– формалізація (строгість методичного підходу);

– повторюваність (необхідність виділення однакових блоків для здешевлення та прискорення розробки);

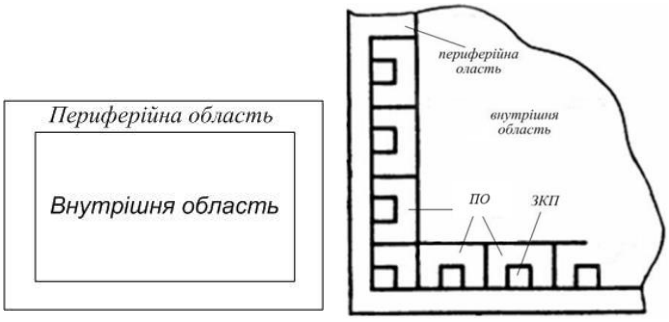
– локальна оптимізація (тобто оптимізація у межах рівня ієрархії).

Сукупність мов моделей, постановок завдань, методів описів деякого ієрархічного рівня прийнято називати рівнем проєктування. Кожен об’єкт у процесі проєктування зазвичай доводиться розглядати з декількох сторін. Різні погляди на об’єкт проєктування прийнято називати аспектами проєктування. Крім того, що використання блочно-

ієрархічного підходу уможливлює створення складних систем, він також спрощує перевірку працездатності, як системи загалом, так і окремих блоків; забезпечує можливість модернізації систем, наприклад, заміни ненадійних блоків із збереженням їх інтерфейсів. Необхідно зазначити, що використання блочно-ієрархічного підходу стосовно програмних систем стало можливим лише після конкретизації загальних положень підходу та внесення деяких змін у процес проєктування.

При використанні блочно-ієрархічного підходу до проєктування проєктовану систему розділяють на ієрархічні рівні. На верхньому рівні використовують найменш деталізоване уявлення, що відображає лише найзагальніші риси та особливості проєктованої системи. На наступних рівнях рівень деталізації опису зростає. Задача конструювання матричних ВІС полягає в переході від заданої логічної схеми до її фізичної реалізації на основі базового матричного кристала (БМК), коли вихідні дані являють собою опис логічної схеми на рівні бібліотечних логічних елементів. У такому випадку необхідно отримати конструкторську документацію для виготовлення працездатної матричної ВІС.

Базовий кристал має вигляд прямокутної багатошарової пластини фіксованого розміру. На рис. 1.2, *а* показано периферійну та внутрішню області БМК. У периферійній області розташовуються зовнішні контактні площадки (ЗКП) для здійснення зовнішнього під'єднання та периферійні осередки (ПО) для реалізації буферних схем (рис. 1.2, *б*).



а) *б)*

Рисунок 1.2 – Основні області БМК (*а*) та структура периферійної області (*б*)

У внутрішній області кристала матричним способом розташовуються макроосередки (МО) для реалізації елементів проєктованих схем (рис. 1.3). Проміжки між макроосередками використовуються для електричних з'єднань. При такому розташуванні макроосередків область для трасування розбивається на горизонтальні та вертикальні канали, а в межах макроосередка розташовуються осередки для реалізації логічних елементів.

Рисунок 1.3 – Структура внутрішньої області

У канальній матриці осередки розташовуються у вигляді масивів, при чому між стовпцями є вільні області, які називаються каналами. За відсутності каналів, осередки розташовуються у вигляді одного великого масиву (рис. 1.4).

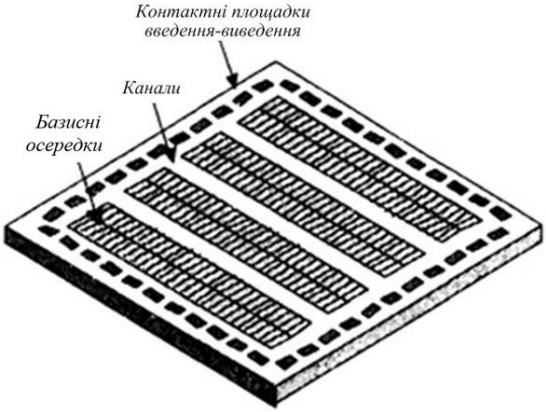


Рисунок 1.4 – Структура БМК

Реалізувавши між осередками належні міжз'єднання, можна отримати прості логічні вентилі, а об'єднавши кілька базових осередків, можна отримати більш складні схеми.

Процес проєктування матричних ВІС можна представити у вигляді таких етапів:

– моделювання функціонування об'єкта;

– розробка топології;

– контроль результатів проєктування та доопрацювання;

– випуск конструкторської документації.

Необхідно зазначити, що ще на етапі проєктування необхідно забезпечити правильне функціонування матричної ВІС, тому що вона є не налаштованим та не ремонтоздатним об'єктом. Для цього використовують два способи. Один з них – це створення макета матричних ВІС на основі дискретних елементів, а інший – математичне моделювання. Перший спосіб пов'язаний з великими витратами часу і високою вартістю, тому макет використовується тоді, коли він спеціально не розробляється, а вже існує. Другий спосіб вимагає створення ефективної системи моделювання схем. На етапі контролю перевіряється адекватність отриманого проєкту вихідним даним. Заключним етапом проєктування є випуск конструкторської документації.

Загальний підхід до вирішення проєктування НВІС полягає в розбитті вихідної задачі проєктування на підзадачі. Як приклад можна навести етап розбиття проєктованої схеми, коли мінімізується число міжблочних зв'язків. Потім об'єкт проєктування поділяється на фрагменти і проєктування кожного фрагмента ведеться самостійно. Таким чином, можна говорити про ієрархічний підхід до проєктування НВІС. Процес вирішення поставленої задачі при цьому замінюється багаторазовим рішенням аналогічних завдань меншої розмірності.

При блочно-ієрархічному проєктуванні НВІС використовують такі рівні описів об'єкта:

– алгоритмічний (основна увага приділяється поведінковому опису НВІС);

– структурний (НВІС представляється у вигляді сукупності основних блоків та з'єднань між ними);

– рівень системи команд (визначаються команди та правила їх виконання апаратурою);

– рівень міжрегістрових передач (основними елементами системи служать регістри);

– рівень логічних вентилів (НВІС представляється сукупністю вентилів та тригерів, а також з'єднань між ними);

– схемний рівень (елементами НВІС є транзистори, діоди, резистори тощо, а опис містить відомості про їх з'єднання);

– рівень фотошаблонів (елементи схемного рівня перетворюються в детальну топологію в тому вигляді, в якому вони будуть реалізовані в кристалі мікросхеми).

1.3 Технологічний маршрут проєктування ВІС

Стосовно проєктування ВІС за допомогою САПР можна ввести поняття технологічного маршруту проєктування ВІС, під яким розуміють будь-який узгоджений між собою набір програм з ППП різних етапів, що забезпечує проєктування даної конкретної схеми по всьому циклу. Залежно від особливостей проєктування конкретних ВІС поняття технологічного маршруту може бути введено і для набору програм у межах одного ППП. Маючи ППП для вирішення завдань на різних етапах проєктування, можна формувати технологічні маршрути, що забезпечують високу якість проєктування в прийнятні терміни.

Наприклад, технологічний маршрут для розрахунку параметрів компонентів формується на підставі таких положень:

– склад ППП у конкретному технологічному маршруті визначається розробником залежно від вимог до параметрів компонента і наявності технологічного процесу його виготовлення;

– у повному технологічному маршруті повинні бути програми, щоб забезпечити вирішення завдань моделювання, аналізу, оптимізації та статистичного аналізу заданого компонента ІС;

– сумарний машинний час, що витрачається на розрахунок компонента, значною мірою залежить від схеми об'єднання програм і порядку їх функціонування, а також від правильного вибору математичної моделі компонента та економічності використовуваних у програмах числових методів.

Різноманітність технічних вимог до ВІС, технологічних процесів їх виготовлення, методик проєктування привело до великої різноманітності маршрутів проєктування. Для реалізації проєкту необхідно розробити індивідуальну систему міжз'єднань транзисторів, тобто щоб реалізувати ВІС на базовому кристалі достатньо спроєктувати та виготовити тільки фотошаблони для змінних шарів комутації. Виготовлену таким чином напівзамовну ВІС називають матричною (МаВІС). Зростання складності проєктів ВІС виключає можливість використання ручних методів для проєктування напівзамовних ВІС.

Технологічний маршрут проєктування МаВІС складається з таких етапів: розробка принципової електричної схеми МаВІС; розробка топології змінних шарів та контроль проєкту МаВІС; розробка технічної документації (програм тестової перевірки працездатності МаВІС та керуючих програм для виготовлення фотошаблонів).

Розробка принципової електричної схеми здійснюється за допомогою підсистеми функціонально-логічного моделювання. Інженер розробляє ескіз принципової електричної схеми, потім описує її на спеціальній мові. Такий опис є одним з розділів завдання для моделювання ВІС на ЕОМ. Текст завдання проходить синтаксичний та семантичний контроль з видачею діагностичних повідомлень про помилки. Після виправлення помилок у вихідному описі починається процес моделювання МаВІС.

Етап функціонально-логічного моделювання завершується, коли при всіх вхідних впливах на виходах схеми виходять необхідні сигнали (реакції на дії). Таким чином, змодельований текст опису схем стає еталонним, а система впливів та реакцій – вихідною інформацією для програм топологічного проєктування.

Етап розробки змінних шарів топології може бути реалізований трьома способами – шляхом ручного проєктування з наступною машинною обробкою, автоматизованого проєктування з подальшою ручною дорозводкою та шляхом автоматичного синтезу. При ручному проєктуванні конструктор відповідно до креслення схеми розробляє ескіз топології МаВІС. Потім ескіз кодується, інформація обробляється і передається на етап контролю.

При автоматизованому проєктуванні вихідною інформацією є еталонний текст опису принципової електричної схеми. Якщо при такому

проєктуванні деяке число зв’язків виявилося нерозведеним, то необхідно змінити вручну розміщення елементів та повторити процедуру трасування. Технологічний маршрут проєктування МаВІС наведено на рис. 1.5.

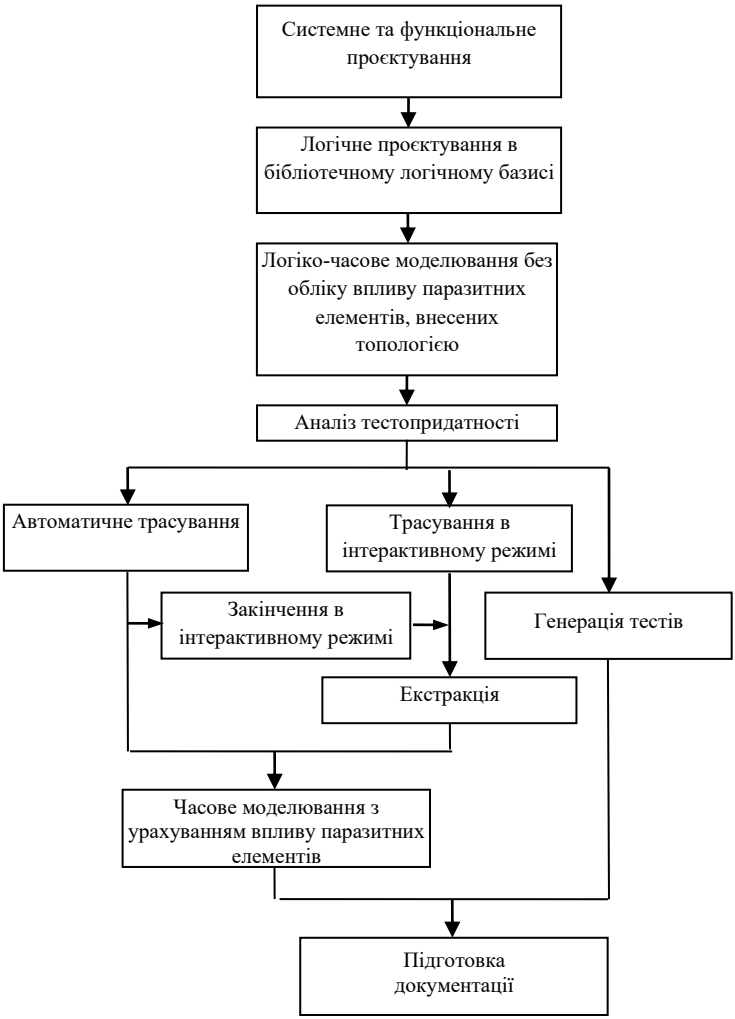


Рисунок 1.5 – Технологічний маршрут проєктування ВІС на БМК

При автоматичному синтезі досягається повна реалізація топології проєкту в автоматичному режимі й тому відпадає необхідність у проведенні контролю. Внаслідок того, що цей процес може бути реалізований при тому ж інформаційному забезпеченні, що й логічне моделювання, то з'являється широка можливість досліджувати працездатність проєкту МаВІС з урахуванням його топологічної реалізації. Для цього за допомогою спеціальної програми-екстрактора з топології відновлюється електрична схема МаВІС з урахуванням міжз'єднань. Ці структури перетворюються в елементи затримки, якими довизначається вихідна електрична схема. Потім при тих же впливах проводиться повторне моделювання схеми. Таким чином, вдається визначати працездатність проєкту МаВІС до етапу виготовлення, що неможливо при інших способах проєктування топології.

Вибір технологічного маршруту топологічного проєктування визначається числом елементів і числом їх виводів. Найбільш поширеним критерієм оцінки якості отриманої топології є критерій максимуму щільності заповнення базового кристала, що дорівнює відношенню числа елементів МаВІС до числа осередків (посадочних місць) кристала. При щільності заповнення 90...100 % застосовують ручне проєктування, при щільності 80...90 % – автоматизоване, а при щільності менше 80 % застосовують автоматичний синтез. До класу повністю замовних ВІС можна віднести схеми, що реалізують проєкти власного застосування. Такими проєктами можуть бути однокристальні ЕОМ, мікропроцесорні ВІС, спеціальні співпроцесори, процесори та ін. Такі проєкти містять довільні логічні побудови.

Методологія проєктування замовних ВІС відрізняється від методології проєктування напівзамовних ВІС. Напівзамовні ВІС найчастіше проєктуються зверху вниз (спадне проєктування), тобто від функціонального опису до реалізації на рівні бібліотечних логічних елементів. Замовні ВІС проєктуються знизу до верху (висхідне проєктування), тобто від проєктування логічних елементів до функціональної реалізації. При такій методології досягаються оптимальні параметри на кожному ієрархічному рівні представлення проєкту.

В якості технічних показників замовних ВІС використовується тактова частота, площа кристала, на якому реалізований проєкт,

потужність розсіювання на одиницю площі кристала, ступінь інтеграції. Для того, щоб параметри були оптимальними, методологія проєктування може змінюватися. Якщо не ставляться жорсткі вимоги до тактової частоти та площі кристала, то можна використовувати автоматичні системи проєктування, які реалізують методологію спадного проєктування, як і при проєктуванні напівзамовних ВІС. Таке проєктування виконується за допомогою інтегрованих САПР (ІНСАПР). Технічні засоби таких САПР будуються за дво- або трирівневим ієрархічним принципом: на першому рівні застосовують суперЕОМ (висока швидкодія та велика оперативна і довгострокова пам'ять); на другому – середні або міні ЕОМ, які обслуговують окремі термінали, пристрої та абонентські пульти, на третьому – міні- або мікроЕОМ, що є індивідуальним робочим місцем користувача САПР.

ІНСАПР є найпоширенішими системами проєктування ВІС. Прикладне ПЗ ІНСАПР містить проблемно-орієнтовані програмні модулі (ПРОПМ), керуючі програми та банки даних. Кожен модуль має вхідний блок (відповідальний за обробку мови опису завдання), блок розв'язання (відповідальний за обробку математичної моделі) та вихідний блок (відповідальний за видачу або передачу результатів роботи блоку розв'язання). Дані ПРОПМ об'єднуються в єдину САПР. Різні ПРОПМ можуть створюватися різними організаціями та інтегруватися в систему, зберігаючи індивідуальні вхідні та вихідні протоколи. У цьому випадку користувач через інтерфейс викликає з банку даних відповідний ПРОПМ та вирішує задачу проєктування ВІС. На рис. 1.6 наведено типовий маршрут проходження замовної ВІС в ІНСАПР.

Окремі ПРОПМ можуть об'єднуватися в ППП цільового призначення зі своїми вхідними мовами в межах даного пакета. Таке об'єднання відіграє величезну позитивну роль у створенні рентабельних та ефективних САПР різного призначення та структури. Крім того, інтегрування йде як по числу розв'язуваних проблем, так і за кількістю та можливостям підключення ПРОПМ. ІНСАПР необхідні для створення і дослідження нових схемно-технологічних рішень на підприємствах-виробниках ВІС різного призначення. Вони інтегрують досвід фахівців-розробників ПРОПМ за багато років. Природно, що такі системи вимагають дуже розвинених обчислювальних засобів.

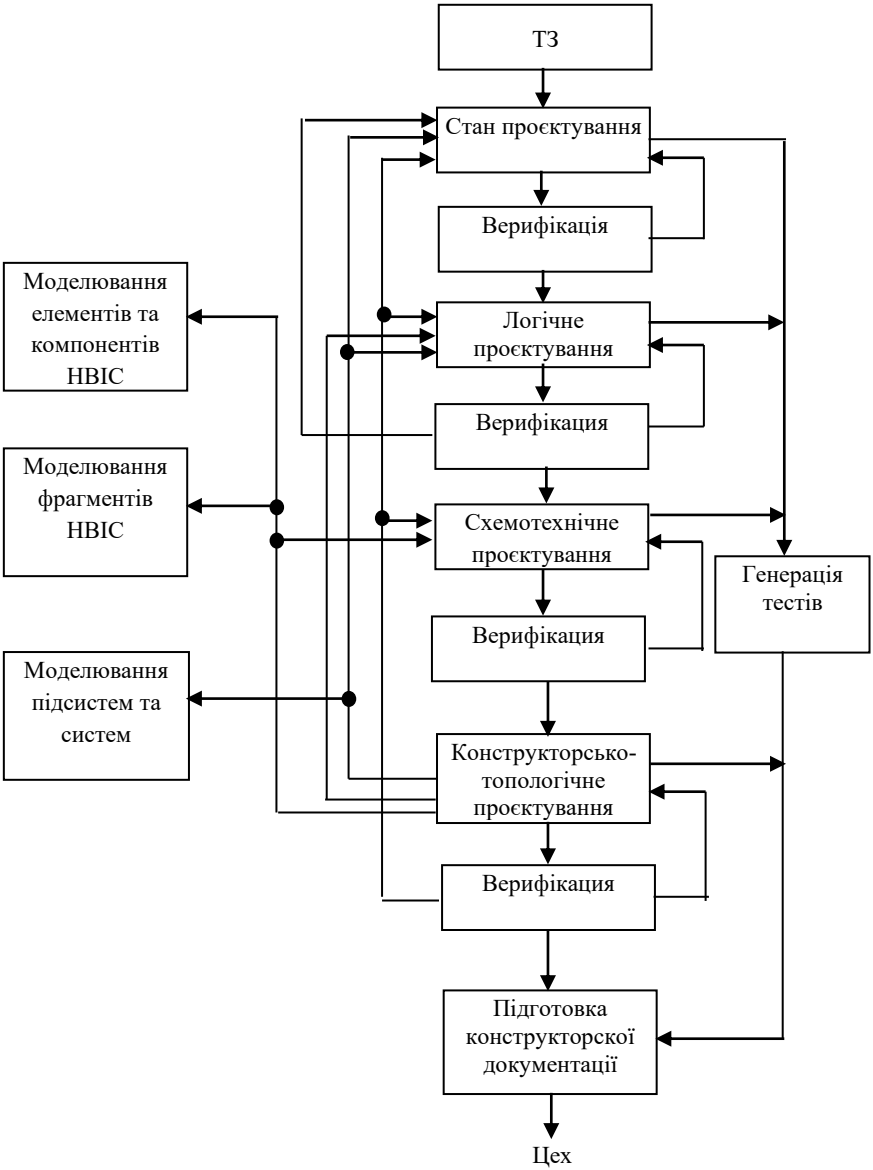


Рисунок 1.6 – Типовий маршрут проходження проєкту ВІС

У складі спеціалізованих мікросхем можуть бути різні блоки, такі як

замовні схеми, схеми на стандартних осередках, схеми на вентильних матрицях, схеми з програмованими сполуками. Головною відмінністю при проєктуванні цих блоків є ступінь свободи вибору при розробці топології. Наприклад, для замовних схем свобода вибору обмежується тільки правилами проєктування топології. В блоках на стандартних осередках (які являють собою раніше розроблені типові блоки) свобода вибору обмежена їх висотою і місцем розташування з'єднань. У блоках на вентильних матрицях топологія елементів строго задана. У схемах з програмованими з'єднаннями топологія задана повністю, а в процесі налаштування проєктувальник створює необхідні з'єднання. Розглянемо послідовність проведення робіт при функціонально-логічному проєктуванні ВІС на БМК згідно зі стратегією висхідного проєктування.

Перший етап полягає в проведенні декомпозиції вихідної логічної схеми на окремі функціональні блоки. З бібліотеки банку даних вибираються елементи, які можуть бути використані в проєктованій ВІС та з них вибирається сукупність функціональних блоків і вузлів, які підлягають розробці. Другий етап – це логічне проєктування на осередках БМК функціональних вузлів. Третій етап – функціональне проєктування окремих блоків. Четвертий етап полягає в розробці логічного проєкту ВІС.

Як було зазначено раніше, існує два види стратегії проєктування: спадне і висхідне. Основним принципом при проєктуванні є одноразова розробка топології компонента кожного типу. Процес використання стратегії спадного проєктування починається з розгляду старшої схеми, а завершується отриманням топологій підсхем самого нижнього рівня. Особливість стратегії такого проєктування полягає в тому, що вона дозволяє цілеспрямовано розподіляти ресурси конструкції кристала та при проєктуванні підсхеми враховувати розташування інших підсхем. Однак, ця стратегія може призводити до недопустимо грубих оцінок розмірів топології підсхеми.

Згідно зі стратегією висхідного проєктування спочатку розробляються топології підсхем самого нижнього рівня, які потім використовуються для розробки топології підсхем більш високих рівнів ієрархії. Важливою перевагою цієї стратегії є те, що вона дозволяє на кожному етапі знати розміри топології всіх необхідних компонентів. До переваг обох

стратегій належить конструктивна нерозривність функціональних вузлів, що створює передумови для використання бібліотек топологій типових підсхем та зниження трудомісткості проєктування, що дозволяє застосовувати більш складні методи проєктування окремих компонентів.

Головним недоліком розглянутих стратегій є порівняно низька ефективність використання площі кристала, тому що при проєктуванні недостатньою мірою враховується специфіка конструкції кристала і топологія підсхеми. В наш час використовуються комбіновані методи, основані на поєднанні принципів обох стратегій з включенням етапів ітераційного покращення рішень. Також одним із недоліків стратегій є те, що їх застосування виявляється виправданим лише в окремих випадках, тому що НВІС на БМК характеризуються високою щільністю елементів.

Для ефективного заповнення площі кристала доцільно допустити, щоб підсхеми одного і того ж типу мали різну форму топології, а забезпечити це можливо шляхом розміщення на кристалі кожної підсхеми по частинах. Отже, ще до розміщення кожну підсхему необхідно розбити на частини (блоки). Мінімальним блоком є логічний елемент. При розробці більш складних ВІС в умовах повної невизначеності, тобто коли відсутня логічна схема, слід застосовувати спадне проєктування.

Розглянемо порядок розв’язання завдань при розробці логічного проєкту ВІС згідно з технологією спадного проєктування. В якості підготовчої стадії здійснюється архітектурне проєктування цифрового пристрою. На даній стадії встановлюють склад і технічні вимоги на ВІС, що входять у пристрій.

На першому етапі проводиться розробка укрупненої схеми пристрою, виконується декомпозиція пристрою на окремі ВІС, апріорно встановлюються вихідні технічні вимоги на кожну ВІС на підставі технічних характеристик пристрою в цілому.

На другому етапі проводиться уточнення алгоритму функціонування і технічних вимог на кожну ВІС, проводиться налагодження кожної алгоритмічної моделі ВІС на відповідних функціональних тестах, потім налагоджені моделі збираються в структурну схему пристрою. Потім розробляються функціональні тести для перевірки правильної роботи пристрою, виконується спільне моделювання пристрою на моделях ВІС. За результатами моделювання уточнюються зв’язки між ВІС і

коригуються технічні вимоги до кожної ВІС. Результати моделювання накопичуються в базі даних САПР. При цьому необхідно зазначити, що розробка і виготовлення різних ВІС, що входять в пристрій, можуть відбуватися нерівномірно – одні ВІС можуть бути спроєктовані і виготовлені, інші можуть знаходитися на стадії проєктування. Тому завжди можна повернутися на етап архітектурного проєктування і провести спільне моделювання пристрою на моделях ВІС різної точності.

Друга стадія – це архітектурне проєктування ВІС. На цьому етапі проводиться розробка і налагодження укрупненої схеми ВІС:

– складається укрупнена схема, яка складається з функціональних блоків;

– складаються алгоритмічні моделі та функціональні тести для перевірки кожного з цих блоків;

– проводиться налагодження алгоритмічних моделей блоків, потім вони збираються та утворюють структурну модель ВІС.

Після описаних вище дій виконують порівняння отриманих результатів з результатами алгоритмічного моделювання ВІС на попередньому етапі. Якщо ці результати збігаються, значить, алгоритмічна і структурна моделі ВІС повністю налагоджені, в іншому випадку доведеться доопрацьовувати або алгоритмічні моделі ВІС, або алгоритмічні моделі функціональних блоків.

Третя стадія – це функціональне проєктування ВІС. На першому етапі проводиться розробка і налагодження функціональних схем блоків (складаються функціональні схеми блоків; складаються алгоритмічні моделі вузлів та функціональні тести, проводиться їх налагодження; складається структурна модель схеми блоків і виконується верифікація; отримані результати порівнюються з результатами алгоритмічного моделювання, проведеного на попередньому етапі, при цьому, якщо результати не збігаються, проводиться доробка або структурної моделі вузла, або функціональних тестів). На другому етапі проводиться розробка і налагодження функціональної схеми ВІС. Заключною стадією проєктування є логічне проєктування ВІС.

Таким чином, у процесі розробки логічного проєкта ВІС за методикою висхідного і спадного проєктування створюються:

– сукупність алгоритмічних моделей вузлів, блоків і ВІС в цілому (в

разі спадного проєктування);

– логічна схема ВІС, розкрита до рівня осередків БМК і подана у вигляді ієрархічного пофрагментного опису взаємно вкладених один в одного багатополюсників;

– ієрархія функціональних тестів для вузлів, блоків та ВІС.

На основі проєкту, який розробляється на етапах логічного і топологічного проєктування, відбувається створення реальних зразків ВІС. Потім на етапі функціонального контролю перевіряється правильність роботи зразка. Тому дуже важливо вже на ранніх етапах проєктування враховувати питання тестування на математичній моделі або на реальному зразку, при якому виявляються несправності ВІС шляхом аналізу станів її виходів на певних наборах вхідних сигналів. Успішне вирішення цієї задачі визначає її найважливіші характеристики (бездефектність проєктування, надійність, стійкість роботи та ін.).

Розрізняють два види тестування ВІС:

1) функціональне – здійснюється на всіх етапах розробки логічної схеми (мета функціонального тестування – перевірити правильність функціонування спроєктованої схеми і відповідність вимогам технічного завдання);

2) функціональний контроль правильності роботи зразків ВІС після їх виготовлення.

Особливість функціонального тестування полягає в тому, що процес розробки схеми багато в чому залишається наближеним і сильно залежить від досвіду розробника, що може привести до непередбачуваності можливих помилок. Для успішного вирішення цієї задачі необхідно створити ієрархію функціональних тестів відповідно до ієрархії фрагментів, з яких складається ВІС. Кожна окремо взята функціональна одиниця (наприклад, акумулятор, лічильник, регістр зсуву) являє собою відносно простий пристрій, для якого можливо побудувати повний тест для перевірки його функціонування. На основі вузла розробляються блоки, логічна функція яких значно складніше. При цьому немає необхідності виконувати повне функціональне тестування кожного вузла, який входить в блок, тому що воно вже було виконано раніше при його розробці.

Особливу увагу при тестуванні необхідно приділяти перевірці коректності зв'язків функціональних вузлів всередині блоку, а також перевірці власних внутрішніх станів блоку. Аналогічно при розробці функціональних тестів на ВІС особливу увагу приділяють перевірці коректності зв'язків між блоками ВІС. Метою функціонального контролю є констатація наявності або відсутності несправностей в конкретному зразку ВІС, які порушують його функціонування. Тобто необхідно встановити відповідність виготовленого зразка проєкту ВІС.

Несправність ВІС – це стан, викликаний непрацездатністю одного або декількох її елементів. Функціональний контроль містить одне дуже серйозне протиріччя між якістю тесту і часом тестування ВІС, що й визначає вартість випробувань.

При функціональному контролі необхідно розробити сукупність контролюючих тестів, призначених для перевірки всіх станів ВІС. Можливими шляхами розв'язання задачі такого контролю є: розробка методів проєктування тестопридатності ВІС, спрямованих на підвищення рівня тестованості ВІС; розвиток методів вбудованого самоконтролю ВІС; розвиток методів автоматичного синтезу та мінімізації контролюючих тестів; розробка методів аналізу якості тестів.

Для оцінки якості тестів застосовують методи моделювання несправностей на ЕОМ, сутність яких полягає в порівнянні результатів моделювання справної схеми та безлічі несправних схем, які отримані шляхом почергового введення в справну схему заданих несправностей. Якщо в тестовій послідовності існує хоча б один тест, який виявляє невідповідність логічних станів виходів справної та несправної схем, то така несправність вважається покритою. На даний час активно розробляються методи, які враховують ймовірні фактори.

1.4 Імітаційне моделювання

Внаслідок того, що реальні системи схильні до випадкових впливів, то виникає необхідність в дослідженні поведінки таких систем. Для цього застосовується імітаційне моделювання, яке в наш час є найбільш ефективним засобом дослідження складних систем. Блок імітаційного моделювання входить в типові архітектури САПР. При імітаційному моделюванні імітуються елементарні події, які відбуваються в системі зі

збереженням логіки їх взаємодії, тобто математична модель відтворює процес функціонування у часі. Імітаційне моделювання характеризується такими перевагами: дає можливість досліджувати особливості функціонування реальної системи; дозволяє вирішувати складні задачі; істотно скорочує вартість і тривалість випробувань у порівнянні з натурним експериментом; є практично реалізованим методом для дослідження складних систем.

Імітаційне моделювання дозволяє вирішувати задачі вибору структури та оцінки впливу різних параметрів, що становить основу САПР. Основна цінність імітаційного моделювання полягає в тому, що в його основі лежить методологія системного аналізу. Вона дозволяє здійснити дослідження системи за схемою операційного дослідження, що включає такі етапи: змістовна постановка задачі; розробка концептуальної моделі; розробка і програмна реалізація імітаційної моделі; перевірка адекватності моделі та оцінка точності результатів моделювання; планування експериментів; прийняття рішень. Це дозволяє використовувати імітаційне моделювання як універсальний підхід для прийняття рішень в умовах невизначеності, а також на практиці застосовувати основні принципи системного підходу для вирішення практичних завдань (враховуючи в моделях фактори, які важко формалізуються). Однак, до імітаційного моделювання слід вдаватися тільки в тих випадках, якщо розробка інших видів моделей не дає задовільних результатів. Це пов’язано з тим, що реалізація імітаційної моделі та проведення експериментів на ній є трудомісткою задачею. Слід звернути увагу на деяку неоднозначність в термінології моделювання, а саме на математичне, імітаційне та статистичне моделювання. Найбільш широкий клас моделей включає в себе математичне моделювання.

Математична модель – це абстрактна модель, представлена на мові математичних співвідношень, яка може бути задана у вигляді аналітичних залежностей або алгоритмів. До цього класу моделей належать диференціальні та інтегральні рівняння та ін.

Імітаційне моделювання – це метод конструювання моделі системи та постановка експериментів на моделі. Як правило, за імітаційну модель береться її програмна реалізація на ЕОМ, а імітаційне моделювання зводиться до проведення експериментів з цією моделлю шляхом

виконання прогонів програми на деякій множині даних. Імітаційна модель працює так само, як і система. Проводиться імітаційне моделювання з метою визначення такої структури та параметрів системи, при яких забезпечуються найкращі показники якості функціонування системи. З математичної точки зору імітаційна модель може розглядатися такою, що складається з рівнянь, які розв'язуються шляхом простеження еволюції їхніх рішень на певному відрізку часу.

Статистичне моделювання – це різновид імітаційного моделювання імовірнісних систем, що базується на методі статистичних випробувань. Розробку імітаційної моделі зазвичай починають з простої вихідної моделі, яку в міру уточнення вихідних даних і характеристик системи ускладнюють і адаптують до нових умов. У той же час імітаційна модель повинна залишатися досить наочною, рівень її деталізації вибирається з урахуванням цілей моделювання, обмежень і вихідних даних. Таким чином, імітаційна модель повинна бути наочною, багаторівневою, адаптивною, розвиватися ітеративним способом, ускладнюватися та коригуватися в процесі її створення. Програмування та налагодження моделі доцільно проводити поетапно, з подальшим збільшенням програмних модулів.

Методологія системного аналізу стосовно до імітаційного моделювання включає кілька етапів.

1. Формулювання проблеми і постановка завдання. На цьому етапі імітаційне моделювання використовується при розробці підсистем різних автоматизованих систем управління (АСУ) та пов'язане з методами аналізу та синтезу виробничих процесів, системами збору, обробки і відображення інформації. Як приклади, в яких може використовуватися імітаційне моделювання, можна навести такі задачі: вибір та обґрунтування технологічного маршруту обробки деталей; вибір організаційної структури управління цехом; аналіз системи прийняття рішень при управлінні цехом підприємства; аналіз роботи системи збору, передачі та обробки інформації; аналіз часу доступу користувачів до інформаційної бази; вибір складу комплексу технічних засобів для обробки інформації в АСУ та ін. З цього переліку задач видно, що імітаційна модель будується як багатоцільова, що дозволяє вирішити кілька різних задач. Слід зазначити, що при формулюванні проблеми, для

вирішення якої розробляється модель, в першу чергу визначають цілі моделювання, потім проводять цілеспрямоване обстеження об'єкта моделювання. В результаті, на змістовному рівні отримують опис основних характеристик системи, вхідні та вихідні параметри, їх взаємозв'язок, зовнішні дії на систему, визначаються основні критерії та обмеження.

2. Розробка концептуальної моделі, тобто абстрактної моделі, яка виявляє причинно-наслідкові зв'язки, властиві досліджуваному об'єкту. Її розробка починається зі складання змістовного опису процесу функціонування об'єкта, який здійснюється відповідно до принципів системного аналізу: формулюються критерії системи або цільові функції, визначається структура системи, її вхідні і вихідні змінні, зовнішні впливи, внутрішні параметри системи. Необхідно зазначити, що неправильна концепція, покладена в основу моделі, неправильні припущення про взаємозв'язок змінних і параметрів призведуть до того, що виконання всіх наступних етапів з побудови імітаційної моделі будуть безглуздими та призведуть до невиправданих витрат.

Можна сказати, що при розробці концептуальної моделі фактично вирішується задача структурної та параметричної ідентифікації в широкому сенсі слова. Важливим моментом є вибір необхідного рівня деталізації опису процесів функціонування системи, тобто визначення ступеня точності опису реального процесу для отримання достовірної інформації шляхом моделювання. Однак, необхідно враховувати, що хоча більш детальна модель буде більш точною та має більш широку сферу застосування, але вона буде дорожча. Тому ступінь деталізації системи при побудові моделі повинен визначатися на основі принципів доцільності. На етапі розробки моделі визначаються вимоги до вхідних даних. Частина даних може бути отримана з технічної документації системи, офіційних звітів та довідників. При цьому, для новостворених систем вихідні дані можуть не існувати – в цьому випадку для їх отримання використовують експертні оцінки, одержувані шляхом опитування групи фахівців в даній області. Необхідно пам'ятати, що отримання вихідних даних є відповідальним етапом та використання неправильних або неточних даних призведе до неправильних результатів моделювання. Слід зазначити, що процес побудови концептуальної

моделі є ітеративним: спочатку будується узагальнена модель, потім модель деталізується, а в подальшому концептуальна модель виноситься на обговорення з користувачем. Такий процес розробки дозволяє виявити допущені неточності та провести коригування. На завершенні даного етапу наводиться логіко-математичний опис модельованої системи з урахуванням динаміки її функціонування. Ті частини моделі, які піддаються математичному опису в аналітичному вигляді, подаються у вигляді аналітичних залежностей, а інші частини моделі представляються у вигляді докладного словесного опису або алгоритму. Після побудови концептуальної моделі перевіряється її адекватність об'єкту, який моделюється.

3. Розробка та програмна реалізація імітаційної моделі. На цьому етапі в першу чергу проводиться остаточна формалізація концептуальної моделі. Така формалізація передбачає побудову структури імітаційної моделі. Структура моделі може бути побудована в термінах мови імітаційного моделювання або пакета прикладних програм.

4. Перевірка адекватності імітаційної моделі.

5. Планування експериментів на моделі.

6. Оцінка точності результатів моделювання.

7. Інтерпретація результатів моделювання та прийняття рішень.

Перераховані етапи моделювання є взаємопов'язаними, а сама процедура побудови імітаційної моделі є ітеративною. Це пов'язано з тим, що після виконання кожного етапу здійснюється перевірка правильності моделі та в разі невідповідності моделі та об'єкта здійснюється повернення до попередніх етапів з метою її коригування.

1.5 «Система на кристалі»

Згідно із законом Мура, кількість транзисторів на кристалі НВІС з кожним роком збільшується приблизно на 60 %. З певного моменту часу обладнання, яке розміщувалося на одній друкованій платі, стало можливим помістити на одному кристалі. При чому це стає вигідним завдяки зменшенню загальної вартості, кількості необхідних мікросхем, енергоспоживання, підвищенню надійності.

Таким чином, на одному кристалі розміщується не тільки конкретний функціональний пристрій, наприклад, центральний мікропроцесор, а й

інші, такі як АЦП, ОЗП, ПЗП, блоки цифрової обробки сигналів, інтерфейсні вузли та ін., що доповнюють його до завершеної системи блоків.

«Система на кристалі» (СНК) або «System-on-a-Chip» (SoC) – це НВІС, яка інтегрує на кристалі різні функціональні блоки, що утворюють закінчений виріб. Оскільки в більшості випадків СНК являє собою цифрову НВІС, яка може також містити ряд аналогових блоків, то для її проєктування використовуються ті ж методи і засоби, що і для НВІС. Структура типової СНК показана на рис. 1.7.

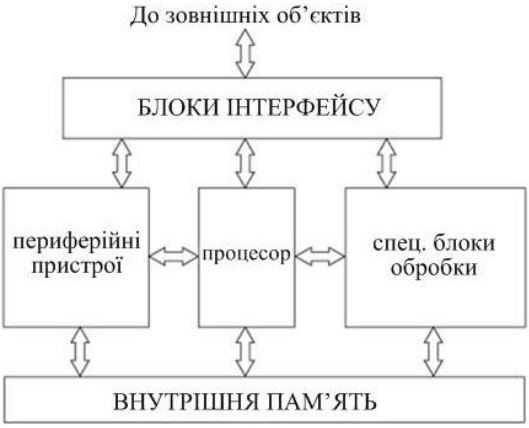


Рисунок 1.7 – Структура «Система на кристалі»

СНК може включати цифрові та аналогові блоки. Основним блоком зазвичай є процесор, а також є спеціалізований блок цифрової обробки сигналів, який виконує специфічні функції. Як периферійні пристрої можуть виступати АЦП; модулі пам'яті можуть входити до складу СНК, а можуть підключатися як зовнішні блоки. Сучасні СНК відрізняються від мікроконтролерів тільки наявністю спеціалізованих блоків обробки даних. При проєктуванні використовують складнофункціональні блоки, такі як процесори, таймери та ін. Ці блоки називають IP-модулями (Intellectual Property) та їх представляють або у вигляді топологічних фрагментів, які можуть бути безпосередньо реалізовані у фізичну структуру кристала, або у вигляді моделей на мові опису апаратури.

Проєктування СНК почали застосовувати на етапі освоєння

технологічних процесів виробництва інтегральних схем рівня 350…250 нм, коли стало можливим здійснити інтеграцію всіх основних цифрових компонентів кінцевого продукту на одному кремнієвому кристалі. Після того, як технологія такого рівня надала можливість інтеграції всієї системи на одному кристалі, а наступні технологічні процеси надали можливість інтеграції ще більш складних систем, виникло питання, як зробити так, щоб повністю заповнити цей кристал. Як основний метод для підвищення продуктивності проєктування було запропоновано повторне використання інтелектуальної власності, починаючи з маленьких блоків (наприклад, інтерфейси з шиною) та далі до великих блоків (наприклад, ядра вбудованих процесорів). Потім, переходячи до наборів блоків, які були попередньо інтегровані та верифіковані – до так званих SoC «платформ».

У теперішній час на зміну спадному проєктуванню приходить спіралеподібна методологія проєктування. У цій моделі перед тим, як перейти до наступного етапу, повинні бути завершені всі проєктні задачі на поточному етапі. При цьому проєктування виконується одночасно за чотирма основними напрямками: розробка програмного забезпечення, розробка коду, логічний синтез, фізичний синтез. У процесі роботи групи розробників постійно обмінюються результатами проєктування.

У процесі розвитку проєктування НВІС типу СНК виникли проблеми, пов’язані з програмно-апаратним та методологічним забезпеченням процесу проєктування. Справа в тому, що з поліпшенням проєктних норм мікроелектронних виробів зростає вартість підготовки виробництва, а, отже, і ціна ризику технічної та ідеологічної помилок. В наш час більшість фірм-розробників НВІС є компаніями, які не мають власної виробничої бази, і виготовлення кристалів здійснюється на потужностях спеціалізованих кремнієвих фабрик, які надають розробникам бібліотеки для логічного синтезу. Фахівці виконують остаточне доопрацювання фотошаблонів та виготовлення кристала. У цьому випадку актуальним стає використання складно-функціональних блоків, які являють собою повністю відпрацьовані елементи, які використовуються в СНК. Це призводить до зростання складності процесу проєктування. Для успішного виконання будь-якого складного проєкту необхідно організувати його ієрархічну декомпозицію. Наприклад, на рівні

регістрових передач проєкт являє собою сукупність арифметичних і логічних вузлів. Логічний рівень описує проєкт на рівні логічних вентилів та тригерів. Набір шаблонів топологічних елементів кристала відповідає геометричному рівню. У цьому випадку поведінку схеми описують системою логічних рівнянь. Зазвичай при проєктуванні ВІС використовується спадна модель проєктування. Таке проєктування має на увазі мінімальну взаємодію між розробниками на різних фазах проєкту.

На першому етапі проєктування повинна бути виконана розробка і аналіз технічних вимог та проведено попереднє моделювання. На виході цього етапу повинна бути проведена повна функціональна перевірка технічних вимог. Слід зазначити, що при цьому задача моделювання ускладнюється необхідністю розробки моделі тестових впливів.

На другому етапі здійснюється опис проєкту за допомогою мови VHDL або Verilog на рівні регістрових передач RTL (Register transfer level). За описом на рівні регістрових передач формується список кіл, який враховує затримки на бібліотечних елементах. Потім розробляється і оптимізується розводка кристала шляхом розміщення цих елементів та міжз'єднань оптимальним чином. Після розробки топології можна повторно виконати формування файла затримок і подальше часове моделювання, яке буде враховувати вплив міжз'єднань. Потім кристал можна передавати до виробництва та проводити подальше тестування зразків. Недоліком такої методології проєктування є підвищена небезпека появи помилок при збільшенні складності проєкту. Крім цього, зрозуміти, наскільки розроблювана ВІС задовольняє пропоновані до неї вимоги, буде ясно тільки у самому кінці процесу проєктування.

Сучасна мікроелектронна технологія забезпечує варіанти реалізації СНК на базі програмованих логічних інтегральних схем (ПЛІС) високої інтеграції (FPGA), а також у вигляді замовної НВІС (ASIC). На даний час існує можливість розміщувати на кристалі практично будь-який за складністю проєкт. Однак, технології верифікації проєкту помітно відстають від технологій та обчислювальних можливостей систем проєктування (як відомо, ступінь інтеграції сучасних НВІС зростає експоненціально). Згідно з традиційним загальноприйнятим алгоритмом проєктування, який полягає в тому, що кожен блок проєктується та верифікується окремо, а спільна верифікація відбувається після

об’єднання в більш великі блоки, є висока ймовірність виявлення помилки на одному з останніх етапів інтеграції. Це призводить до додаткових витрат часу і ресурсів на внесення змін. Розв’язання цієї проблеми полягає у вдосконаленні засобів тестування, а також зміні методики верифікації.

1.6 Проєктування цифрових пристроїв на ПЛІС

Скорочення обсягів компонентів схеми і підвищення ступеня інтеграції – це загальна тенденція розвитку сучасної елементної бази мікроелектроніки. При цьому, як відомо, зменшення лінійних розмірів активних елементів на кристалі приводить до поліпшення їх функціональних характеристик. Ця тенденція привела до створення технології виробництва НВІС з мінімальною проєктною нормою 45 нм. Це майже в два рази менше від умовної верхньої нанотехнологічної межі в 100 нм, починаючи з якої прийнято відносити об’єкти до області наноінженерії. Кількість компонентів у сучасних системах перевищує сотні мільйонів, тому методи проєктування наносистем мають суттєві відмінності від методів, які раніше застосовувалися для проєктування електронних систем, а це можливо тільки при використанні потужних обчислювальних комплексів і систем автоматизації проєктування.

Однією з комбінаторних задач є задача топологічного проєктування об’єктів наноінженерії. Ця задача належить до класу складних задач, тому що час, необхідний для отримання оптимального рішення з ростом ступеня інтеграції НВІС як об’єкта наноінженерії, зростає в експоненційній залежності. Для її вирішення застосовують евристичні алгоритми, використання яких приводить до рішення, близького до оптимального. Вартість кристалів НВІС складається з вартості їх проєктування (яку можна знизити, якщо стандартизувати його топологію) та вартості виробництва (яку можна знизити, підвищивши ступінь інтеграції), які зростають майже пропорційно числу пластин, запущених у виробництво. Ці два чинники є взаємно виключаючими. Зокрема, для кристалів, що виготовляються дрібними серіями при великій номенклатурі виробництва, важливо знизити роль першого з названих чинників.

На теперішній час майже всі мікропроцесори проєктуються

інтерактивним способом, тобто таким, в якому на кожній ітерації результати роботи САПР оцінюються та контролюються розробником. При цьому необхідно зазначити, що якщо застосовувати тільки засоби САПР, то це призведе до збільшення розмірів кристала. Однак, при «ручному» проєктуванні при досить великих витратах часу цей вид проєктування дозволяє отримати кристал НВІС менших розмірів.

Блочно-ієрархічне проєктування топології передбачає поділ логічної структури НВІС за принципом спадного проєктування, тобто логічну схему подають у вигляді окремих блоків з прийнятними розмірами. Потім блоки розміщують на кристалі та виконують трасування міжз'єднань між ними від нижнього до верхнього рівня.

На рис. 1.8 показана топологія НВІС – спочатку на кристалі розташовують блоки 1 – 5, а потім у блоці 5 розташовують блоки А, Б, В та ін.

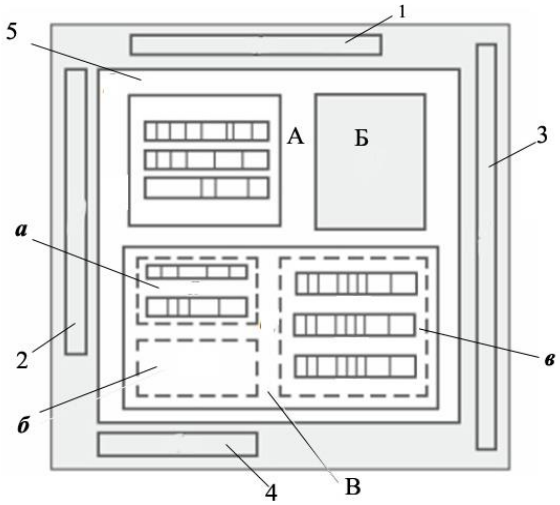


Рисунок 1.8 – Топологія НВІС

Етапу топологічного проєктування об'єктів наноінженерії передує логічне проєктування, яке в наш час реалізується з використанням опису схем на мові VHDL. Раніше при розробці схем (наприклад, тригерів) використовувалися булеві рівняння. Однак, у тому випадку, коли проєкт

містить тисячі тригерів, велике число логічних рівнянь робить їх застосування непрактичним. Тому в наш час розробники віддають перевагу графічному поданню проєкта. Це розширює можливості застосування логічних рівнянь, використовуючи на додаток до тригерів додаткові кола, що дозволяє розміщувати більше число компонент з меншою кількістю зусиль, ніж потрібно було б в методі булевих рівнянь.

Одним із найцікавіших напрямків сучасної цифрової електроніки є ПЛІС, які в міру свого ускладнення та збільшення розмірів починають користуватися великим попитом. ПЛІС являють собою цифрові інтегральні мікросхеми, що складаються з програмованих логічних блоків і програмованих з'єднань між цими блоками. Можливість конфігурувати ці пристрої дозволяє розробникам вирішувати безліч різних завдань. Перші ПЛІС з'явилися в середині 1980-х років. У той час вони використовувалися в основному для вирішення деяких завдань обробки даних, для створення прототипів замовних мікросхем або для створення спеціальних стендів, на яких перевірялася фізична реалізація нових алгоритмів. Основна перевага ПЛІС перед іншими спеціалізованими схемами – це малий час виготовлення необхідних замовних варіантів схем. Зникає необхідність звертатися до виробників ІМС для нанесення металічної маски і установки кристала в корпус.

Багато сучасних ПЛІС містять вбудовані мікропроцесорні ядра, високошвидкісні інтерфейси введення/виведення, використовуються в пристроях зв'язку, в радіолокації та обробці зображень. Вартість ПЛІС набагато нижча від вартості замовних інтегральних схем, до того ж внесення змін до пристрою при використанні ПЛІС не викликає особливих труднощів та істотно скорочує терміни виходу таких пристроїв на ринок.

ПЛІС – це за своєю суттю замовна мікросхема. Для програмування використовуються програматори, різні налагоджувальні середовища, за допомогою яких можна задати бажану структуру цифрового пристрою або у вигляді принципової електричної схеми, або у вигляді програми на мовах опису апаратури.

Проєктування цифрових пристроїв на ПЛІС можна розділити на такі етапи:

1) системний етап – весь проєкт розбивається на частини,

визначаються їх призначення та взаємозв'язок, а також приймається рішення про способи реалізації цих частин;

2) структурно-алгоритмічний і функціонально-логічний – ці етапи базуються на ітераційному введенні та верифікації описів паралельно функціонуючих процесів, кожен з яких реалізує заданий алгоритм.

При цьому пристрій можна описувати такими способами:

– з використанням мов опису (VHDL, Verilog);

– схемотехнічним (за допомогою програми візуального проєктування, після закінчення якого схема перетворюється в мовний опис);

– графічним поданням в спеціалізованому редакторі, що забезпечує перетворення отриманого графічного подання в мовний опис;

– описом комбінаційної логіки за допомогою таблиць станів та карт Карно;

3) конструкторсько-технологічний етап, який поділяється на такі підзадачі:

а) підзадача синтезу – перетворення вихідного схемотехнічного опису пристрою в опис, який може бути оптимально реалізований на обраній ПЛІС. На цій стадії використовуються різні методи оптимізації опису, спрямовані на досягнення найкращих результатів з точки зору мінімуму необхідних ресурсів кристала і споживаної потужності;

б) глобальне розміщення – створення найкращих умов для локального розміщення та трасування;

в) локальне розміщення – детальне призначення логічних ресурсів макроділянок частинам схеми;

г) трасування – визначення зв'язків між логічними блоками.

Після виконання кожної підзадачі проводиться верифікація отриманого опису, для чого застосовуються різні засоби моделювання та аналізу. Після виконання трасування та верифікації результатів автоматично може бути згенеровано файл, який містить інформацію про комутацію та функціональності всіх ресурсів кристала. На заключному етапі маршруту проєктування виконується програмування ПЛІС та верифікація пристрою.

Проєктування пристроїв на основі ПЛІС виконується із застосуванням спеціалізованих САПР, яке полягає в послідовному використанні наданих програмних засобів (такий процес називається маршрутом

проєктування). На рис. 1.9 наведено узагальнений маршрут проєктування, який використовується САПР компанії Xilinx.

Рисунок 1.9 – Узагальнений маршрут проєктування

Проєктування починається зі створення проєкту, потім вибирається одна з доступних програм синтезу, програма моделювання, мова опису. Необхідно зазначити, що маршрут реалізації проєкту типу CPLD відрізняється від маршруту реалізації FPGA наявністю етапу зборки. Після стадії трасування генерується файл конфігурації ПЛІС, після чого розробник задає послідовність початкового завантаження конфігурації в ПЛІС та програмує всі необхідні пристрої.

ПЛІС діляться на дві основні групи:

1) CPLD (Complex Programmed Logic Device) – мають невисоку ціну та невелику кількість ресурсів, складаються з блоків логічних вентилів,

об’єднаних програмованою комутаційної матрицею;

2) FPGA (Field Programmed Gate Array) – містять більше ресурсів та більш складні блоки у порівнянні з CPLD. Головною відмінністю FPGA від CPLD є потреба при включенні живлення виконувати завантаження конфігурації. Тобто для FPGA необхідно мати зовнішній постійний запам’ятовуючий пристрій (ПЗП), а CPLD зберігають логічну структуру навіть після відключення живлення. Однак, ряд останніх моделей FPGA також мають модифікації з енергонезалежною конфігураційною пам’яттю.

На сьогоднішній день лідерами у виробництві FPGA є компанії Altera та Xilinx. Програмне забезпечення Xilinx давно відомо як засіб, що забезпечує високу продуктивність розроблюваних пристроїв і є простим та недорогим продуктом.

Altera є одним з найбільших розробників ПЛІС та концентрується на розробці схем і модулів на основі мов VHDL, Verilog і AHDL. Її основні вироби – це програмовані мікросхеми, а також послуги з перетворення проєктів під ПЛІС для масового виробництва. Компанія також випускає програми для розробки вбудованого ПЗ для ПЛІС і компілятори під ядро процесора власної розробки. На сьогоднішній день програмні продукти Max+Plus II та Quartus II.

Програмний продукт Max+Plus II – це середовище для розробки цифрових пристроїв на базі ПЛІС, яке забезпечує виконання таких етапів: введення проєктів; підготовку даних для програмування ПЛІС; верифікацію проєктів; програмування ПЛІС.

Програмний продукт Quartus II надає повний цикл для створення СНК, а також надає розробнику широкі можливості при компіляції проєкту. Для програмування FPGA використовуються мови опису апаратури HDL (Hardware Description Language). Серед них найбільш популярні Verilog та VHDL (Very high speed integration circuits Hardware Description Language). Основне завдання цих мов – описувати електричні цифрові кола подібно графічним схемам.

Традиційні мови програмування не придатні для опису апаратури, тому що вони створювалися для опису алгоритмів, а це зовсім не те ж саме, що цифрове коло. Мови програмування характеризуються синтаксисом і семантикою. Синтаксис визначає граматичні правила

написання програм та використання конструкцій мови, а семантика – значення цих конструкцій.

Програми, написані на традиційних мовах програмування, моделюють послідовний процес. Внаслідок того, що результат виконання окремих операторів часто впливає на наступні оператори, то не можна міняти їх місцями. VHDL та Verilog багато в чому схожі, але мають різний синтаксис і розрізняються в деяких деталях.

У VHDL закладена можливість ієрархічного проєктування. Ця мова була розроблена з метою формального опису логічних схем для всіх етапів розробки електронних систем.

Мова VHDL використовується в таких цілях:

– автоматизації перетворення вихідного опису схеми в опис на більш низькому рівні;

– опис поведінки цифрових пристроїв у часі та при зміні вхідних впливів;

– опис структури цифрових пристроїв з різним ступенем деталізації;

– опис тестових впливів при моделюванні пристроїв;

– моделювання цифрових пристроїв.

Засобами мови VHDL можливе проєктування на різних рівнях абстракції, за синтаксисом ця мова великою мірою схожа на універсальні мови програмування. Однак, для опису складних об’єктів використовуються специфічні елементи мови (сигнали, порти, конфігурації та ін.).

Verilog – мова опису апаратури, що використовується для опису та моделювання електронних систем. Це досить проста мова, схожа з мовою програмування С.

VHDL та Verilog вважаються звичайними мовами для розробки цифрових пристроїв, у той час як SystemVerilog представляє розширену версію Verilog (воня виглядає гібридним розширенням Verilog – містить HDL-опис апаратури і об’єктно-орієнтований підхід до програмування). У Verilog та VHDL розробники можуть представити необхідну функціональність у вигляді тексту програми, а потім провести моделювання і відкоригувати програму.

VHDL – це строго типизована і детермінована мова, та більш детальна у порівнянні з Verilog. Завдяки своїй структурі, VDHL на ранніх стадіях

процесу розробки дозволяє відстежувати більше помилок, ніж Verilog. На теперішній час інженери ефективно застосовують як VHDL, так і Verilog та SystemVerilog для своїх розробок СНК.

1.7 Теоретичні відомості про електромагнітну сумісність

Електромагнітна сумісність (ЕМС) електронного пристрою – це його здатність функціонувати з необхідною якістю сумісно з іншими пристроями в умовах можливого впливу непередбачених завад, не створюючи при цьому недопустимих завад іншим пристроям. Забезпечення ЕМС технічних пристроїв є самостійним науково-технічним напрямком, який охоплює численні аспекти радіотехніки, електроніки та електротехніки. Проблема ЕМС належить до найважливіших в електроенергетиці, а її економічний характер свідчить про величезні збитки, що виникають внаслідок недотримання її вимог.

Технічні характеристики електронного пристрою можна поділити на функціональні, які визначають властивості пристрою, та такі, що впливають на ЕМС. Для забезпечення ЕМС пристроїв необхідно раціонально вибирати частоти радіоканалів для певних радіослужб, виконувати розрахунки з урахуванням функціональних характеристик та характеристик ЕМС. Технічні заходи поділяються на конструкторсько-технологічні та схемотехнічні.

Загальна концепція методології створення та функціонування пристроїв з урахуванням ЕМС базується на системному підході до вирішення задачі забезпечення ЕМС, тобто на виявленні усіх факторів, які впливають на неї при сумісному функціонуванні розроблюваної системи або пристрою, а також визначення причин виникнення цих факторів і встановлення зв’язків та взаємодій між ними. Системний підхід призводить до багатоплановості рішення задачі забезпечення ЕМС на всіх рівнях та комплексності рішень у напрямку підвищення завадозахищеності та зниження енергії завад в їх джерелах та середовищі поширення.

Електромагнітна завада – це електромагнітний, електричний або магнітний процес, який викликає небажані зміни у корисному сигналі при його передачі, прийомі або перетворенні до заданого вигляду. У системах електропостачання електромагнітні завади існують завжди: усі

елементи електроустаткування генерують електромагнітні завади, тобто є джерелами емісії електромагнітних завад. В області ЕМС під джерелом мається на увазі лише джерело завади, яка виникає неумисно або через недостатності прийнятих заходів технічних та організаційних мір.

