МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ПРИКАРПАТСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАСИЛЯ СТЕФАНИКА

Кафедра комп'ютерної інженерії та електроніки

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни
"Комп'ютерні технології в наукових дослідженнях"
для студентів спеціальності 171 "Електроніка"

Затверджено на засіданні
кафедри комп'ютерної інженерії та
електроніки
протокол № _ від _____ 2019 р.
та методичною радою
фізико-технічного факультету
протокол № _ від _____ 2019 р.

Івано-Франківськ
2019

Методичні вказівки до виконання лабораторних робіт з дисципліни “Комп’ютерні технології в наукових дослідженнях” для студентів спеціальності 171 “Електроніка” / Укл. Грига В.М. – Івано-Франківськ, Прикарпатський національний університет, ВДВ ЦІТ, 2019. – 112 с.

Укладач: Грига В.М., к.т.н., доцент

Рецензенти: Дунець Р.Б., д. т. н., проф., завідувач кафедрою спеціалізованих комп’ютерних систем Національного університету “Львівська політехніка”;

Николайчук Я.М., д. т.н., проф., завідувач кафедрою спеціалізованих комп’ютерних систем Тернопільського національного економічного університету.

ЗМІСТ

ЛАБОРАТОРНА РОБОТА №1

Тема роботи: Технології моделювання та дослідження роботи D-тригерів в середовищі System C.

Мета роботи: засвоїти навики роботи в середовищі SystemC_Win1.0. Навчитися проектувати і моделювати роботу D-тригера з використанням бібліотек System C.

Теоретичні відомості

SystemC – це бібліотека класів C++, яка виникла у відповідь на все більш зростаючі потреби у мові, яка б збільшувала продуктивність роботи розробників електронних систем. Сучасні системи зазвичай містять в своєму складі програмну та апаратну частини, що розробляються у вкрай жорстких часових рамках та вимогах до продуктивності, що вимагає всебічної функціональної перевірки розробленої системи для запобігання виникненню помилок.

SystemC пропонує значний виграш у продуктивності завдяки поєднанню процесів проектування програмної та апаратної частин системи на вищому рівні абстракції. Цей рівень надає розробнику можливість вже на початку проектування побачити структуру системи в цілому, таким чином легше уявляючи процес взаємодії між її окремими частинами, виявити «проблемні місця», провести верифікацію на більш ранніх етапах проектування. Власне кажучи, *SystemC* не є тією «панацеєю», що обіцяє вирішити усі проблеми розробників. Однак, при появі цієї бібліотеки разом з бібліотекою тестування *SystemC Verification Library* в них поєдналися багато характеристик, які були повністю відсутні чи слабо представлені в інших мовах проектування.

Характеристики бібліотеки SystemC

SystemC є засобом сумісного проектування апаратури та програмного забезпечення, а також опису архітектури складних систем завдяки наступним характеристикам:

- *Модулі* – у *SystemC* є поняття контейнеру класу, що має назву «модуль». Це є ієрархічна структура, що може містити інші модулі та (чи) процеси.

- *Процеси* – процеси використовуються для функціонального опису системи. Процеси входять до складу модулів. SystemC підтримує три різні абстракції процесів для використання розробниками ПЗ та апаратури.

- *Порти* – модулі містять порти, що використовуються для з'єднання з іншими модулями. *SystemC* підтримує одно – та двонаправлені порти.

- *Сигнали* – *SystemC* підтримує як складні, так і прості сигнали. Складні сигнали можуть мати більш ніж одне джерело.

- *Великий набір типів даних* - *SystemC* має великий набір типів даних для підтримки багатьох областей проектування та рівнів абстракції. Типи даних з фіксованою точністю використовуються для швидкої симуляції. Типи даних з довільною точністю використовуються для операцій з великими значеннями.

- *Сигнали синхронізації* – *SystemC* має спеціальний набір сигналів синхронізації.

D-тригер

Тригером називають послідовнісну схему з позитивним зворотнім зв'язком і двома стійкими станами 0 і 1 (тобто тригер володіє властивістю пам'яті). В загальному випадку тригер може мати асинхронні входи попередньої установки, тактовий (синхронізуючий), та інформаційні входи. До основних типів тригерів відносяться:

- тригер з роздільною установкою станів (RS-трігер),

- D - тригер,

- універсальний тригер (JK - тригер),

- тригер з рахунковим входом (T - тригер).

D-тригер має два входи: інформаційний вхід D(ata) і вхід управління записом/запам'ятовуванням L(oad)/L(atch) - звідси його друга назва: "защіпка" (рис. 1). Останній вхід часто позначають символом C (Clock). Вихідний сигнал Q приймає значення, рівне вхідному D, при C = 1 і зберігає попереднє значення Q(t+dt)= Q(t) при C = 0.

Таблиця 1

Поточний стан			Наступний стан		Назва режиму
C	D	*Q(t)*	*Q(t+dt)*	$\overline{Q}$ (t+dt)	
0	X	Q	Q	~Q	Зберігання
1	0	X	0	1	Установка в "0" (скидання)
1	1	X	1	0	Установка в "1" (установка)

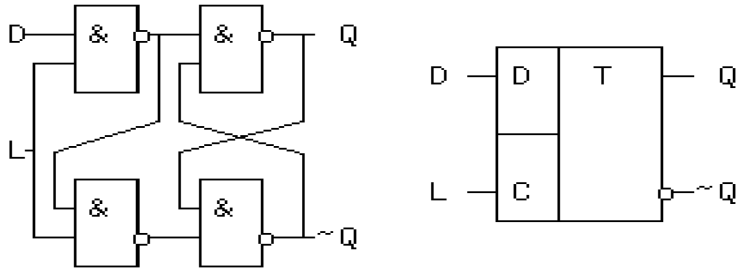


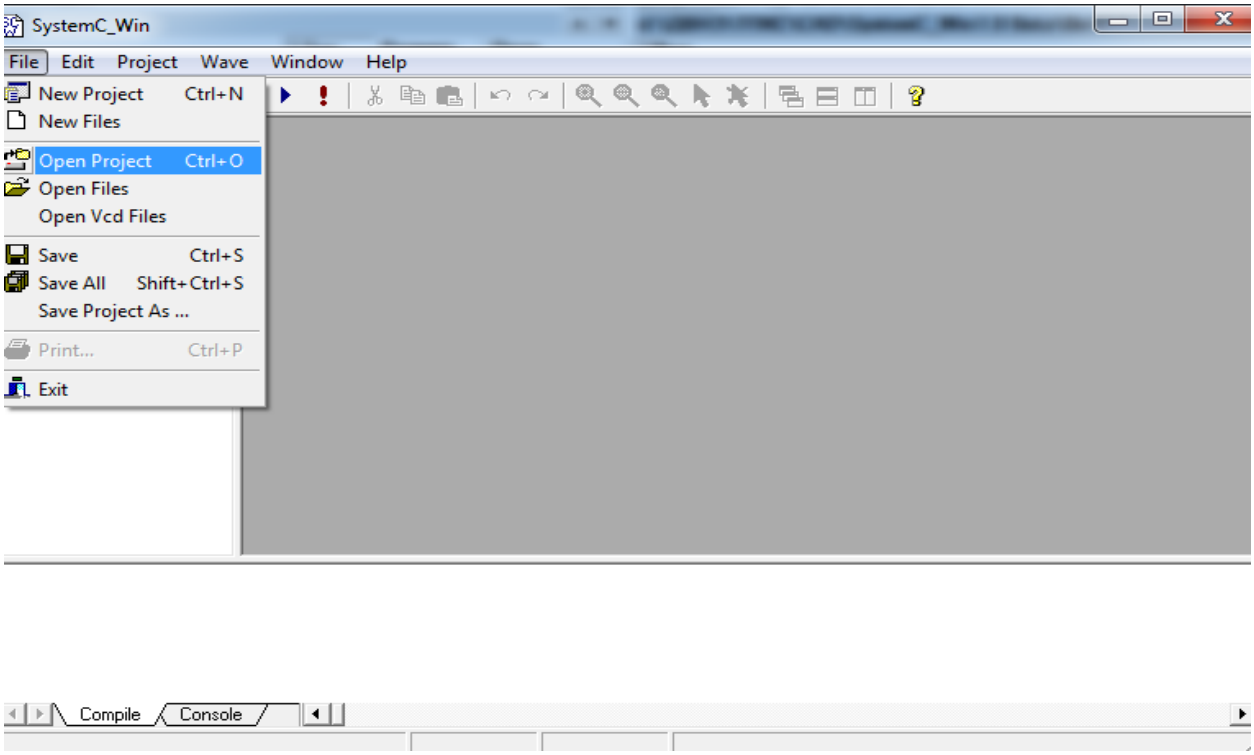
Рис. 1. D-тригер

Рівняння D-тригера: Q(t+dt) = ~(~(C*D) * ~(Qt*~(~(D*C) * C))).

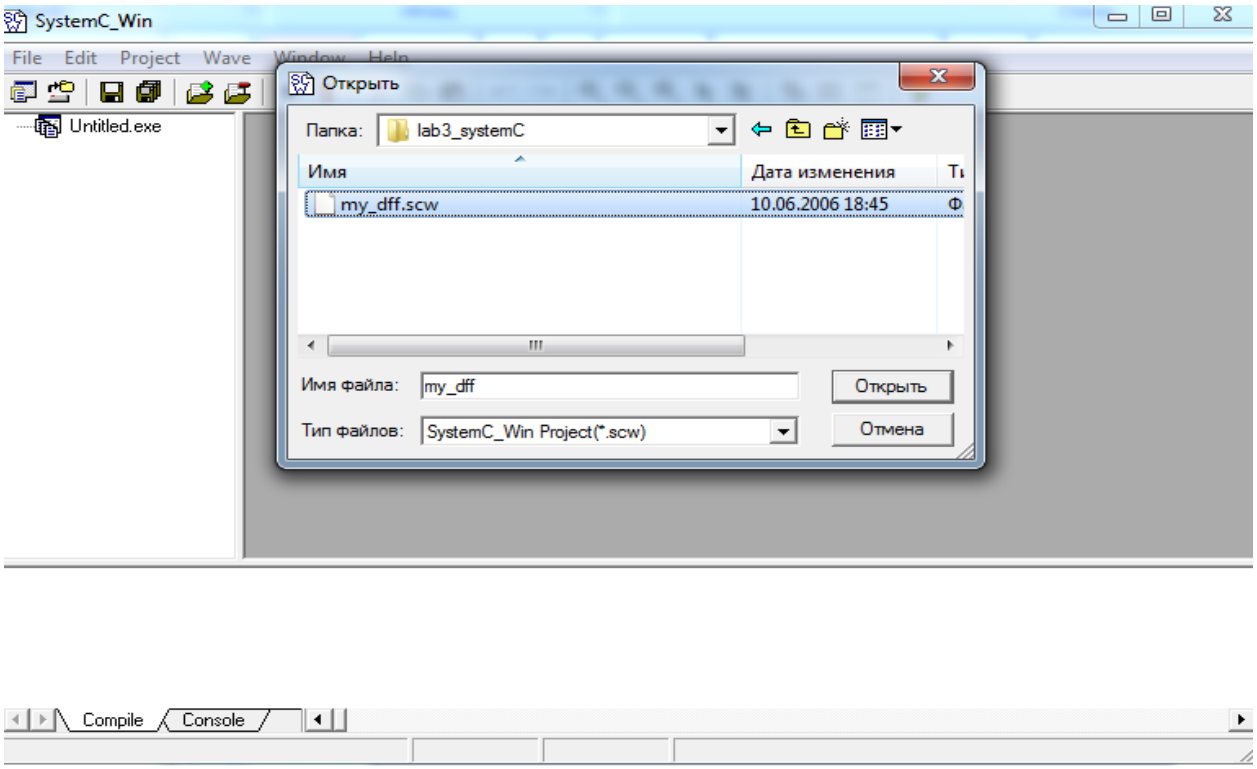
Хід виконання роботи

1. Для початку роботи з бібліотекою класів *SystemC* потрібно виконати інсталяцію Borland CPP компілятора. В якості компілятора можна використати безкоштовний C++ компілятор від фірми Borland (BCPP_freeline). Після розархівування файлів компілятора запускаємо на виконання файл SystemC_Win.exe з директорії *SystemC_Win1.0 Beta\Bin*. При першому запуску система видає повідомлення про необхідність вказати шлях до C++ компілятора, що і потрібно зробити, вказавши шлях до директорії, куди було розархівовано компілятор вибравши директорію Borland і папку BCC55.

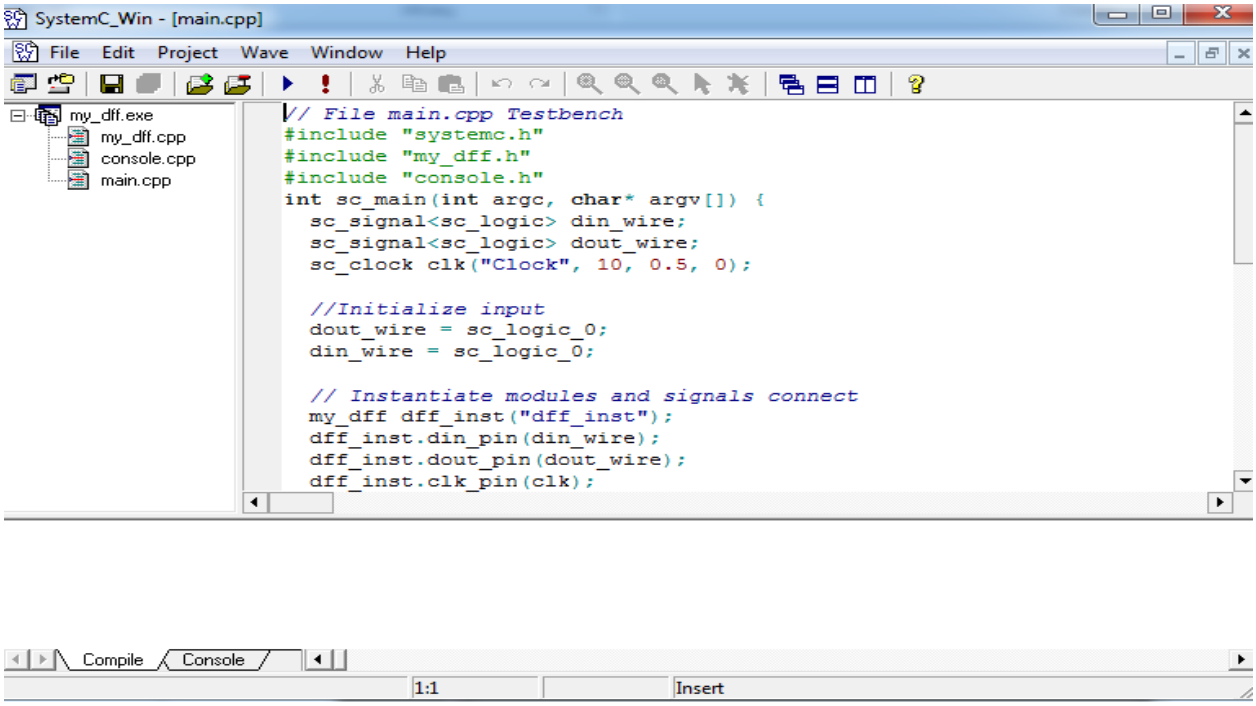
2. У меню *File\Open Project* вибираємо готовий проект lab1_SystemC.



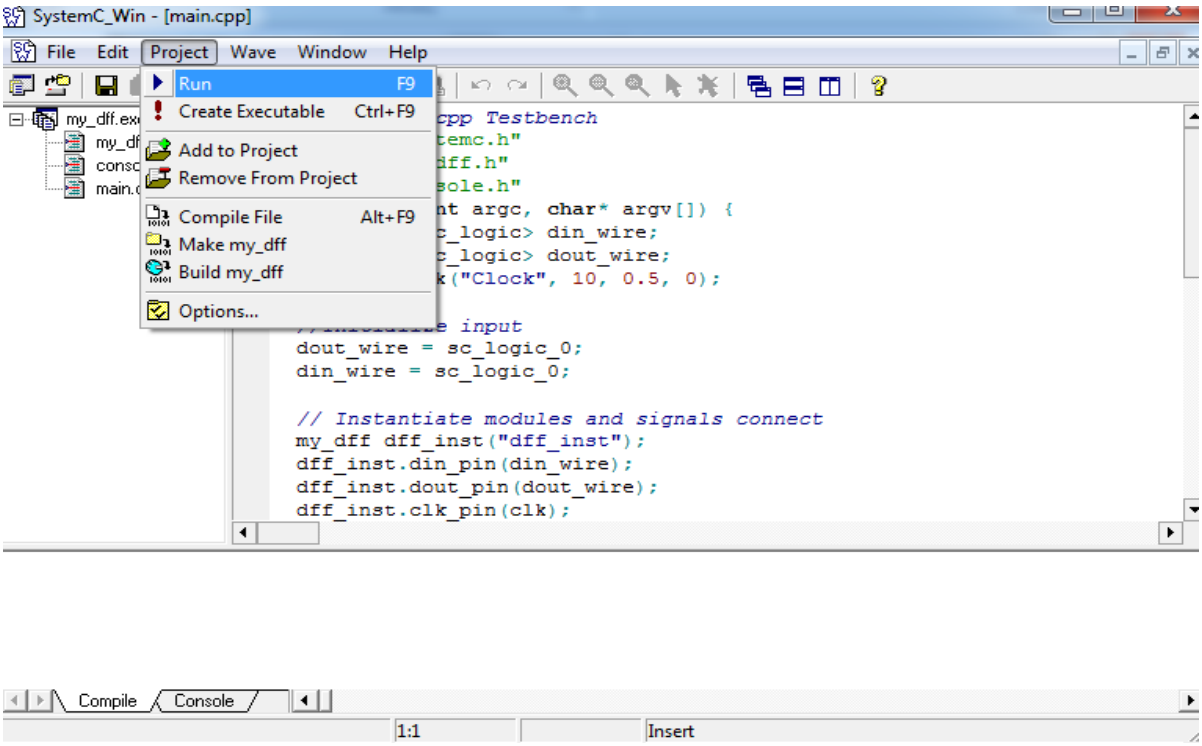
3. Відкриваємо файл проекту my_dff.scw.



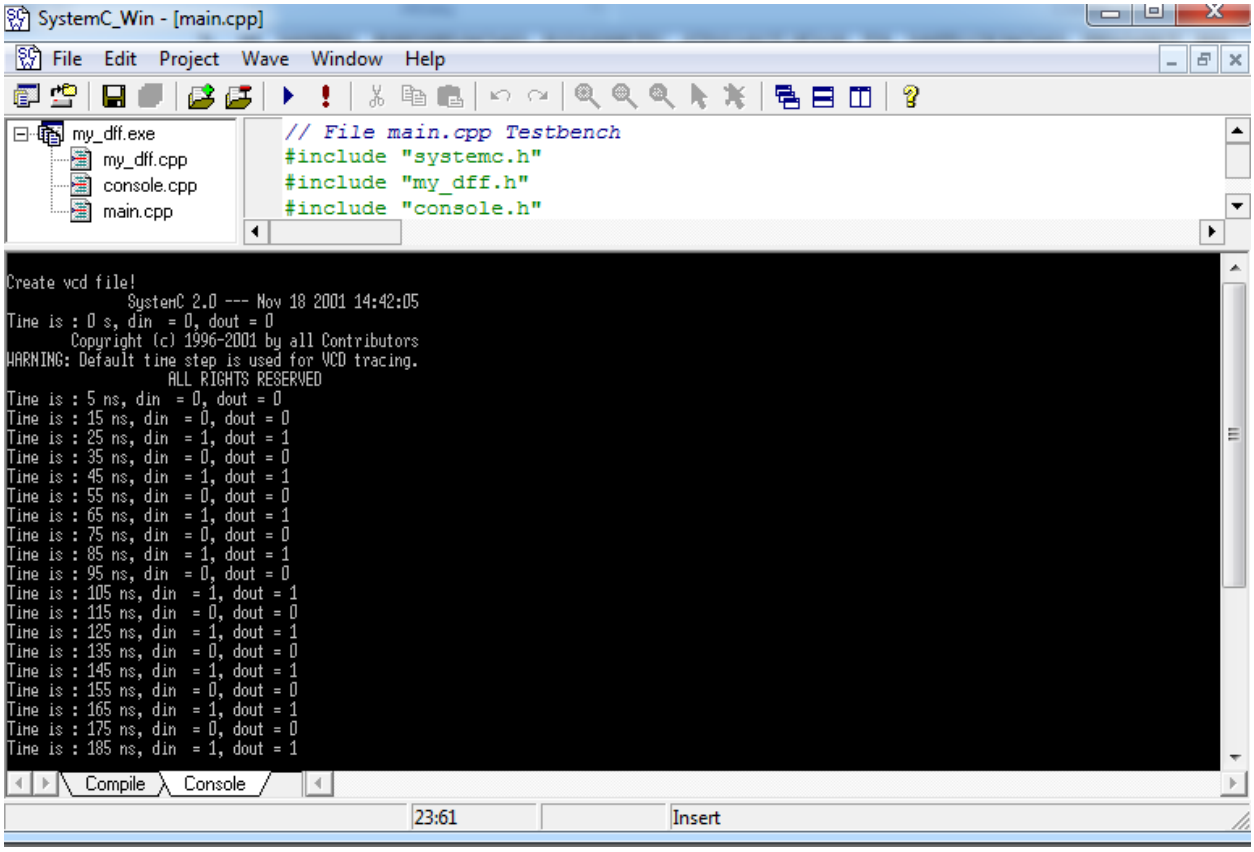
4. Відкриваємо C++ файли проекту та аналізуємо їх.



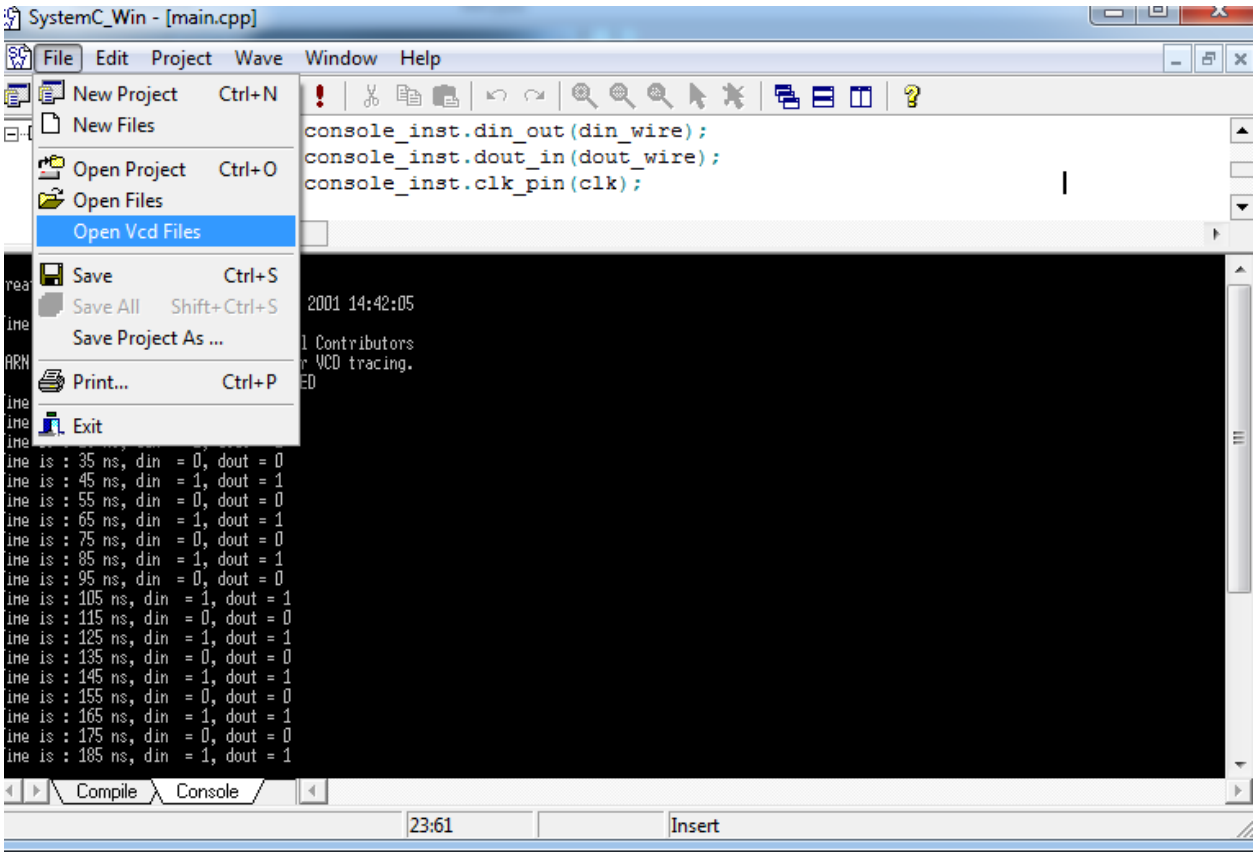
5. В меню вибираємо команду *Project/Run* та запускаємо проект на виконання.



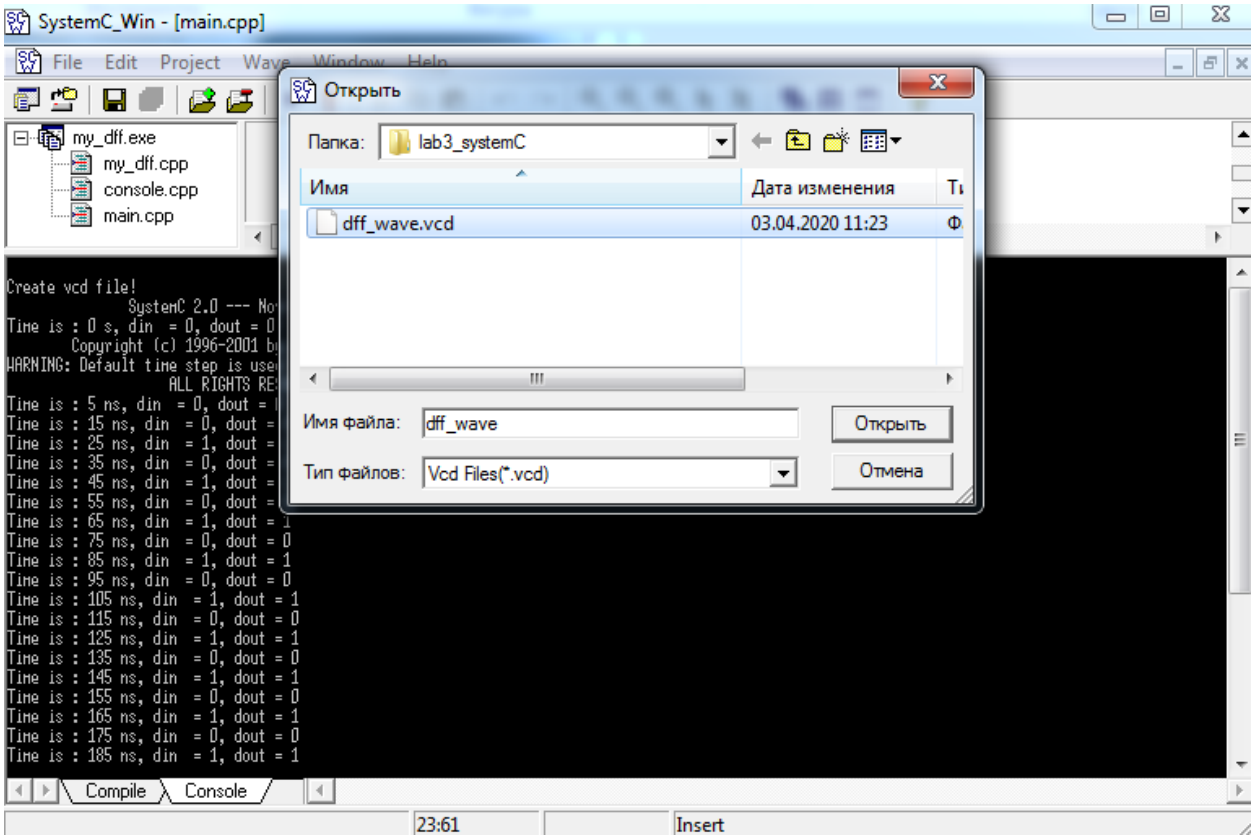
6. Спостерігаємо в консолі проекту результати роботи.



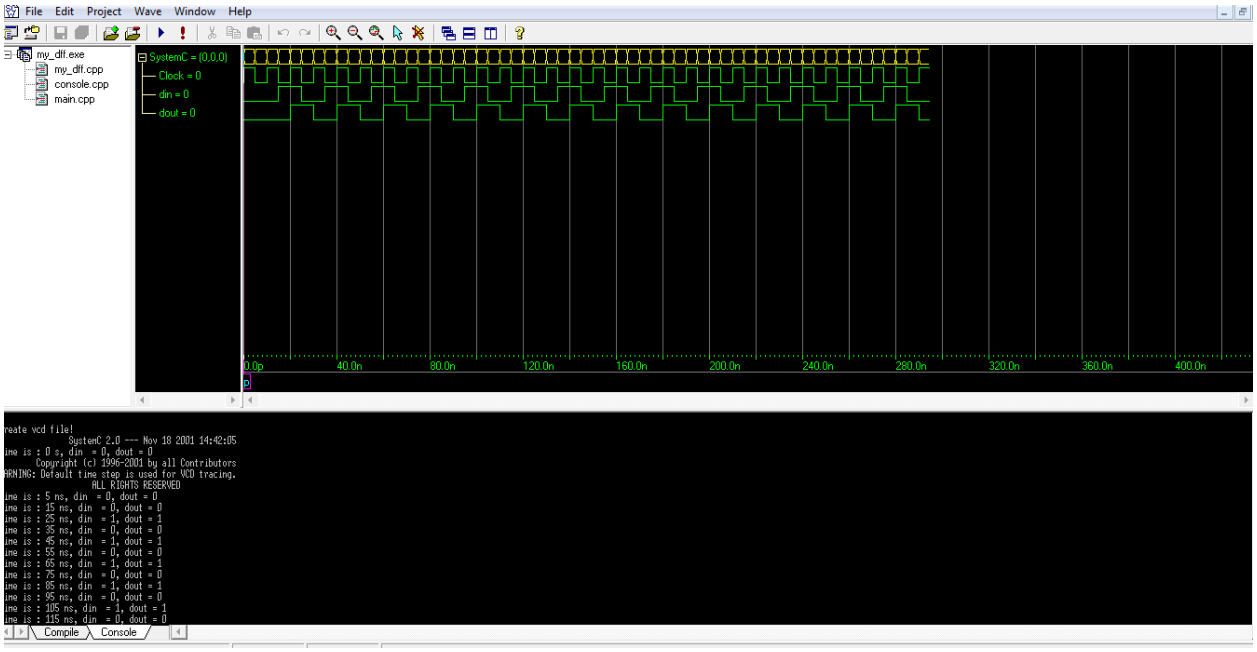
7. Наступним кроком в меню вибираємо File -> Open Vcd Files.



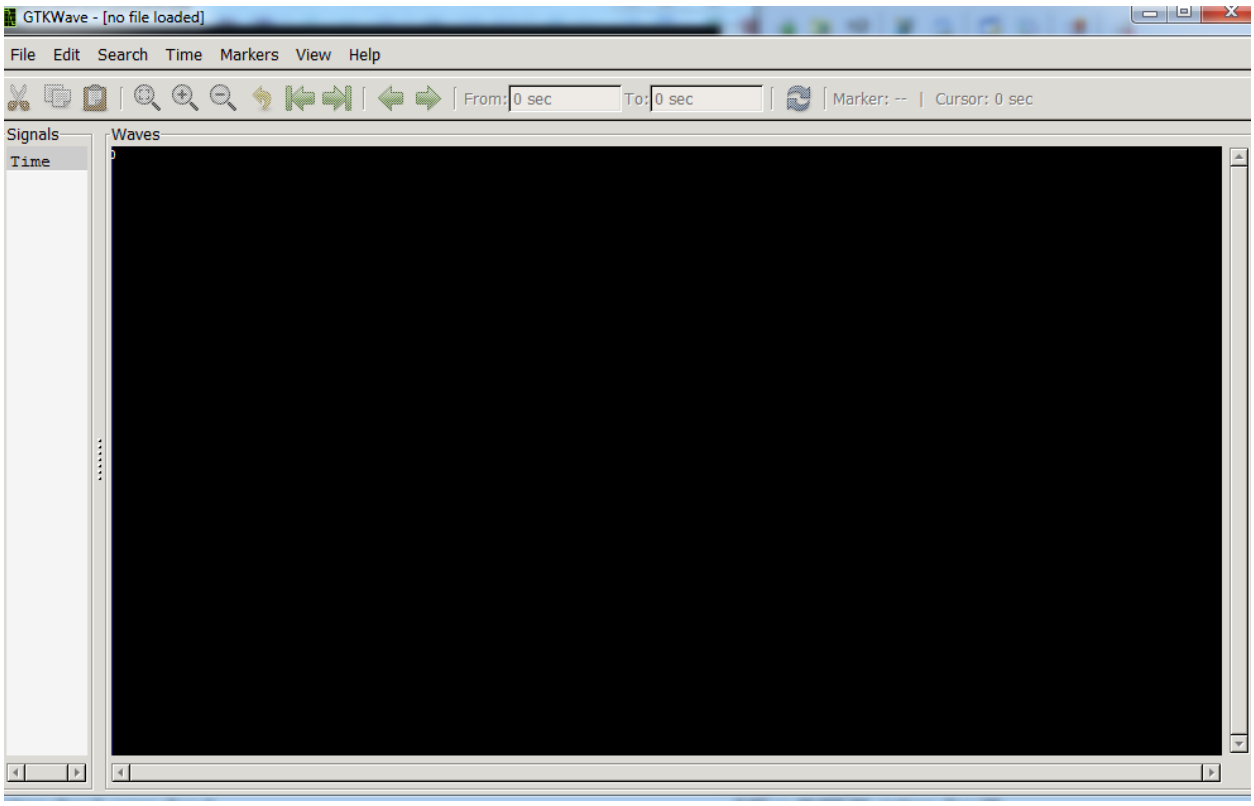
8. Відкриваємо файл dff_wave.vcd



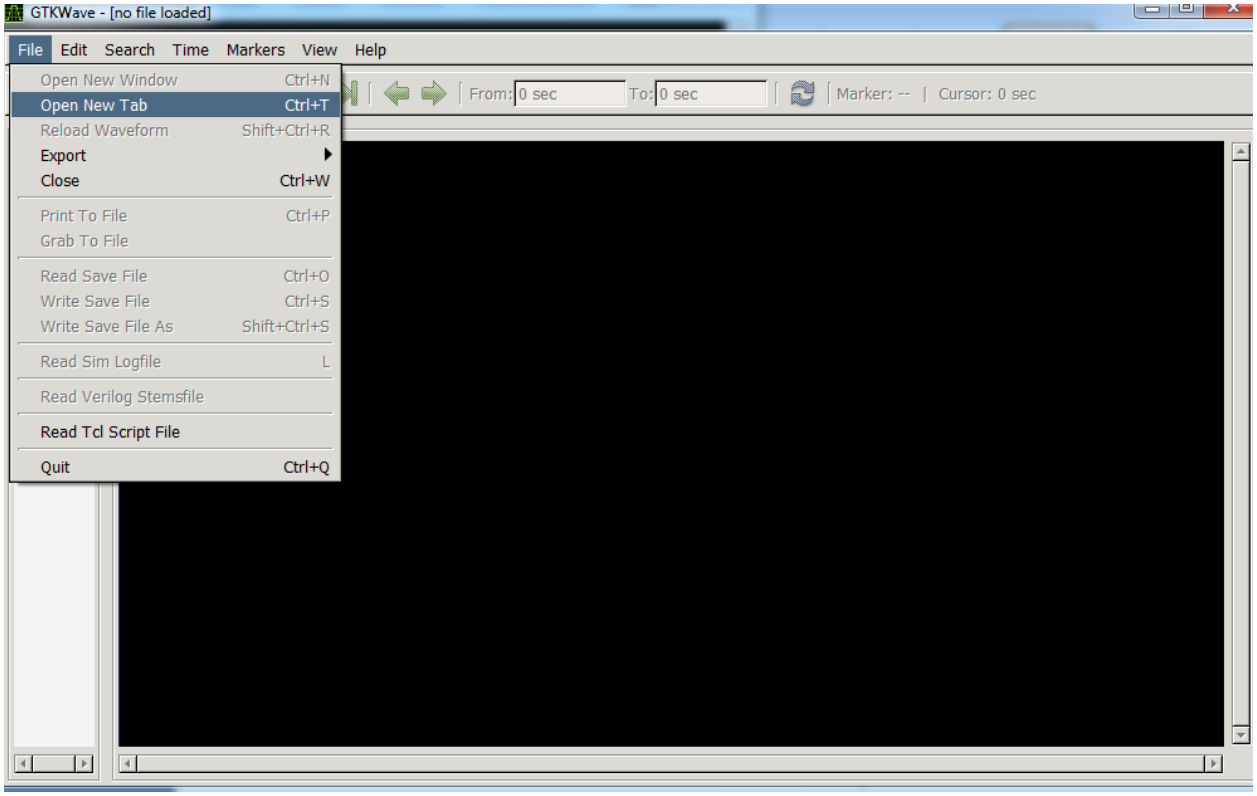
9. Аналізуємо отриману діаграму роботи D-тригера



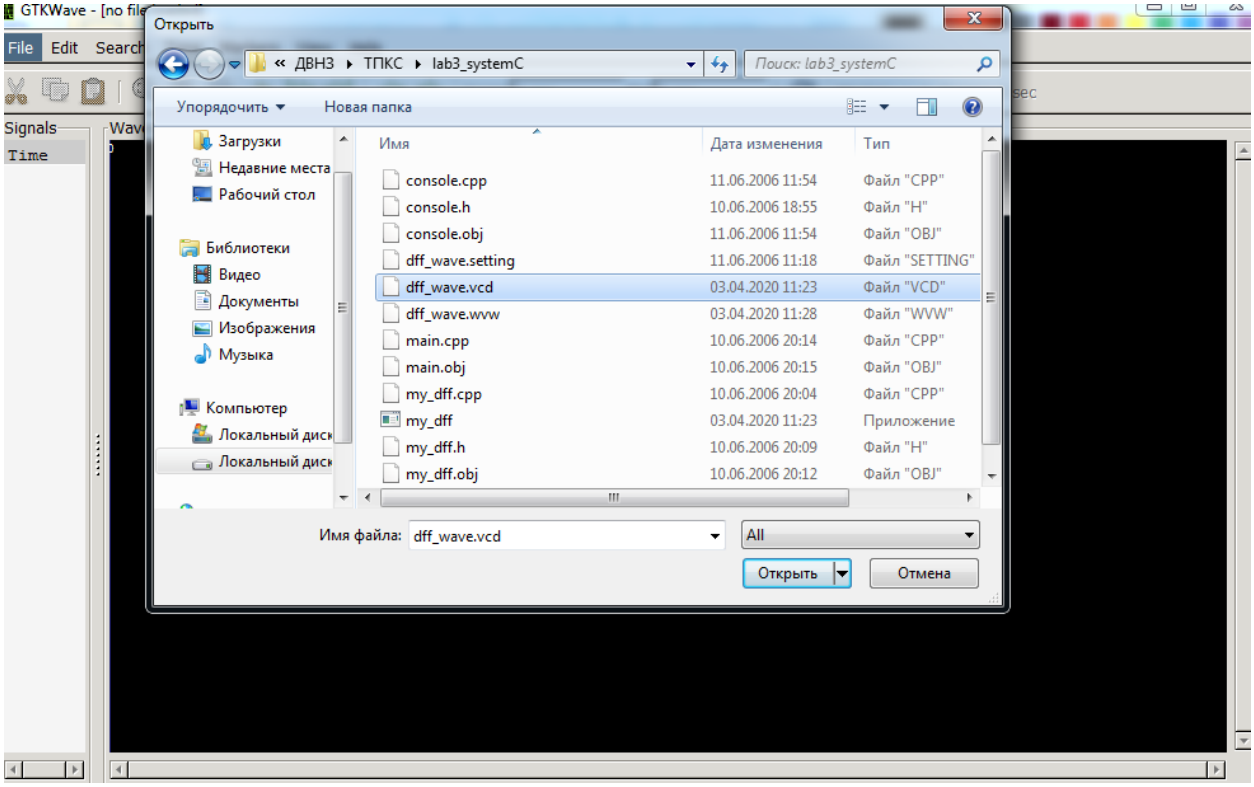
9. Запуск діаграми також можна виконати в середовищі GTKwave, інтерфейс якого наведено нижче.