Електромагнітні завади можна класифікувати на стаціонарні (завади від антени радіопередаючого пристрою), індустріальні (завади від електронних, електротехнічних, радіотехнічних пристроїв, що використовуються у побуті, промисловості, транспорті та ін.), контактні, природні.

Для того, щоб спланувати ЕМС системи, повинні бути відомі електромагнітні обставини, механізми зв'язку і чутливість приймача завад між двома електричними колами. Напруженість ближніх магнітних та електричних полів у вільному просторі зворотно пропорційна квадрату відстані від елемента, що збуджує поле. При аналізі впливу електромагнітних завад особлива увага приділяється причині їх виникнення, шляхів поширення, особливостям впливу на різні об'єкти, а також заходам, методам і засобам захисту і придушення цих завад.

За типом поширення виділяють просторові та кондуктивні завади. Перші характеризуються впливом через електромагнітне поле, що випромінюється і поширюється в просторі, а другі проникають в апаратуру по провідникових каналах зв'язку та електроживлення. Кондуктивними називають завади, що виникають при зв'язку через загальний опір, наприклад, через заземлюючі шини або джерела живлення. При цьому струми від різних схем протікають через загальний опір, падіння напруги на якому від кожного струму буде завадою для інших схем.

За місцем розташування джерела завад щодо досліджуваного електронного пристрою розрізняють на зовнішні (викликані процесами в інших пристроях), внутрішні (виникають як електромагнітні явища та зв'язки, не передбачені схемою та конструкцією пристрою) та власні (є шумами компонентів, пов'язаними з функціонуванням самого пристрою). За типом сигналу завади розрізняють на випадкові та детерміновані. Завади бувають імпульсними, широкосмуговими і вузькосмуговими. Така класифікація дозволяє оцінити завадову обстановку в цілому, проаналізувати та сформувати підхід для усунення небажаного впливу

конкретного виду завад на пристрій.

У технічній літературі користуються спеціальними термінами для характеристики завадової обстановки:

1) приймачем (рецептором) завад є будь-які системи або їх складові частини аж до елементів та окремих компонентів, на яких позначається дія завад;

2) сприйнятливість – це міра реакції приймача на заваду, що характеризує його здатність знижувати якість функціонування під впливом завади;

3) завадостійкість – це властивість приймача зберігати якість функціонування при дії завади, тобто протистояти їй за рахунок схемотехнічних заходів;

4) завадозахищеність – це властивість приймача зберігати якість функціонування та протидіяти завадам за рахунок схемотехнічних, конструктивно-технологічних та інших заходів захисту, що не змінюють принципів дії та побудови приймача.

Вирішення проблеми ЕМС системи зазвичай починається з вивчення електромагнітної обстановки, тобто сукупності електромагнітних, електричних та магнітних полів, а також струмів та напруг завад, які можуть впливати на функціонування цієї системи.

Одним із видів ненавмисних електромагнітних завад є перехресні завади – це небажане явище, при якому при близькому паралельному розташуванні провідників, сигнал на одному з них (джерело завади) викликає зміну сигналу на іншому (приймач завади). Це може спричинити неправильне функціонування електронного пристрою. Дія перехресної завади проявляється на вхідному тракті електронного пристрою як зміна амплітуд або фаз, що складають спектр корисного сигналу. Перехресні завади можуть виникати всередині мікросхем, а також на друкованих платах, роз’ємах, корпусах мікросхем та кабелях. Якщо на друкованих платах є дві або більше сигнальних провідних доріжки на протяжних ділянках, які розміщенні паралельно та близько одна до одної, то через взаємний вплив на входах «пасивних» кіл можуть виникати завади.

Перехресні завади викликаються паразитними ємнісними, індуктивними або провідними зв’язками електричного ланцюга.

Ємнісний зв’язок викликає раптове збільшення струму в лініях, що призводить до відбиття на лініях передачі. Існують також індуктивні зв’язки, що створюються магнітними полями. В основному вони викликають шум електроживлення. Перехресні завади зменшуються пропорційно квадрату відстані, а ступінь впливу пропорційна поверхні плати, тобто і відстані між доріжками. Існує широкий спектр можливих дій щодо зниження перехресних завад, пов’язаних зі зміною дистанції між провідниками, перепорядкування сигналів у провідниках та їх екранування.

Цифрова схема повинна бути спроєктована таким чином, щоб на її виході не з’являлися напруги, які виходять за визначені діапазони. Але за певних обставин це може статися, наприклад, якщо вихід цифрової мікросхеми сильно навантажиться, то вона не зможе генерувати напругу на певних рівнях через обмежену ефективність джерела сигналу. Синхронні шини використовуються для паралельної передачі. Вони характеризуються дуже гарною стійкістю до перехресних завад, які можуть з’явитися тільки в момент зміни стану, причому у вузькому часовому вікні.

Перехресні завади не впливають на якість сигналу, поки приймач досить довго чекає на його стабілізацію перед вибіркою шини. Якщо перехресні завади виникають під час зміни стану сигналу, їхнім єдиним ефектом є джитер (брязкіт фази). Асинхронні та несгруповані сигнали завжди чутливі до перехресних завад.

При проєктуванні високошвидкісної плати необхідно враховувати проблему цілісності сигналу, направляти і перевіряти конструкцію цієї плати. Проблема цілісності сигналу викликає збільшення частоти сигналу, зменшення розміру друкованої плати, збільшення щільності розведення та зменшення товщини проміжного шару, викликане збільшенням кількості шарів плати. Проєктування на рівні плати та на рівні системи є складним процесом і включає такі етапи: проєктування схеми, вибір мікросхеми, схематичне проєктування, компонування друкованої плати і трасування. Під час проєктування необхідно знайти перехресні завади на різних етапах і вжити заходів щодо їх придушення. Розрахунок перехресних завад дуже складний. На амплітуду сигналу перехресних завад впливає ступінь зв’язку та відстань між провідниками.

У процесі проєктування високошвидкісних друкованих плат потрібно як детальне розуміння теоретичних концепцій, так і постійне накопичення досвіду та постійне вдосконалення теорії. У той же час вміле використання відповідного програмного забезпечення також може скоротити цикл проєктування, що підвищить конкурентоспроможність та відіграє важливу роль в успішному завершенні проєктування.

В залежності від кількості шарів друкованого монтажу плати розділяють на односторонні, двосторонні та багатосторонні. Одно- та двосторонні плати дешевші та прості в виготовленні, але їх електричні характеристики не можливо визнавати задовільними, тому для підвищення швидкодії цифрової апаратури слід віддавати перевагу багатошаровим платам, використання яких в більшості випадків є єдиним випадком реалізації мікромініатюрної апаратури.

Системи провідників, що утворюють лінію зв'язку з локалізації електромагнітного поля, можливо поділити на наступні види: одношарові без екранування, двошарові з екрануючою плоскістю, трьохшарові з двома екранними площинами. Зі збільшенням кількості екрануючих площин в цих системах електромагнітне поле більше зосереджується в області між провідником та екраном, що призводить до зменшення рівня індукційних завад.

Конструювання багатошарових друкованих плат з урахуванням ЕМС може бути зведено до наступних етапів:

– вибір базової структури плати із заданою кількістю сигнальних та потенціальних шарів;

– розрахунок геометричних розмірів ланок, що входять до складу базової структури;

– за наявності ряду базових структур, що відповідають заданій кількості сигнальних та потенціальних сигналів, вибір структури з найменшою товщиною;

– перевірка реалізації плати порівнянням її товщини з технологічно можливою.

До переваг багатошарових друкованих плат з позиції внутрішньої апаратурної ЕМС належить стабільність електричних параметрів ліній зв'язку, можливість забезпечення необхідного хвилевого опору лінії конструкторськими способами, достатній ступінь екранування

сигнальних провідників, які розміщені у внутрішніх шарах плати. Екранування є конструкторським засобом забезпечення послаблення електромагнітного поля завад в межах певного простору. Його мета – підвищити завадозахищеність та забезпечити ЕМС електронних пристроїв. Конструкції, що реалізують вказані вимоги, називаються екранами.

Дія електромагнітних екранів наступна: електромагнітне поле проникає в стінку екрану та збуджує в ній заряди або індукційні струми, власні поля яких накладаються на початкове поле, при цьому частково або повністю компенсують його. Екрани можуть використовуватись як для окремих функціональних вузлів, блоків апаратури, так і для електронного пристрою в цілому. Для виготовлення екранів використовуються матеріали, які мають високу провідність для потоків діючих полів та які за рахунок індукції здатні створювати протидіюче магнітне поле. Основним засобом послаблення кондуктивних завад, що утворюються в колах живлення та комутації постійного та змінного струмів апаратури, є фільтрація. Фільтри знижують ці завади як від зовнішніх, так і від внутрішніх джерел.

Основними вимогами до функціонування фільтру є:

– забезпечення заданої ефективності фільтрації в необхідному частотному діапазоні;

– обмеження допустимого падіння постійної або змінної напруги на фільтру при максимальному струмі;

– обмеження за вимогами техніки безпеки допустимого значення реактивної складової струму на основній частоті;

– забезпечення допустимих нелінійних спотворень напруги живлення, визначаючих вимоги до лінійності фільтру;

– елементи фільтру повинні обиратися з урахуванням номінальних струмів та напруг електричного кола.

Цифрові інтегральні схеми можуть створювати та сприймати завади зовні. Для зменшення завад, які виникають в цифрових логічних схемах, необхідно працювати з сигналами, що мають більше часу наростання та спаду та меншу амплітуду, обмежити кількість одночасно перемикаючих сигналів та використовувати ефективні методи шунтування та заземлення. Для підвищення стійкості схем від зовнішніх завад слід

використовувати повільні схеми синхронізації. Якщо система має довгі провідники, бажано використовувати диференційні передаючі та приймаючі пристрої, що з'єднані симетричними лініями зв'язку, для зниження рівня завад.

Шунтуючі конденсатори є джерелами імпульсного струму, що споживають цифрові схеми при перемиканні, зменшують падіння напруг в колах живлення та заземлення та сприяють фільтрації завад, що утворюються джерелами живлення та заземлення. В діапазоні частот 30 МГц – 1 ГГц тактуючі синхросигнали та їх гармоніки є основною причиною випромінювальних завад. Парні гармоніки можна зменшити, якщо використовувати синхросигнали з 50 % коефіцієнтом заповнення.

Слід намагатися зменшити кількість інтегральних схем, які керуються кожним тактовим синхросигналом. Якщо синхросигнали повинні поступати на декілька плат, в якості буферу бажано використовувати вхідні логічні елементи на тригерах Шмітта, а також обмежити розмах напруги та значення швидкості наростання амплітуди основних синхронізуючих сигналів. Проблему завад можливо вирішити шляхом ретельної синхронізації системи. Для зменшення перехідних струмів, що виникають в джерелах живлення та пристроях заземлення, слід керувати невеликою групою мікросхем за допомогою рознесених тактуючих сигналів.

Розділ 2
ВИКОРИСТАННЯ САПР OrCAD ДЛЯ МОДЕЛЮВАННЯ ПРИСТРОЇВ

Практична робота 1
Висхідне та спадне проєктування

Мета роботи: виконати висхідне та спадне проєктування для ієрархічних пристроїв та оцінити переваги і недоліки кожного виду автоматизованого проєктування.

Створення проєкту в OrCAD Capture

При створенні нового проєкту схеми в *Capture*, рекомендується уникати довгих назв шляхів та імен файлів та використання спеціальних символів для позначення кола, вузла, проєкту або бібліотеки: ?, @, ~, #, &,%, «,!, (), <, =,>, [], *. Для створення нового проєкту запускаємо *Orcad Capture*. В меню *File* вибираємо *New Project*. В діалоговому вікні (рис. 2.1) визначимо назву проєкту як *Lab1*. Вибираємо *Analog or Mixed A/D*, визначаємо каталог, в якому будуть створені файли проєкту.

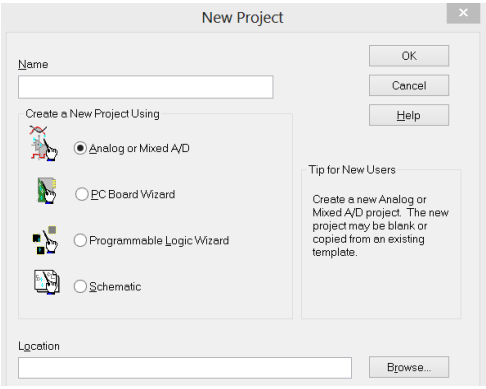


Рисунок 2.1 – Діалогове вікно *New Project*

У вікні *Project Manager* клацнемо правою кнопкою миші (ПКМ) на SCHEMATIC1 та виберемо *Rename* в меню (рис. 2.2). Змінимо назву схемного каталога та сторінки схеми на CODER.

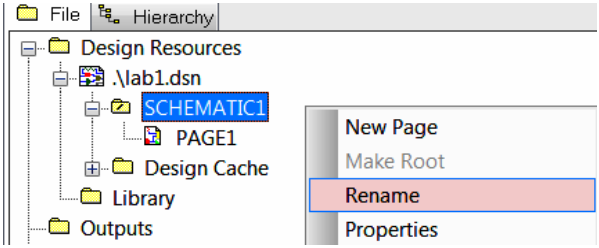


Рисунок 2.2 – Перейменування схемного каталога

Перед початком створення проєкту потрібно визначити характеристики проєкту. В меню *Options* вибираємо *Design Template*. Шрифти для різних об'єктів задаються на вкладці *Fonts*, написи кутового штампа задаються на вкладці *Title Block*, на вкладці *Page Size* встановлюється розмір сторінки схеми (рис. 2.3).

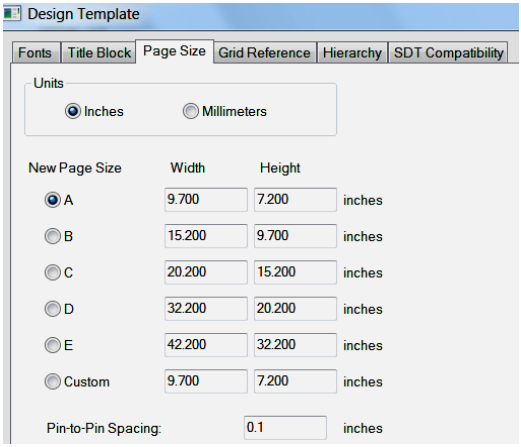


Рисунок 2.3 – Вкладка *Page Size* діалогового вікна *Design Template*

У полі *Pin-to-Pin Spacing* вказується мінімальна відстань між виводами в умовному графічному позначенні (УГП) елементів, а також цей параметр визначає розмір кроку сітки.

Параметри рамки навколо сторінки схеми задаються на вкладці *Grid Reference* (рис. 2.4):

– *Alphabetic* – нумерація полів в алфавітному порядку;

– *Numeric* – нумерація полів у числовому порядку;

– *A scending* – простановка номерів полів рамки (в порядку зростання);

– *Count* – кількість полів на рамці по горизонталі та вертикалі;

– *Width* – ширина рамки по горизонталі та вертикалі;

– *Descending* – простановка номерів полів рамки (в порядку спадання);

– *Title Block Visible* – відображення основного напису на дисплеї (*Displayed*) та при виведенні на друк (*Printed*);

– *Border Visible* – відображення границь сторінки на дисплеї (*Displayed*) та при виведенні на друк (*Printed*);

– *Grid Reference Visible* – відображення меж листа схеми на дисплеї (*Displayed*) та при виведенні на друк (*Printed*);

– *ANSI grid references* – зображення рамки листа схеми за стандартом *ANSI*.

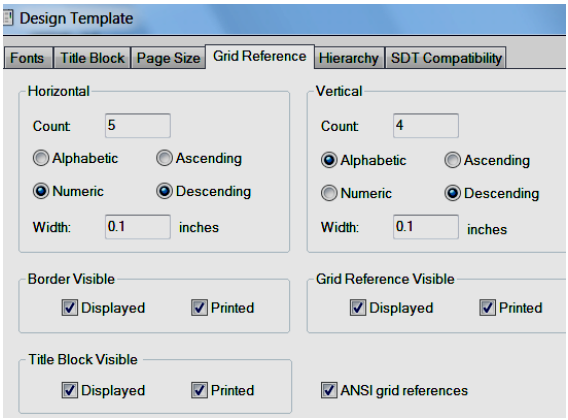


Рисунок 2.4 – Вкладка *Grid Reference* діалогового вікна *Design Template*

Параметри, які приймаються за замовчуванням при створенні нових ієрархічних блоків (*Hierarchical Blocks*) та елементів (*Parts*), вказуються на вкладці *Hierarchy* (рис. 2.5): *Primitive* – примітивні компоненти; *Nonprimitive* – компоненти з ієрархічною структурою.

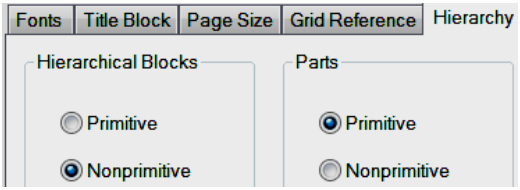


Рисунок 2.5 – Вкладка *Hierarchy* діалогового вікна *Design Template*

Створення плоского (однорівневого) проєкту

Створимо простий плоский проєкт *Coder*. Натиснувши кнопку *Place Part* , в діалоговому вікні виберемо бібліотеку (кнопка *Add Library*), з якої елемент повинен бути доданий, а потім помістимо його на сторінці.

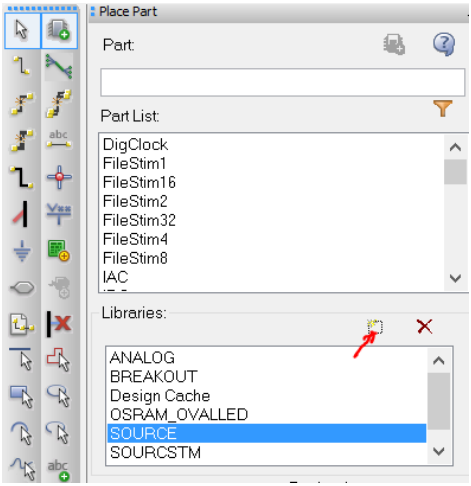


Рисунок 2.6 – Діалогове вікно *Place Part*

За допомогою вікна *Browse File* в каталозі */tools/capture/library/pspice* виберемо бібліотеку 7400.0LB та натиснемо *Open*. Бібліотека 7400 з'явиться в списку *Libraries*. У списку *Part List* виберемо 7404 та клацнемо *OK*. Аналогічним чином розміщуємо інші елементи. Клацнемо ПКМ та виберемо *EndMode*. В меню *Place* виберемо *Wire* та з'єднаємо елементи (рис. 2.7).

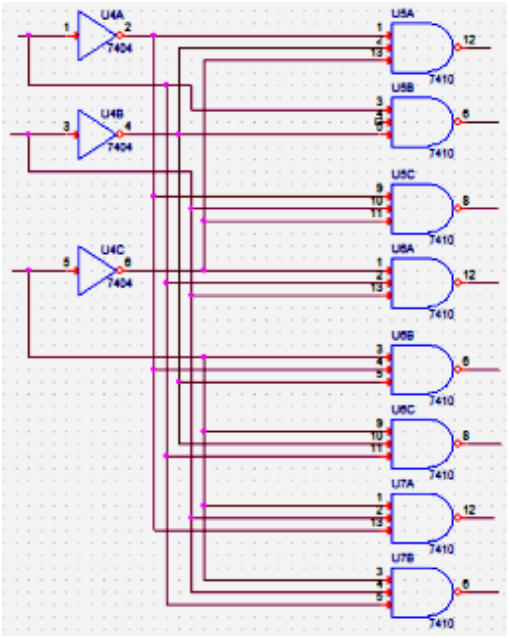


Рисунок 2.7 – З’єднані елементи

Якщо необхідно з’єднати елементи на схемі за допомогою шин, виконуються такі операції:

1. У меню *Place* виберемо *Bus*. Потім у меню *Place* виберемо *BusEntry* для розміщення виводів окремих кіл.

2. Для призначення назв (псевдонімів) шин і кіл, що входять до їх складу, в меню *Place* виберемо *Net Alias*. При призначенні псевдонімів кіл, їх номера автоматично збільшуються на одиницю (наприклад *A1*, *A2*, *A3*). Назва шини записується у вигляді *A* [1…3].

Для додавання вхідних та вихідних портів до проєкту виконаємо таку послідовність операцій:

1. В меню *Place* виберемо *Hierarchical Port* (можна натиснути кнопку *Place Port* на панелі інструментів). Відкриється діалогове вікно *Place Hierarchical Port*.

2. У полі *Libraries* виберемо *CAPSYM*.

3. Добавимо вхідні порти *PORTRIGHT-R* (рис. 2.8).

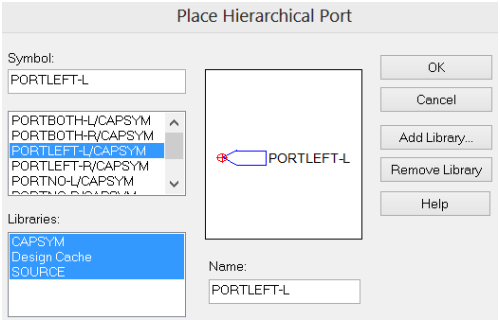


Рисунок 2.8 – Вибір портів

4. Додаємо вісім вихідних портів. Для цього виберемо *PORTLEFT-L* в бібліотеці *CAPSYM*.

5. Для перейменування портів двічі клацнемо по назві порту та в діалоговому вікні *Display Properties* змінимо значення властивостей *Name* вхідних портів на *X1..X3* та вихідних портів на *Z0...Z7* (рис. 2.9). Збережемо Проєкт, натиснувши кнопку *Save*.

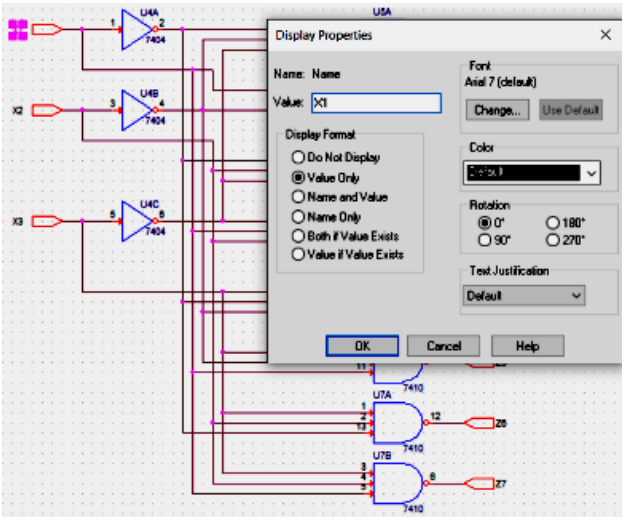


Рисунок 2.9 – Діалогове вікно *Display Properties*

У *Capture* можна створити ієрархічні проєкти, використовуючи один з таких методів: *Bottom-up method* (висхідний метод) або *Top-down method* (спадний метод).

Висхідний метод створення ієрархічного проєкту

При створенні ієрархічного проєкту з використанням висхідної методології необхідно виконати такі операції:

– створити проєкт самого нижчого рівня;

– створити проєкти більш високого рівня (у вигляді ієрархічних блоків).

Розглянемо створення проєкту *DC* за висхідною методологією. Для цього слід виконати такі операції:

1. Створимо проєкт самого нижчого рівня (в нашому випадку це вже створений проєкт *Coder*).
2. У вікні *Project Manager* клацнемо ПКМ на lab0.dsn та виберемо *New Schematic*, потім присвоїмо йому ім’я *DC*.
3. Щоб зробити *Coder* кореневою схемою високорівневого проєкту, клацнемо ПКМ на *DC* та в меню виберемо *Make Root*. Каталог *DC* переміститься вгору, та на ній з’явиться коса риска.
4. Клацнемо ПКМ на *DC* та виберемо *New Page*.
5. У діалоговому вікні *New Page in Schematic* визначимо назву сторінки як *DC* та клацнемо *OK*. Нижче схемного каталога *DC* додається нова сторінка *DC* (рис. 2.10).

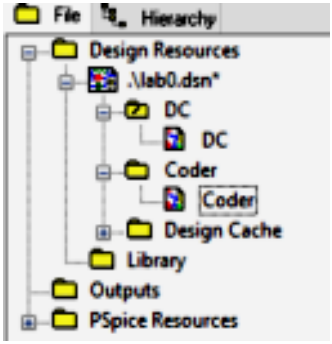


Рисунок 2.10 – Створення нової сторінки

6. Щоб відкрити сторінку для редагування, двічі клацнемо на її імені лівою кнопкою миші (ЛКМ).

7. У меню *Place* виберемо *Hierarchical Block*, в діалоговому вікні якого визначимо позиційне позначення *Reference* як *DC*, *Implementation Type* як *Schematic View*, *Implementation name* як *CODER* та клацнемо *OK*.

8. Зобразимо прямокутник на сторінці схеми. При цьому на сторінці з'явиться ієрархічний блок з вхідними та вихідними портами (рис. 2.11).

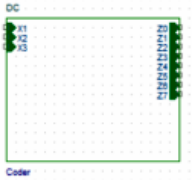


Рисунок 2.11 – Ієрархічний блок

9. Якщо буде потрібно, змінимо розміри блоку (можна переставити вхідні та вихідні порти). Для перевірки правильності ієрархічного блоку клацнемо на ньому ПКМ та виберемо *Descend Hierarchy*. Відкриється створений раніше проєкт *Coder*.

10. Додаємо джерело сигналу. У діалоговому вікні *Place Part* натиснемо кнопку *Add Library*, щоб додати бібліотеку SOURCESTM.OLB (вона розташована в каталозі */tools/capture/library/pspice*.

11. У *Part List* виберемо *DigStim1* та клацнемо *OK* (рис. 2.12). УГП елемента «приклеїться» до курсору.

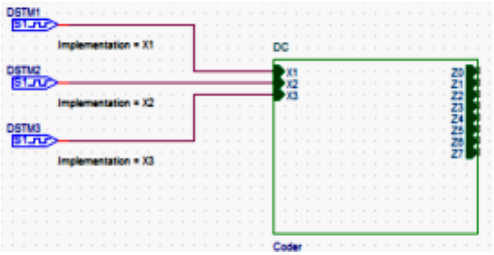


Рисунок 2.12 – Додавання джерел вхідних сигналів

16. Помістимо джерело сигналів поруч із трьома вхідними портами.

17. Клацнемо ПКМ на схемі та виберемо *End Mode*.

18. Визначимо значення *Implementation property* як *X1*, *X2*, *X3*.

19. Вхідні провідники можна об'єднати в шину. Додаємо вихідні порти та змінимо їх імена на *Z0...Z7* (рис. 2.13).

20. Збережемо проєкт.

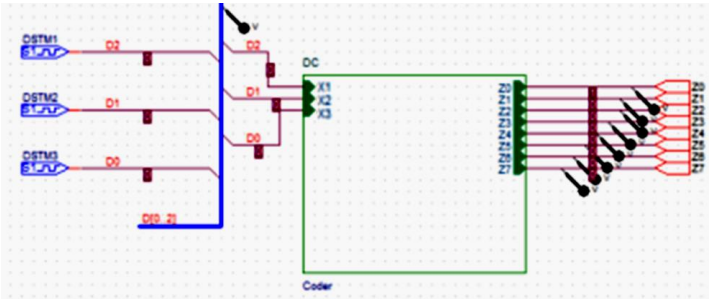


Рисунок 2.13 – Схема дешифратора

Результати моделювання показані на рис. 2.14.

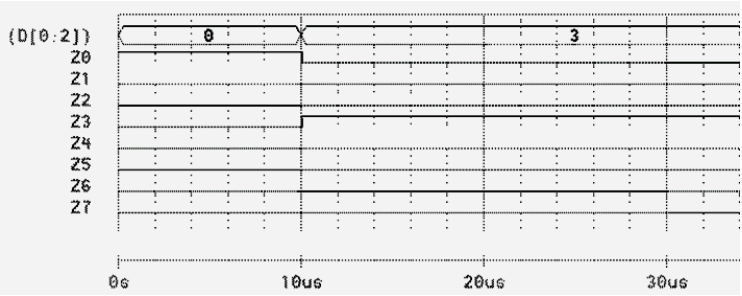


Рисунок 2.14 – Результати моделювання

Спадний метод створення ієрархічного проєкту

При створенні спадного ієрархічного проєкту скористаємося такою послідовністю операцій:

– створимо проєкт верхнього рівня, використовуючи функціональні

блоки, входи та виходи яких відомі;

– створимо проєкт схеми для функціонального блоку, який використовується в проєкті верхнього рівня.

Розглянемо операції, які необхідно виконати для створення проєкту *Coder* за спадною методологією.

Створимо проєкт верхнього рівня:

1. У меню *Place* виберемо *Hierarchical Block* або натиснемо кнопку *Place hierarchical block* панелі інструментів.

2. У діалоговому вікні *Place Hierarchical Block* визначимо позиційне позначення (*Reference*) як *DC, Implementation Type* як *Schematic View, Implementation name* як *CODER*. На відміну від ієрархічного блоку, створеного за висхідною методологією, він не має приєднаних портів.

3. Виберемо ієрархічний блок і в меню *Place* виберемо *Hierarchical Pin*.

4. У діалоговому вікні *Place Hierarchical Pin* визначимо назву виводу як *X1, Type* як *Input, Width* як *Scalar* та натиснемо *OK*.

5. Помістимо вхідний вивід на ієрархічний блок.

6. Додаємо вхідні виводи *X2*, *X3* та вісім вихідних виводів *Z0...Z7*, як показано на рис. 2.15.

7. Закінчимо створення схеми, додавши для цього порти, провідники та джерела сигналів. Збережемо проєкт.

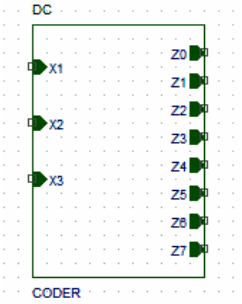


Рисунок 2.15 – Ієрархічний блок з портами

Розробимо проєкт нижчого рівня (в нашому випадку таким проєктом є схема *Coder*). Для цього виконаємо:

1. Клацнемо ПКМ на ієрархічному блоці та в меню виберемо *Descend Hierarchy*.

2. Відкриється діалогове вікно *New Page in Schematic*. Визначимо назву сторінки як *DC*. Нова сторінка схеми з'явиться з трьома вхідними портами *X1, X2, X3* та вихідними портами *Z0...Z7* (рис. 2.16).

Рисунок 2.16 – Сторінка схеми з вхідними та вихідними портами

На цій сторінці можна накреслити схему *Coder*, використовуючи операції, описані раніше при створенні однорівневого проєкту.

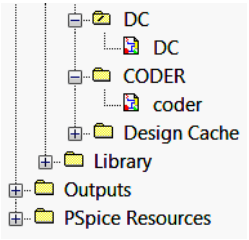


Рисунок 2.17 – Діалогове вікно з доданим каталогом *CODER*

Додавши до ієрархічного блоку джерела вхідних сигналів та вихідні порти, отримуємо схему, аналогічну зображеній на рис. 2.13.

Навігація в ієрархічному проєкті

Щоб переміститися на більш низький рівень ієрархії, клацнемо ПКМ на ієрархічному блоці та виберемо *Descend Hierarchy*. Так само для пересування вгору по ієрархії клацнемо ПКМ та виберемо *Ascend Hierarchy*. Опції меню *Ascend Hierarchy* та *Descend Hierarchy* доступні також у меню *View*.

Працюючи з ієрархічними проєктами, можна робити зміни в ієрархічних блоках, так само як в схемах на самому нижчому рівні. Зберігати ієрархічні рівні зі змінами можна, використовуючи опції *Synchronize*, доступні в меню *View*. Якщо були зроблені зміни на самому нижчому рівні проєкту і необхідно, щоб вони були відображені на більш високому рівні, виберемо *Synchronize Up*. Якщо були зроблені зміни в ієрархічному блоці та необхідно, щоб вони були відображені в усіх зразках блоку, виберемо *Synchronize Across*. Якщо були зроблені зміни в ієрархічному блоці та необхідно, щоб вони були відображені в проєкті самого нижчого рівня, виберемо *Synchronize Down*.

Створення УГП для схеми

Замість того, щоб створювати ієрархічний блок для проєкту *Coder*, можна згенерувати для нього УГП, а потім багаторазово використовувати його в будь-якому іншому проєкті. В цьому розділі згенеруємо УГП для схеми *Coder*, яка створена раніше. Для цього виконаємо операції:

1. У вікні *Project Manager* виберемо курсором каталог *CODER*, в меню *Tools* виберемо *Generate Part* та визначимо місце розташування файла проєкту, який містить схему, для якої має бути створено УГП.

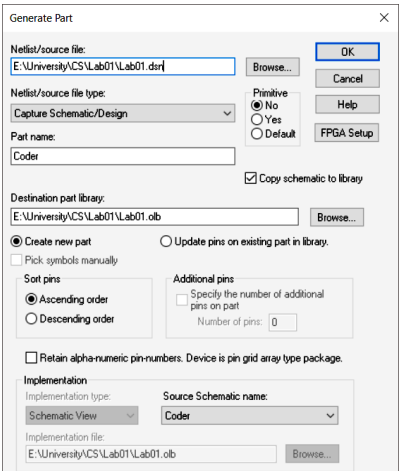


Рисунок 2.18 – Діалогове вікно *Generate Part*

4. У списку *Netlist/source file type* визначимо тип проєкту як *Capture Schematic*; в текстовому полі *Part Name* визначимо назву УГП як *CODER*.

5. Визначимо назву та місце розташування бібліотеки, яка буде містити створюване УГП (наприклад, lab01.olb).

6. Якщо необхідно, щоб вихідна схема була збережена поряд з новим УГП, поставимо позначку *Copy schematic to library*. Переконаємося, що обрано опцію *Create new part*. Для визначення схемного каталога виберемо *CODER* зі списку *Source Schematic name*.

Натиснувши *OK*, згенеруємо УГП *Coder*. Нова бібліотека lab01.olb буде створена та відображена під каталогом *Outputs* у вікні *Project Manager* (рис. 1.19). Він також буде доданий в діалоговому вікні *Place Part*. Можна тепер для додавання схеми *Coder* до будь-якого проєкту використовувати діалогове вікно *Place Part*.

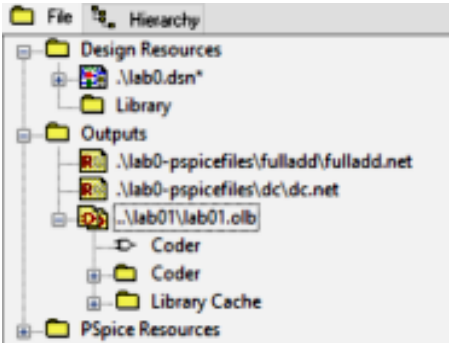


Рисунок 2.19 – Створена бібліотека

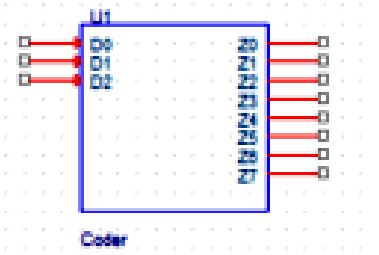


Рисунок 2.20 – УГП дешифратора

Додавання позиційних позначень елемента

Для передачі проєкту схеми в ORCAD *Layout* (програму розведення друкованих плат) всі компоненти в проєкті повинні бути однозначно визначені позиційним позначенням елементів. У *Capture* можна призначити позиційне позначення або ручне, або при використанні команди *Annotate*.

У нашому проєкті анотація на даному етапі не потрібна, тому що за замовчуванням позиційні позначення елемента приєднані до всіх компонентів. Розстановка позиційних позначень виконана автоматично – за замовчуванням *Capture* додає позиційне позначення елемента на всі компоненти, що поміщаються на сторінку схеми. Якщо потрібно, можна відключити цю особливість, виконавши такі операції:

1. У меню *Options* виберемо *Preferences*.
2. У діалоговому вікні *Preferences* виберемо закладку *Miscellaneos*.
3. У полі *Auto Reference* приберемо позначку перед командою *Automatically reference placed parts*.
4. Щоб зберегти ці установки, натиснемо *OK*.

Якщо компоненти в проєкті не мають позиційних позначень, необхідно виконати команду *Annotate*.

Щоб призначити позиційне позначення елемента на компоненти в створеному проєкті за допомогою команди *Annotate*, виконаємо наступні операції:

1. У вікні *Project Manager* виберемо файл lab0.dsn.
2. У меню *Tools* виберемо *Annotate* або натиснемо кнопку *Annotate* на панелі інструментів. Діалогове вікно команди *Annotate* наведено на рис. 1.21, на ньому є такі поля:

Scope (можливості):

- *Update entire design* – оновити позиційні позначення і пакувальну інформацію всього проєкту;
- *Update selection* – оновити позиційні позначення та іншу інформацію обраної частини проєкту.

Action (операції):

- *Incremental reference update* – оновити позиційні позначення та пакувальну інформацію компонентів, у яких замість номера проставлений знак питання «*?*», при цьому номери компонентів

збільшуються на одиницю;

- *Unconditional reference update* – оновлення позиційних позначень і пакувальної інформації всіх компонентів в обраній області;
- *Reset part reference to «?»* – заміна номерів компонентів на «?»;
- *Add Int ersheet Reference* – додавання посилань на інші сторінки;
- *Delete Intersheet Reference* – видалення посилань на інші сторінки.

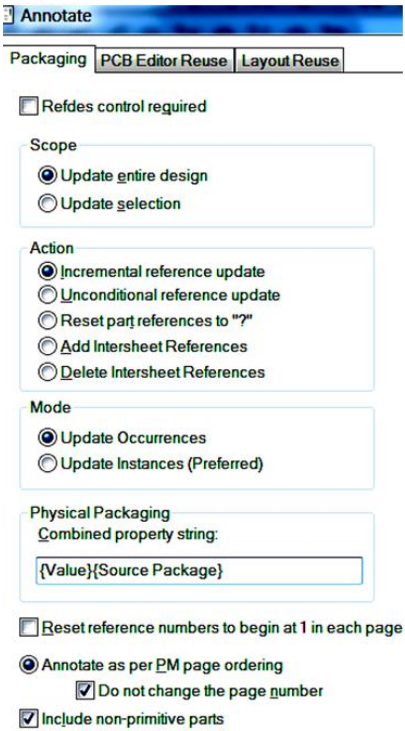


Рисунок 2.21 – Діалогове вікно команди *Annotate*

Mode (режим):

– *Update Occurrences* – оновлення параметрів усіх індивідуальних зразків компонента;

– *Update Instances* – оновлення параметрів компонента та всіх посилань на нього;

– *Physical Packaging* (автоматична упаковка компонентів відповідно до зазначених властивостей);

– *Combined property string* – рядок властивостей;

– *Reset reference numbers to begin at 1 in each page* – починати з одиниці нумерацію позиційних позначень однотипних компонентів на кожній сторінці;

– *Do not change the page number* – не змінювати номер сторінки.

3. У позиції *Packaging* діалогового вікна *Annotate* визначимо необхідність зміни повного проєкту або тільки елемента проєкту. Виберемо опцію *Update entire design*.

4. У полі *Actions* виберемо опцію *Incremental reference update*.

5. Для створеного проєкту виберемо опцію *Update Occurrence*.

Зауваження: коли вибирається опція *Update Occurrence*, можна отримати попередження. Необхідно ігнорувати це повідомлення, тому що для всіх складних ієрархічних проєктів режим *Occurrence* найбільш прийнятний.

6. Для інших опцій приймемо значення за замовчуванням та клацнемо *OK* для збереження обраних установок. Відкриється вікно *Undo Warning message*.

7. Натиснемо *Yes*. Відкриється вікно звіту з повідомленням, що буде створена анотація.

8. Натиснемо *OK*. В проєкті будуть проставлені позиційні позначення компонентів і він буде збережений. Можна розглянути оновлені позиційні позначення на сторінці схеми.

Створення звіту *Cross reference*

Використовуючи *Capture*, можна створити звіт *Cross reference* для всіх елементів в проєкті. Звіт *Cross reference* містить таку інформацію, як назва елементів, їх позиційні позначення та бібліотеки, з яких було обрано елемент.

Для генерації звіту *Cross reference*:

1. У меню *Tools* виберемо *Cross References* або на панелі інструментів натиснемо кнопку *Cross Reference Parts*.

2. У діалоговому вікні *Cross Reference Parts* виберемо опцію *Cross reference entire design*. Якщо необхідно згенерувати звіт *Cross reference*

для специфічного каталога, виберемо перед відкриттям діалогового вікна *Cross Reference Parts* схемний каталог.

3. У полі *Mode* виберемо опцію *Use Occurrences*.

4. Визначимо звіт, який необхідно створити.

5. Якщо необхідно, щоб звіт автоматично був виведений на екран, поставимо позначку в полі *View Output*.

6. Натиснемо OK, щоб згенерувати звіт.

Завдання для виконання практичної роботи

Виконати висхідне та спадне проєктування заданого пристрою згідно з табл. 2.1.

Таблиця 2.1

№ вар.	Пристрій
1	4-розрядний суматор
2	16-розрядний суматор
3	12-розрядний суматор
4	шифратор 8×3
5	3-розрядний регістр
6	мультиплексор 5×1
7	8-розрядний суматор
8	мультиплексор 6×1
9	6-розрядний суматор
10	двійковий лічильник з $К_р$=5
11	двійковий лічильник з $К_р$=3
12	двійковий лічильник з $К_р$=4
13	4-розрядний регістр
14	дешифратор на 6 виходів
15	дешифратор на 10 виходів

Практична робота 2

Створення власного елемента на основі базових компонент

Мета роботи: придбати практичні навички зі створення власного елемента в САПР OrCAD.

У системі OrCAD використовується бібліотечний метод проєктування, який полягає в тому, що пристрій «збирається» з окремих компонент, які об'єднуються в бібліотеки. Користувач може створювати новий опис компонент та включати їх в існуючі бібліотеки, а також може створювати власні бібліотеки. На принциповій схемі компонент подається у вигляді УГП, його прийнято називати символом (*Part, Symbol*). На друкованій платі той же компонент виглядає інакше. Тут головне передати його фізичні розміри та форму, щоб з'ясувати, скільки місця буде потрібно для його розміщення, такий опис називають конструкторським чи фізичним, він зберігається в бібліотеці з розширенням *.lib. Для моделювання проєкту потрібно мати опис функції компонента, тобто алгоритм його роботи. Символи діляться на основні та допоміжні.

Допоміжні символи являють собою невелику групу, куди входять з'єднувачі сторінок, порти ієрархічних блоків, «кутові штампи» та символи підключення кіл живлення. Основні символи поділяються на прості та ієрархічні. Прості символи називають структурними примітивами, і в ході проєктування вони не підлягають деталізації. Примітиви є об'єктами з відомою поведінкою, тобто вони завжди мають функціональний опис.

Ієрархічні символи в OrCAD реалізовані у вигляді ієрархічних блоків (*Hierarchical Block*) або примітивних символів. Ієрархічні символи мають два види опису – зовнішній та внутрішній. При зниженні рівня опису символу, він перетворюється в структуру, показуючи «заховану» в ньому підсхему або VHDL-код. Внутрішній опис у більшості випадків – це підпорядкована схема, яку називають схемою заміщення або еквівалентною схемою. Серед простих символів можна виділити символи реальних та абстрактних компонент. Основна відмінність абстрактних

компонент полягає в тому, що для них не вказується конкретна фізична реалізація. Наприклад, можна зобразити двовходовий мультиплексор, описати його функцію та використовувати потім в проєкті, хоча насправді такої мікросхеми немає. Зважаючи на відсутність даних про фізичну реалізацію, абстрактні компоненти не дозволяють довести проєкт до реального втілення. Проте вони широко застосовуються на перших етапах функціонального проєктування та при побудові ієрархічних блоків.

Розглянемо процес створення власного компонента та всіх його частин на прикладі мультиплексора. Для початку створимо власну бібліотеку. Запустимо графічний редактор OrCAD *Capture* та виберемо *File/New/Library*. Спробуємо скопіювати в свою бібліотеку якісь компоненти з системної бібліотеки, наприклад, з ttl.olb. Командою *File/Open/Library* завантажимо в редактор OrCAD Capture обидві бібліотеки та розташуємо їх вікна поруч (рис. 2.22). ЛКМ виділимо один елемент, наприклад, 7404, потім у меню виберемо *Copy* та вставимо з буфера у вікно власної бібліотеки (рис. 2.23).

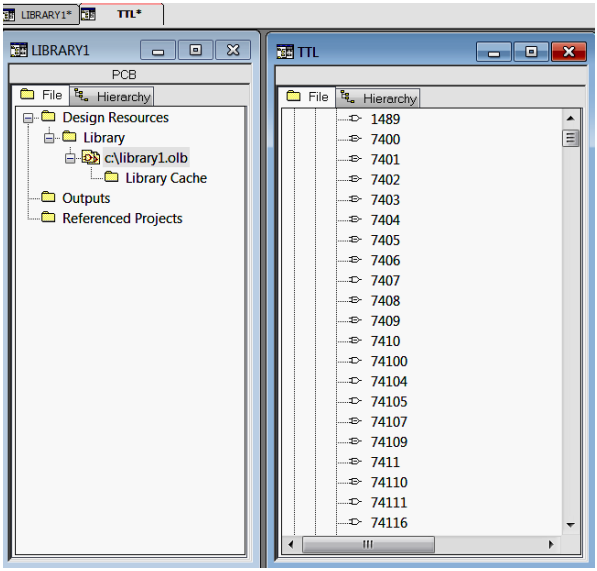


Рисунок 2.22 – Вигляд бібліотеки *ttl*

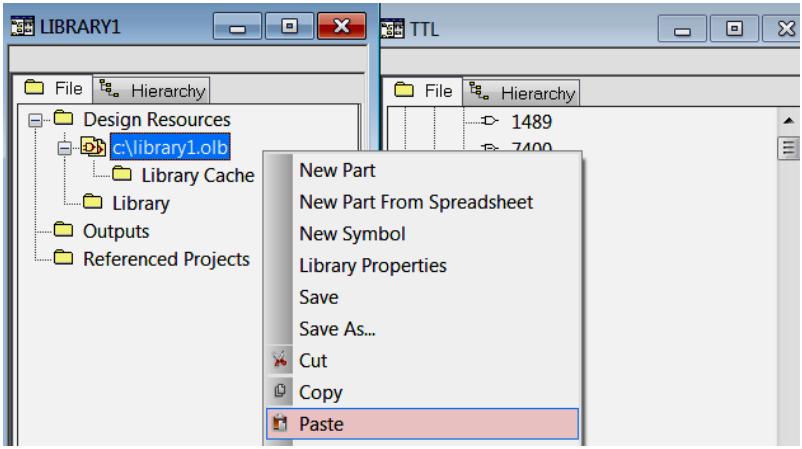


Рисунок 2.23 – Процес копіювання елементів з бібліотеки ttl

Описаним вище прийомом скопіюємо у власну бібліотеку компоненти 7408 та 7432. Тепер перейменуємо ім'я власної бібліотеки на NEW_LIBRARY.olb. З цією метою клацнемо ПКМ на рядку з назвою бібліотеки, та коли відкриється меню, виконаємо команду «*Save as*». Введемо потрібне ім'я та збережемо бібліотеку (рис. 2.24).

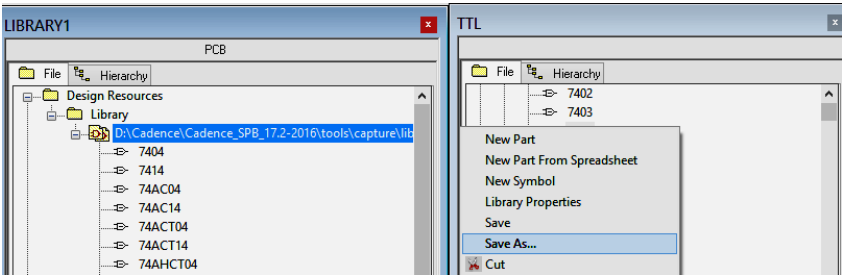


Рисунок 2.24 – Перейменування і збереження бібліотеки

Проєктування ієрархічних символів легко можна виконати з дотриманням усіх вимог стандартів, що є важливою перевагою пакета OrCAD. Як приклад, накреслимо схему заміщення ієрархічного символу, вибравши для цього мультиплексор на чотири входи. Створимо новий

проєкт *File/New/Project*. Підключимо до проєкту власну бібліотеку *new_library.olb* (рис. 2.25) та створимо в ньому схему мультиплексора (рис. 2.26).

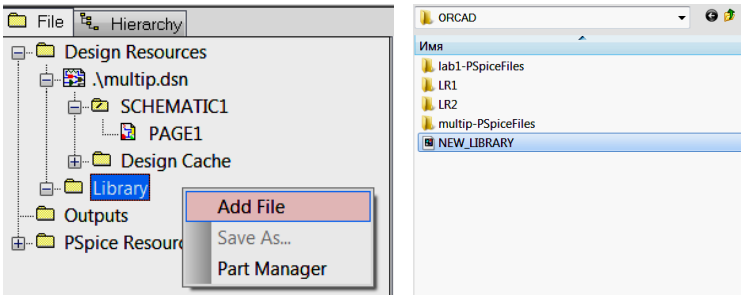


Рисунок 2.25 – Підключення власної бібліотеки

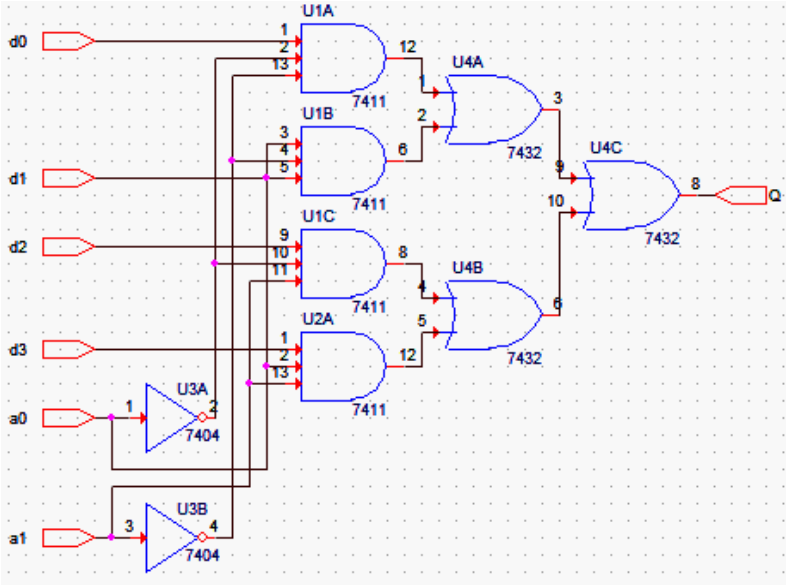


Рисунок 2.26 – Схема мультиплексора

Назвемо відкритий за замовчуванням каталог *schematic1* іншим

ім’ям, наприклад, *multip*. Збережемо проєкт та закриємо робоче вікно зі схемою заміщення. Тепер відкриваємо бібліотеку *new_library.olb* та копіюємо в неї каталог *multip* (рис. 2.27), при цьому у файловій структурі бібліотеки з’явиться потрібний каталог, а в кеші бібліотеки (*Library Cache*) – відповідні компоненти, що входять в схему заміщення (рис. 2.28).

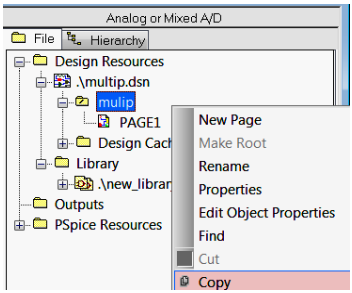


Рисунок 2.27 – Процес копіювання схеми мультиплексора до власної бібліотеки

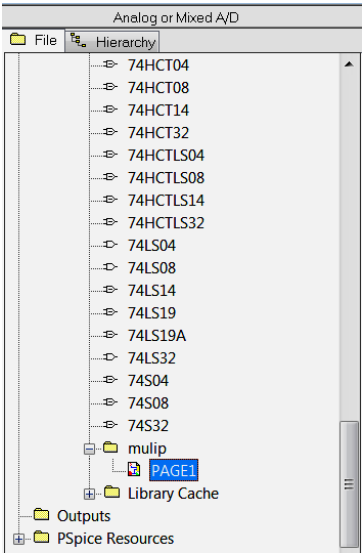


Рисунок 2.28 – Вигляд власної бібліотеки

Тепер необхідно створити зовнішній опис ієрархічного символу. Клацнемо на каталозі з назвою бібліотеки ПКМ та виконаємо команду *New Part* (рис. 2.29).

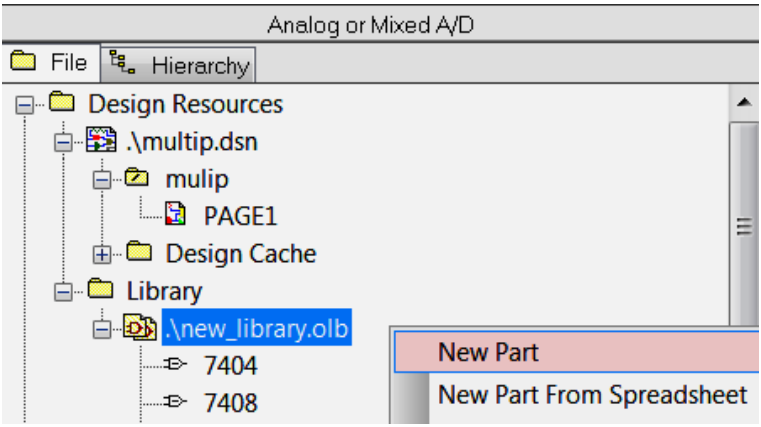


Рисунок 2.29 – Створення *New Part*

Відкриється панель *New Part Properties*. Введемо в полі *Name* ім'я символу MULT та зробимо посилання на схему заміщення ієрархічного символу (рис. 2.30).

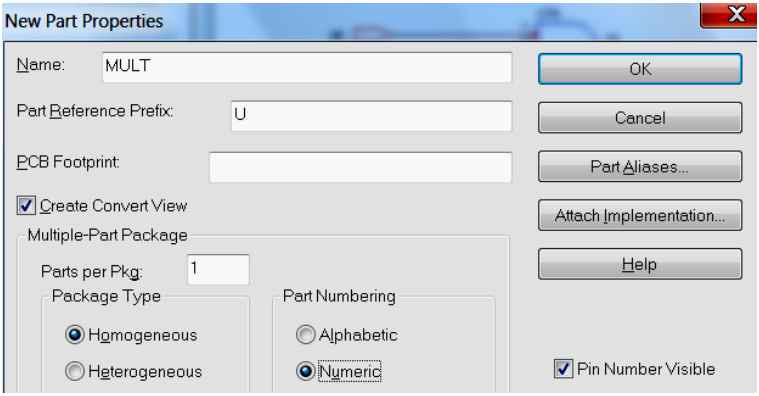


Рисунок 2.30 – Вікно властивостей *New Part*

В полі *PCB Footprint* задається тип корпусу, в якому розміщується реальний компонент. Для абстрактного символу така інформація не має сенсу, тому залишимо його порожнім. Встановимо позначку *Create Convert View*, щоб показати, що для проєктованого елемента існує логічне еквівалентне позначення. Натиснемо кнопку *Attach Implementation* (підключиться реалізація). У нашому випадку це схема заміщення, що зберігається в каталозі *multip*.

Відкриється діалогова панель з тією ж назвою. У верхнє поле *Implementation Type* введемо тип внутрішнього опису: *Schematic View* – схема заміщення, в наступне поле *Implementation* (реалізація) – ім'я каталога, де вона зберігається, нижнє поле залишимо порожнім, тому що схема заміщення знаходиться в тій же самій бібліотеці, що й символ (рис. 2.31).

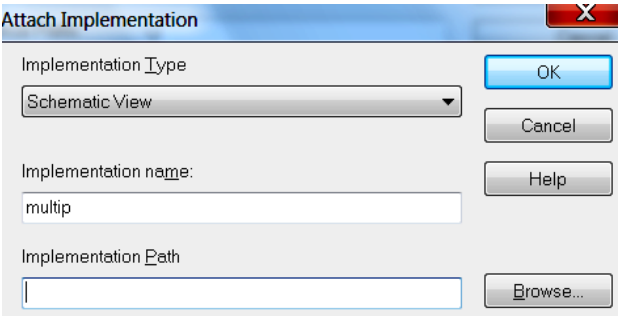


Рисунок 2.31 – Підключення реалізації

У наступному вікні *Parts per Pkg* потрібно встановити число елементів, що розміщуються в корпусі мікросхеми. Залишимо тут цифру «1», задану за замовчуванням. Зазначимо, що при створенні компонента в розділі *Package Type* можна змінити його тип та вибрати однорідні або неоднорідні символи. У першому випадку в корпусі можуть розміщуватися різні деталі, проте це зустрічається досить рідко та навіть не підтримується деякими САПР. У розділі *Part Numbering* (спосіб нумерації символів – літерний або цифровий) виберемо *Numeric*.

Натискаємо *OK*, з'являється вікно для редагування. Пунктирна лінія

означає межі позначення (не включаючи входи і виходи). Для накреслення ліній УГП нам потрібно на верхній панелі інструментів вибрати *Place/Line/*.

У даному меню також присутня можливість креслити прямі кути, еліпси, дуги, еліптичні дуги, криві лінії, квадрати і прямокутники, замкнуті криві лінії. Для прикладу зобразимо ромб. Необхідно зазначити, що при створенні будь-якої форми УГП його область виділення завжди буде чотирикутником (пунктирна лінія). Подальше додавання входів і виходів відбувається саме до цієї області, тобто їх не можна буде візуально приєднати до ромба (в нашому прикладі). Але напис на виводах можна переміщати куди завгодно (рис. 2.32).

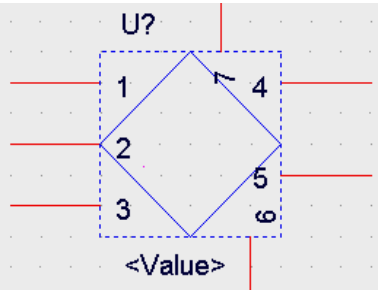


Рисунок 2.32 – Приклад приєднаних виводів

Накреслимо мультиплексор у вигляді прямокутника і за допомогою команди меню *Place/Text/* отримаємо УГП, як показано на рис. 2.33.

Тепер додаємо контакти (входи/виходи). На відміну від ліній та тексту, які є тільки графічним відображенням на схемі, контакти служать для з'єднання компонентів та відображення УГП. Для цього виберемо інструмент *Place Pin* (рис. 2.34). У ньому задається ім'я контакту, номер контакту в корпусі (актуально тільки для реальних компонентів), форма контакту та його тип. За допомогою кнопки *User Properties* задаються і редагуються параметри (нам досить задати ім'я та тип).

Після завершення розміщення всіх контактів отримаємо УГП, як показано на рис. 2.35.

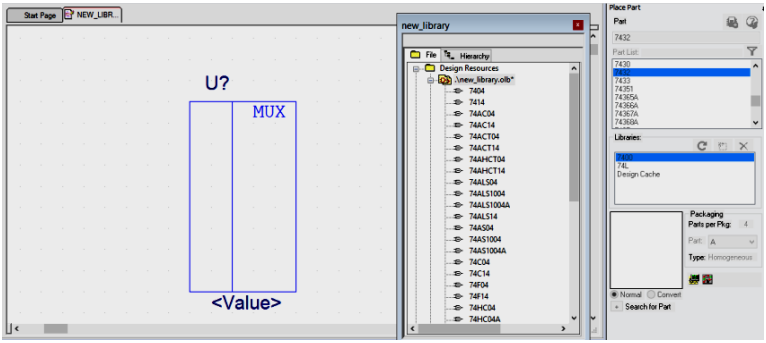


Рисунок 2.33 – УГП мультиплексора

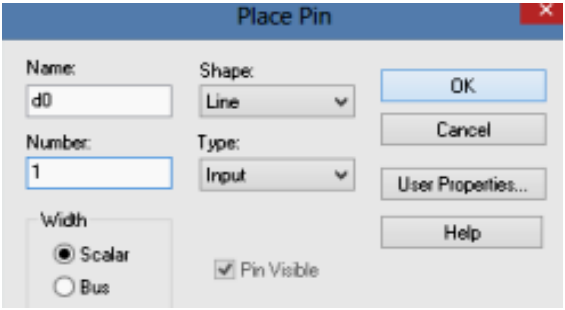


Рисунок 2.34 – Вікно *Place Pin*

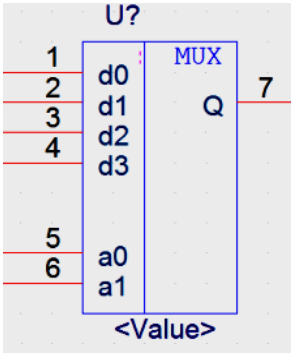


Рисунок 2.35 – УГП мультиплексора

Збережемо бібліотеку та закриємо її. Тепер створимо новий проєкт *mult.opj*, підключимо до нього власну бібліотеку *new_library.olb*, як показано на рис. 2.36.

Розмістимо у вікні графічного редактора одну копію ієрархічного символу *MULT* (рис. 2.37) та повідомимо редактору OrCAD *Capture*, що символ *MULT* не є примітивом, а є ієрархічним символом. Для цього двічі клацнемо на ньому ЛКМ та, коли відкриється вікно *Property Editor*, змінимо значення властивості *Primitive* з *default* (за замовчуванням символ вважається примітивом) на *NO* (рис. 2.38).

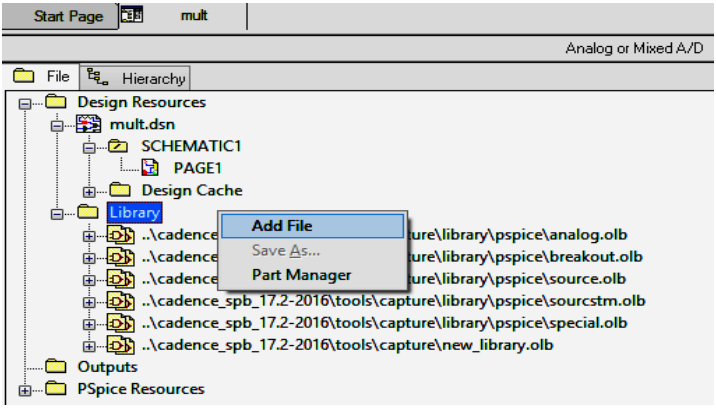


Рисунок 2.36 – Підключення власної бібліотеки

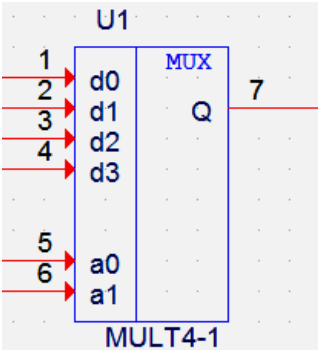


Рисунок 2.37 – Розміщення мультиплексора на робочій сторінці

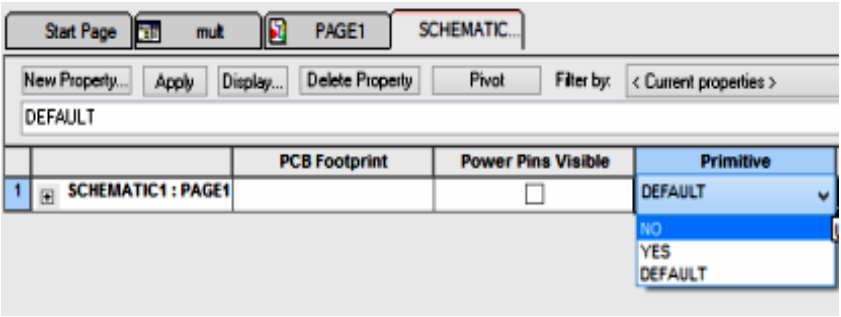


Рисунок 2.38 – Задання властивостей мультиплексора

Переконаємося, що символ тепер має підпорядковану схему. Для цього досить виконати команду *View/Descend Hierarchy*, при цьому редактор повинен знизити ієрархію опису та показати схему заміщення символу *MULT* (рис. 2.39).

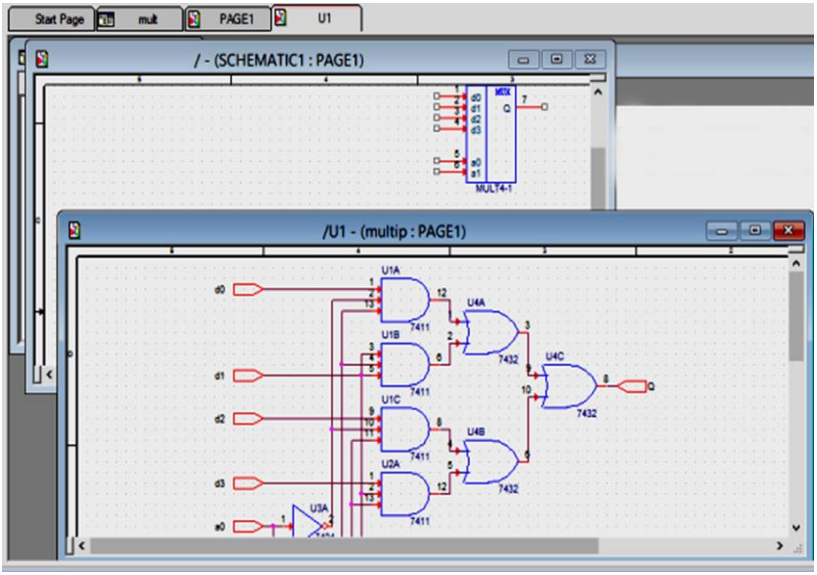


Рисунок 2.39 – УГП мультиплексора та його підпорядкована схема

Залишається промоделювати схему, яка містить усього один ієрархічний символ. Для цього підключимо до входу мультиплексора генератори сигналів (рис. 2.40).

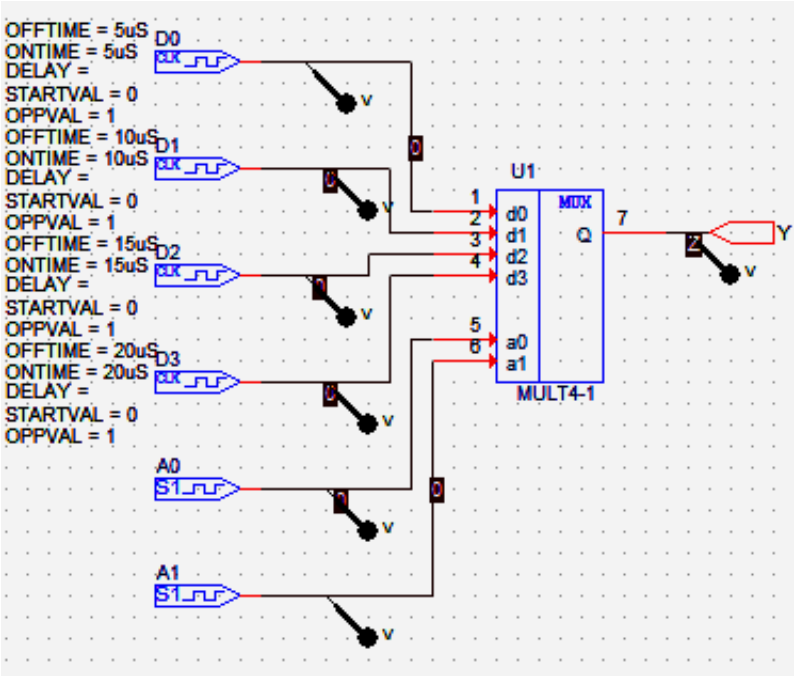


Рисунок 2.40 – Схема мультиплексора з підключеними генераторами вхідних сигналів

Результати моделювання схеми свідчать про правильне створення компонента та коректність його роботи.

На даному етапі отримано так званий абстрактний компонент, який можна використовувати при розробці та моделюванні схем.

Завдання для виконання практичної роботи

Створити власний елемент в САПР OrCAD (табл. 2.2).

Таблиця 2.2

№ варіанту	Форма логічного елемента	Функція, яка виконується елементом
1	прямокутник	$F = A + \overline{B} + C + D$
2	квадрат	$F = A + B + A \cdot B \cdot C \cdot D$
3	паралелограм	$F = A + B + A \cdot \overline{C} \cdot D$
4	трикутник	$F = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D$
5	коло	$F = (A + B) \cdot \overline{C} \cdot \overline{D}$
6	зірка	$F = A \cdot B + \overline{C}$
7	ромб	$F = (A + B) \cdot \overline{C}$
8	прямокутник	$F = \overline{(A \cdot B \cdot C) \cdot A}$
9	квадрат	$F = A \cdot \overline{B} \cdot \overline{C}$
10	паралелограм	$F = A + B + \overline{C} \cdot D$
11	трикутник	$F = A \cdot B \cdot \overline{C} \cdot D$
12	квадрат	$F = A \cdot \overline{B} \cdot \overline{C} + A$
13	зірка	$F = A \cdot B + C + D$
14	ромб	$F = A + B \cdot C + A \cdot B \cdot C$
15	паралелограм	$F = A + B \cdot C \cdot D + A \cdot \overline{C}$

Практична робота 3

Програмування макросу власного елемента

Мета роботи: виконати програмування макросу елемента в середовищі OrCAD.

Для програми *PSpice* завдання на моделювання заноситься в текстові файли. Знання їх форматів при графічному введенні схеми бажано, тому що дозволяє створювати шаблони нових символів компонентів, а також дозволяє складати текстові описи макромоделей та полегшує пошук помилок при налагодженні схеми. При графічному введенні схем створюються три файли завдання з одним і тим же ім'ям: *.NET (таблиця з'єднань), *.ALS (список підключення кіл до виводів компонентів) та *.CIR (список директив моделювання). При моделюванні в *PSpice* безпосередньо завантажується файл *.CIR, в якому містяться посилання на інші файли. Для його складання вручну спочатку рисується принципова схема пристрою, що моделюється, та присвоюються імена всім її вузлам. Стандартне позначення вузлів наведено в табл. 2.3.

Таблиця 2.3 – Позначення вузлів

Ім'я вузла	Напруга/ рівень	Опис
0	0 В	Аналогова «земля»
$ G_CD4000_VDD	5 В	Джерело живлення КМОН ІС
$ G_CD4000_VSS	0 В	Джерело живлення КМОН ІС
$ G_DPWR	5 В	Джерело живлення ТТЛ ІС
$ G_DGND	0 В	Джерело живлення ТТЛ ІС
$ D_HI	«1»	Логічна «1»
$ D_LO	«0»	Логічний «0»
$ D_X	«X»	Невизначений логічний стан
$ D_NC	–	Не підключений до схеми вивід цифрового компонента

Вузол $ D_NC застосовується для позначення невикористовуваних виводів (NC означає Not Connected – немає підключення). При посиланні

на цифрові імена вузлів використовуються круглі дужки, наприклад *V* (6) – це потенціал вузла 6. Імена вузлів у вигляді алфавітно-цифрових символів з посиланнями на них беруться у квадратні дужки [ ], щоб відрізнити їх від імен компонентів. Наприклад, потенціал вузла IN позначається як *V* ([IN]), а *V* (I2) – напруга на джерелі струму I2. В програмі існує угода, що всі вузли, імена яких починаються з символів $ G_, є *глобальними,* наприклад вузол $ G_POS. Глобальні вузли використовуються в схемах, що мають макромоделі. Після іменування вузлів складають завдання на моделювання, яке заноситься у файл з розширенням *.CIR, в якому *перший рядок файлу* – це рядок заголовка (виводиться у вигляді заголовка у вихідному файлі), *рядки коментарів* містять символ «*» в першій позиції (кінець рядка після знака «;» сприймається як коментар), *останній рядок файлу* має команду .END, а *рядок продовження* починається з символу «+» в першій позиції (максимальна довжина рядка 132 символи). Пробіли, коми або знаки рівності еквівалентні (програма *PSpice* не розрізняє великі та малі літери).

Окремі фрагменти схеми заміщення компонентів має сенс оформляти у вигляді макромоделі, опис якої починається директивою .SUBCKT та закінчується директивою .ENDS. Між ними поміщаються описи компонентів:

.SUBCKT <*ім'я макромоделі*> <*список вузлів*>
+ [OPTIONAL: << *вузол інтерфейсу* > = <*значення за замовчуванням*>>*}
+ [PARAMS: <ім'я *параметра*> = <*значення*> *}
+ [TEXT: <<*ім'я текстової змінної*> – <*текст* >> *]
{Опис компонентів}
.ENDS [*ім'я макромоделі*]

Ключове слово OPTIONAL використовується для специфікації одного або більше необов'язкових вузлів макромоделі – вказуються ім'я вузла та його значення за замовчуванням. Якщо при виклику макромоделі ці вузли не вказуються, використовуються їх значення за замовчуванням. Після ключового слова PARAMS наводиться список параметрів, значення яких передаються з основного кола в макромодель.

Після ключового слова TEXT – текстова змінна, яка передається з опису на основному колі в опис макромоделі, яка використовується тільки при моделюванні цифрових пристроїв. Між директивами .SUBCKT та .ENDS можна розташовувати опис інших макромоделей та інші директиви.

Високорівневе моделювання суматора

Задача високорівневого моделювання – не розкривати структуру об'єкта та змоделювати його роботу, тобто такий інструмент повинен імітувати не тільки функцію, а й часові співвідношення в схемі. Основна ідея цього методу така – зобразити модель цифрового пристрою у вигляді двох частин, перша з яких (логічний блок) повинна моделювати логіку функціонування з нульовими затримками, а друга (блок затримки) повинна імітувати реальні затримки. Звичайна макромодель являє собою список з'єднань, тобто опис компонента на більш низькому рівні, де він поданий у вигляді структури (функціональної або принципової схеми). Логіко-часова модель не має списку з'єднань, а містить два або три (LOGICEXP, PINDLY та CONSTRAINT) примітиви.

Примітив LOGICEXP (*Logic Expression*) описує логіку роботи компонента та має такий вигляд:

Uxxx LOGICEXP (<число входів>, <число виходів>)
+ <живлення> <земля>
+ <вхідний_вузол_1>...<вхідний_вузол_m>
+ <вихідний_вузол_1>...<вихідний_вузол_n>
+ <ім'я_моделі> < ім.'я_моделі_вхід/вихід>
+ [IO_LEVEL=<рівень_моделі_інтерфейса>]
+ [MNTYMXDLY=<вибір_значення_затримки>]
+ LOGIC:
+ <логічне_значення>

Логічне призначення може включати логічні вирази для проміжних змінних та вихідних вузлів у такій формі:

<ім'я_вихідного_вузла> = {<логічний вираз>}
<проміжна_змінна> = {<логічний вираз>}

Кожен згаданий вихідний вузол повинен мати тільки один логічний

вираз, а проміжні змінні, які призначені один раз, можуть використовуватися в наступних виразах багаторазово. Розглянемо, як виглядає цей примітив для суматора ADDER.

Функціональна модель суматора ADDER з нульовими затримками:

```
.SUBCKT Adder_log C0 A1 B1 A2 B2 SUM1 SUM2 C2
; директива початку макромоделі
+ OPTIONAL: DPWR = $G_DPWR DGND = $G_DGND
; контакти живлення та землі
+ PARAMS:   MNTYMXDLY=0  IO_LEVEL=0
; ініціалізація параметрів
U82LOG LOGICEXP(5,8) DPWR DGND
; ім’я примітиву має починатися з літери «U»
+ C0   A1   B1   A2   B2   SUM1 SUM2 C2
; входи та виходи логічного блоку
+ D0_GATE IO_STD IO_LEVEL={IO_LEVEL}
; посилання на стандартну модель вхід/вихід
+ LOGIC:
; початок логічної секції
+ C0A1B1 = { ~( (C0 & A1) | (C0 & B1) | (A1 & B1) ) }
; логічний вираз для вихідного вузла макромоделі
.ENDS Adder_log
; директива завершення макромоделі
```

Додаємо до цієї макромоделі примітив PINDLY, який відтворює затримки. З його додаванням необхідно внести зміни в логічний блок LOGICEXP. У даному випадку на виходи C0_I A1_I B1_I A2_I B2_I (Internal – внутрішній) здійснюється пряма трансляція вхідних сигналів C0 A1 B1 A2 B2. Вони не зазнають жодних логічних перетворень (Buffering) та використовуються в блоці PINDLY як внутрішні (додаткові) вузли. Сигнали SUM1_O SUM2_O C2_O є виходами з LOGICEXP та входами для PINDLY. Для цих сигналів у блоці затримки створюються шляхи, проходячи через які сигнал отримує необхідне запізнювання. В блок затримки не можна заводити сигнали безпосередньо з зовнішніх входів, тому їх транслюють через логічний блок навіть у тому випадку, якщо вони не зазнають логічних

перетворень. Від того, з якого входу надходить конкретна подія (перемикання), залежить запізнювання вихідного сигналу. Зміни станів на входах контролюються функціями:

CHANGED (<вузол>, <інтервал_часу>)
CHANGED_LH (<вузол>, <інтервал_часу>)
CHANGED_HL (<вузол>, <інтервал_часу>).

Функція CHANGED повертає значення TRUE, якщо вузол змінював свій стан на інтервалі часу. Функція CHANGED_LH контролює фронт, а функція CHANGED_HL – спад сигнала у вузлі. Функції TRN_LH та TRN_HL стежать за зміною вихідних вузлів, для яких обчислюються затримки. Замість рівнів L (низький рівень) та H (високий рівень) можна використовувати також Z (високоомний вихід) та $ (будь-який рівень). Ці функції не мають аргументів та просто фіксують зміни станів вихідних вузлів.

Функціональна макромодель суматора (ADDER) з урахуванням затримок має вигляд:

```
.SUBCKT Adder_log_dly C0 A1 B1 A2 B2 SUM1 SUM2 C2
; нове ім’я макромоделі
U82DLY PINDLY(3,0,5) DPWR DGND
; початок примітиву «PINDLY»
+ SUM1 SUM2 C2
; виходи логічного блоку
+ C0 A1 B1 A2 B2
; входи, від яких залежить затримка
+ SUM1_O SUM2_O C2_O
; вузли, для яких обчислюється затримка
+ IO_STD MNTYMXDLY={MNTYMXDLY} IO_LEVEL={IO_LEVEL}
; стандартна модель вхід/вихід
BOOLEAN:
+ ANY_CH_AB = { CHANGED(A2,0) | CHANGED(B2,0) |
CHANGED(A1,0) | CHANGED(B1,0) }
; секція визначає проміжні змінні, які будуть
; використані для обчислення затримки
+ PINDLY:
```

```
+ SUM1_O = {
; оператор обчислює конкретну затримку
+    CASE(
+      CHANGED(C0,0) & TRN_HL, DELAY(-1,-1,40NS),
+      ANY_CH_AB & TRN_LH, DELAY(-1,-1,40NS),
+      ANY_CH_AB & TRN_HL, DELAY(-1,-1,35NS),
+      CHANGED(C0,0) & TRN_LH, DELAY(-1,-1,34NS),
+      DELAY(-1,-1,41NS); DEFAULT
+      )
+    }
+   SUM2_O = {
+    CASE(
+      ANY_CH_AB & TRN_LH, DELAY(-1,-1,40NS),
+      CHANGED(C0,0) & TRN_HL, DELAY(-1,-1,42NS),
+      CHANGED(C0,0) & TRN_LH, DELAY(-1,-1,38NS),
+      ANY_CH_AB & TRN_HL, DELAY(-1,-1,35NS),
+      DELAY(-1,-1,43NS); DEFAULT
+      )
+    }
+   C2_O = {
+    CASE(
+      ANY_CH_AB & TRN_LH, DELAY(-1,-1,40NS),
+      ANY_CH_AB & TRN_HL, DELAY(-1,-1,35NS),
+      CHANGED(C0,0) & TRN_HL, DELAY(-1,17NS,27NS),
+      CHANGED(C0,0) & TRN_LH, DELAY(-1,12NS,19NS),
+      DELAY(-1,18NS,41NS) ; DEFAULT
+      )
+    }
.ENDS Adder_log_dly
```

У PINDLY можуть бути використані вирази для затримок, які включають оператор вибору CASE. Він перевіряє умовні вирази в порядку їх проходження, поки не буде виявлено перший справжній вираз, після чого затримці присвоюється значення з рядка, який відповідає виконаній умові.

Оператор DELAY (<min>, <typ>, <max>) визначає відповідно мінімальну, типову і максимальну затримки.

Приклад створення інвертора

Ознайомившись зі структурою та описом макромоделі та декількома примітивами (LOGICEXP та PINDLY), створимо власний інвертор. Логіка роботи даного елемента проста: отримавши на вході сигнал – на виході отримуємо сигнал, протилежний за значенням.

Функціональна макромодель інвертора з урахуванням затримок:

```
.SUBCKT Example X1_I Y_O
+ OPTIONAL: DPWR = $G_DPWR DGND = $G_DGND
+ PARAMS: MNTYMXDLY=0 IO_LEVEL=0
U82LOG LOGICEXP(1,2) DPWR DGND
+ X1_I
+ X1  Y
+ D0_GATE IO_STD IO_LEVEL={IO_LEVEL}
+
+ LOGIC:
+ X1 = {X1_I}
+ Y = {(~X1)}
U82DLY PINDLY(1,0,1) DPWR DGND
+ Y
+ X1
+ Y_O
+ IO_STD MNTYMXDLY={MNTYMXDLY} IO_LEVEL={IO_LEVEL}
+
+ BOOLEAN:
+ ANY_CH_AB = { CHANGED(X1,0) }
+
+ PINDLY:
+ Y_O = {
+ CASE(
+ ANY_CH_AB & TRN_LH, DELAY(-1,-1,40NS),
+ ANY_CH_AB & TRN_HL, DELAY(-1,-1,40NS),
```

```
+ CHANGED(X1,0) & TRN_HL,DELAY(-1,17NS,27NS),
+ CHANGED(X1,0) & TRN_LH,DELAY(-1,12NS,19NS),
+ DELAY(-1,18NS,41NS);DEFAULT
+ )
+ }
.ENDS
```

При додаванні створеного елемента в проєкт необхідно в *Simulation Profile – Configuration Files – Library* вказати шлях до файлу з описом (example.lib). В іншому випадку буде виникати помилка, яка вказує, що «підсхема» елемента не визначена.

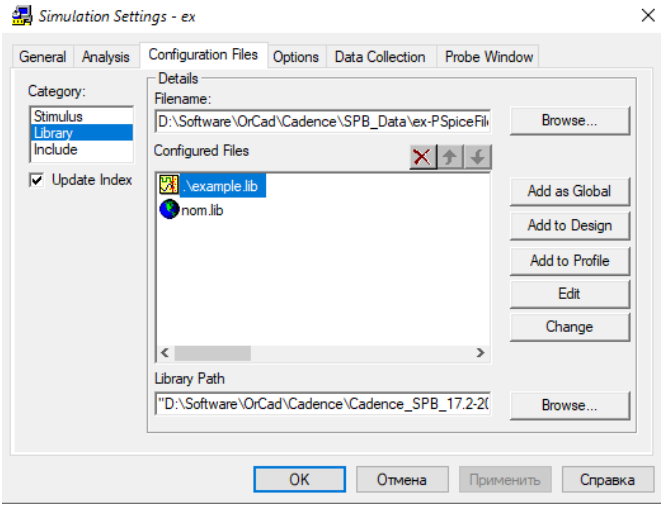


Рисунок 2.41 – Вікно налаштувань профілю симуляції

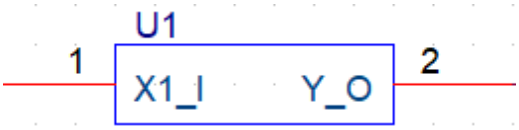


Рисунок 2.42 – Створений елемент

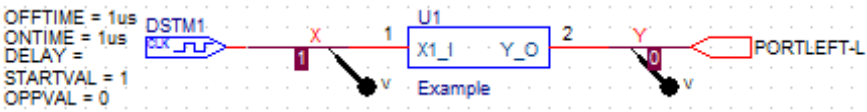


Рисунок 2.43 – Зібрана схема

Результати моделювання з урахуванням затримок наведені на рис. 2.44.

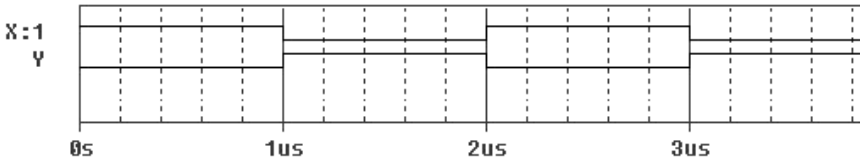


Рисунок 2.44 – Результати моделювання

Створення суматора

Використовуючи інформацію, наведену вище, створюємо файл з описом (adder.lib) елемента в SPICE-форматі з таким змістом:

```
* 2-BIT BINARY FULL ADDER
.SUBCKT Adder C0_I A1_I B1_I A2_I B2_I SUM1_O SUM2_O C2_O
+ OPTIONAL: DPWR = $G_DPWR DGND = $G_DGND
+ PARAMS:   MNTYMXDLY=0  IO_LEVEL=0
U82LOG LOGICEXP(5,8) DPWR DGND
+ C0_I A1_I B1_I A2_I B2_I
+ C0   A1   B1   A2   B2   SUM1 SUM2 C2
+ D0_GATE IO_STD IO_LEVEL={IO_LEVEL}
+
+ LOGIC:
+ C0 = { C0_I }
+ A1 = { A1_I }
+ B1 = { B1_I }
+ A2 = { A2_I }
+ B2 = { B2_I }
```

```
+ A2BAR = { ~A2 }
+ B2BAR = { ~B2 }
+
+ C0A1B1 = { ~( (C0 & A1) | (C0 & B1) | (A1 & B1) ) }
+ SUM1 = { (C0 & C0A1B1) | (A1 & C0A1B1) | (B1 & C0A1B1) | (C0 & A1
& B1) }
+ C2 = { ~( ( C0A1B1 & A2BAR) | (C0A1B1 & B2BAR) | (A2BAR &
B2BAR) ) }
+ SUM2 = { ~( (C0A1B1 & C2) | (C2 & A2BAR) | (C2 & B2BAR) |
+ (A2BAR & B2BAR & C0A1B1) ) }
U82DLY PINDLY(3,0,5) DPWR DGND
+ SUM1 SUM2 C2
+ C0 A1 B1 A2 B2
+ SUM1_O SUM2_O C2_O
+ IO_STD MNTYMXDLY={MNTYMXDLY} IO_LEVEL={IO_LEVEL}
+
+ BOOLEAN:
+ ANY_CH_AB = { CHANGED(A2,0) | CHANGED(B2,0) |
CHANGED(A1,0) | CHANGED(B1,0) }
+
+ PINDLY:
+ SUM1_O = {
+ CASE(
+ CHANGED(C0,0) & TRN_HL, DELAY(-1,-1,40NS),
+ ANY_CH_AB & TRN_LH, DELAY(-1,-1,40NS),
+ ANY_CH_AB & TRN_HL, DELAY(-1,-1,35NS),
+ CHANGED(C0,0) & TRN_LH, DELAY(-1,-1,34NS),
+ DELAY(-1,-1,41NS) ; DEFAULT
+ )
+ }
+ SUM2_O = {
+ CASE(
+ ANY_CH_AB & TRN_LH, DELAY(-1,-1,40NS),
+ CHANGED(C0,0) & TRN_HL, DELAY(-1,-1,42NS),
+ CHANGED(C0,0) & TRN_LH, DELAY(-1,-1,38NS),
```

```
+ ANY_CH_AB & TRN_HL, DELAY(-1,-1,35NS),
+ DELAY(-1,-1,43NS) ; DEFAULT
+ )
+ }
+ C2_O = {
+ CASE(
+ ANY_CH_AB & TRN_LH, DELAY(-1,-1,40NS),
+ ANY_CH_AB & TRN_HL, DELAY(-1,-1,35NS),
+ CHANGED(C0,0) & TRN_HL, DELAY(-1,17NS,27NS),
+ CHANGED(C0,0) & TRN_LH, DELAY(-1,12NS,19NS),
+ DELAY(-1,18NS,41NS) ; DEFAULT
+ )
+ }
.ENDS
```

В пакеті OrCAD в пункті меню *File* виконуємо *New – Library* та вводимо назву. Потім при активному менеджері проєкту виконуємо *Tools – Generate Part*, вказуємо створений нами файл adder.lib та тип (*PSpice model library*), назву бібліотеки, в яку він буде поміщений. Буде згенеровано символьний опис елемента (рис. 2.45). Редагуємо його (змінюємо розміри корпусу, параметри контактів) та створюємо проєкт для моделювання (рис. 2.46). Результати моделювання з урахуванням затримок наведені на рис. 2.47.

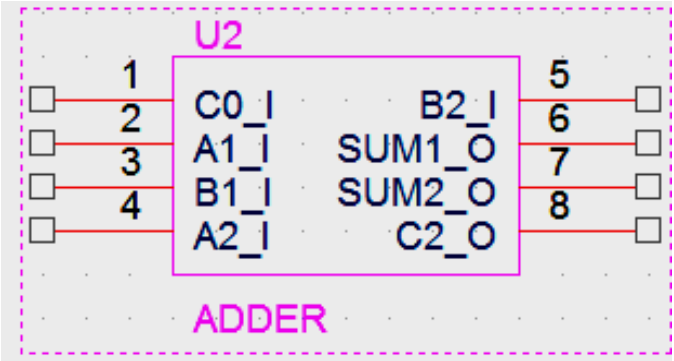


Рисунок 2.45 – Створений елемент

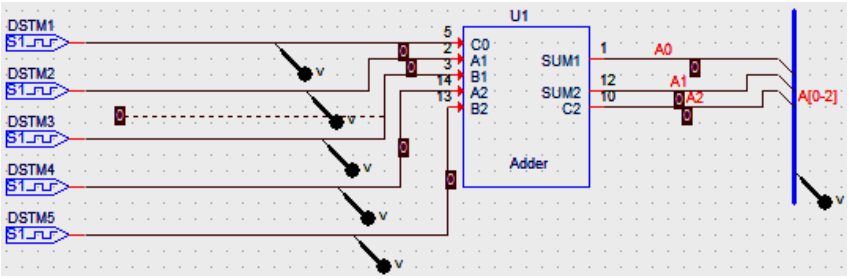


Рисунок 2.46 – Зібрана схема

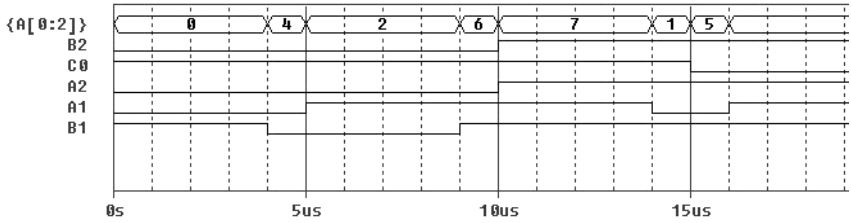


Рисунок 2.47 – Результати моделювання

Створення мультиплексора

Використовуючи інформацію, наведену вище, створюємо файл з описом (mux_16_1.lib) елемента в SPICE-форматі з таким змістом:

.SUBCKT 74150 Enable A B C D E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 E10 E11 E12 E13 E14 E15 W ; позначення назв входів та виходів елемента
+ OPTIONAL: DPWR=$G_DPWR DGND=$G_DGND
+ PARAMS: MNTYMXDLY=0 IO_LEVEL=0
*
UMUXLOG LOGICEXP(21,22) DPWR DGND
; вказівка кількості контактів елемента
+ Enable A B C D E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 E10 E11 E12 E13 E14 E15
*
+ GBAR_I A_I B_I C_I D_I E0_I E1_I E2_I E3_I E4_I E5_I E6_I E7_I E8_I E9_I

```
+ E10_I E11_I E12_I E13_I E14_I E15_I W_O O
; _I – позначення входу, _O – позначення виходу
*
+ D0_GATE IO_STD IO_LEVEL={IO_LEVEL}
+ LOGIC:
+ GBAR_I = { Enable }
+ A_I = { A } ; присвоєння входу A імені A_I
+ B_I = { B }
+ C_I = { C }
+ D_I = { D }
+ E0_I = { E0 }
+ E1_I = { E1 }
+ E2_I = { E2 }
+ E3_I = { E3 }
+ E4_I = { E4 }
+ E5_I = { E5 }
+ E6_I = { E6 }
+ E7_I = { E7 }
+ E8_I = { E8 }
+ E9_I = { E9 }
+ E10_I = { E10 }
+ E11_I = { E11 }
+ E12_I = { E12 }
+ E13_I = { E13 }
+ E14_I = { E14 }
+ E15_I = { E15 }
+ IA = { ~A_I } ; IA має значення, яке інвертоване від A_I
+ IB = { ~B_I }
+ IC = { ~C_I }
+ ID = { ~D_I }
+ IG = { GBAR_I }
+ IE0 = { E0 & IA & IB & IC & ID & IG }; логічне «І»
+ IE1 = { E1 & A_I & IB & IC & ID & IG }
+ IE2 = { E2 & IA & B_I & IC & ID & IG }
+ IE3 = { E3 & A_I & B_I & IC & ID & IG }
```

```
+ IE4 = { E4 & IA & IB & C_I & ID & IG }
+ IE5 = { E5 & A_I & IB & C_I & ID & IG }
+ IE6 = { E6 & IA & B_I & C_I & ID & IG } ; логічне «АБО»
+ IE7 = { E7 & A_I & B_I & C_I & ID & IG }
+ IE8 = { E8 & IA & IB & IC & D_I & IG }
+ IE9 = { E9 & A_I & IB & IC & D_I & IG }
+ IE10 = { E10 & IA & B_I & IC & D_I & IG }
+ IE11 = { E11 & A_I & B_I & IC & D_I & IG }
+ IE12 = { E12 & IA & IB & C_I & D_I & IG }
+ IE13 = { E13 & A_I & IB & C_I & D_I & IG }
+ IE14 = { E14 & IA & B_I & C_I & D_I & IG }
+ IE15 = { E15 & A_I & B_I & C_I & D_I & IG }
+ W_O = { (IE0 | IE1 | IE2 | IE3 | IE4 | IE5 | IE6 | IE7 | IE8 |
+ IE9 | IE10 | IE11 | IE12 | IE13 | IE14 | IE15) }
* nW_O = { ~W }
*
UMUXDLY PINDLY (1,0,21) DPWR DGND
+ W_O
*
+ GBAR_I A_I B_I C_I D_I E0_I E1_I E2_I E3_I E4_I E5_I E6_I E7_I E8_I
E9_I
+ E10_I E11_I E12_I E13_I E14_I E15_I
*
+ W
+ IO_STD MNTYMXDLY={MNTYMXDLY} IO_LEVEL={IO_LEVEL}
+ BOOLEAN:
+ DATA = { CHANGED(E0_I,0) | CHANGED(E1_I,0) | CHANGED(E2_I,0)
| CHANGED(E3_I,0) |
+ CHANGED(E4_I,0) | CHANGED(E5_I,0) | CHANGED(E6_I,0) |
CHANGED(E7_I,0) |
+ CHANGED(E8_I,0) | CHANGED(E9_I,0) | CHANGED(E10_I,0) |
CHANGED(E11_I,0) |
+ CHANGED(E12_I,0) | CHANGED(E13_I,0) | CHANGED(E14_I,0) |
CHANGED(E15_I,0) }
+ SELECT = { CHANGED(A_I,0) | CHANGED(B_I,0) | CHANGED(C_I,0) |
```

```
CHANGED(D_I,0) }
+ PINDLY:
+ W = {
+ CASE(
+ SELECT & TRN_LH, DELAY(-1,23NS,35NS),
+ SELECT & TRN_HL, DELAY(-1,22NS,33NS),
+ CHANGED(GBAR_I,0) & TRN_HL, DELAY(-1,21NS,30NS),
+ CHANGED(GBAR_I,0) & TRN_LH, DELAY(-1,15.5NS,24NS),
+ DATA & TRN_HL, DELAY(-1,13NS,20NS),
+ DATA & TRN_LH, DELAY(-1,8.5NS,14NS),
+ DELAY(-1,24NS,36NS)
+ )
+ }
* nW = {
* CASE(
* SELECT, DELAY(-1,26NS,38NS),
* CHANGED(GBAR_I,0) & TRN_HL, DELAY(-1,27NS,36NS),
* CHANGED(GBAR_I,0) & TRN_LH, DELAY(-1,21.5NS,30NS),
* DATA, DELAY(-1,18NS,25NS),
* DELAY(-1,29NS,41NS)
* )
* }
; встановлення затримки для елемента
*
.ENDS
*$
```

Запускаємо OrCAD, в пункті меню *File* виконуємо *New – Library*, вводимо назву, потім виконуємо *Tools – Generate Part*, вказуємо створений нами модельний файл mux_16_1.lib, тип файлу (*PSpice model library*), назву бібліотеки, в яку він буде поміщений.

Згенерується символьний опис елемента (рис. 2.48). Редагуємо його (змінюємо розміри корпусу, параметри контактів) та створюємо проєкт для моделювання (рис. 2.49).

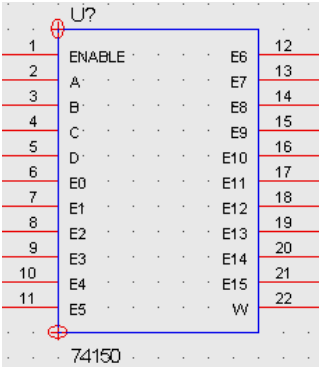


Рисунок 2.48 – УГП елемента

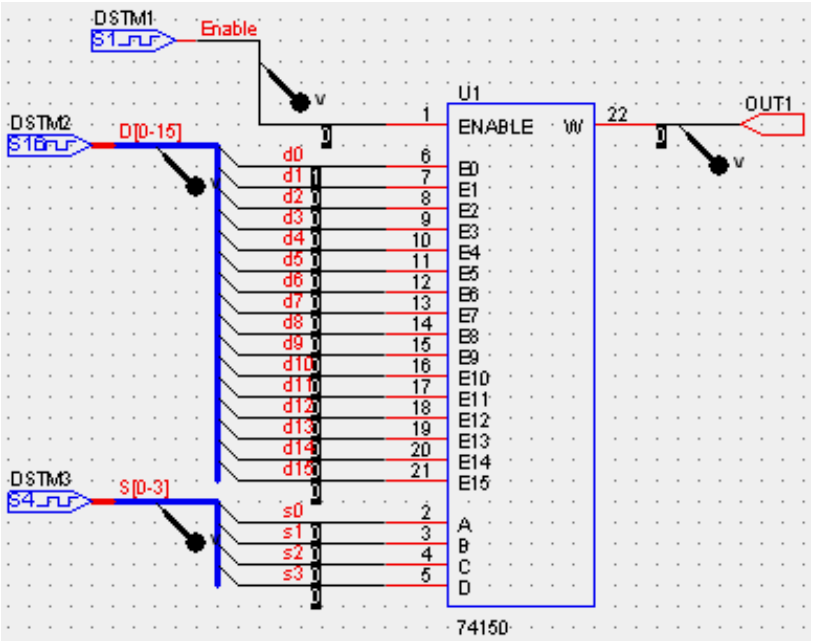


Рисунок 2.49 – Зібрана схема

Результати моделювання з урахуванням затримок наведені на рис. 2.50.

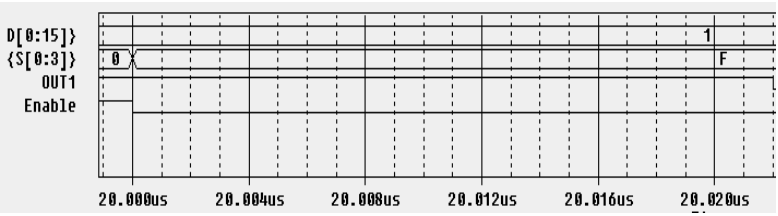


Рисунок 2.50 – Результати моделювання

Завдання для виконання практичної роботи

Виконати створення власного елемента (згідно з табл. 2.4) в системі автоматизованого проєктування OrCAD.

Таблиця 2.4

№ вар.	Логічний вираз
1	$F = \overline{X1} \cdot \overline{X2} \cdot X3 + \overline{X1} \cdot X2 \cdot X3 + X1 \cdot X2 \cdot \overline{X3}$
2	$F = X1 \cdot \overline{X2} + \overline{X1} \cdot X2$
3	$F = \overline{X1} \cdot \overline{X2} + \overline{X1} \cdot X2 + \overline{X3}$
4	$F = X1 \cdot X2 + X1 \cdot X3 + X2 \cdot X3$
5	$F = \overline{X1} \cdot X3 + X1 \cdot X2 \cdot \overline{X3}$
6	$F = X1 \cdot \overline{X2} \cdot X3 + X1 \cdot X2 \cdot \overline{X3} + X1 \cdot X2 \cdot X3$
7	$F = \overline{X1} \cdot \overline{X2} \cdot X3 + \overline{X1} \cdot X2 \cdot \overline{X3} + X1 \cdot \overline{X2} \cdot \overline{X3} + X1$
8	$F = X1 \cdot \overline{X2} \cdot X3 + \overline{X1} \cdot X2 \cdot X4$
9	$F = \overline{X1} \cdot \overline{X2} \cdot X3 + \overline{X1} \cdot X2$
10	$F = \overline{X1} \cdot \overline{X2} \cdot X3 + \overline{X2} + X1 \cdot X2$
11	$F = \overline{(X1 \cdot X2 \cdot X3)} + \overline{(X1 \cdot X2)}$
12	$F = \overline{(X1 \cdot X2 \cdot X3)} + \overline{(X1 \cdot X2)} + \overline{X1} \cdot X2$
13	$F = \overline{(X1 \cdot X2 \cdot X3)} + \overline{X1} + X2$
14	$F = \overline{(X1 \cdot X2)} + \overline{X2} + X2 \cdot X3 \cdot X4$
15	$F = \overline{(X1 + X2)} + \overline{(X1 + X3)}$

Практична робота 4

Використання програми PSpice системи OrCAD для аналогового моделювання цифрових блоків

Мета роботи: виконати аналогове моделювання пристрою з використанням програми PSpice.

У даній роботі для моделювання схем буде використано програму *PSpice (Personal Simulation Program with Integrated Circuit Emphasis)* – програма симуляції аналогової і цифрової логіки, описаної мовою SPICE, яка призначена для персональних комп'ютерів. *PSpice* був першою версією програми SPICE, пізніше до неї додали програму для перегляду та аналізу осцилограм (*Probe*). В наступних версіях поліпшувалася продуктивність та розширювався перелік платформ, які підтримуються.

Програма *Pspice* розраховує такі характеристики електронних кіл:

– режим кола по постійному струму в «робочій точці» (*Bias Point*);

– малосигнальні передавальні функції в режимі по постійному струму (*Transfer Function*);

– характеристики лінеаризованого кола в частотній області при впливі одного або декількох сигналів (*AC Sweep*);

– перехідні процеси при впливі сигналів різної форми (*Transient Analysis*);

– спектральний аналіз (*Fourier Analysis*);

– «статистичні випробування» за методом Монте-Карло та розрахунок найгіршого випадку (*Monte Carlo/Worst Case*);

– багатоваріантний аналіз при варіації температури (*Temperature*) та ряд інших параметрів.

За допомогою модуля *PSpice Optimizer* виконується параметрична оптимізація. Кожному виду розрахунку відповідає певна директива. Їх повний перелік наведено в табл. 2.5.

Запуск програми та створення вхідного файлу

Щоб застосовувати *Spice* для аналізу кіл, необхідно створити вхідний файл (за допомогою існуючих текстових редакторів). Файли, якими користується програма *PSpice*, мають розширення *.cir.

Таблиця 2.5 – Директиви моделювання

Ім’я	Призначення
1	2
Розрахунок стандартних характеристик	
.AC	Розрахунок частотних характеристик
.DC	Розрахунок режиму по постійному струму
.FOUR	Спектральний аналіз
.NOISE	Розрахунок рівня внутрішнього шуму
.OP	Передача в вихідний файл параметрів схеми, яка лінеаризована біля робочої точки
.SENS	Розрахунок малосигнальної чутливості в режимі по постійному струмі
Управління видачею результатів	
.PLOT	Представлення результатів розрахунку у вихідному файлі у вигляді графіків, побудованих в текстовому режимі
.PRINT	Представлення результатів розрахунку у вихідному файлі у вигляді таблиць
.PROBE	Передача даних у графічний постпроцесор Probe
.VECTOR	Створення файлу з результатами моделювання цифрових пристроїв
.WATCH	Видача проміжних результатів аналізу на екран програми *PSpice* в текстовому вигляді
.WIDTH	Призначення довжини рядків вихідного файлу
Багатоваріантний аналіз	
.STEP	Варіація параметрів
.TEMP	Призначення температури навколишнього середовища
Допоміжні файли, визначення функцій і параметрів	
.END	Кінець завдання
.FUNC	Визначення функції
.INC	Включення у вхідній файл іншого файлу
.LIB	Підключення бібліотеки моделей компонентів
.PARAM	Визначення глобальних параметрів
Статистичний аналіз	
.MC	Статистичний аналіз за методом Монте–Карло
.WCASE	Розрахунок найгіршого випадку

Продовження табл. 2.5

1	2
Моделі пристроїв	
.ENDS	Кінець опису макромоделі
.DISTRIBUTION	Табличне визначення закону розподілу випадкових величин
.MODEL	Опис моделей компонентів
.SUBCKT	Початок опису макромоделі
Задання початкових умов	
.IC	Задання початкових умов
.LOADBIAS	Зчитування з файла вузлових потенціалів схеми
.NODESET	Задання вузлових потенціалів по постійному струму на початковій ітерації
.SAVEBIAS	Запис у файл вузлових потенціалів схеми
Інші директиви	
.ALIASES	Початок списку відповідностей імен виводів графічних позначень компонентів іменам кіл схеми, до яких вони підключені
.ENDALIASES	Кінець списку відповідностей
.EXTERNAL	Специфікація зовнішніх портів
.OPTIONS	Установка параметрів і режимів роботи програми
.STIMLIB	Завдання імені файла з описом зовнішніх впливів
.STIMULUS	Завдання зовнішніх впливів
.TEXT	Завдання текстових змінних, текстових виразів або імен файлів
*	Коментар
;	Коментар у кінці рядка
+	Продовження рядка

Запускаємо програму *PSpice* та виберемо в меню *File – New – Text File*. При створенні вхідного файлу для схеми необхідно завжди починати з ескізу та розмічання вузлів. У схемі повинен бути присутній нульовий вузол (0), який є опорним. Інші вузли повинні мати цифрові або літерні маркування. Вибираємо ім’я вхідного файлу з розширенням *.cir,

яке вказує на вхідний файл, в який включаємо команду для кожного компонента схеми. Команди для компонентів можуть йти в довільній послідовності, першою командою вхідного файлу повинна йти назва або опис. Якщо в першому рядку буде опис компонента, він буде проігнорований. Останньою повинна бути команда .END.

Зауваження: не натискайте ENTER після введення останнього рядка (.END), тому що в цьому випадку програма вирішить, що ви хочете приступити до наступного аналізу.

Логічні схеми

Робоча версія *PSpice* містить більше сотні логічних пристроїв. На рис. 2.51 показана схема, яка містить логічний елемент АБО-НІ.

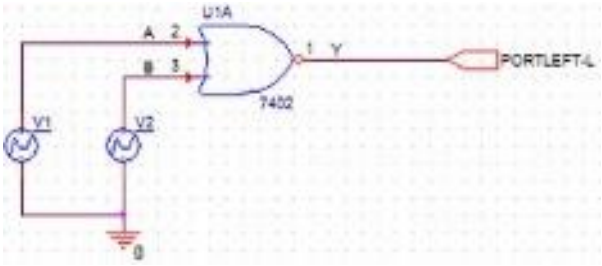


Рисунок 2.51 – Логічний елемент АБО-НІ

Напруги на двох входах *A* та *B* мають амплітуду 5 В. Схема АБО-НІ вводиться за допомогою виклику підпрограми (командою *X*), в якій вузли 1, 2, 3 належать до входів *A*, *B* та виходу *Y*. При виклику підпрограми пристрій називається 7402. Остаточний вигляд вхідного файлу:

Digital Circuit Using NOR gate
VCC 4 0 5V
X 1 2 3 7402
V1 1 0 PWL(0s 0V 0.1ms 5V 1s 5V 1.0001s 0V 2s 0V 2.0001s 5V 3s 5V 3.0001s 0V 4s 0V 4.0001s 5V 5s 5V)
V2 2 0 PWL(0s 0V 1.5s 0V 1.50001s 5V 2.5s 5V 2.50001s 0V 3.5s 0V 3.50001s 5V 3.7s 5V 3.70001s 0V 5s 0V)

```
R 4 3 100k
.opt nopage
.lib eval.lib
.tran 0.01ms 5s
.probe
.end
```

Розглянемо зміст наведеного файлу. В нього входять вхідні дані компонентів схеми. Для кожного компонента вони показані в окремому рядку, що містить дані: тип елемента, вузли, значення основного параметра (опір, напруга та ін.). Позначення .OPT (скорочення від .OPTIONS) nopage запобігає перериванню моделювання після закінчення сторінки вихідного файлу. Команда .END обов'язкова для будь-якого вхідного файлу. Команда .PROBE показує, що при аналізі повинна бути використана програма побудови графіків *Probe*. Зазначимо, що порядок проходження вузлів в запису R 4 3 100k означає, що умовним позитивним напрямком струму буде вважатися напрям від вузла 4 до вузла 3. Для запуску моделювання вибираємо *Simulation* – *Run*. Коли моделювання закінчиться, на екрані виникне область, в якій може бути наведений графік. Це вікно програми *Probe*, підключеної до складу *PSpice*. Щоб отримати графік, який називається *Trace*, вибираємо *Trace* – *Add Trace* (рис. 2.52).

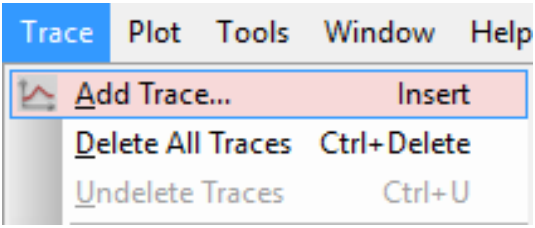


Рисунок 2.52 – Вибір елементів, які будуть відображатися на графіку

У вікні *Trace Expressions* вибираємо напругу $V(1)$, $V(2)$, $V(3)$, як показано на рис. 2.53.

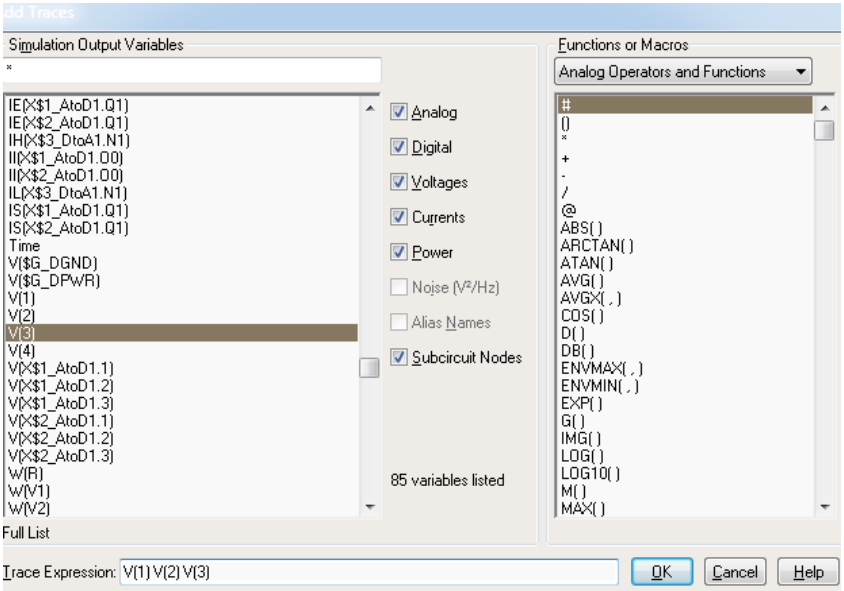


Рисунок 2.53 – Трасування напруг

В результаті отримаємо вікно, на якому зображено сигнали. При необхідності програма дозволяє змінити зовнішній вигляд графіка. Для цього необхідно вибрати *Tools* – *Options*. У відкритому вікні вибираємо вкладку *Color Settings*, а в полі *Background* (фон) вибираємо білий колір (рис. 2.54).

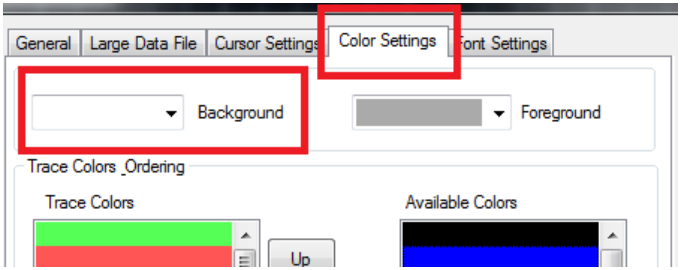


Рисунок 2.54 – Вибір вкладки зміни кольору графіків

Для зміни кольору графіка необхідно клацнути ПКМ по лінії графіка та вибрати *Trace Property*. У відкритому вікні змінимо колір (*Color*) та товщину (*Widht*) лінії.

Щоб отримати в *Probe* напруги V (1) V (2) V (3) у вигляді трьох окремих графіків, розташованих один під одним, необхідно вибрати *Plot – Add Plot to Window*, як показано на рис. 2.55.

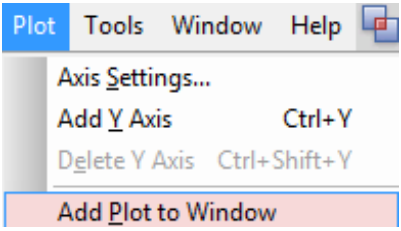


Рисунок 2.55 – Меню додавання графіків

Потім клацнути ПКМ по полю, яке з'явилося, і вибравши *Add Trace*, додати напругу $V(2)$. Аналогічно виконуємо дії для $V(1)$. Результат показано на рис. 2.56.

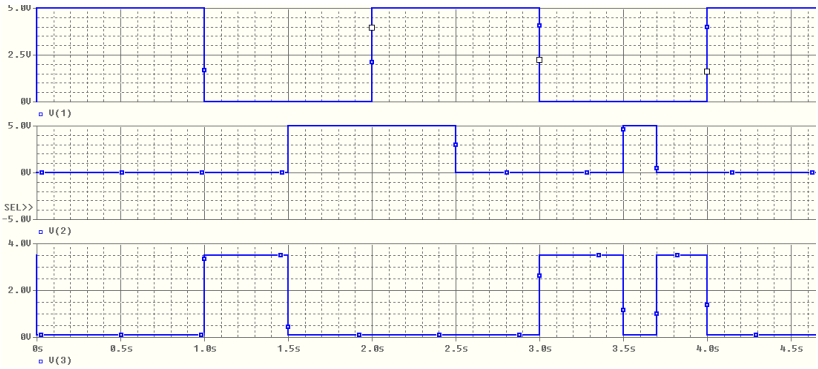


Рисунок 2.56 – Результат моделювання

Для перегляду вихідного файла вибираємо *View – Output File* (рис. 2.57).

```
Digital Circuit Using NOR gate

****     CIRCUIT DESCRIPTION

*********************************************************************************


VCC 4 0 5V
X 1 2 3 7402
V1 1 0 PWL(0s 0V 0.1ms 5V 1s 5V 1.0001s 0V 2s 0V 2.0001s 5V 3s 5V 3.0001s 0V 4s 0V 4.0001s 5V 5s 5V)
V2 2 0 PWL(0s 0V 1.5s 0V 1.50001s 5V 2.5s 5V 2.50001s 0V 3.5s 0V 3.50001s 5V 3.7s 5V 3.70001s 0V 5s 0V)
R 4 3 100k
.opt nopage
.lib eval.lib
.tran 0.01ms 5s
.probe
.end
```

Рисунок 2.57 – Перегляд програми опису схеми

Розглянемо схему, наведену на рис. 2.58.

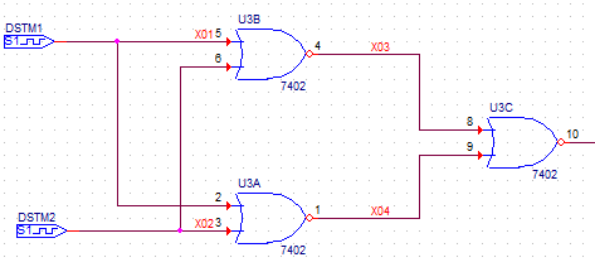


Рисунок 2.58 – Схема з трьома елементами АБО-НІ

Вхідний файл буде мати такий вигляд:

Digital Circuit Using NOR gate
VCC 6 0 5V
X02 1 2 3 7402
X12 1 2 4 7402
X22 3 4 5 7402
V1 1 0 PWL(0s 0V 0.2s 5V 1s 5V 1.2s 0V 2s 0V 2.0001s 5V 3s 5V 3.0001s 0V 4s 0V 4.0001s 5V 5s 5V)
V2 2 0 PWL(0s 0V 1.5s 0V 1.7s 5V 2.5s 5V 2.50001s 0V 3.5s 0V 3.50001s 5V 3.7s 5V 3.70001s 0V 5s 0V)
R 6 5 100k

```
.opt nopage
.lib eval.lib
.tran 0.01ms 5s
.probe
.end
```

Результат моделювання наведено на рис. 2.59.

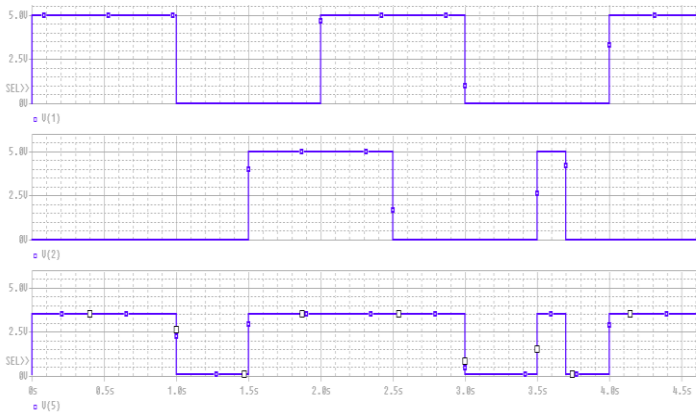


Рисунок 2.59 – Результат моделювання схеми

Розглянемо схему, наведену на рис. 2.60 (для виконання аналогового моделювання в схему був доданий резистор *R*).