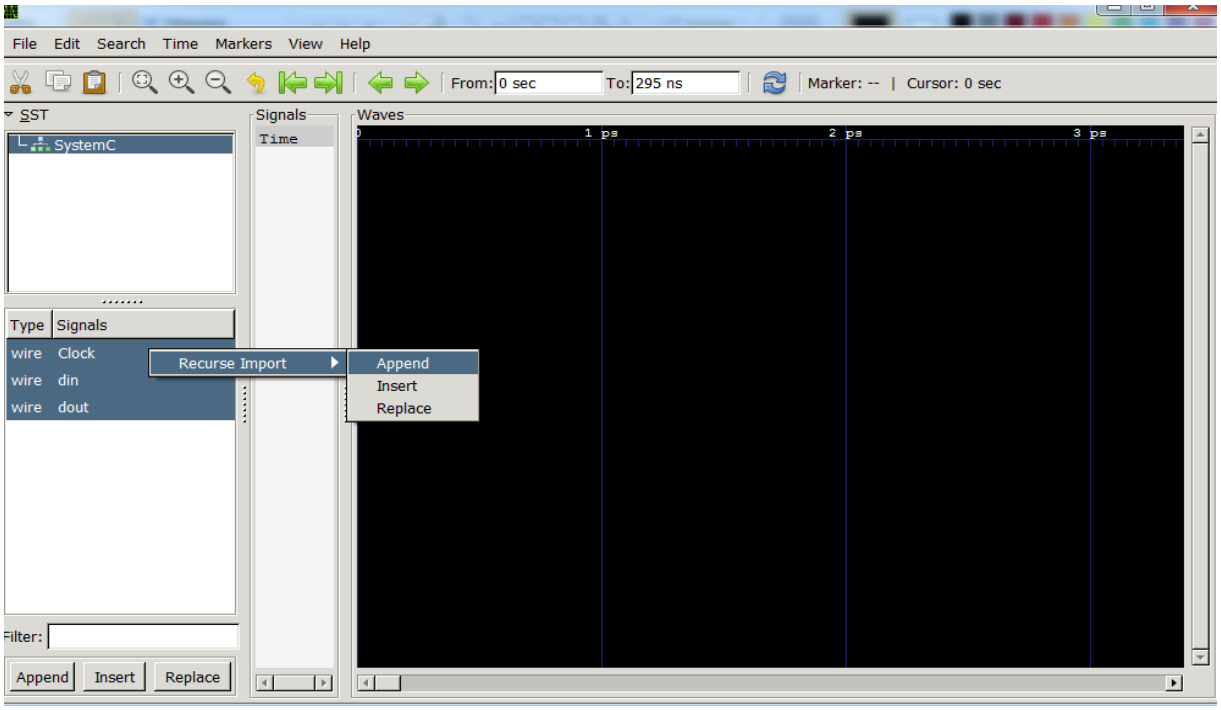
10. В основному меню вибираємо File -> Open New Tab.



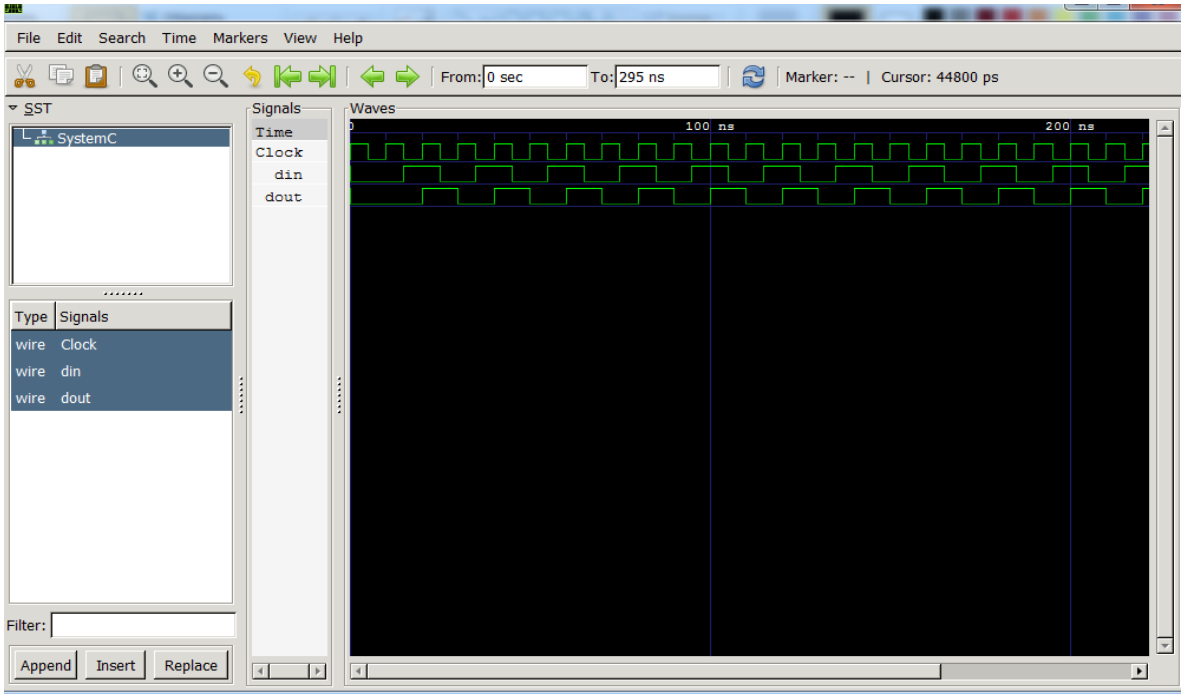
11. Вибираємо в папці проекту файл dff_wave.vcd.



12. Вибираємо сигнали проекту. Виділяємо їх та натискаємо праву кнопку миші після чого вибираємо опції Recurce Import -> Append та натискаємо кнопку Yes.



13. Спостерігаємо та аналізуємо результати роботи.



Завдання до лабораторної роботи №1

Реалізуйте та промоделюйте роботу різнотипних тригерів в Sysyem C.

№	Назва пристрою
1	Синхронний D-тригер
2	Асинхронний D-тригер
3	Синхронний RS-тригер
4	Асинхронний RS-тригер
5	Синхронний T-тригер
6	Асинхронний T-тригер
7	Синхронний JK-тригер
8	Асинхронний JK-тригер
9	Синхронний D-тригер
10	Асинхронний D-тригер
11	Синхронний RS-тригер
12	Асинхронний RS-тригер

Зміст звіту

1. Титульна сторінка.
2. Назва та мета лабораторної роботи.
3. Структура проекту my_dff, призначення кожної складової частини, опис команд та параметрів.
4. Результати моделювання роботи D-тригера та їх аналіз.
5. Висновки.

Контрольні запитання

1. Що таке SystemC?
2. Назвіть характеристики бібліотеки SystemC?
3. Які основні модулі системи SystemC_Win?
4. Що таке D-тригер?
5. Які бібліотеки SystemC використовуються в проекті my_dff?
6. Назвіть основні файли проекту та їх призначення ?

ЛАБОРАТОРНА РОБОТА №2

Тема роботи: Технології моделювання та дослідження роботи багаторозрядного суматора в середовищі System C.

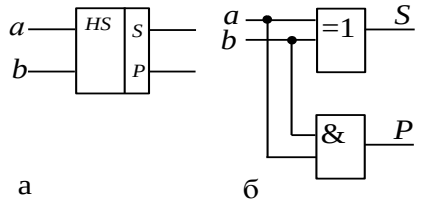
Мета роботи: Навчитися проектувати та моделювати роботу багато-розрядного суматора з використанням бібліотек System C.

Теоретичні відомості

За принципом побудови і типом використаних елементів розрізняють комбінаційні та накопичуючі суматори. Результати проміжного порозрядного додавання у накопичуючих суматорах зберігається (запам'ятовується) в елементарних комірках пам'яті, функцію яких викоиують тригери. Комбінаційні суматори не мають запам’ятовувачів. У них додавання двійкових чисел здійснюється позиційним паралельним кодом одночасно. Як і у всіх комбінаційних пристроїв, результат на виході у комбінаційних суматорах зникає зразу після припинення дії вхідних сигналів. Тому до складу комбінаційних суматорів, як правило, входять вхідні та вихідні регістри, тобто пристрої, що здатні записувати чи перезаписувати проміжний результат підсумовування у послідовному або у паралельному коді.

Напівсуматор – це пристрій (рис. 1), шо має два входи (для доданків a і b) і два виходи (суми S і переносу P), який призначений для виконання арифметичних дій за правилами, що наведені у табл. 1. З таблиці істинності видно, що напівсуматор виконує елементарне додавання двох однорозрядних двійкових чисел та підсумовування отриманого результату з переносом у наступний старший розряд. Тому логічна структура напівсуматора має відображати стан обох виходів згідно з виразами

$$S = a\bar{b} \vee \bar{a}b = a \oplus b;$$
$$P = ab$$



а б

Таблиця 1

a	*b*	*P*	*S*
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Рис. 1. Структура напівсуматора та таблиця істинності його роботи.

Однак у логіці роботи напівсуматора не передбачено переносу з сусіднього молодшого розряду, тому напівсуматор може здійснювати додавання тільки у молодшому розряді двійкових чисел. Поява одиниці переносу при додаванні двох розрядів (числа і переносу) дещо змінює правила підсумовування двійкових чисел. Такий однорозрядний суматор потребує ще один (третій) вхід переносу з сусіднього молодшого розряду. Для цього служить повний суматор.

Повний суматор (рис. 2) реалізує процедуру додавання двох однорозрядних двійкових чисел з урахуванням переносу з молодшого розряду. Тому він має три входи (a_i, b_i, P_i) і два виходи (S_i і P_{i+1}). Логіка роботи повного суматора відображена у табл. 2, де a_i, b_i –доданки двійкових чисел в *i*-му розряді; Pi, Pi_{+1}– переноси, відповідно з молодшого розряду *i* в сусідній старший розряд *i+1*; *S* – утворена сума в *i*-му розряді.

Для додавання двох *n*-розрядних двійкових чисел *A* і *B* потрібно, очевидно, використати *n* однорозрядних повних суматорів. При цьому можуть бути два способи підсумовування – послідовне і паралельне. Додавання чисел в послідовних суматорах відбувається порозрядно, послідовно в часі. В паралельних суматорах додавання всіх розрядів багаторозрядних чисел відбувається одночасно.

Таблиця 2

a_i	b_i	P_i	P_{i+1}	S_i

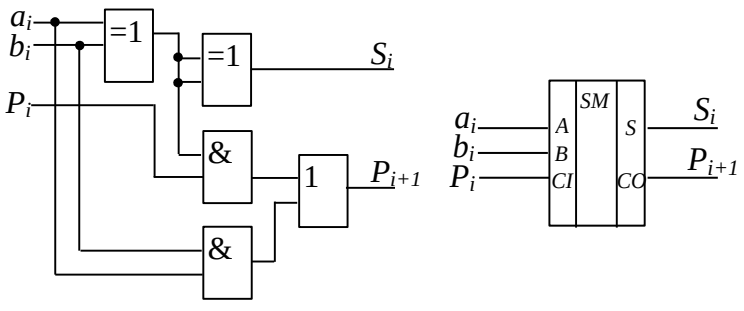
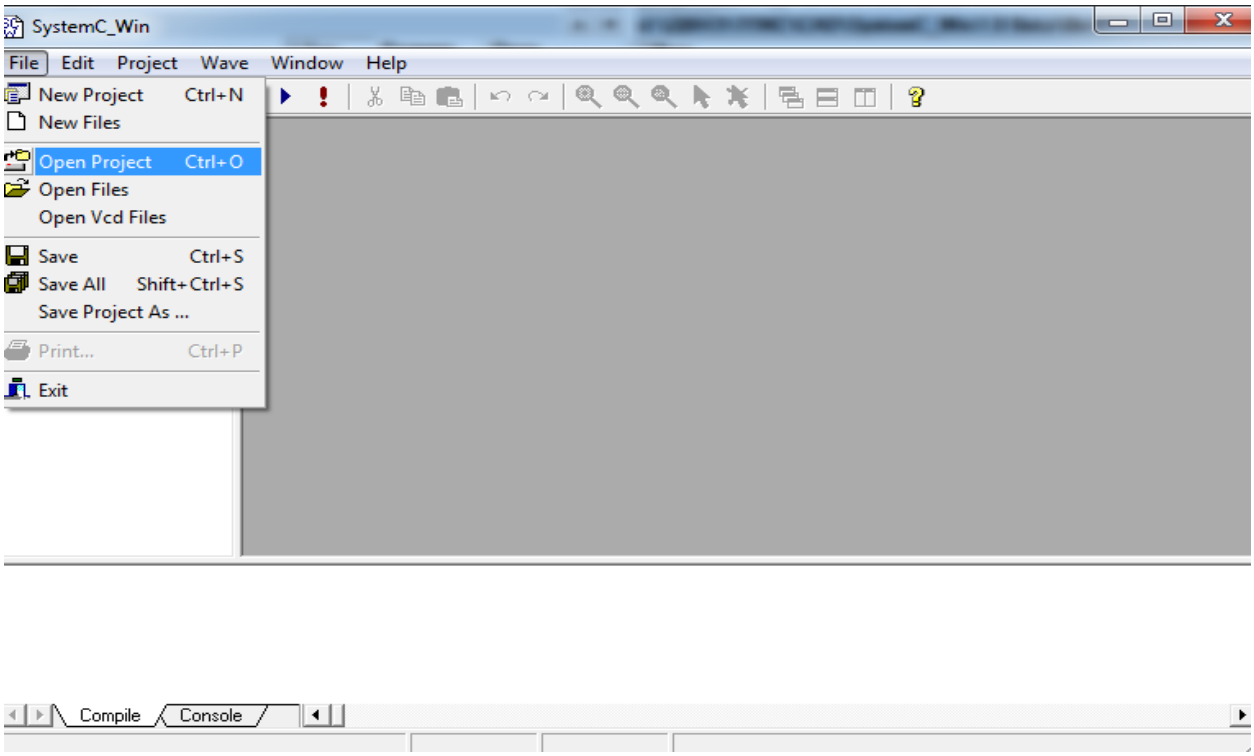


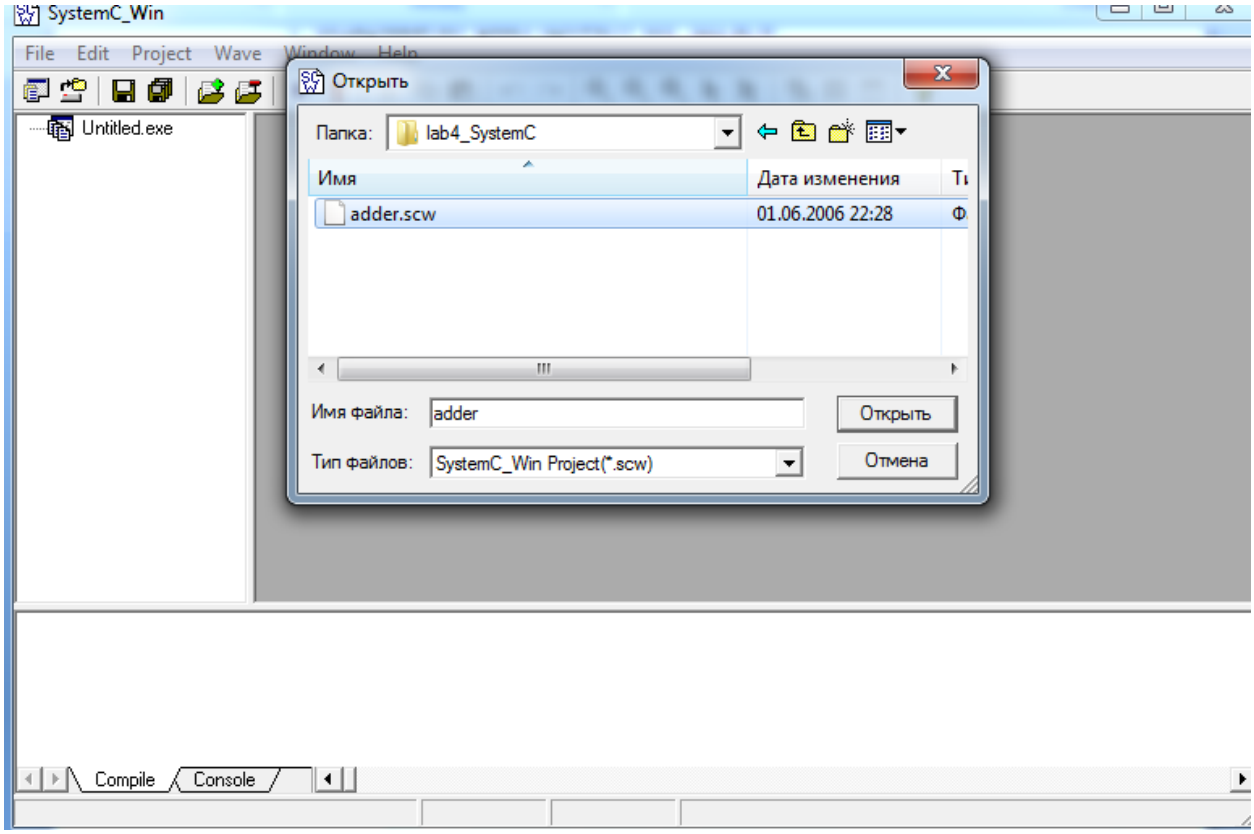
Рис. 3. Структура повного суматора та таблиця істинності його роботи

Хід виконання роботи

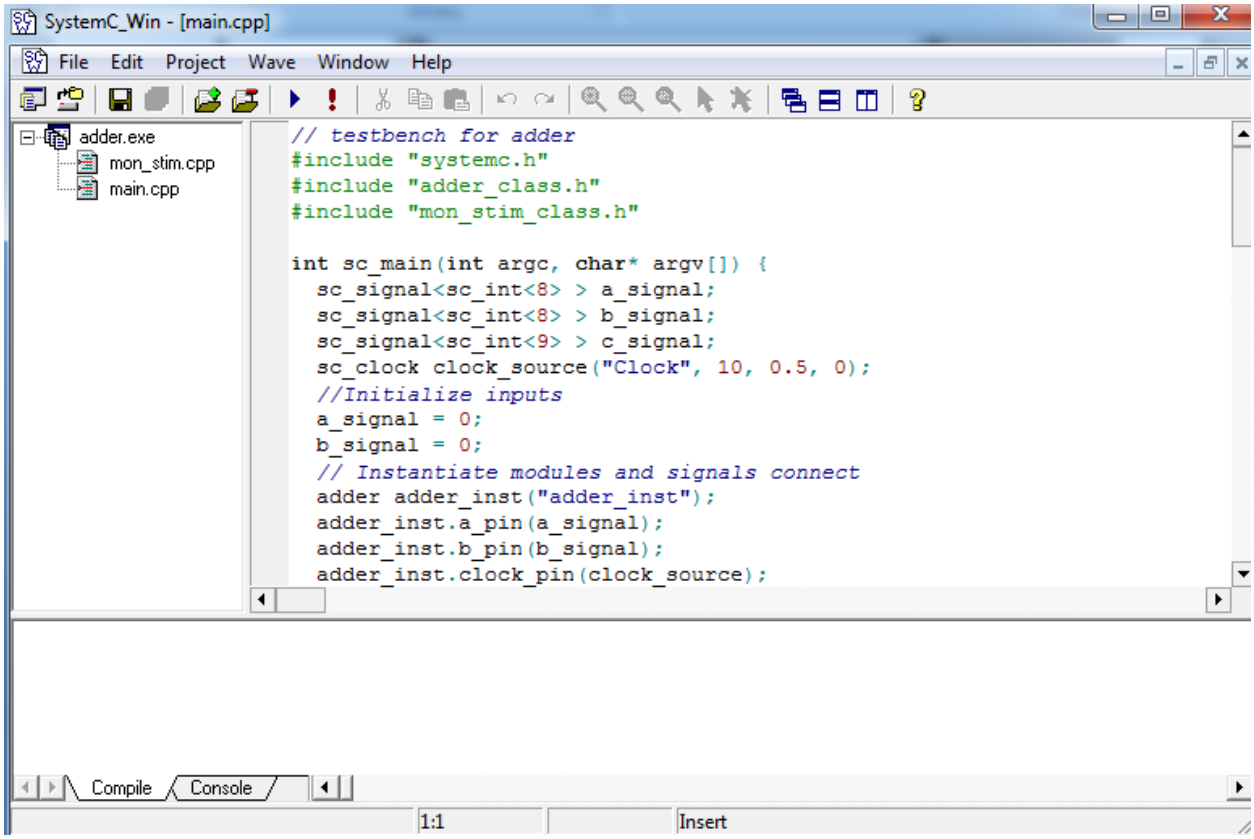
1. У меню *File\Open Project* вибираємо готовий проект lab2_SystemC.



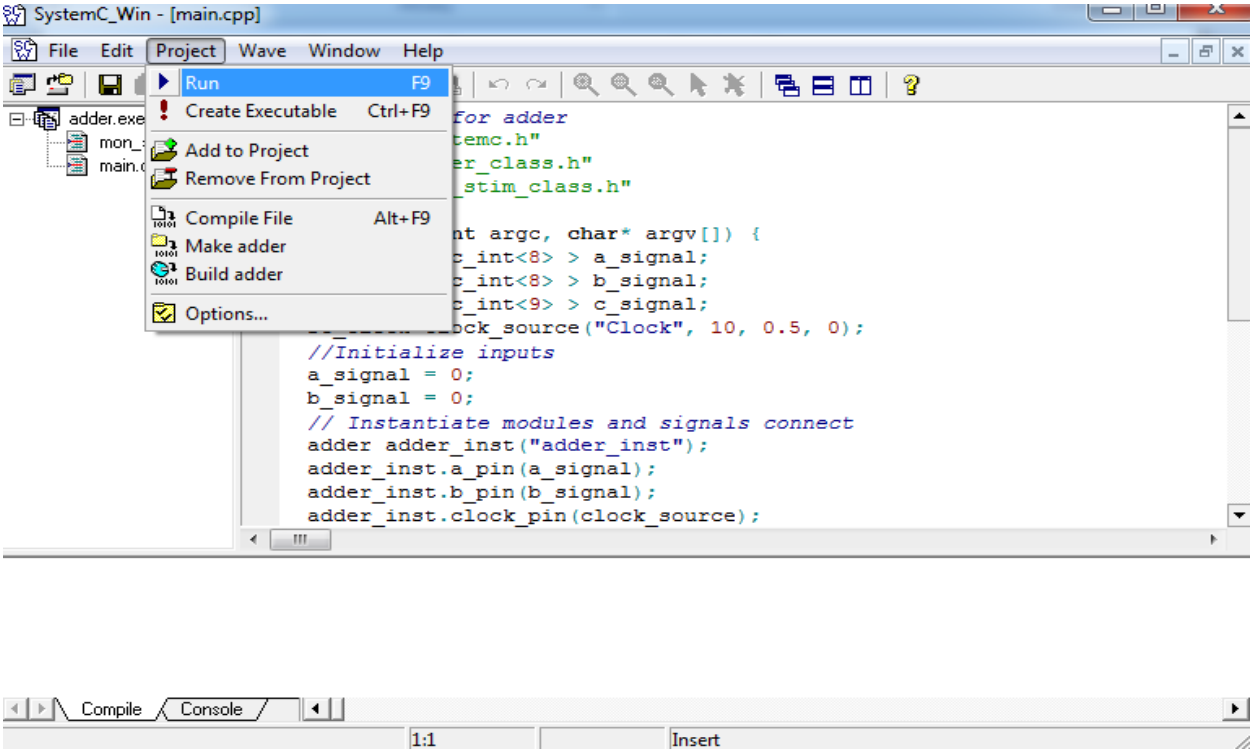
2. Відкриваємо файл проекту adder.scw.



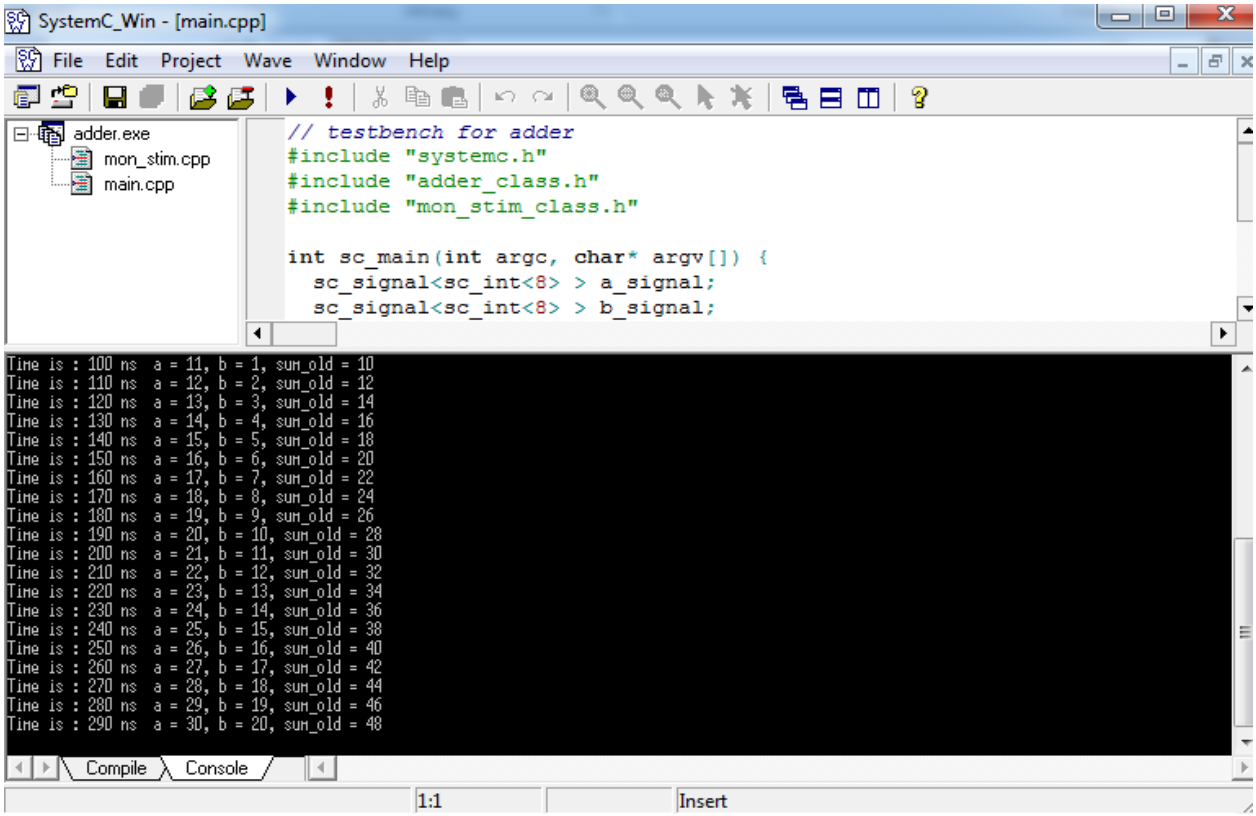
3. Відкриваємо C++ файли проекту та аналізуємо їх.



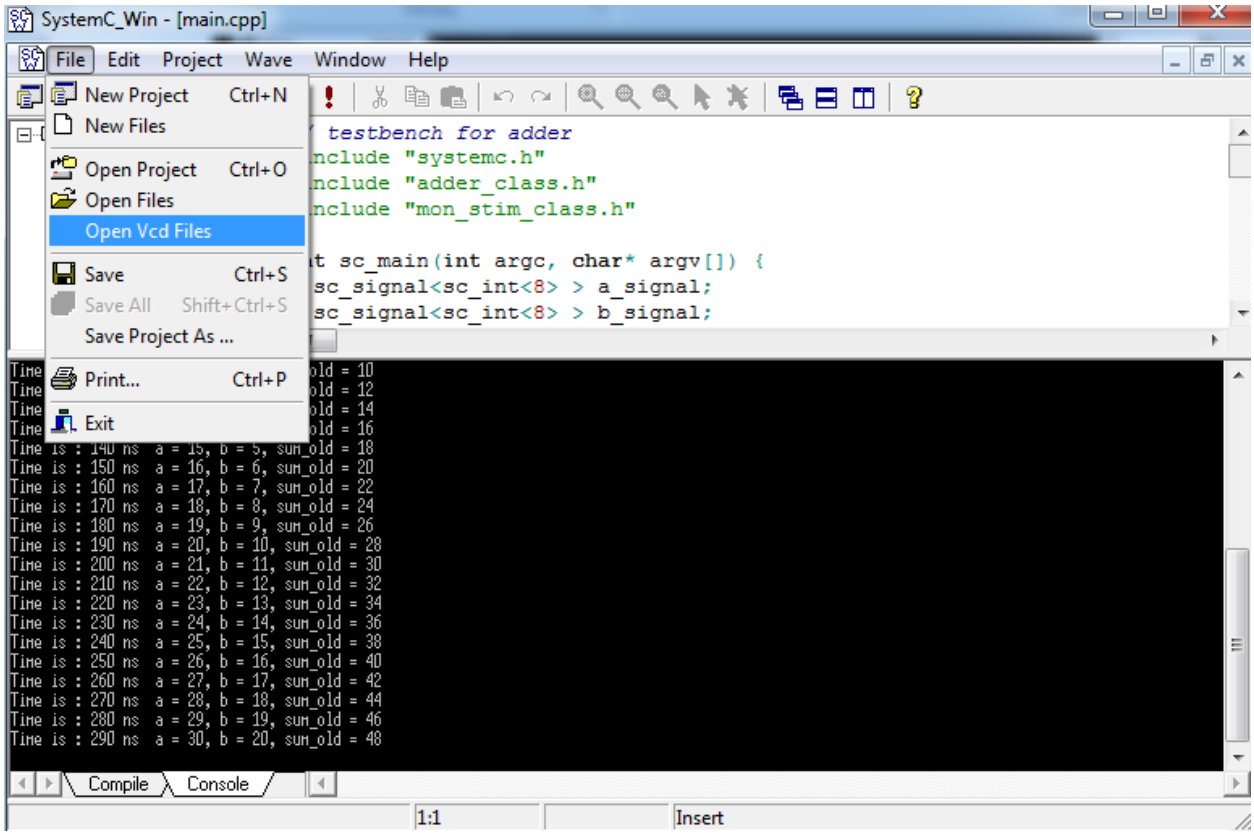
4. В меню вибираємо команду *Project/Run* та запускаємо проект на виконання.



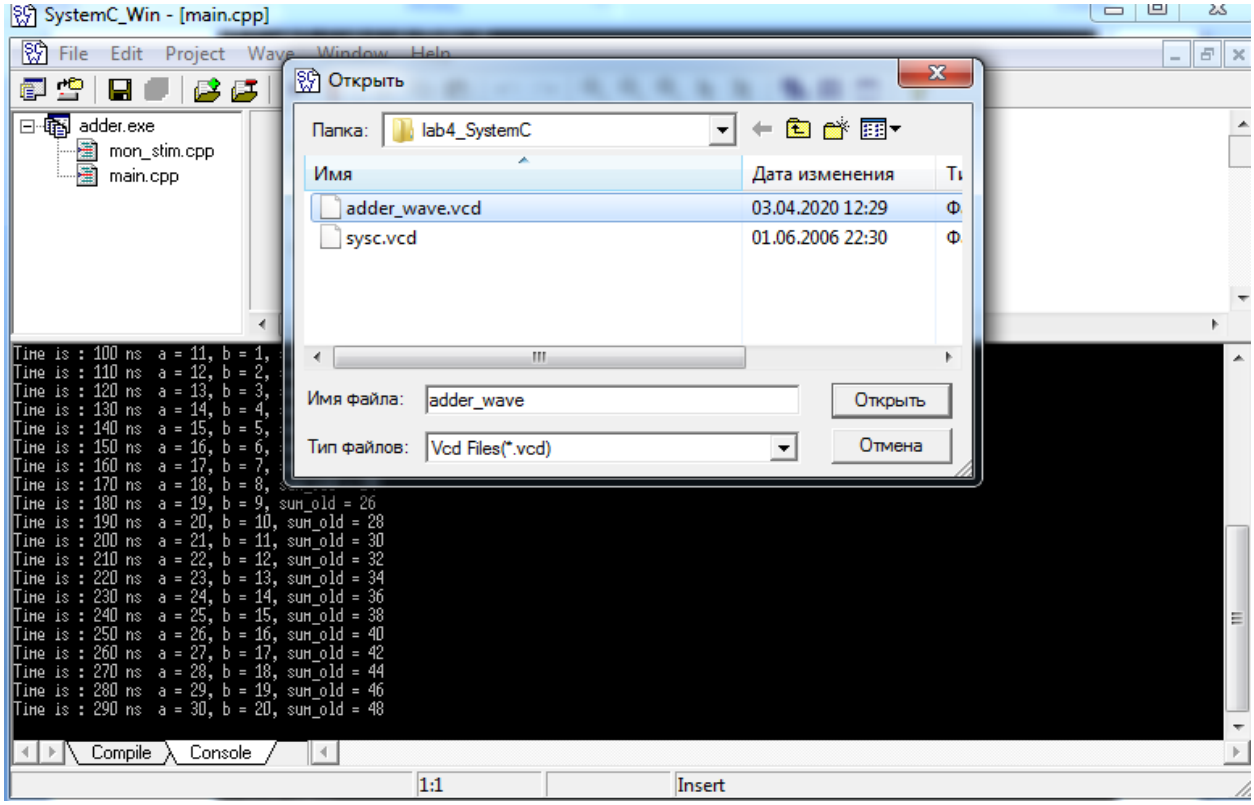
5. Спостерігаємо в консолі проекту результати роботи.



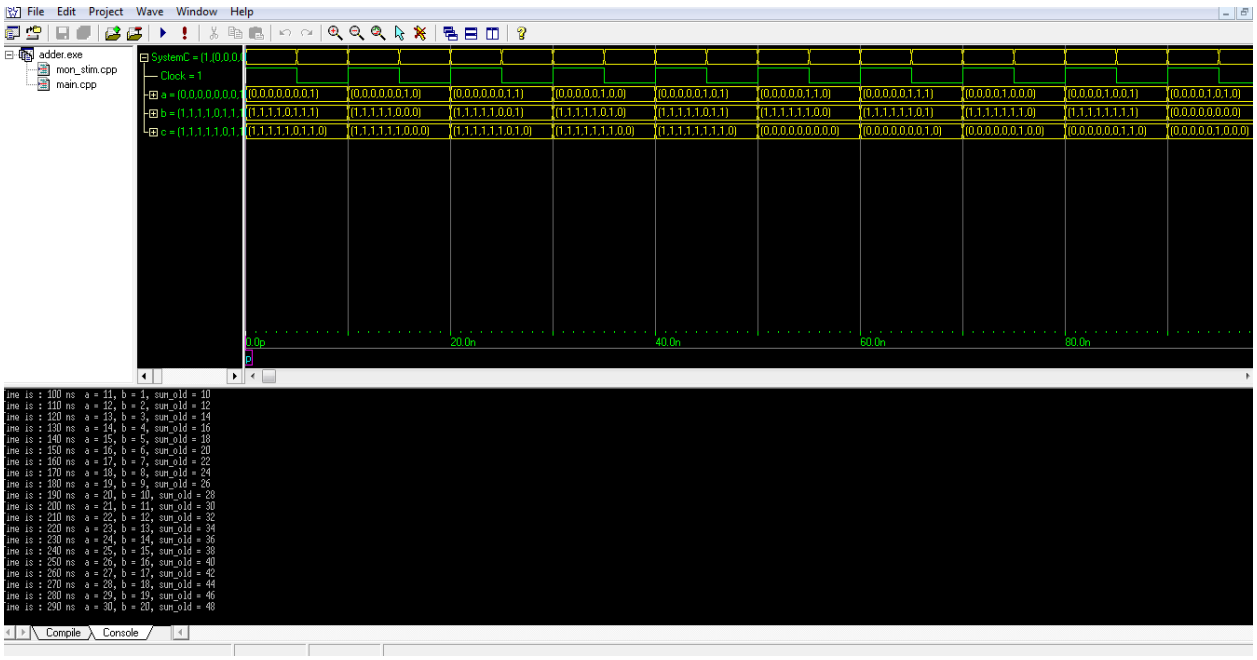
6. Наступним кроком в меню вибираємо File -> Open Vcd Files.



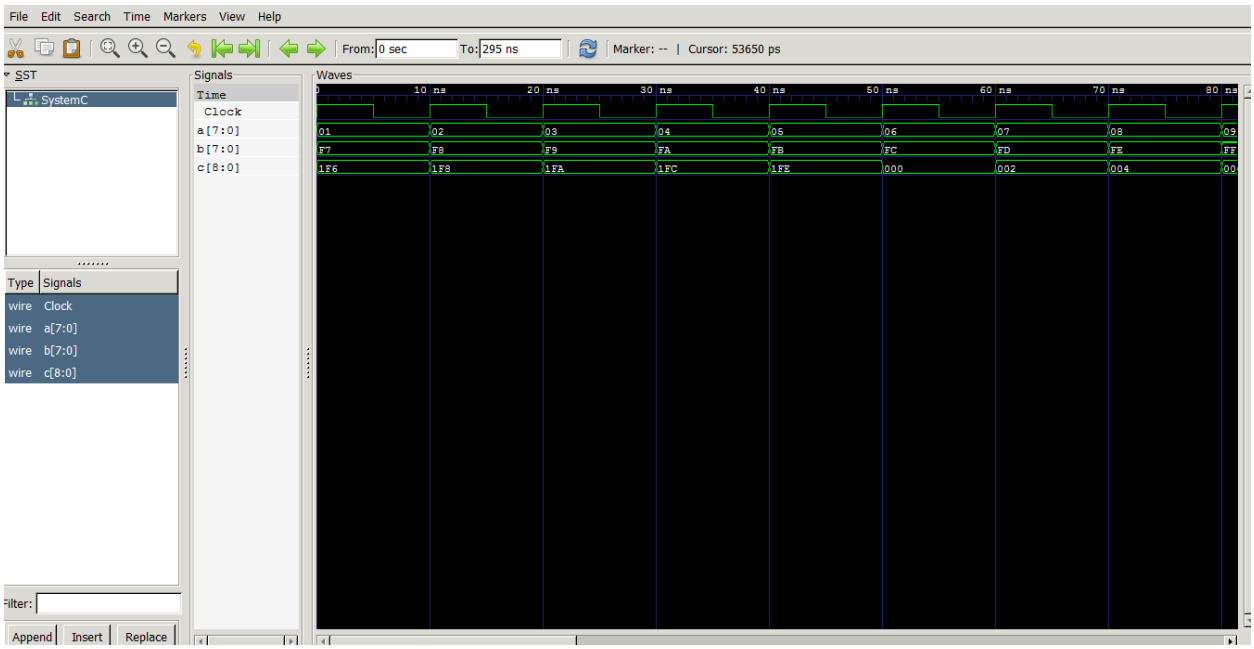
7. Відкриваємо файл adder_wave.vcd



8. Аналізуємо отриману діаграму роботи двійкового суматора.



9. Запуск діаграми роботи суматора в середовищі GTKwave.



Зміст звіту

1. Титульна сторінка.
2. Назва та мета лабораторної роботи.
3. Структура програми моделювання роботи багаторозрядного суматора.
4. Результати моделювання роботи суматора.
5. Висновки.

Завдання до лабораторної роботи №2

Реалізуйте та промоделюйте роботу багаторозрядного суматора в Sysyem C.

№	Назва пристрою	Розрядність вхідних даних (a і b)
1	Багаторозрядний суматор	6
2	Багаторозрядний суматор	8
3	Багаторозрядний суматор	10
4	Багаторозрядний суматор	12
5	Багаторозрядний суматор	14
6	Багаторозрядний суматор	16
7	Багаторозрядний суматор	18
8	Багаторозрядний суматор	20
9	Багаторозрядний суматор	22
10	Багаторозрядний суматор	24

11	Багаторозрядний суматор	28
12	Багаторозрядний суматор	32

Контрольні запитання

1. Що таке SystemC?
2. Назвіть характеристики бібліотеки SystemC?
3. Які основні модулі системи SystemC_Win?
4. Що таке суматор?
5. Які бібліотеки SystemC використовуються в проекті my_sum?
6. Назвіть основні файли проекту та їх призначення?

ЛАБОРАТОРНА РОБОТА №3

Тема роботи: Технології моделювання та дослідження роботи лічильника в середовищі System C.

Мета роботи: навчитися проектувати та моделювати роботу лічильника з використанням бібліотек System C.

Теоретичні відомості

Лічильник - послідовнісна схема, що перетворює поступаючі на вхід імпульси в паралельний двійковий код, відповідний їх кількості.

Лічильники виконуються на трігерах з рахунковим входом (Т-трігерах). За способом рахунку лічильники можуть бути підсумовуючі, віднімаючі і реверсивні. У підсумовуючому лічильнику при подачі на вхід імпульсу код числа, що зберігається у лічильнику, зростає на одиницю, а у віднімаючому - зменшується на одиницю. Отже, підсумовуючий лічильник виконує прямий, а віднімаючий - обернений /зворотній/ підрахунок числа імпульсів, що надійшли на його вхід. Реверсивний лічильник може працювати в режимі прямого та оберненого підрахунку.

За способом перемикання трігерів лічильники поділяються на асинхронні і синхронні. У асинхронних лічильниках трігери перемикаються послідовно (асихронно) від розряду до розряду, а в синхронних одночасно. Один Т-трігер забезпечує коефіцієнт перерахунку M = 2, а n трігерів - $M = 2^n$. При підсумовуванні імпульсів необхідно формувати переніс з i-го в (i+1)-ий розряд за наступними правилами:

1) Переніс CR з i-го в (i+1)-ий розряд формується, якщо у всіх розрядах з i-го по 0-й записана максимальна для даної системи числення цифра, при цьому розряди молодше (i+1)-го обнулюються. На прямих виходах трігерів цих розрядів Qi формується негативний перепад, а на інверсних - позитивний.

2) Якщо в лічильнику використовуються трігери з прямим динамічним входом, то сигнал переносу в підсумовуючому лічильнику знімається з інверсних виходів попередніх трігерів, а якщо трігери з інверсним динамічним входом, то сигнал переносу береться з прямих виходів.

Асинхронний лічильник з послідовним переносом можна побудувати на JK-трігерах з інверсними динамічними входами. Як приклад візьмемо чотирьохрозрядний лічильник (рис. 1). Чотири двійкові розряди лічильника забезпечують M = 16 варіантів вихідного коду.

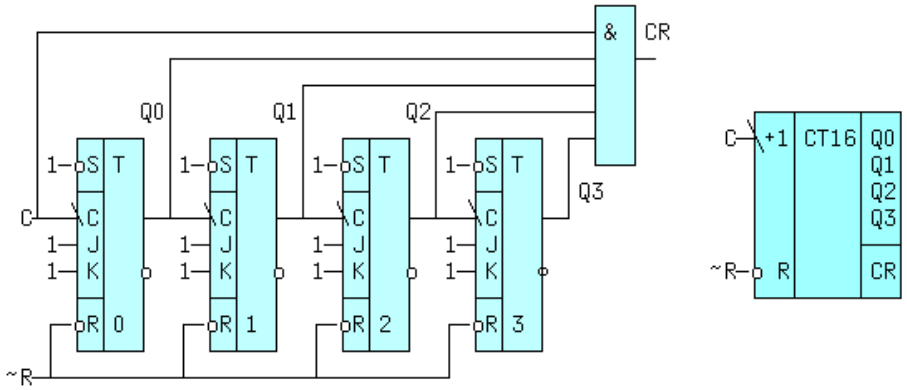


Рис. 1. Асинхронний лічильник.

Сигнали переносу повинні зніматися з прямих виходів трігерів, які перемикаються послідовно один за одним, тобто асихронно. Трігери працюють в режимі рахунку (J=K=1). Лічильник доповнений схемою формування прискореного переносу CR (Carry), вихід якої може бути підключений до рахункового входу C наступного такого ж лічильника. Входи ~R всіх трігерів об'єднані, а на входи ~S подана "1", що дозволяє скидати лічильник сигналом ~R=0. Рахунковий вхід підсумовуючого лічильника позначається "+1". Часові діаграми схеми, без врахування затримки сигналу, приведені на рис. 2.

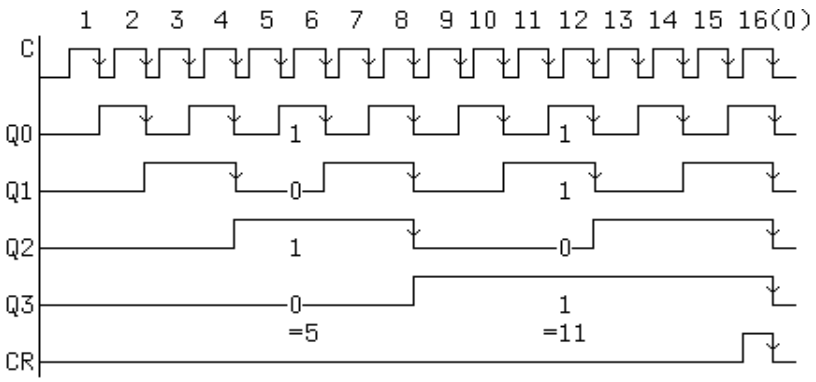


Рис. 2. Часові діаграми двійкового лічильника.

Аналіз часових діаграм дозволяє зробити ряд висновків:

1) Після n-го по рахунку вхідного імпульсу код на виходах Q = Q3Q2Q1Q0 = n, наприклад після 5-го код $Q = 0101_2 = 5_{10}$, а після 11-го - $Q = 1011_2 = 11_{10}$, тобто схема дійсно є лічильником.

2) З приходом активного фронту 16-го імпульсу всі трігери скидаються і далі процес повторюється, тобто модуль рахунку M=16.

3) Схема також є дільником частоти вхідних імпульсів на 2 в степені (i+1), де i - номер трігера, з якого знімається вихідний сигнал.

4) Якщо знімати вихідний код з інверсних виходів, то початкове значення $Q=Q0°Q1°Q2°Q3=1111_2=15_{10}$, тобто максимальному числу для чотирьох розрядів, і далі, з приходом наступного імпульсу, код на виходах зменшується на 1. В цьому випадку лічильник називають віднімаючим. Такого ж результату можна добитися, якщо знімати переніс з протилежних виходів трігерів, а код, як і раніше, з прямих. Рахунковий вхід віднімаючого лічильника позначається "-1".

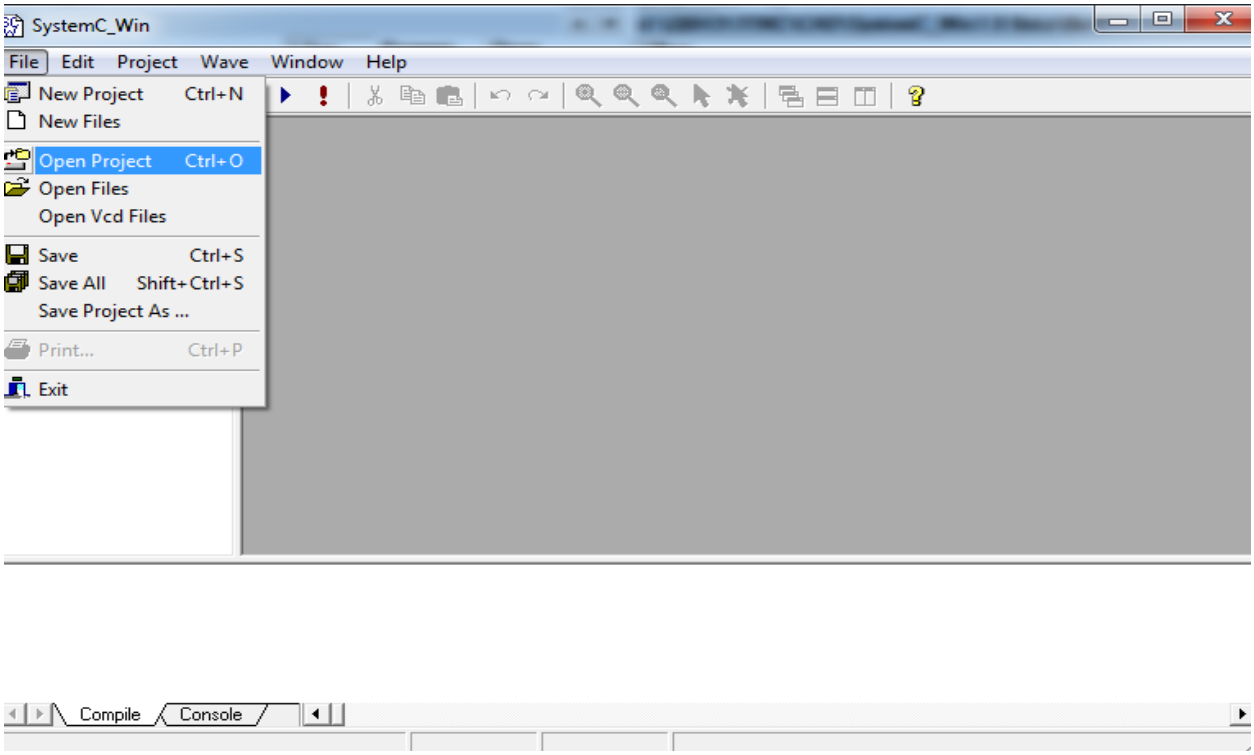
5) Задній фронт імпульсу переносу співпадає з моментом переходу всіх трігерів з 1 в 0 для підсумовуючого лічильника, і з моментом переходу з 0 в 1 - для віднімаючого.

Швидкість рахунку або максимальна частота вхідних імпульсів визначається затримкою сигналу від моменту приходу активного фронту рахункового імпульсу до появи нового коду на виході останнього трігера: tзт.р. лічильника = n * tзт.р. трігера, де n - кількість трігерів. Тоді

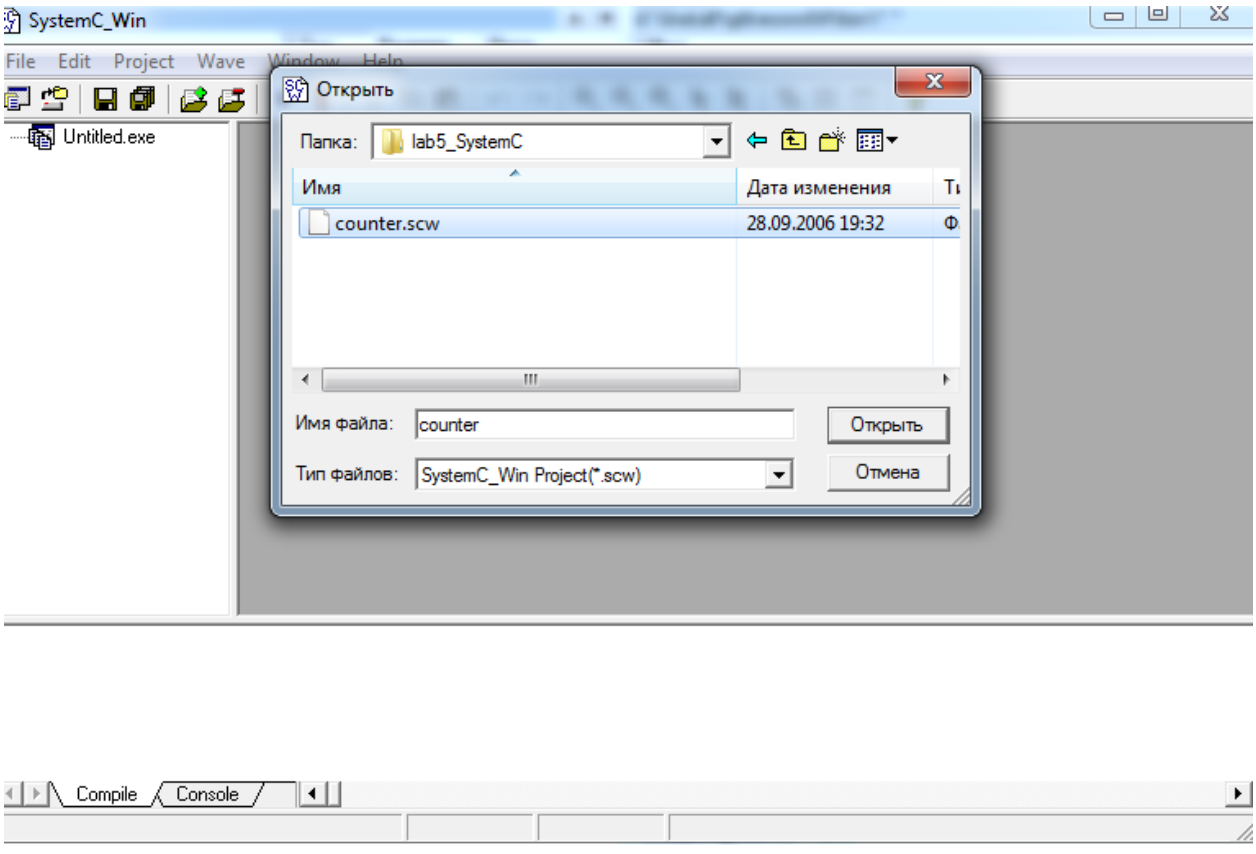
Fмакс.рах < 1/tзт.р .лічильника.

Хід виконання роботи

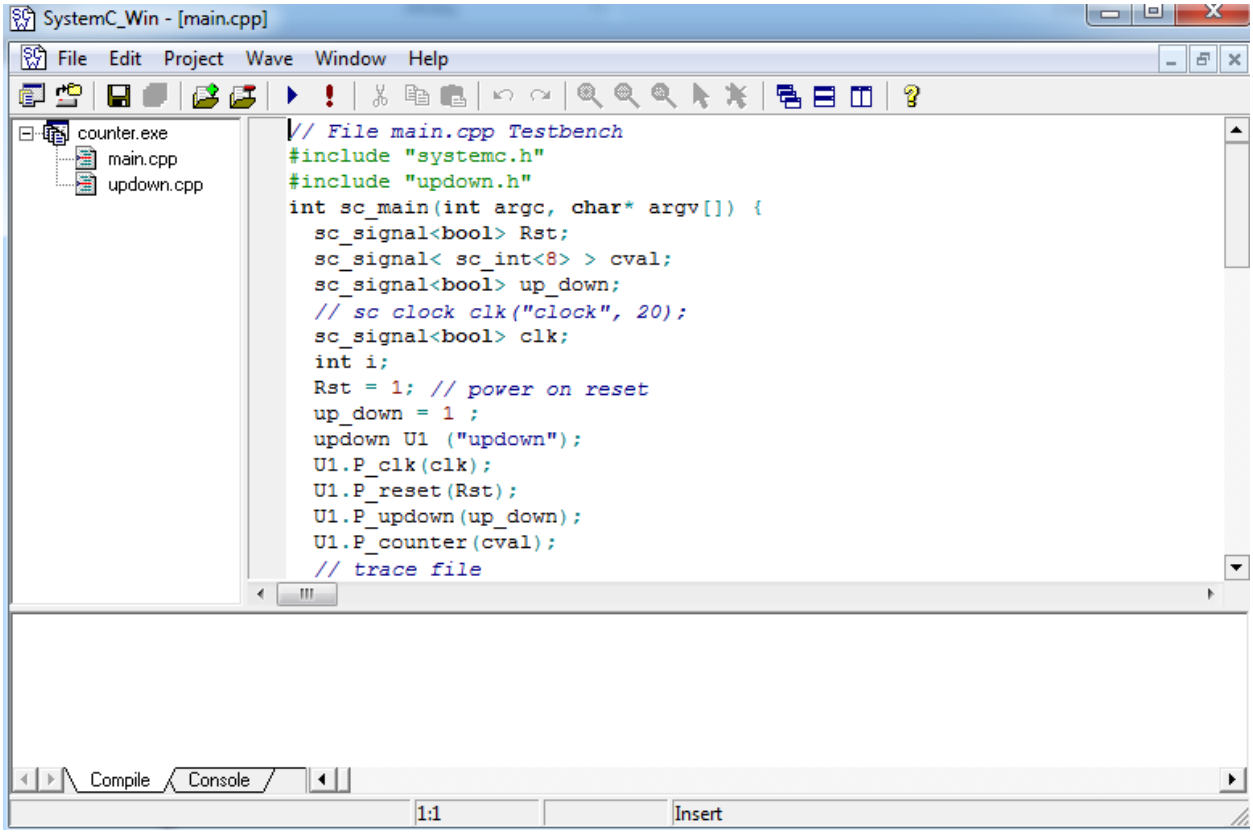
1. У меню *File\Open Project* вибираємо готовий проект lab3_SystemC.



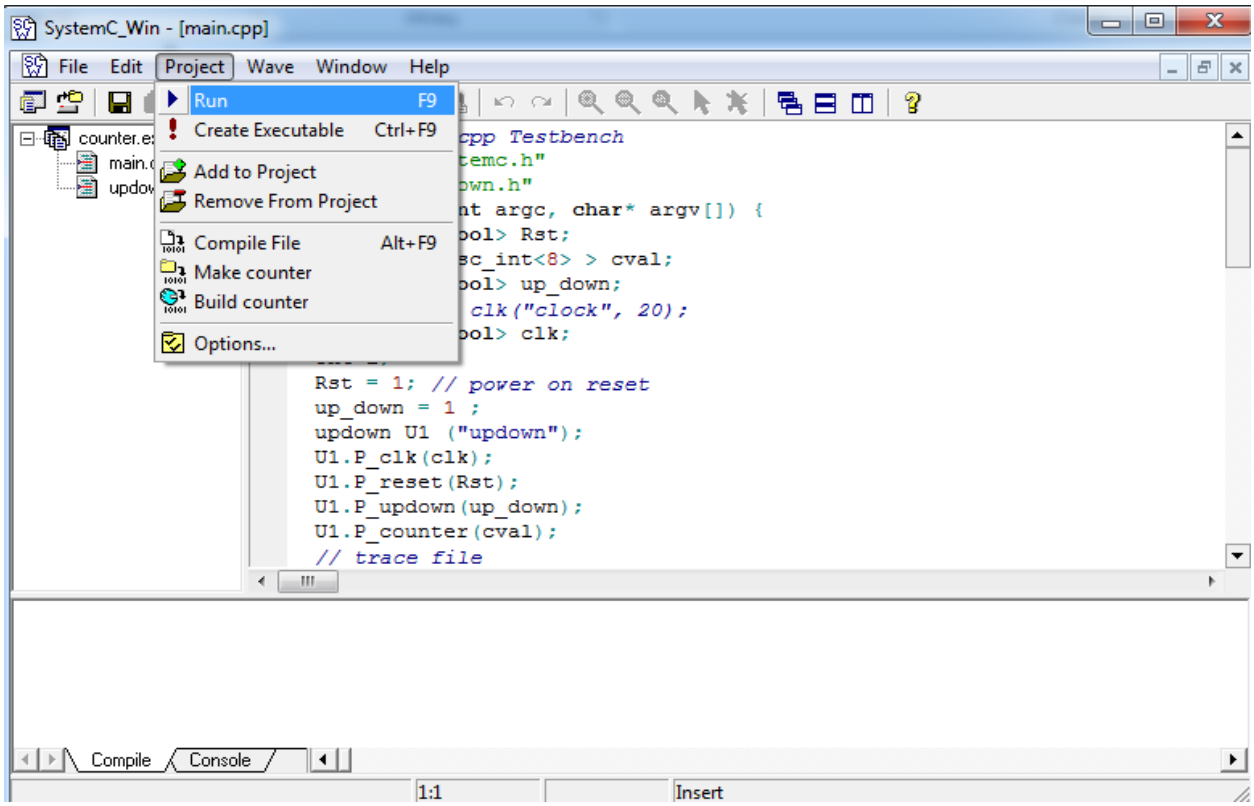
2. Відкриваємо файл проекту counter.scw.



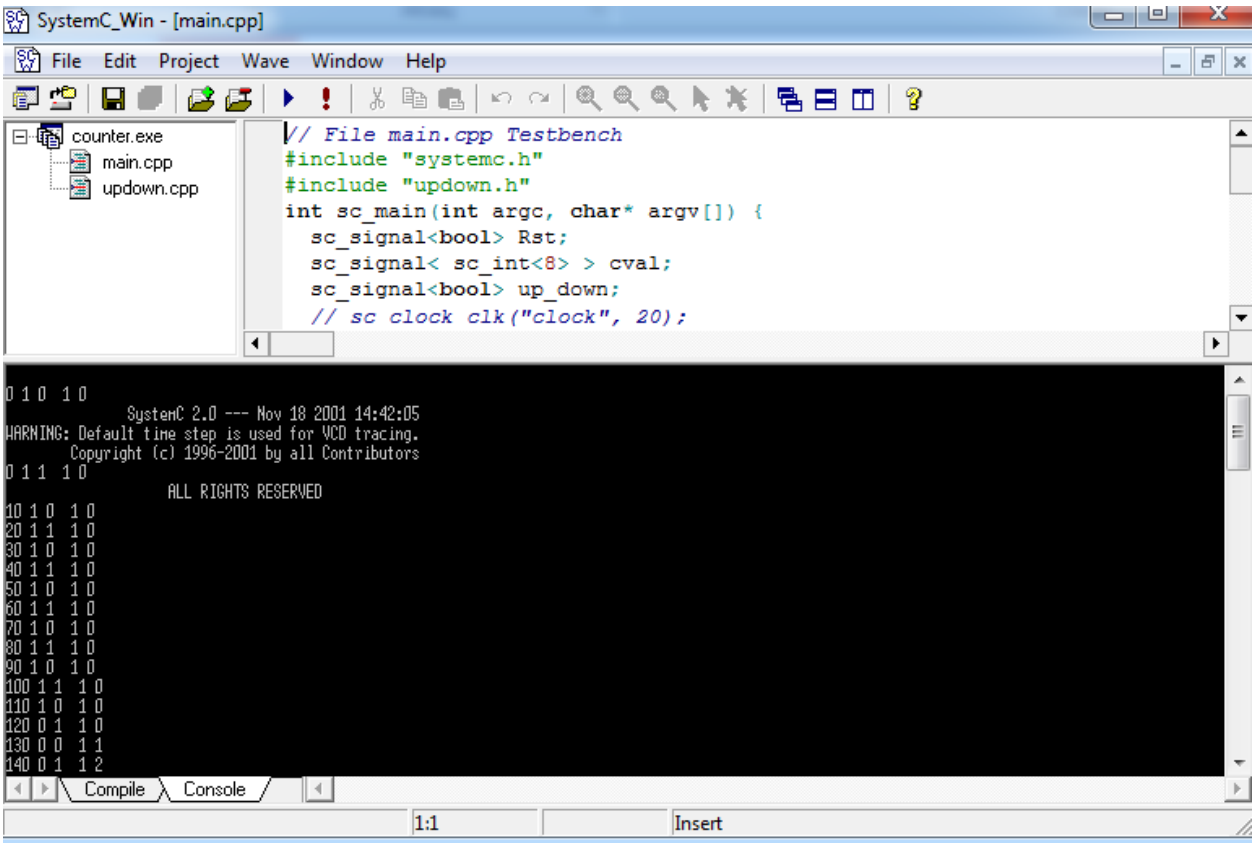
3. Відкриваємо C++ файли проекту та аналізуємо їх.



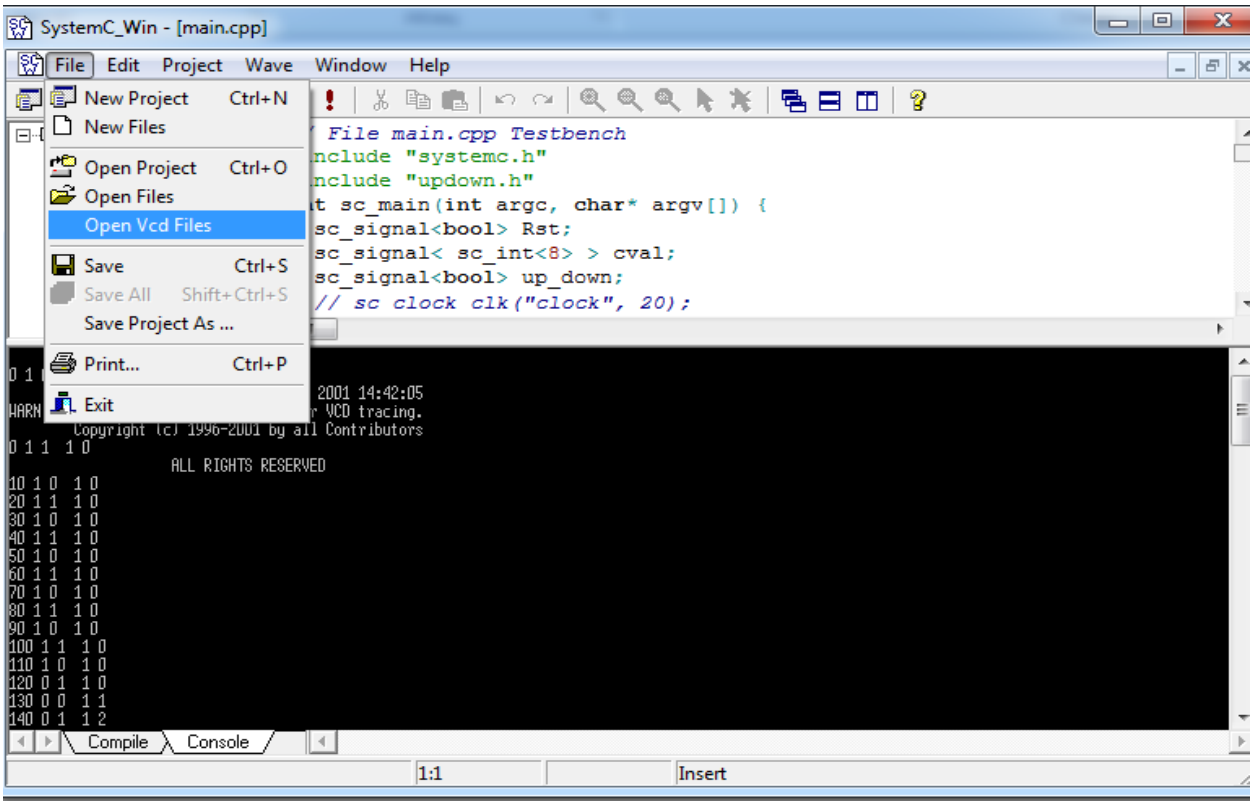
4. В меню вибираємо команду *Project/Run* та запускаємо проект на виконання.



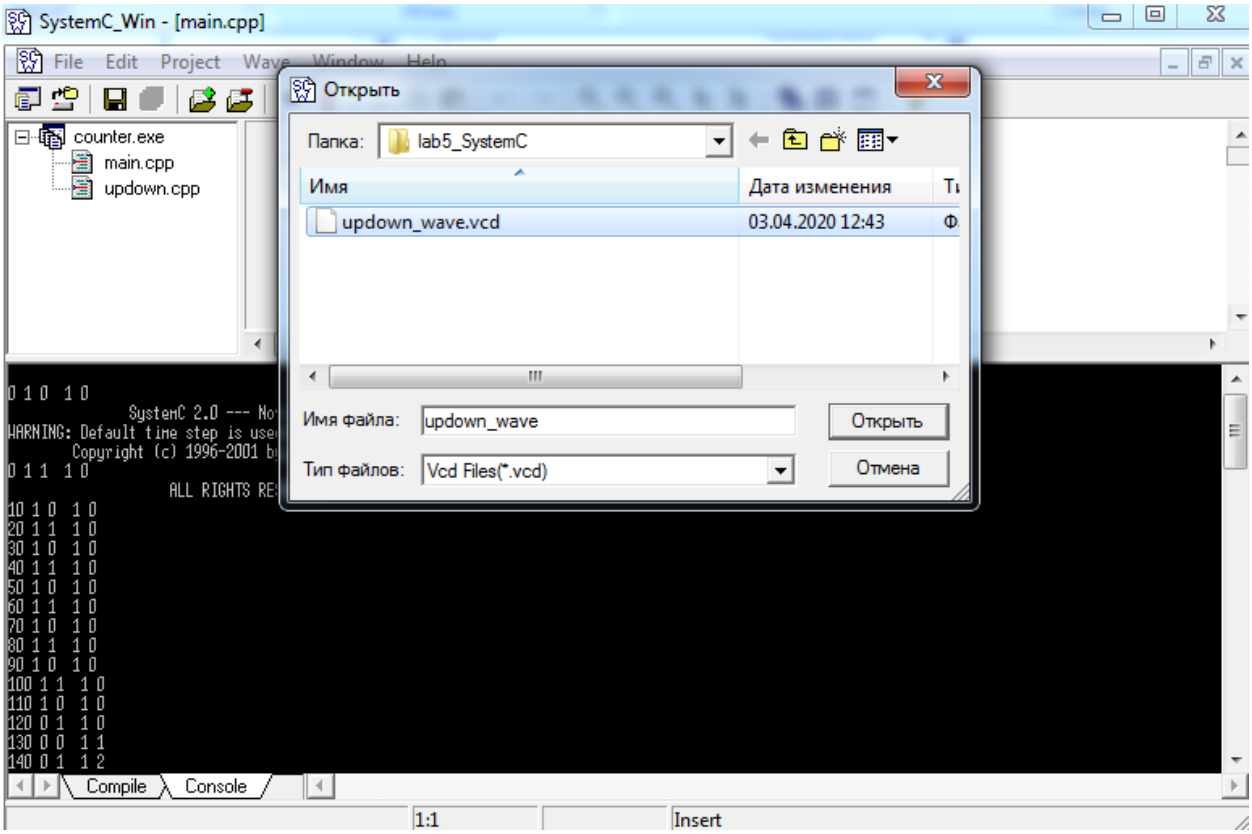
5. Спостерігаємо в консолі проекту результати роботи.



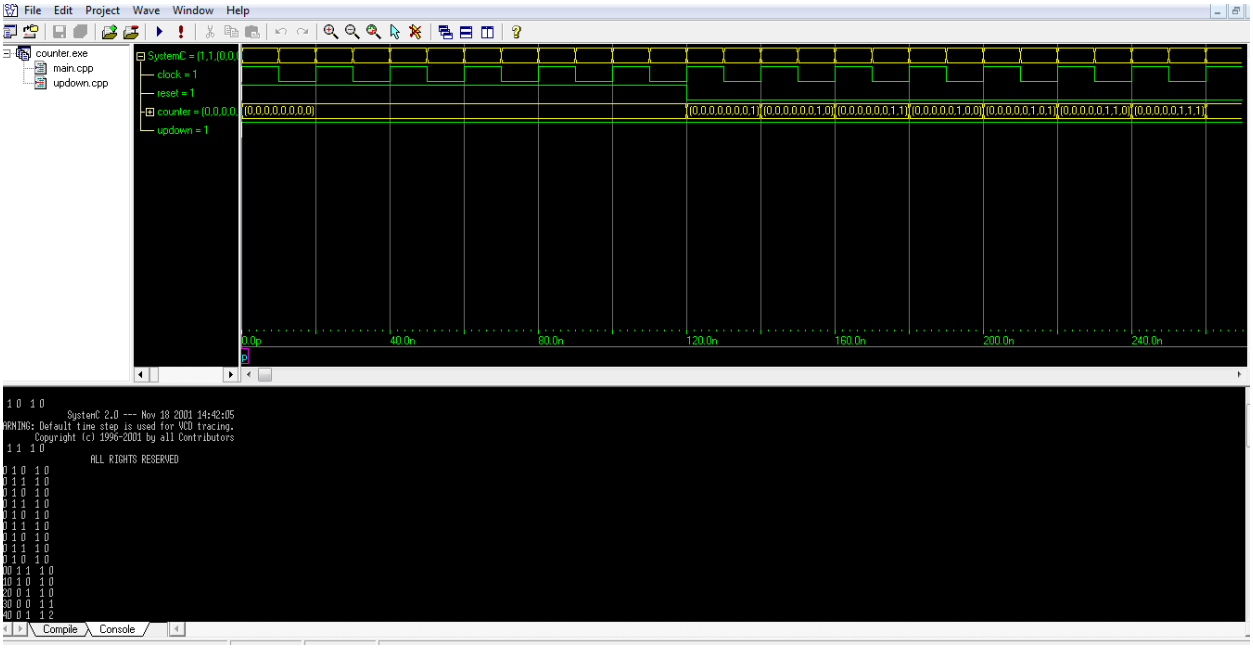
6. Наступним кроком в меню вибираємо File -> Open Vcd Files.