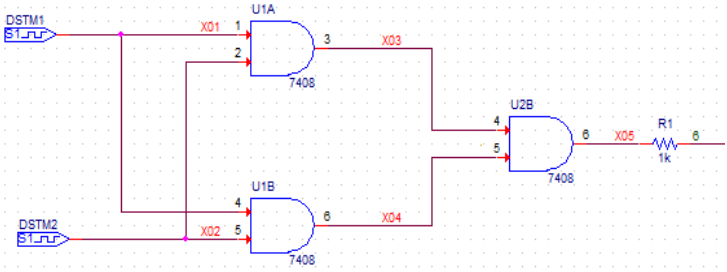


Рисунок 2.60 – Схема з трьома елементами 2І та резистором

Вхідний файл буде мати такий вигляд:

Digital Circuit Using AND gate
VCC 6 0 5V
X02 1 2 3 7408
X12 1 2 4 7408
X22 3 4 5 7408
V1 1 0 PWL(0s 0V 0.1ms 5V 1s 5V 1.0001s 0V 2s 0V 2.0001s 5V 3s 5V 3.0001s 0V 4s 0V 4.0001s 5V 5s 5V)
V2 2 0 PWL(0s 0V 1.5s 0V 1.50001s 5V 2.5s 5V 2.50001s 0V 3.5s 0V 3.50001s 5V 3.7s 5V 3.70001s 0V 5s 0V)
R 6 5 100k
.opt nopage
.lib eval.lib
.tran 0.01ms 5s
.probe
.end

Результат моделювання наведено на рис 2.61.

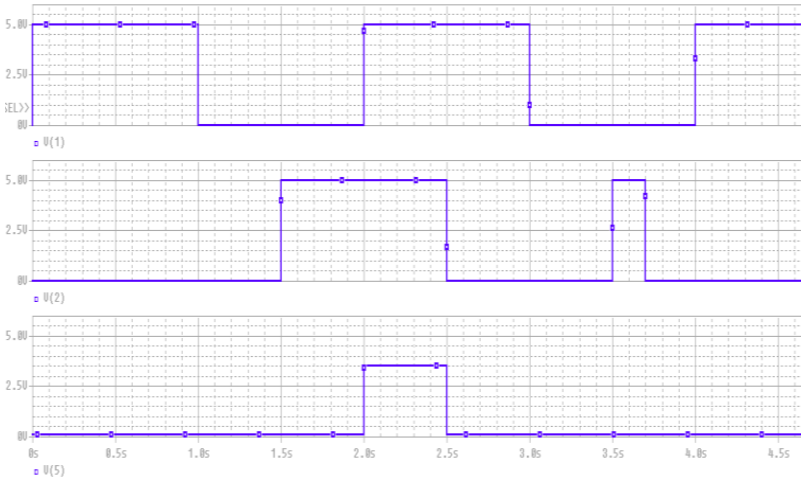


Рисунок 2.61 – Результат моделювання

Розглянемо схему, яка наведена на рис. 2.60, з іншими вхідними параметрами для V1 та V2:

Digital Circuit Using NOR gate
VCC 6 0 5V
X02 1 2 3 7402
X12 1 2 4 7402
X22 3 4 5 7402
V1 1 0 PWL(0s 0V 0.2s 5V 1s 5V 1.2s 0V 2s 0V 2.2s 5V 3s 5V 3.2s 0V 4s 0V 4.2s 5V 5s 5V)
V2 2 0 PWL(0s 0V 1.5s 0V 1.7s 5V 2.5s 5V 2.7s 0V 3.5s 0V 3.7s 5V 3.8s 5V 4s 0V 5s 0V)
R 6 5 100k
.opt nopage
.lib eval.lib
.tran 0.01ms 5s
.probe
.end

Результат моделювання наведено на рис. 2.62 – 2.63.

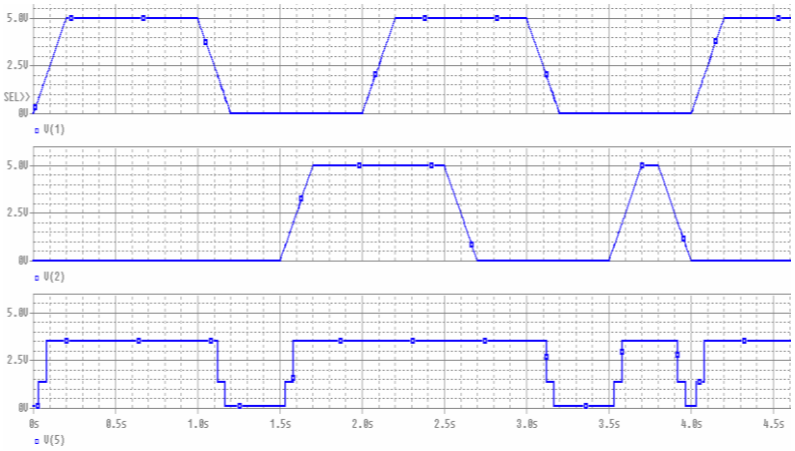


Рисунок 2.62 – Результати моделювання

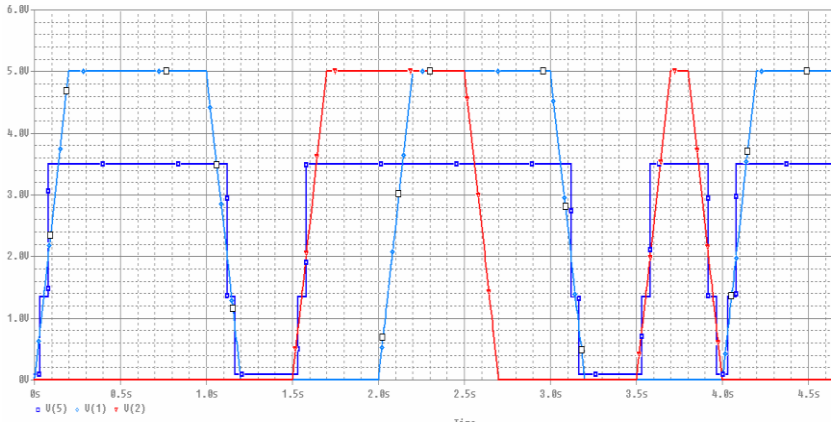


Рисунок 2.63 – Результати моделювання

Для складання схем з безлічі елементів буде зручно подавати схему у вигляді орієнтованого графа, в якому логічні елементи – вершини, а з'єднання між ними – ребра графа. Таким чином, отриманий граф може бути поданий як список вершин з вхідними та вихідними ребрами.

Розглянемо приклад схеми, яка виконує логічний вираз:

$$Y = \overline{A}B + C\,.$$

На рис. 2.64 наведено схему для обчислення, яка створена у вигляді Schematic, а на рис. 2.65 наведено граф.

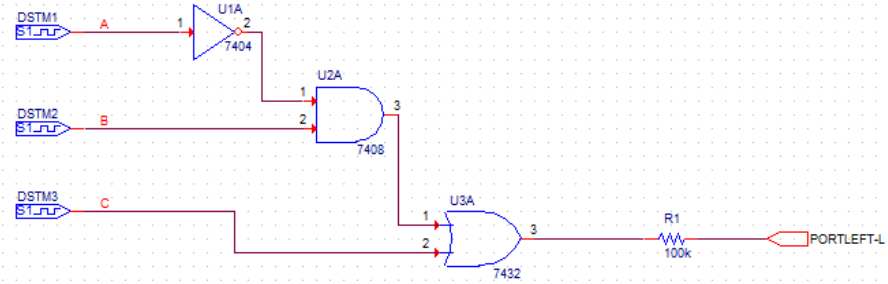


Рисунок 2.64 – Схема у Schematic

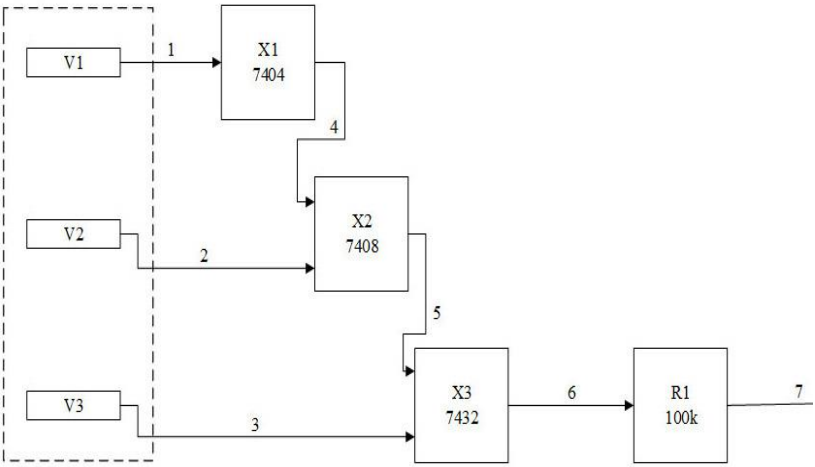


Рисунок 2.65 – Граф пристрою

Як видно на рис. 2.65, кожна вершина має назву та відповідний код логічного елемента або, як у випадку з резистором, його опір. Ребра нумеруються за бажанням користувача.

Загальну структуру файлу .cir можна подати таким чином:

1. Рядок назви

Digital Circuit Using Func3 gate

2. Живлення схеми

VCC 7 0 5V

У цьому рядку перше число – вихід схеми, друге – вхід (0 вказує, що це джерело сигналу).

3. Опис логічних вузлів у форматі

IM'Я вхід1...вхідN вихід1…вихідN опис елемента

X1 1 4 7404

4. Опис джерел сигналу

IM'Я вихід 0 PWL (t1 V1 t2 V2 .... tN VN)

V1 1 0 PWL(0s 0V 0.2s 5V 1s 5V 1.2s 0V 2s 0V 2.2s 5V 3s 5V 3.2s 0V 4s 0V 4.2s 5V 5s 5V)

5. Опис резистора

Внаслідок того, що обидва виходи резистора рівнозначні, порядок написання вхідних та вихідних ребер неважливий.

R 6 7 100k

6. Завершальні рядки

.opt nopage
.lib nom.lib
.tran 0.01ms 5s
.probe
.end

Таким чином, код для моделювання пристрою буде виглядати так:

Digital Circuit Using Logic gate
VCC 7 0 5V
X1 1 4 7404
X2 4 2 5 7408
X3 3 5 6 7432
V1 1 0 PWL(0s 0V 0.2s 5V 1s 5V 1.2s 0V 2s 0V 2.2s 5V 3s 5V 3.2s 0V 4s 0V 4.2s 5V 5s 5V)
V2 2 0 PWL(0s 0V 1.5s 0V 1.7s 5V 2.5s 5V 2.7s 0V 3.5s 0V 3.7s 5V 3.8s 5V 4s 0V 5s 0V)
V3 3 0 PWL(0s 0V 0.2s 5V 1.2s 5V 1.4s 0V 2.2s 0V 2.4s 5V 3.4s 5V 3.6s 0V 3.8s 0V 4s 5V 4.2s 5V)
R 6 7 100k
.opt nopage
.lib nom.lib
.tran 0.01ms 5s
.probe
.end

Результати моделювання описаного пристрою наведено на рис. 2.66.

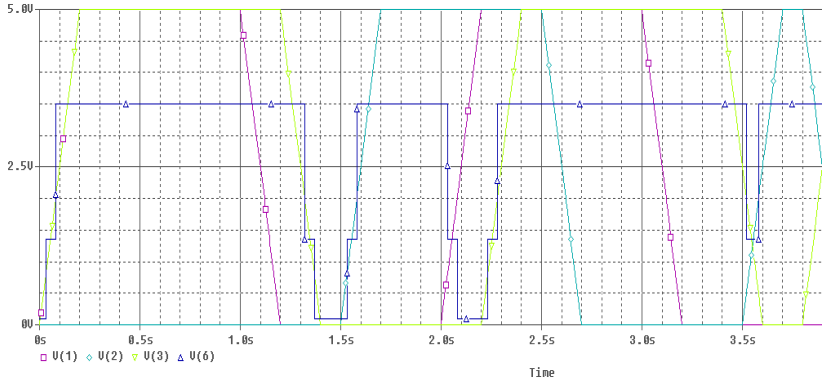


Рисунок 2.66 – Результати моделювання

Завдання для виконання практичної роботи

Написати програму та виконати аналогове моделювання створеного в практичній роботі № 3 пристрою. На вхід спроєктованої схеми подавати сигнали трапецієподібного вигляду.

Практична робота 5

Використання програми Stimulus Editor для тестування пристрою

Мета роботи: розглянути процес опису та створення вхідних впливів, які використовуються для тестування пристрою за допомогою додатка *Stimulus Editor* САПР OrCAD.

Проєктування сучасної електронної апаратури – це ітераційний процес, що складається з етапів функціонального проєктування, розробки принципової схеми та друкованої плати, виготовлення, випробувань і внесення змін у початковий проєкт. САПР електронних пристроїв OrCAD є прикладом повністю закінченої системи наскрізного проєктування, яка містить засоби креслення принципових та функціональних схем, засоби електричного моделювання, засоби розробки друкованих плат, засоби оформлення креслень.

Дана система містить такі складові частини:

– графічний редактор схем OrCAD *Capture*;

– програмний засіб для розробки друкованих плат OrCAD *Layout*;

– засіб моделювання цифрових пристроїв та інтерфейс з програмами проєктування логічних інтегральних схем, що програмуються, OrCAD *Express*;

– програма моделювання аналогових пристроїв OrCAD *PSpice*;

– програма моделювання аналого-цифрових пристроїв OrCAD *PSpice A/D*;

– програма параметричної оптимізації OrCAD *PSpice Optimizer*;

– редактор файлів електричних моделей елементів OrCAD *PSpice Model Editor*;

– редактор файлів, що містять опис сигналів складної форми OrCAD *PSpice Stimulus Editor*.

Процес створення електронного пристрою за допомогою САПР OrCAD розбивають на декілька етапів: створення схеми пристрою; тестування створеної друкованої плати (моделювання роботи пристрою); підготовка конструкторської документації.

Призначення утиліти Stimulus Editor

Утиліта (редактор) *Stimulus Editor* дозволяє створювати та перевіряти вхідні сигнали для аналізу перехідних процесів у компонентах пристрою. *Stimulus Editor* дозволяє створювати та редагувати джерела напруги, джерела струму, цифрові сигнали.

Для створених аналогових (цифрових) сигналів *Stimulus Editor* створює файли (з розширенням .STL). Сигнали створюються в автоматизованому режимі та для роботи з ними не потрібно знати їх синтаксис.

Список файлів *Stimulus* на вкладці *Configuration Files* діалогового вікна *Simulation Settings* (рис. 2.67) дозволяє переглянути список файлів сигналів, що належать поточній схемі. На цій же вкладці діалогового вікна можна додавати, видаляти та змінювати конфігураційний файл. Список відображає всі поточні зміни файлів сигналів.

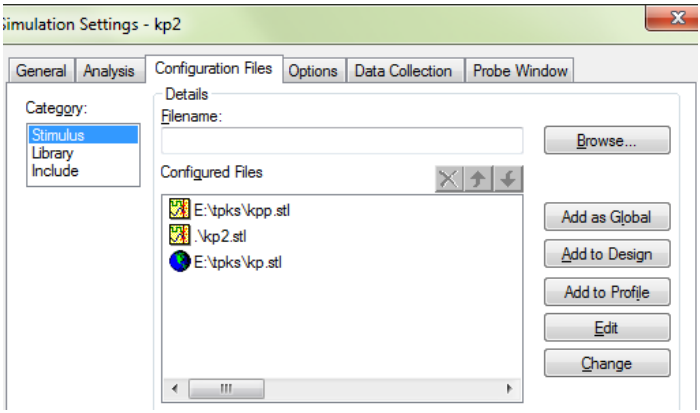


Рисунок 2.67 – Діалогове вікно *Simulation Settings*

Файли сигналів можуть бути налаштовані як глобальні (відзначаються перед назвою значком) – для редактора схем *Capture*, або локальні – для поточного проєкту.

Коли *Stimulus Editor* запускається з редактора схем *Capture*, файли сигналів автоматично додаються в список як локальні для поточного проєкту. Також нові файли сигналів можуть бути додані в список

введенням назви файлу в текстове поле *Filename*, а потім натисканням *Add to Profile* (конфігурація профайла), *Add to Design* (локальна конфігурація), *Add as Global* (глобальна конфігурація).

Вставка та налаштування параметрів генератора сигналів

Запустимо *Stimulus Editor* з меню *Cadence – Release – PSpise Accessories.* Виберемо *File – New.* Коли *Stimulus Editor* запускається вперше, необхідно упорядкувати установки масштабування для налаштування графіків. Можна використовувати команду *Axis Settings* в меню *Plot* (рис. 2.68).

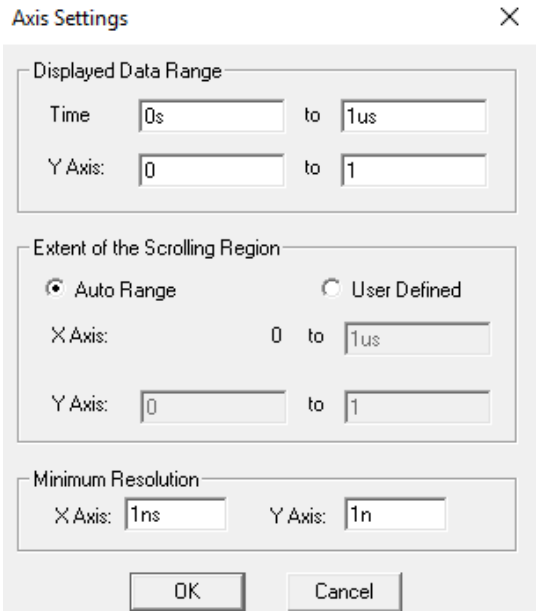


Рисунок 2.68 – Встановлення параметрів

Параметри, які відображаються в *Displayed Data Range*, визначають, яка частина цих сигналів буде показана на екрані. Параметр *Extent of the Scrolling Region* встановлює абсолютні межі видимого діапазону. В *Minimum Resolution* визначається мінімальний дозвіл по вісях *X* та *Y*.

Визначення виду сигналу

Для визначення виду сигналу необхідно:

1. Розмістити на схемі зразок символу генератора сигналу з бібліотеки символів SOURCSTM.OLB. Для прикладу виберемо DIGSTIM1 (рис. 2.69).

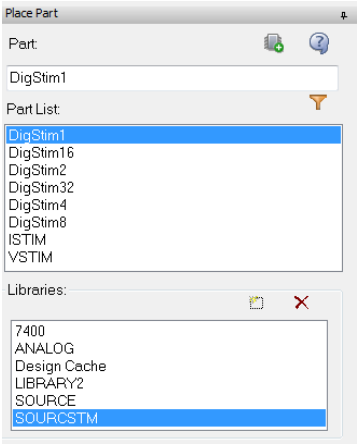


Рисунок 2.69 – Вибір генератора сигналу

2. Клацнути ЛКМ на імені генератора для його вибору.

3. Для запуску *Stimulus Editor* клацнути ПКМ на символі генератора в схемі. У контекстному меню (рис. 2.70) вибрати пункт *Edit PSpice Stimulus*.

4. У діалоговому вікні *New Stimulus* надрукуємо DIGSTIM1 в текстовому полі *Name*. В *Digital* виберемо *Signal* (рис. 2.70).

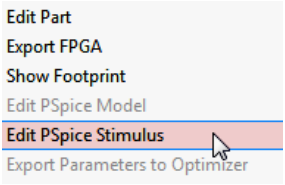


Рисунок 2.70 – Контекстне меню генератора

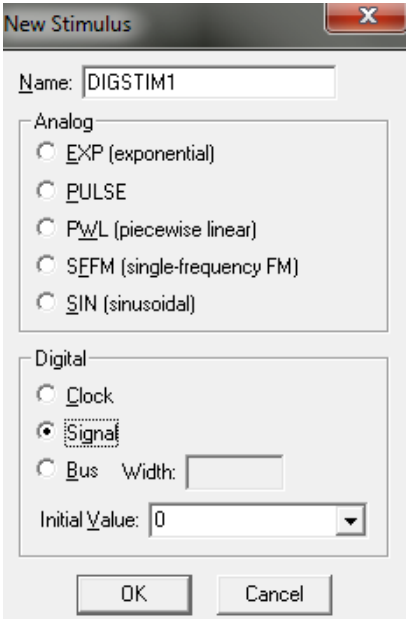


Рисунок 2.71 – Діалогове вікно *New Stimulus*

Налаштування параметрів логічного стану цифрового сигналу

При визначенні цифрового сигналу можна виконувати такі дії: додавати момент часу зміни логічного стану (*transition*); видаляти, редагувати, переміщати момент часу зміни логічного стану (ці операції не можуть бути застосовані до сигналу генератора *Clock*).

Для додавання моментів часу зміни логічного стану необхідно:

1. Вибрати *File – New*.

1. В меню *Plot* виконати команду *Axis Settings*.

2. Ввести необхідні значення в текстові поля *Displayed Range for Time* та *Timing Resolution* для додавання моментів часу зміни логічного стану (рис. 2.72).

3. Вибрати цифровий сигнал, який необхідно відредагувати. При виборі *Edit – Add* курсор миші набуває вигляду олівця (рис. 2.73).

4. Клацнути ЛКМ по сигналу в тому місці, де потрібна зміна логічного стану.

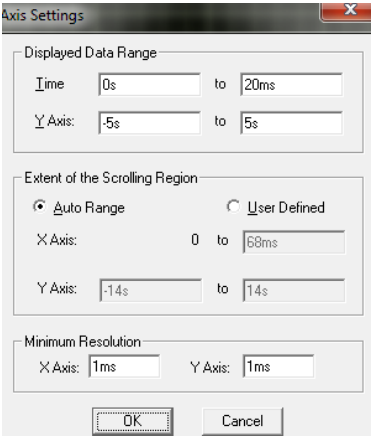


Рисунок 2.72 – Діалогове вікно *Axis Settings*

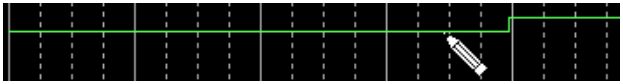


Рисунок 2.73 – Приклад редагування цифрового сигналу

5. Повторити п. 4 для додавання додаткових моментів часу зміни логічного стану.

6. Після закінчення редагування клацнути ПКМ для виходу з режиму редагування.

Для переміщення моментів часу зміни логічного стану:

1. Клацанням миші відмітити точку переходу в новий стан.

2. При необхідності можна використовувати клавішу SHIFT для вибору додаткових функцій.

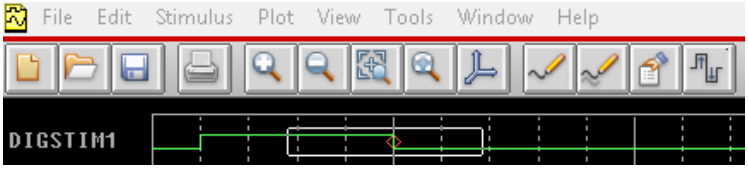


Рисунок 2.74 – Переміщення часу зміни сигналу

Для редагування моментів часу зміни логічного стану необхідно:

1) виконати одну з таких дій:

– вибрати момент часу зміни логічного стану, який потрібно відредагувати (потім в меню *Edit* – виконати пункт *Attributes*);

– двічі клацнути ЛКМ на точці зміни логічного стану, яку необхідно відредагувати;

2) у діалоговому вікні *Edit Digital Transition* (рис. 2.75) відредагувати час та значення для моменту часу зміни логічного стану;

3) натиснути кнопку *OK*.

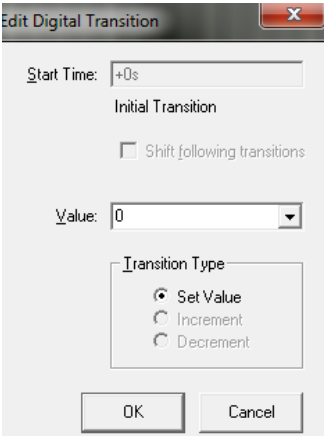


Рисунок 2.75 – Діалогове вікно *Edit Digital Transition*

Для видалення моментів часу зміни логічного стану необхідно:

1. Клацнути ЛКМ по моменту часу зміни логічного стану.

2. Якщо необхідно, можна використовувати клавішу SHIFT.

3. У меню *Edit* виконати команду *Delete*.

Для створення сигналу *Clock* необхідно:

1. Вибрати у *Stimulus Editor* генератор сигналів, який потрібно використовувати як *Clock*.

2. У меню Stimulus вибрати *Change Type* (рис. 2.76). Відкриється діалогове вікно *Change Stimulus Type*. В *Type* необхідно зазначити перемикач *Clock* (рис 2.77). Натиснути кнопку *OK*.

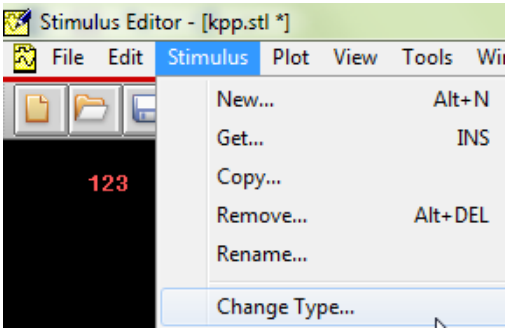


Рисунок 2.76 – Команди меню *Stimulus*

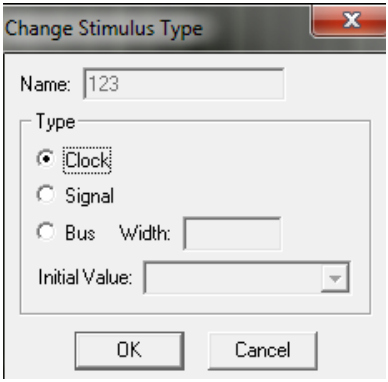


Рисунок 2.77 – Діалогове вікно *Change Stimulus Type*

3. Ввести значення для атрибутів сигналу *Clock*, використовуючи діалогове вікно *Clock Attributes*. Як приклад, на рис. 2.78 в *Clock Attributes* введені такі параметри: частота (*Frequency*) – «20 МГц», коефіцієнт заповнення імпульсної послідовності (*Cycle*) – «0,5», початкове значення (*Initial Value*) – «1» та часова затримка (*Time Delay*) – «5 нс».

4. У меню *File* виконати команду *Save*.

Для зміни атрибутів сигналу *Clock* необхідно або двічі клацнути ЛКМ на назві сигналу *Clock* зліва від осі, або клацнути на назві сигналу *Clock* та в меню *Edit* виконати команду *Attributes.* Потім можна виконати необхідні зміни атрибутів сигналу *Clock*.

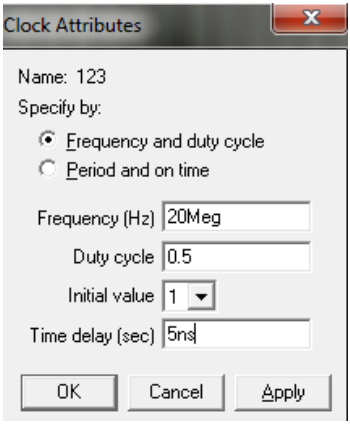


Рисунок 2.78 – Діалогове вікно *Clock Attributes*

При створенні нового сигналу в меню виберіть тип сигналу. Сигнали розділені на дві групи – аналогові та цифрові. В даному випадку вибираємо *Clock* з групи *Digital*. У текстовому полі введіть назву сигналу (рис. 2.79). Натискаємо *OK*.

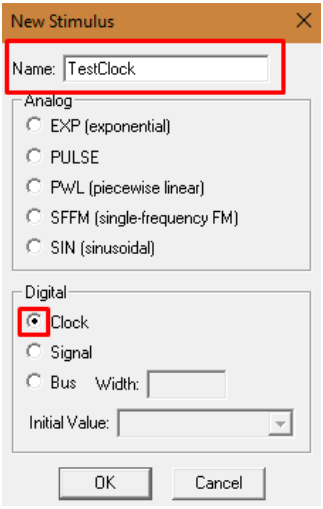


Рисунок 2.79 – Вибір типу сигналу

З’явиться вікно налаштування сигналу, а також сигнал, який поки що є прямою лінією. У вікні налаштування сигналу можна визначити два режими: частотний та часовий. Є кнопка *Apply*, яка дозволяє переглядати результати, не закриваючи вікно змін. Кнопка *OK* збереже поточний стан сигналу та закриє вікно.

Частотний режим

Frequency (частота) – задається в Гц.

Duty cycle – визначає співвідношення часу знаходження сигналу в положенні «1» (0.5 – порівну, 1 – тільки в положенні «1», 0 – тільки в положенні «0»).

Initial value – початкове значення сигналу.

Time delay – затримка початку.

Рисунок 2.80 – Частотний режим

Часовий режим

Period – час, за який відбудеться повний цикл.

On time – аналог того ж, що й у частотному режимі. Різниця лише в тому, що тепер це число від «0» до значення періоду (чим більше значення, тим більше часу буде сигнал у положенні «1»).

Рисунок 2.81 – Часовий режим

Створення довільного цифрового сигналу

Створіть новий сигнал, але тепер виберіть пункт *Signal*. Можна також указати початкове значення сигналу (рис. 2.82). Натисніть *OK*.

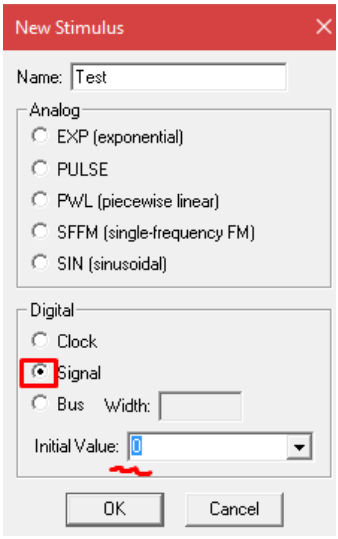


Рисунок 2.82 – Встановлення початкового значення сигналу

З’явиться сигнал, який являє собою пряму лінію. Для його редагування використовуємо інструмент редагування: *Edit – Add*.

Клацнемо ЛКМ по сигналу. Якщо з’являється повідомлення з попередженням про неможливість встановлення точки (рис. 2.83), зайдіть в *Plot – Axis Settings* та в нижньому полі вікна, що з’явилося (рис. 2.84), встановіть значення 1 ns.

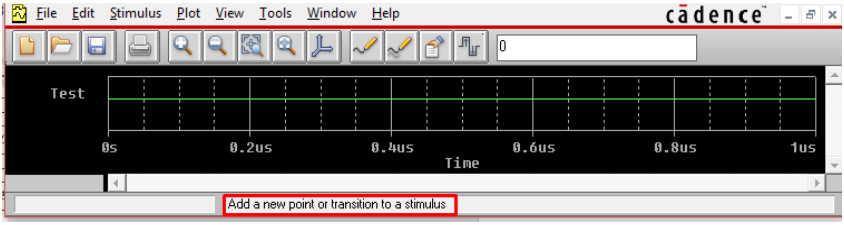


Рисунок 2.83 – Повідомлення про помилку

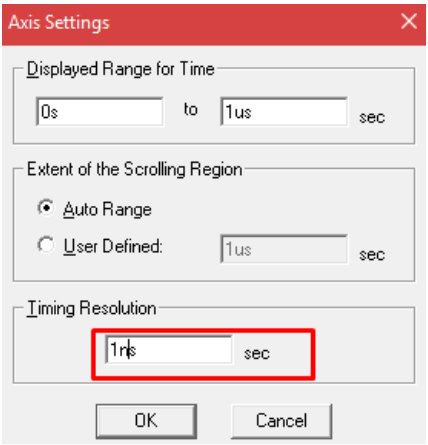


Рисунок 2.84 – Встановлення *Timing Resolutions*

«Олівцем» клацнемо по сигналу – це створить опорну точку. Значення сигналу інвертується за поточною точкою та до кінця сигналу за умови, що далі немає інших опорних точок (рис. 2.85).

Рисунок 2.85 – Змінений сигнал

Для створення багаторозрядних сигналів (шин) необхідно виконати:

– створити багаторозрядний цифровий сигнал;

– розставити моменти часу зміни логічного стану;

– визначити додатково вид системи числення для багаторозрядного сигналу.

Для створення багаторозрядних джерел сигналів необхідно:

1. У меню *Stimulus* виконати команду *New.*

2. У текстове полі *Name* діалогового вікна *New Stimulus* (рис. 2.86) ввести ім’я джерела – *Bus.*

3. В *Digital* вибрати перемикач *Bus.*

4. Для встановлення кількості виходів (*bus width*) в текстовому полі *Width* необхідно ввести їх необхідну кількість.

5. Натиснути кнопку *OK.*

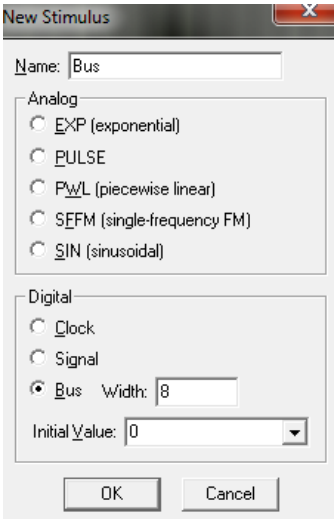


Рисунок 2.86 – Діалогове вікно *New Stimulus*

Для вставки моментів часу зміни логічного стану необхідно:

1. У меню *Plot* виконати команду *Axis Settings*.

2. Ввести потрібні значення в текстові поля *Displayed Range for Time* та *Timing Resolution* для додавання моментів часу зміни логічного стану:

а) постійне значення: 12,

б) приріст: +12; H,

в) негативний приріст: –12; O.

3. У меню *Edit* виконати команду *Add*.

4. У полі цифрового значення на панелі інструментів (рис. 2.101) ввести текстовий опис, що визначає вигляд багаторозрядного сигналу. Якщо не вводиться значення вигляду системи числення (*radix*), *Stimulus Editor* використовує вигляд системи числення за замовчуванням.

Рисунок 2.87 – Поле цифрового значення сигналу
на панелі інструментів

5. Клацнути ЛКМ по сигналу.

6. Повторити п. 4 – 5 стільки разів, скільки необхідно.

7. Після закінчення клацнути ПКМ для виходу з режима редагування (рис. 2.88).

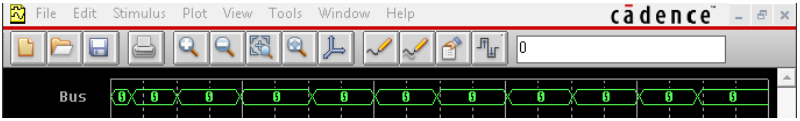


Рисунок 2.88 – Отримані багаторозрядні джерела сигналів

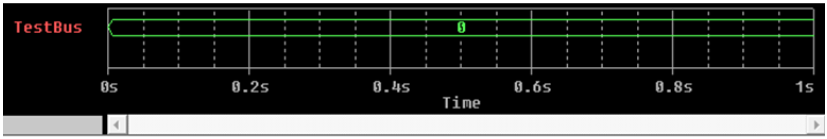
Для встановлення системи числення за замовчуванням необхідно:

1. У меню *Tools* виконати команду *Options*.

2. У списку *Radix* рамки *Bus Display Defaults* вибрати потрібний вигляд системи числення за замовчуванням.

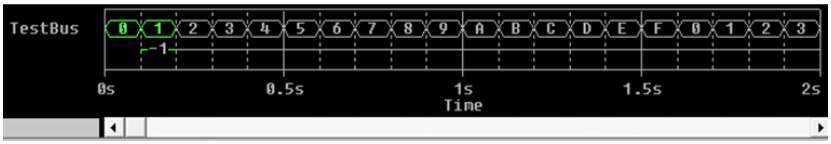
3. Натиснути кнопку *OK*.

Шину також можна редагувати «олівцем», проте це не дуже зручно. Найпростіше редагувати шину за допомогою текстового режиму. Виконаємо деякі зміни. На рис. 2.89 наведено графік сигналу до редагування, а на рис. 2.90 – графік сигналу після редагування.




```
* by Stimulus Editor -- Serial Number: 0 -- Version 17.2.0
;!Stimulus Get
;! TestBus Digital
;!Ok
;!Plot Axis_Settings
;!Xrange 0s 1us
;!AutoUniverse
;!XminRes 1ns
;!YminRes 1n
;!Ok
.STIMULUS TestBus STIM (4, 1111) ;! Hex
+   +0s 0000
```

Рисунок 2.89 – Вигляд до редагування




```
* by Stimulus Editor -- Serial Number: 0 -- Version 17.2.0
;!Stimulus Get
;! TestBus Digital
;!Ok
;!Plot Axis_Settings
;!Xrange 0s 1us
;!AutoUniverse
;!XminRes 1ns
;!YminRes 1n
;!Ok
.STIMULUS TestBus STIM (4, 1111) ;! Hex
+   +0s 0000
+ Repeat Forever
+ 0.1s INCR BY 0001
+ Endrepeat
```

Рисунок 2.90 – Вигляд після редагування

Розглянемо задачу створення сигналу, який подібний до існуючого. Наприклад, є сигнал, який показано на рис. 2.91. Потрібно створити сигнал, який складається з трьох послідовностей (рис. 2.92).

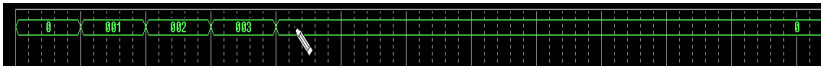


Рисунок 2.91 – Цифровий багаторозрядний сигнал

Рисунок 2.92 – Сформований цифровий багаторозрядний сигнал

Дану задачу можна вирішити, використовуючи стандартний текстовий редактор. В цьому файлі є послідовність моментів часу зміни логічного стану, яка створена в начальному сигналі. Можна змінити опис сигналу, повторюючи його кілька разів, для цього необхідно виконати:

1. В *Stimulus Editor* зберегти та закрити файл сигналу.

2. В стандартному текстовому редакторі відкрити файл сигналу.

3. Знайти послідовність рядків, що містять текст, який потрібно повторити. Кожен рядок починається в момент часу зміни логічного стану або зміни значення сигналу. Перед цим рядком вставимо наступний рядок:

+ Repeat for n_times,

де *n_times* є або позитивним цілим числом (число повторень), або ключовим словом FOREVER, яке забезпечує повторення цієї послідовності необмежене число разів.

5. Нижче цього рядка вставити рядок:

+ Endrepeat.

6. У меню *File* виконати команду *Save*.

Якщо послідовність необхідно повторити три рази, то файл сигналу потрібно змінити, як показано нижче (додавши рядки, виділені жирним шрифтом):

+ Repeat for 3
+ +0s 000000000
+ 250us INCR BY 000000001
+ 500us 000000010

+ 750us INCR BY 000000001

+ 1 ms 000000000

+ End repeat

Створення аналогових сигналів

Створення експоненціального сигналу

Створіть новий сигнал, вибравши пункт *EXP* (рис. 2.93), введіть ім'я та натисніть *OK*.

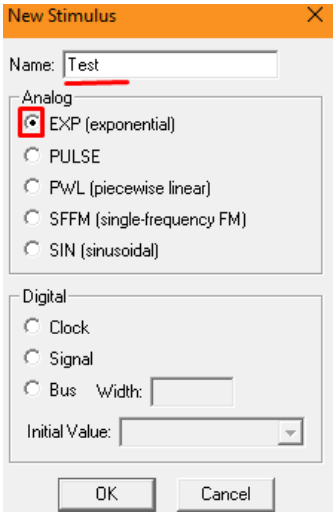


Рисунок 2.93 – Вибір типу сигналу

На рис. 2.94 показано вікно налаштувань:

Initial Value – початкове значення (можна встановити будь-яке);

Peak value – максимальне значення (можна встановити будь-яке);

Rise delay – затримка початку сигналу на певний час;

Rise time constant – вказує, через скільки секунд значення сигналу буде вище за 60 %;

Fall Delay – момент часу, коли сигнал почне спадати;

Fall time constant – вказує, через скільки секунд значення сигналу буде знаходитися нижче 60 %.

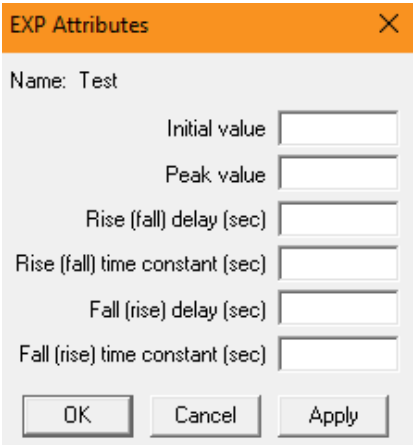


Рисунок 2.94 – Вікно налаштування сигналу

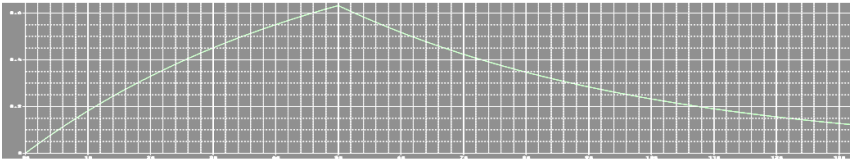


Рисунок 2.95 – Експоненціальний сигнал

Створення імпульсного сигналу.

Створіть новий сигнал, вибравши пункт *PULSE*, введіть ім'я та натисніть *OK*.

На рис. 2.96 показано вікно налаштування сигналу:

Initial Value – початкове значення (можна встановити будь-яке);

Peak value – максимальне значення (можна встановити будь-яке);

Delay – затримка;

Rise time – тривалість фронту;

Fall time – тривалість спаду;

Pulse width – час, в якому сигнал буде знаходиться в своєму максимумі;

Period – період сигналу.

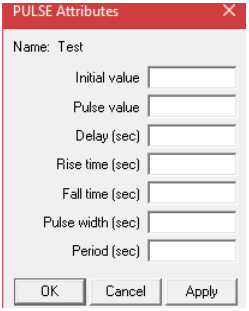


Рисунок 2.96 – Вікно налаштування сигналу

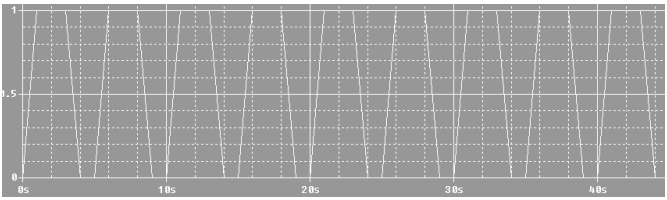


Рисунок 2.97 – Імпульсний сигнал

Довільний прямолінійний графік

Створіть новий сигнал, вибравши пункт *PWL*, та введіть ім'я. За допомогою «олівця» прокладаємо лінії. Задавати ламану лінію зручно в текстовому режимі. На рис. 2.98 наведено код фігури, зображеної на рис. 2.99. Координати точок обведені прямокутником.

```
;!Ok
.STIMULUS dsa PWL
+ TIME_SCALE_FACTOR = 1
+ VALUE_SCALE_FACTOR = 1
+    ( 0, 0 )
+    ( 1e-007, 4.95e-007 )
+    ( 1.05e-007, 3.74e-007 )
+    ( 2.2e-007, 3.42e-007 )
+    ( 2.28e-007, 3.82e-007 )
+    ( 2.29e-007, 3.44e-007 )
+    ( 4e-007, 3.75e-007 )
+    ( 4.25e-007, 4.24e-007 )
+    ( 4.38e-007, 4.51e-007 )
+    ( 4.86e-007, 3.79e-007 )
+    ( 4.87e-007, 3.78e-007 )
+    ( 5.27e-007, 2e-009 )
```

Рисунок 2.98 – Код фігури

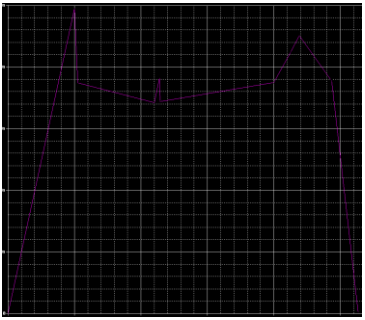


Рисунок 2.99 – Фігура, створена за допомогою ламаних ліній

Синусоїдальні сигнали

Створіть новий сигнал, вибравши пункт *SIN*, введіть ім'я та натисніть *OK*. На рис. 2.100 показано вікно налаштування сигналу:

Offset value – значення вертикального зсуву (визначає, на скільки пунктів зрушити графік щодо осі *Y*);

Amplitude – амплітуда (максимальне або мінімальне значення);

Frequency – частота;

Time delay – зрушення щодо осі *X* праворуч на певний час;

Damping factor (фактор згасання) – чим більше значення, тим сильніше коливання загасають, а потім знову збільшуються. Якщо поставити «1», тоді коливання будуть слабкішими або сильнішими в 2 рази на кожному новому періоді щодо попереднього;

Phase angle – фаза коливання (задається в градусах).

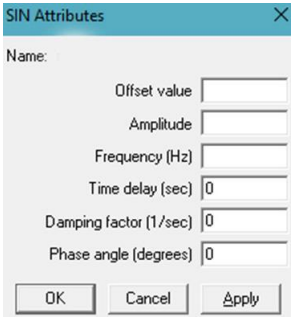


Рисунок 2.100 – Вікно параметрів синусоїдального сигналу

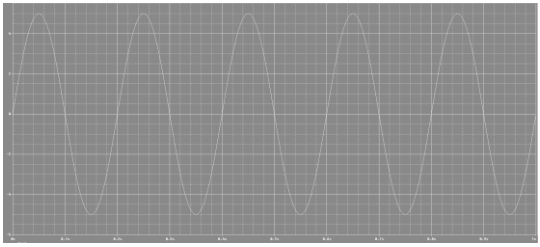


Рисунок 2.101 – Синусоїдальний сигнал

Створення кусково-лінійного сигналу

Для створення кусково-лінійного сигналу необхідно:

1. Відкрити існуючу схему або створити нову.

2. У меню *Place* виконати команду *Part*. Знайти файл бібліотеки символів «SOURCSTM.OLB», вибрати «VSTIM» та позначити його на схемі.

3. Клацнути ЛКМ по властивості *Implementation*.

4. У діалоговому вікні *Display Properties* (рис. 2.102) ввести в поле *Value* ім'я створюваного сигналу – «*Vfirst*».

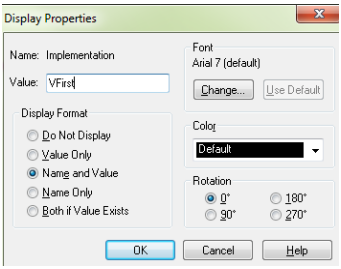


Рисунок 2.102 – Діалогове вікно *Display Proporties*

5. Якщо робота здійснюється в новій схемі – виконати команду *Save* меню *File*.

6. Для вибору символу VSTIM необхідно клацнути по ньому ЛКМ.

7. Клацнути ПКМ по генератору на схемі та вибрати *Edit PSpice Stimulus* в контекстному меню.

8. Запуститься *Stimulus Editor* та відобразиться діалогове вікно *New Stimulus* (рис. 2.103), на якому показано, що сигнал має назву *Vfirst.*

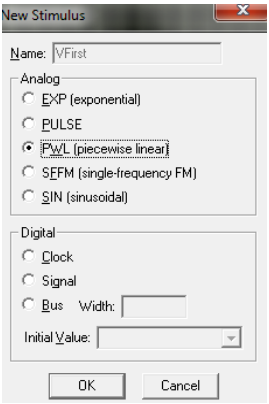


Рисунок 2.103 – Діалогове вікно *New Stimulus*

8. Зазначити перемикач *PWL* та натиснути кнопку *OK.* Курсор набуде вигляду олівця. Повідомлення в рядку стану внизу екрана вкаже, що відбувається процес додавання нових точок даних для сигналу.

9. Перемістити курсор та натиснути ЛКМ. Таким чином додаються та з'єднуються точки, які визначають сигнал (рис. 2.104).

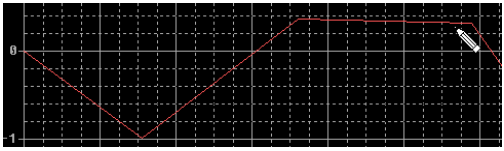


Рисунок 2.104 – Редагування (створення) сигналу

10. Щоб зупинити додавання точок клацніть ПКМ.

11. У меню *File* виконайте команду *Save*.

12. Для видалення точки клацніть на мітці-маніпуляторі та натисніть кнопку *Delete*. Для додавання точок зламу сигналу *PWL* виконайте команду *Add* в меню *Edit* (можна скористатися кнопкою панелі

інструментів *Add Point*).

Для створення синусоїдального сигналу з частотою 10 кГц та змінною амплітудою необхідно виконати наступні кроки:

1. Відкрити існуючу схему та помістити на ній символ джерела VSTIM. Клацнути ПКМ по ньому та вибрати *Edit PSpice Stimulus.*

3. У діалоговому вікні *New Stimulus* виконати команду *Cancel*.

4. Визначити параметри сигналу: в меню *Tools* виконати команду *Parameters* та в текстовому полі *Definition* ввести AMP = 1 (рис. 2.105).

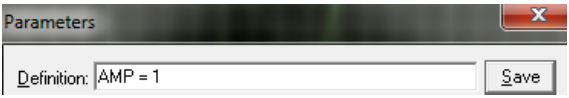


Рисунок 2.105 – Діалогове вікно *Parameters*

5. Виконати команду *New* меню *Stimulus*, в діалоговому вікні ввести ім’я сигналу Vsin та вибрати *SIN* (рис. 2.106).

6. Визначити інші властивості сигналу в діалоговому вікні SIN *Attributes* (рис. 2.107): ввести «0» в полі *Offset Value*; ввести {AMP} в поле *Amplitude* (фігурні дужки вказують, що вираз необхідно оцінювати під час моделювання); ввести «10k» в поле *Frequency* та натиснути *OK*; в меню *File* виконати команду *Save*.

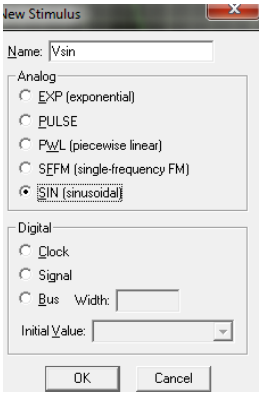


Рисунок 2.106 – Діалогове вікно *New Stimulus*

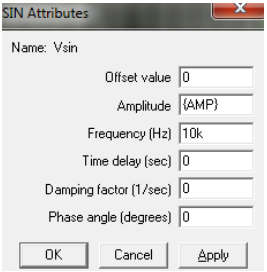


Рисунок 2.107 – Діалогове вікно SIN *Attributes*

8. У редакторі схем *Capture* помістити та визначити символ PARAM:

– у меню *Place* виконати команду *Part*;

– вибрати символ PARAM в SPECIAL.OLB;

– помістити символ на схемі та двічі клацнути ЛКМ по ньому;

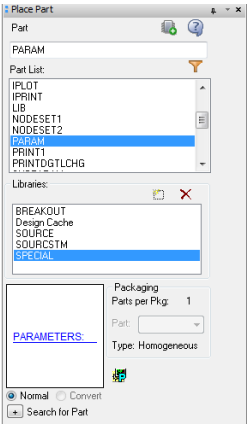


Рисунок 2.108 – Діалогове вікно *Place Part*

– натиснути New для додавання нового користувача властивості AMP;

– ввести AMP (без фігурних дужок) в поле *Name* редактора властивостей *Property Editor*;

– встановити значення властивості AMP рівним «1» (рис. 2.109).

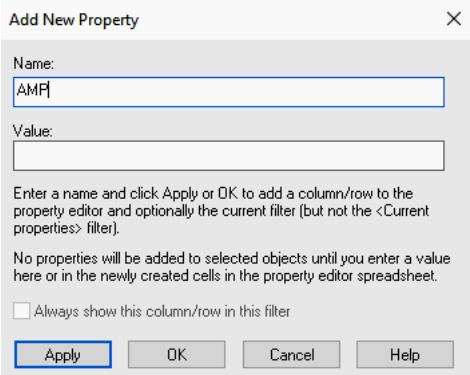


Рисунок 2.109 – Діалогове вікно *Add New Property*

9. Встановити такі властивості:

– в меню *Pspice* виберемо *Edit Simulation Profile* та виберемо опцію *Parametric Sweep*;

– виберемо *Global Parameter* в *Sweep variable*;

– виберемо *Linear* в *Sweep type;*

10. Введемо AMP в текстове поле *Parameter.*

11. Визначимо значення для текстових полів *Start Value, End Value* та *Increment*.

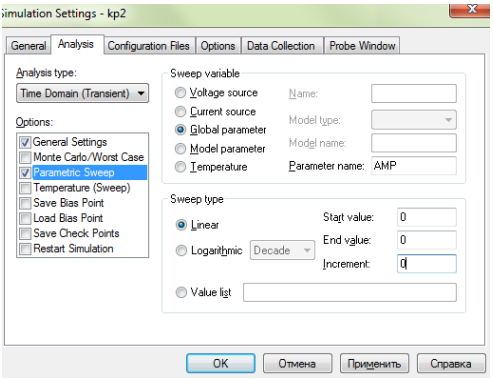


Рисунок 2.110 – Діалогове вікно *Simulation Settings*

Можна встановити аналіз Transient, AC або DC та виконати моделювання. Використовуємо *Part editor* в *Capture* для створення і редагування нових символів джерел сигналу з властивостями, наведеними в табл. 2.6.

Таблиця 2.6 – Властивості символів

Властивість	Значення
Implementation type	PSpice Stimulus
Implementation	Назва моделі сигналу
STIMTIPE	Тип сигналу набуває значення ANALOG або DIGITAL

Для редагування існуючих сигналів необхідно запустити *Stimulus Editor*, вибрати *File/Open* для відкриття необхідної бібліотеки сигналів. Потім двічі клацнути по назві графіка (внизу осі *X* для аналогових та зліва від осі *Y* для цифрових сигналів). Відкриється вікно *Stimulus Attributes*, де можна змінити атрибути сигналу, спостерігаючи ефект від змін.

Створення призначеного для користувача сигналу в «ручному» режимі

У *Stimulus Editor* сигнал може бути створений вручну, а його технічні характеристики збережені у файлі. Ці характеристики потім можуть бути пов’язані зі зразком сигналу на схемі або символом джерела сигналу в бібліотеці символів.

Для конфігурації сигналу вручну виконаємо такі дії:

1. Запустимо *Stimulus Editor*, вибравши *Stimulus Editor* в меню Windows Пуск*/OrCAD.../PSpice Accessories/.*

2. Відкриємо файл сигналу, вибравши *Open* в меню *File.*

3. Створимо один або кілька сигналів для використання в схемі.

Для кожного сигналу:

– поставимо будь-яку назву;

– забезпечимо технічні вимоги до режиму перехідних процесів;

– у меню *File* виберемо *Save.*

4. У схематичному редакторі сторінки сконфігуруємо вихідний файл *Stimulus Editor* для схеми:

– для відображення діалогового вікна *Simulation Settings* в меню *Pspice* виберемо *Edit Simulation Profile*;

– у діалоговому вікні *Simulation Settings* виберемо вкладку *Configuration Files*;

– натиснемо *Include* в полі *Category* для відображення списку файлів *Include* (рис. 2.111);

– введемо назву файла.

– для поточного профайла натиснемо кнопку *Add to Profile*. Для локального проєкту натиснемо кнопку *Add to Design*. Для глобального проєкту натиснемо кнопку *Add as Global.*

– натиснемо *OK.*

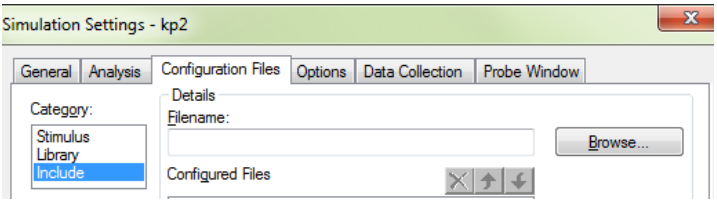


Рисунок 2.111 – Діалогове вікно *Simulation Settings*

5. Модифікуючи будь-який зразок сигналу на схемі або символи в бібліотеці символів, пошлемося на нові технічні вимоги сигналу.

6. Зв'яжемо технічні характеристики сигналу для режиму перехідних процесів із зразком джерела сигналу:

– помістимо символ генератора сигналу на схемі: VSTIM, ISTIM або DIGSTIM та клацнемо на кожному з них;

– у меню *Edit* виберемо *Properties;*

– виберемо середовище *Implementation*, введемо назву сигналу і натиснемо *Apply*.

Designator	Graphic	ID	Implementation
	VSTIM.Normal		NAME

Рисунок 2.112 – Фрагмент діалогового вікна *Properties*

– закінчимо завдання будь-яких прикладів VSTIM або ISTIM, вибираючи в меню *Edit – Properties* та редагуючи їх атрибути для DC та AC аналізу: в DC введемо значення постійної складової напруги (струму), в AC введемо значення амплітуди гармонійної складової напруги (струму);

– закриємо *Property editor*.

7. Для глобальної зміни сигналу для символу:

– виберемо символ, який необхідно відредагувати;

– у меню *Edit* виберемо *Part* для запуску *Part editor*;

– створимо або змінимо опис символу, визначаючи властивість: *Implementation* – назва сигналу, як визначено в *Stimulus Editor*.

Завдання для виконання практичної роботи

Виконати за допомогою програми *Stimulus Editor* створення синусоїдального, імпульсного, експоненціального сигналів з періодом $T = N$ мкс (де N – номер варіанта).

Практична робота 6

Використання OrCAD для моделювання цифрових блоків на основі методу Монте-Карло

Мета роботи: виконати моделювання пристрою та дослідити його за допомогою методу Монте-Карло.

Метод Монте-Карло – загальна назва групи методів, які основані на отриманні великої кількості реалізацій випадкового процесу, сформованого таким чином, щоб його ймовірнісні характеристики співпадали з аналогічними величинами розв'язуваної задачі.

Метод Монте-Карло – це метод імітації для приблизного відтворення реальних явищ, який дозволяє побудувати модель, мінімізуючи та максимізуючи значення даних. Побудова моделі починається з визначення функціональних залежностей в реальній системі, після чого можна отримати рішення.

Не існує єдиного методу Монте-Карло, цей термін описує широко використовуваний клас підходів. Ці підходи використовують у своїй основі єдиний шаблон:

1) визначити область можливих вхідних даних;

2) випадковим чином згенерувати вхідні дані за допомогою деякого заданого розподілу ймовірностей;

3) виконати детерміновані обчислення над вхідними даними;

4) проміжні результати окремих розрахунків звести в кінцевий результат.

Ідея методу Монте-Карло полягає в тому, що в обчислювальній машині створюється процес перетворення цифрових даних, аналогічний реальному процесу. Ймовірнісні характеристики обох процесів (реального і змодельованого) співпадають з певною точністю.

Головний недолік цього методу полягає в тому, що він не може замінити аналітичні методи при розрахунку істотно нових явищ, де потрібне розкриття якісних закономірностей. Перевага методу Монте-Карло полягає в тому, що він здатний «спрацювати» там, де відмовляють інші методи.

Моделювання роботи елемента 4І-НІ

Запустимо САПР OrCAD та зберемо схему, як показано на рис. 2.113.

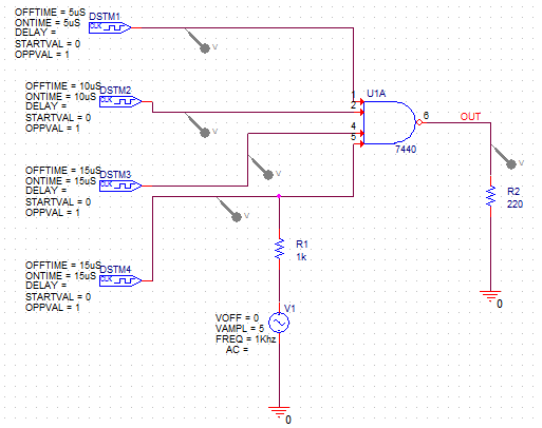


Рисунок 2.113 – Зібрана схема

Отримані в результаті моделювання дані (рис. 2.114) хоча й надають інформацію про поведінку зібраного пристрою, однак не дають можливості розглянути деякі важливі фактори, які можуть вплинути на роботу реального пристрою. Саме для цього й може бути застосований в даному випадку метод моделювання Монте-Карло.

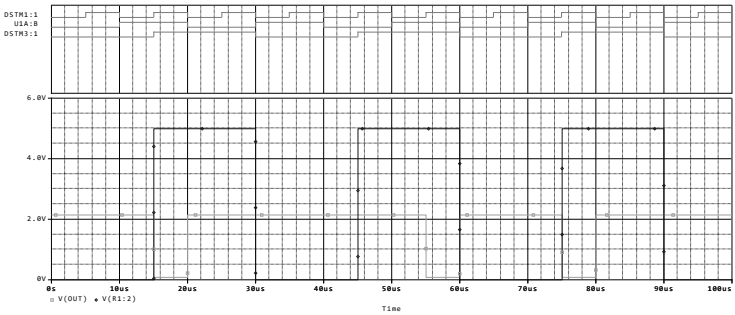


Рисунок 2.114 – Результат моделювання схеми

Моделювання роботи схеми методом Монте-Карло за допомогою PSpice

Внесемо певні зміни в схему, встановивши допустимі відхилення (так звану «толерантність») в налаштуваннях обох резисторів 10 % (рис. 2.115).

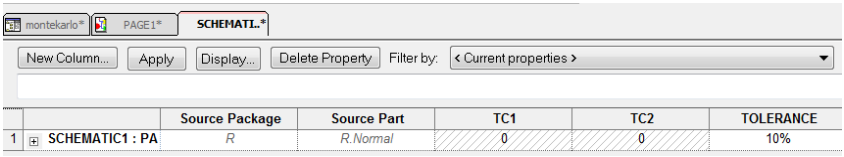


Рисунок 2.115 – Встановлення допустимого відхилення

Тепер в налаштуваннях профілю симуляції вибираємо *Monte Carlo* та встановимо всі необхідні параметри, як показано на рис. 2.116 (результати моделювання показані на рис. 2.117).

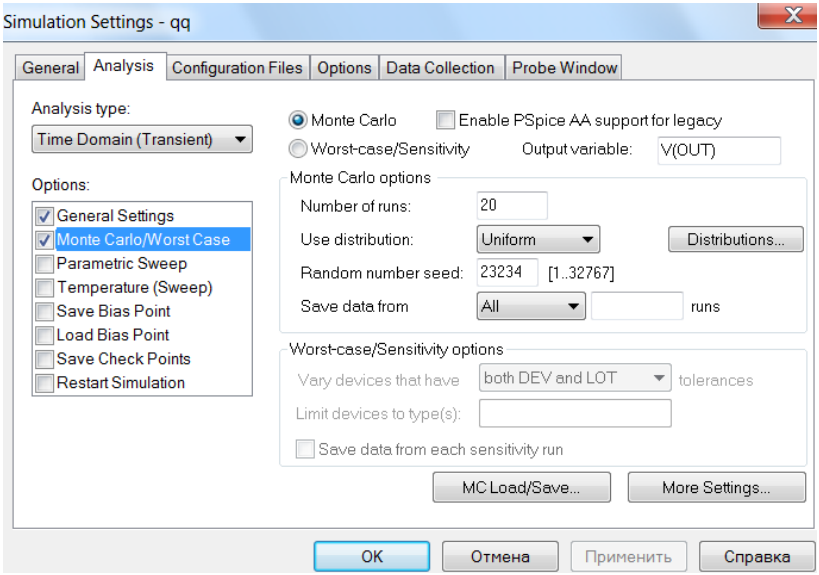


Рисунок 2.116 – Налаштування профілю симуляції

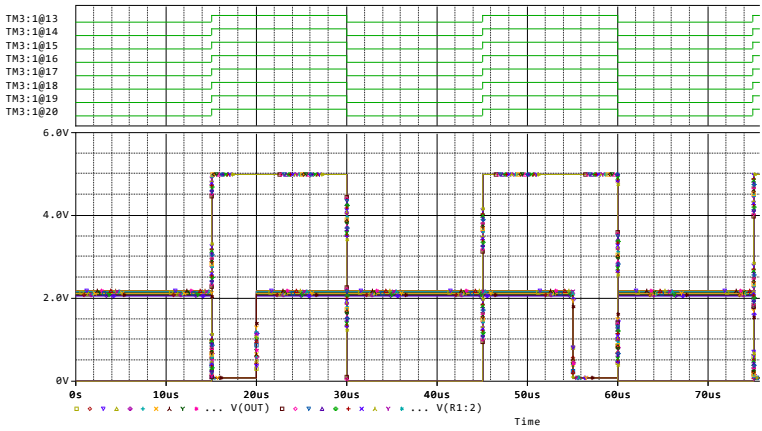


Рисунок 2.117 – Результати моделювання з використанням методу Монте-Карло в *PSpice*

Як видно з рис. 2.117, на виході схеми ми отримуємо напругу в діапазоні 2,13...2,15 В. Це пов'язано з тим, що на виході пристрою підключено резистор опором 220 Ом та допустимим відхиленням 10 %, що дозволяє отримувати різні значення напруги за різною кількістю ітерацій.

Зміна значення напруги на виході обумовлена тим, що при запуску кожної ітерації параметри резисторів на вході та виході схеми змінюються з допустимим відхиленням випадковим чином. При відсутності резистора на виході, ми також отримали б різні значення напруги, але в іншому діапазоні.

Тепер приберемо всі графіки (меню *Trace – Delete all Traces*) та додаємо новий графік, що відображає максимальні значення вихідної напруги (меню *Trace – Add Trace*, в правому списку вибрати MAX (), а в лівому – V (OUT)).

Якщо все буде зроблено правильно (рис. 2.132), то в нижньому текстовому полі буде MAX (V (OUT)). Результат моделювання наведено на рис. 2.133. Незважаючи на те, що отримані результати дозволяють отримати уявлення про те, яка може бути напруга на виході схеми, та в якому діапазоні вона може змінюватися, скористатися ними майже

неможливо. Пов’язано це з тим, що для 20 запусків будується така кількість графіків, яку неможливо адекватно розмістити на одній координатній сітці. Тому треба при моделюванні схеми з використанням методу Монте-Карло використовувати не стандартний *PSpice*, а *PSpice Advanced*.

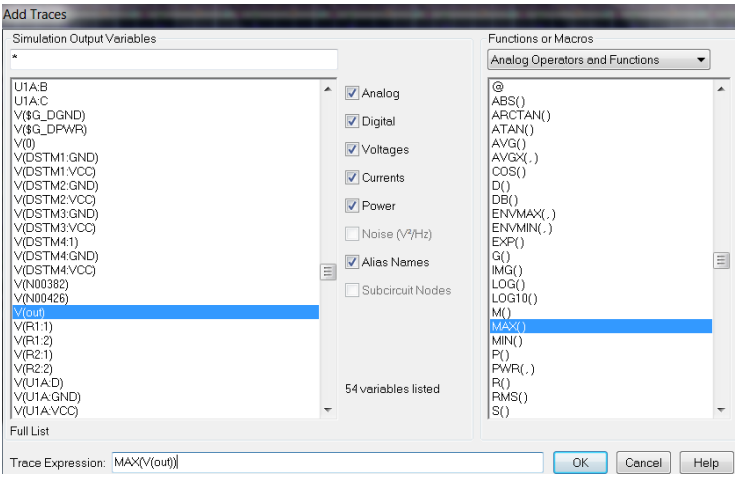


Рисунок 2.118 – Додавання нового графіка

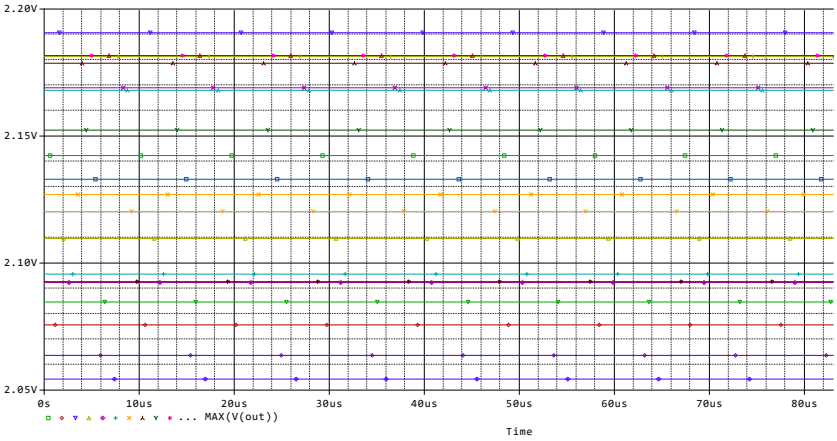


Рисунок 2.119 – Результати моделювання

Моделювання роботи схеми методом Монте-Карло за допомогою PSpice Advanced

Скористаємося раніше розробленою схемою з елементом 4I-HI. Вибираємо *PSpice – Advanced Analysis – Monte Carlo*. Після цього відкриється вікно, як показано на рис. 2.120.

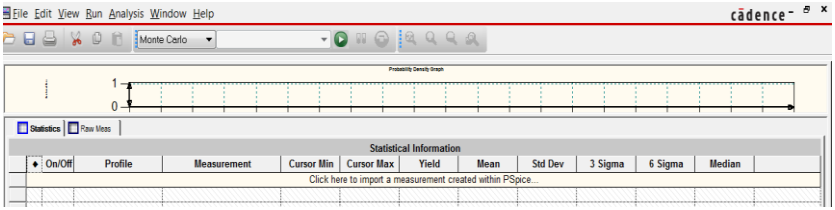


Рисунок 2.120 – Вікно *PSpice Advanced Analysis*

Вибираємо в меню *Analysis – Monte Carlo – Create New Measurements*. У вікні в правому списку вибрати Max (1), а в лівому – V (Out) та натиснути *OK*. Тепер необхідно налаштувати профіль симуляції, виставивши потрібну кількість ітерацій та інші параметри. Для цього вибираємо пункт меню *Edit – Profile Setting* та вводимо необхідні дані у відповідні поля (рис. 2.121):

Number of runs – 20 ітерацій (для кожної ітерації параметри компонента з допустимими відхиленнями будуть випадковим чином змінюватися);

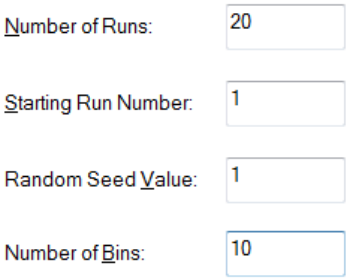


Рисунок 2.121 – Вікно налаштування

Starting run number – початковий номер ітерації (за замовчуванням дорівнює «1») ;

Random seed value – випадкове значення початкового числа для послідовності псевдовипадкових чисел (дорівнює «1»). Генератор випадкових чисел буде використовувати це значення для створення послідовності випадкових чисел;

Number of bins – 10 стовпчиків графіка (гістограми). Це число визначає число секцій в гістограмі.

Запускаємо симуляцію і отримуємо графік функції розподілу ймовірностей (PDF), як показано на рис. 2.122. На рис. 2.123 наведено графік інтегральної функції розподілу (CDF).

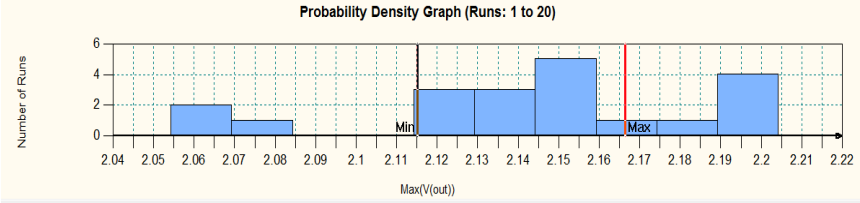


Рисунок 2.122 – Графік функції розподілу ймовірностей

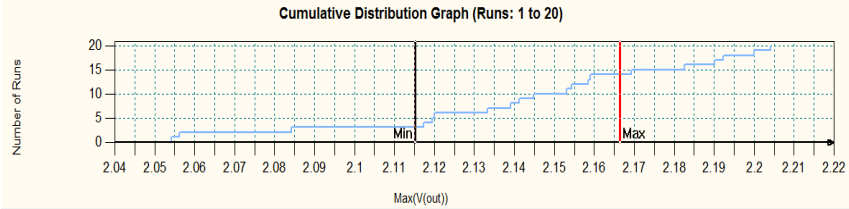


Рисунок 2.123 – Графік інтегральної функції розподілу

Моделювання роботи суматора з використанням методу Монте-Карло

Виконаємо моделювання роботи більш складного пристрою. Спочатку збираємо схему однорозрядного суматора (рис. 2.124), на основі якого збираємо схему чотирирозрядного суматора (рис. 2.125).

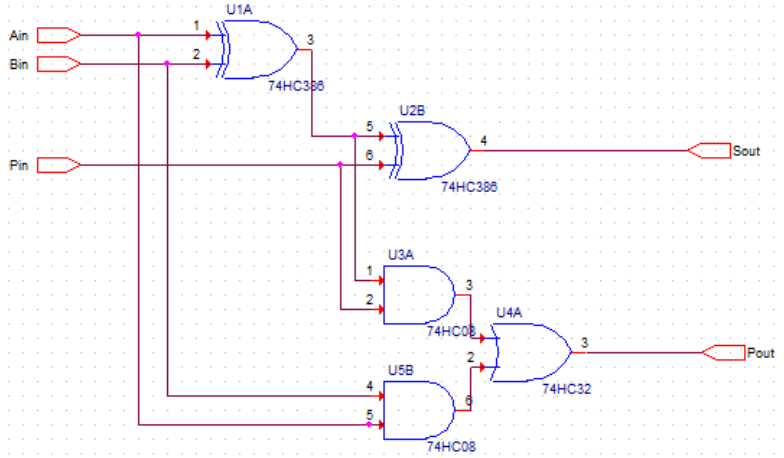


Рисунок 2.124 – Схема однорозрядного суматора

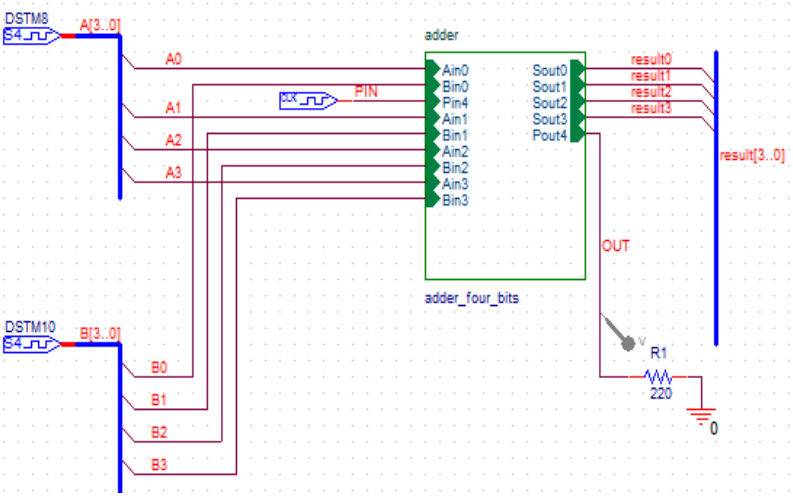


Рисунок 2.125 – Схема дослідження чотирирозрядного суматора

Внутрішня структура макросу чотирирозрядного суматора наведена на рис. 2.126. Налаштування генераторів вхідних сигналів показано на рис. 2.127.

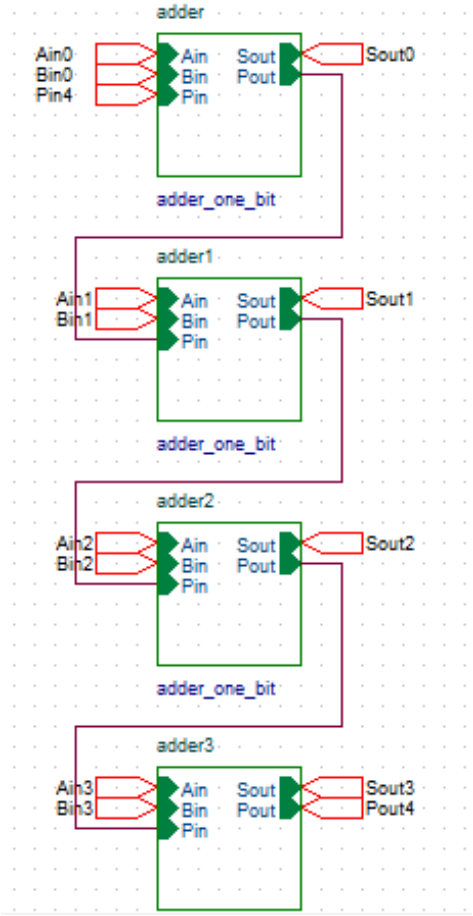


Рисунок 2.126 – Внутрішня структура макросу чотирирозрядного суматора

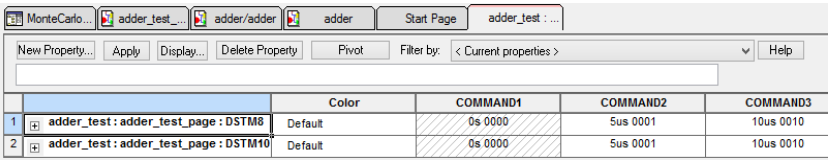


Рисунок 2.127 – Налаштування генераторів вхідних сигналів

Після цього виконуємо раніше описані дії для моделювання отриманого пристрою за допомогою методу Монте-Карло: графік функції розподілу ймовірностей (PDF) наведено на рис. 2.128, графік інтегральної функції розподілу (CDF) наведено на рис. 2.129.

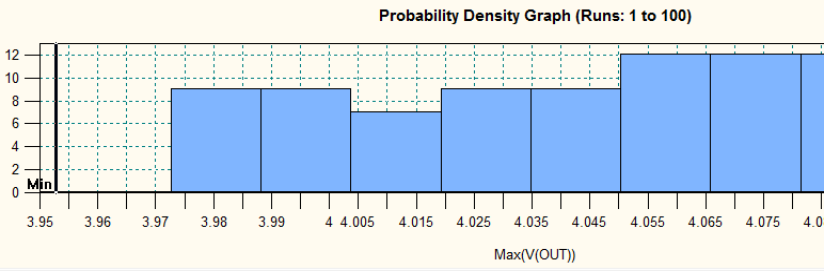


Рисунок 2.128 – Графік функції розподілу ймовірностей

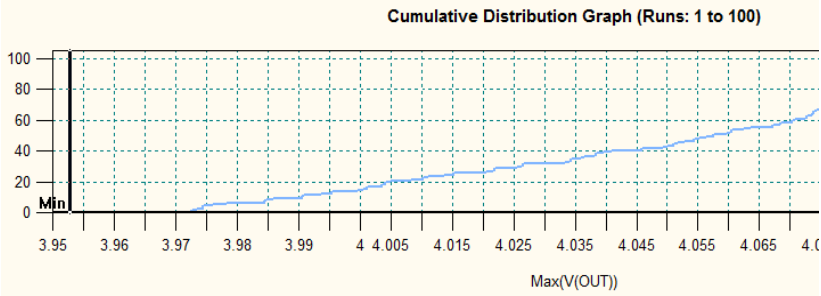


Рисунок 2.129 – Графік інтегральної функції розподілу

Із рис. 2.128 видно, що розподіл максимальної напруги зміщений в бік максимальної позначки. При цьому очевидним стає той факт, що інтегральна функція розподілу має досить великий кут підйому. За цим спостереженням можна зробити висновок, що через різні явища (наприклад, через «перегони» сигналів) значення напруги з часом змінюються. При цьому затримки викликають появу випадкових небажаних сигналів у різні моменти часу, що відображається на графіку функції розподілу ймовірностей.

Завдання для виконання практичної роботи

Виконати моделювання за методом Монте-Карло заданого пристрою відповідно до табл. 2.7.

Таблиця 2.7

№ вар.	Пристрій
1	4-розрядний суматор
2	6АБО
3	5І-НІ
4	шифратор 8×3
5	3АБО-НІ
6	мультиплексор 5×1
7	5АБО-НІ
8	мультиплексор 6×1
9	6-розрядний суматор
10	*D*-тригер
11	*RS*-тригер
12	*JK*-тригер
13	4-розрядний регістр
14	дешифратор на 6 виходів
15	дешифратор на 10 виходів

Практична робота 7
Трасування друкованих плат

Мета роботи: набути практичних навичок зі створення, редагування та трасування друкованих плат у САПР OrCAD за допомогою редактора Allegro PCB.

Приклад виконання завдання

Трасування друкованих плат PCB (*printed circuit board*) *routing* – це покроковий процес прокладання провідників в САПР друкованих плат. Як правило, трасування є заключним етапом конструкторського проєктування і полягає у визначенні ліній, що з'єднують контакти елементів та компонентів. САПР OrCAD для трасування схем, зібраних в OrCAD *Capture*, використовує компонент *Allegro*.

Для створення та трасування друкованої плати необхідно виконати такі етапи:

1) зібрати схему пристрою та призначити необхідні для трасування властивості її елементів;

2) виконати операції анотування, перевірки правил та створення списку з'єднань;

3) провести розміщення компонентів на платі та трасування ліній, що їх з'єднують.

Проєктування будь-якого пристрою починається з принципової схеми – це графічна модель, що демонструє елементи пристрою та зв'язки між ними за допомогою УГП. Ця схема не показує фізичну структуру пристрою, а показує лише логічну структуру і зв'язок між елементами.

Розглянемо всі дії на прикладі 4-розрядного суматора (рис. 2.130). На рис. 2.131 наведено УГП елемента 7408 (елемент I);

– зверху елемент підписаний *U6A*, що відповідає назві конкретного елемента на схемі;

– напис 7408 знизу – це тип елемента, назва мікросхеми, що його реалізує;

– номери 1, 2, 3 біля виводів елемента – це номери ніжок мікросхеми, які відповідають цим входам.

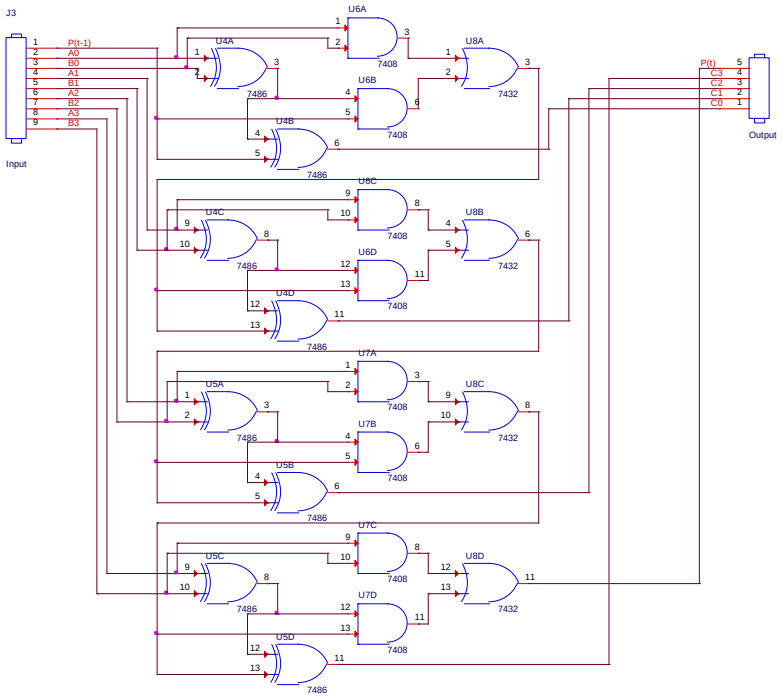


Рисунок 2.130 – Схема суматору

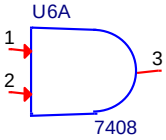


Рисунок 2.131 – УГП елемента І

Умовне зображення внутрішньої структури мікросхеми 7408 наведено на рис. 2.132. У деяких елементів одного типу назви збігаються за першими двома символами, а їх входи і виходи пронумеровані поспіль. Це означає, що у реальної схеми вони реалізуються всередині однієї мікросхеми. Тому для коректності розміщення елементів на друкованій платі у вигляді мікросхем потрібно задати форму посадкового місця для кожного з елементів.

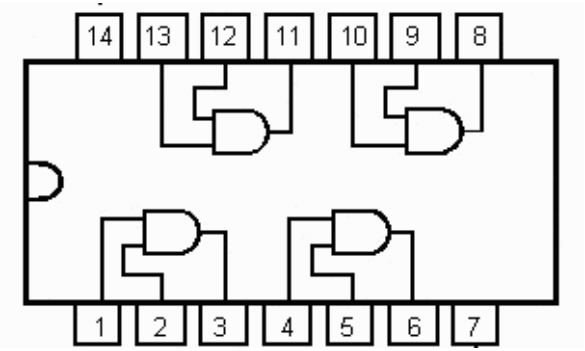


Рисунок 2.132 – Мікросхема 7408

Посадкове місце на друкованій платі (*PCB footprint*) задається однойменним параметром у властивостях елемента. Для його зміни треба клацнути ПКМ по елементу і вибрати пункт *Edit Proprties*, а потім відредагувати стовпчик *PCB Footprint* (рис. 2.133).

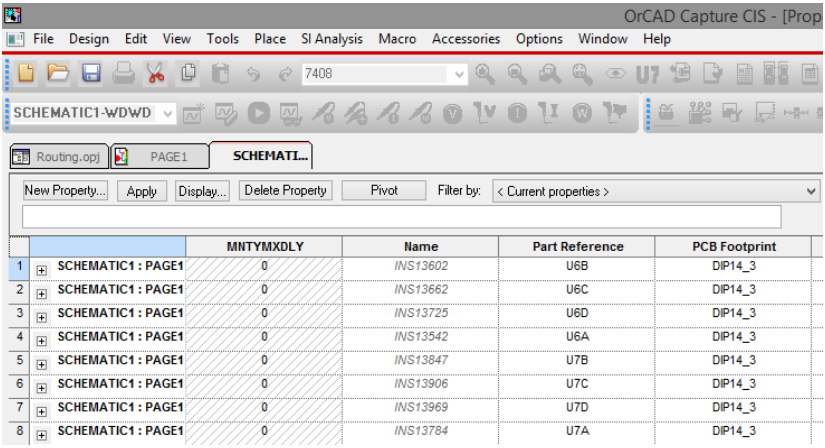


Рисунок 2.133 – Завдання посадкових місць елементів

Елементи одного типу можна редагувати спільно (щоб виділити кілька елементів одночасно, потрібно утримувати клавішу *Ctrl*). При такому редагуванні можна скопіювати в колонці значення одного

елемента, а для решти «розтягнути» межі осередку за куточок. Але такий підхід не застосовується до елементів різного типу, оскільки призводить до конфлікту назв.

Значення редагованої колонки для використовуваних у схемі логічних елементів DIP14_3, а для конекторів – *JumperN*, де *N* – кількість входів конектора. Конектори знаходяться в бібліотеці *CONNECTOR*. Після завдання потрібних властивостей всіх елементів можна переходити до наступного етапу. Тепер операції виконуватимуться вже не з окремими елементами, а зі схемою загалом.

Зберігаємо всі зміни та закриваємо сторінки схеми. Подальші операції проводитимуться у вкладці *File* вікна проєкту зі схемами (*schematic*). Для початку необхідно зробити проєктовану схему кореневою (позначається знаком "/"). Для цього клацніть ПКМ по схемі та виберіть пункт *Make root* (рис. 2.134).

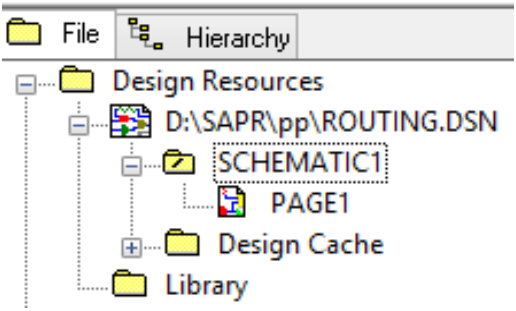


Рисунок 2.134 – Призначення кореневої директорії

Перед тим, як передавати схему до редактора друкованих плат, необхідно сформувати додаткову інформацію про неї за допомогою наступних операцій:

– анотування (проставлення позиційних номерів);

– перевірка правил;

– складання списку з’єднань.

Ці операції здійснюються за допомогою команд випадаючого меню *Tools* (рис.2.135).

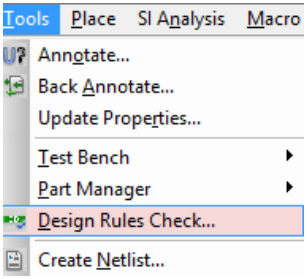


Рисунок 2.135 – Меню *Tools*

Перша операція – проставлення позиційних номерів виконується за допомогою команди *Annotate*. Вона полягає у призначенні унікального номера кожному елементу проєкту. Вікно налаштування параметрів операції наведено на рис. 2.136.

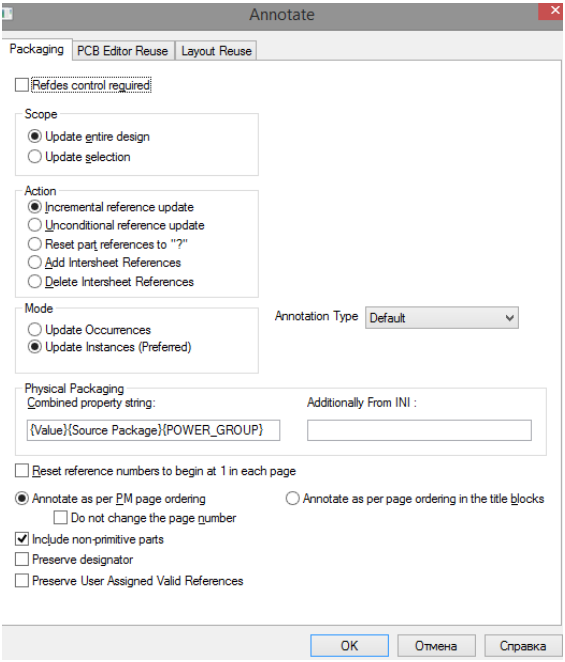


Рисунок 2.136 – Вікно команди *Annotation*

Значення основних параметрів:

Update entire design – пронумерувати елементи для проєкту;

Update selection – пронумерувати елементи для вибраної сторінки;

Incremental reference update – нумерація елементів провадиться у зростаючому порядку;

Unconditional reference update – проглядаються всі номери, визначаються відсутні і далі за зростанням;

Reset part reference to "?" – скинути нумерацію;

Add/Delete Intersheet References – проставити/видалити позиційні позначення номерів сторінок;

Physical Packaging – специфікує властивості об'єднання елементів в один корпус;

Reset reference numbers to begin at 1 inpage schematic – нумерує елементи, починаючи з одиниці в кожний схемний каталог;

Do not change the page number – схемні сторінки не перенумеровуються.

Нумерування проводиться за готовим проєктом. Для цього можна використовувати стандартні параметри. Перед внесенням змін до проєкту рекомендується скинути нумерацію за допомогою вибору дії *Reset part reference to* "?" у вікні анотування.

Перед проведенням операції надається запит на підтвердження нумерування. Для продовження виконання команди натискаємо *OK* (рис. 2.137).

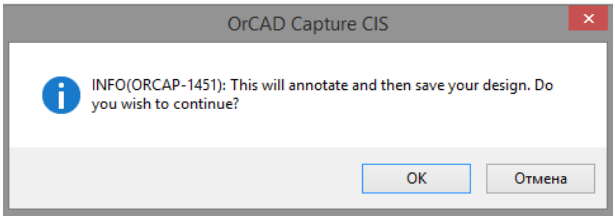


Рисунок 2.137 – Запит на підтвердження інструкції

Після нумерування необхідно перевірити схему на наявність помилок та відповідність правилам. Для цього використовується пункт

меню *Design Rules Check*. Під час його виклику відкривається вікно (рис. 2.138), у якому задаються параметри перевірки.

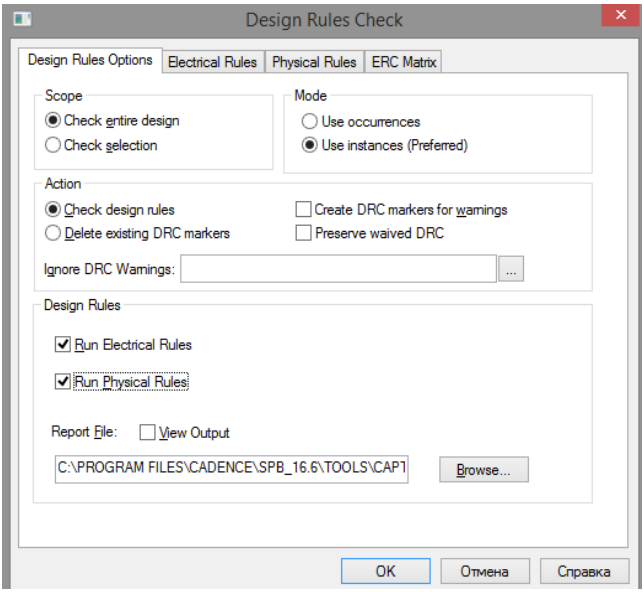


Рисунок 2.138 – Вікно перевірки правил

Перші дві групи опцій у першій вкладці аналогічні попередній команди. У третій групі визначаються спільні дії команди:

Check design rules – перевірка на відповідність електричним правилам;

Delete existing DRC markers – видалення наявних міток;

Create DRC markers for warnings – створення маркерів для попереджень. Програма обов'язково розміщує маркер для помилок, визначених ERC матрицею (якщо ви вибрали цю опцію, вона також створить маркери для попереджень).

Четверта група опцій задає які саме правила перевірятимуть – електричні та/або фізичні. Залежно від вибору цих опцій можна налаштувати параметри наступних двох вкладок, які визначають, як саме перевіряти ці правила (рис. 2.139 – 2.140).

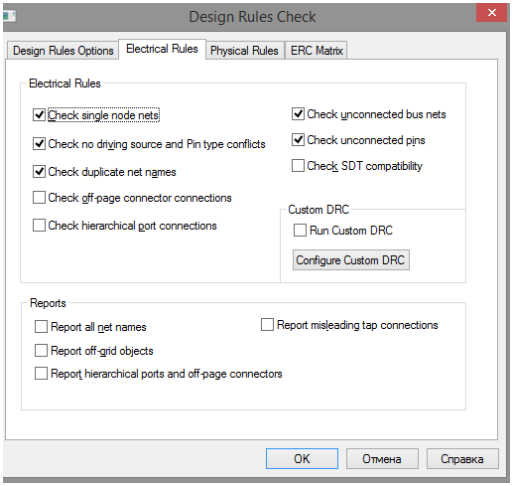


Рисунок 2.139 – Вікно налаштування електричних правил

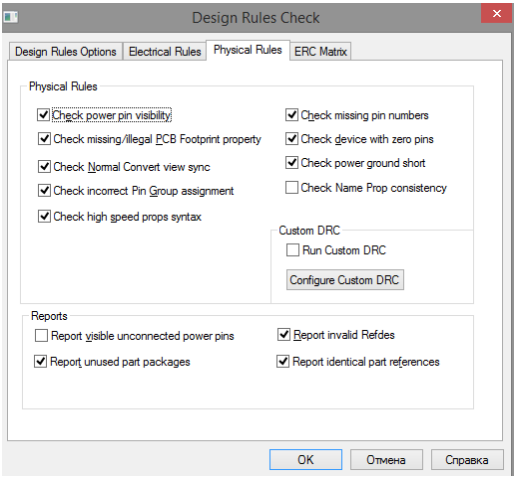


Рисунок 2.140 – Вікно налаштування фізичних правил

Для перевірки проєкту вибирається перевірка електричних та фізичних правил. Для цього відзначимо пункти *Run Electrical Rules* та *Run Phisical Rules*, а інші параметри залишимо за замовчуванням.

За наявності помилок отримуємо повідомлення (рис. 2.141), в якому пропонується вивести помилки в сеансовий журнал (*session log*), який представлений у нижній частині вікна.

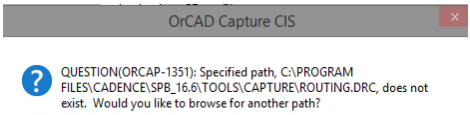


Рисунок 2.141 – Повідомлення про наявність помилок

Остання операція складання списку з'єднань – має найбільш безпосереднє відношення до трасування елементів на друкованій платі. Ця команда складає список, що містить опис з'єднань між частинами схеми. Вона викликається через пункт *Create Netlist* меню *Tools*.

Вікно, що відкривається по цій команді, містить різні вкладки, залежно від того, для чого призначається цей список. Оскільки працюємо з PCB Editor, вибираємо першу вкладку (рис. 2.142).

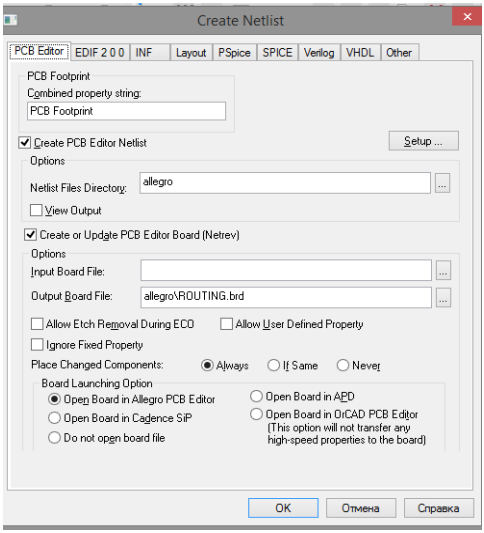


Рисунок 2.142 – Вікно створення списку з'єднань

Тут у другій групі можна визначити директорію, до якої буде розміщено список (*Netlist Files Directory*). Потім необхідно поставити позначку в пункті *Create or Update PCB Editor Board*, щоб створити плату. Після цього задаються її параметри:

Input Board File – файл, який буде взятий за основу (не обов'язковий параметр);

Output Boaed File – файл, в який буде поміщена створена плата.

У групі *Board Launching Option* наводяться варіанти програм, за допомогою яких плата буде відкрита. Оскільки працюємо з PCB Editor, вибираємо пункт *Open Board in Allegro PCB Editor*.

Після заповнення параметрів (у нашому випадку *Create or Update PCB Editor Board*) натискаємо *OK* для створення списку з'єднань друкованої плати та відкриття її в редакторі. Відкриється вікно, яке демонструє процес створення списку та оновлення плати (рис. 2.143). У разі помилок буде виведено відповідне повідомлення, а самі помилки наведені у журналі сеансу. Якщо все пройде успішно, запуститься PCB Editor.

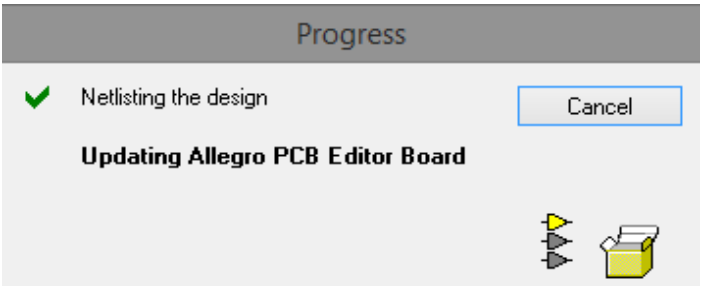


Рисунок 2.143 – Процес створення списку з'єднань та плати

Тепер переходимо до третього етапу створення друкованої плати. Після успішного створення списку з'єднань відбувається автоматичний запуск PCB Editor і з'явиться вікно з варіантами, який саме продукт використовуватиметься (рис. 2.144). У нашому випадку підходить Allegro PCB SI XL, натискаємо *OK*.

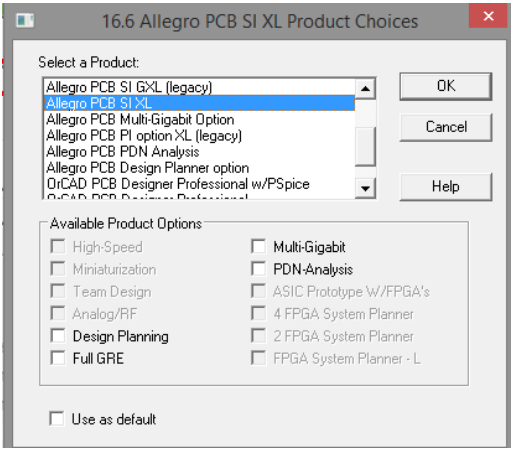


Рисунок 2.144 – Вибір продукту

Щоб змінити колірну схему в PCB Editor, потрібно вибрати пункт меню *Display –> Color/Visibility*. Колір фону можна зробити білим, як показано на рис. 2.145. Для більшої наочності, щоб змінити колір плати у вікні *Color Dialog* у вкладці *Conductor*, можна змінити кольори ліній трасування та точок.

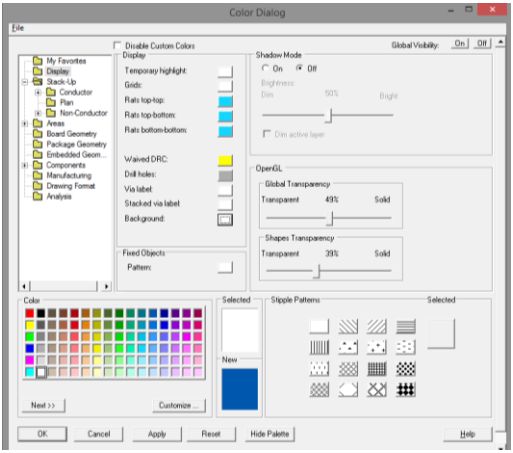


Рисунок 2.145 – Вікно зміни колірної схеми

Після вибору продукту з’явиться вікно PCB Editor, у якому потрібно виконувати фінальну стадію створення друкованої плати. Мають бути виконати такі етапи: завдання контурів плати, розміщення компонентів і трасування провідників. У пункті меню *Setup\Design Parameters* виберемо необхідні параметри в блоці *Size* (рис. 2.146).

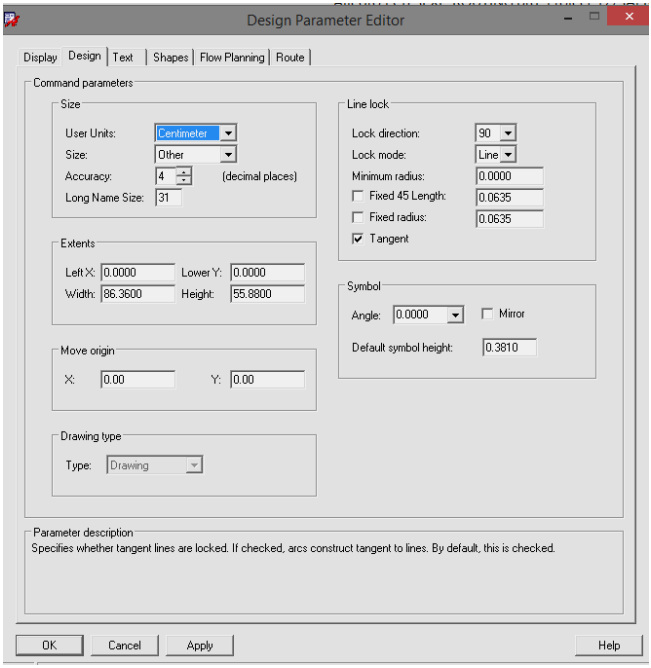


Рисунок 2.146 – Вибір параметрів плати

Перед тим, як розміщувати компоненти, треба створити плату, на якій їх можна розмістити, для цього вибираємо пункт меню *Setup*, підменю *Outlines*, пункт *Board Outline*.

Вікно, що відкрилося (рис. 2.147), супроводжує процес нанесення меж плати – клацніть ЛКМ на робочому просторі два рази для завдання верхнього лівого та нижнього правого кутів. Після завершення креслення необхідно вибрати *Close* у вікні *Board Outline***.** Отримаємо межі друкованої плати, з якою можна тепер працювати (рис. 2.148).

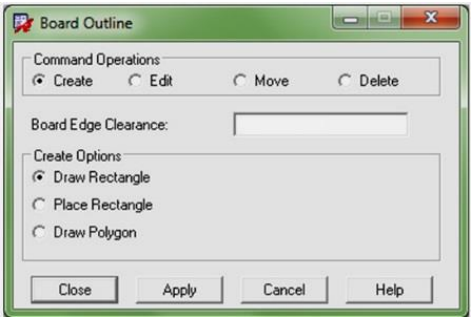


Рисунок 2.147 – Вікно, що супроводжує нанесення меж плати

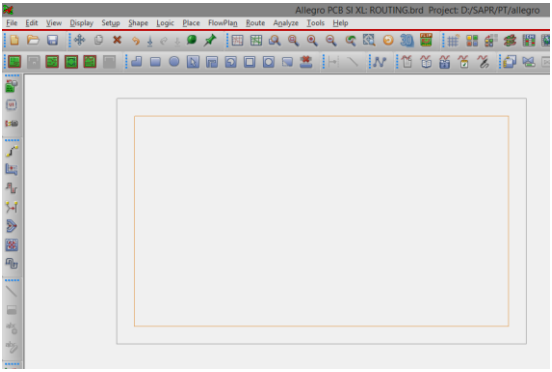


Рисунок 2.148 – Отримані межі плати

Є два способи розміщення елементів на платі: автоматичний і ручний – вони знаходяться у пункті меню *Place* (рис. 2.149).

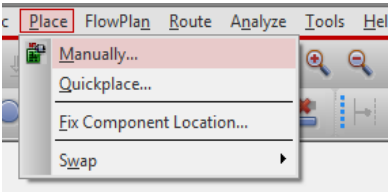


Рисунок 2.149 – Меню *Place*

Для автоматичного розміщення вибираємо пункт *Quickplace*. Відкриється відповідне вікно (рис. 2.150), у якому можна задати параметри розміщення.

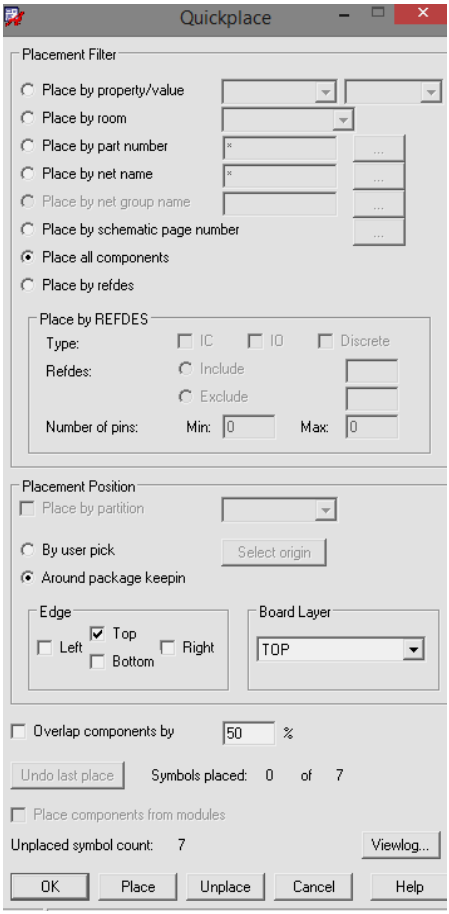


Рисунок 2.150 – Вікно *QuickPlace*

Перша група визначає, які елементи треба помістити та пропонує набір фільтрів. Для розміщення всіх компонентів вибираємо пункт *Place all components*. Друга група визначає де будуть розміщені елементи. Тут

задаються шар розміщення та межі. Після завдання параметрів необхідно натиснути кнопку *Place*, щоб скасувати нанесення *Unplace*. Якщо результат задовольнив нас – натискаємо *OK* (рис. 2.151).

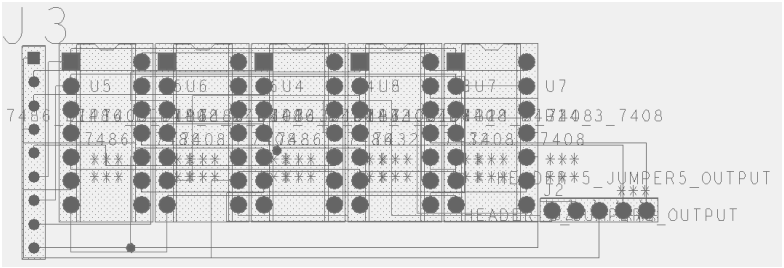


Рисунок 2.151 – Розміщені елементи

Після розміщення компонентів початковий розмір плати виявляється не оптимальним. Для редагування її меж користуємося режимом редагування *Edit* (рис. 2.152).

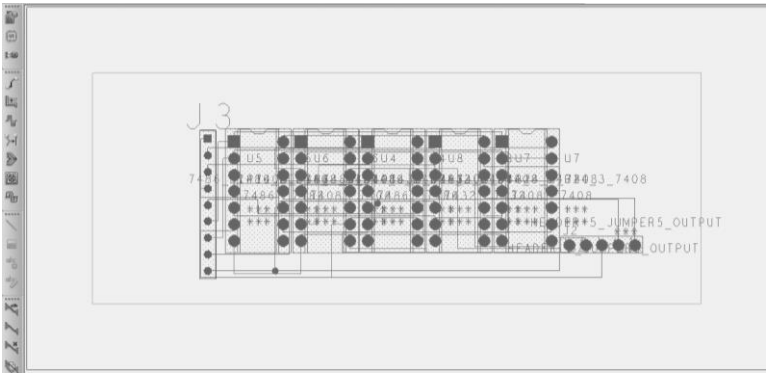


Рисунок 2.152 – Розміщення елементів після редагування меж плати

Для ручного розміщення елементів необхідно вибрати пункт меню *Manually*. Відкриється вікно *Placement* (рис. 2.153), в якому зліва буде список елементів пристрою.

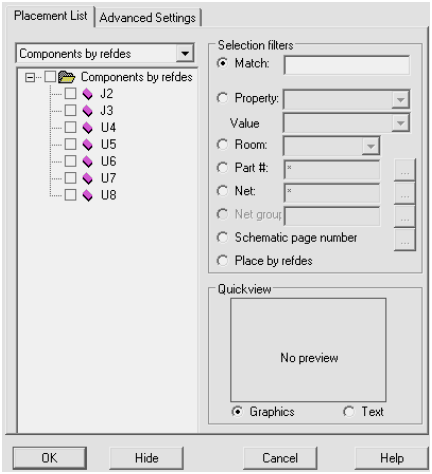


Рисунок 2.153 – Вікно ручного розміщення

Для розміщення елемента необхідно поставити позначку поряд з його ім’ям та розмістити на платі кліком ЛКМ. Після завершення розміщення необхідно натиснути *OK*.

При розміщенні елемента *J3* з’явиться вікно *Create Temporary Package*. Треба натиснути *OK* і розмістити елемент *J3*. Результат ручного розміщення наведено на рис. 2.154. Цей результат зрозуміліший, але менш оптимальний.

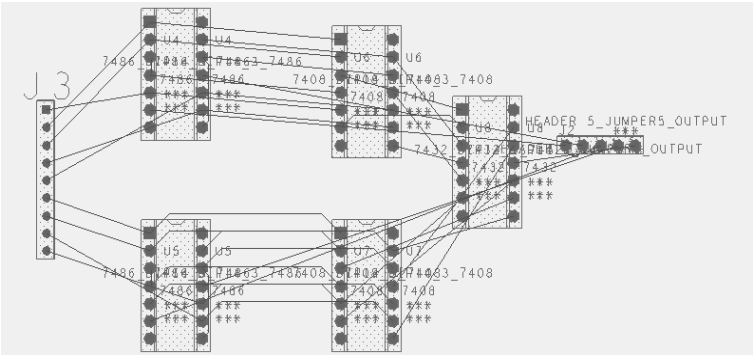


Рисунок 2.154 – Розміщення елементів

Останній етап проєктування друкованих плат – це трасування провідників. Усі команди трасування розміщені у пункті меню *Route* (рис. 2.155).

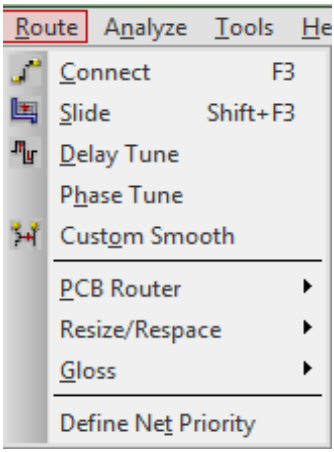


Рисунок 2.155 – Меню *Route*

Для ручного розміщення елементів призначено першу групу команд меню. Щоб скоригувати, потрібно вибрати ЛКМ зв'язок, потім вибрати пункт меню з відповідною командою і намітити шлях.

Після закінчення клацнути ПКМ і вибрати *Done*:

– команда *Connect* дозволяє з'єднати точки, задаючи траєкторію;

– *Slide* зрушує об'єкти, у тому числі прокладені зв'язки;

– *Delay Tune* додає на лінії згини, що додають затримку;

– *Phase Tune* додає згини, які викликають зсув по фазі;

– *Custom Smooth* згладжує лінії.

Розміщення провідників – складне математичне завдання, яке у користувача забирає багато часу та сил, не гарантуючи оптимальності при цьому.

Автоматичне прокладення здійснюється через пункт меню *Route*, підменю *PCB Router*, пункт *Route Automatic*. Після вибору цієї команди відкриється вікно автоматичного трасування (рис. 2.156), у якому можна специфікувати деякі опції алгоритму прокладання провідників. Після

цього необхідно натиснути кнопку *Route* для здійснення трасування. Також можна натиснути *Undo* для скасування трасування та *Close* для закриття вікна. Результат автоматичного трасування наведено на рис. 2.157.

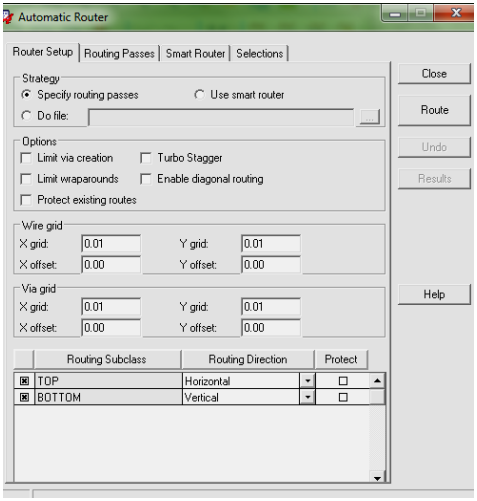


Рисунок 2.156 – Вікно автоматичного трасування

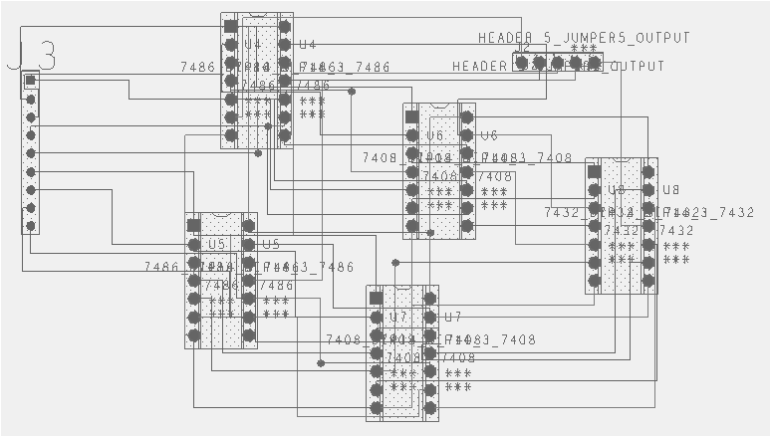


Рисунок 2.157 – Результат трасування схеми

Ручне розміщення мікросхем та трасування корисні, але забирають більше часу, воно не оптимальне та може призводити до помилок. Результат повністю автоматичного трасування наведено на рис. 2.158.

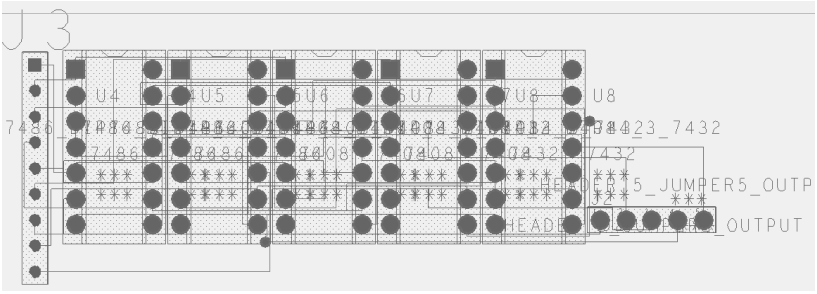


Рисунок 2.158 – Результат після автотрасування

Завдання для виконання практичної роботи

Виконати трасування друкованої плати для розробленого в практичній роботі № 1 пристрою в САПР OrCAD.

Практична робота 8
Дослідження роботи пристроїв з урахуванням ЕМС

Мета роботи: виконати дослідження пристрою для оцінки його роботи з урахуванням ЕМС.

Приклад виконання завдання

В якості пристрою для дослідження обираємо шифратор 12 –> 1 (рис. 2.159), результати моделювання якого наведені на рис. 2.160.

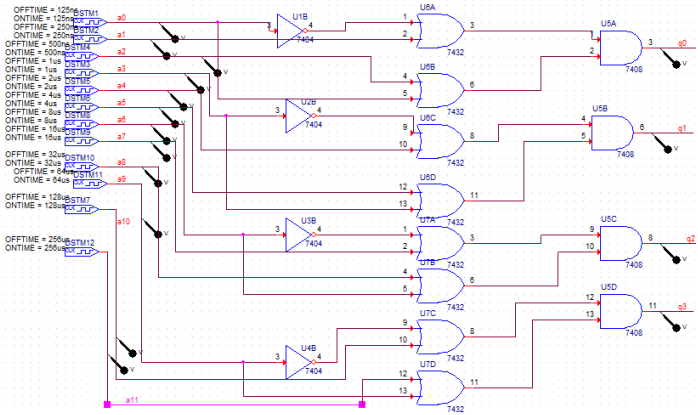


Рисунок 2.159 – Схема шифратора

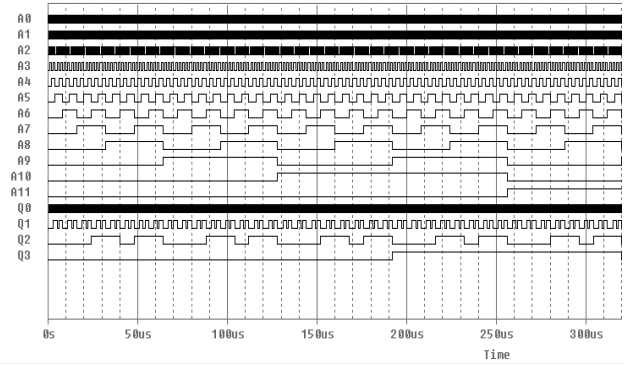


Рисунок 2.160 – Результати моделювання

Виконаємо заміну генераторів на вході схеми на 12-розрядний конектор, а на виході поставимо 4-розрядний конектор (рис. 2.161).