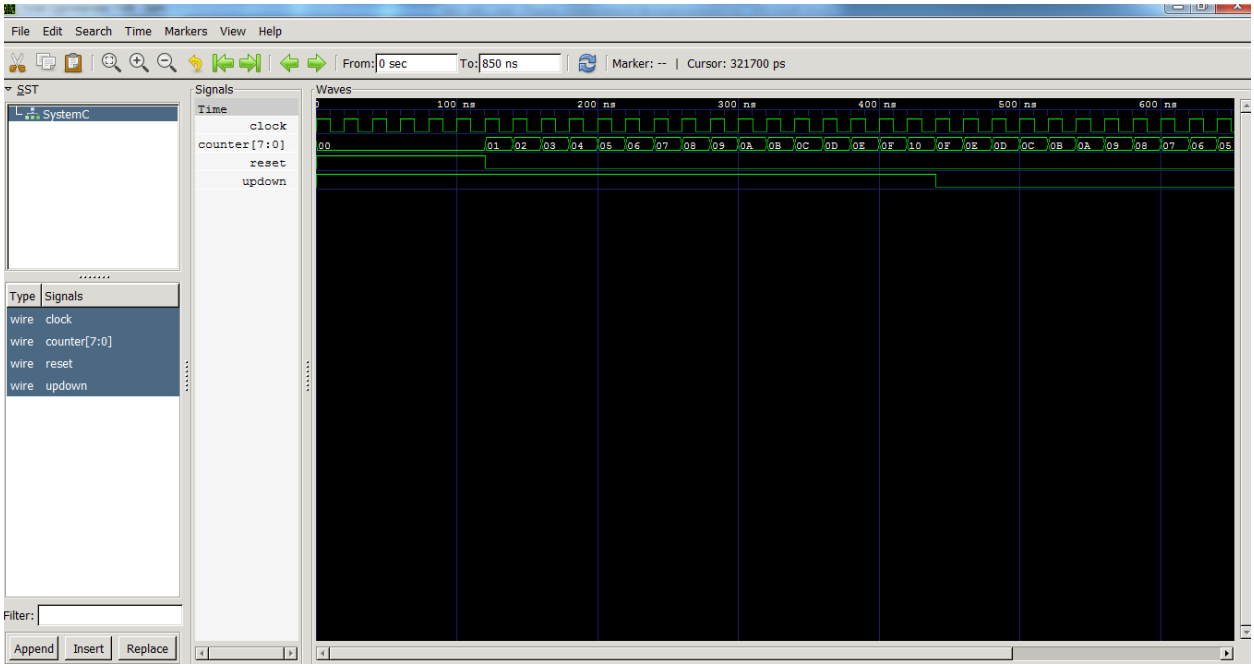
7. Відкриваємо файл updown_wave.vcd



8. Аналізуємо отриману діаграму роботи двійкового лічильника.



9. Запуск діаграми роботи лічильника у двох режимах в середовищі GTKwave.



Зміст звіту

1. Титульна сторінка.
2. Назва та мета лабораторної роботи.
3. Структура програми моделювання асинхронного лічильника.
4. Результати роботи моделювання лічильника.
5. Висновки.

Завдання до лабораторної роботи №3

Реалізуйте та промоделюйте роботу синхронного двійкового лічильника в Sysyem C.

№	Назва пристрою	Тип підрахунку	Розрядність
1	Двійковий лічильник	up	8
2	Двійковий лічильник	down	12
3	Двійковий лічильник	up	16
4	Двійковий лічильник	down	20
5	Двійковий лічильник	up	24

6	Двійковий лічильник	down	32
7	Двійковий лічильник	up	8
8	Двійковий лічильник	down	12
9	Двійковий лічильник	up	16
10	Двійковий лічильник	down	20
11	Двійковий лічильник	up	24
12	Двійковий лічильник	down	32

Контрольні запитання

1. Що таке SystemC?
2. Назвіть характеристики бібліотеки SystemC?
3. Які основні модулі системи SystemC_Win?
4. Що таке лічильник?
5. Які бібліотеки SystemC використовуються в проекті my_count?
6. Назвіть основні файли проекту та їх призначення?

ЛАБОРАТОРНА РОБОТА №4

з дисципліни "Комп'ютерні технології в наукових дослідженнях"

Тема: технології створення простих проектів в середовищі розробки програмовних логічних пристроїв Quartus II фірми Altera (Intel).

Мета: отримати навики роботи при створенні проекту в інтегрованому середовищі Quartus II.

Короткі теоретичні відомості

Програмне забезпечення Altera Quartus II надає багатофункціональне середовище проектування, яке може бути легко переналаштоване під конкретні вимоги. Це ідеальне середовище для проектування на основі ПЛІС закінчених систем на кристалі (SOPS). Програмне забезпечення Quartus II містить засоби для всіх фаз проектування із застосуванням ПЛІС, як FPGA так і CPLD структур. На рис. 3 зображено загальну структурну схему проектування в середовищі Quartus II.

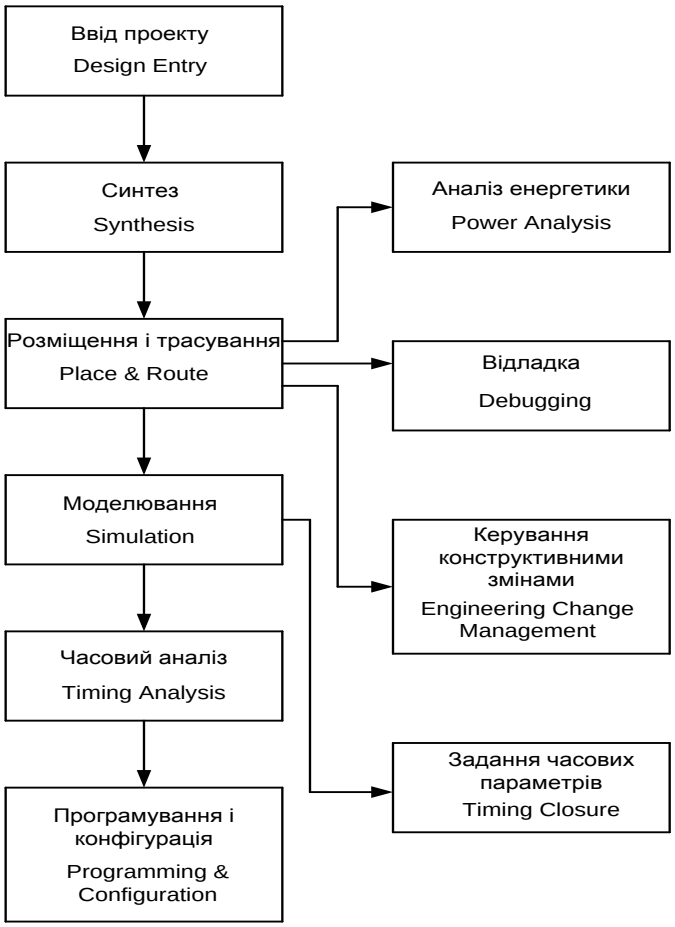


Рис. 1. Структурна схема проектування в середовищі Quartus II.

Наступна послідовність дій описує типовий порядок проектування в середовищі Quartus II за допомогою графічного інтерфейсу користувача:

1. Створення нового проекту, вказівка типу використовуваної мікросхеми або сімейства за допомогою команди New Project Wizard (меню File).
2. Створення вихідного файлу проекту на мовах Verilog HDL, VHDL, або Altera Hardware Description Language (AHDL) за допомогою текстового редактора (Text Editor). Крім того, можна створити блок-схему проекту в графічному редакторі (Block Editor) використовуючи символи, що представляють інші вихідні файли проекту або логічні елементи. За допомогою команди MegaWizard® Plug-In Manager (менюTools) можна створювати різні варіанти мегафункцій і IP-ядер для включення їх в файл проекту.
3. Вказівка початкових налаштувань проекту за допомогою редактора призначень (Assignment Editor), діалогового вікна Settings (меню Assignments), редактора топології (Floorplan Editor), і / або застосовуючи фіксовані логічні блоки (LogicLock ™).
4. Створення проекту на системному рівні за допомогою генератора систем на кристалі (SOPC Builder) або генератора систем ЦГЗ (DSP Builder).
5. Створення програмних файлів для процесорного ядра Nios® за допомогою редактора програмного забезпечення (Software Builder).
6. Синтез проекту за допомогою модуля аналізу і синтезу (Analysis & Synthesis).
7. Виконання функціонального моделювання проекту за допомогою симулятора (Simulator) і команди Generate Func tional Simulation Netlist.
8. Виконання розміщення і трасування проекту за допомогою модуля трасування (Fitter).
9. Проведення попереднього аналізу споживаної потужності за допомогою програми PowerPlay Power Analyzer.
10. Проведення аналізу тимчасових затримок проекту за допомогою програми аналізатора тимчасових затримок (Timing Analyzer).
11. Виконання моделювання проекту з урахуванням тимчасових затримок за допомогою симулятора.
12. Поліпшення тимчасових характеристик проекту за допомогою повторного фізичного синтезу, використання фіксованих логічних блоків, налаштувань в діалоговому вікні Settings і в редакторі призначень.
13. Створення файлу для програмування мікросхеми за допомогою модуля асемблера (Assembler).

14. Програмування мікросхеми за допомогою утиліти вибору програм (Programmer) і обладнання Altera; або перетворення формату файлу для програмування.
15. Налагодження проекту за допомогою вбудованого логічного аналізатора (SignalTap® II Logic Analyzer), генератора контрольних точок (SignalProbe ™).

Основною робочою одиницею в середовищі Quartus II є проект. Він створюється за допомогою спеціальної утиліти (New Project Wizard меню File). При створенні нового проекту задається робоча директорія, призначається ім'я проекту і ім'я файлу верхнього рівня ієрархії. Додатково можна вказати вихідні файли проекту, призначені для користувача бібліотеки, використовувані САПР сторонніх фірм, вбрання сімейство мікросхем (або використовувати автоматичний вибір сімейства компілятором Quartus II).

Порядок виконання роботи.

Крок 1: Створення нового проекту для використання в циклі лабораторних робіт.

Запустіть САПР Quartus II. В меню Пуск Виберіть Всі програми-> Altera -> Quartus II 7.2 .-> Quartus II 7.2 (32-Bit), для запуску.

1. Викличте утиліту New Project Wizard. В меню File Виберіть New Project Wizard .... Відкриється нове вікно. Якщо на екрані з'явилося вікно Introduction, натисніть Next.

2. Приступите до створення проекту за допомогою утиліти New Project Wizard. У вікні 1 (рисунок 1) задайте інформацію, представлену в таблиці 1.

Таблиця 1. Параметри вікна 1 New Project Wizard

Робоча директорія проекту	<*Lab_install_directory*> \ Pipemult \
ім'я проекту	pipemult
Файл верхнього рівня ієрархії	pipemult

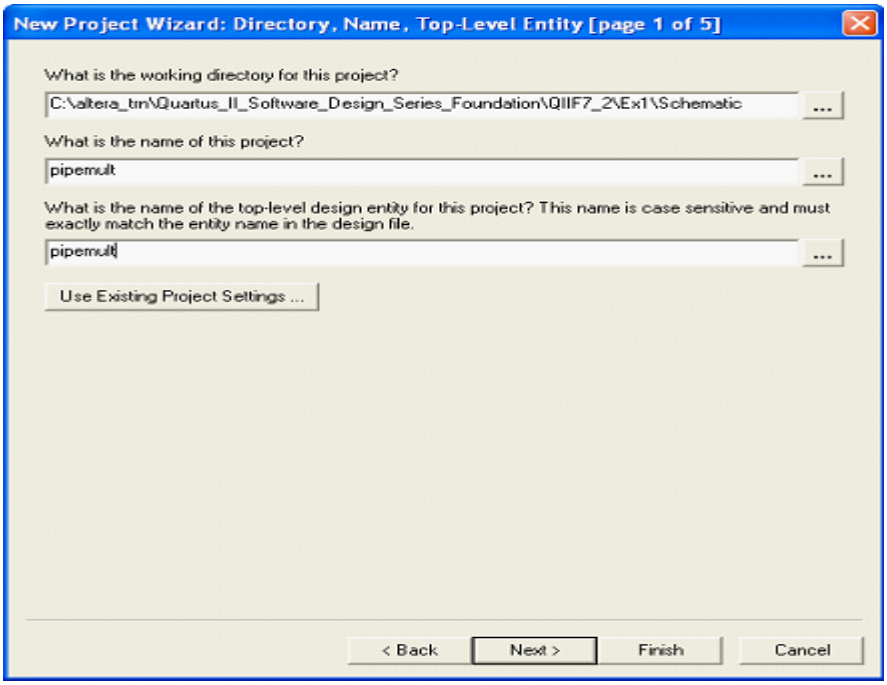


Рис. 2. Вікно створення проекту.

4. Натисніть **Next,** щоб перейти на наступне вікно.

5. У вікні 2 (рис. 2) натисніть кнопку (...) і виберіть файл верхнього рівня ієрархії **pipemult.bdf.** Він розташовується в робочій директорії проекту. натисніть **Open,** потім **Add,** щоб додати файл в проект. натисніть **Next.**

6. У вікні 3 (рис. 3), виберіть сімейство **(Family) Cyclone II.** У правій частині вікна, в розділі **Show in 'Available device' list,** встановіть наступні значення: в рядку **Package** виберіть **FBGA, Pin count** виберіть **256, Speed grade** виберіть **Fastest.** Ці настройки обмежують список доступних мікросхем. У вікні **Available devices** виберіть мікросхему**EP2C5F256C6.** натисніть **Next.**

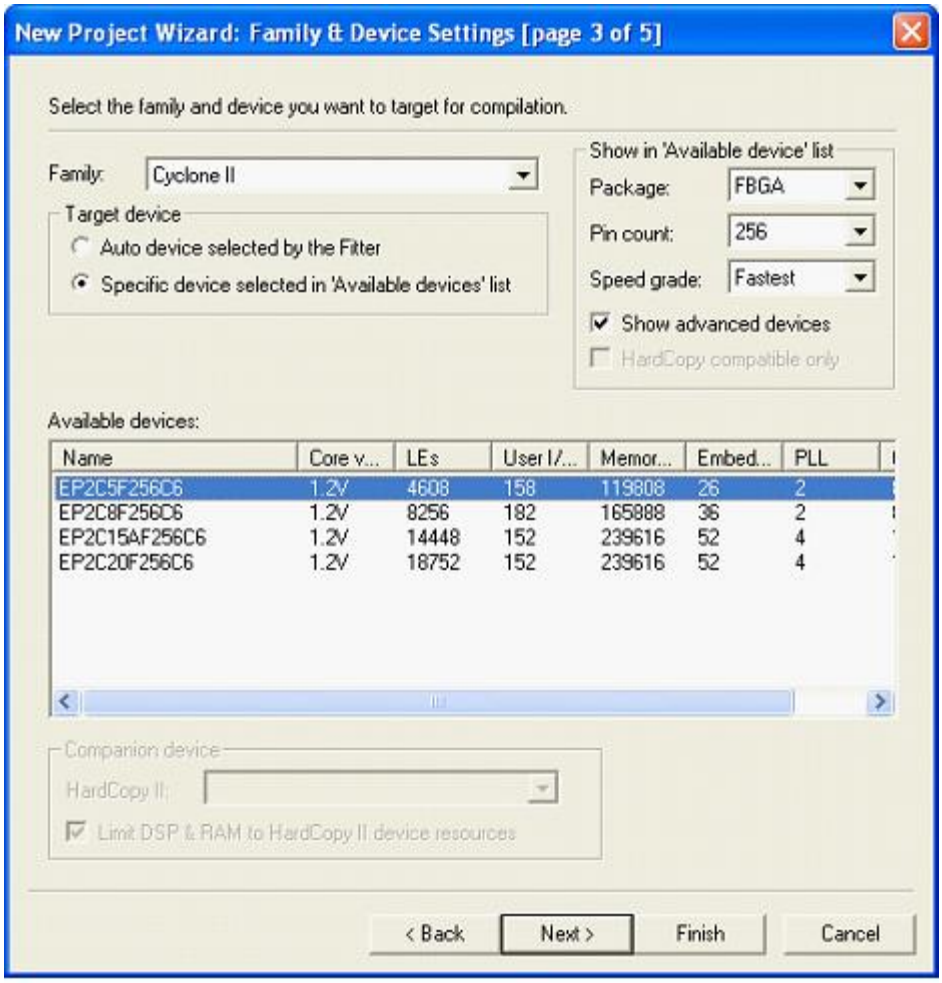


Рис. 3. Вікно вибору сімейства архітектури пристрою.

7. У вікні4 (рис. 4), ви можете вказати додатково використовуються САПР сторонніх виробників. В даному циклі вправ ми працюємо тільки в середовищі **Quartuis II.** натисніть **Next** для продовження.

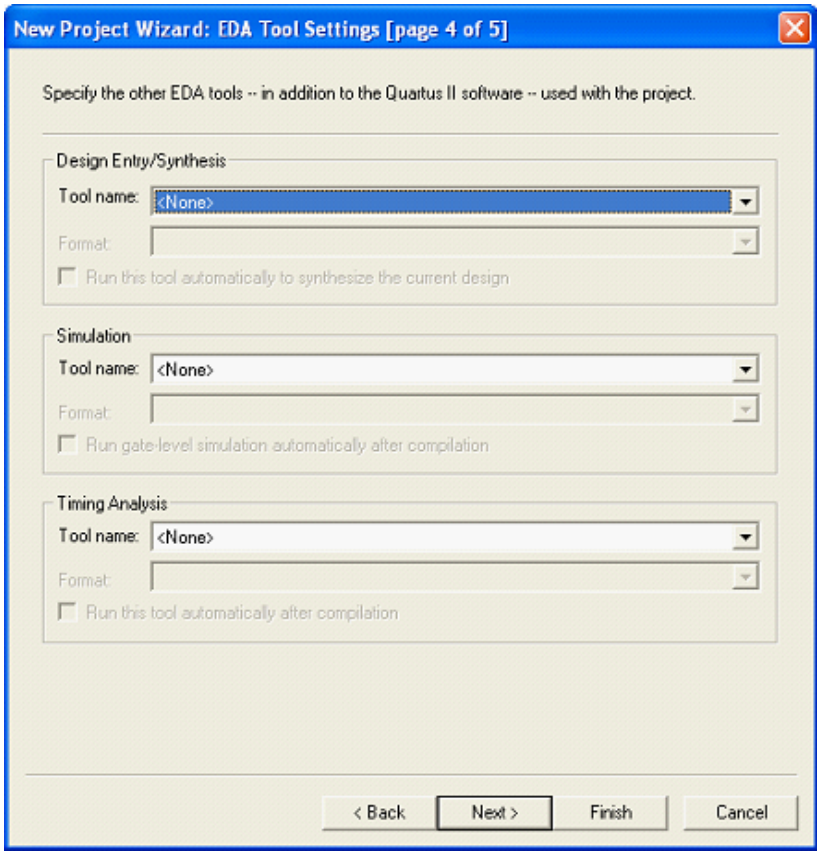


Рис. 4. Вікно вибору додатково використовуваних САПР.

8. На підсумковій сторінці (рис. 5) натисніть **Finish.** Проект створений.

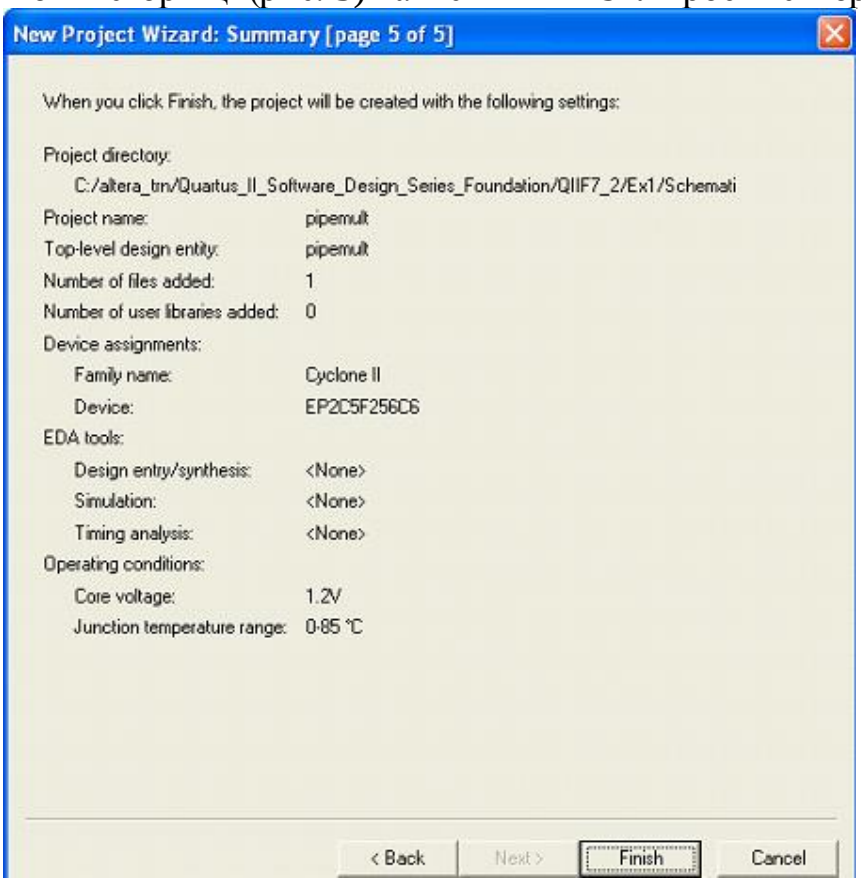


Рис. 5. Вікно завершення створення проекту.

Виходити з середовища Quartus II немає необхідності, якщо ви збираєтеся продовжити виконання вправи. Закритий проект завжди модно відкрити за допомогою команди File -> Open Project. команда File -> Open дозволяє відкрити окремий файл (замість проекту), запобігаючи виконання різних дій, пов'язаних з обробкою проекту, наприклад компіляцію.

Розробка конвеєрного помножувача.

На рисунку 6 зображено схематичне представлення файлу верхнього рівня ієрархії, який ви повинні реалізувати. Він складається з помножувача і блоку оперативної пам'яті RAM. Дані подаються на помножувач від зовнішнього джерела, а результат зберігається в блоці пам'яті, який теж управляється від зовнішнього джерела. Потім дані читаються з блоку оперативної пам'яті, використовуючи незалежну шину адреси для читання.

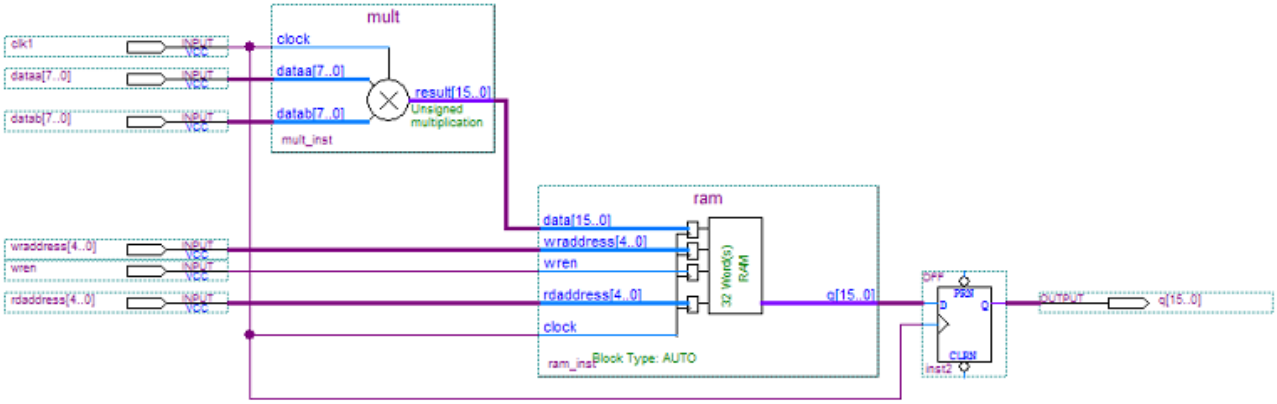


Рис. 6. Конвеєрний помножувач.

Крок 1: Створіть помножувач 8x8 за допомогою утиліти MegaWizard® Plug-in Manager.

Виберіть Tools -> MegaWizard Plug-In Manager. У вікні, виберіть опцію Create a new custom megafunction variation. натисніть Next.

2.Виберіть потрібну мегафункцію. У вікні2 (рис. 7), розширте каталог **Arithmetic** і виберіть **LPM_MULT.**

3. У випадаючому меню праворуч вкажіть використовується сімейство мікросхем **Cyclone II.**

Вибір сімейства мікросхем дозволяє утиліті MegaWizard Plug-In Manager визначити її доступні ресурси і доступні мегафункції для нього. Можна створити мегафункцію для іншого сімейства мікросхем, при цьому необхідно вибрати відповідне сімейство і створити для нього новий проект.

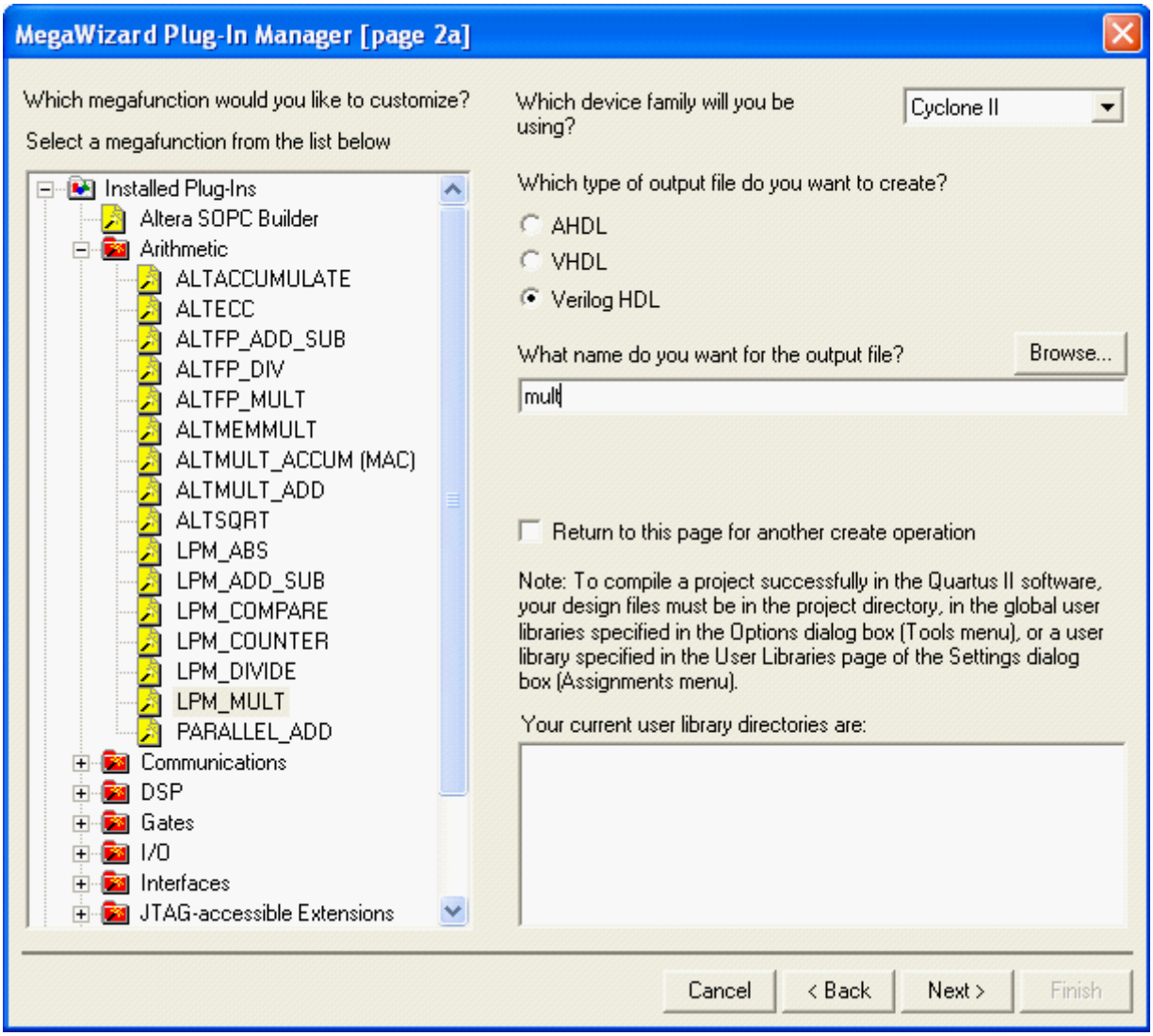


Рис. 7. Вікно каталогу **Arithmetic.**

4. Виберіть мову опису інтерфейсу мегафункції (VHDL або Verilog HDL) і вкажіть робочу директорію, де буде збережена мегафункція. Якщо ви розробляєте схемотехнічний проект, виберіть VHDL або Verilog за бажанням.

5. Задайте ім'я вихідного файлу - mult. Ви можете додати його в кінці обраного шляху до потрібної директорії або не вказувати повний шлях для автоматичного збереження файлу в робочій директорії проекту.

6. Натисніть Next.

7. У вікні3 (General) (рис. 8) вкажіть розрядність вхідних шин даних dataa і datab (вони можуть бути встановлені автоматично). У нашому приклад це 8 біт. Натисніть Next.

8. У вікні4 **(General 2)** (рис. 9) залиште всі настройки за замовчуванням **(datab** не є константою, використовується беззнакове множення і спосіб реалізації помножувача вибирається за замовчуванням). натисніть **Next.**

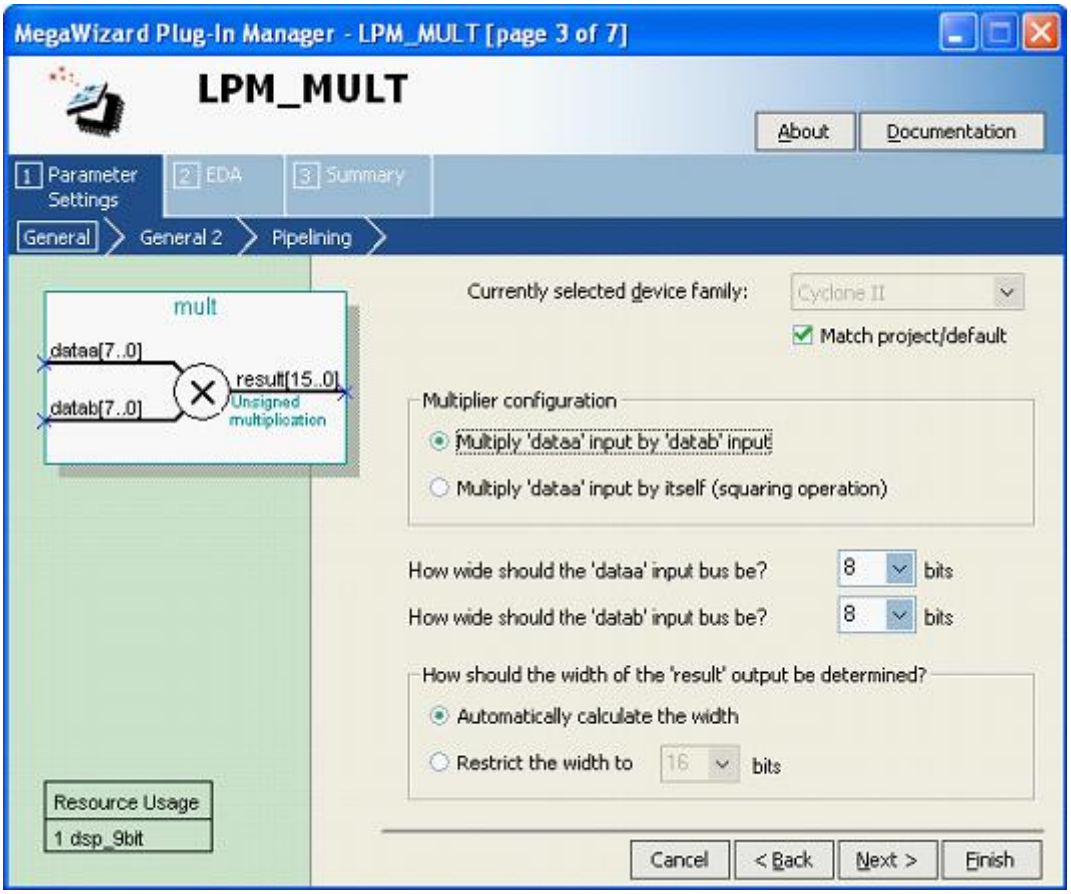


Рис. 8. Вікно вибору розрядності вхідних шин.

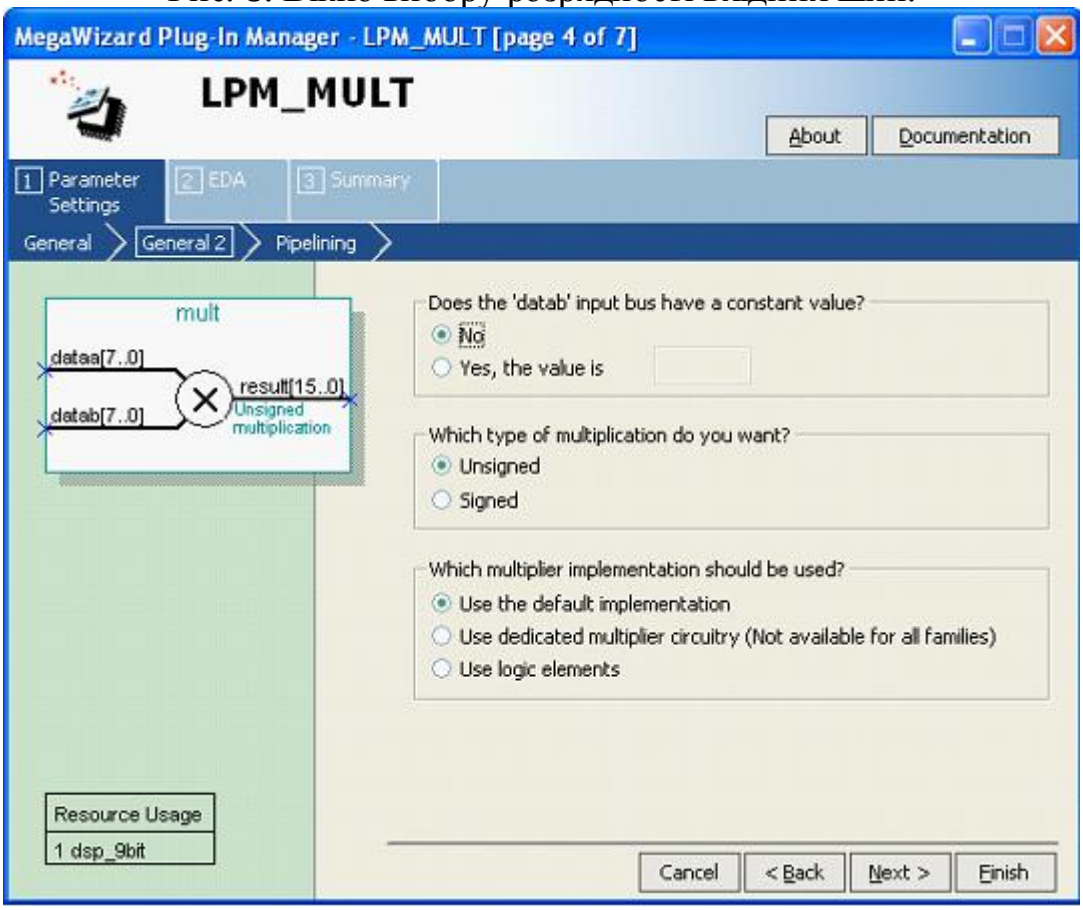


Рис. 9. Вікно вибору способу реалізації помножувача.