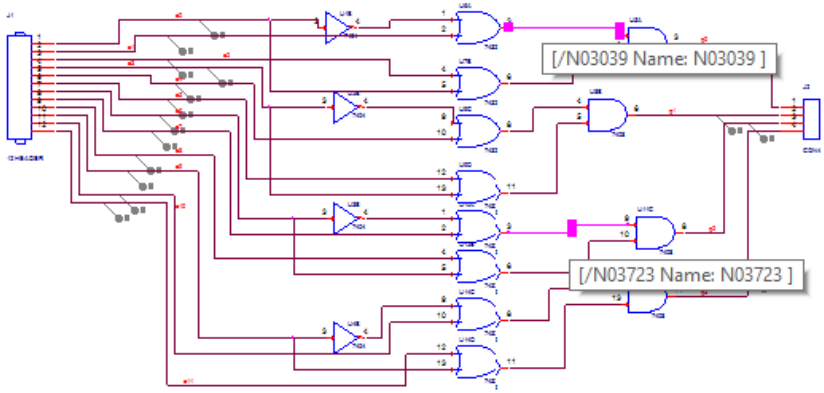


Рисунок 2.161 – Схема шифратора з конекторами

Встановлюємо значення параметру PCB Footprint для всіх елементів схеми, як показано на рис. 2.162 (логічним елементам присвоюємо тип DIP14_3, конекторам – Jumper, інші параметри є стандартними).

encoder_t...* | PAGE1* | encoder_r...*

New Property... | Apply | Display... | Delete Property | Pivot | Filter by: < Current properties > | Help

		PCB Footprint	Implementation	Designator	Color	Part Reference
1	encoder_routing : PAGE1 : J1	Jumper4			Default	J1
2	encoder_routing : PAGE1 : J3	Jumper12			Default	J3
3	encoder_routing : PAGE1 : U32	DIP14_3	7408	C	Default	U32C
4	encoder_routing : PAGE1 : U32	DIP14_3	7408	D	Default	U32D
5	encoder_routing : PAGE1 : U32	DIP14_3	7408	A	Default	U32A
6	encoder_routing : PAGE1 : U32	DIP14_3	7408	B	Default	U32B
7	encoder_routing : PAGE1 : U33	DIP14_3	7404	C	Default	U33C
8	encoder_routing : PAGE1 : U33	DIP14_3	7404	D	Default	U33D
9	encoder_routing : PAGE1 : U33	DIP14_3	7404	B	Default	U33B
10	encoder_routing : PAGE1 : U33	DIP14_3	7404	A	Default	U33A
11	encoder_routing : PAGE1 : U34	DIP14_3	7432	B	Default	U34B
12	encoder_routing : PAGE1 : U34	DIP14_3	7432	C	Default	U34C
13	encoder_routing : PAGE1 : U34	DIP14_3	7432	D	Default	U34D
14	encoder_routing : PAGE1 : U34	DIP14_3	7432	A	Default	U34A
15	encoder_routing : PAGE1 : U35	DIP14_3	7432	C	Default	U35C
16	encoder_routing : PAGE1 : U35	DIP14_3	7432	D	Default	U35D
17	encoder_routing : PAGE1 : U35	DIP14_3	7432	B	Default	U35B
18	encoder_routing : PAGE1 : U35	DIP14_3	7432	A	Default	U35A

Рисунок 2.162 – Встановлення значення параметру PCB Footprint

В ході виконання анотації та налаштування системи анотування програма автоматично призначає усім провідникам назви, які використовуються в подальшому для трасування, результатом чого є успішне завершення операції (рис. 2.163).

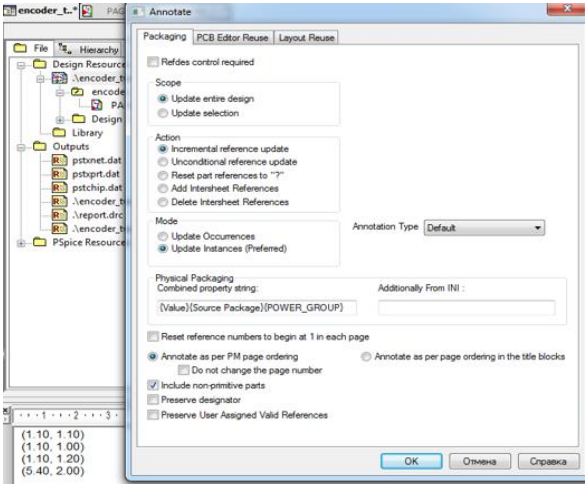


Рисунок 2.163 – Генерування анотації для схеми пристрою

Для виконання перевірки електричних і фізичних правил задаються параметри процедури перевірки (рис. 2.164).

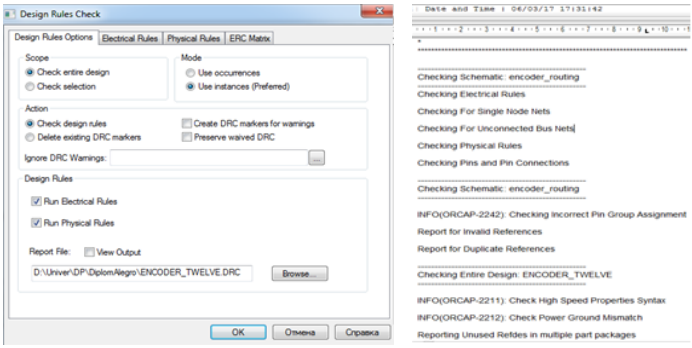


Рисунок 2.164 – Перевірка правил

Тепер виконуємо побудову списку провідників і з’єднань (рис. 2.165), після чого з’явиться можливість обрати програму Allegro PCB SI XL для подальшої роботи.

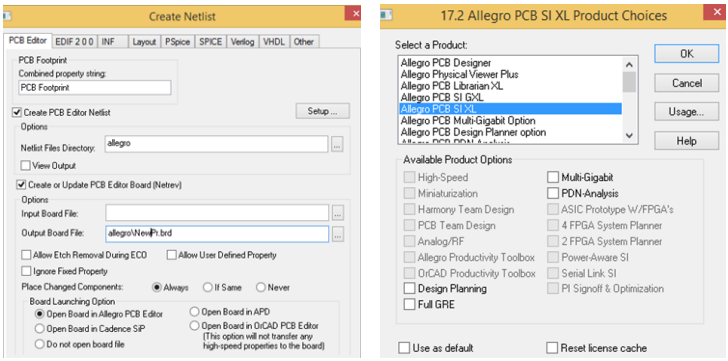


Рисунок 2.165 – Складання списку з’єднань проєкту та вибір програми

Для розміщення елементів на друкованій платі обираємо розділ інтерфейсу *Place –> Manualy*. Після цього з’являється вікно, в якому почергово обираємо кожний елемент та встановлюємо його на платі (рис. 2.166).

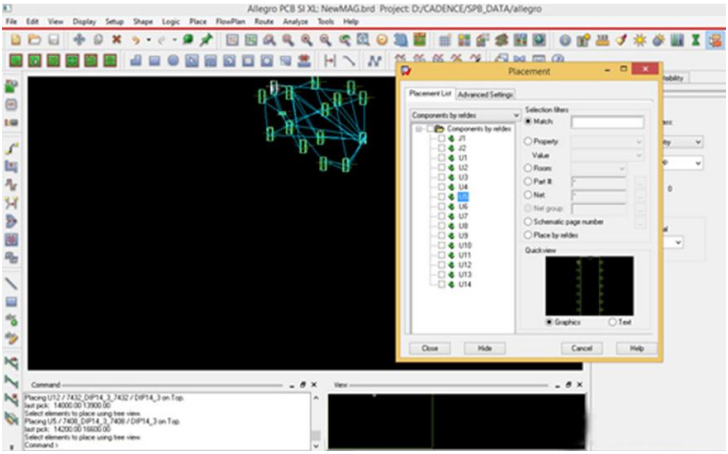


Рисунок 2.166 – Розміщення елементів на платі

Тепер задаємо товщину провідників 1,2 мілідюйма, матеріал – мідь. В якості матеріалу діелектрика обираємо склотекстоліт (рис. 2.167).

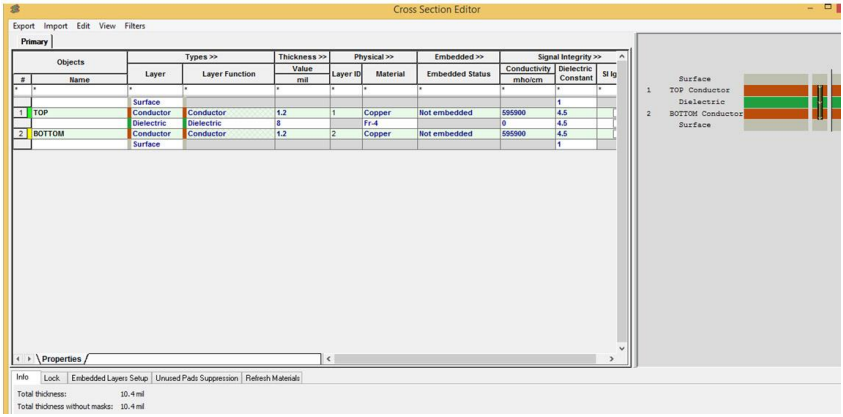


Рисунок 2.167 – Задання товщини та типу матеріалу

Тепер встановлюємо мінімально допустиму відстань між провідниками (рис 2.168).

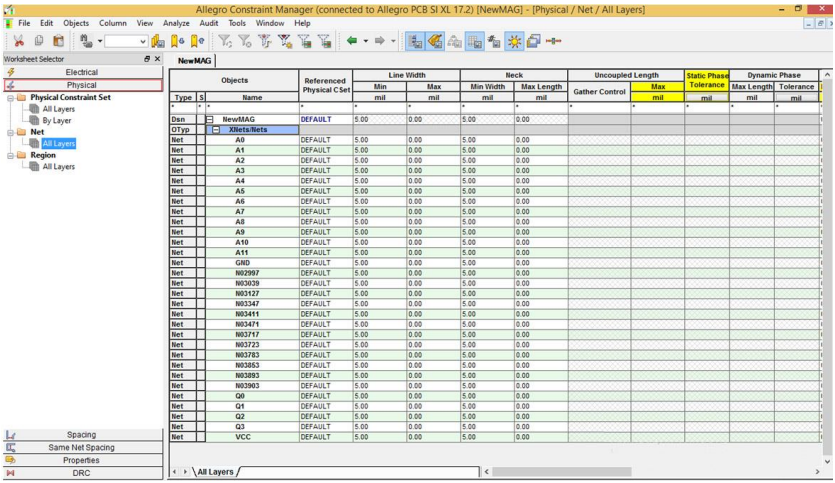


Рисунок 2.168 – Задання мінімальної відстані між провідниками

Виконуємо автоматичне трасування за допомогою команди *Route –> Route Automatic*. Процес виконання трасування показано на рис. 2.169, а його результат – на рис. 2.170.

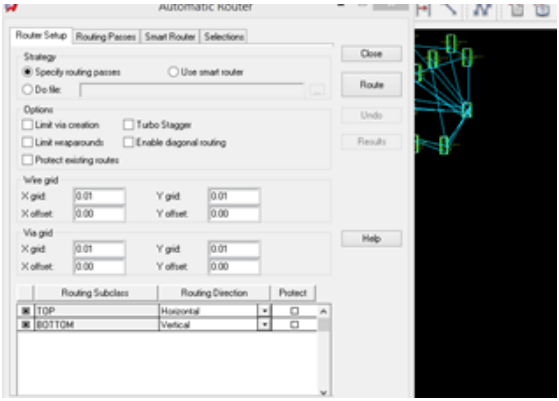


Рисунок 2.169 – Виконання трасування

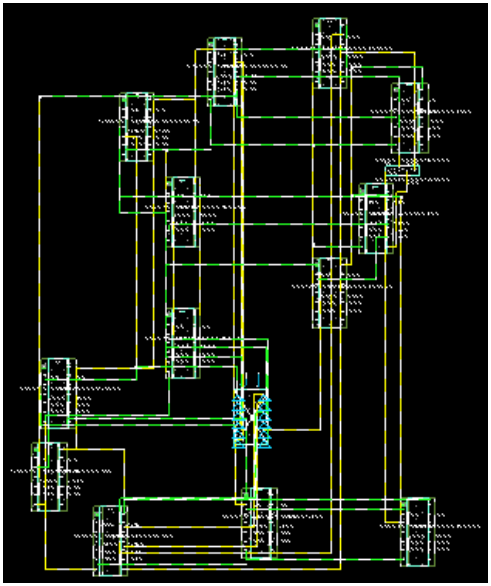


Рисунок 2.170 – Результат трасування

Вказуємо необхідні значення напруги для VCC і GND (*Logic –> Identify DC Nets*) – встановлюємо напруги 5 V і 0 V (рис. 2.171).

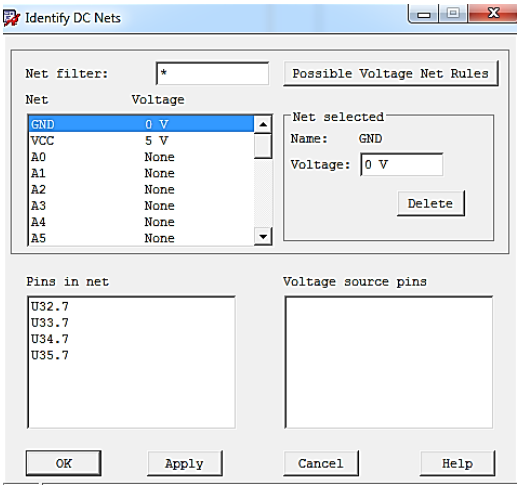


Рисунок 2.171 – Встановлення напруг

Для аналізу цілісності сигналів і перехресних завад в цифрових друкованих платах використовуються IBIS-моделі. Формат IBIS – це формат зовнішнього опису електронного пристрою без урахування його внутрішньої структури і особливостей функціонування. Параметри IBIS-моделей отримують на основі вольт-амперної характеристики для різних логічних станів виводів по постійному струму, паразитних параметрів корпусу і перехідних характеристик на резистивному навантаженні.

Створюємо IBIS-модель для кожного елементу схеми: *Analyze –> Model Assignment* (рис. 2.172).

Після виконання налаштувань та проведення трасування, було обрано провідники, що будуть виступати у ролі *slave* та *master*.

Провідник-*master* – це провідник, який є джерелом сигналу, в даному дослідженні це провідник N03723 (рис. 2.173). Провідник-*slave* – це провідник, який є приймачем сигналу (в даному дослідженні це провідник N03039).

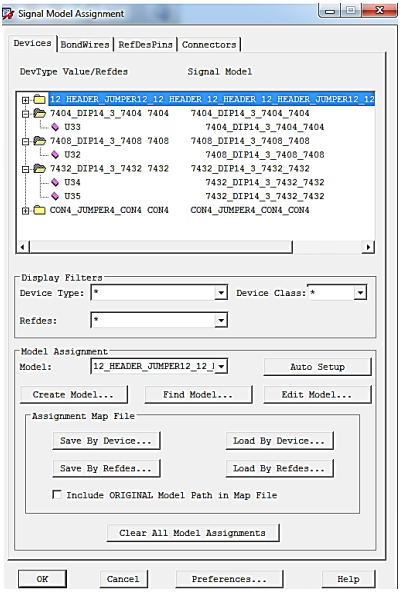


Рисунок 2.172 – Створення IBIS моделей для елементів плати

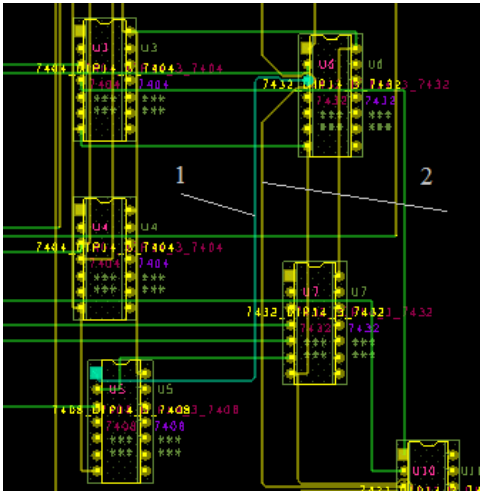


Рисунок 2.173 – Вибір провідників для дослідження
(1 – провідник-*slave*, 2 – провідник-*master*)

Дослідження роботи пристрою при різній довжині провідників

Виконаємо дослідження зміни значення напруги перехресної завади між двома паралельними провідниками за умови зміни довжини їх прилеглих частин.

Спочатку виконаємо заміри провідників та встановимо їх довжини (4344 мілідюймів). Виконуємо побудову звіту перехресних завад для провідника-*slave* для заданої довжини за допомогою розділу інтерфейсу *Analys –> Probe*, для цього у вікні обираємо провідник з номером, що відповідає номеру провідника-*slave* (N03039), та обираємо пункт *Creat report*.

У новому вікні ставимо позначки: *Crosstalk Detailed*; швидкість симуляції *Slow*, *Highlighted Net Only* (виділений в даний момент провідник); *Each Neighbor* – сусідній провідник є *master*; *Pulse* – подача імпульсних псевдовипадкових сигналів на вхід провідника *master* (рис. 2.174).

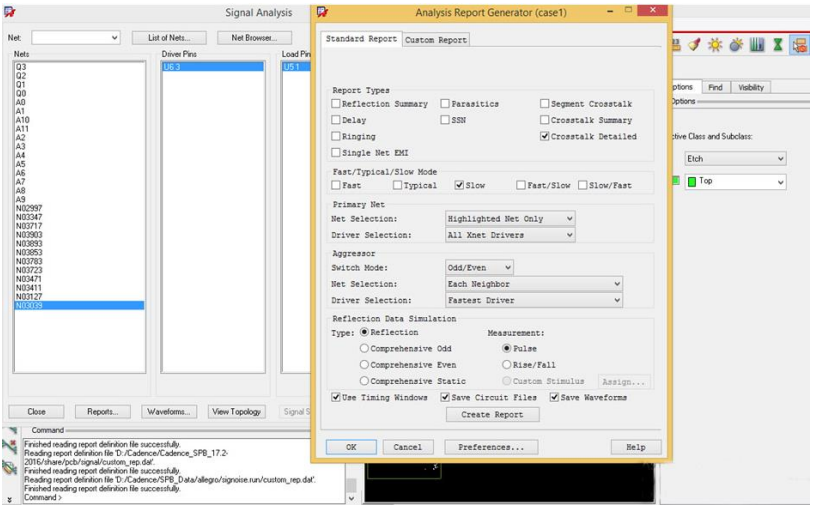


Рисунок 2.174 – Вибір параметрів для створення звіту перехресних завад

В отриманому звіті значення напруги перехресних завад знаходиться в розділі *Single Neighbor Crosstalk*. В результаті отримані значення напруги завади (рис. 2.175) заносимо до табл. 2.8.

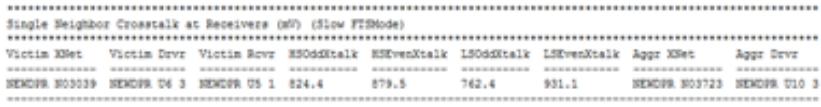


Рисунок 2.175 – Значення напруг перехресних завад при довжині прилеглої частини провідників 4344 мілідюймів

Таблиця 2.8 – Змодельовані ситуації перехресних завад

Перехресна завада	Умови виникнення перехресної завади		Значення, мВ
	Постійний логічний рівень *slave*	Перехід *master*	
HSOddXtalk	HI	HI – LO	824,4
HSEvenXtalk	HI	LO – HI	879,5
LSOddXtalk	LO	LO – HI	726,4
LSEvenXtalk	LO	HI – LO	931,1

Максимальне значення напруги у цьому звіті є LSEvenXtalk, що дорівнює 931,1 мВ, коли логічний рівень сигналу *slave* нуль (0 В), а *master* перемикається з одиниці в нуль (з 5 В до 0 В).

Далі виконуємо аналогічні дії за умови скорочення довжини прилеглої частини провідників до 2020, 1620, 1050, 630, 320, 300 мілідюймів. Звіти до цих досліджень наведені на рис. 2.176 – 2.181.

Значення напруг перехресних завад, які були отримані для довжини прилеглої частини провідників 2020 мілідюймів, наведені в табл. 2.9.

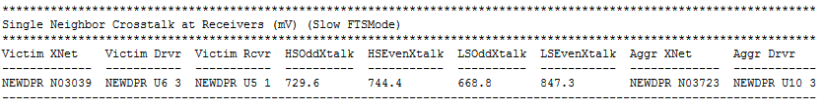


Рисунок 2.176 – Значення напруг перехресних завад при довжині прилеглої частини провідників 2020 мілідюймів

Максимальне значення напруги у цьому звіті є LSEvenXtalk (847,3 мВ), коли логічний рівень сигналу *slave* нуль (0 В), а *master* перемикається з одиниці в нуль (з 5 В до 0 В).

Таблиця 2.9 – Змодельовані ситуації перехресних завад

Перехресна завада	Умови виникнення перехресної завади		Значення, мВ
	Постійний логічний рівень *slave*	Перехід *master*	
HSOddXtalk	HI	HI – LO	729,6
HSEvenXtalk	HI	LO – HI	744,4
LSOddXtalk	LO	LO – HI	668,8
LSEvenXtalk	LO	HI – LO	847,3

```
*****************************************************************************************************************
Neighbor Crosstalk at Receivers (mV) (Fast/Slow FTSMode)
*****************************************************************************************************************
XNet    Victim Drvr  Victim Rcvr  HSOddXtalk  HSEvenXtalk  LSOddXtalk  LSEvenXtalk  Aggr XNet      Aggr Drvr
------  -----------  -----------  ----------  -----------  ----------  -----------  -------------  ------------
N03039  NEWDPR U6 3  NEWDPR U5 1  414.4       309.5        360.3       362.9        NEWDPR N03723  NEWDPR U10 3
-----------------------------------------------------------------------------------------------------------------
```

Рисунок 2.177 – Значення напруг перехресних завад при довжині прилеглої частини провідників 1620 мілідюймів

Значення напруг перехресних завад при довжині прилеглої частини провідників 1620 мілідюймів наведені в табл. 2.10.

Таблиця 2.10 – Змодельовані ситуації перехресних завад

Перехресна завада	Умови виникнення перехресної завади		Значення, мВ
	Постійний логічний рівень *slave*	Перехід *master*	
HSOddXtalk	HI	HI – LO	414,4
HSEvenXtalk	HI	LO – HI	309,5
LSOddXtalk	LO	LO – HI	360,3
LSEvenXtalk	LO	HI – LO	362,9

Максимальне значення напруги у цьому звіті є HSOddXtalk (414.4 мВ), коли логічний рівень сигналу *slave* одиниця (5 В), а *master* перемикається з одиниці в нуль (з 5 В до 0 В).

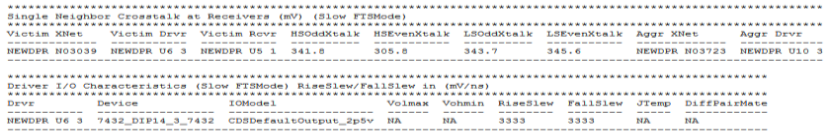


Рисунок 2.178 – Значення напруг перехресних завад при довжині прилеглої частини провідників 1050 мілідюймів

Значення напруг перехресних завад при довжини прилеглої частини провідників 1050 мілідюймів наведені в табл. 2.11.

Таблиця 2.11 – Змодельовані ситуації перехресних завад

Перехресна завада	Умови виникнення перехресної завади		Зна-чення, мВ
	Постійний логічний рівень *slave*	Перехід *master*	
HSOddXtalk	HI	HI – LO	341,8
HSEvenXtalk	HI	LO – HI	305,8
LSOddXtalk	LO	LO – HI	343,7
LSEvenXtalk	LO	HI – LO	345,6

Максимальне значення напруги у цьому звіті є LSEvenXtalk (345,6 мВ), коли логічний рівень сигналу *slave* нуль (0 В), а *master* перемикається з одиниці в нуль (з 5 В до 0 В).

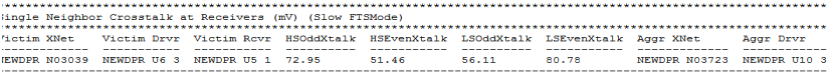


Рисунок 2.179 – Значення напруг перехресних завад при довжині прилеглої частини провідників 630 мілідюймів

Значення напруг перехресних завад при довжини прилеглої частини провідників 630 мілідюймів наведені в табл. 2.12.

Таблиця 2.12 – Змодельовані ситуації перехресних завад

Перехресна завада	Умови виникнення перехресної завади		Значення, мВ
	Постійний логічний рівень *slave*	Перехід *master*	
HSOddXtalk	HI	HI – LO	72,95
HSEvenXtalk	HI	LO – HI	51,46
LSOddXtalk	LO	LO – HI	56,11
LSEvenXtalk	LO	HI – LO	80,78

Максимальне значення напруги у цьому звіті є LSEvenXtalk (80,78 мВ), коли логічний рівень сигналу *slave* нуль (0 В), а *master* перемикається з одиниці в нуль (з 5 В до 0 В).

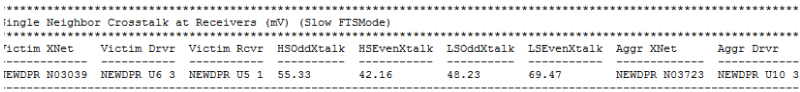


Рисунок 2.180 – Значення напруг перехресних завад при довжині прилеглої частини провідників 320 мілідюймів

Значення напруг перехресних завад при довжині прилеглої частини провідників 320 мілідюймів наведені в табл. 2.13.

Таблиця 2.13 – Змодельовані ситуації перехресних завад

Перехресна завада	Умови виникнення перехресної завади		Значення, мВ
	Постійний логічний рівень *slave*	Перехід *master*	
HSOddXtalk	HI	HI – LO	55,33
HSEvenXtalk	HI	LO – HI	42,16
LSOddXtalk	LO	LO – HI	48,23
LSEvenXtalk	LO	HI – LO	69,47

Максимальне значення напруги у цьому звіті є LSEvenXtalk (69,47 мВ), коли логічний рівень сигналу *slave* нуль (0 В), а *master* перемикається з одиниці в нуль (з 5 В до 0 В).

```
*****************************************************************************************************************
Single Neighbor Crosstalk at Receivers (mV) (Slow FTSMode)
*****************************************************************************************************************
Victim XNet  Victim Drvr  Victim Rcvr  HSOddXtalk  HSEvenXtalk  LSOddXtalk  LSEvenXtalk  Aggr XNet  Aggr Drvr
-----------  -----------  -----------  ----------  -----------  ----------  -----------  ---------  ---------
-----------------------------------------------------------------------------------------------------------------
```

Рисунок 2.181 – Значення напруг перехресних завад при довжині прилеглої частини провідників меншої 300 мілідюймів

При зменшенні довжини прилеглої частини провідників до значення, яке менше 300 мілідюймів, можна побачити відсутність напруг перехресних завад. Після виконаних досліджень можна зробити висновок, що із зменшенням довжини прилеглої частини провідників зменшується значення напруги перехресних завад. А при встановленні довжини, що не перевищує 300 мілідюймів, відповідне поле звіту перехресних завад було порожнє, тому можна сказати, що перехресні завади не виникають або не впливають на функціонування пристрою.

На рис. 2.182 наведено графік напруг перехресних завад при довжині прилеглої частини провідників 4344 мілідюймів. На цьому графіку зображено імпульсний сигнал, який подається на провідник-*master* N3723 (U11 9) та графік сигналу на провіднику-*slave* N03039 (U5 1), амплітуда завади в цьому випадку досягає 1 В.

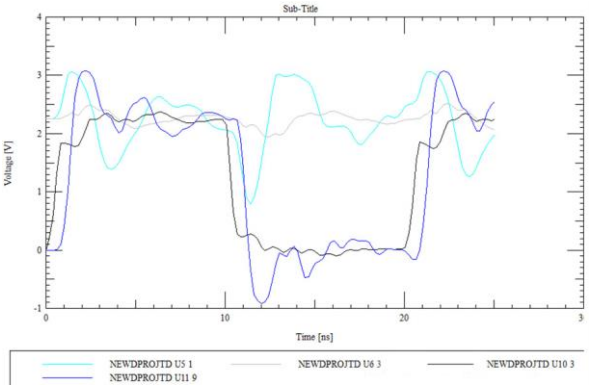


Рисунок 2.182 – Графік напруг перехресних завад при довжині прилеглої частини провідників 4344 мілідюймів

На рис. 2.183 наведено графік напруг перехресних завад при зменшенні довжини прилеглої частини провідників до 2020 мілідюймів.

В цьому випадку значення амплітуди перехресної завади зменшилось до 750 мВ.

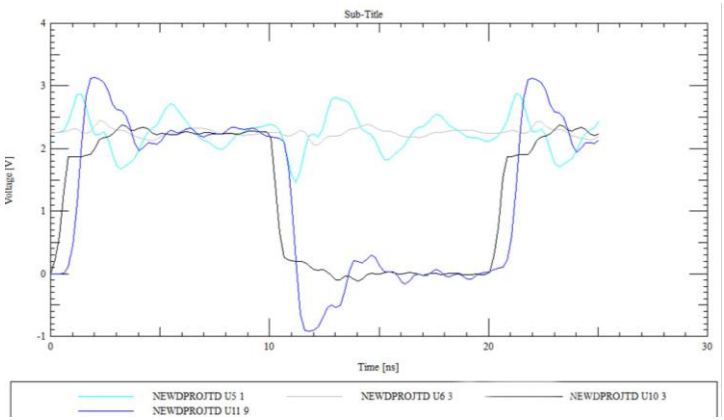


Рисунок 2.183 – Графік напруг перехресних завад при довжині прилеглої частини провідників 2020 мілідюймів

На рис. 2.184 наведено графік напруг перехресних завад при довжині прилеглої частини провідників 1620 мілідюймів. На провідник-*master* (N03723) подається імпульсний псевдовипадковий сигнал (U11 9). В цьому випадку отримуємо спотворення сигналу на провіднику-*slave* (U5 1). Значення амплітуди перехресної завади зменшилося до 550 мВ.

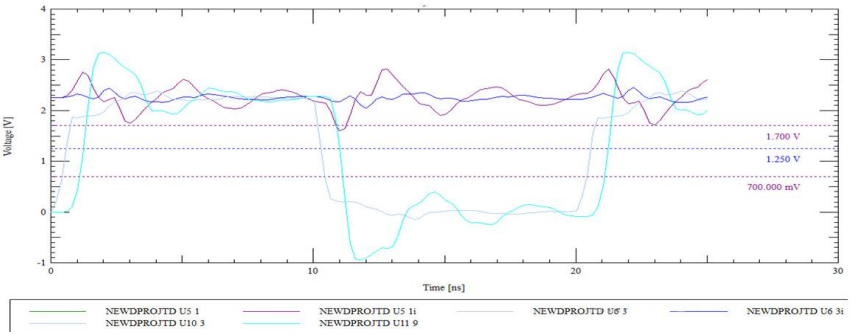


Рисунок 2.184 – Графік напруг перехресних завад при довжині прилеглої частини провідників 1620 мілідюймів

На рис. 2.185 наведено графік напруг перехресних завад при довжині прилеглої частини провідників 360 мілідюймів. На провідник-*master* N03723 подається імпульсний сигнал (U11 9). Сигнал на провіднику-*slave* N03039 (U5 1) має спотворену форму, амплітуда завади зменшилась до 100 мВ. Частота імпульсного сигналу, що подавався у всіх випадках, становить $f = 50$ МГц, а період $T = 20$ нс.

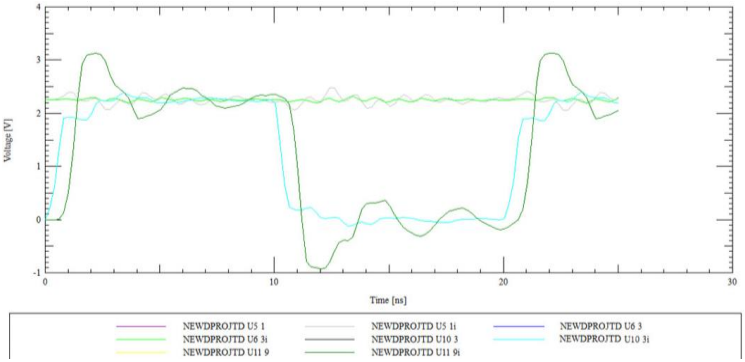


Рисунок 2.185 – Графік напруг перехресних завад при довжині прилеглої частини провідників 360 мілідюймів

Дослідження роботи пристрою при різній товщині провідників

Дослідження виконуємо при довжині прилеглої частини провідників 1460 мілідюймів. Побудова звітів перехресних завад виконується аналогічно попереднім дослідженням.

На рис. 2.186 наведені значення напруг перехресних завад при товщині провідників 1,2 мілідюйми (матеріал – срібло).

```
**********************************************************************************************************************************
Single Neighbor Crosstalk at Receivers (mV) (Slow FTSMode)
**********************************************************************************************************************************
Victim XNet        Victim Drvr      Victim Rcvr      HSOddXtalk  HSEvenXtalk  LSOddXtalk  LSEvenXtalk  Aggr XNet          Aggr Drvr
-----------------  ---------------  ---------------  ----------  -----------  ----------  -----------  -----------------  ----------------
NEWDPROJTD N03039  NEWDPROJTD U6 3  NEWDPROJTD U5 1  410.9       440.3        355.8       496.5        NEWDPROJTD N03723  NEWDPROJTD U10 3
----------------------------------------------------------------------------------------------------------------------------------

**********************************************************************************************************
Driver I/O Characteristics (Slow FTSMode) RiseSlew/FallSlew in (mV/ns)
```

Рисунок 2.186 – Значення напруг перехресних завад при товщині провідників 1,2 мілідюйми

Згідно з рис. 2.186, максимальне значення напруги перехресної завади становить 496,5 мВ, у LSEvenXtalk.

Наступним кроком було збільшення товщини провідників до 5 мілідюймів у вікні *Setup –> Cross-Section* та створення звіту перехресних завад. В цьому випадку максимальне значення напруг перехресних завад становить 707,9 мВ при HSOddXtalk (рис. 2.187).

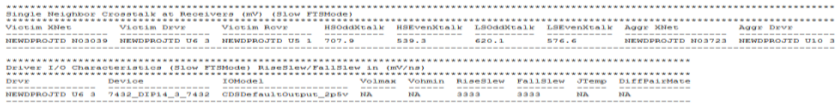


Рисунок 2.187 – Значення напруг перехресних завад при товщині провідників 5 мілідюймів

Далі було аналогічно змінено товщину до 6 мілідюймів та отримані значення перехресних завад (рис. 2.188). Максимальне значення напруги в цьому випадку становить 770.8мВ, при HSOddXtalk.

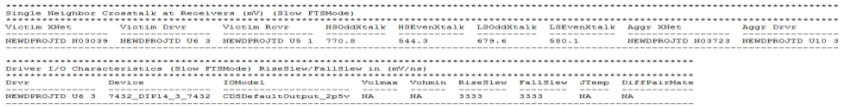


Рисунок 2.188 – Значення напруг перехресних завад при товщині провідників 6 мілідюймів

Результати досліджень наведені в табл. 2.14, з яких можна зробити висновок, що при зменшені товщини досліджуваних провідників зменшуються перехресні завади.

Таблиця 2.14 – Значення напруг перехресних завад при різних товщинах провідників

Товщина	HSOdd Xtalk, мВ	HSEven Xtalk, мВ	LSOdd Xtalk, мВ	LSEven Xtalk, мВ
1,2 мілідюймів	410.9	440.3	355.8	496.5
5 мілідюймів	707,9	539,3	620,1	576,6
6 мілідюймів	770,8	544,3	679,6	580,1

Дослідження роботи пристрою при різних матеріалах провідників

Дані про використані для дослідження матеріали та їх електропровідність наведені в табл. 2.15. Для встановлення матеріалу провідників виконуємо *Setup –> Cross-Section* і у вікні встановлюємо параметри провідників і діелектриків.

Дослідження будемо виконувати при довжині прилеглої частини провідників 550 мілідюймів, товщини провідників 5 мілідюймів, відстані між провідниками 1,2 мілідюйма. Після відповідних налаштувань були отримані звіти, які містять дані про величину перехресної завади і максимальні значення отриманих напруг (табл. 2.15).

Таблиця 2.15 – Значення напруг перехресних завад при різних матеріалах провідників

Матеріал	Значення електро-проводності, См/м	Максимальна напруга перех-ресної завади, мВ	Тип перехресної завади
Срібло	62500000	141.6	LSEvenXtalk
Платина	10000000	141.3	LSEvenXtalk
Мідь	59500000	143	LSEvenXtalk
Алюміній	35000000	142.7	LSEvenXtalk
Сталь	14000000	128.4	LSEvenXtalk
Золото	43000000	142.8	LSEvenXtalk
Нікель	14000000	141.9	LSEvenXtalk

Проаналізувавши отримані дані можна зробити висновок, що зміна матеріалу провідників несуттєво впливає на зміну величини напруги перехресної завади.

Дослідження роботи пристрою при різних відстанях між провідниками

Виконаємо дослідження значення напруг перехресних завад при відстані між провідниками 5 та 6 мілідюймів. Спочатку задаємо значення мінімальної відстані між провідниками при трасуванні (5 мілідюймів),

потім створюємо звіт напруг перехресних завад для провідника-*slave* N03039 та отримуємо значення напруг перехресних завад (рис. 2.189).

```
*****************************************************************************************************************
Single Neighbor Crosstalk at Receivers (mV) (Slow FTSMode)
*****************************************************************************************************************
Victim XNet    Victim Drvr  Victim Rcvr  HSOddXtalk  HSEvenXtalk  LSOddXtalk  LSEvenXtalk  Aggr XNet      Aggr Drvr
-------------  -----------  -----------  ----------  -----------  ----------  -----------  -------------  -------------
NEWMAG N03347  NEWMAG U2 4  NEWMAG U9 9  587.3       436.5        481         517.2        NEWMAG N03893  NEWMAG U14 11
-----------------------------------------------------------------------------------------------------------------
```

Рисунок 2.189 – Значення напруг перехресних завад
при відстані між провідниками 5 мілідюймів

Збільшуємо відстань між провідниками до 6 мілідюймів (рис. 2.190) і проводимо автоматичне трасування (рис. 2.191). Створюємо звіт напруг перехресних завад (рис. 2.192), з якого видно, що при відстані між провідниками 6 мілідюймів значення напруг перехресних завад відсутнє.

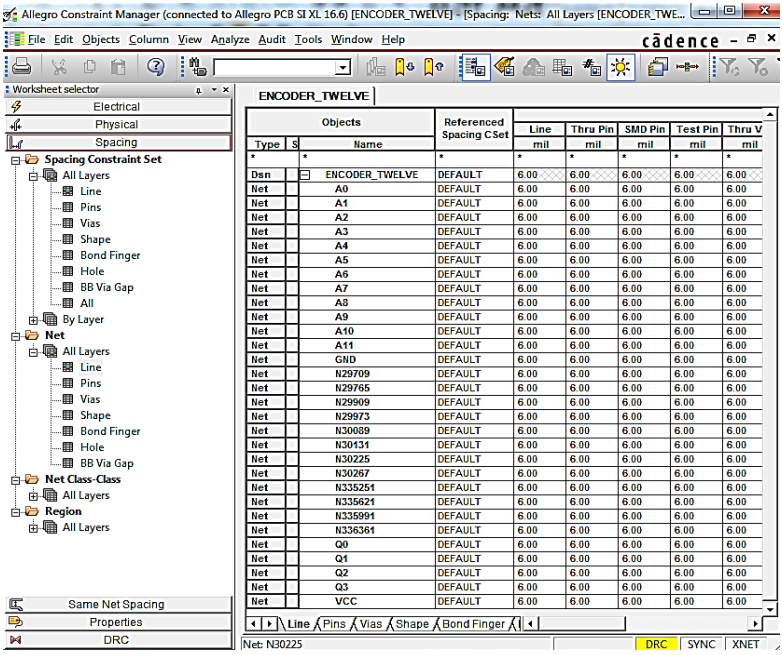


Рисунок 2.190 – Збільшення відстані між провідниками до 6 мілідюймів

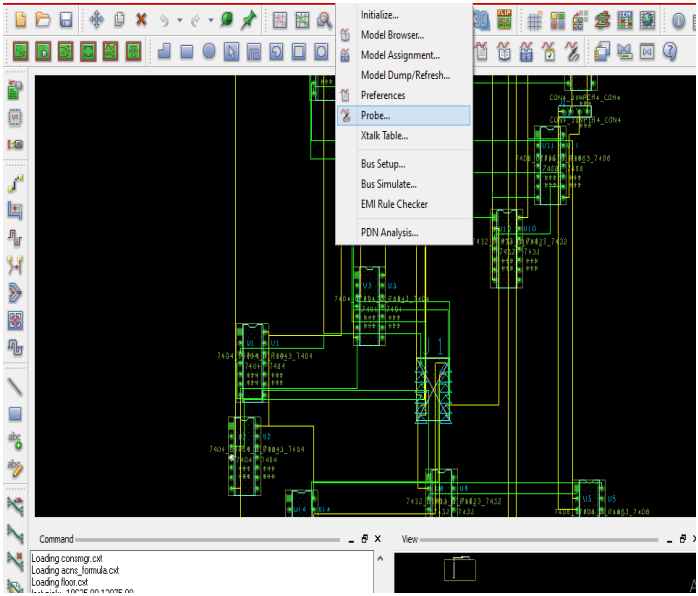


Рисунок 2.191 – Результат трасування

```
*****************************************************************************************************************
Single Neighbor Crosstalk at Receivers (mV) (Slow FTSMode)
*****************************************************************************************************************
Victim XNet  Victim Drvr  Victim Rcvr  HSOddXtalk  HSEvenXtalk  LSOddXtalk  LSEvenXtalk  Aggr XNet  Aggr Drvr
-----------  -----------  -----------  ----------  -----------  ----------  -----------  ---------  ---------
-----------------------------------------------------------------------------------------------------------------
```

Рисунок 2.192 – Значення напруг перехресних завад
при відстані між провідниками 6 мілідюймів

Дослідження роботи пристрою при різній кількості шарів між провідниками

Проводимо дослідження для виявлення перехресних завад між провідниками за умови їх знаходження в різних шарах друкованої плати – від знаходження в одному спільному шарі до розміщення в шарах, які знаходяться на протилежних рівнях.

Для виконання цього дослідження вибираємо *Setup –> Cross-Section -> Add Layer*, встановлюємо 5 шарів провідників та 4 шари діелектриків, які розміщенні між ними (рис. 2.193).

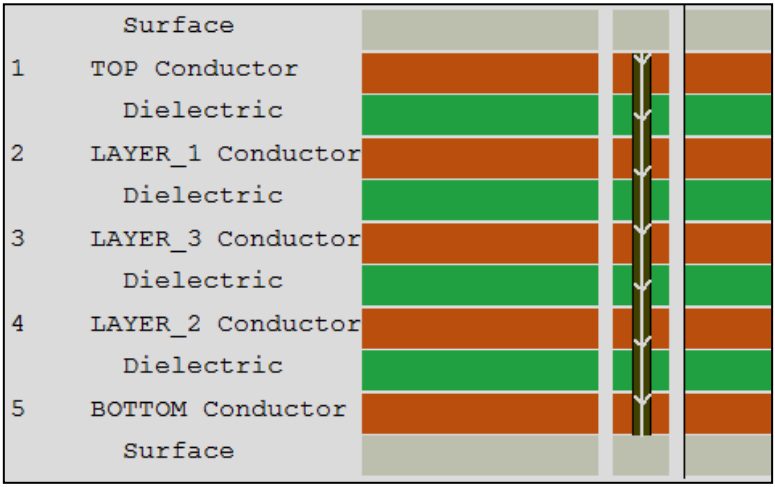


Рисунок 2.193 – Встановлення 5 провідних та 4 діелектричних шарів

Дослідження щодо виявлення величини перехресної завади проводимо за встановленими параметрами товщини шарів провідників 1,2 мілідюймів та 5 мілідюймів, товщин шарів діелектриків 8 мілідюймів та 16 мілідюймів. Довжина прилеглої частини провідників – 450 мілідюймів. Отримані під час моделювання значення напруг перехресних завад наведені в табл. 2.16 – 2.17.

Таблиця 2.16 – Значення напруг перехресних завад при товщині шару провідників 1,2 мілідюйма при розміщенні в різних шарах плати

Товщина діелектрика, мілідюймів	Провідники, розміщені в 1 шарі	Провідники, розміщені в 1 і 2 шарах	Провідники, розміщені в 1 і 3 шарах	Провідники, розміщені в 1 і 4 шарах	Провідники, розміщені в 1 і 5 шарах
8	129.3 мВ	116.5 мВ	94.28 мВ	76.06 мВ	69.89 мВ
16	145.9 мВ	113.2 мВ	83.79 мВ	68.72 мВ	62.89 мВ

Таблиця 2.17 – Значення напруг перехресних завад при товщині шару провідників 5 мілідюймів при розміщенні в різних шарах плати

Товщина діелектрика, мілідюймів	Провідники, розміщені в 1 шарі	Провідники, розміщені в 1 і 2 шарах	Провідники, розміщені в 1 і 3 шарах	Провідники, розміщені в 1 і 4 шарах	Провідники, розміщені в 1 і 5 шарах
8	170.4 мВ	138 мВ	103 мВ	77.01 мВ	73.26 мВ
16	180.3 мВ	123.8 мВ	88 мВ	65.74 мВ	59.32 мВ

Проаналізувавши отримані дані, можна зробити висновок, що при збільшенні товщини діелектричного шару зменшується значення напруги перехресної завади, однак не так сильно, як за умови розміщення провідників, які розміщені на шарах різних рівнів.

Також з'ясовано, що між провідниками з більш високими значеннями товщини, утворюються перехресні завади більшої величини при знаходженні в одному шарі. Однак, в разі розміщення провідників в різних шарах, завади зменшуються.

Значення напруги перехресної завади менше у провідників з більшою товщиною (5 мілідюймов), якщо вони розташовані в 1 і 5, 1 і 4 шарах при товщині шару діелектрика 16 мілідюймов, коли сумарна відстань між шарами при 1 і 4 дорівнює 58 мілідюймів.

На рис. 2.194 наведені результати моделювання: імпульсний сигнал (показаний чорний кольором) подається на вхід провідника-*master*; сигнал провідника-*slave* повинен мати постійний логічний рівень одиниці за відсутності перешкод в даному моделюванні в разі знаходження в одному шарі (показаний синім кольором), а також в разі 1 і 5 шарів при 16 мілідюймах діелектрика між шарами (показаний сірим кольором). Таким чином, отримуємо зменшення амплітуди спотворення сигналу на провіднику-*slave*.

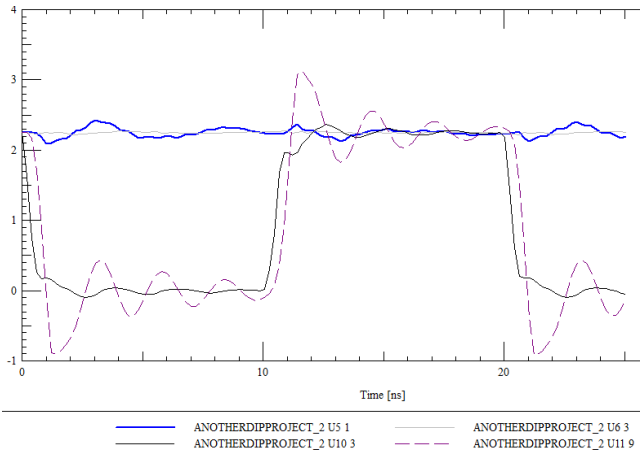


Рисунок 2.194 – Зменшення амплітуди спотворення сигналу при розміщенні провідників в різних шарах і збільшення відстані між ними

Дослідження роботи пристрою при різній кількості прилеглих провідників

Проведемо дослідження, яке визначає вплив збільшення кількості некоректно розміщених провідників в одному шарі. Для цього було виконано моделюванню за умови можливої взаємодії між 2, 5 та 10 провідниками (рис. 2.195), а саме створені умови для виникнення перехресних завад між провідниками для подальшого аналізу їх величини.

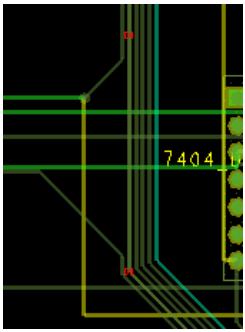


Рисунок 2.195 – Збільшення кількості прилеглих провідників

За умови взаємодії між двома провідниками, що знаходяться в 1 шарі, було сформовано звіти значень напруг перехресних завад (рис. 2.196), які занесені до табл. 2.18 (максимальне значення становить 730,3 мВ у HSOddXtalk). Після зміни кількості провідників були отримано дані про величину перехресних завад для 5 та 10 провідників, звіт про наявність та величину перехресних завад для 5 прилеглих провідників наведено на рис. 2.197, а для 10 провідників – на рис. 2.198.

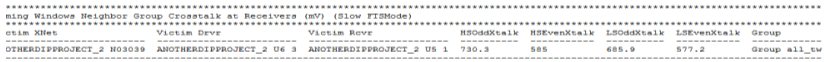


Рисунок 2.196 – Значення напруг перехресних завад при 2 прилеглих провідниках

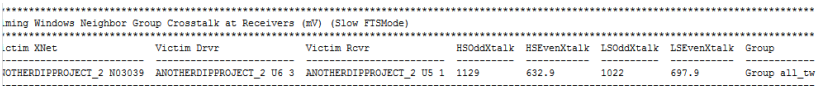


Рисунок 2.197 – Значення напруг перехресних завад при 5 прилеглих провідниках

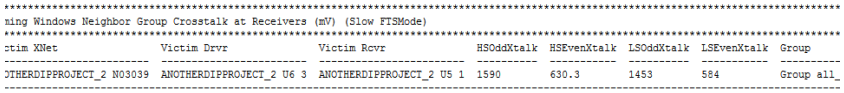


Рисунок 2.198 – Значення напруг перехресних завад при 10 прилеглих провідниках

Таблиця 2.18 – Значення напруг перехресних завад залежно від кількості провідників

Кіль-кість	HSOddXtalk	HSEvenXtalk	LSOddXtalk	LSEvenXtalk
2	730.3	585	685.9	577.2
5	1129	632	1022	697.9
10	1590	630.3	1453	584

Графіки, які демонструють залежність зміни значення напруги перехресної завади від зміни кількості провідників наведені на рис. 2.199 (2 провідники) і рис. 2.200 (10 провідників).

На цих графіках можна побачити імпульсні сигнали і зміну амплітуди напруги завади на провідниках-*slave*.

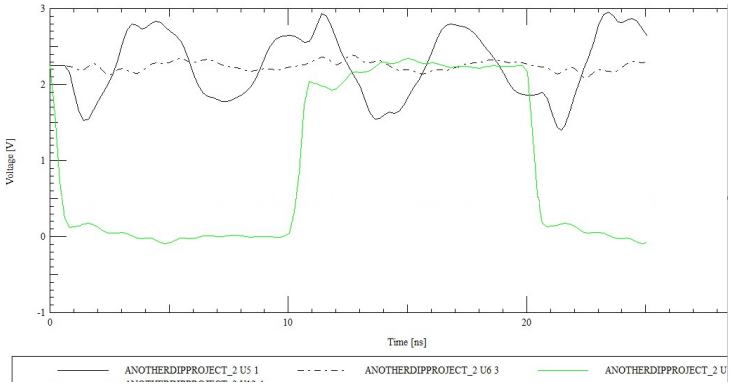


Рисунок 2.199 – Зміна сигналу при використанні 2 провідників

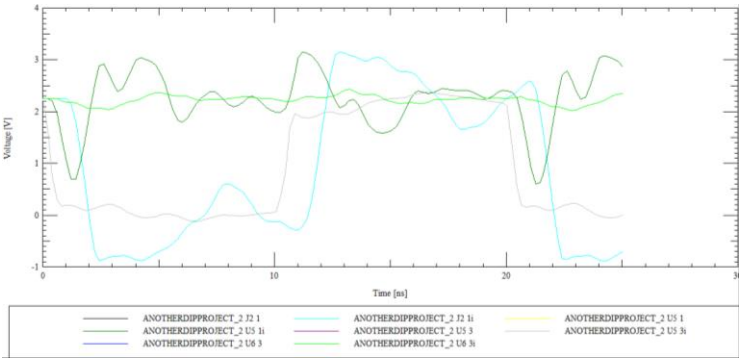


Рисунок 2.200 – Зміна сигналу при використанні 10 провідників

Завдання для виконання практичної роботи

Виконати дослідження розробленого в практичній роботі № 1 пристрою для оцінки його роботи з урахуванням EMC.

РОЗДІЛ 3
ВИКОРИСТАННЯ САПР OrCAD ДЛЯ МОДЕЛЮВАННЯ СКЛАДНИХ ПРИСТРОЇВ

Лабораторна робота 1

Розробка функціональної схеми пристрою. Розбиття схеми на ієрархічні рівні. Моделювання елементів нижнього ієрархічного рівня

Мета роботи: отримання та закріплення практичних навичок моделювання логічних елементів у САПР OrCAD.

Теоретичні відомості

Функціональна схема цифрового пристрою може бути розбита на ієрархічні рівні, найпростішим з яких є рівень логічних елементів, які служать базою для проєктування більш складних пристроїв. Саме тому дослідження параметрів елементної бази є важливим для подальшого проєктування та коректного функціонування розробленого пристрою.

САПР OrCAD має кілька серій логічних елементів, які відрізняються не лише різним набором логічних елементів, а й внутрішніми параметрами цієї логіки.

Через те, що проходження вхідного сигналу через елемент вимагає певного часу, встановлення рівня вихідного сигналу відбувається через часовий інтервал, який є одним із найважливіших параметрів логіки та називається затримкою елемента. Як правило, йдеться про кілька наносекунд, які, тим не менш, при створенні складних пристроїв з великою кількістю логічних елементів, накопичуючись, можуть призвести до порушень в роботі розробленого пристрою. Затримки елементів різних серій можуть суттєво відрізнятися, тому не рекомендується комбінувати елементи різних серій.

Завдання: розробити функціональну схему пристрою згідно з табл. 3.1. Виконати моделювання елементів нижнього ієрархічного рівня, отримати часові діаграми та оцінити затримку кожного елемента.

Таблиця 3.1

№ вар.	Індивідуальне завдання	Серія мікро-схем
1	Схема додавання-віднімання двійкових 6-розрядних чисел	7400
2	Швидкодіючий паралельний 4-розрядний помножувач	74HC
3	Матричний помножувач двійкових 4-розрядних чисел	74L
4	Схема переведення 8-розрядних двійкових чисел у двійково-десятковий код	7400
5	Схема додавання-віднімання двійкових 8-розрядних чисел	74AC
6	Матричний помножувач двійкових 8-розрядних чисел	74AS
7	Схема додавання-віднімання двійкових 8-розрядних чисел	74S
8	Схема порівняння 12-розрядних кодів	74F
9	Схема, що реалізує 15 логічних функцій від чотирьох аргументів	74S
10	Схема переведення двійково-десяткових чисел у двійкову систему числення	7400
11	Схема помножувача двійкових 3-розрядних чисел	74HC
12	Схема реалізації операцій логічного додавання, віднімання, «суми за модулем 2» та інверсії	74AS
13	Схема ділення двійкових чисел	74AS
14	Схема піднесення числа до степеня	7400
15	Схема перетворення додаткового коду в прямий та зворотний на основі арифметико-логічного пристрою	74HC

Приклад виконання завдання

Розробимо пристрій множення двох 8-розрядних чисел з аналізом молодших розрядів другого множника та зсувом першого множника у бік старших розрядів. Серія – 7400.

Для реалізації алгоритму множення (рис. 3.1) необхідні:

1) 16-розрядний регістр для часткової суми;

2) 8-розрядні регістри для множників;

3) 16-розрядний суматор;

4) лічильник імпульсів визначення кінця множення;

5) тригер для запам’ятовування старшого розряду множника.

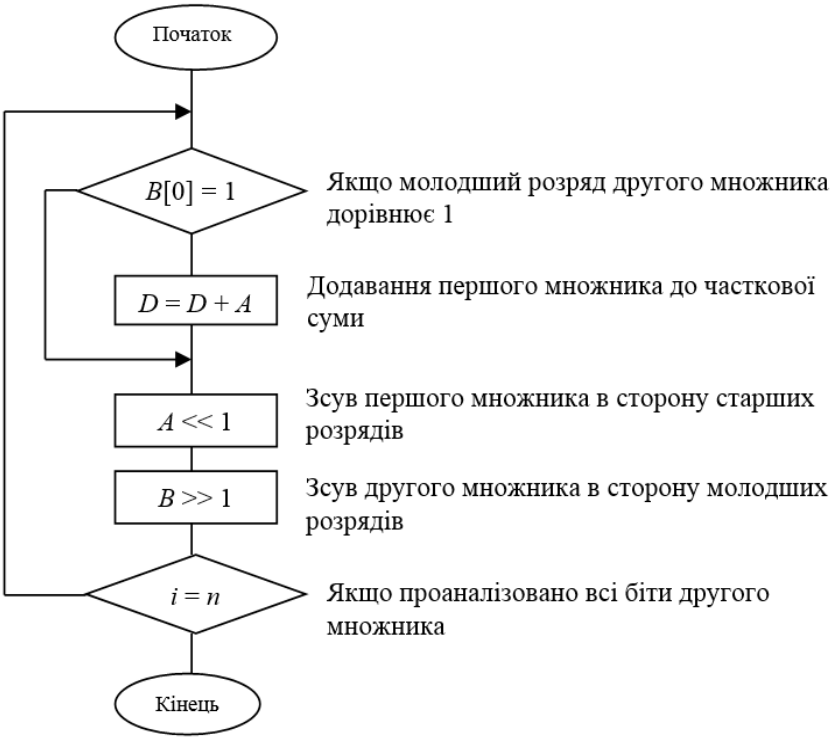


Рисунок 3.1 – Алгоритм множення у загальному вигляді

Функціональна схема розроблюваного пристрою наведена на рис. 3.2. Виконаємо розбиття схеми на п’ять ієрархічних рівнів:

– елементи 1-го рівня ієрархії:

НІ, 2І, 2І-НІ, 3І, 3І-НІ, 2АБО-НІ, 3АБО-НІ, 2-ВИКЛЮЧНЕ АБО.

– елементи 2-го рівня ієрархії:

тригери RS, D, JK, T; суматор; мультиплексор; пристрій керування.

– елементи 3-го рівня ієрархії:

4-розрядний зсувний регістр; 4-розрядний суматор; 4-розрядний лічильник; 16-розрядний паралельний регістр.

– елементи 4-го рівня ієрархії:

16-розрядний зсувний регістр; 16-розрядний суматор; 8-розрядний регістр.

– елементи 5-го рівня ієрархії:

елементом 5-го рівня ієрархії є пристрій множення двох восьмирозрядних чисел.

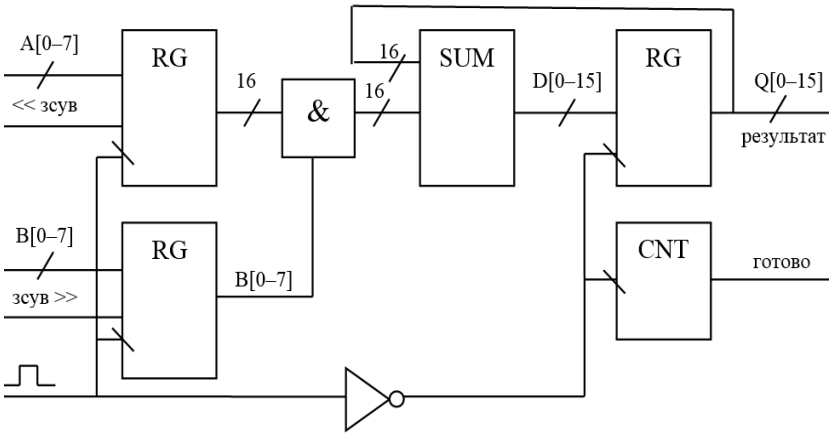


Рисунок 3.2 – Функціональна схема пристрою

Тепер створимо новий проєкт, для цього запускаємо OrCAD Capture, потім у головному меню вибираємо *File – New – Project*. Вводимо ім'я та каталог для створюваного проєкту. Відповідно до індивідуального завдання (табл. 3.1) вибираємо елементну базу.

Потім, застосовуючи мінімаксний метод обліку затримок логічних елементів, виконаємо моделювання:

– подаємо на входи елементів можливі комбінації цифрових сигналів;

– отримуємо часові діаграми вхідних і вихідних сигналів, а також оцінюємо затримку кожного елемента.

Для виконання цих дій, спочатку розмістимо на робочому полі логічні елементи. Для цього в головному меню обираємо *Place – Part*, потім додаємо бібліотеку. В нашому випадку це бібліотека 7400 та *SOURCE* – у вікні натискаємо кнопку *Add Library* (рис. 3.3) і вибираємо необхідні бібліотеки у вигляді файлів з розширенням *.olb, які можна знайти у каталозі *CADENCE\SPB_16.5\tools\capture\library\pspice*.

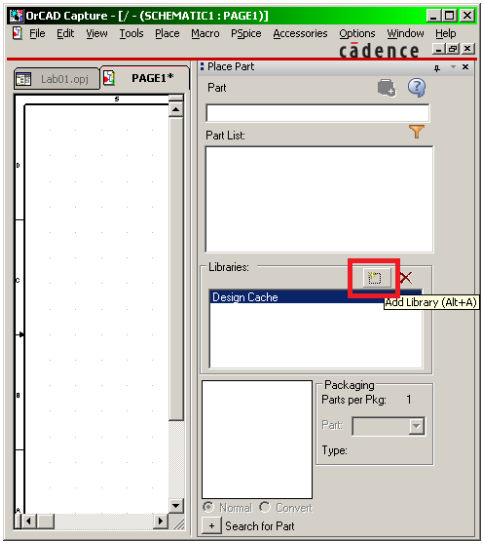


Рисунок 3.3 – Додавання бібліотеки

На рис. 3.4 наведено схему дослідження, яка складається з логічних елементів та генератора сигналів.

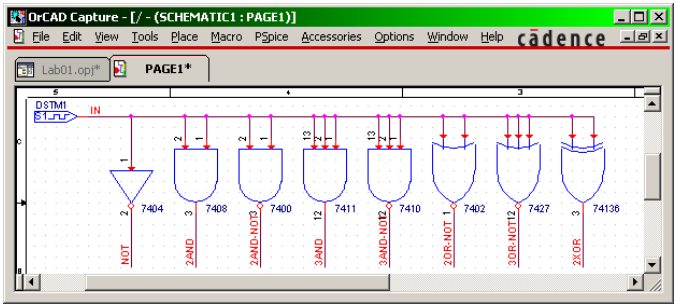


Рисунок 3.4 – Схема дослідження

Тепер все готове до виконання головної операції – моделювання. Для цього потрібно створити профіль моделювання: у меню вибрати *PSpice – New Simulation Profile*. У вікні вводимо ім’я профілю і

натискаємо *Create*. Перед нами з'являється вікно конфігурації профілю моделювання, в якому вводимо час моделювання 2 мкс (рис. 3.5).

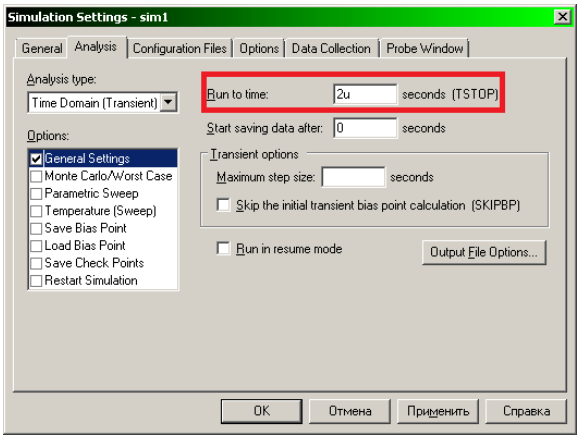


Рисунок 3.5 – Завдання часу моделювання

У вкладці *Options* категорії *Gate-level Simulation* виставляємо параметр *Worst-case* (min/max), як показано на рис. 3.6.

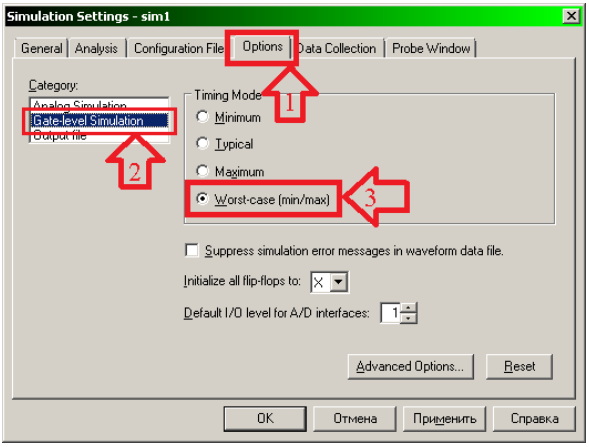


Рисунок 3.6 – Підключення мінімаксного режиму

Тепер у головному меню *вибираємо PSpice – Markers – Voltage Level* та встановлюємо «вольтметр» на провіднику (рис. 3.7).

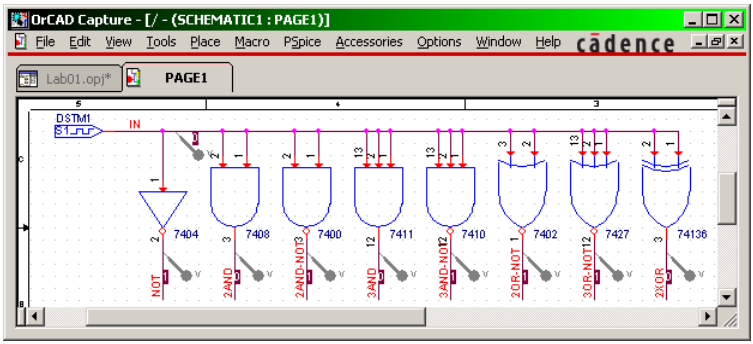


Рисунок 3.7 – Розташування «вольтметрів»

Для моделювання роботи пристрою в головному меню вибираємо *PSpice – Run*. В результаті отримуємо часові діаграми (рис. 3.8).

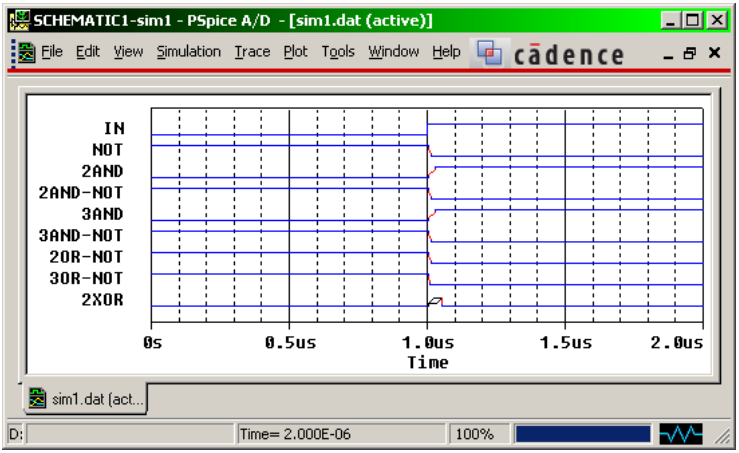


Рисунок 3.8 – Часові діаграми

Щоб визначити період затримки та невизначеності, збільшуємо масштаб часової діаграми (рис. 3.9) і заносимо дані в табл. 3.2.

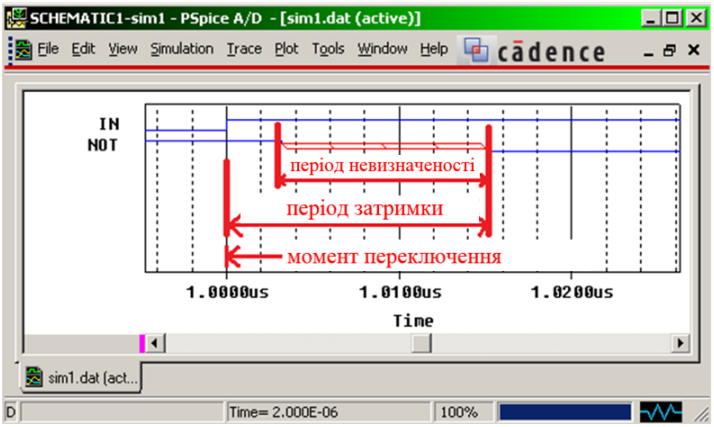


Рисунок 3.9 – Часові діаграми елемента *NOT*

Таблиця 3.2

Елемент (бібліотека 7400)	Період затримки $T_з$, нс	Період невизначеності $T_{невиз}$, нс
НІ	15	11,8
2І	27	20
2І–НІ	15	12,2
3І	27	20,2
3І–НІ	15	12,2
2АБО-НІ	15	11,8
3АБО-НІ	11	8,2
2-ВИКЛЮЧНЕ АБО	55	50,2

Лабораторна робота 2

Моделювання елементів другого ієрархічного рівня

Мета роботи

Розробка функціональної схеми пристрою. Отримання та закріплення практичних навичок проєктування та моделювання елементів другого ієрархічного рівня у САПР OrCAD.

Завдання: виконати проєктування та моделювання елементів другого ієрархічного рівня.

Приклад виконання завдання

Елементами другого ієрархічного рівня для заданого пристрою є: тригери RS, D; суматори; мультиплексори. Ці схеми реалізуються на елементах першого ієрархічного рівня.

Моделювання *D* – тригера

На рис. 3.10 наведено схему *D*-тригера.

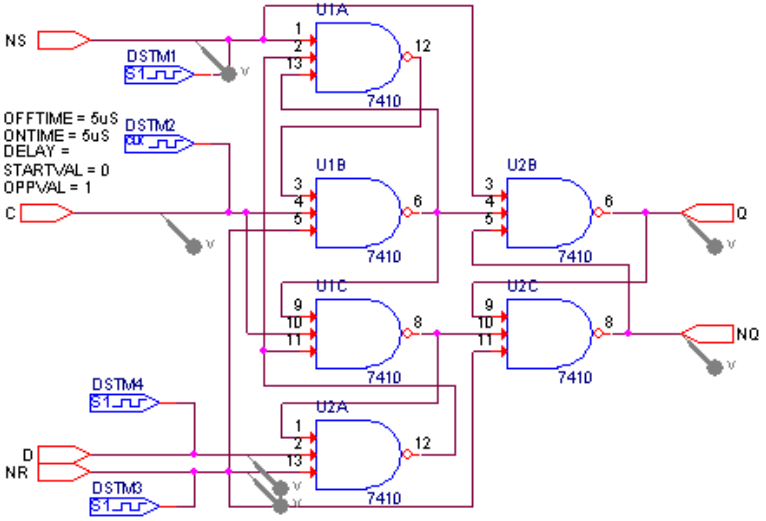


Рисунок 3.10 – *D*-тригер

Виконуємо моделювання *D*-тригера (рис. 3.11). Визначаємо час невизначеності (42 нс) та час затримки (52 нс) для виходу *NQ* при переході від «1» до «0» (рис. 3.12).

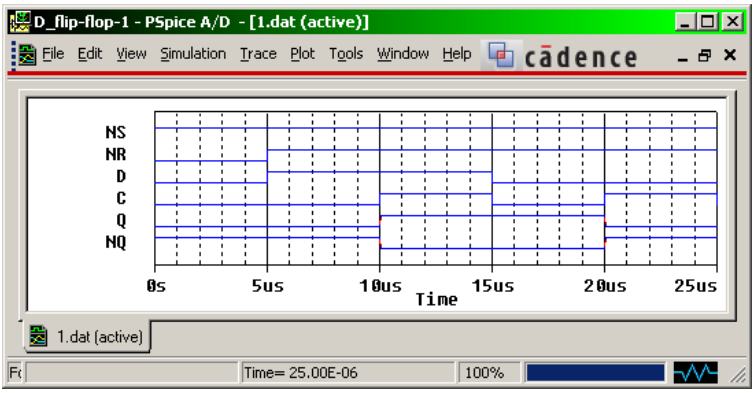


Рисунок 3.11 – Часові діаграми роботи *D*-тригера

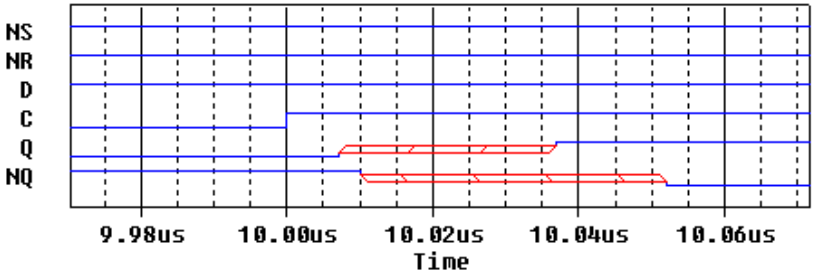


Рисунок 3.12 – Часові діаграми

Моделювання *RS*-тригера

Внаслідок того, що пристрої, які моделюються в даній роботі, є кінцевими і будуть використовуватися в наступних лабораторних роботах, то для кожного пристрою створюватимемо свою окрему схему, при чому все це будемо робити в одному проєкті. Для цього в браузері файлів даного проєкту (рис. 3.13) необхідно клацнути ПКМ по назві проєкту з розширенням *.dsn і вибрати пункт *New Schematic* .

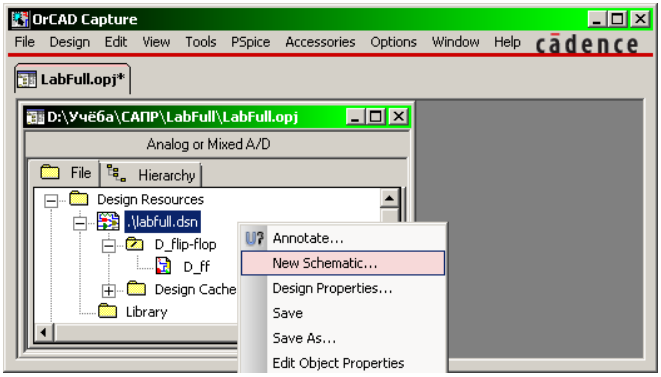


Рисунок 3.13 – Створення нової схеми у проєкті

Тепер створюємо лист, на якому будемо збирати схему пристрою (*RS*-тригер) – це пункт *New Page*.

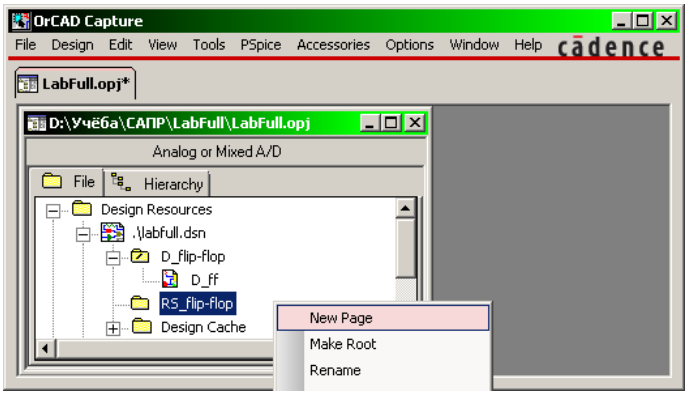


Рисунок 3.14 – Створення нової сторінки

На рис. 3.15 показано двоступінчастий синхронний *RS*-тригер з додатковим входом скидання. Щоб промоделювати цю схему, її необхідно зробити кореневою у файловій ієрархії проєкту. Для цього викликаємо меню схеми *RS_flip-flop* і вибираємо пункт *Make Root* (рис. 3.16).

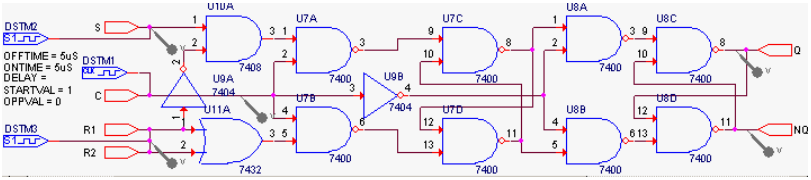


Рисунок 3.15 – Схема *RS*-тригера

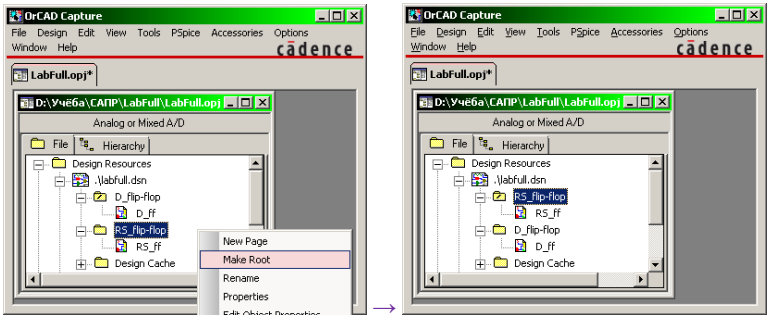


Рисунок 3.16 – Призначення кореневої схеми

Для кожної схеми потрібно створювати профіль моделювання. Визначаємо час невизначеності (59,2 нс) та час затримки (74 нс) для виходу *NQ* при переході від «0» до «1» (рис. 3.17).

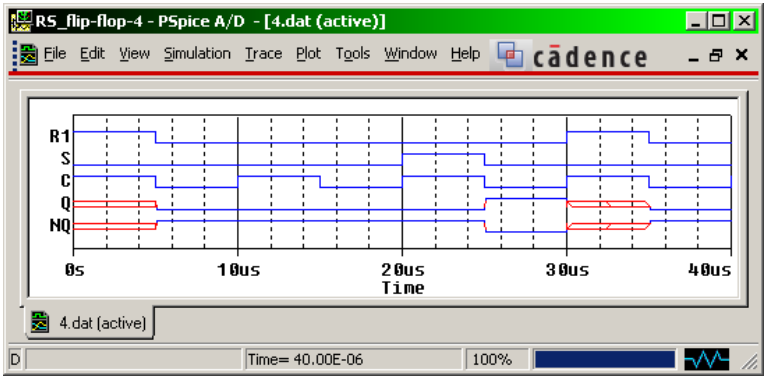


Рисунок 3.17 – Часова діаграма роботи *RS* -тригера

Моделювання *JK* - тригера

На рис. 3.18 наведена схема *JK*-тригера – це тригер, який при подачі «1» на вхід *J* та «0» на вхід *K* переключається в «1». При подачі на ці входи сигналу «0» – переходить в режим зберігання, а при подачі сигналу двох «1» відбувається постійне перемикання станів. Синхронізація виконується по спаду імпульсу (рис. 3.19).

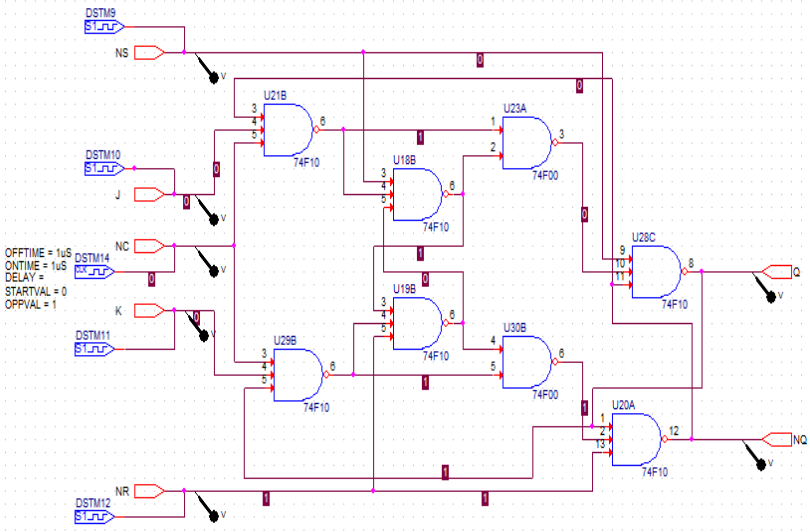


Рисунок 3.18 – Схема *JK*-тригера

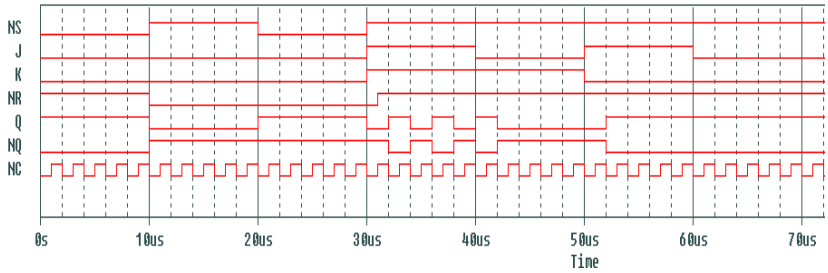


Рисунок 3.19 – Часові діаграми *JK*-тригера

Моделювання *JK*-тригера в мінімаксному режимі показало

неприйнятні результати, тому було обрано стандартний режим *Typical*. Визначаємо час затримки (25 нс) для виходу *NQ* під час переходу від «0» до «1» (рис. 3.20).

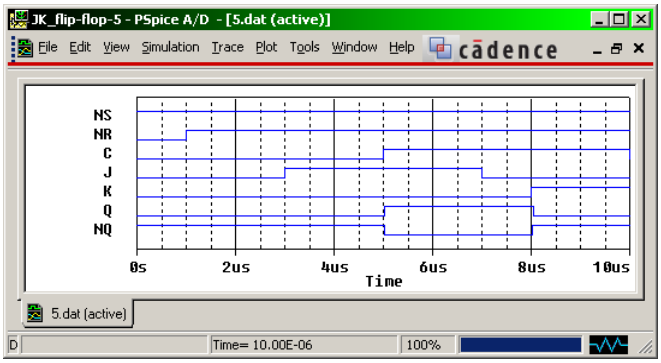


Рисунок 3.20 – Часова діаграма роботи *JK*-тригера

Моделювання суматора

На рис. 3.21 представлена схема одного з варіантів побудови повного однорозрядного суматора.

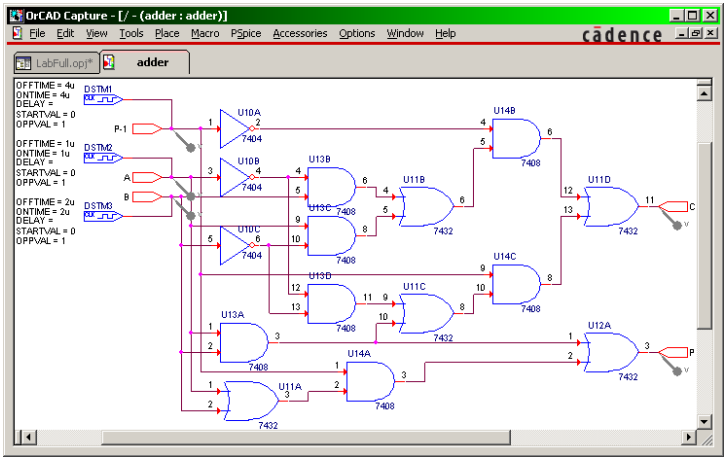


Рисунок 3.21 – Схема суматора

Визначаємо час невизначеності (95 нс) та час затримки (106 нс) для виходу *P* при переході від «0» до «1» (рис. 3.22).

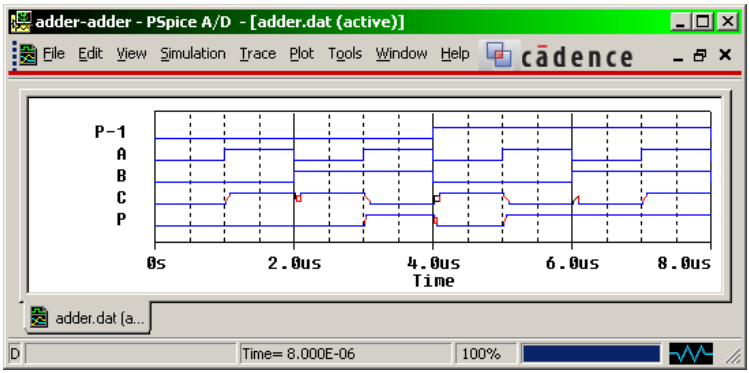


Рисунок 3.22 – Часова діаграма роботи суматора

Моделювання мультиплексора

На рис. 3.23 представлена схема мультиплексора 4×1. Визначаємо час невизначеності (42,2 нс) та час затримки (57 нс) для виходу *Y* при переході від «0» до «1» (рис. 3.24).

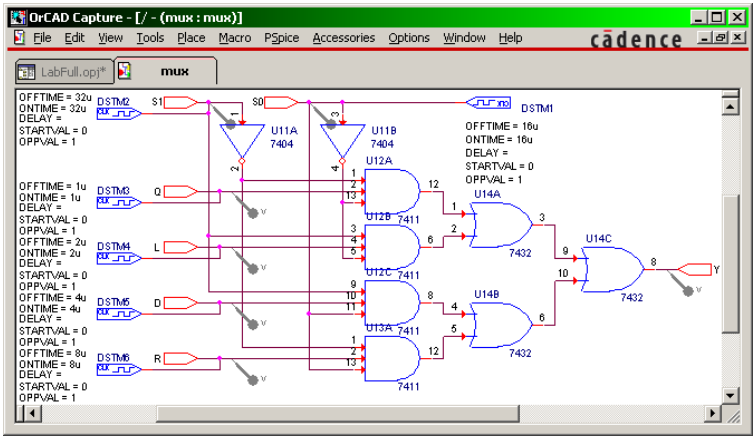


Рисунок 3.23 – Схема мультиплексора

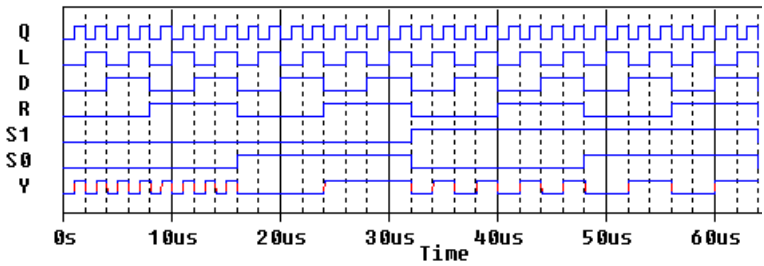


Рисунок 3.24 – Часова діаграма роботи мультиплексора

Моделювання керуючого пристрою

На рис. 3.25 представлена схема керуючого пристрою, який дозволяє пропускати через себе вхідні дані на вихід незмінними, якщо на керуючий вхід подана «1», інакше – «0».

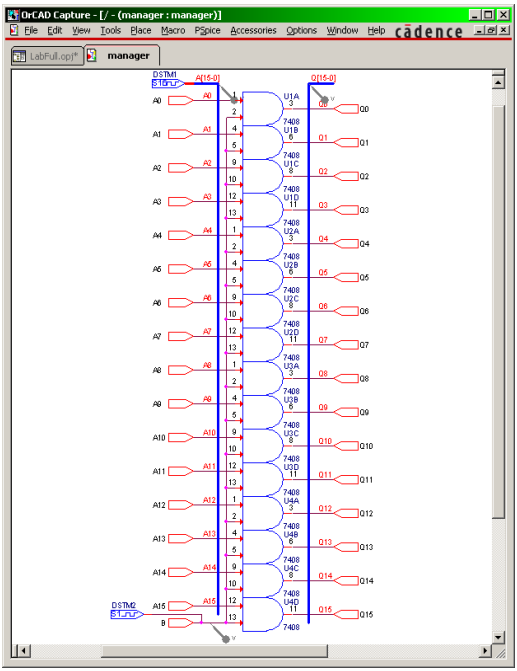


Рисунок 3.25 – Схема керуючого пристрою

Визначаємо період невизначеності та затримки (рис. 3.26) та заносимо дані в табл. 3.3.

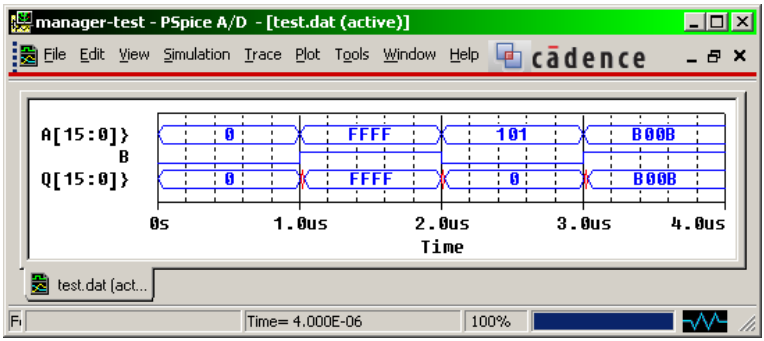


Рисунок 3.26 – Часова діаграма роботи керуючого пристрою

Таблиця 3.3

Пристрій	Період затримки, нс	Період невизначеності, нс
D- тригер	52	42
RS -тригер	74	59,2
JK -тригер	25	-
суматор	106	95
Мультиплексор	57	42,2
Пристрій керування	27	7

Лабораторна робота 3

Моделювання елементів третього ієрархічного рівня

Мета роботи

Розробка функціональної схеми пристрою. Отримання та закріплення практичних навичок проєктування та моделювання елементів третього ієрархічного рівня у САПР OrCAD.

Завдання: виконати проєктування та моделювання елементів третього ієрархічного рівня.

Приклад виконання завдання

Елементами третього рівня ієрархії в нашому випадку є чотирирозрядний регістр, суматор, лічильник, а також 16-розрядний регістр. У цій та наступних лабораторних роботах будемо використовувати раніше створені нами елементи другого ієрархічного рівня.

Моделювання 4-розрядного регістру

Створюємо нову схему і лист (рис. 3.27). Розміщуємо на листі ієрархічний блок мультиплексора. Для цього потрібно в головному меню вибрати *Place – Hierarchical Block*. У вікні в полі *Reference* пишемо будь-яке ім'я, а в полі *Implementation name* пишемо конкретне ім'я схеми (*mux*), як показано на рис. 3.27.

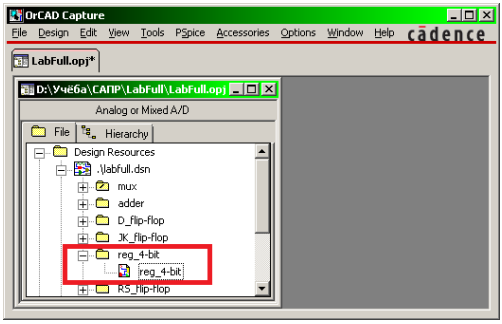


Рисунок 3.27 – Створення нової схеми

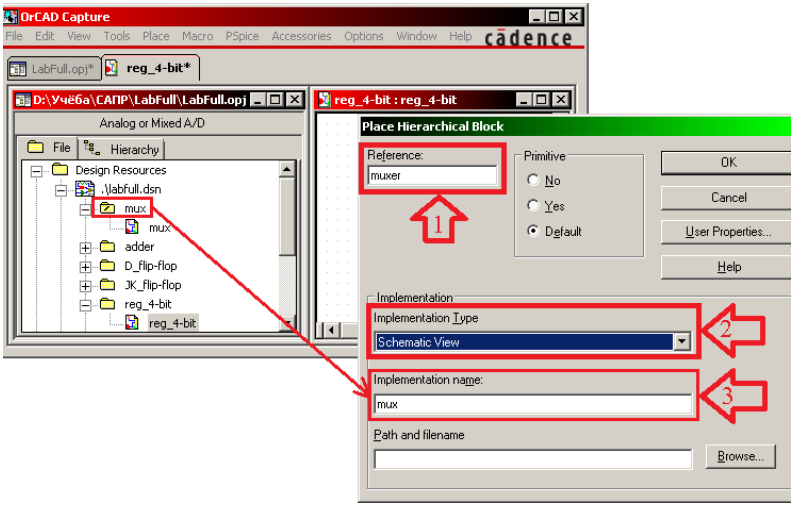


Рисунок 3.28 – Процес створення ієрархічного блоку

Після натискання кнопки *OK* рисуємо мультиплексор на листі у вигляді прямокутника довільних розмірів. Можна довільно змінювати розміри блоків та розташування портів (рис. 3.29).

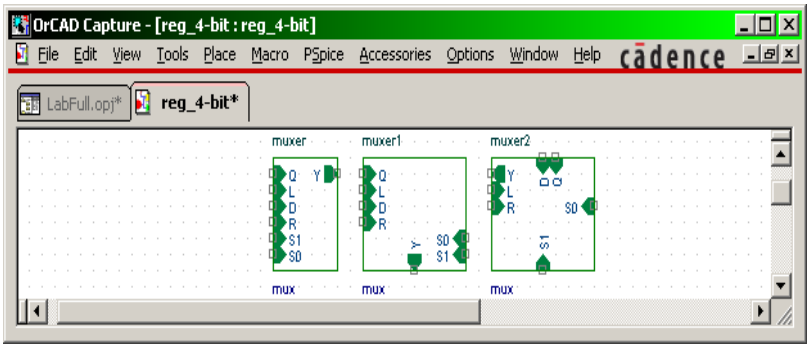


Рисунок 3.29 – Ієрархічні блоки на робочому листі

На рис. 3.30 представлений 4-розрядний зсувний регістр зі зворотною синхронізацією.

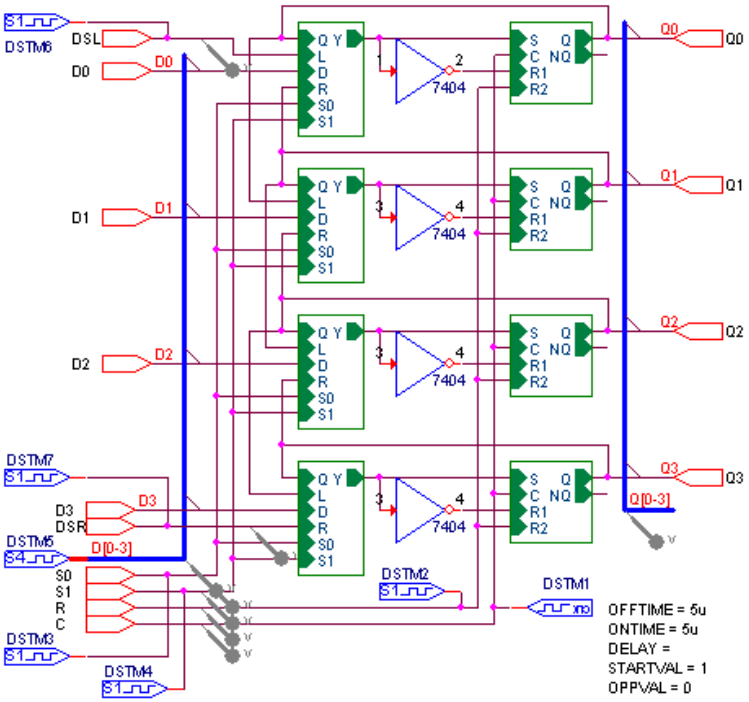


Рисунок 3.30 – Схема регістру

Елемент з назвою *DSTM5* типу *STIM4* з бібліотеки *SOURCE* є аналогом *STIM1*, але здатний «видати» на виході 4-бітне число (рис. 3.31).

COMMAND1	COMMAND2	COMMAND3
0s 0000	15u 0011	40u 0010

Рисунок 3.31 – Завдання вхідних сигналів у *STIM4*

Для 4-бітного числа, яке видає *STIM4*, використовується шина, яка зображена на схемі у вигляді широкого провідника синього кольору.

Для з'єднання однорозрядних провідників та шин використовуються «з'єднувачі», які можна вибрати в головному меню *Place – Bus Entry*.

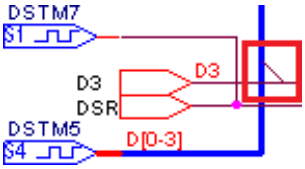


Рисунок 3.32 – Вид з’єднувача для шини

Важливу роль в даному випадку відіграють імена провідників та шин. Як бачимо із рис. 3.30, імена провідників такі: *D0*, *D1*, *D2*, *D3*, а ім’я шини, до якої вони приєднані, – *D* [0–3].

Правило побудови імені однобітового провідника таке:

ім’я <*індекс_розряду_в_шині*>;

а для шини:

ім’я [<*індекс_старшого_розряду*> – <*індекс_молодшого_розряду*>].

Щоб уникнути помилок, порти блоків не можна залишати без провідників. У нашому випадку порт *NQ* у *RS*-тригері повинен бути підключений до провідника, інакше буде помилка та неможливість виконати моделювання пристрою. При моделюванні обрано режим *Typical*, оскільки мінімаксний режим викликає збої. Надалі будемо використовувати лише режим *Typical*. Визначаємо час затримки (37 нс) для виходу *Q* [0–3] під час переходу «3» в «2» (рис. 3.33). Більш точно момент перемикання показано на рис. 3.34.

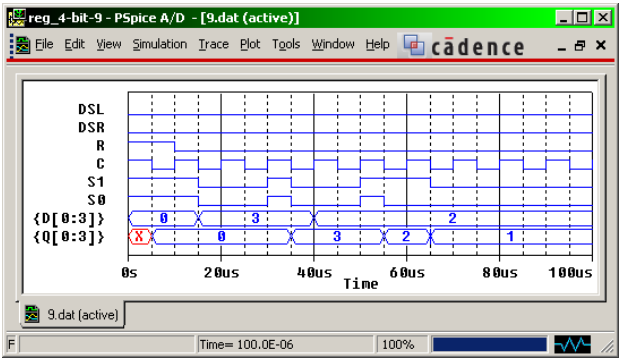


Рисунок 3.33 – Часова діаграма роботи регістру

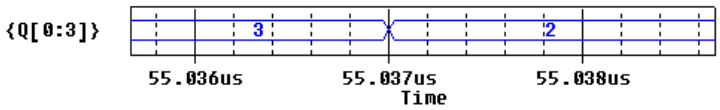


Рисунок 3.34 – Момент перемикання на виходах регістру

Моделювання 4-розрядного суматора

На рис. 3.35 показано 4-розрядний суматор із послідовним переносом.

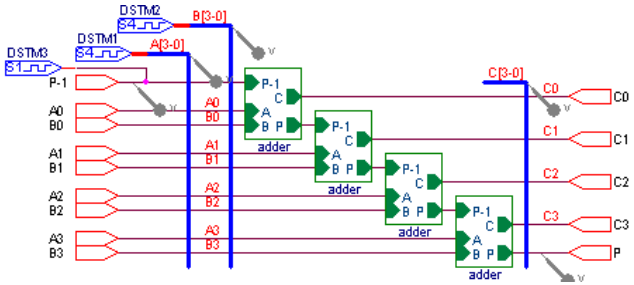


Рисунок 3.35 – Схема суматора

Для з'єднання шини та провідників можна не використовувати *Bus Entry*, якщо шина та провідник перетинаються між собою, або шина та провідник дотримуються правил іменування. Визначаємо час затримки (116,5 нс) виходу *C* [3–0] (рис. 3.36).

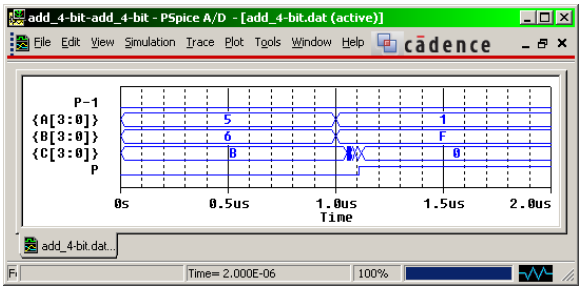


Рисунок 3.36 – Часова діаграма

Моделювання 4-розрядного лічильника

На рис. 3.37 показана схема 4-розрядного лічильника із прямою синхронізацією на основі *D*-тригерів.

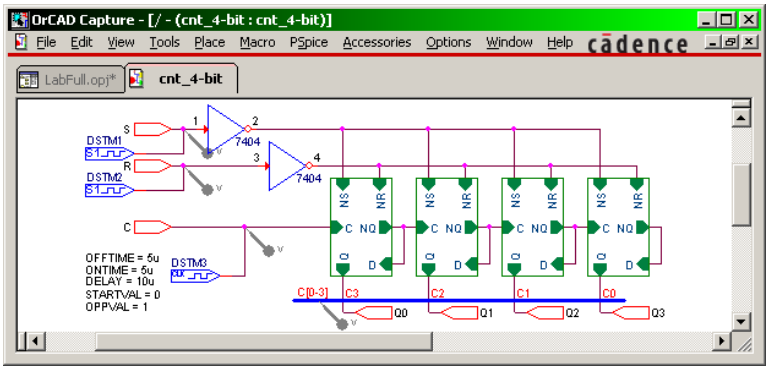


Рисунок 3.37 – Схема лічильника

Визначаємо час затримки (79 нс) для виходу *C* [0-3] (*Q0*:*Q3*) (рис. 3.38).

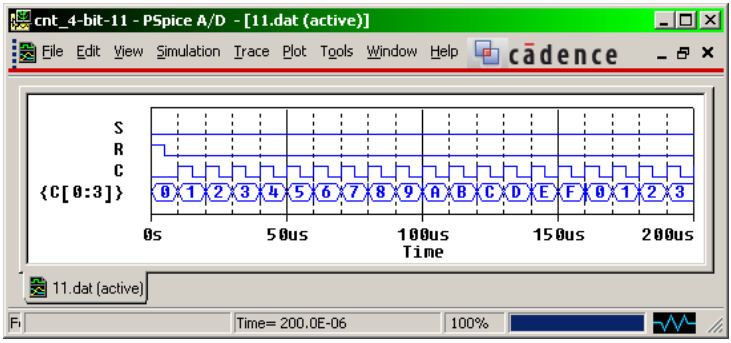


Рисунок 3.38 – Часова діаграма роботи лічильника

Моделювання 16-розрядного регістру

На рис. 3.39 наведено схему 16-розрядного регістру на основі *D*-тригерів.

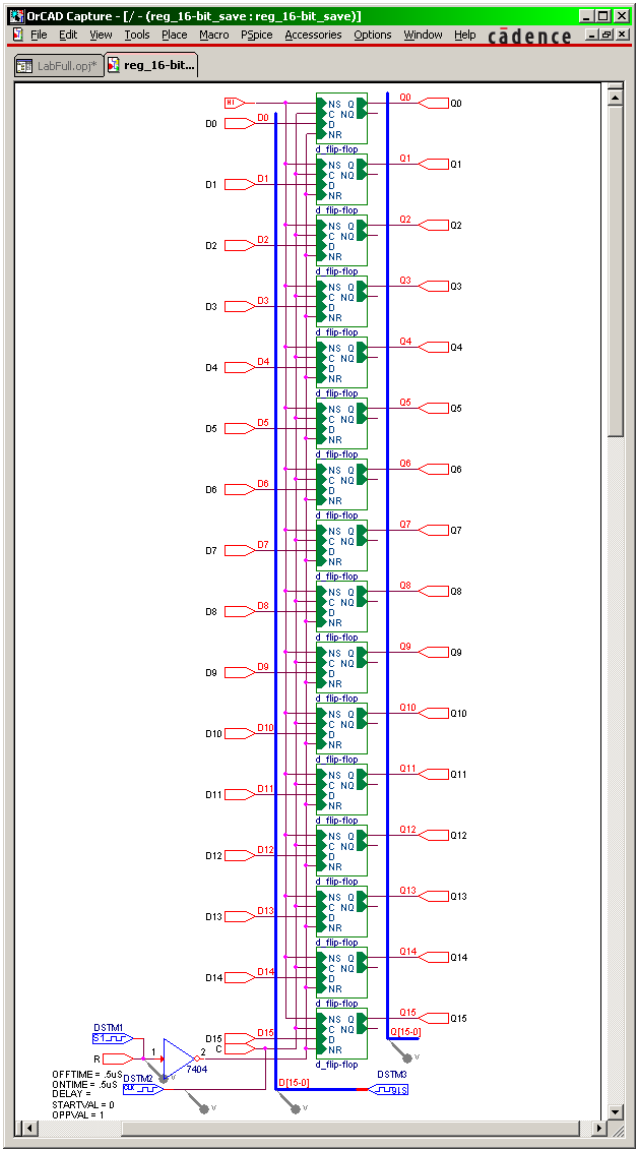


Рисунок 3.39 – Схема регістру

Визначаємо період затримки (рис. 3.40) та заносимо дані в табл. 3.4.

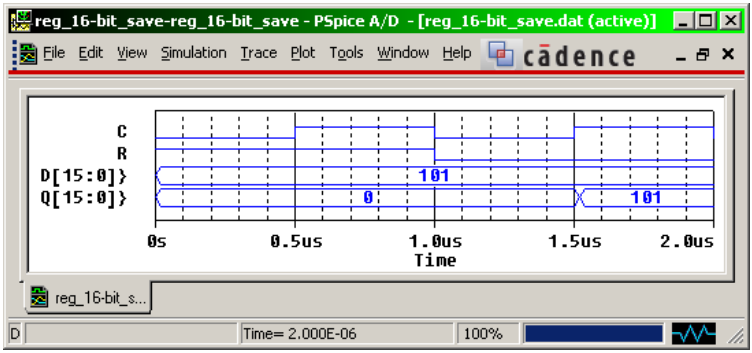


Рисунок 3.40 – Часова діаграма роботи регістру

Таблиця 3.4

Пристрій	Період затримки, нс
4-розрядний зсувний регістр	37
Суматор	116,5
Лічильник	79
16-розрядний паралельний регістр	18

Лабораторна робота 4

Моделювання елементів четвертого ієрархічного рівня

Мета роботи

Розробка функціональної схеми пристрою. Отримання та закріплення практичних навичок проєктування та моделювання елементів четвертого ієрархічного рівня у САПР OrCAD.

Завдання: виконати проєктування та моделювання елементів четвертого ієрархічного рівня.

Приклад виконання завдання

Елементами четвертого ієрархічного рівня для заданого пристрою є: 16-розрядний зсувний регістр; 8-розрядний регістр; 16-розрядний суматор. Ці схеми реалізуються на елементах третього ієрархічного рівня.

Моделювання 16-розрядного зсувного регістру

На рис. 3.41 показана схема 16-розрядного регістру зі зсувом, паралельним записом та зворотною синхронізацією.

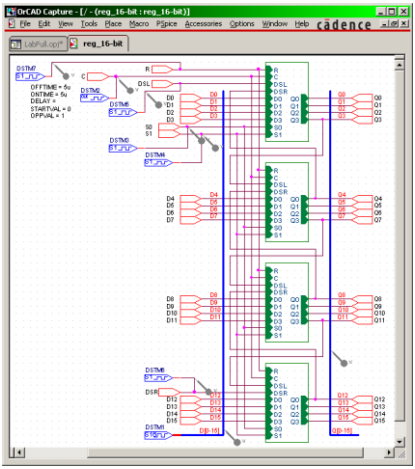


Рисунок 3.41 – Схема регістру

Елемент з назвою *DSTM1* типу *STIM16* з бібліотеки *SOURCE* є аналогом *STIM4*, тільки здатний «видати» на виході 16-бітне число. Визначаємо час затримки (37 нс) для виходу *Q* [0-15] (рис. 3.42).

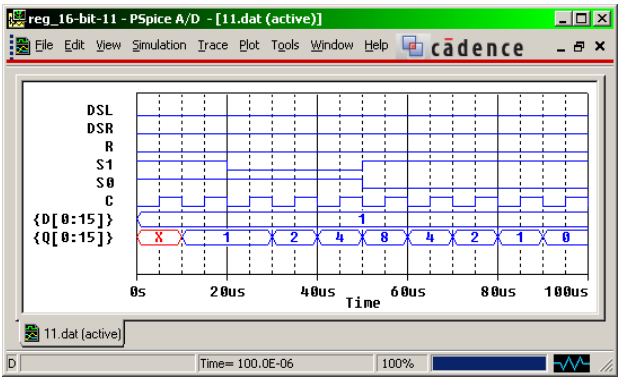


Рисунок 3.42 – Часова діаграма роботи регістру

Моделювання 8-розрядного регістру

На рис. 3.43 показана схема 8-розрядного регістру зі зсувом, паралельним записом та зворотною синхронізацією. Визначаємо час затримки (37 нс) для виходу *Q* [0-7] (рис. 3.44).

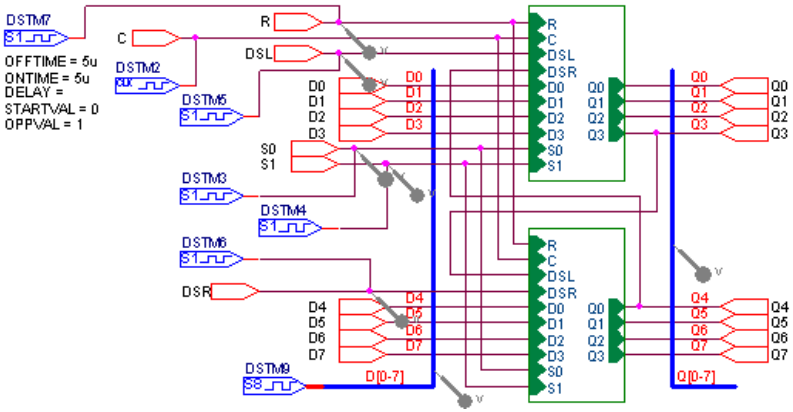


Рисунок 3.43 – Схема 8-розрядного регістру

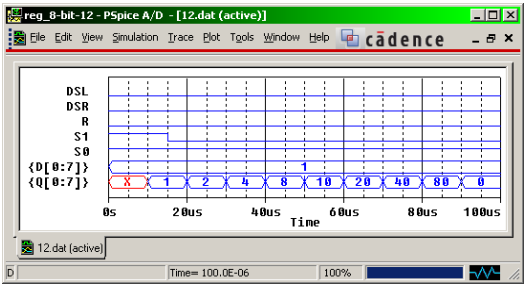


Рисунок 3.44 – Часова діаграма

Моделювання 16-розрядного суматора

На рис. 3.45 показана схема 16-розрядного суматора. Визначаємо час затримки (рис. 3.46) та заносимо ці дані в табл. 3.5.

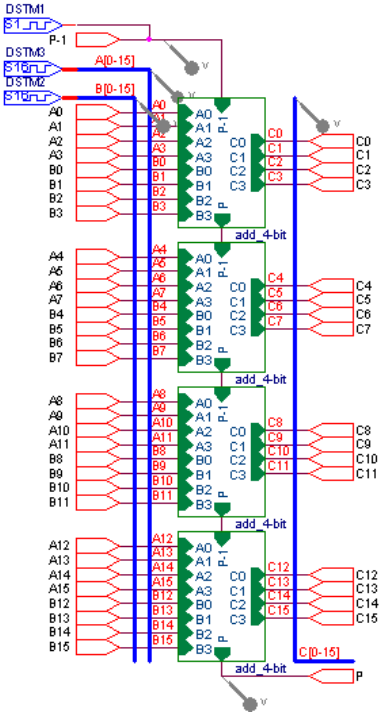


Рисунок 3.45 – Схема суматора

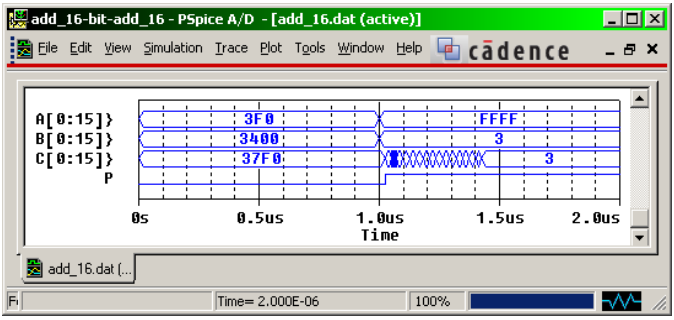


Рисунок 3.46 – Часова діаграма роботи суматора

Таблиця 3.5

Пристрій	Час затримки, нс
16-розрядний регістр	37
8-розрядний регістр	37
16-розрядний суматор	422,5

Лабораторна робота 5

Моделювання та аналіз правильності функціонування розробленого пристрою

Мета роботи

Виконати моделювання схеми розробленого пристрою та провести аналіз його функціонування в САПР OrCAD.

Завдання: виконати моделювання і аналіз функціонування розробленого пристрою.

Приклад виконання завдання

Схема розробленого пристрою будується на основі всіх ієрархічних рівнів, які були розглянуті в попередніх лабораторних роботах (рис. 3.47).

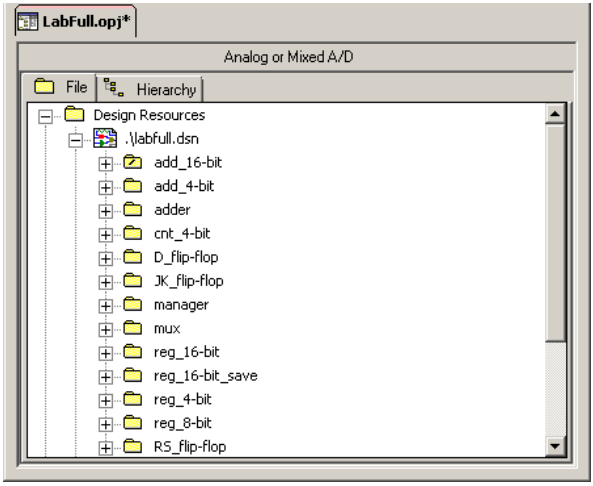


Рисунок 3.47 – Тип структури файлів з ієрархією

Схема створюється виходячи з функціональної схеми (рис. 3.2), яка наведена в лабораторній роботі № 1 (рис. 3.48). Результати моделювання пристрою наведені на рис. 3.49.

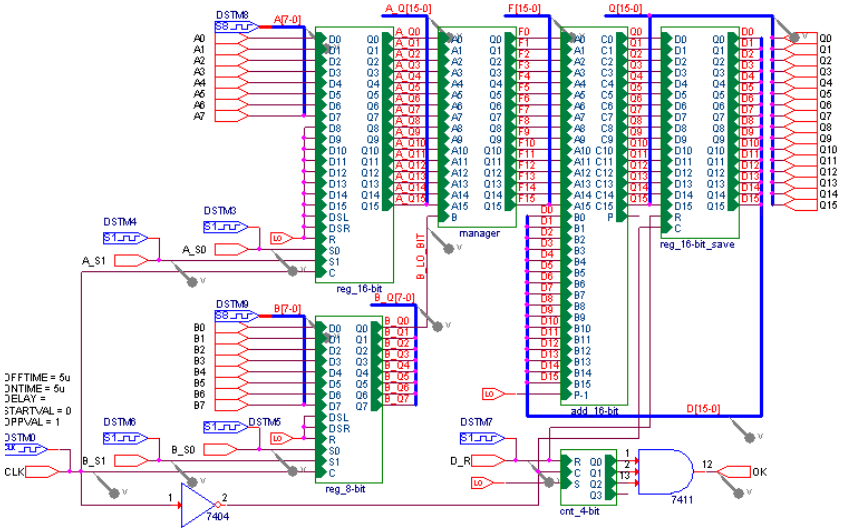


Рисунок 3.48 – Схема пристрою множення

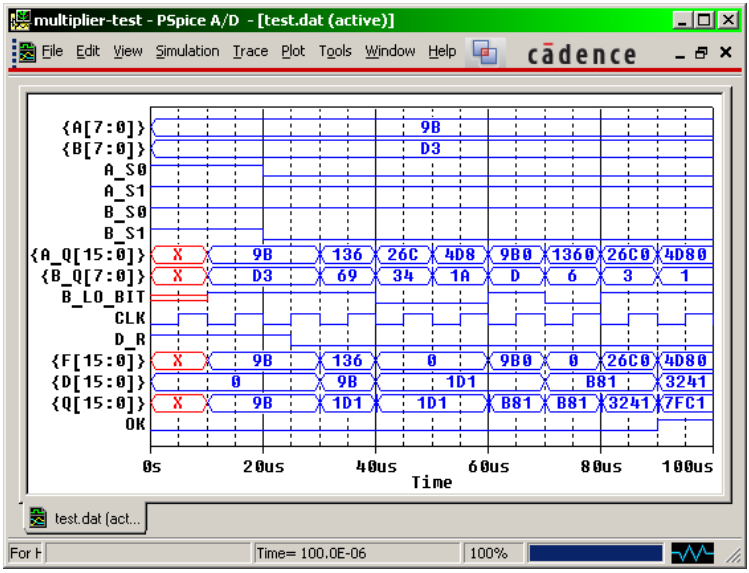


Рисунок 3.49 – Часова діаграма

Для перевірки правильності функціонування виконаємо множення двох чисел: $9B_{16} \times D3_{16} = 7FC1_{16}$ (табл. 3.6).

Таблиця 3.6

№ такту	Дія / стан
1	A = 00000000 **10011011** $_2$ = 9B $_{16}$ B = 1101001 | **1** $_2$ = 1 D = 0 D = A + D = 9B $_{16}$
2	A(<<) = 0000000 **10011011** 0 $_2$ = 136 $_{16}$ B(>>) = 0110100 | **1** $_2$ = 1 D = A + D = 1D1 $_{16}$
3	A(<<) = 000000 **10011011** 00 $_2$ = 26C $_{16}$ B(>>) = 0011010 | **0** $_2$ = 0
4	A(<<) = 00000 **10011011** 000 $_2$ = 4D8 $_{16}$ B(>>) = 0001101 | **0** $_2$ = 0
5	A(<<) = 0000 **10011011** 0000 $_2$ = 9B0 $_{16}$ B(>>) = 0000110 | **1** $_2$ = 1 D = A + D = B81 $_{16}$
6	A(<<) = 000 **100110110** 0000 $_2$ = 1360 $_{16}$ B(>>) = 0000011 | **0** $_2$ = 0
7	A(<<) = 00 **100110110** 00000 $_2$ = 26C0 $_{16}$ B(>>) = 0000001 | **1** $_2$ = 1 D = A + D = 3241 $_{16}$
8	A(<<) = 0 **100110110** 000000 $_2$ = 4D80 $_{16}$ B(>>) = 0000000 | **1** $_2$ = 1 D = A + D = 7FC1 $_{16}$

Значення часткових сум *D* збігаються з результатами моделювання (рис. 3.49). Після завершення обчислень пристрій повідомляє про готовність сигналом *OK*.

Лабораторна робота 6

Дослідження розробленого пристрою на швидкодію

Мета роботи: виконати дослідження розробленого пристрою на швидкодію.