9. У вікні5 **(Pipelining)** (рис. 10) Виберіть **Yes, I want an output latency of 2 clock cycles** (затримка видачі результату на 2 циклу тактової частоти). натисніть **Next.**

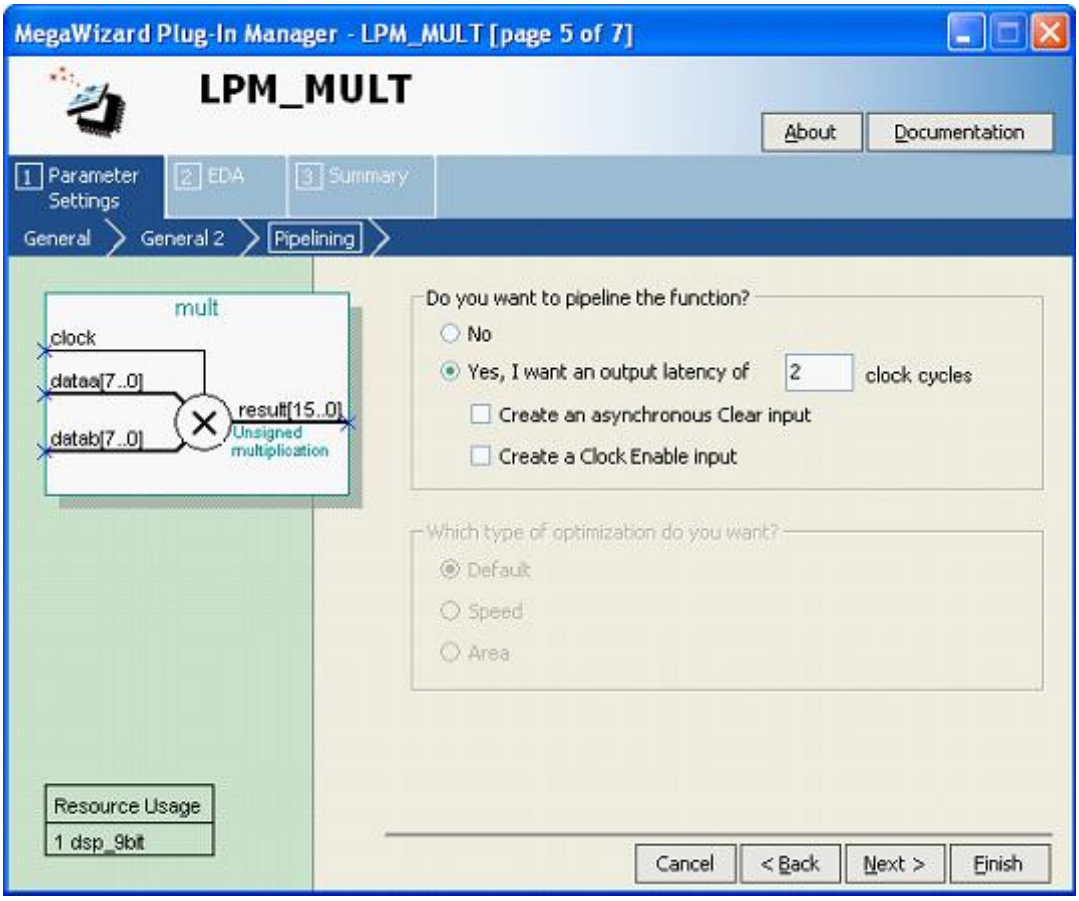


Рис. 10. Вікно вибору затримка видачі результату.

10. Далі потрібно повинні перейти на сторінку 6 (друга закладка - **EDA)** (рис. 11)**.** Тут вказується файл **Lpm,** який використовується сторонніми САПР для моделювання мегафункції **LPM_MULT** сторонні САПР застосовуватися не будуть тому потріно натиснути **Next.**

11. У вікні7 (рис. 12) необхідно відзначити створювані файли згідно з таблицею 2.

Таблиця 2. Приклад вибору параметрів для створюваного файлу

Тип вихідного файлу проекту	**Файли, які необхідно створити в MegaWizard Plug-In**
Schematic	mult (.vhd or .v) & mult.bsf

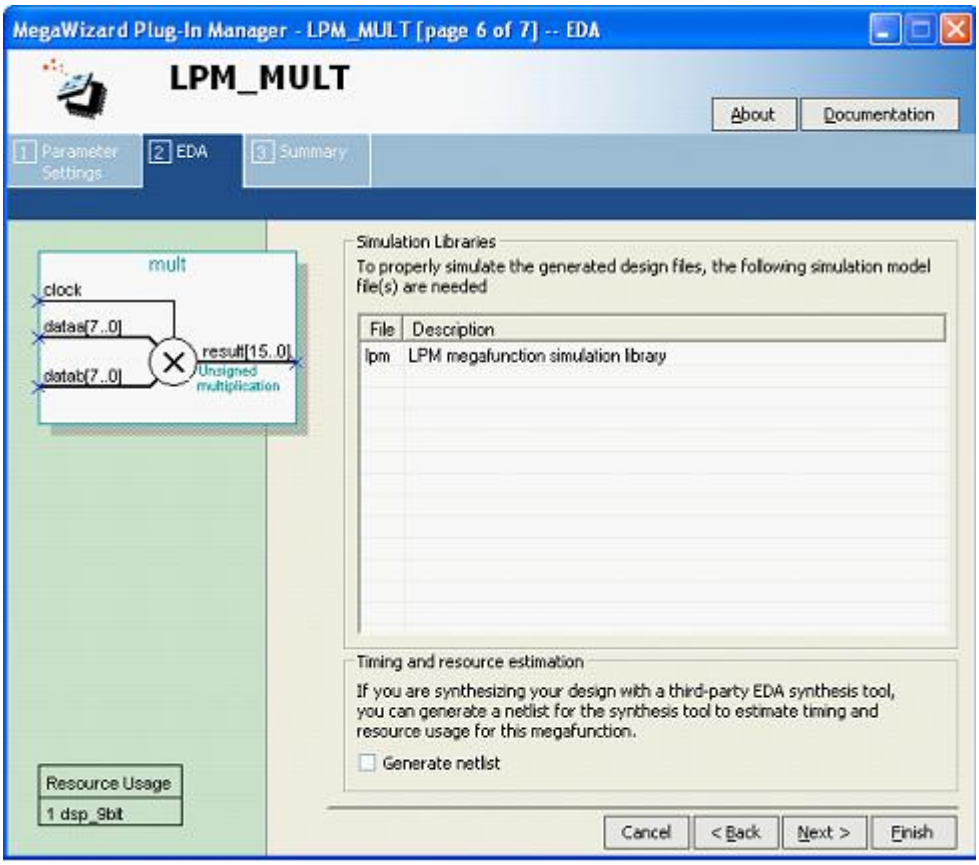


Рис. 11. Вікно вибору файлу **Lpm.**

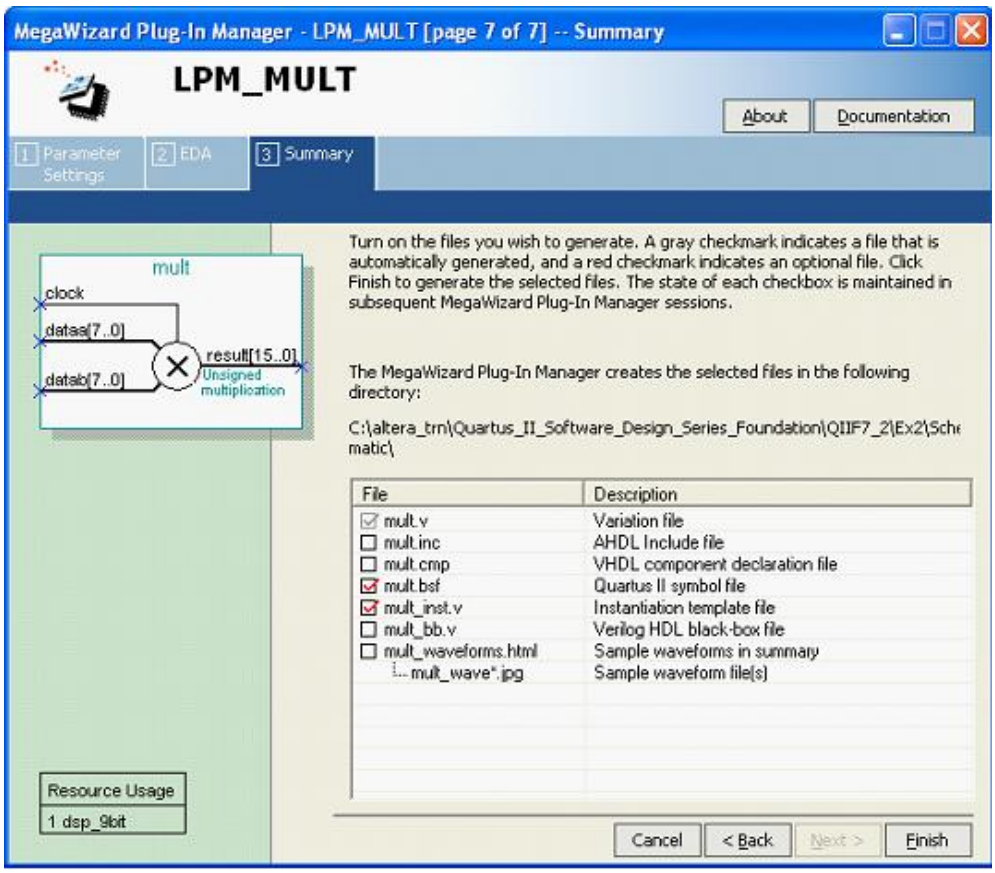


Рис. 12. Вікно вибору параметрів для створюваного файлу.

12. Натисніть **Finish** для завершення створення мегафункції.

Помножувач створений.

Якщо, з деяких причин, мегафункція створена некоректно або Ви щось пропустили, відкрийте знову MegaWizard Plug-In Manager в меню Tools. Виберіть опцію редагування мегафункції і виправте помилки. Натисніть Finish.

Крок 2:

Створіть 32 × 16 RAM за допомогою утиліти MegaWizard Plug-In Manager

1. Ще раз відкрийте MegaWizard Plug-In Manager (Tools -> MegaWizard Plug-In Manager). Виберіть Create a new custom megafunction variation та натисніть Next.

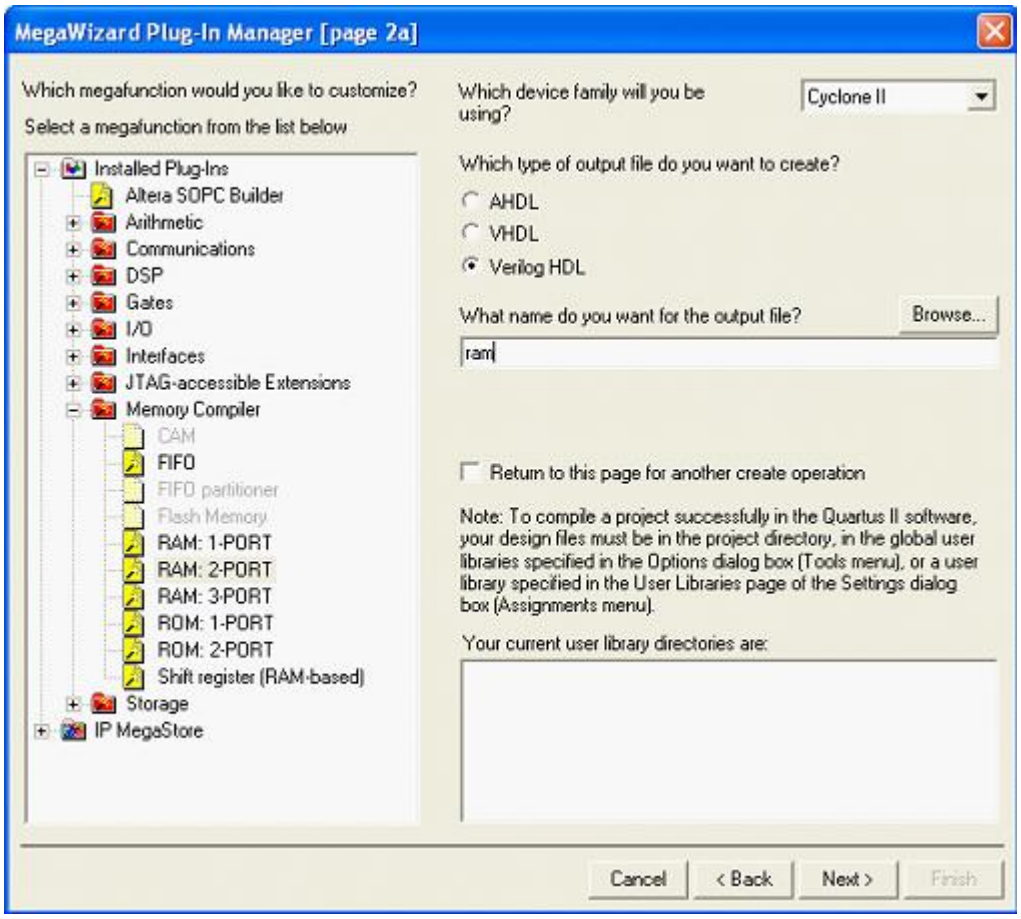


Рис. 13. Вікно вибору мегафункції.

2. Виберіть відповідну мегафункцію. У вікні2 (рис. 13) розкрийте папку **Memory Compiler** і виберіть **RAM: 2-PORT.**

3. Як і для реалізації помножувача, виберіть сімейство мікросхем **Cyclone II** і мову файлу опису мегафункції - **VHDL** або **Verilog HDL.**

4. У рядку імені вихідного файлу вкажіть **ram.**

5. Натисніть **Next.**

6. У вікні3 (рисунок 14), виберіть **With one read port and one write port,** в розділі вказівки способу використання двухпортовой пам'яті. Інші налаштування залиште за умовчанням (розмір пам'яті визначається кількістю слів). натисніть **Next.**

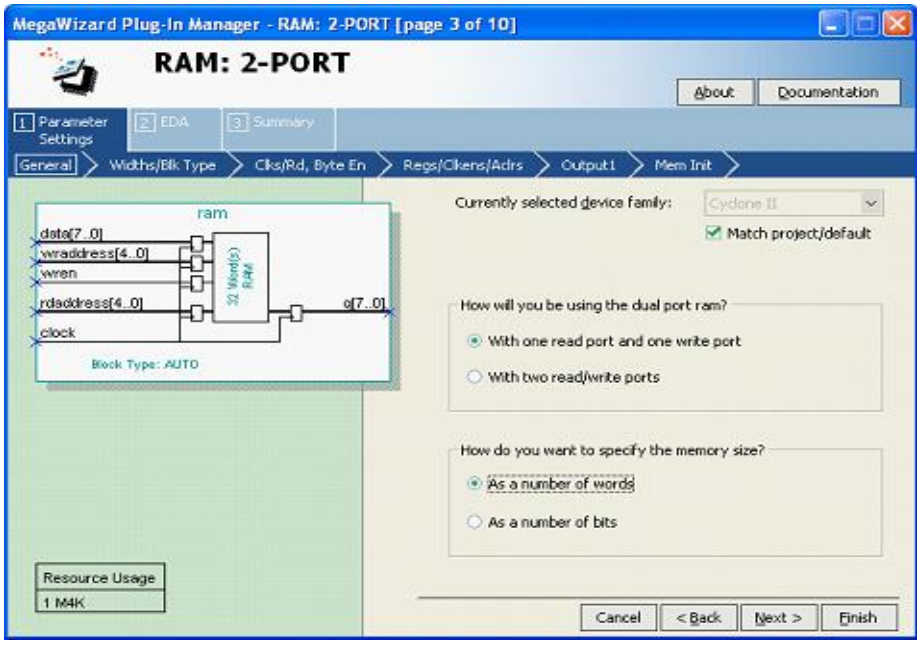


Рис. 14. Вікно вибору способу використання двохпортової пам'яті.

7. У вікні4 **(Widths / Blk Type)** (рис. 15), встановіть розрядність вхідного порту data_a рівну 16 біт (в розділі **Read / Write Ports).** Потім виберіть 32 - кількість 16-розрядних слів пам'яті**(16-bit words of memory).** Примітка: ви будете бачити напис "8-bit words of memory" до тих пір, поки не змініте значення розрядності вхідного порту. Інші налаштування залиште за умовчанням. натисніть **Next** 2 рази.

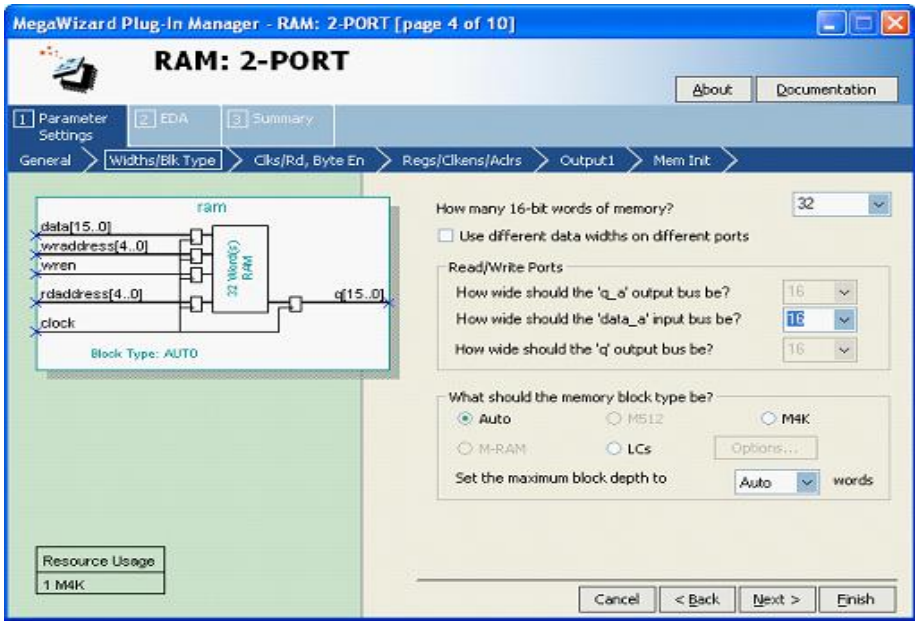


Рис. 15. Вікно вибору розрядності вхідного порту

8. У вікні6 **(Regs / Clkens / Aclrs)** (рисунок 16), відключіть опцію **Read output port (s) 'q',** щоб виключити установку вихідних регістрів для порту **q.** Інші налаштування залиште за умовчанням і натисніть **Next** 2 рази.

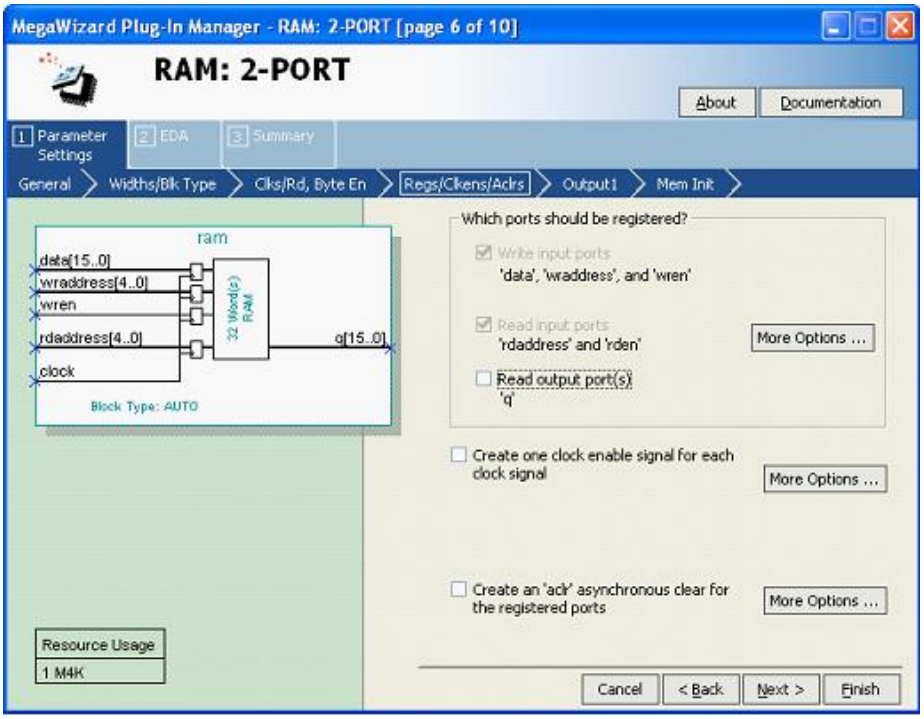


Рис. 16. Вікно настроювання параметрів вихідних регістрів.

9. У вікні8 **(Mem Init)** (рис. 17) виберіть **Yes,** щоб вказати файл ініціалізації блоку пам'яті. Після того, як поле стало доступним, впишіть ім'я файлу: **ram.hex.** (створення цього файлу відбудеться пізніше) та натисніть **Next.**

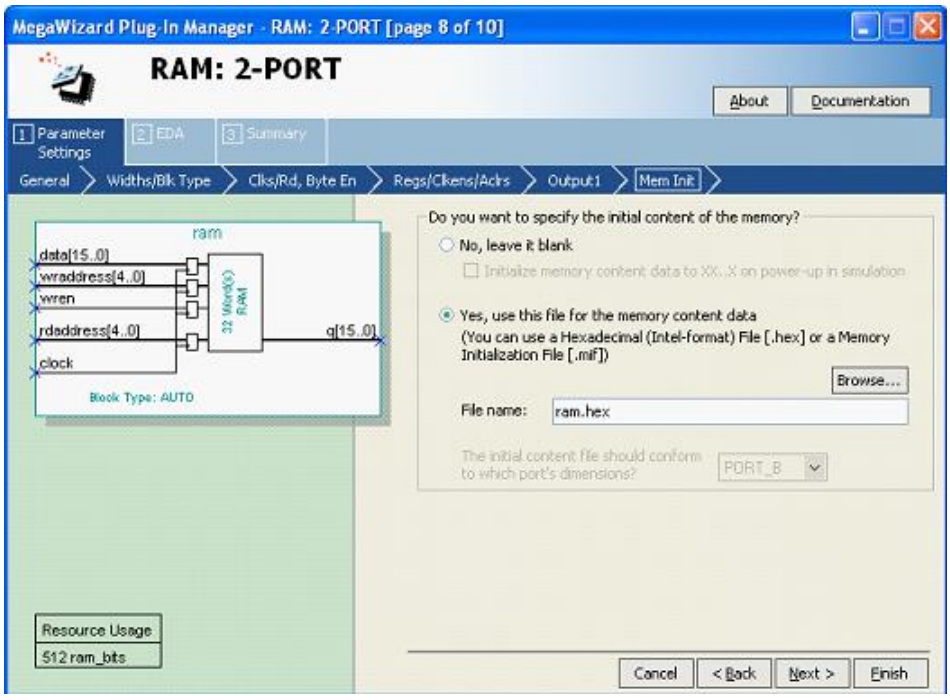


Рис. 17. Вікно ініціалізації блоку пам'яті

10. У вікні11 відображається файл **altera_mf,** необхідний для моделювання цієї мегафункції за допомогою сторонніх САПР. натисніть **Next.**

11. Виберіть ті ж файли для створення мегафункції **ram,** які ви вибирали для мегафункції **mult** раніше (Крок 1, пункт 11).

Тепер є створені два компонента, необхідних для даного проекту. Далі необхідно створити HEX файл, що описує вміст для ініціалізації оперативної пам'яті RAM.

Крок 3: Створіть HEX файл за допомогою редактора Memory Editor

1. У меню **File** виберіть команду **New** або натисніть знак в робочій панелі.

2. У діалоговому вікні перейдіть до закладки **Other Files** і виберіть **Hexadecimal (Intel-Format) File** (рис. 19)**.** натисніть **OK.**

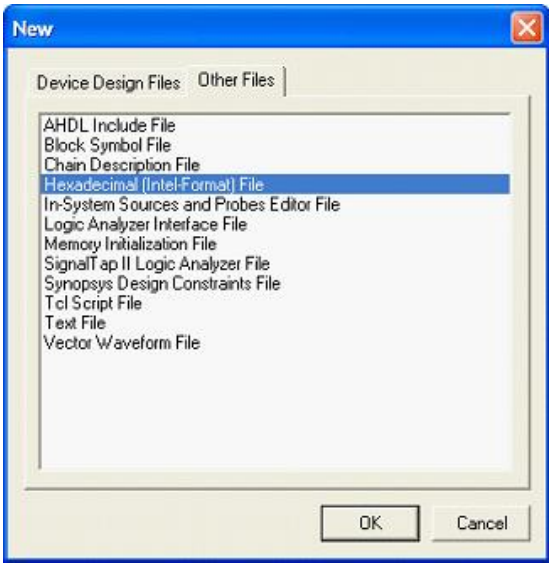


Рис. 19. Вікно створення HEX файлу.

3. У діалоговому вікні налаштувань розміру області пам'яті встановіть наступні значення: кількість слів - **number of words - 32,** розмір слова - **word size - 16.** та натисніть **Ok** (рис. 20).

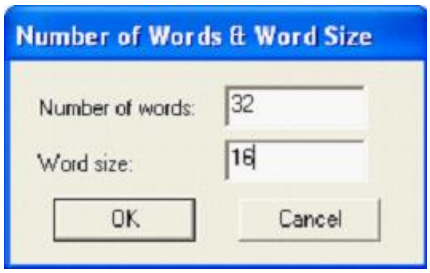


Рис. 20. Вікно налаштування розміру області пам'яті.

Вікно редактора пам'яті **Memory Editor** відображає заданий вами простір пам'яті. Якщо простір пам'яті відображається не так, як показано на рис. 21, можна змінити кількість осередків в ряду (меню **View)** на 16 і формат відображуваних значень на **Hexadecimal.**

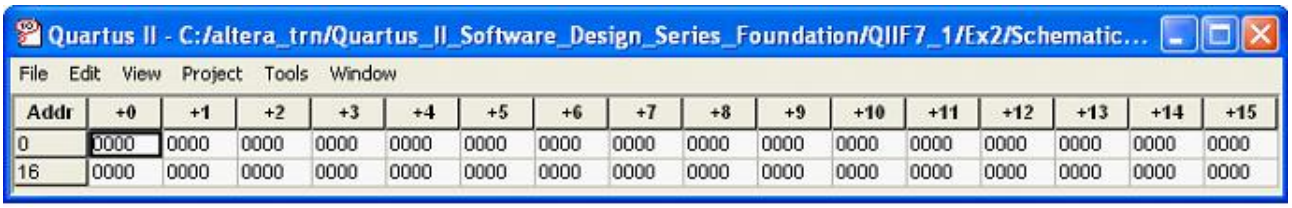


Рис. 21. Вікно простору пам'яті.

- Виділіть всю область пам'яті, натисніть праву кнопку миші і виберіть **Custom Fill Cells** в контекстному меню.
- За допомогою діалогового вікна **Custom Fill Cells,** введіть потрібні значення для ініціалізації області пам'яті. Можна вибрати одну з таких дій:
  - Повторювана послідовність: Введіть послідовність значень, які будуть повторюватися в пам'яті, відокремлюючи їх пропуском або комою.
  - Наростаюча/спадна послідовність: Введіть початкове значення і значення величини, на яку буде змінюватися кожне наступне значення (збільшуватися або зменшуватися).
- Збережіть файл як **ram.hex.** Закрийте його.

Додайте блоки в проект і створіть потрібні зв'язки.

1. Відкрийте файл **pipemult.bdf.** Можна використовувати команду **Open** в меню **File,** або натиснути значок на робочій панелі інструментів, або двічі натиснути на імені файлу в вікні навігатора проекту Project Navigator.

Це файл верхнього рівня ієрархії для даного проекту. Насправді, вам було б необхідно вибрати блоки **ram** і **mult** і виконати вручну їх з'єднання. В інтересах економії часу, для вас створений майже завершений файл, який не має блоків **ram** і **mult,** а також вихідної шини **q [15..0].**

2. Натисніть двічі лівою кнопкою миші в будь-якому вільному місці схемотехнічного вікна. У вікні **Symbol,** натисніть, щоб розкрити папку **Project.** Двічі натисніть лівою кнопкою миші на елементі **mult.** Потім натисніть ліву кнопку миші в полі креслення, щоб вставити вибраний елемент у вказане місце.

Примітка: Три вхідних порти в лівій частині помножувача повинні збігатися зі входами, присутніми на схемі. Якщо це не так, ви можливо, неправильно створили мегафункцію. У цьому випадку натисніть **Esc,** щоб скасувати вставку символу, відкрийте знову утиліту MegaWizard Plug-In Manager і виберіть опцію **Edit an existing megafunction variation.** Вкажіть елемент **mult** і виправте помилки. Повторіть установку символу в потрібному місці.

3. Правою клавішею миші натисніть на блоці **mult** і виберіть команду **Properties** в контекстному меню. У діалоговому вікні **Symbol Properties,** в рядку **Instance name,** змініть ім'я **inst** на **mult_inst.**

4. Натисніть **OK.**

5. Знову відкрийте вікно **Symbol.** Виберіть елемент **ram** і встановіть його в зазначене місце.

Примітка: 4 нижніх входи блоку **ram** повинні співпасти з вхідними портами. Вхід data повинен залишитися не підключеним.

6. Також, як і для блоку **mult,** для елемента **ram** відкрийте діалогове вікно **Symbol Properties** і змініть ім'я **inst** на **ram_inst.**

7. Відкрийте вікно **Symbol** і в рядку **Name** введіть ім'я елемента **output.** Елемент повинен визначитися автоматично.

8. Натисніть **OK** і встановіть елемент біля вихідного контакту регістра **inst2.** Двічі клацніть в області вказівки імені компонента і змініть **pin_name** на **q [15..0].**

9. Натисніть кнопку (виклик інструменту для створення шин), потім з'єднайте між собою вихід елемента **mult (Result)** і вільний вхід елемента **ram (Data).** В результаті ви повинні отримати схему, представлену на рисунку 22.

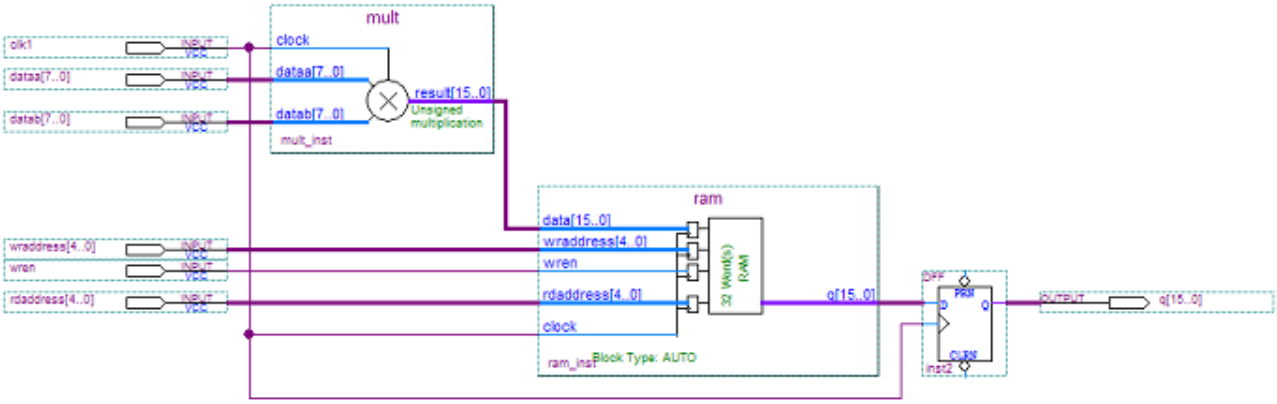


Рис. 22. Кінцева схема створюваного пристрою

10. Збережіть файл **pipemult.bdf.**

Збережіть, перевірте проект та оформіть звіт з відповідним описом проблем та помилок, які виникли в процесі створення проекту:

1. У меню **Processing** виберіть команду **Start -> Start Analysis & Elaboration.** Дана команда виконує перевірку наявності всіх файлів в проекті і правильність їх підключень.

2. Натисніть **OK,** коли аналіз завершиться. Якщо були виявлені будь-які помилки, перевірте всі зв'язки або повторно викличте MegaWizard Plug-In Manager для виправлення помилок в мегафункціях.

3. Оформіть звіт з відповідним описом проблем та помилок, які виникли в процесі створення проекту

Завдання до лабораторної роботи №4

Розробіть багаторозрядний конвеєрний помножувач та промоделюйте його роботу згідно наступної таблиці.

№	Тип пристрою	Розрядність вхідних даних, біт
1	Конвеєрний помножувач	8
2	Конвеєрний помножувач	12
3	Конвеєрний помножувач	16
4	Конвеєрний помножувач	20
5	Конвеєрний помножувач	24
6	Конвеєрний помножувач	32
7	Конвеєрний помножувач	64
8	Конвеєрний помножувач	16
9	Конвеєрний помножувач	32
10	Конвеєрний помножувач	64

Зміст звіту

1. Титульна сторінка.
2. Назва та мета лабораторної роботи.
3. Наведіть схему створеного пристрою та її програмний код.
4. Висновки.

Контрольні запитання

1.Алгоритм процесу проектування в средовищі Quartus II.

2. Вибір сімейства мікросхем за допомогою утиліти MegaWizard Plugin Manager.

3. Процес додавання нових блоків в проект і створення потрібних зв'язків.

4. Збереження та тестування створеного проекту.

ЛАБОРАТОРНА РОБОТА №5

Тема роботи: технології розробки програмовних систем на кристалі *PSoC Designer* фірми *Cypress Semiconductor*.

Мета роботи: засвоїти навики роботи в інтегрованому середовищі PSoC Designer та вивчити основні архітектурні особливості сімейства мікроконтролерів CY8C29X66.

Короткі теоретичні відомості

1.1. Особливості архітектури PSoC

PSoC (англ. *Programmable System-on-Chip*, програмована система на кристалі — програмована система, яка вміщує функціональні складові цілого пристрою на одному кристалі. На відміну від звичайних мікроконтролерів, крім процесорного ядра, PSoC має матрицю цифрових та аналогових блоків. Розробляється компанією Cypress Semiconductor. Завдяки конфігурованим аналоговим та цифровим блокам, стає можливим створення усередині мікросхеми PSoC таких функцій, як АЦП, ЦАП, компаратора, ФНЧ, температурного давача, аудіовиходу тощо.

Фірма Cypress MicroSystems запропонувала оригінальне сімейство мікроконтролерів з набором реконфігурированих аналогових і цифрових периферійних модулів архітектури PSoC (Programmable System on Chip). Контролер побудований на базі традиційного процесорного ядра M8C Гарвардської архітектури, що використовується фірмою Cypress упродовж багатьох років у контролерах клавіатури і мишки малої вартості. Це класичне ядро з CISC архітектурою на основі акумулятора, час виконання команд - від 4 до 15 циклів тактового генератора процесора при максимальній частоті ядра 24 МГц. Основних регістрів ядра - 5 (лічильник команд CPU_PC, акумулятор CPU_A, покажчик стека CPU_SP, індексний регістр CPU_X і регістр прапорців CPU_F).

Система команд підтримує 10 режимів адресації, характеризується високою щільністю коду й оптимізована для програмування мовою ассемблера. Слід зазначити високу щільність коду при програмуванні мовою ассемблер і неоптимальну систему команд для програмування на мовах високого рівня, зокрема на С.

Щодо традиційних інтегрованих периферійних пристроїв, то PSoC має їхній досить повний набір: сторожовий таймер/таймер сну, детектор зниження напруги живлення і скидання при включенні живлення і зниженні живильного напруги, один чи два апаратних помножувача з нагромадженням. Помножувач дозволяє виконувати множення двох знакових байтових цілих з формуванням 16 бітного результату і накопичувати результати множення в 32-бітному акумуляторі. Ця особливість робить зручним використання процесорів PSoC для деяких задач цифрової обробки сигналів. Апаратний дециматор і регістр послідовного наближення (SAR) призначені для побудови сігма-дельта і аналого-цифрових перетворювачів (АЦП) послідовного наближення відповідно. Вбудований підвищувальний імпульсний стабілізатор напруги здатний забезпечити запуск процесора при напрузі живлення від 1.0 В, що дуже зручно для систем з батарейним живленням (зауважимо, що максимальний струм, який можна одержати від цього перетворювача не перевищує 5-10 мА, а процесор характеризується не дуже високою економічністю, зокрема, струм споживання тільки ядра може досягати 8 мА при тактовій частоті ядра 3 МГц і напрузі живлення 5 В).

Система переривань - пріоритетна, з фіксованими пріоритетами джерел переривань. Кожна лінія порту вводу-виводу може генерувати переривання (вектор переривання загальний для різних портів вводу-виводу). Відзначимо недолік організації переривань від зовнішніх пристроїв, що виявляється у відсутності регістра прапорців, який би дозволив визначити, який саме біт порту згенерував переривання. При виконанні обробника переривання автоматично забороняються, але програміст може дозволити переривання вищих пріоритетів у коді обробника переривання.

Таблиця 1.

Основні характеристики архітектури PSoC

Тип	ROM	RAM	Кількість портів вводу-виводу	Цифрові блоки (тип)	Аналогові блоки	Напруга живлення
CY8C21123	4KB Flash	256bytes	6-8	2-Basic 2-Comms	4, Type "E"	2.4V до 5.25V
CY8C21223	4KB Flash	256bytes	12-20	2-Basic 2-Comms	4, Type "E"	2.4V до 5.25V
CY8C21234	8KB Flash	512bytes	12-16	2-Basic 2-Comms	4, Type "E"	2.4V до 5.25V
CY8C21323	4KB Flash	256bytes	16-24	2-Basic 2-Comms	4, Type "E"	2.4V до 5.25V
CY8C21334	8KB Flash	512bytes	16-20	2-Basic 2-Comms	4, Type "E"	2.4V до 5.25V
CY8C21434	8KB Flash	512bytes	28-32	2-Basic 2-Comms	4, Type "E"	2.4V до 5.25V
CY8C22113	2KB Flash	256bytes	6-8	2-Basic 2-Comms	3:1-CT 2-SC	3.0V до 5.25V
CY8C22213	2KB Flash	256bytes	16-20	2-Basic 2-Comms	3:1-CT 2-SC	3.0V до 5.25V
CY8C24123	4KB Flash	256bytes	6-8	2-Basic 2-Comms	6:2-CT 4-SC	3.0V до 5.25V
CY8C24223	4KB Flash	256bytes	16-20	2-Basic 2-Comms	6:2-CT 4-SC	3.0V до 5.25V
CY8C24423	4KB Flash	256bytes	24-28	2-Basic 2-Comms	6:2-CT 4-SC	3.0V до 5.25V
CY8C25122	4KB Flash	256bytes	6-8	4-Basic 4-Comms	12:4-CT 8-SC	3.0V до 5.25V
CY8C26233	8KB Flash	256bytes	16-20	4-Basic 4-Comms	12:4-CT 8-SC	3.0V до 5.25V
CY8C26443	16KB Flash	256bytes	24-28	4-Basic 4-Comms	12:4-CT 8-SC	3.0V до 5.25V
CY8C26643	16KB Flash	256bytes	40-44	4-Basic 4-Comms	12:4-CT 8-SC	3.0V до 5.25V
CY8C26443	16KB Flash	256bytes	44-48	4-Basic 4-Comms	12:4-CT 8-SC	3.0V до 5.25V
CY8C27143	16KB Flash	256bytes	6-8	4-Basic 4-Comms	12:4-CT 8-SC	3.0V до 5.25V
CY8C27243	16KB Flash	256bytes	16-20	4-Basic 4-Comms	12:4-CT 8-SC	3.0V до 5.25V
Y8C27243	16KB Flash	256bytes	16-20	4-Basic 4-Comms	12:4-CT 8-SC	3.0V до 5.25V
CY8C27243	16KB Flash	256bytes	16-20	4-Basic 4-Comms	12:4-CT 8-SC	3.0V до 5.25V
CY8C27443	16KB Flash	256bytes	24-28	4-Basic 4-Comms	12:4-CT 8-SC	3.0V до 5.25V
CY8C27443	16KB Flash	256bytes	24-28	4-Basic 4-Comms	12:4-CT 8-SC	3.0V до 5.25V
CY8C27443	16KB Flash	256bytes	24-28	4-Basic 4-Comms	12:4-CT 8-SC	3.0V до 5.25V
CY8C27466	32KB Flash	2K	24-28	4-Basic 4-Comms	12:4-CT 8-SC	3.0V до 5.25V
CY8C27543	16KB Flash	256bytes	40-44	4-Basic 4-Comms	12:4-CT 8-SC	3.0V до 5.25V
CY8C27566	32KB Flash	2K	40-44	4-Basic 4-Comms	12:4-CT 8-SC	3.0V до 5.25V
CY8C27643	16KB Flash	256bytes	44-48	4-Basic 4-Comms	12:4-CT 8-SC	3.0V до 5.25V
CY8C27666	32KB Flash	2K	44-48	4-Basic 4-Comms	12:4-CT 8-SC	3.0V до 5.25V
CY8C27866	32KB Flash	2K	64-100	4-Basic 4-Comms	12:4-CT 8-SC	3.0V до 5.25V
CY8C29466	32KB Flash	2K	24-28	8-Basic 8-Comms	12:4-CT 8-SC	3.0V до 5.25V
CY8C29566	32KB Flash	2K	40-44	8-Basic 8-Comms	12:4-CT 8-SC	3.0V до 5.25V
CY8C29666	32KB Flash	2K	44-48	8-Basic 8-Comms	12:4-CT 8-SC	3.0V до 5.25V
CY8C29866	32KB Flash	2K	64-100	8-Basic 8-Comms	12:4-CT 8-SC	3.0V до 5.25V

Пояснення до таблиці: у мікроконтролерах PSoC є 2 типи цифрових модулів (basic і communication) і 3 основних типи аналогових модулів - на основі операційних підсилювачів з резистивним зворотнім зв'язком і конденсаторами, що переключаються, 2 типи.

Відзначимо інші корисні особливості процесора архітектури PSoC:

- можливість внутрішньосхемного і самопрограмування;
- зручний механізм захисту обраних ділянок flash пам'яті від стирання, запису, зовнішнього читання;

- невеликий розмір сегмента flash пам'яті - 64 байта зручний при імітації EEPROM з повним індивідуальним захистом кожного сегмента;
- висока навантажувальна здатність цифрових і аналогових виходів - до 25 мА і 40 мА відповідно;
- можливість вибіркового використання внутрішніх резисторів, що підтягують, до ліній живлення і землі і можливість гнучкого налаштування в процесі роботи;
- 3 внутрішніх тактових генератори та вбудована гнучка система фазового автопідлаштування частоти. Високостабільний (погрішність 2.5 % у всьому промисловому температурному діапазоні) 24 МГЦ генератор дозволяє в багатьох випадках використовувати процесор без зовнішнього кварцового резонатора; внутрішній економічний 32 кГц генератор зручний для використання як генератор для сторожового таймера; 32.768 кГц генератор із кварцовою стабілізацією частоти призначений для прикладних програм, що вимагають підвищеної стабільності тактової частоти;

Відзначимо, що генератор із кварцовою стабілізацією частоти дуже чутливий до зовнішніх перешкод, що змушує застосовувати спеціальні міри для забезпечення усталеної роботи генератора (захисні кільця на друкованій платі, відмовлення від використання сусідніх висновків процесора).

Але найбільш корисні й унікальні особливості процесорів архітектури PSoC складаються в наявності реконфігурованих аналогових і цифрових модулів. Мікроконтролери PSoC мають у своєму складі аналогові і цифрові модулі. Існує три типи аналогових модулів: на основі операційних підсилювачів з резистивним зворотним зв'язком і два типи модулів на основі конденсаторів, що комутуються.

На основі аналогових модулів можуть бути реалізовані фільтри різних типів, змішувачі сигналів, підсилювачі з перемінним коефіцієнтом підсилення, аналого-цифрові і цифро-аналогові перетворювачі різних типів, компаратори. Відзначимо, що в аналогових модулях на основі операційних підсилювачів з резистивним зворотним зв'язком присутній низькоспоживаючий компаратор (15 u), спеціально призначений для „пробудження” мікроконтролера з режиму сну. Мікроконтролер

PSoC має 4,8,16 (у залежності від моделі) цифрових модулів двох типів, на основі яких можуть бути реалізовані лічильники і таймери різної розрядності (8-32), генератори широтно-імпульсних сигналів, асинхронні і синхронні послідовні приймач/передавач, приймач/передавач даних по протоколі IrDA, генератори псевдовипадкових послідовностей, апаратні схеми формування циклічних контрольних сум (CRC).

Функції кожного аналогового і цифрового модуля визначаються вмістом відповідних регістрів керування. Шляхом зміни вмісту цих регістрів під час виконання програми різні реконфігуровані модулі можуть виконувати різні функції в різні моменти часу. Зокрема, при розробці системи керування двигуном для виміру струму два аналогових модулі можуть бути відконфігуровані як інструментальний підсилювач. Для виміру напруги один з цих модулів може використовуватися як підсилювач з перемінним коефіцієнтом підсилення. Аналогічно, цифрові модулі можуть змінювати функції в процесі роботи. Зокрема, асинхронний передавач може використовуватися як 8-бітний лічильник, коли передача даних не здійснюється.

Досить гнучко реалізована система з'єднань входів і виходів модулів із зовнішніми контактами процесора і між собою. Аналогічно як і в попередньому випадку, усі з'єднання визначаються вмістом визначених регістрів і можуть бути змінені в процесі роботи. Зокрема, дуже легко реалізуються функції аналогових і цифрових мультиплексорів і демультиплексорів, комутаторів аналогових і цифрових сигналів.

1.2. Характеристики мікроконтролерів сімейства CY8C29x66

Нижче наведено основні характеристики мікроконтролерів сімейства CY8C29x66:

- Процесор з гарвардською архітектурою
- Швидкодія процесора M8C до 24 Мгц
- Помножувач 8x8, 32-разр. накопичувач
- Мала споживана потужність при високій швидкодії
- Робоча напруга 3.0В.5.25В

- Робоча напруга від 1.0В при використанні вбудованого імпульсного перетворювача

- Промисловий температурний діапазон: -40°С.+85°С

Периферійні пристрої (блоки PSoC)

- 12 аналогових блоків PSoC (rail-to-rail), що містять:
  - АЦП з дозволом до 14 розр та ЦАП з дозволом до 9 розр.
  - Підсилювачі з програмованим підсиленням
  - Програмовані фільтри і компаратори
- 16 цифрових блоків PSoC, які містять:
  - 8.32-разр. таймери, лічильники і ШИМ
  - Модулі CRC і PRS
  - До 4 повнодуплексних UART
  - Декілька ведучих або підлеглих SPI
  - Схему підключення до всіх ліній входу-виходу
- Комплексні периферійні пристрої за рахунок поєднання блоків

Прецизійна програмована синхронізація

- Внутрішній генератор ±2,5% 24/48 Мгц
- 24/48 Мгц з опційним кварцовим резонатором 32,768 кгц
- Опційний зовнішній генератор частотою до 24 Мгц
- Внутрішній генератор для сторожового таймера і таймера режиму сну

Гнучка вбудована пам'ять

-32 кбайт флэш-пам'яті програм із зносостійкістю 50 тис. запису/стирання

- 2 кбайт статичного ОЗУ для зберігання даних
- Внутрішньосистемне послідовне програмування (ISSP)
- Часткове оновлення флэш-пам'яті
- Гнучкі режими захисту
- Емуляція EEPROM у флэш-пам'яті

Програмована конфігурація виходів

- Здатність навантаження на всіх лініях I/O 25 ма (вхідний струм)

- Всі лінії входів-виходів можуть переводитися в один з наступних станів: підтягаючий резистор до плюса або до мінуса, високоімпедансне, двотактний вихід або відкритий стік

- До 12 аналогових входів на всіх лініях I/O

- 4 аналогових виходу із здатністю навантаження 40 ма на будь-яких лініях I/O

- Переривання, що конфігурується, на всіх лініях I/O

Додаткові системні ресурси

- Ведучий, підлеглий і багатомастерний інтерфейс I2C з частотою синхронізації до 400 кгц

- Сторожовий таймер і таймер режиму сну

- Програмований детектор зниження напруги

- Вбудована схема супервізора

- Вбудоване прецизійне джерело опорної напруги (ION)

Повний набір засобів для проектування

- Вільне програмне забезпечення для проектування (PSoC™ Designer)

- Повнофункціональний внутрішньосхемний емулятор і програматор

- Повношвидкісна емуляція

- Складна структура точок переривання

- 128 кбайт пам'яті трасування

- Комплексні події

- Сі-компілятори, Асемблер

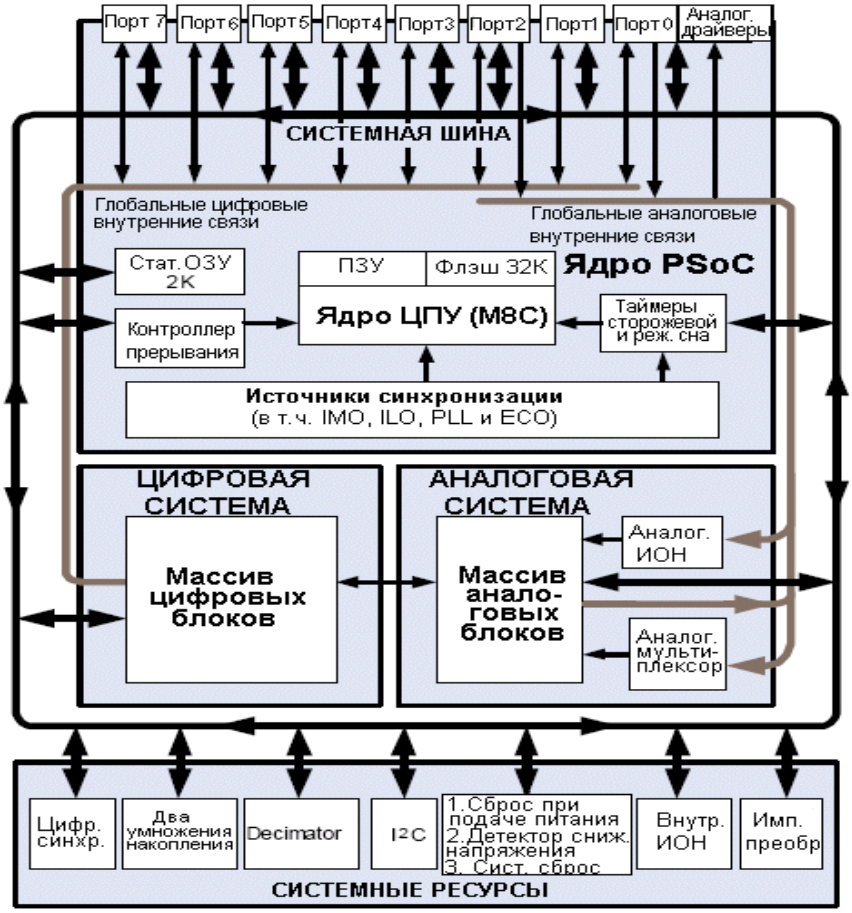


Рис. 1. Структурна схема мікроконтролера сімейства CY8C29x66.

1.3. Цифрова підсистема мікроконтролерів сімейства CY8C29x66.

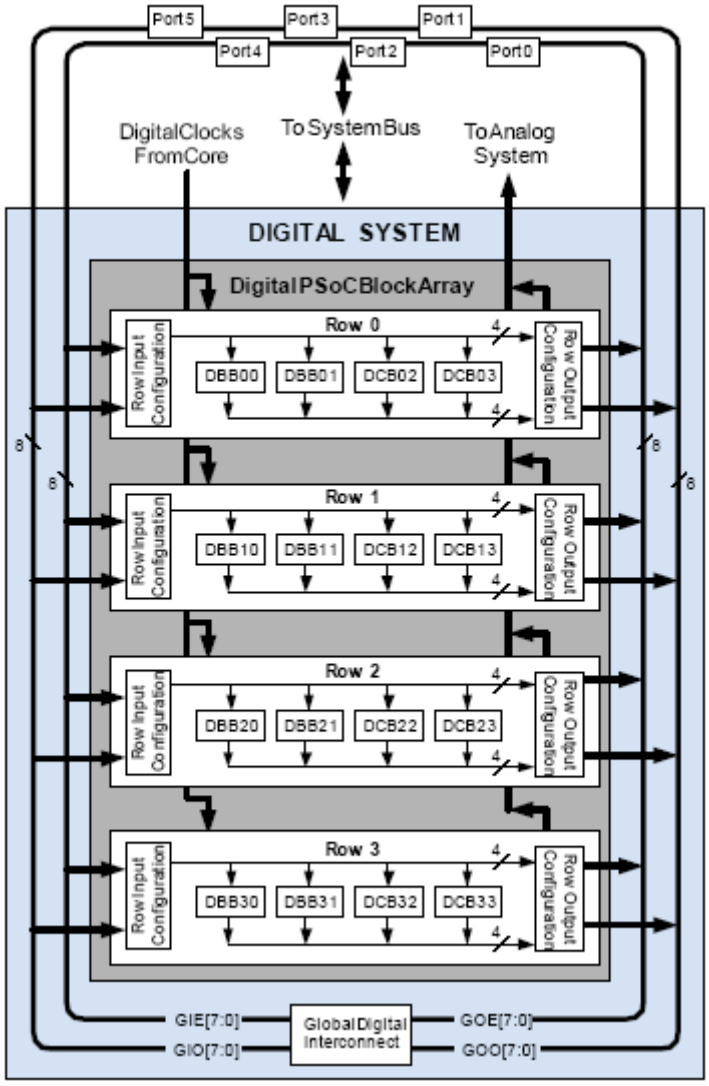


Рис. 2. Структурна схема цифрового блоку мікроконтролера CY8C29x66.

Цифрова система складається з 16-ти цифрових PSoC блоків. Кожен блок є 8-бітним ресурсом який може бути використаний один або скомбінований з іншими блоками з 8, 16, 24 та 32-бітними периферійними пристроями, які називаються користувацькими модулями.

Конфігурація цифрових периферійних пристроїв влючає наступний список поданий нижче:

-широтно-імпульсні модулятори (від 8 до 32 біт)

-широтно-імпульсні модулятори з мертвою зоною (від 8 до 32 біт)

-лічильники (від 8 до 32 біт)

-таймери (від 8 до 32 біт)

-універсальний асинхронний прийомопередатчик (8-бітний з вибірковою парністю)

-послідовний інтерфейс (до 2-х штук)

-I2C інтерфейс (1 доступний яу системний ресурс)

-Інфрачервоний порт (IrDA)

-генератор випадкових псевдо послідовностей

Цифрові блоки можуть бути приєднані до будь-яких портів вводу/виводу через серію глобальних шин які можуть напрявляти будь-який сигнал на будь-який пін.

1.4. Аналогова підсистема мікроконтролерів сімейства CY8C29x66

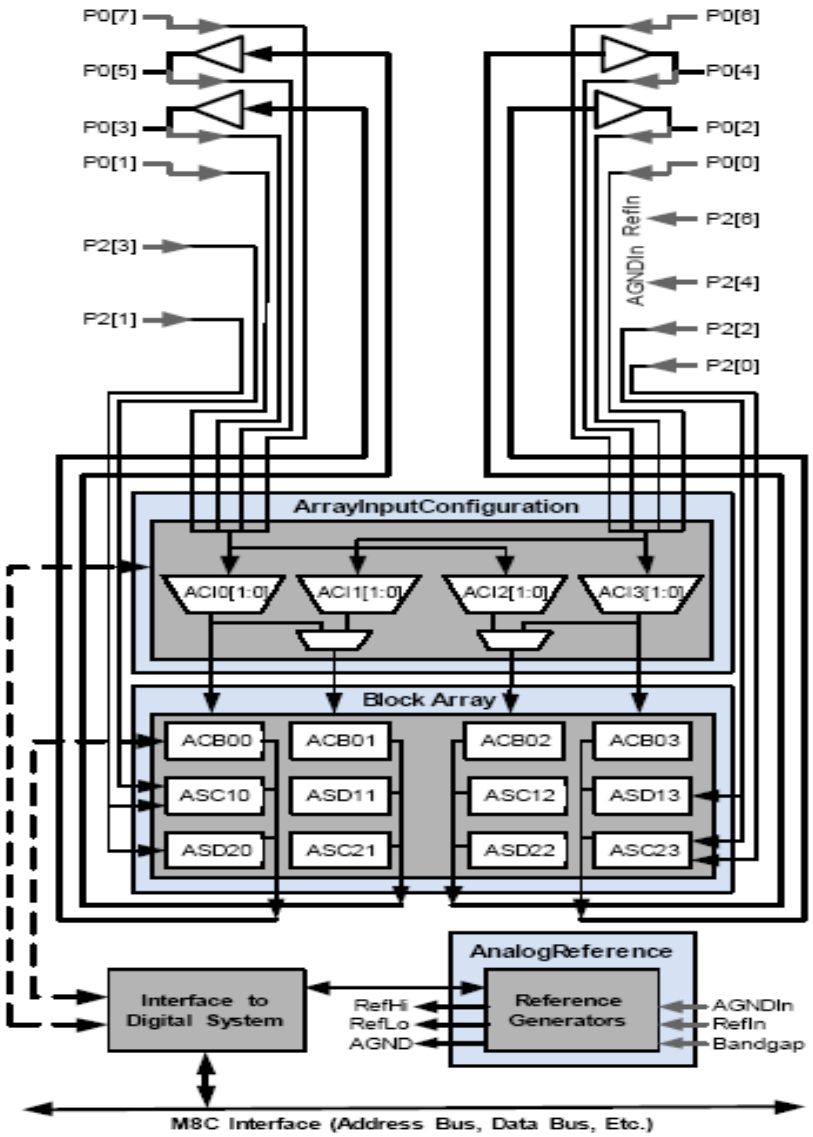


Рис. 3. Структурна схема аналогового блоку мікрропроцесора CY8C29x66.

Аналогова система складається з 12-ти блоків, кожен з яких складається з операційного підсилювача, який дозволяє створювати потік косплексних аналогових сигналів. Аналогові периферійні пристрої є дуже гнучкими і можуть бути зроблені для підтримки специфічних програмних потреб. Деякі з багатьох загальних функцій (найбільш підтримуваних, як модулів корисувачів) є подані нижче.

-Аналогово-цифровий перетворювач (до 4-х, з 6-ти до 14-ти бітовим розширенням)

- Фільтри (2,4,6 і 8 дорожні)

- Підсилювачі (до 4-х)

- Компаратори (до 4-х, з 16-ма вибірковими границями)

- Цифрово-аналоговий перетворювач (до 4-х, з 6 до 9 бітного розширення)

- Змішування цифро-аналогових перетворювачів (до 4-х, від 6 до 9 бітового розширення)

- 1.3В зв’язок (як системний ресурс)

- DTMF номеронабирач

- Модулятори

- Корелятори

- Піковий детектор

- Можливо багато інших топологій

На основі аналогових модулів можуть бути реалізовані фільтри різних типів, змішувачі сигналів, підсилювачі з перемінним коефіцієнтом підсилення, аналого-цифрові і цифро-аналогові перетворювачі різних типів, компаратори. Відзначимо, що в аналогових модулях на основі операційних підсилювачів з резистивним зворотним зв'язком присутній низькоспоживаючий компаратор (15 u), спеціально призначений для „пробудження” мікроконтролера з режиму сну. Мікроконтролер PSoC має 16 цифрових модулів двох типів, на основі яких можуть бути реалізовані лічильники і таймери різної розрядності (8-32), генератори широтно-імпульсних сигналів, асинхронні і синхронні послідовні приймач/передавач,

приймач/передавач даних по протоколі IrDA, генератори псевдовипадкових послідовностей, апаратні схеми формування циклічних контрольних сум (CRC).

Функції кожного аналогового і цифрового модуля визначаються вмістом відповідних регістрів керування. Шляхом зміни вмісту цих регістрів під час виконання програми різні реконфігуровані модулі можуть виконувати різні функції в різні моменти часу. Зокрема, при розробці системи керування двигуном для виміру струму два аналогових модулі можуть бути відконфігуровані як інструментальний підсилювач. Для виміру напруги один з цих модулів може використовуватися як підсилювач з перемінним коефіцієнтом підсилення. Аналогічно, цифрові модулі можуть змінювати функції в процесі роботи. Зокрема, асинхронний передавач може використовуватися як 8-бітний лічильник, коли передача даних не здійснюється.

Досить гнучко реалізована система з'єднань входів і виходів модулів із зовнішніми контактами процесора і між собою. Аналогічно як і в попередньому випадку, усі з'єднання визначаються вмістом визначених регістрів і можуть бути змінені в процесі роботи. Зокрема, дуже легко реалізуються функції аналогових і цифрових мультиплексорів і демультиплексорів, комутаторів аналогових і цифрових сигналів.

На рис.4 подано зовнішній вигляд мікроконтролера CY8C29466-24PXI фірми Cypress.

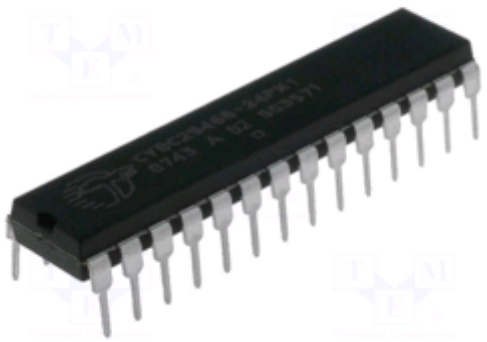


Рис. 4. Вигляд мікроконтролера CY8C29466-24PXI.

В таблиці 2. наведені технічні характеристики мікроконтролера сімейства CY8C29466-24PXI.

Таблиця 2.

Технічні характеристики мікроконтролера сімейства CY8C29466-24PXI.

Ядро	M8C
Ширина шини даних	8 bit
Максимальна тактова частота	24 MHz
Розмір програмної пам'яті	32 KB
Розмір ОЗП даних	2 KB
Робоча напруга живлення	3 V to 5.25 V
Діапазон робочих температур	- 40 C to + 85 C
Розрядність АЦП	14 bit
Доступні аналогові/цифрові канали	12
Розмір ПЗП даних	32 Kb
Кількість програмованих входів/виходів	24
Серія процесора	CY8C29x66

1.5.Опис середовища розробки PSoC Designer

Інтерфейс середовища PSoC Designer (рис.5) складається з декількох режимів, використання кожного з яких необхідне для проектування майбутнього спеціалізованого пристрою. Розробка починається з вибору необхідного мікроконтролера, розміщення необхідних модулів в блоках цього мікроконтролера і закінчується написанням програмного коду на мові С чи асемблер і відлагодженням пристрою.

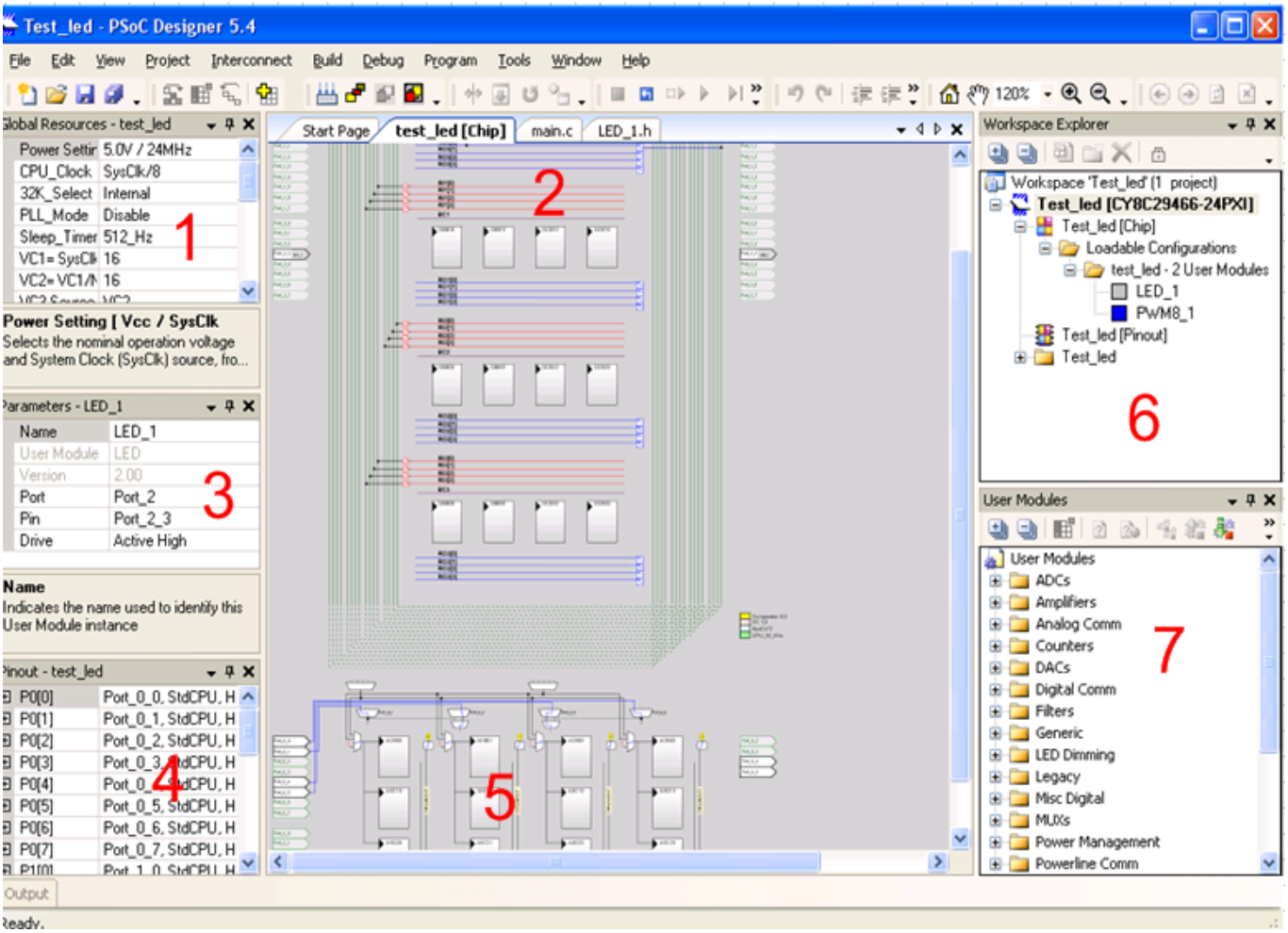


Рис. 5. Основний вигляд інтегрованого середовища PSoC Designer.

Нижче наведено короткий опис основних модулі створеного проекту інтегрованого середовища PSoC Designer:

1. Настройки глобальних параметрів вибраного мікроконтролера.
2. Набір цифрових блоків і портів вводу/виводу. В кожний цифровий блок можна вставляти модулі периферії і підключати до виводів портів.
3. Настройка параметрів вибраного модуля.
4. Настройка вхідних та вихідних портів.
5. Набір аналогових блоків і портів вводу/виводу.
6. Структура проекту і згенеровані бібліотечні файли.
7. Модулі периферії, які вставляються в цифрові і аналогові блоки.

Нижче наведено основні режими інтегрованого середовища розробки PSoC Designer:

1.6. Device Editor

Цей режим призначений для того, щоб користувач міг детальніше ознайомитися з обраною архітектурою, пристроями, які він може розташувати в архітектурі. Цей режим поділяється на два підрежими: User Module Selection View та Interconnector View.

- User Module Selection View(огляд модуля для підбору компонентів);

В цьому підрежимі PSoC Designer надає можливість підбирати користувачу необхідні компоненти, довідатися про внутрішню будову та відслідковувати, щоб їх кількість не перевищила дозволений рівень.

Компоненти поділені на 14 основних груп пристроїв:

- АЦП;
- підсилювачі;
- послідовні аналогові пристрої;
- лічильники;
- ЦАП;
- послідовні цифрові пристрої;
- фільтри;
- generic (аналоговий блок з конденсаторами, що перемикаються);
- misc digital (цифровий інвертор, EEPROM, LCD інструменти);
- мультиплексори;
- модулятори ширини імпульсу;
- генератори псевдовипадкових послідовностей;
- температурний давач;
- таймери.

- Interconnector View (огляд з`єднань);

Цей підрежим дозволяє детальніше налаштувати параметри архітектури, такі як частота роботи процесора, напругу споживання, напругу відключення і т.п. Можливе налаштування кожного параметра індивідуально. Нижче подано перелік параметрів і стани портів:

Вікно посередині дозволяє спростити проектування, оскільки наглядно показує розробнику структуру архітектури та розташування компонентів на чипі, а також можливі з`єднання і користувач вже планує, які зв`язки та яку трасу задіяти, показані порти і його можливі з`єднання. Це набагато спрощує процес проектування для людини розробника.

Вікно зліва дозволяє налаштувати параметри виводів(контактів) мікросхеми, встановити характер сигналу, який на нього подається.

1.7. Application Editor

Цей режим призначений для формування розробником файлів опису роботи, сформованої архітектури. Оскільки при створенні проекту провідною була обрана мова програмування С, то програму можна описати нею у головному файлі main.c.

Нижче розташоване вікно компілятора, де можна спостерігати поетапно весь цей процес.

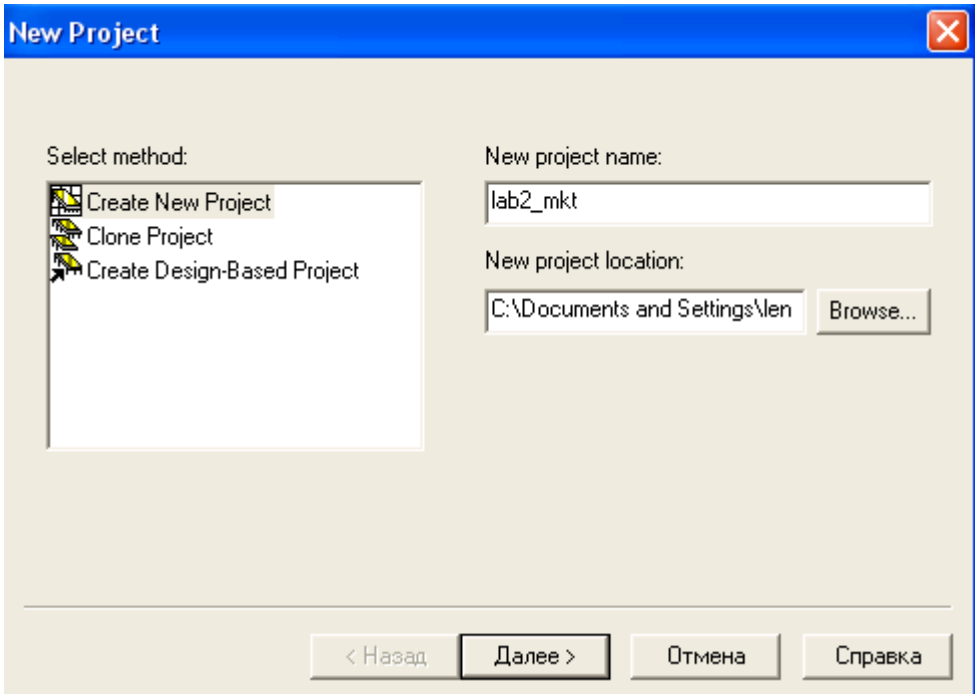
1.8. Debugger

Режим відлагодження додає вікно з вмістимим комірок пам`яті ОЗП, двох банків пам`яті, конфігурації, а також Flash-пам`яті. Для більшої наглядності, додано вікно, де можна відстежувати зміну зазначених глобальних значень.

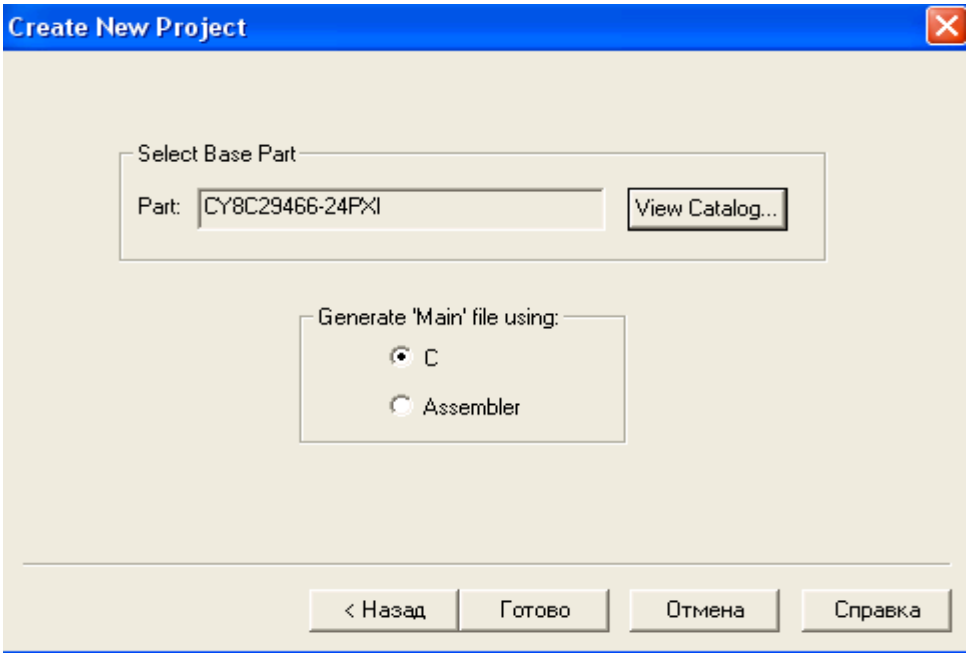
Основне меню прикладної програми містить стандартний набір функцій роботи з файлами, проектами, вікнами, з довідковою інформацією і т.п.