Завдання: виконати дослідження розробленого пристрою на швидкодію. Визначити оптимальну частоту вхідних сигналів.

Приклад виконання завдання

Розраховуємо приблизне значення максимально допустимої частоти вхідних імпульсів:

$$F_{min} = 1 / T_{min},$$

де T_{min} – сума максимальних затримок елементів пристрою.

Максимальний час затримки суматора становить 422,5 нс, проте при роботі даного пристрою не використовується операція переносу, тому максимальна затримка дорівнює $t_з$ = 125 нс.

Таким чином, отримуємо:

$$T_{min} = 2 \times t_з = 2 \times 125 = 250 \text{ нс},$$

$$F_{max} = 1 / 250 \times 10^{-9} = 4 \text{ МГц}.$$

Перевіримо отримані дані, встановивши час перемикання в генераторі синхроімпульсів t = 125 нс (рис. 3.50).

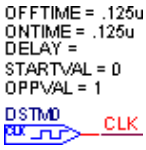


Рисунок 3.50 – Вид генератора та його параметрів

Для нормальної роботи пристрою час перемикання (з «1» в «0») елементів з іменами *DSTM3* і *DSTM6* при перемиканні синхроімпульсу (*DSTM0*) в 5u, було встановлено рівним 20 u (*A_S0* = *B_S1* = 20 u).

Для пристрою з ім’ям *DSTM7* (*D_R*) за тих же умов час перемикання становив 25u. Змінивши параметри генератора синхроімпульсів (*DSTM0*), необхідно змінити час перемикання для цих пристроїв.

Коефіцієнт k = 0,125 u / 5 u = 0,025.

Час перемикання:

$$DSTM3 = DSTM6 = 20 \text{ u} \times k = 20 \text{ u} \times 0{,}025 = 0{,}5 \text{ u},$$
$$DSTM7 = 25 \text{ u} \times k = 25 \text{ u} \times 0{,}025 = 0{,}625 \text{ u}.$$

Результати моделювання наведені на рис. 3.51, з якого видно, що результат обчислень (7FC1) на виході *Q* [15:0] збігається з сигналом вдалого закінчення *OK*. Це означає, що пристрій виконує свою функцію.

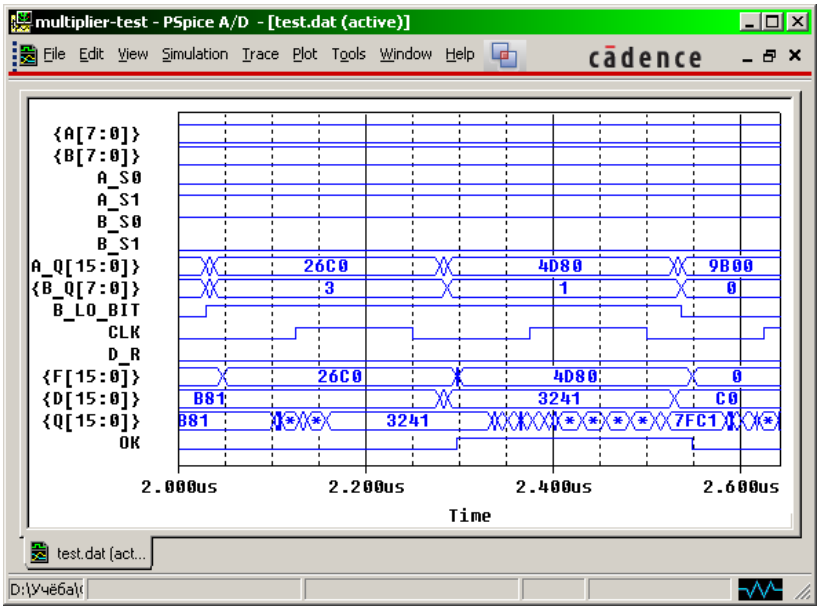


Рисунок 3.51 – Часова діаграма

У разі збільшення частоти до 5 МГц отримаємо часову діаграму, яка наведена на рис. 3.52, з якого видно, що хоча ми й спостерігаємо результати обчислень (7FC1) на виході *Q* [15:0], але час його появи не збігається з сигналом вдалого закінчення *OK*.

Отже, пристрій множення не виконує свою функцію, тому для підвищення швидкодії пристрою необхідно замінити його елементи на більш швидкодіючі.

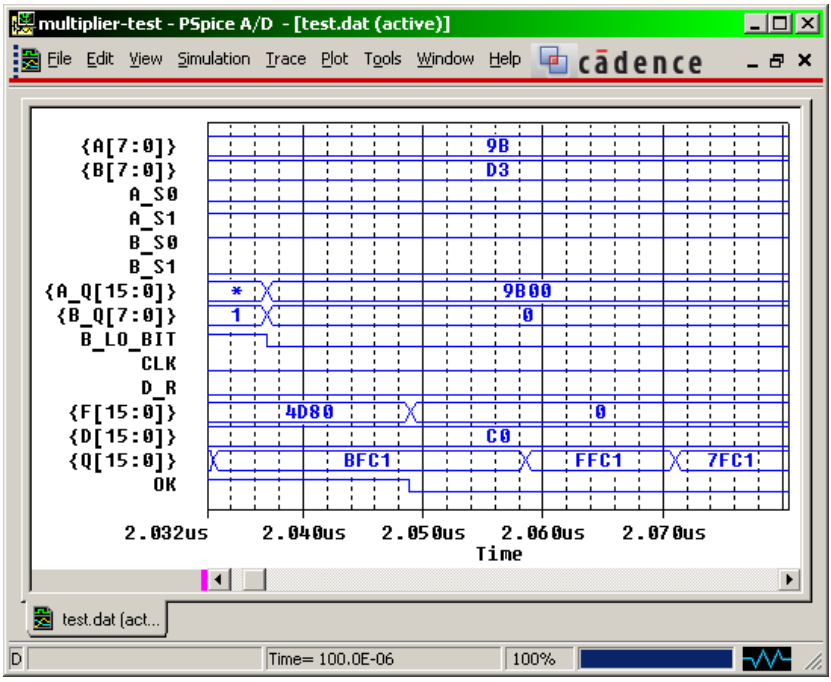


Рисунок 3.52 – Результати моделювання при збільшенні частоти вхідних сигналів

Лабораторна робота 7

Оцінка похибки виконання заданих операцій на розробленому пристрої

Мета роботи: виконати дослідження розробленого пристрою для оцінки похибки виконання заданих операцій.

Завдання: виконати оцінку похибки виконання заданих операцій на розробленому пристрої та на пристрої, який виконує аналогічні операції на аналогових блоках.

Приклад виконання завдання

Для оцінки похибки виконання заданих операцій збираємо схему, яка наведена на рис. 3.53. Два аналогові сигнали подаються через аналого-цифровий перетворювач (АЦП) на входи пристрою. Виходи пристрою підключені до цифро-аналогового перетворювача (ЦАП). Паралельно, ці аналогові сигнали множаться аналоговим блоком множення. Це дозволяє побачити похибку виконання заданої операції.

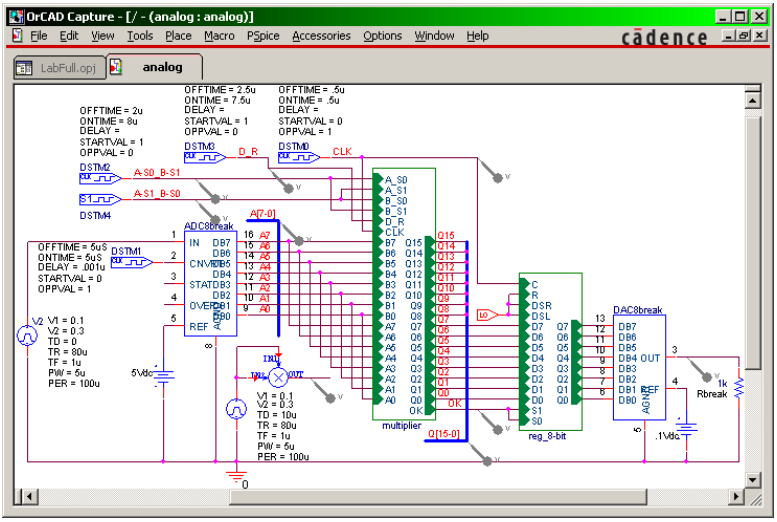


Рисунок 3.53 – Схема пристрою множення

ЦАП і АЦП знаходяться у бібліотеці *BREAKOUT* і називаються *DAC* та *ADC*. Аналоговий блок множення знаходиться в бібліотеці *ABM* і називається *MULT*. Опорна напруга ЦАП на виході цифрового помножувача розраховується за формулою:

$$V(REF) = \frac{V(OUT) \cdot 2^m}{DB},$$

де m – число двійкових розрядів, DB – цифровий код на вході, V (OUT) – необхідна напруга на виході.

Для схеми на рис. 3.55 маємо:

V (OUT) = 5 × 5 = 25 (В) – це напруга, яка виникає при множенні двох сигналів напругою 5 В:

$$V(REF) = \frac{25 \cdot 255}{254} \approx 25{,}1\,(B).$$

На рис. 3.54 ступінчастий сигнал – це результат цифрового множення, а поряд з ним плавна лінія – це результат аналогового множення. Похибкою є різниця між графіками результатів аналогового та цифрового множень.

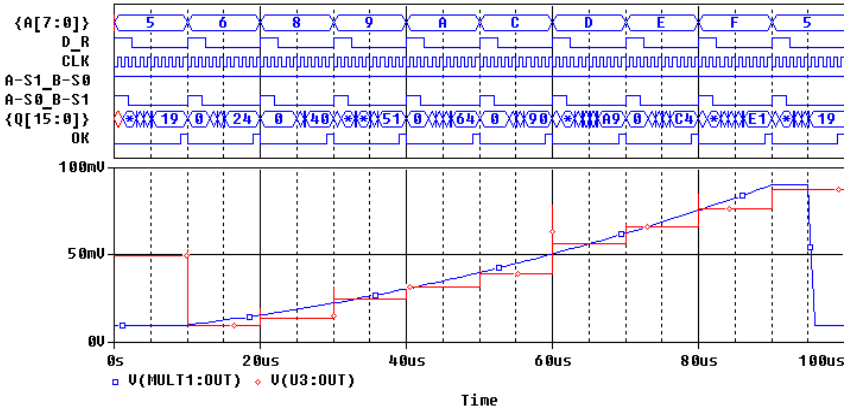


Рисунок 3.54 – Результати моделювання

Лабораторна робота 8

Використання Simulation Manager для управління процесами моделювання

Мета роботи: Розглянути можливості менеджера моделювання *Simulation Manager*.

Завдання: по черзі промоделювати дві схеми пристроїв, контролюючи значення напруги, та записати результати у файл для заданого режиму відповідно табл. 3.7.

Таблиця 3.7 – Індивідуальні завдання

№ вар.	Схема № 1		Схема № 2	
	Пристрій	Режим	Пристрій	Режим
1	5I-HI	Transient	*RS*-тригер	AC sweep
2	3-розрядний суматор	DC sweep	6-розрядний суматор	Transient
3	*D*-тригер	AC sweep	шифратор 8×3	DC sweep
4	6АБО	Transient	*D*-тригер	DC sweep
5	*JK*-тригер	AC sweep	7I-HI	Transient
6	мультиплексор 5×1	DC sweep	4-розрядний регістр	AC sweep
7	*D*-тригер	Transient	6АБО-HI	Transient
8	мультиплексор 6×1	DC sweep	2-розрядний суматор	Transient
9	*RS*-тригер	AC sweep	3АБО-HI	DC sweep
10	5АБО-HI	Transient	*D*-тригер	AC sweep
11	2-розрядний суматор	Transient	мультиплексор 3×1	AC sweep
12	4АБО-HI	DC sweep	*JK*-тригер	Transient
13	*JK*-тригер	AC sweep	*RS*-тригер	DC sweep
14	2-розрядний суматор	Transient	дешифратор на 6 виходів	DC sweep
15	*RS*-тригер	AC sweep	8АБО-HI	Transient

Приклад виконання завдання

Програма *PSpice* включає програму *Simulation Manager* (менеджер

моделювання), яка забезпечує звичний і легкий у використанні інтерфейс для управління декількома процесами моделювання. Діалогове вікно програми *PSpice Simulation Manager* наведено на рис. 3.55.

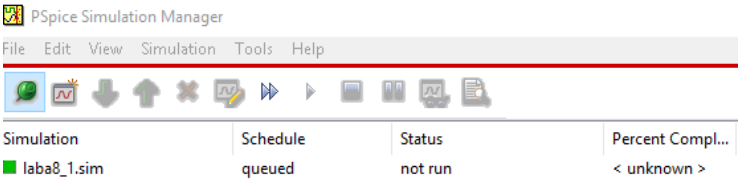


Рисунок 3.55 – Вікно програми *Simulation Manager*

Програма *Simulation Manager* дозволяє виконувати такі операції:

– змінювати порядок моделювання в черзі;

– додавати або видаляти моделювання;

– спостерігати прогрес моделювання;

– запускати, зупиняти або переривати моделювання.

Для виконання черговості в *Simulation Manager* в редакторі схем *Capture* на вкладці *Probe Window* діалогового вікна *Simulation Setting* (рис. 3.56) приберемо позначку у вікні *Display Probe Window* (ігнорувати автоматичний запуск *Probe*).

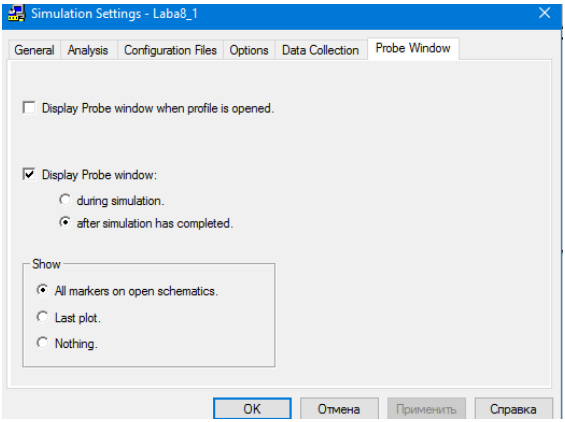


Рисунок 3.56 – Вкладка *Probe Window*

Щоб побачити результати моделювання, необхідно завантажити його вручну. Програма *Simulation Manager* запускається або з *PSpice*, або кожний раз при запуску нового моделювання, після чого вона активна протягом моделювання. Можна також викликати програму *Simulation Manager* з меню *Пуск* Windows. Кольорові піктограми в стовпці *Schedule* зліва від назви файла показують поточний стан моделювання.

У табл. 3.8 наведені пояснення до піктограм, що показують стан моделювання.

Таблиця 3.8 – Пояснення до піктограм

	Моделювання зупинене в незакінченому стані та не може бути продовжено
	Моделювання виконується в цей час
	Моделювання тимчасово призупинене та може бути продовжено
	Моделювання зупинене в незакінченому стані та не може бути продовжено

У табл. 3.9 наведені пояснення до коментарів у стовпчику *Schedule*, а у табл. 3.10 – пояснення до коментарів у стовпчику *Status.* В табл. 3.11 описані функціональні можливості *Simulation Manager* для кожної з версій *PSpice*.

Таблиця 3.9 – Пояснення до коментарів у стовпчику *Schedule*

Коментарі	Пояснення
queued	Моделювання в черзі, буде виконано в порядку, в якому воно перераховано в черзі
running	Моделювання виконується та відбувається в цей час, відображається в інформації стовпця *status*
on hold	Моделювання тимчасово припинено
stopped	Моделювання повністю виконано або було зупинене через помилку

Таблиця 3.10 – Пояснення до коментарів у стовпчику *Status*

Стан	Пояснення
Not run	Моделювання ще не було запущено (встановлення за замовчуванням)
<Status>	Основна інформація стовпця <status> про просування аналізу, яка відображається для моделювання, яке в цей час виконується
paused	Моделювання тимчасово припинено (вручну або автоматично)
Complete–no errors	Моделювання виконано до кінця та не має помилок
errors	Моделювання виконано частково та зупинене, тому що були виявлені помилки
<%>	Відсоток виконаного процесу моделювання (збільшується в міру виконання моделювання)

Таблиця 3.11 – Функціональні можливості *Simulation Manager*

Версії *PSpice*	Функціональні можливості
PSpice Demo, PSpice A/D Basics	Може бути виконано або призупинене тільки одне моделювання (черга виконання: один за одним)
PSpice, PSpice A/D	Одне моделювання може бути виконане та безліч моделювань може бути призупинене (черга виконання: один за одним)

Розглянемо, як програма *Simulation Manager* ідентифікує помилки в процесі моделювання.

Аварійне моделювання – якщо моделювання неможливе з якої-небудь причини, *Simulation Manager* буде зупиняти розвиток змін. Потім, після певного періоду часу, *Simulation Manager* виконає зупинку моделювання та автоматично запустить наступну задачу з черги.

Тимчасове припинення моделювання – якщо тимчасове припинення моделювання відбудеться автоматично і потребує ручного втручання для продовження, *Simulation Manager* автоматично запустить наступну задачу з черги.

Пауза в моделюванні – якщо моделювання зупинене автоматично, *Simulation Manager* буде автоматично запускати наступну задачу з черги.

У *Simulation Manager* можна встановлювати будь-яке число серій моделювань, які виконуються один за одним. В той же час, можна виконувати іншу задачу в *PSpice*. Для кожного нового моделювання буде призначена категорія *queued*. Це моделювання буде виконано після виконання всіх інших задач у черзі. Якщо задача додана, можна змінити її положення в черзі: запустити, зупинити, тимчасово припинити або зробити інші зміни в її статусі.

Для додавання моделювання в чергу:

1. Вибрати *File – Add Simulation* або натиснути кнопку *Add Simulation* на панелі інструментів.

2. У вікні *Add Simulation* двічі клацнути по файлу з розширенням *.SIM або *.CIR.

У *Simulation Manager* можна легко управляти серіями моделювань в черзі. Для запуску моделювання з *Simulation Manager*:

1. Виберемо моделювання зі списку.

2. В меню *Simulation* виберемо *Run* або натиснемо кнопку *Run Selected* на панелі інструментів.

Для зупинки моделювання в *Simulation Manager*:

1. Виберемо моделювання, яке виконується в цей час.

2. У меню *Simulation* виберемо *Stop* або натиснемо кнопку *Stop Selected* на панелі інструментів.

Для тимчасового припинення моделювання з *Simulation Manager*:

1. Виберемо моделювання, яке в цей час виконується.

2. У меню *Simulation* виберемо *Pause* або натиснемо кнопку *Pause Selected* на панелі інструментів.

Якщо для задач, які обрані зі списку в *Simulation Manager*, необхідно відобразити результати моделювання, то потрібно запустити моделювання в *PSpice.*

Для запуску моделювання в *PSpice*:

1. Виберемо моделювання, яке потрібне запустити в *PSpice.*

2. У меню *View* виберемо *Simulation Results.*

Програма *PSpice* активізується і результати обраного моделювання відображаються в *Probe*. Якщо в цей момент виконується моделювання,

то можна спостерігати зміну сигналів у вікні графіків. Кожен раз при додаванні нового моделювання, в процесі виконання попереднього, *Simulation Manager* повинен вирішити, як поводитися з новою задачею. Згідно з налаштуванням за замовчуванням нове моделювання додається в кінець черги, а виконання задачі, яка моделюється в цей час, триває.

Можна змінювати встановлення за замовчуванням таким чином, що *Simulation Manager* буде запускатися при кожному новому моделюванні негайно та зупиняти або тимчасово припиняти будь-яку задачу поточного виконання.

Опції можна вибирати згідно з роз'ясненнями, наведеними нижче (діалогове вікно *Options* можна відображати кожен раз, коли додаються нові моделювання, чи не показувати його). Для встановлень за замовчуванням для *Simulation Manager* в меню *Tools* виберемо *Options* – з'явиться діалогове вікно (рис. 3.57).

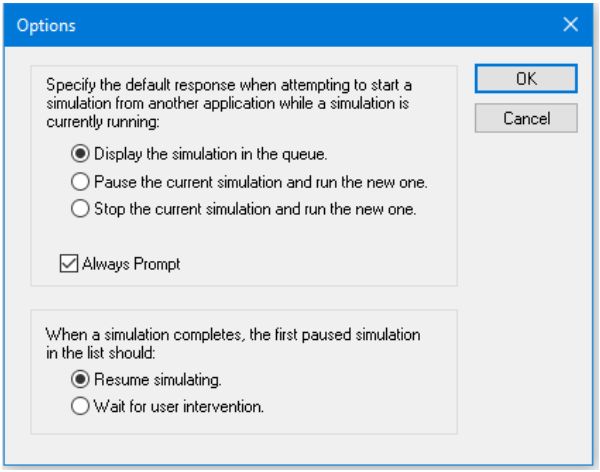


Рисунок 3.57 – Діалогове вікно *Options*

Якщо необхідно, діалогове вікно *Options* буде з'являтися кожен раз, коли додається моделювання – для цього повинна бути встановлена позначка в *Always Prompt* (встановлення за замовчуванням). Опис деяких команд програми *Simulation Manager* наведено в табл. 3.12.

Таблиця 3.12 – Опис команд *Simulation Manager*

Команда	Призначення
Меню File	
Add Simulation	Відкриття файлів моделювання (.SIM) або файлів схем (.CIR)
Меню View	
Simulation Results	Відображення результатів моделювання в *PSpice*
Output File	Відкриття вихідного файлу для обраного моделювання та відображення його в *PSpice*
Меню Simulation	
Run Queued Items	Виконання всіх моделювань в черзі
Queue Selected	Зміна всіх обраних моделювань
Reset Queue	Скидання всіх моделювань
Edit Settings	Відкриття профайла для обраного моделювання (для зміни встановлень аналізу)
Меню Tools	
Options	Дозволяє змінювати встановлення за замовчуванням

Збереження проєкту, схеми або бібліотеки

Редактор схем *Capture* забезпечує функціональні можливості, які дозволяють легко зберігати та архівувати проєкти на кожній стадії розвитку. Команда *Save* зберігає всі відкриті документи, які посилаються на проєкт. Коли зберігають проєкти разом зі схемами і бібліотеками, відповідні властивості зберігаються, тому міняти позиційні позначення немає необхідності.

Команда *Save As* зберігає файли залежно від вибору в *Project manager*:

– якщо обрана одна або кілька схем (або бібліотек), можна зберегти кожен файл зі змінами;

– якщо в *Project manager* не вибрані схеми та бібліотеки, можна швидко зберегти проєкт.

Коли зберігається проєкт, *Capture* автоматично створює резервний файл з розширенням *.DBK. Коли зберігається бібліотека, *Capture* автоматично створює резервний файл з розширенням *.OBK. Якщо

зберігається схематична сторінка або символ, резервний файл не створюється. В *Capture* файл схеми (*.DSN) завжди супроводжується файлом проєкту (*.OPJ).

Архівування за допомогою редактора схем *Capture*

Можна зберегти проєкт (*.OPJ) та всі пов'язані з ним файли в різних каталогах, а також створити zip-файл. Можна також архівувати зовнішні проєкти, глобальні бібліотеки моделей *PSpice* та глобальні файли, що включаються в проєкт *PSpice* або довідкові дані для компонентів. Можна використовувати команду *Archive Project* в меню *File* для архівування проєкту. Ця команда дозволяє зберігати всі файли, які пов'язані з проєктом, в каталозі, визначеному для архівування та в одиночному файлі *.zip.

Для архівації проєкту:

1. Переконайтеся, що проєкт, який необхідно заархівувати, активний і схематичні сторінки цього проєкту не відкриті.

2. У меню *File* виберемо *Archive Project*. З'явиться діалогове вікно *Archive Project* (рис. 3.58).

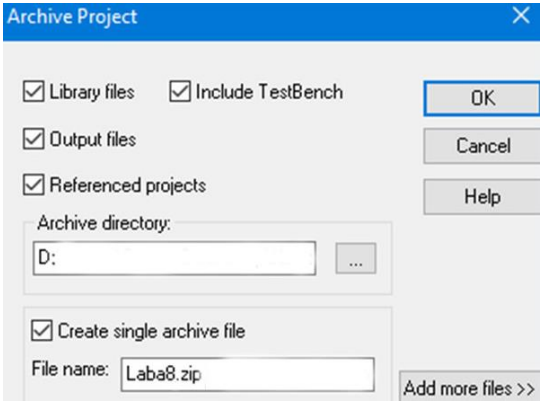


Рисунок 3.58 – Діалогове вікно *Archive Project*

3. Виберемо файли, які необхідно помістити в проєкті. Якщо не вибрані опції (*Library files*, *Output files* або *Referenced projects*), *Capture* за

замовчуванням архівує тільки файли проєкту (*.OPJ) та схеми (*.DSN). Опція *Include TestBench* складає звіт про всі файли і симуляції. Для проєктів *PSpice* профайли моделювання та локальні файли (*.LIB, *.STL, *.INC) будуть завжди архівувати разом з проєктом. Опція *Output files* архівує також вихідні файли для *PSpice*-проєктів, такі, як *.DAT та *.OUT.

4. Виберемо каталог, в якому необхідно зберегти файли. З'явиться діалогове вікно *Select Directory* (можна також ввести шлях до каталога архіву в текстове поле *Archive directory*).

5. Виберемо каталог, в якому потрібно архівувати проєкт, та натиснемо *OK*.

6. Натиснемо *OK* в діалоговому вікні *Archive Project*. *Capture* архівує проєкт з вибраними файлами в зазначеному каталозі та відобразить повідомлення про помилки в *Session Log*. При створенні нового каталога архіву поточний каталог не змінюється. Установки, які визначаються в діалоговому вікні *Archive Project*, зберігаються у файлі CAPTURE.INI (ці установки будуть використовуватися, коли запуститься наступна сесія архівування).

Інші вихідні опції

У *Capture* використовується спеціальний символ WATCH1, який дозволяє контролювати значення напруги для трьох режимів схеми: *DC sweep, AC sweep* або *Transient*. Результати відображаються в *PSpice.*

Для відображення значень напруги у вікні *PSpice*:

1. Помістимо та приєднаємо символ WATCH1 (з *PSpice* бібліотеки SPECIAL.OLB) до аналогової схеми (рис. 3.59).

2. Двічі клацнемо на зразок символу WATCH1 для відображення електронної таблиці *Parts*.

3. У стовпці властивостей ANALYSIS введемо DC, AC або TRAN (*transient*) для типу аналізу, результати якого необхідно спостерігати.

4. У стовпцях властивостей LQ та HI введемо значення відповідно до нижньої та верхньої меж напруги для даної точки схеми. Якщо результати переміщуються за певні межі, *PSpice* тимчасово припиняє моделювання, щоб можна було оцінити поведінку.

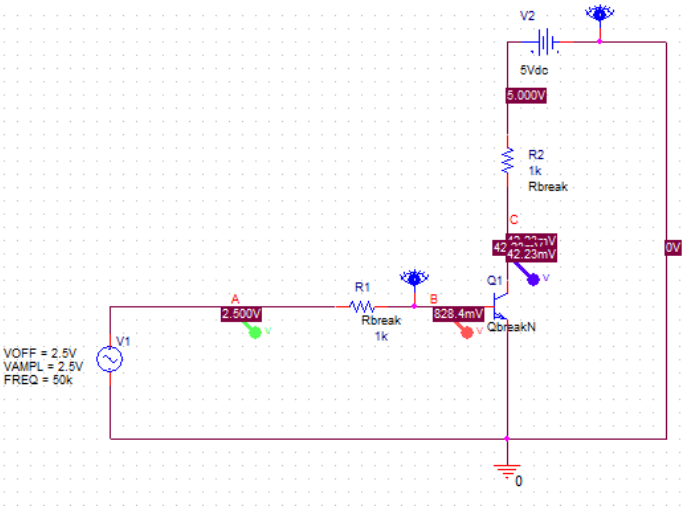


Рисунок 3.59 – Схема дослідження

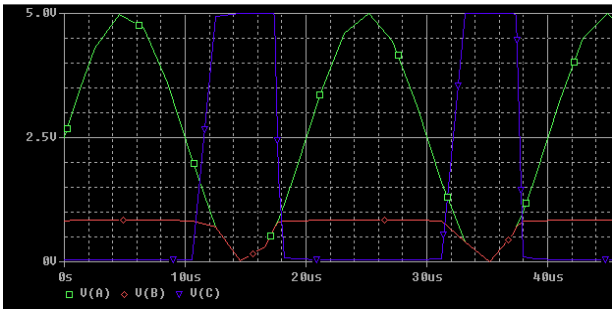


Рисунок 3.60 – Результати моделювання

Запис додаткових результатів у вихідний файл PSpice

Capture має додаткові символи, які дозволяють зберігати додаткові результати моделювання у вихідному файлі *PSpice* для роздруківки графіків та таблиць.

Для відображення вихідного файлу *PSpice* після виконання моделювання в меню *Simulation* виберемо *Examine Output*. Можна згенерувати графіки напруги та струму для будь-яких видів аналізу: DC sweep, AC sweep або Transient.

Для генерації графіків напруги або струму у вихідному файлі виконаємо:

1. Помістимо та приєднаємо будь-які символи з табл. 3.13 (з *PSpice* – бібліотеки SPECIAL.OLB).

Таблиця 3.13 – Символи для відображення графіків

Символ	Графік відображає
	Напруга на ділянці кола, до якої приєднано символ
	Диференціальна напруга між двома колами, до яких приєднано символ
	Струм

2. Двічі клацнемо на зразок символу для відображення електронної таблиці *Parts.*

3. Виберемо тип аналізу, який хочемо відобразити: DC, AC або TRAN.

4. У стовпці *Analysis type*, який потрібно відобразити (DC, AC або TRAN), введемо будь-які значення (Y, YES або «1»).

5. Якщо було вибрано тип аналізу AC, визначимо вихідний формат:

5.1 Виберемо один з таких вихідних форматів: MAG (амплітуда), PHASE, REAL, IMAG (уявне) або DB.

5.2 Введемо будь-які значення (Y, YES або «1»).

5.3 Повторимо п. 5.1 – 5.2 для інших обраних вихідних форматів аналізу AC. Якщо формат знято, PSpice за замовчуванням вибере MAG.

6. Повторимо п. 2 – 4 для інших типів аналізу, які необхідно відобразити.

Для генерації таблиці зміни цифрового стану у вихідному файлі помістимо символ PRINTDGTLCHG (з *PSpice* бібліотеки SPECIAL.OLB) та приєднаємо його до кола.

У *Capture* є символ VECTOR, який дозволяє зберігати результати цифрового моделювання у векторному файлі (коли коло з символом VECTOR змінює стан, дані записуються у векторний файл).

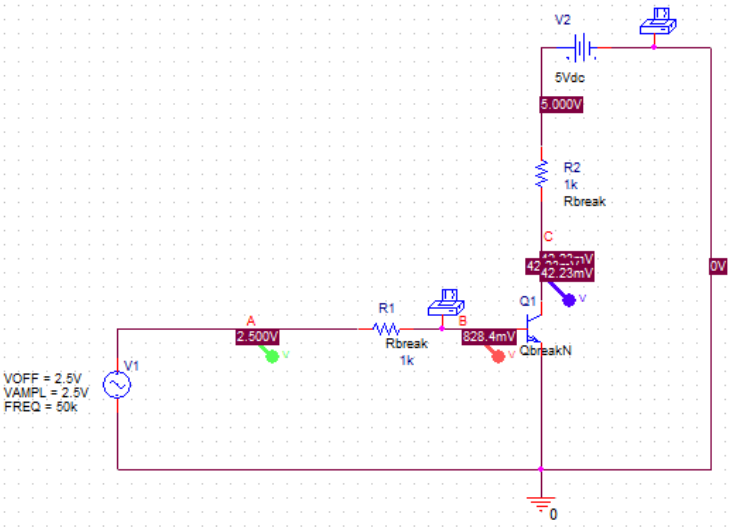


Рисунок 3.61 – Встановлення символу VECTOR

Для створення файлу тестового вектора схеми необхідно виконати:

1. Символ VECTOR приєднаємо до провідника (шини) на виході зразка цифрового символу.

2. Двічі клацнемо на VECTOR для відображення електронної таблиці *Parts*.

3. Встановимо властивості символу, як описано в табл. 3.14.

Таблиця 3.14 – Шрифт VECTOR

Властивість	Визначення
POS	Позиція стовпчика у файлі (чинний діапазон значень 1...25)
FILE	Назва векторного файлу
RADIX	Основа системи числення
BIT	Коли символ VECTOR приєднується до провідника, позиція біта серед шести або восьми одиночних цифр
SIGNAMES	Назви сигналів, які з'являються в заголовку файлу

4. Повторити п. 1 – 3 для числових тестових векторів, які потрібно створити.

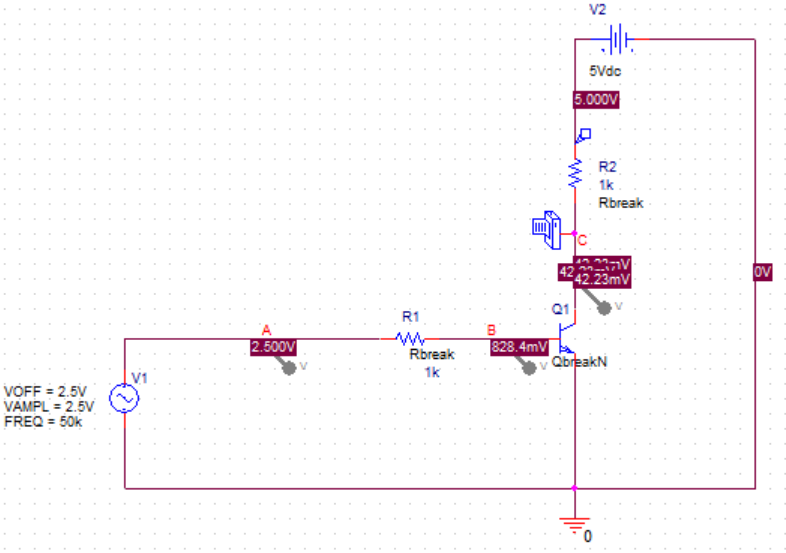


Рисунок 3.62 – Різні встановлення символу VECTOR

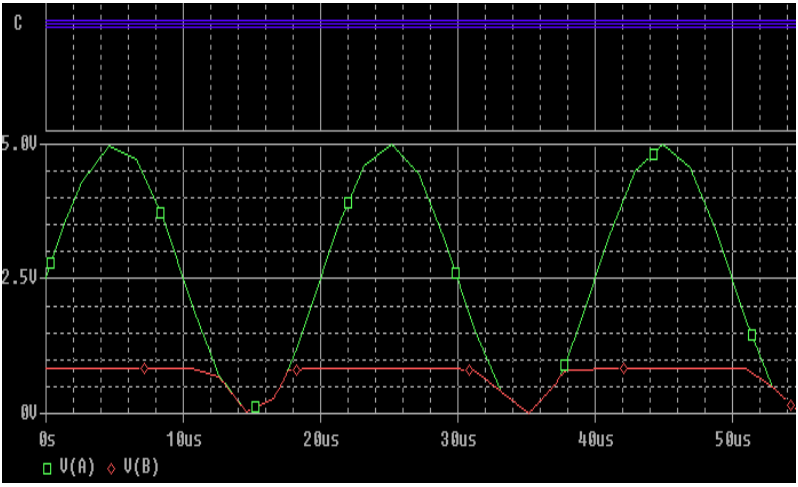


Рисунок 3.63 – Результати моделювання

Список літератури

1. Леонов С.Ю., Гладких Т.В. Автоматизоване проектування пристроїв в комп‘ютерних системах: навчальний посібник. – 2-е вид. випр. та доп. – Харків: НТУ "ХПІ", 2024. – 283 с.

2. Ларін В.Ю., Харченко В.П. Автоматизація схемотехнічного проектування: підручник / Київ: НАУ, 2017. – 190 с.

3. Макаренко В.В. Програмні засоби проектування: навчальний посібник / Київ: КПІ ім. Ігоря Сікорського, 2018. – 244 с.

4. Orcad Capture. User’s Guide / Cadence Design Systems, 2000. – 374 p.

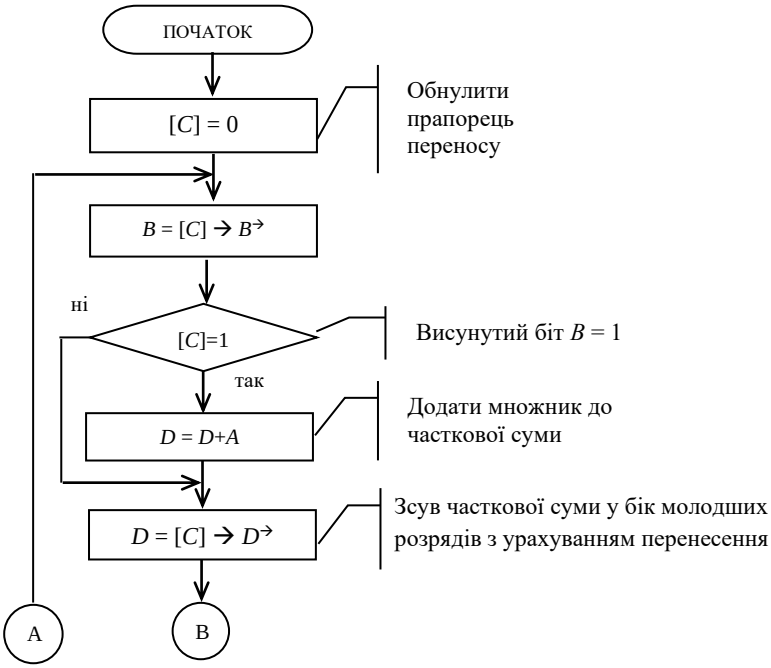
5. OrCAD Flow Tutorial / Cadence Design Systems, 2004. – 106 p.

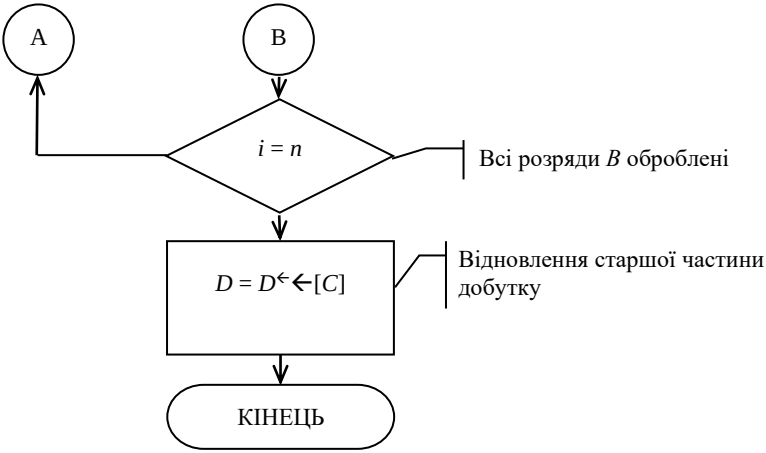
6. Kraig Mitzner Complete PCB Design Using OrCad Capture and Layout / Elsevier Inc., 2007. – 529 p.

ДОДАТОК

Алгоритм множення 1

Реалізація команди множення чисел проводиться шляхом покрокового складання значення множника, що зсувається, і часткової суми, за наявності ненульового біта відповідного розряду множника. Даний алгоритм характеризується тим, що для множника та часткової суми використовуються регістри однієї величини. Результат (добуток) при цьому зберігається в парі, яка відповідає частковій сумі та множника, яка поступово заповнюється нулями, причому старша частина зберігається в регістрі часткової суми, а молодша – у регістрі множника. Розглянемо алгоритм, в якому A і B – множники, D – часткова сума.





Приклад

$A = 6_{10} = 0110_2$ (множник); $B = 3_{10} = 0011_2$ (множник);

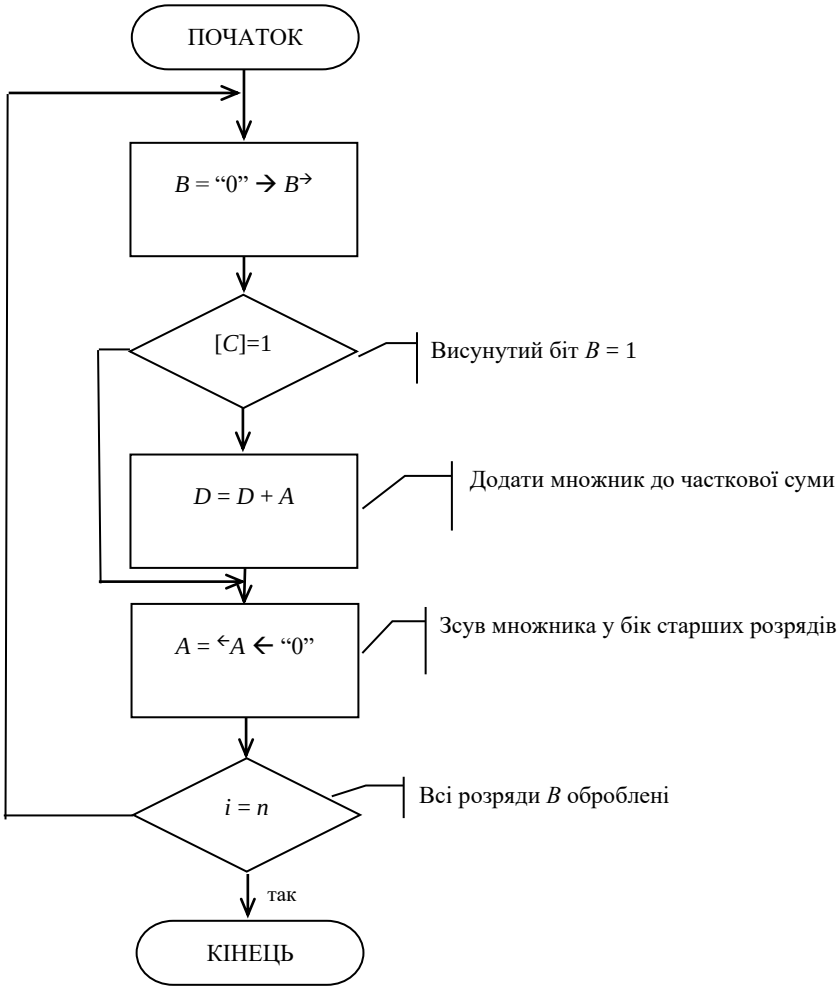
$D = 0000_2$ (часткова сума)

№ такту	**Дія**		**Прапорець [*C*]**
1	Зсув *B* праворуч	*B* = **[0]001**|1	1
2	[*C*] = 1, наступн. додаємо множник до часткової суми	*D* = *D* + *A* 0000 + 0110 0|0110	0
3	Зсув *D* праворуч	*D* = **[0]011**|0	0
4	Зсув *B* праворуч	*B* = **[0]000**|1	1
5	[*C*] = 1, наступн. додаємо множник до часткової суми	*D* = *D* + *A* 0011 + 0110 0|1001	0
6	Зсув *D* праворуч	*D* = **[0]100**|1	1
7	Зсув *B* праворуч	*B* = **[1]000**|0	0
8	Зсув *D* праворуч	*D* = **[0]010**|0	0
9	Зсув *B* праворуч	*B* = **[0]100**|0	0
10	Зсув *D* праворуч	*D* = **[0]001**|0	0
11	Зсув *B* праворуч	*B* = **[0]010**|0	0
12	Зсув *D* праворуч	*D* = **[0]000**|1	1
13	Зсув *D* ліворуч	*D* = 0|**000[1]**	0

Результат у *D*:*B* : 0001:0010. Бачимо, що $00010010_2 = 18_{10}$.

Алгоритм множення 2

Алгоритм характеризується тим, що для першого множника та часткової суми використовуються регістри подвійної розрядності, а для другого множника – одинарної, який дорівнює N. Результат зберігається в регістрі часткової суми. Розглянемо алгоритм, в якому A і B – множники, D – часткова сума.



Приклад

$A = 6_{10} = 00000110_2$ (множник); $B = 3_{10} = 0011_2$ (множник);

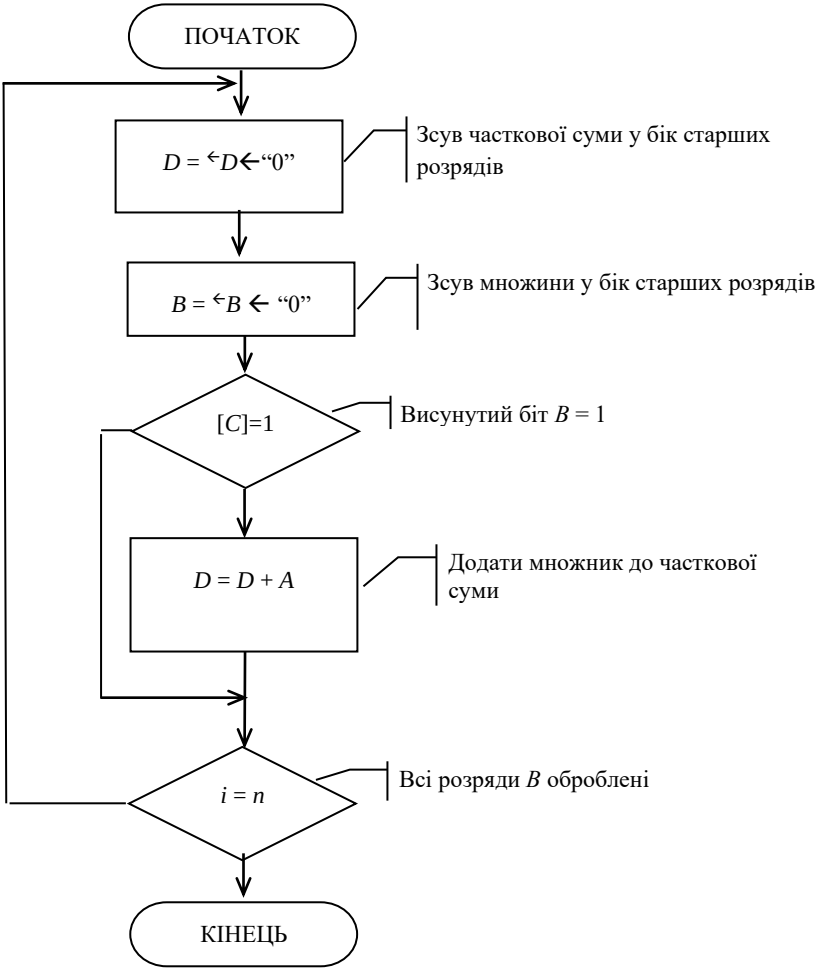
$D = 00000000_2$ (часткова сума)

№ такту	**Дія**		**Прапорець [*C*]**
1	Зсув *B* праворуч	*B* = **0001**|1	1
2	[*C*] = 1, наступн. додаємо множник до часткової суми	*D* = *D* + *A* 00000000 + 00000110 0| 00000110	0
3	Зсув *A* ліворуч	A = 0|**00001100**	0
4	Зсув *B* праворуч	*B* = **0000**|1	1
5	[*C*] = 1, наступн. додаємо множник до часткової суми	*D* = *D* + *A* 00000110 + 00001100 0|00010010	0
6	Зсув *A* ліворуч	*A* = 0|**00011000**	0
7	Зсув *B* праворуч	*B* = **0000**|0	0
8	Зсув *A* ліворуч	*A* = 0|**00110000**	0
9	Зсув *B* праворуч	*B* = **0000**|0	0
10	Зсув *A* ліворуч	*A* = 0|**01100000**	0

Результат у *D*: 00010010. Бачимо, що $00010010_2 = 18_{10}$.

Алгоритм множення 3

Алгоритм характеризується тим, що для часткової суми використовуються регістр подвійної розрядності, а для множників – одинарної, що дорівнює N. Результат зберігається в регістрі часткової суми. Розглянемо алгоритм, в якому A і B – множники, D – часткова сума.



Приклад

$A = 6_{10} = 0110_2$ (множник); $B = 3_{10} = 0011_2$ (множник);

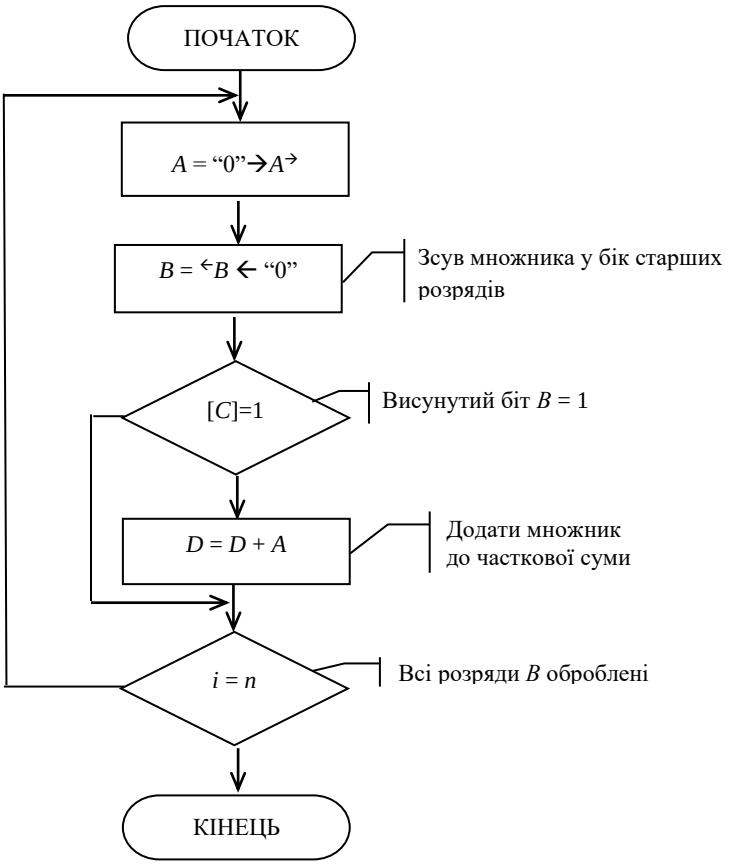
$D = 00000000_2$ (часткова сума).

№ такту	**Дія**		**Прапорець [*C*]**
1	Зсув *D* ліворуч	*D* = 0|**00000000**	0
2	Зсув *B* ліворуч	*B* = 0|**0110**	0
3	Зсув *D* ліворуч	*D* = 0|**00000000**	0
4	Зсув *B* ліворуч	*B* = 0|**1100**	0
5	Зсув *D* ліворуч	*D* = 0|**00000000**	0
6	Зсув *B* ліворуч	*B* = 1|**1000**	1
7	[*C*] = 1, наступн. додаємо множник до часткової суми	*D* = *D* + *A* 00000000 + 00000110 0| 00000110	0
8	Зсув *D* ліворуч	*D* = 0|**00001100**	0
9	Зсув *B* ліворуч	*B* = 1|**0000**	1
10	[*C*] = 1, наступн. додаємо множник до часткової суми	*D* = *D* + *A* 00001100 + 00000110 0|00010010	0

Результат у *D*: 00010010. Бачимо, що $00010010_2 = 18_{10}$.

Алгоритм множення 4

Алгоритм характеризується тим, що для часткової суми і регістру першого множника використовуються регістри подвійної розрядності, а для другого множника – одинарної, яка дорівнює N. Крім цього, значення першого множника записується в старшу частину регістра, щоб проводити його зсув в сторону молодших розрядів, поступово зменшуючи його значення. Аналіз розрядів множника ведеться, починаючи зі старших. Результат зберігається у регістрі часткової суми. Розглянемо алгоритм, в якому A і B – множники, D – часткова сума.



Приклад

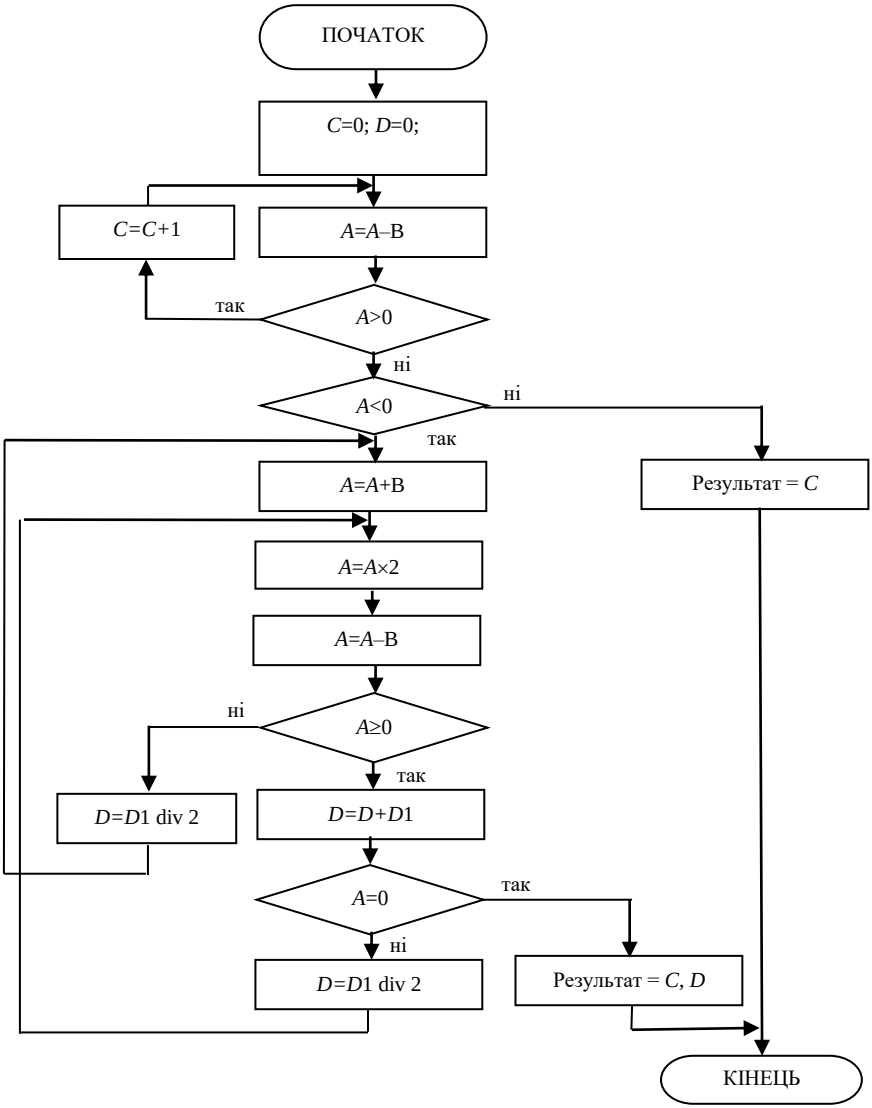
$A = 01100000_2$; $B = 0011_2$;

$D = 00000000_2$ (часткова сума).

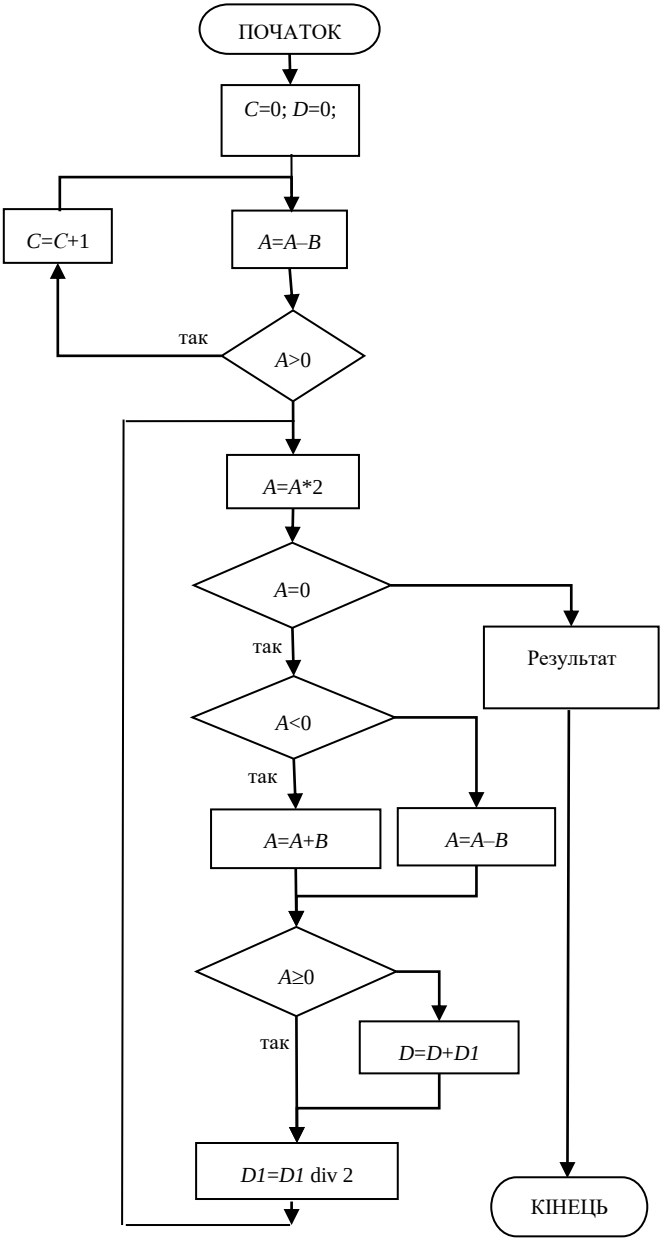
№ такту	**Дія**		**Прапорець [*C*]**
1	Зсув *A* праворуч	*D* = **00110000**|0	0
2	Зсув *B* ліворуч	*B* = 0|**0110**	0
3	Зсув *A* праворуч	*D* = **00011000**|0	0
4	Зсув *B* ліворуч	*B* = 0|**1100**	0
5	Зсув *A* праворуч	*A* = **00001100**|0	0
6	Зсув *B* ліворуч	*B* = 1|**1000**	1
7	[C] = 1, наступн. додаємо множник до часткової суми	*D* = *D* + *A* 00000000 + 00001100 0| 00001100	0
8	Зсув *A* праворуч	*D* = **00000110**|0	0
9	Зсув *B* ліворуч	*B* = 1|**0000**	1
10	[*C*] = 1, наступн. додаємо множник до часткової суми	*D* = *D* + *A* 00001100 + 00000110 0|00010010	0

Результат у D: 00010010. Бачимо, що $00010010_2 = 18_{10}$.

Алгоритм ділення (з відновленням залишку)



Алгоритм ділення (без відновлення залишку)



ЗМІСТ

Навчальне видання

Навчальний посібник
для студентів спеціальності «Комп'ютерна інженерія»

ТЕХНОЛОГІЯ АВТОМАТИЗОВАНОГО ПРОЄКТУВАННЯ КОМП'ЮТЕРНИХ СИСТЕМ ТА ЇХ СКЛАДОВИХ

Укладачі:
ЛЕОНОВ Сергій Юрійович
ГЕЙКО Геннадій Вікторович

Відповідальний за випуск проф. Олександр ЗАКОВОРОТНИЙ
Роботу до видання рекомендував проф. Микола ЗАПОЛОВСЬКИЙ

В авторській редакції

План 2024 р., поз. 85.
Підп. до друку 15.07.2024. Формат 60x84 1/16.
Папір офсет. Друк ризографічний. Ум. друк. арк. 5,1.

НТУ "ХПІ", 61002, Харків, вул. Кірпічова, 2,
Видавничий центр НТУ "ХПІ"
Свідоцтво ДК № 116 від 10.07.2000 р.

Виготовлено у ТОВ ВПП "Контраст".
Україна, 61166, м. Харків, пр. Науки, 40, оф. 221.
Св-во: ДК №1778 від 05.05.2004.