Хід виконання роботи

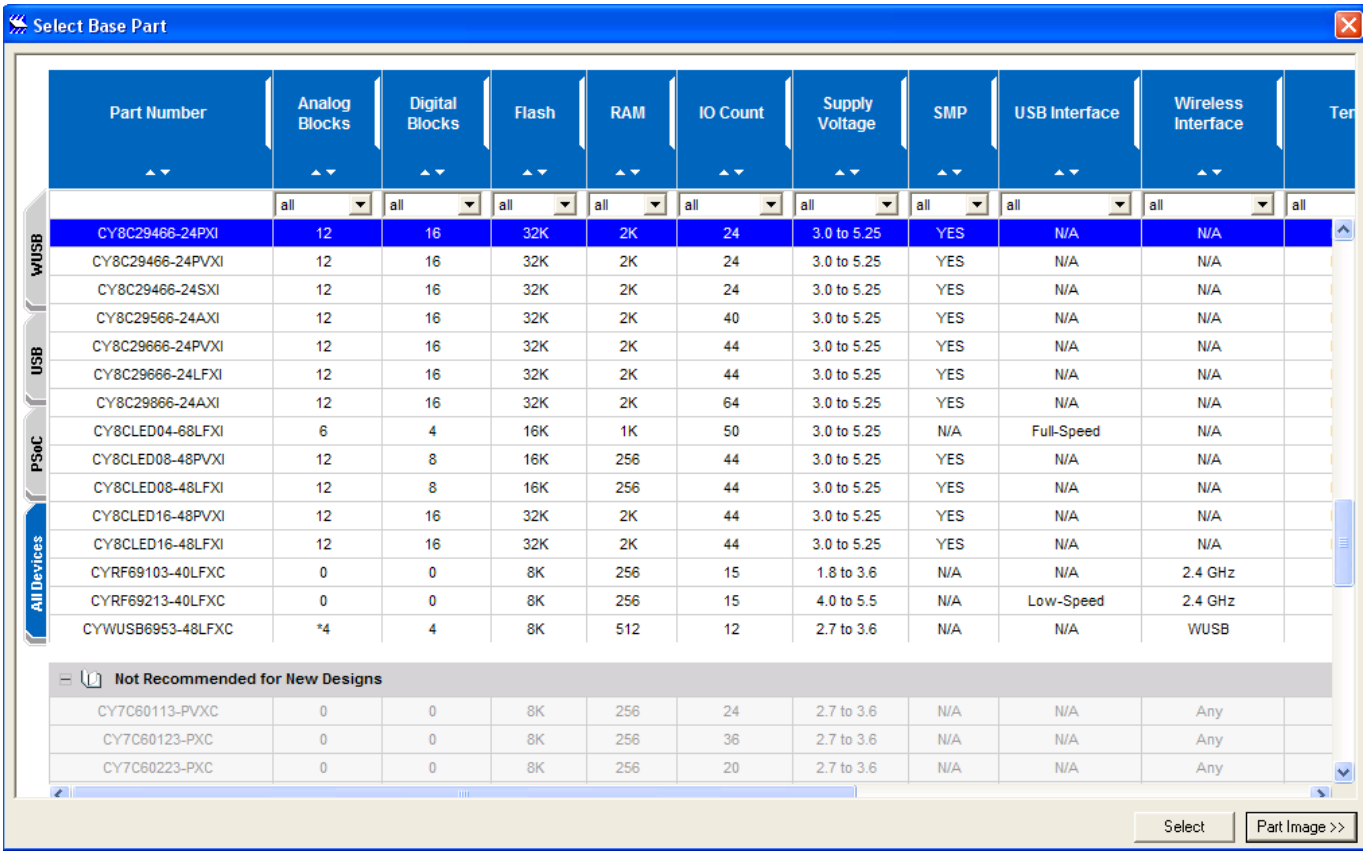
1. Завантажуємо інтегроване середовище PSoC Disigner заходим в File -> New Project -> Create New. Задаємо ім’я проету та вибираємо шлях на диску для його розміщення.



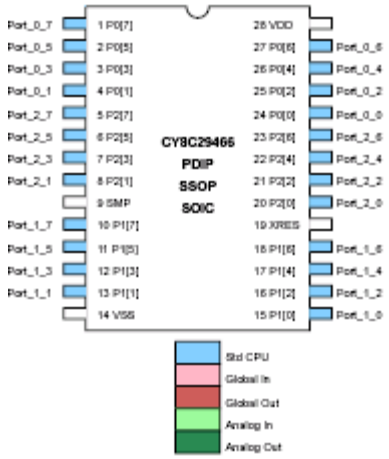
2. Далі в заголовку Generate ‘Main’ file using вибараємо мову програмування С на якій будемо описувати головну функцію системи.



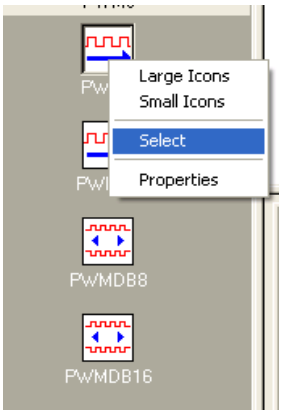
3. В заголовку Select Base Part натискаємо опцію View Catalog та вибираємо з каталогу тип мікроконтролера - CY8C29466 - 24PXI.



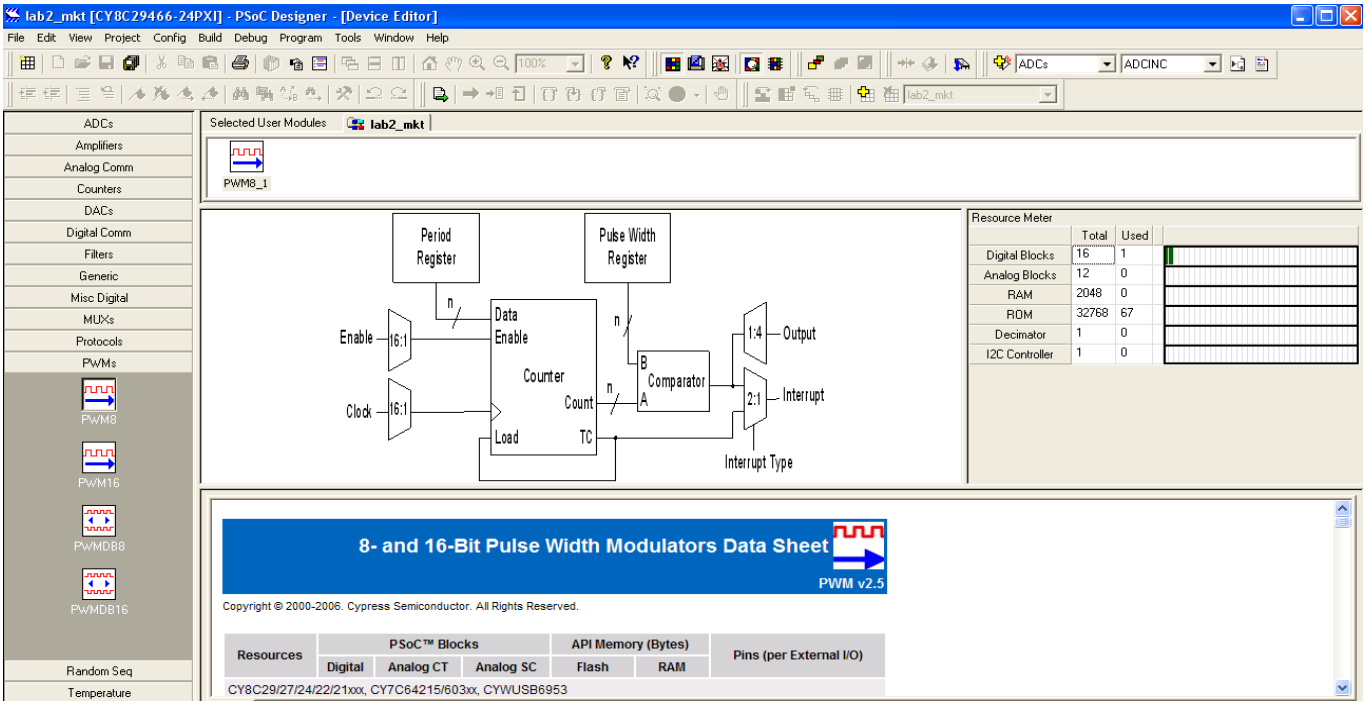
4. Натиснувши кнопочку Part Image можна побачити розпіновку вибраного мікроконтролера.



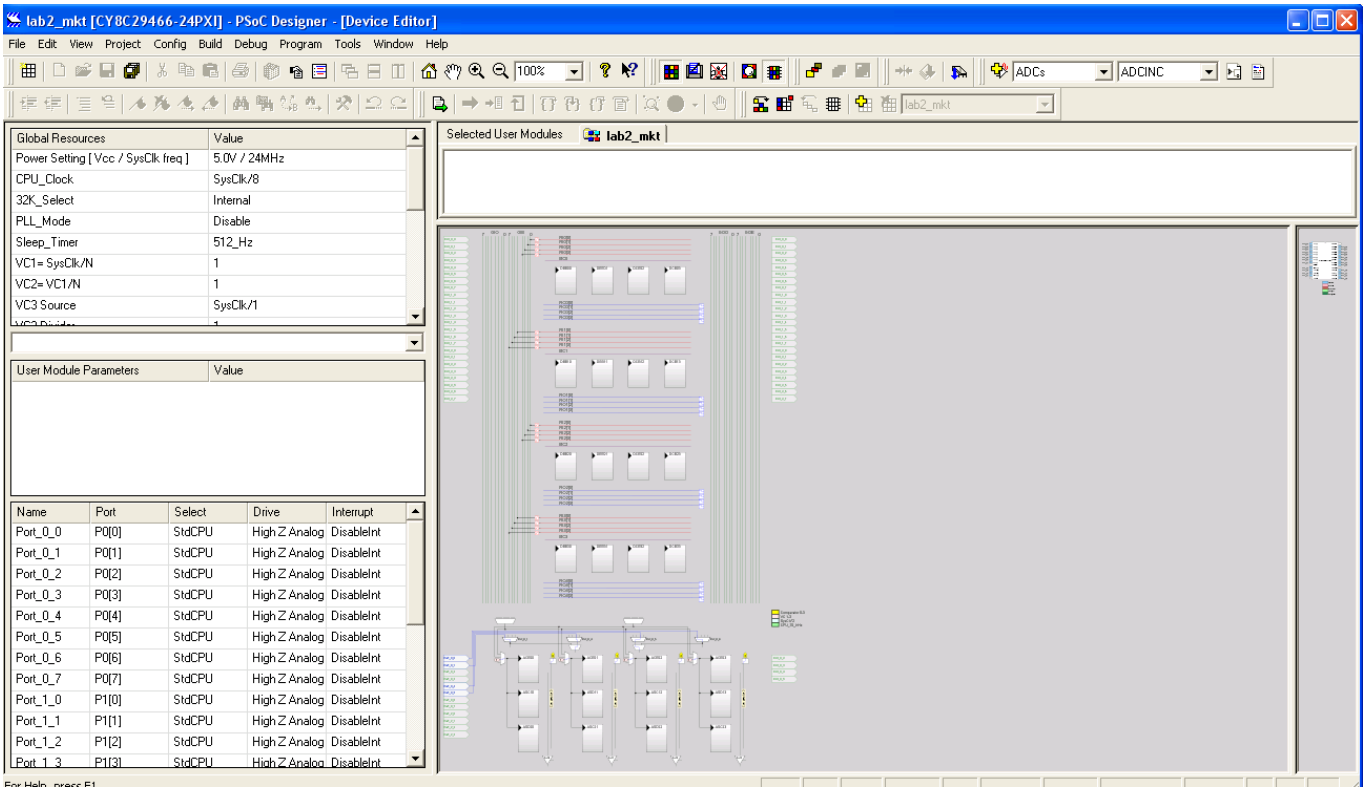
5. Далі натискаємо кнопку Готово. З’являється вікно в режимі Device Editor. В даному вікні зліва (Select User Modules) вибираємо з каталогу модулів PWMs - модуль PWM8 (Широтно-імпульсний модулятор).



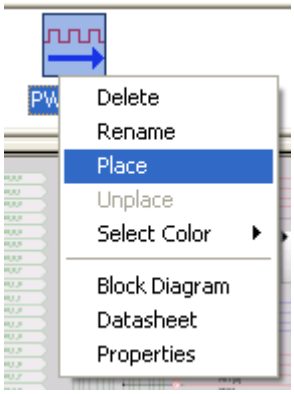
6. Вибравши даний модуль бачимо його функціональну схему і опис його основних характеристик (Data Sheet).



7. Далі на панелі інструментів середовища PSoC Designer натискаємо кнопку -  Interconnect View.

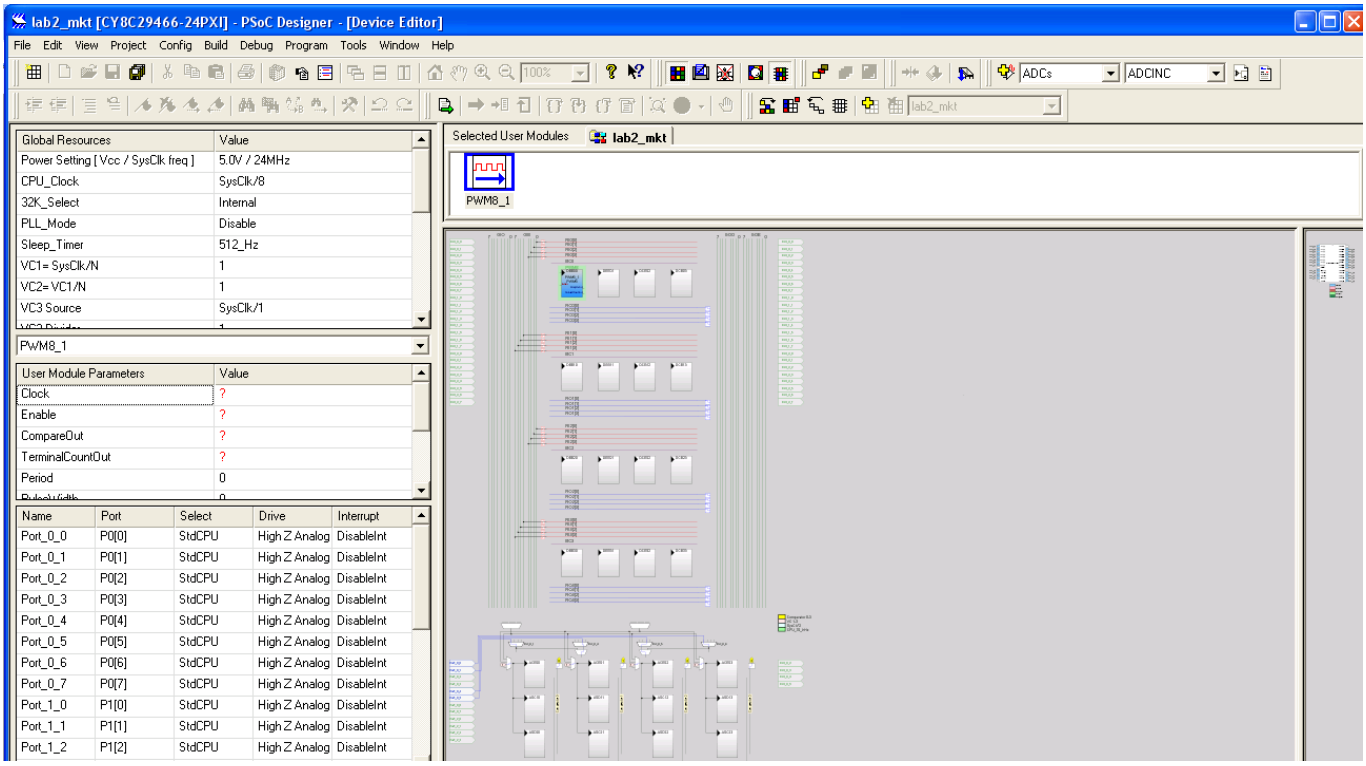


8. Далі встановивши курсор на модуль PWM8 та натиснувши праву кнопку миші вибираємо опцію Place.

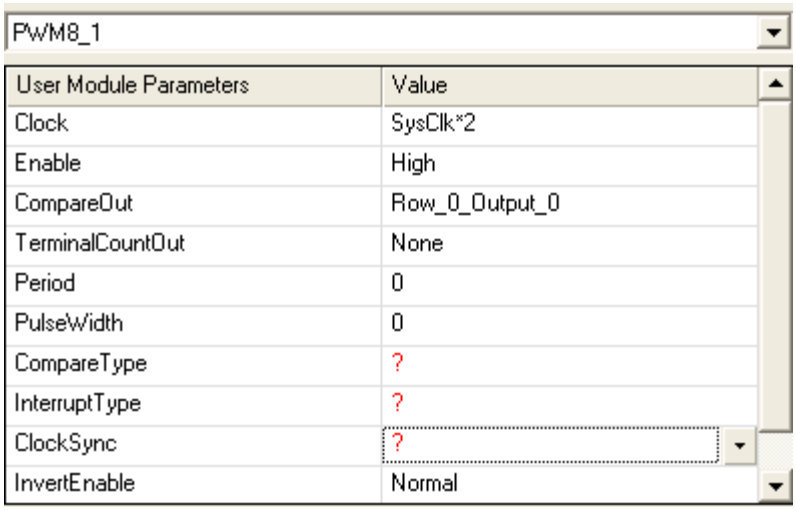


9. На наступному прінтскріні бачимо, що модуль PWM8 розмістився в одному цифровому блоці внутрішньої структури мікроконтролера CY8C29466 -

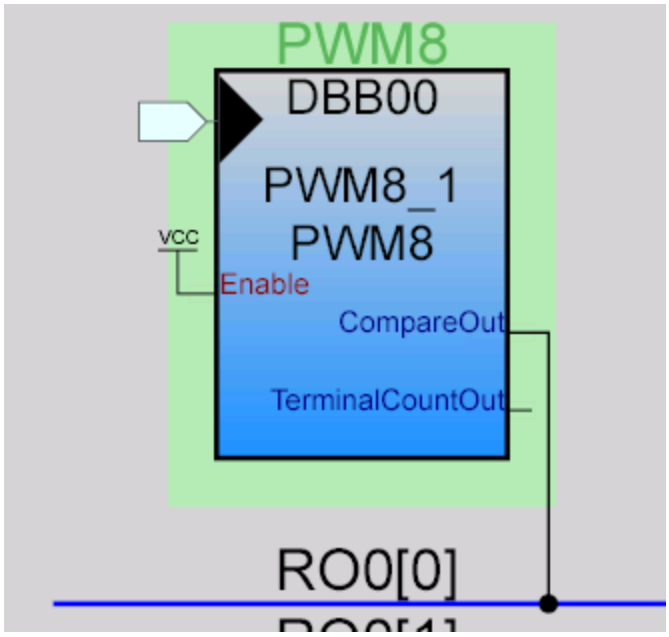
4PXI. Зліва появилося вікно User Module Parameters.



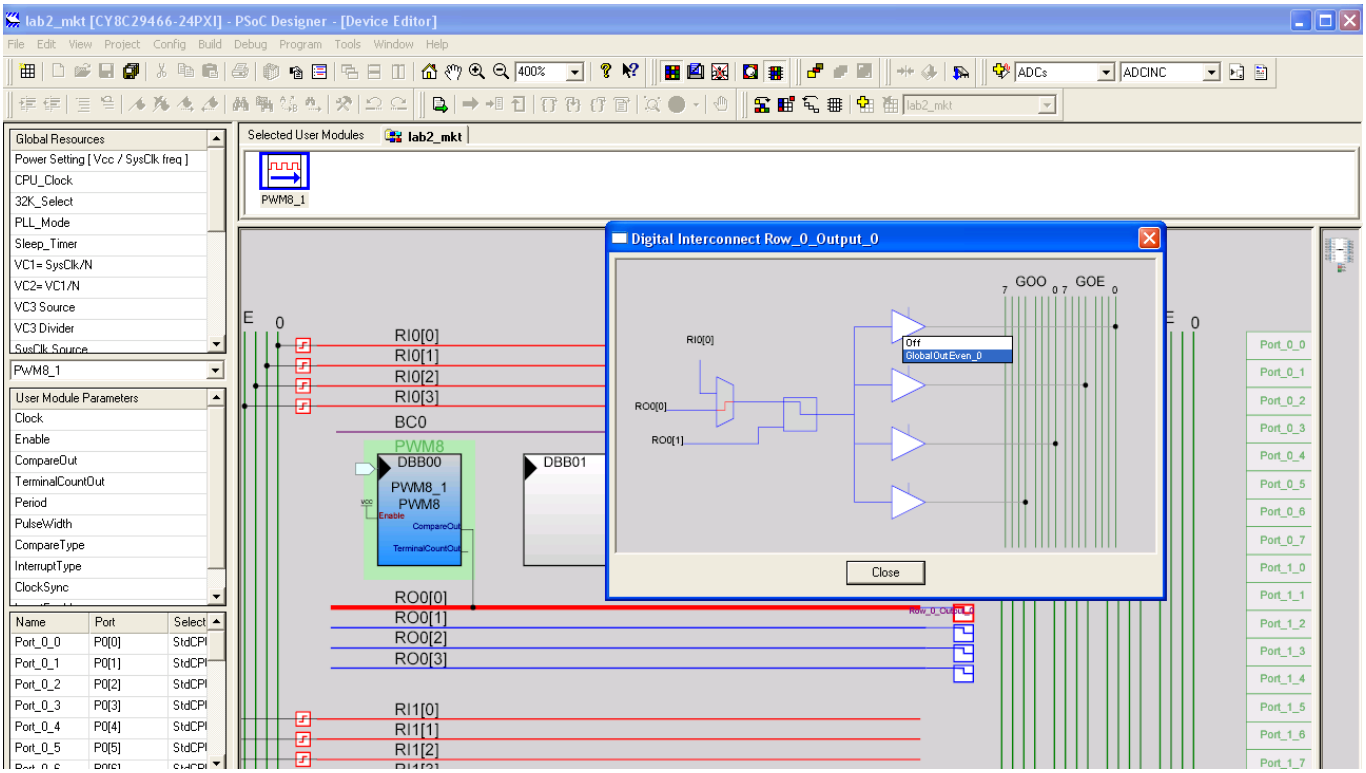
10. Задамо наступні параметри модуля PWM8:



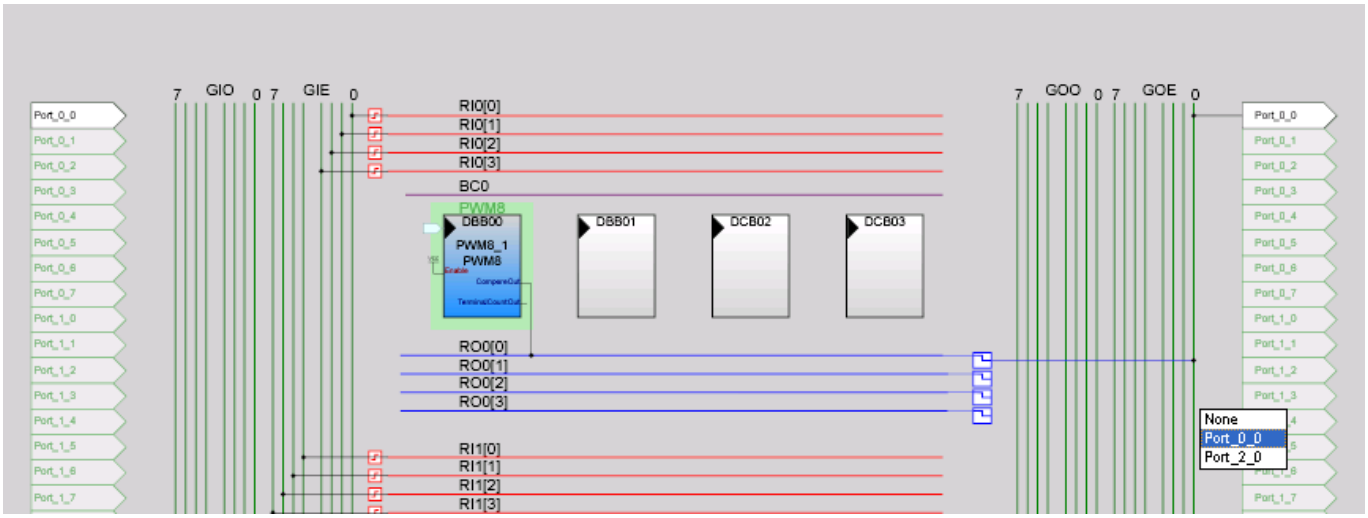
11. Після встановлення даних параметрів для модуля PWM8, можна побачити результат їх дії. Вихід модуля PWM8 під'єднано до локальної шини Row_0_Output_0.



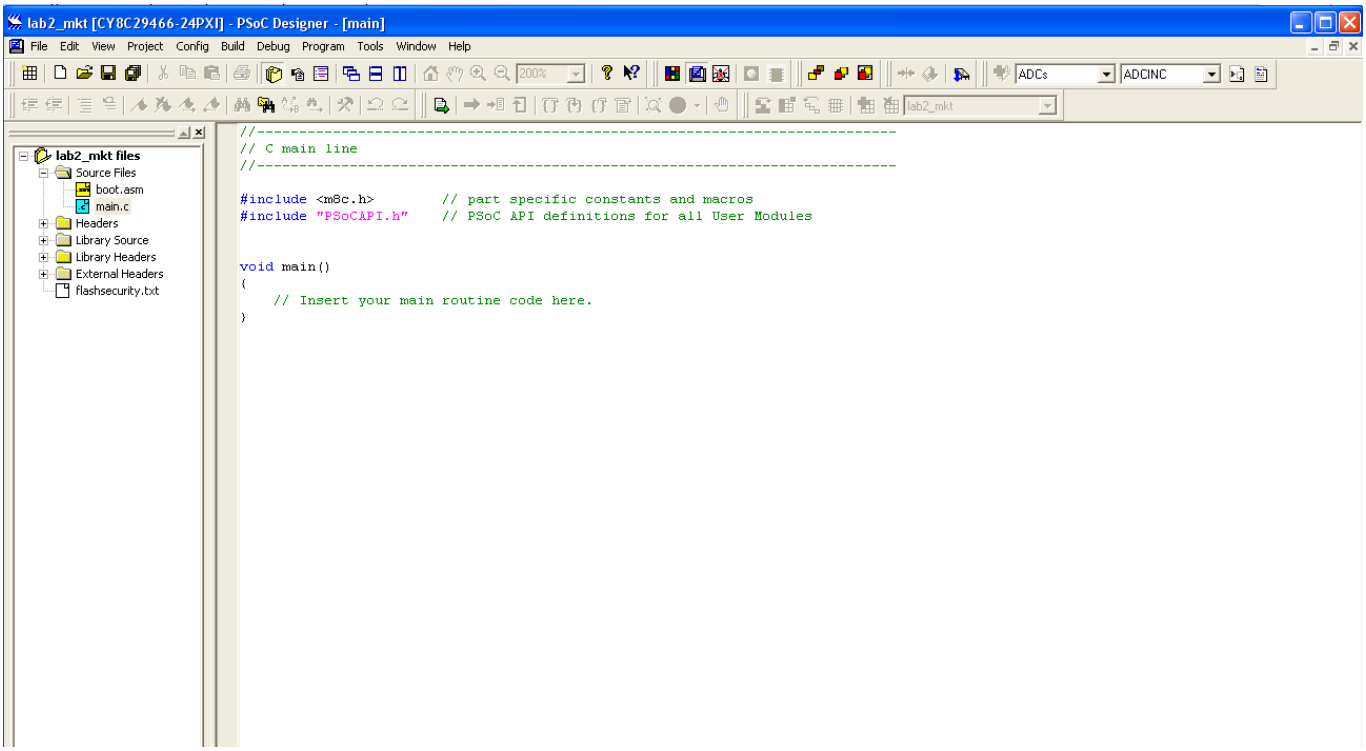
12. Далі потрібно вихід локальної шини Row_0_Output_0 під'єднати до виходу глобальної шини GlobalOutEvent_0.



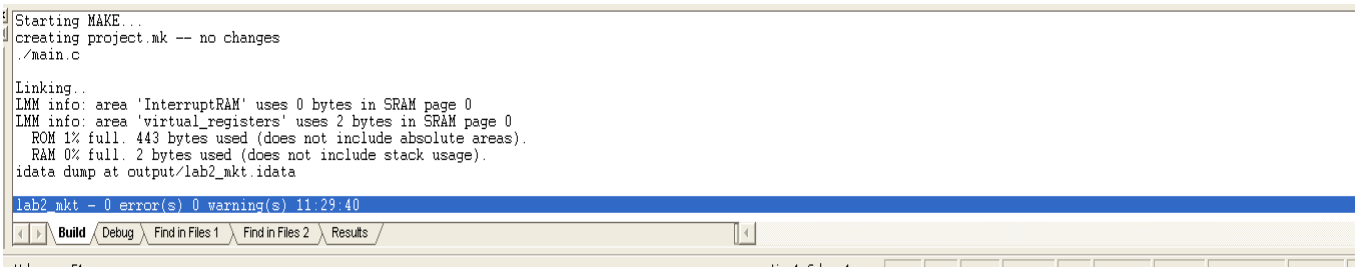
13. Наступним кроком потрібно вихід глобальної шини GlobalOutEvent_0 підєднати до вихідного порту мікроконтролера – вибравши Port_0_0.



14. Далі в меню середовища PSoC Designer вибираємо View -> Application Editor, та відкриваємо у вкладці Source Files файл main.c.



15. В меню середовища PSoC Designer вибираємо Build -> Build F7. Нижче наведено результат лінкування проекту.



16. В файлі main.c набираємо наступний код прошивки мікроконтролера використовуючи API-функції модуля PWM8 взяті з його документації (Data Sheet) або у вкладці вікна проекту External Headers вибравши файл pwm8_1.h.

```
1  #include          // part specific constants and macros
2  #include "PSoCAPI.h"    // PSoC API definitions for all User Modules
3
4
5  void main(void)
6  {
7      M8C_EnableGInt ; // Uncomment this line to enable Global Interrupts
8      // Insert your main routine code here.
9      PWM8_1_Start();
10     PWM8_1_EnableInt();
11     while(1);
12 }
```

17. Якщо набраний код пройшов етап лінкування без помилок, то можна переходити до програмування мікроконтролера вибравши в меню опцію Program -> Program Part.

Завдання до лабораторної роботи №5

Реалізувати простий проект з вибором одного цифрового або аналогового модуля з бібліотеки елементів згідно варіанту та описати його програмну частину.

Номер варіанту	Тип пристрою	Розрядність пристрою
1.	Аналог-цифровий перетворювач	10
2.	Цифро-аналоговий перетворювач	10
3.	Широтно-імпульсний модулятор	10
4.	Аналоговий мультиплексор 8->1	8
5.	Цифровий мультиплексор 8->1	

6.	Компаратор	16
7.	Суматор	16
8.	Арифметико-логічний пристрій	16
9.	Регістр зсуву	16
10.	Асинхронний лічильник	8

Зміст звіту

1. Титульна сторінка.
2. Назва та мета лабораторної роботи.
3. Навести вмістимек основного файлу main.c використовуючи його API-функції.
4. Висновки.

Контрольні запитання

1. Основні особливості PSoC фірми Cypress?
2. Основні характеристики та особливості процесорів архітектури PSoC?
3. До якої архітектури за способом побудови належать мікроконтролери сімейства CY8C29x66?
4. До якої архітектури за системою команд належать мікроконтролери сімейства CY8C29x66?
5. Який температурний діапазон роботи мікроконтролерів сімейства CY8C29x66?
6. Назвати ядро з якого складаються мікроконтролери сімейства CY8C29x66 та його тактову частоту?
7. Що входить в структуру мікроконтролерів сімейства CY8C29x66?
8. Основне призначення цифрових блоків мікроконтролерів сімейства CY8C29x66 та їх основні характеристики.
9. Основне призначення аналогових блоків мікроконтролерів сімейства CY8C29x66 та їх основні характеристики.

10. Основне призначення інтегрованого середовища розробки PSoC Designer та його основні модулі та режими?
11. Які мови програмування використовуються в інтегрованому середовищі розробки PSoC Designer?
12. Які основні технічні характеристики мікроконтролера сімейства CY8C29466-24PXI фірми Cypress?
13. Назвіть з яких основних вузлів складається мікроконтролер CY8C29466?
14. До якої архітектури за способом побудови належать мікроконтролер CY8C29466?
15. Назвіть основне призначення та характеристики широтно-імпульсного модулятора (ШІМ).
16. Наведіть основні API-функції на мові С модуля PWM (ШІМ).

ЛАБОРАТОРНА РОБОТА №6

Тема роботи: технології проектування комп'ютерних пристроїв на базі програмовних логічних інтегральних схем *Vivado Design Suite* фірми *Xilinx*.

Мета роботи: засвоїти навики роботи в середовищі *Vivado Design Suite* та навчитися організовувати обмін даними з використанням інтерфейсу UART.

Описання проекту

Проект складається з UART приймача, який приймає вхідні дані з клавіатури і висвічує бінарний еквівалент натиснутих символів на 8 світлодіодах. Коли кнопка натискається, то переставляються старша і молодша частини байта символу. Блок діаграма UART приймача показана на рис.1.

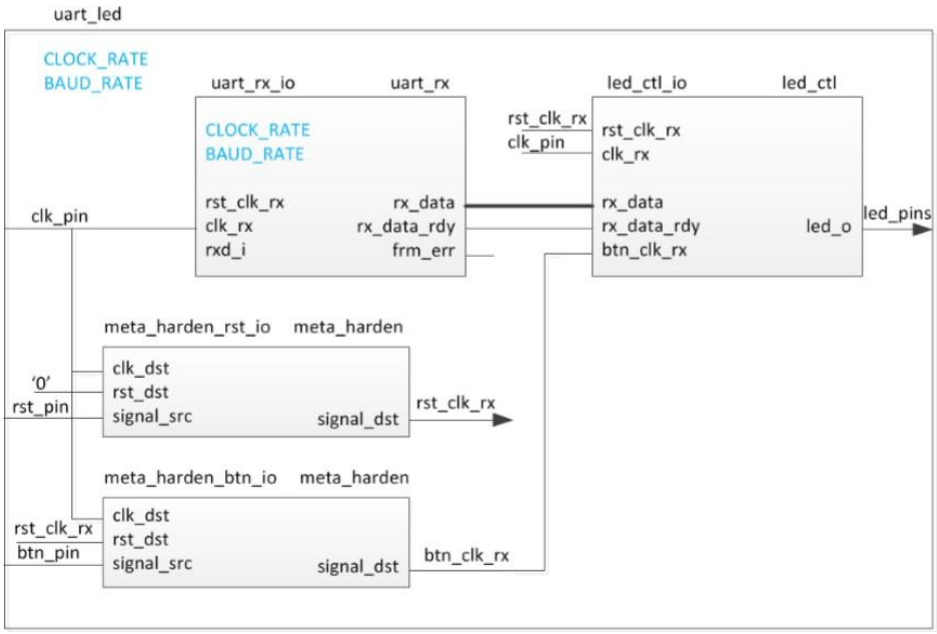
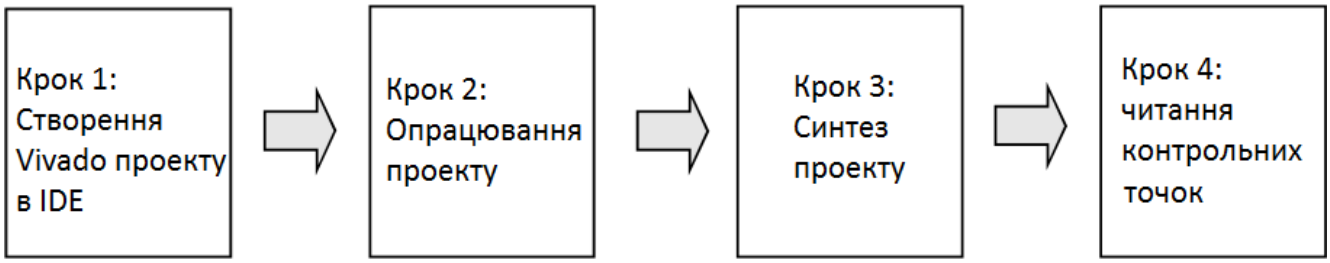


Рис. 1. Завершений проект.

Загальна послідовність кроків



Крок 1. Створення Vivado проекту з використання середовища IDE

1.1. Запустити Vivado і створити проект для XC7A100TCSG324-1 (Nexys4 DDR) або XC7A35TCPG236-1 (Basys3), або XC7A200TSBG484-1 (Nexys Video) пристрою, і вибрати мову Verilog HDL. Використати існуючі Verilog початкові файли uart_led_pins_<board>.xdc and uart_led_timing.xdc, що знаходяться у каталозі <2015_2_artix7_sources>\lab2.

Посилання на <**2015_2_artix7_labs**> є підкаталогом в **c:\xup\fpga_flow\2015_2_artix7_labs** каталозі і <**2015_2_artix7_sources**> є підкаталогом в **c:\xup\fpga_flow\2015_2_artix7_sources** каталозі. Посилання на <**board**> означає **Nexys4 DDR**, **Basys3** або Nexys Video.

Відкрити Vivado вибором **Start > All Programs > Xilinx Design Tools > Vivado 2015.2**

> Vivado 2015.2

1.1.1. Клацнути **Create New Project** для запуску помічника. Появиться діалогове вікно *A Create a New Vivado Project*. Клацнути на кнопку **Next**.

1.1.2. Клацнути кнопку Browse у полі *Project location field форми* **New Project** form, вибрати <**2015_2_artix7_labs**>, і клацнути **Select.**

1.1.3. Ввести **lab6** у поле *Project name*. Переконатися, що у квадраті *Create Project Subdirectory* є позначка. Клацнути кнопку **Next**.

1.1.4. У формі *Project Type* вибрати опцію **RTL Project** і клацнути кнопку **Next**.

1.1.5. У формі *Add Sources,* використовуючи випадаючі кнопки, вибрати **Verilog** як *Target Language* і *Simulator Language*.

1.1.6. Клацнути на кнопку **Green Plus** , потім на кнопку **Add Files…,** знайти у полі *Look in* каталог <**2015_2_artix7_sources>\lab6**, вибрати всі Verilog файли (*led_ctl.v, meta_harden.v, uart_baud_gen.v, uart_led.v, uart_rx.v, and uart_rx_ctl.v*), клацнути кнопку **OK**. У формі *Add Sources* появляться вибрані файли. Далі клацнути кнопку **Next** для отримання форми *Add Existing IP*.

1.1.7. Так як в проект не добавляються ніякий **IP** (intelectual property), *клацнути кнопку*

Next для отримання форми *Add Constraints.*

Add Constraints.

1.1.8. Клацнути на кнопку **Green Plus**, потім на кнопку **Add Files…** і продивтися каталог **c:\xup\fpga_flow\2015_2_artix7_sources\lab6** (якщо потрібно), вибрати *uart_led_timing.xdc* і клацнути **Open**.

1.1.9. Клацнути кнопку **Next**.

1.1.10. У формі *Default Part*, використовуючи опцію **Parts** і різні випадаючі поля секції **Filter** (family Artix-7), вибрати **XC7A100TCSG324-1** (для Nexys4 DDR), XC7A35TCPG236-1 (для Basys3) або XC7A200tsbg484-1 (для Nexys Video).

1.1.11. Клацнути кнопку **Next**.

1.1.12. У формі *New Project Summary* клацнути кнопку **Finish** для створення Vivado проекту.

1.2. Аналіз ієрархії початкових файлів проекту.

1.2.1. У закладці *Sources*, розкрити вхід **uart_led** і переглянути ієрархію модулів нижчого рівня.

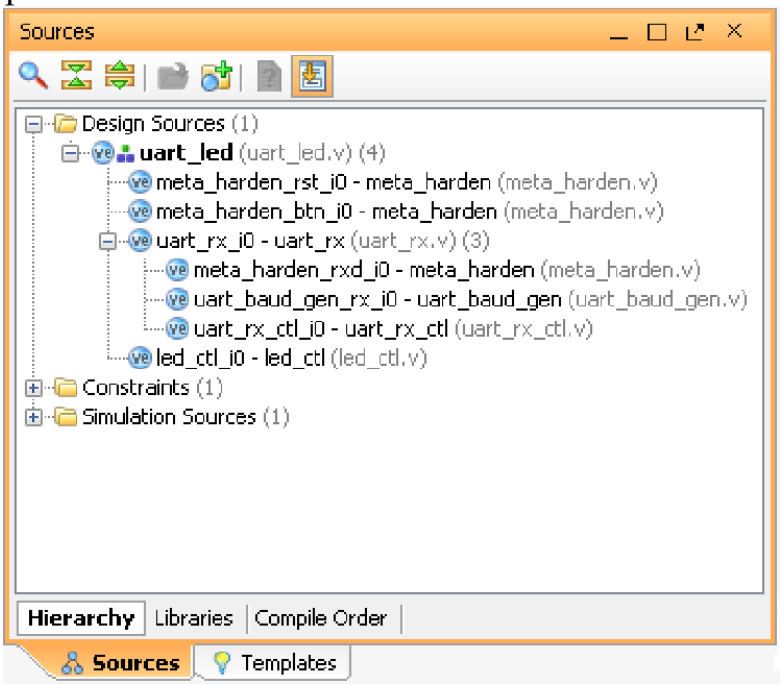


Рис. 2. Відкриття початкових файлів.

1.2.2. Подвійним клацанням на вході **uart_led** переглянути його вміст.

Зауважимо, що у Verilog коді (uart_baund_gen.v), параметри BAUD_RATE і CLOCK_RATE визначені як 115200 і 100 MHz відповідно, які показані на діаграмі проекту (рис. 1). Також зауважимо, що модулі нижчого рівня також добавлені. Модулі meta_harden використовуються для синхронізації асинхронного перевантаження (reset) і кнопкового (push-button) введення.

1.2.3. Розкрити **uart_rx_i0** екземпляр і переглянути його ієрархію.

Цей модуль використовує baud rate генератор і машину скінченного стану. Вивід rxd_pin опитується із частотою baud rate x16.

1.3. Відкрити файл uart_led_timing.xdc і проаналізувати його вміст.

1.3.1. У закладці *Sources*, розкрити каталог *Constraints* і подвійним клацанням на вході **uart_led_timing.xdc** відкрити файл у текстовому режимі.

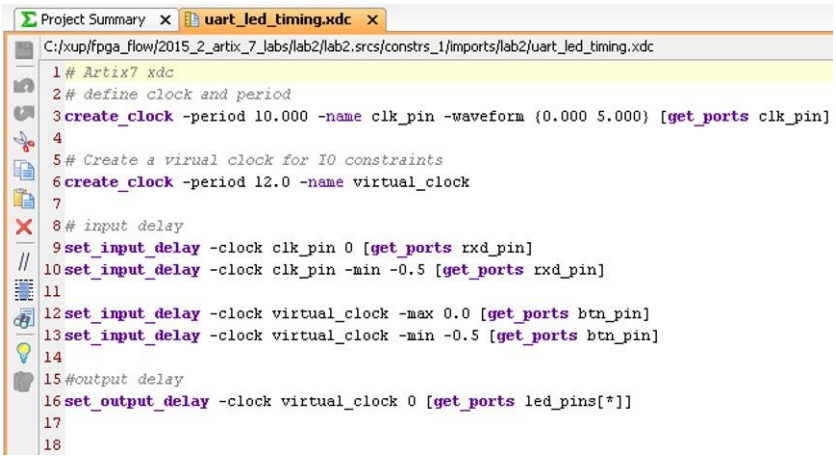


Рис. 3. Часові обмеження.

Крок 2. Опрацювання проекту

2.1. Опрацювати і виконати RTL аналіз над початковим файлом.

2.1.1. У вікні *Flow Navigator*, форми *Project manager*, рядок *RTL Analysis*, розкрити вибір *Open Elaborated Design Analysis tasks* і клацнути на **Schematic.**

Проект (модель) буде опрацьований і висвітиться логічний вид проекту.

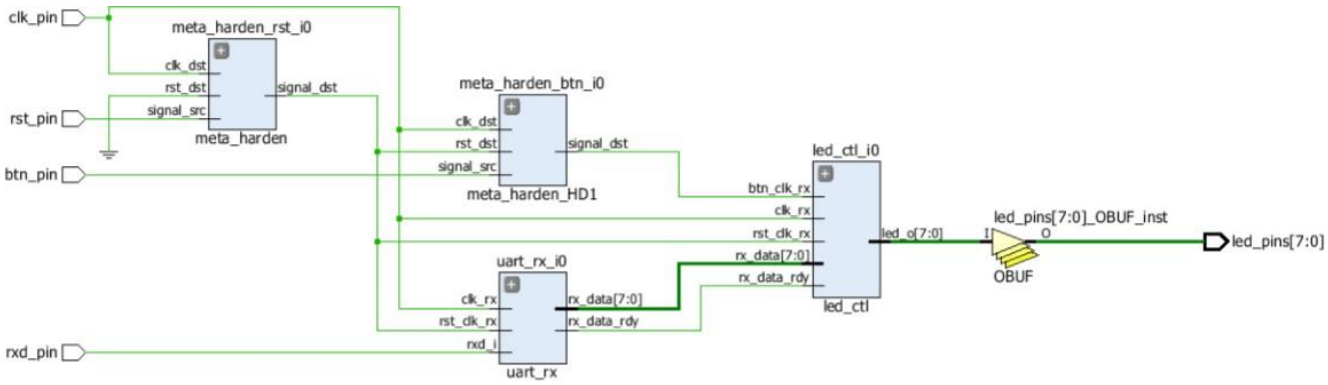


Рис. 4. Логічний вид проекту.

Можна побачити чотири компоненти на верхньому рівні, 2 екземпляри meta_harden, один екземпляр uart_rx і один екземпляр led_ctl.

2.1.2. Для того щоб побачити з чого генерується uart_rx_i0, потрібно подвійно клацнути на екземплярі uart_rx_i0, вибрати у вікні *Sources* модуль uart_rx_i0.v і рядок 84.

2.1.3. Подвійне клацання на екземплярі uart_rx_i0 схематичної діаграми дозволяє побачити компоненти нижчого рівня.

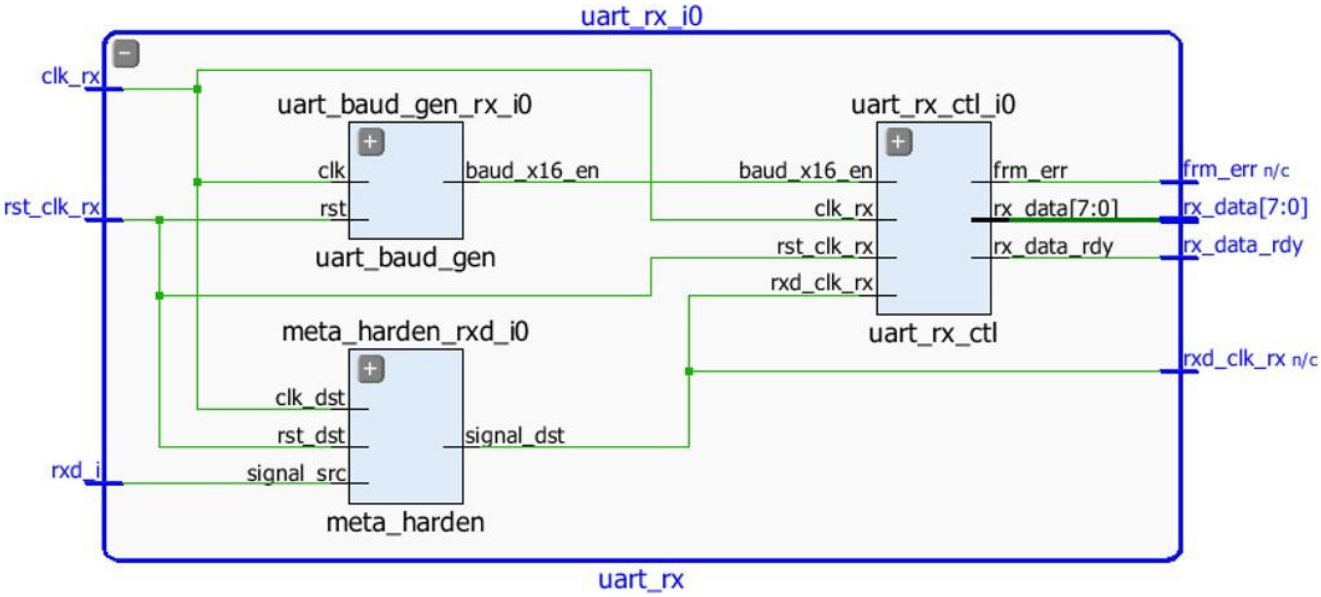


Рис. 5. Компоненти нижчого рівня модуля uart_rx_i0

2.1.4. У вікні *Flow Navigator*, форми *Project manager*, рядок *RTL Analysis*, розкрити вибір
Open Elaborated Design Analysis tasks і клацнути на **Report Noise**.

2.1.5. Клацнути **OK** для генерації звіту з іменем **ssn_1**.

2.1.6. Коли появиться звіт ssn_1 то у ньому можна побачити нерозміщені порти Ports, підсумки Summary, і деталі портів I/O Bank підсвічені червоним кольором, тому що не було зроблено назначення виводів. Потрібно зауважити, що тільки вихідні виводи позначаються як такі, що для них виконаний шумовий аналіз.

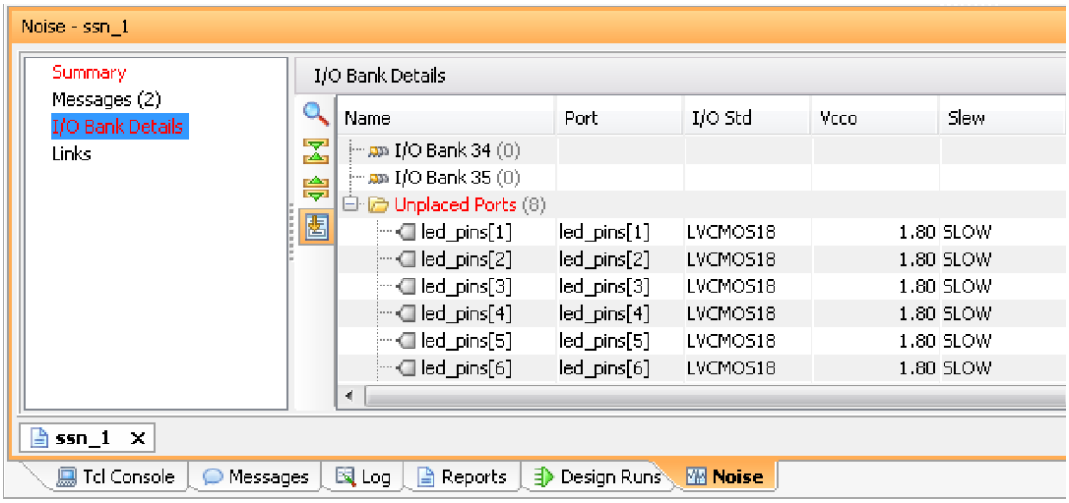


Рис. 6. Загальний звіт

2.1.7. У вікні *Flow Navigator*, форми *Project manager* зробити вибір *Add Sources*, у формі, що появиться Add Souces вибрати опцію *Add or Create Constraints* і клацнути **Next.**

2.1.8. Клацнути на кнопці **Green Plus** у центрі поля, потім на кнопці **Add Files…,** перейти у каталог <**2015_2_artix7_sources**>**\lab6** і вибрати фал **uart_led_pins_<board>.xdc** (залежно від цільової плати), клацнути **OK**, і тоді клацнути **Finish** для добавлення обмежень на розміщення виводів.

Необхідно початкові файли були модифіковані і інструменти це виявили, показуючи у прямокутнику повідомлення про необхідність перевантаження проекту.

2.1.9. У прямокутнику повідомлення клацнути на посилання **Reload**. Обмеження будуть оброблені.

2.1.10. У вікні *Flow Navigator*, форми *Project manager*, рядок *RTL Analysis*, розкрити вибір *Elaborated Design* і клацнути на посиланні **Report Noise,** а потім клацнути **OK** для генерації звіту з іменем **ssn_1**. Цього разу не має появитися ніяких помилок (не має червоного).

Крок 3. Синтез проекту

3.1. Синтезувати проект за допомогою інструментів синтезу Vicado і проаналізувати підсумковий вивід проекту.

3.1.1. У вікні *Flow Navigator*, вибрати *Synthesis* і клацнути на **Run Synthesis**.

Процес синтезу буде запущений для файлу uart_led.v і всіх файлів його ієрархії. Коли процес завершиться появиться діалогове вікно *Synthesis Completed* з трьома опціями.

3.1.2. Вибрати опцію *Synthesized Design* і клацнути **OK** якщо потрібно переглянути вихід синтезу.

Клацнути **Yes**, щоб закрити опрацьований проект, якщо висвітиться діалогове вікно.

3.1.3. Якщо не появиться закладка **Project Summary** тоді вибрати **Windows/Project Summary** або клацнути на іконці **Project Summary (Σ)**.

3.1.4. У вікні **Project Summary**, формі *Utilization* клацнути на вибір **Table** і заповнити наступну інформацію.

Запитання 1.

Переглянути таблицю, знайти і записати число, яке використовується наступними елементами:

FF: _____

LUT: _____

I/O: _____

BUFG: _____

3.1.5. У вікні *Flow Navigator* вибрати *Synthesis* рядок *Synthesized Design* і клацнути на

Schematic для перегляду синтезованого проекту у схематичному виді.

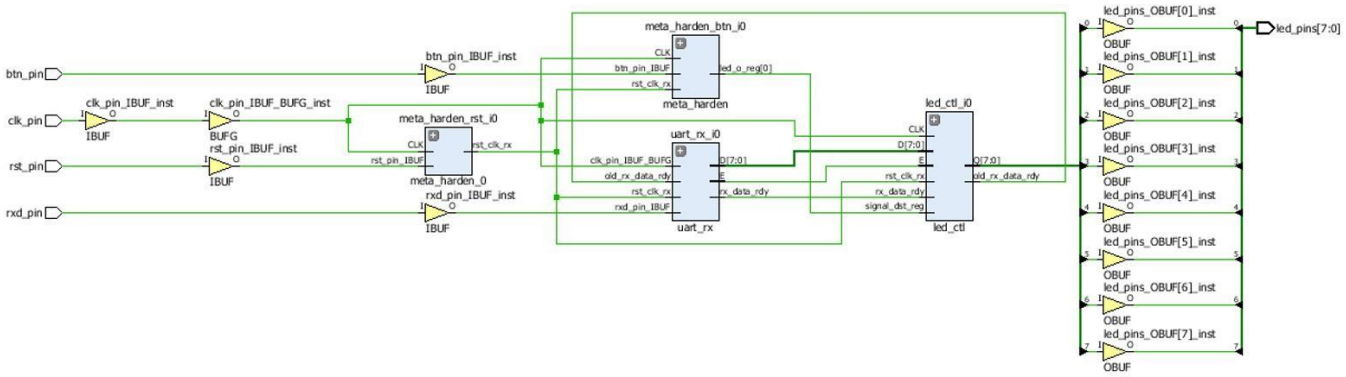


Рис. 7. Схематичний вид синтезованого проекту

Зауважимо, що IBUF і OBUF автоматично додаються до проекту так як входи і виходи буферизуються. Тут ще добавлено чотири модулі нижчого рівня.

3.1.6. Подвійно клацнути на екземплярі **uart_rx_i0** на схематичному виді щоб переглянути екземпляри нижчого рівня.

3.1.7. Вибрати екземпляр **uart_baud_gen_rx_i0**, клацнути правою клавішею, і вибрати *Go To Source.*

Зауважимо, що рядок 84 підсвічується. Також зауважимо, що параметри CLOCK_RATE і BAUD_RATE передаються викликуваному модулю.

3.1.8. Подвійно клацнути на екземпляр **meta_harden_rxd_io,** щоб побачити як реалізується схема синхронізації з використанням двох тригерів (FFs – flip-flops). Ця синхронізація потрібна для зменшення ймовірності метастабільності.

3.1.9. Клацнути ( ) на схематичному виді для повернення до свого батьківського блоку.

3.2. Аналіз часового звіту.

3.2.1. У вікні *Flow Navigator* вибрати *Synthesis* рядок *Synthesized Design* і клацнути на

Report Timing Summary.

3.2.2. Клацнути **OK** для генерації звіту Timing_1.

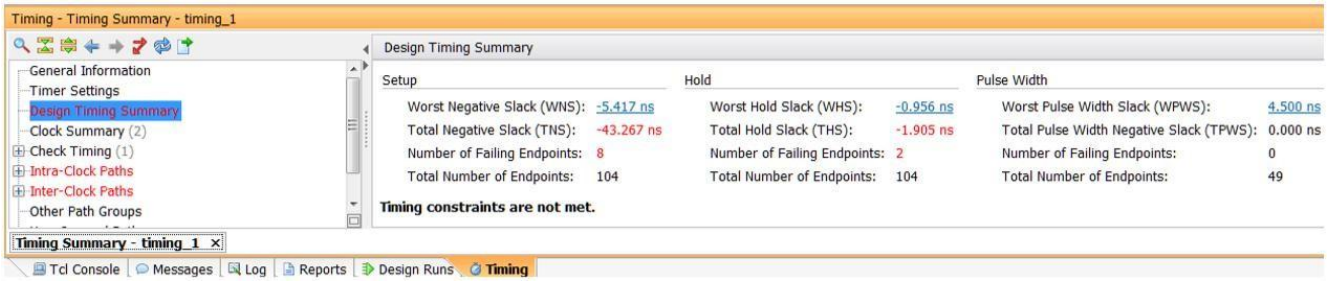
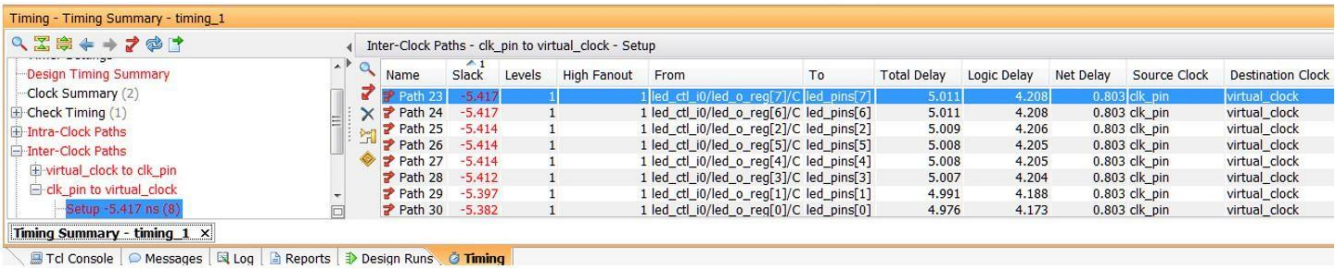


Рис. 8. Часовий звіт для Nexys4 DDR

Зауважимо, що входи Design Timing Summary і Inter-Clock Paths у лівій закладці підсвічені червоним, що вказує на порушення часового обмеження. У правій закладці, інформація згрупована у стовпцях Setup, Hold і Width.

Під стовпцем є посилання Setup Worst Negative Slack (WNS), клацання на ньому відкриває, сформовані шляхи з недоліками. Total Negative Slack підсвічений червоним, що вказує на загальне число недоліків у проекті і Number of Failing Endpoints вказує на загальне число незадовільних кінцевих точок.

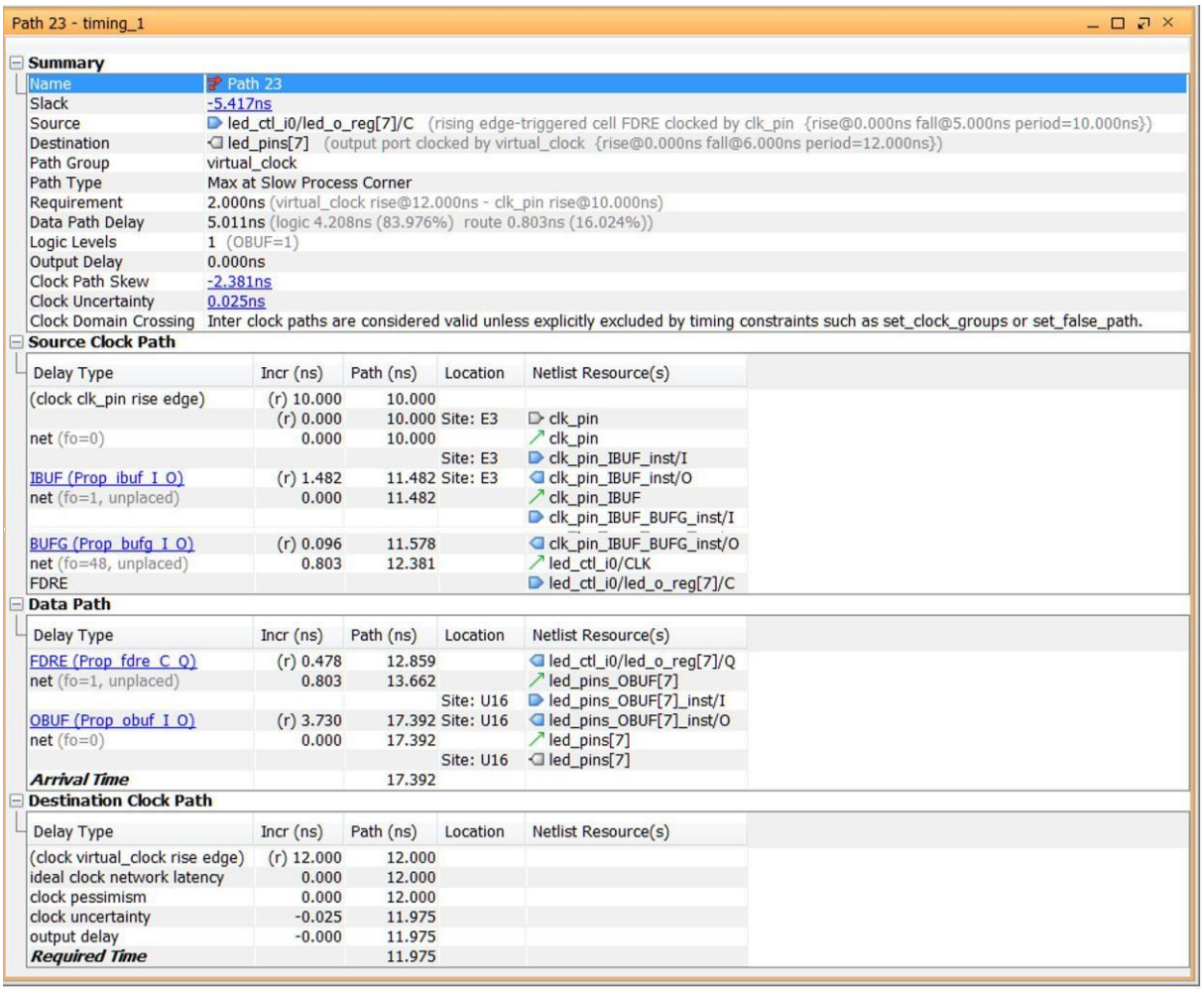
3.2.3. Клацнути на посилання WNS і переглянути 8 незадовільних шляхів.



Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Source Clock	Destination Clock
Path 23	-5.417	1	1	led_ctl_i0/led_o_reg[7]/C	led_pins[7]	5.011	4.208	0.803	clk_pin	virtual_clock
Path 24	-5.417	1	1	led_ctl_i0/led_o_reg[6]/C	led_pins[6]	5.011	4.208	0.803	clk_pin	virtual_clock
Path 25	-5.414	1	1	led_ctl_i0/led_o_reg[2]/C	led_pins[2]	5.009	4.206	0.803	clk_pin	virtual_clock
Path 26	-5.414	1	1	led_ctl_i0/led_o_reg[5]/C	led_pins[5]	5.008	4.205	0.803	clk_pin	virtual_clock
Path 27	-5.414	1	1	led_ctl_i0/led_o_reg[4]/C	led_pins[4]	5.008	4.205	0.803	clk_pin	virtual_clock
Path 28	-5.412	1	1	led_ctl_i0/led_o_reg[3]/C	led_pins[3]	5.007	4.204	0.803	clk_pin	virtual_clock
Path 29	-5.397	1	1	led_ctl_i0/led_o_reg[1]/C	led_pins[1]	4.991	4.188	0.803	clk_pin	virtual_clock
Path 30	-5.382	1	1	led_ctl_i0/led_o_reg[0]/C	led_pins[0]	4.976	4.173	0.803	clk_pin	virtual_clock

Рис. 9. 8 незадовільних шляхів Nexys4 DDR

Подвійне клацанням на шляху **Path 23** відкриває вікно із отриманими параметрами шляху.



Name	Path 23
Slack	-5.417ns
Source	led_ctl_i0/led_o_reg[7]/C (rising edge-triggered cell FDRE clocked by clk_pin {rise@0.000ns fall@5.000ns period=10.000ns})
Destination	led_pins[7] (output port clocked by virtual_clock {rise@0.000ns fall@6.000ns period=12.000ns})
Path Group	virtual_clock
Path Type	Max at Slow Process Corner
Requirement	2.000ns (virtual_clock rise@12.000ns - clk_pin rise@10.000ns)
Data Path Delay	5.011ns (logic 4.208ns (83.976%) route 0.803ns (16.024%))
Logic Levels	1 (OBUF=1)
Output Delay	0.000ns
Clock Path Skew	-2.381ns
Clock Uncertainty	0.025ns
Clock Domain Crossing	Inter clock paths are considered valid unless explicitly excluded by timing constraints such as set_clock_groups or set_false_path.

Source Clock Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock clk_pin rise edge)	(r) 10.000	10.000		
	(r) 0.000	10.000	Site: E3	clk_pin
net (fo=0)	0.000	10.000		clk_pin
			Site: E3	clk_pin_IBUF_inst/I
IBUF (Prop_ibuf_I_O)	(r) 1.482	11.482	Site: E3	clk_pin_IBUF_inst/O
net (fo=1, unplaced)	0.000	11.482		clk_pin_IBUF
				clk_pin_IBUF_BUFG_inst/I
BUFG (Prop_bufg_I_O)	(r) 0.096	11.578		clk_pin_IBUF_BUFG_inst/O
net (fo=48, unplaced)	0.803	12.381		led_ctl_i0/CLK
FDRE				led_ctl_i0/led_o_reg[7]/C

Data Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
FDRE (Prop_fdre_C_Q)	(r) 0.478	12.859		led_ctl_i0/led_o_reg[7]/Q
net (fo=1, unplaced)	0.803	13.662		led_pins_OBUF[7]
			Site: U16	led_pins_OBUF[7]_inst/I
OBUF (Prop_obuf_I_O)	(r) 3.730	17.392	Site: U16	led_pins_OBUF[7]_inst/O
net (fo=0)	0.000	17.392		led_pins[7]
			Site: U16	led_pins[7]
Arrival Time		17.392		

Destination Clock Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock virtual_clock rise edge)	(r) 12.000	12.000		
ideal clock network latency	0.000	12.000		
clock pessimism	0.000	12.000		
clock uncertainty	-0.025	11.975		
output delay	-0.000	11.975		
Required Time		11.975		

Рис. 10. Найбільш незадовільний шлях для Nexys4 DDR

Зауважимо, що це тільки оціночні значення. Мережа специфікувалася як нерозміщувана і були назначені всі значення за замовчування (0.800 нс).

Дійсні затримки маршрутизації не розглядувалися.

3.3. Генерація звіту про використані ресурси і споживану потужність.

3.3.1. У вікні *Flow Navigator* вибрати *Synthesis* рядок *Synthesized Design* і клацнути на **Report Utilization** та клацнути **OK** для генерації звіту про використані ресурси.

Ресурси	Використання	Доступних	Використано, %
Slice LUTs	41	63400	0.06
Slice registers	48	126800	0.04
IO	12	120	10.00
Clocking	1	32	3.12

Рис. 11. Звіт про використання ресурсів для Nexsys4 DDR

Запитання 2

Продивитися звіт знайти число використань кожного з наступних ресурсів: FF:_____
LUT: _____
I/O: _____
BUFG: _____

3.3.2. Вибрати вхід Slice LUTs у лівій закладці і знайти використання екземплярів ресурсів нижчого рівня. Можна розкрити екземпляри у правій закладці щоб побачити повну ієрархію використань.

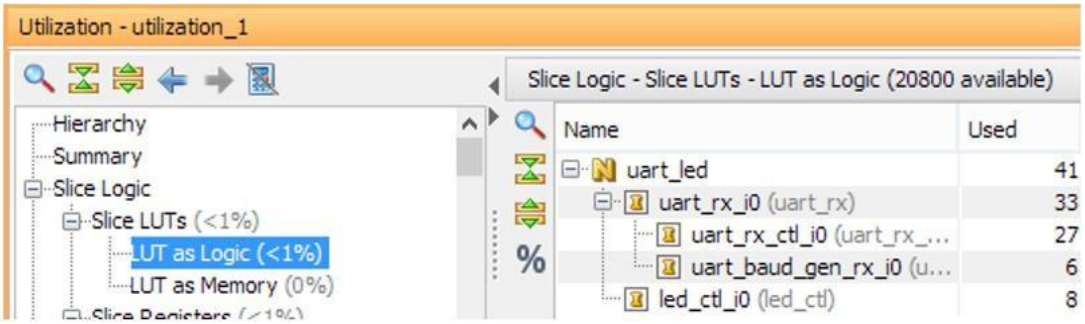
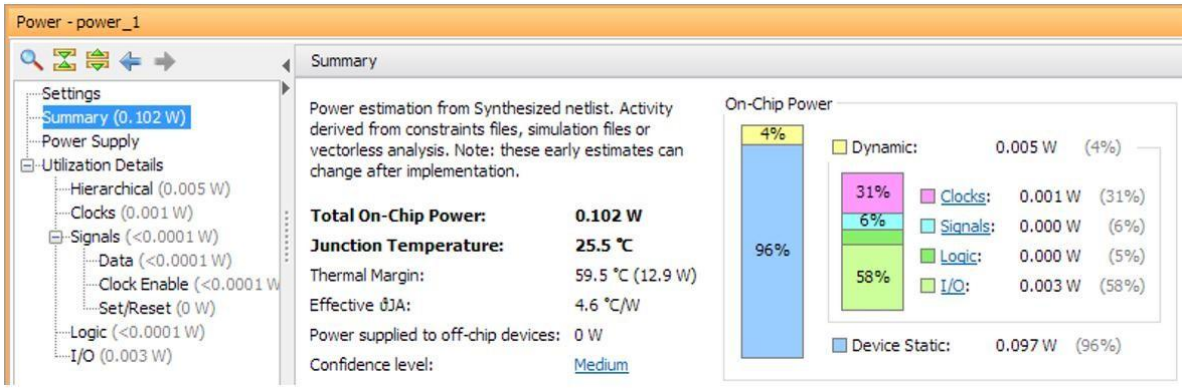


Рис. 12. Використання модулів нижчого рівня у Nexys4 DDR

3.3.3. У вікні *Flow Navigator* вибрати *Synthesis* рядок *Synthesized Design* і клацнути на **Report Power** та клацнути **OK** для генерації звіту оціночного споживання енергії з використанням значень за замовчуванням.

Зауважимо, що це є тільки оціночні значення, так як відсутні дані для запуску моделювання і не має точного значення частоти або була введена



інформація про середовище.

Рис. 13. Оцінка споживання потужності для Nexys4 DDR

Запитання 3

Із звіту потужності, знайти % споживаної потужності, який використовується кожним із наступних елементів:

Можна перемістити вказівник миші на прямокутник з процентами щоб подивитися споживання.

3.4. Записати контрольну точку для того щоб проаналізувати результати без виконання реального процесу синтезу.

3.4.1. Вибрати **File > Write Checkpoint…** для збереження обробленого проекту так, щоб він міг бути відкритий пізніше для аналізу.

3.4.2. Появиться діалогове вікно показуючи ім'я файлу за замовчуванням у поточному каталозі проекту.

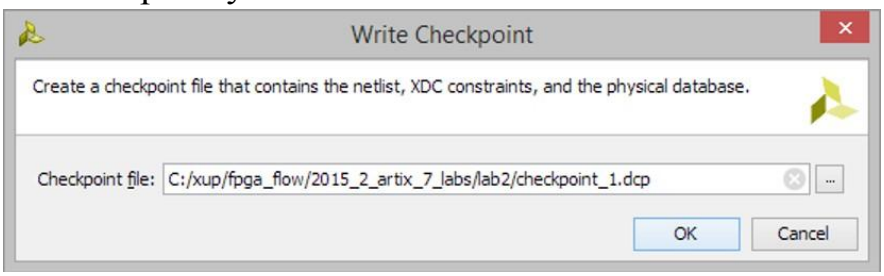


Рис. 14. Запис контрольної точки

3.4.3. Клацнути **OK**.

3.5. Змінити установлення синтезу в проекті на плоский (flatten). Перезапустити синтез проекту і проаналізувати результати.

3-5-1. У вікні *Flow Navigator* вибрати *Project Manager* клацнути на рядок *Project Settings*. Клацнути на **Project Settings**. У вікні *Project Settings*, яке відкриється, вибрати **Synthesis**.

3-5-2. Розкрити випадаючий список **flatten_hierarchy** і вибрати елемент **full** для отримання плоского проекту.

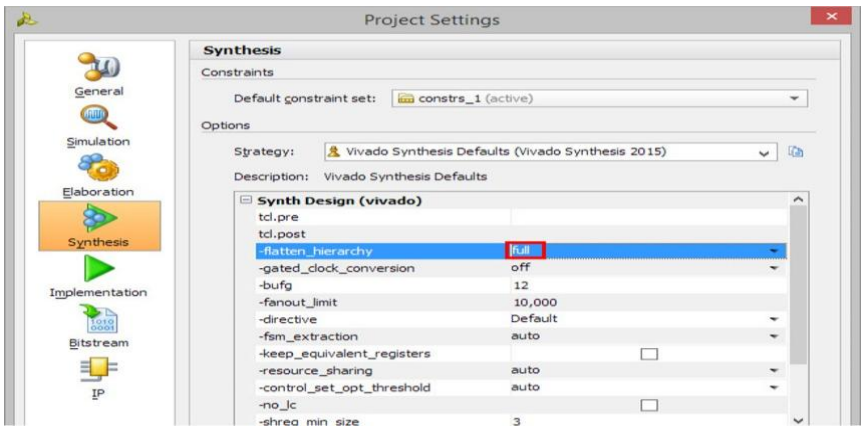


Рис. 15. Вибір опції для отримання плоского проекту

Клацнути **ОК**.

3.5.3. Появиться діалогове вікно Create New Run із запитом на нове виконання так як установлення були змінені.

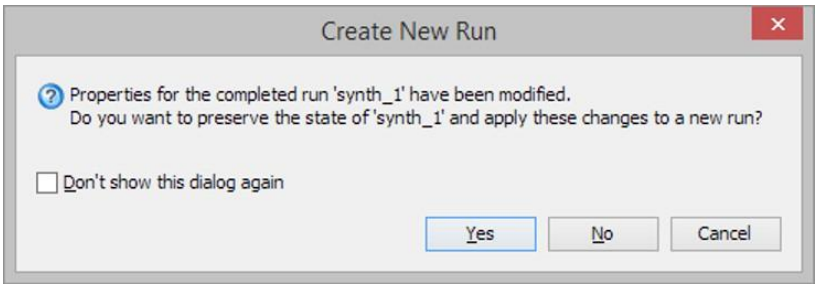


Рис. 16. Діалогове вікно Create New Run

3.5.4. Клацнути **Yes**.

3.5.5. Змінити ім’я з **synth_2** на **synth_flatten** і клацнути **OK**.

3.5.6. У вікні *Flow Navigator* вибрати Synthesis клацнути на рядок **Run Synthesis** для синтезу проекту.

3.5.7. Клацнути **Save, OK**, і знову **OK** для збереження синтезованого проекту і збереження обмежень.

Діалогове вікно Reload Design може повторно появитися. Клацнути **Cancel**.

3.5.8. Клацнути **OK** для відкриття синтезованого проекту коли процес синтезу закінчиться.

3.5.9. У вікні *Flow Navigator* вибрати *Synthesis* клацнути на *Open Synthesized Design* і на **Schematic** для перегляду синтезованого проекту у схематичному виді.

Зауважимо, що проект є повністю плоский.

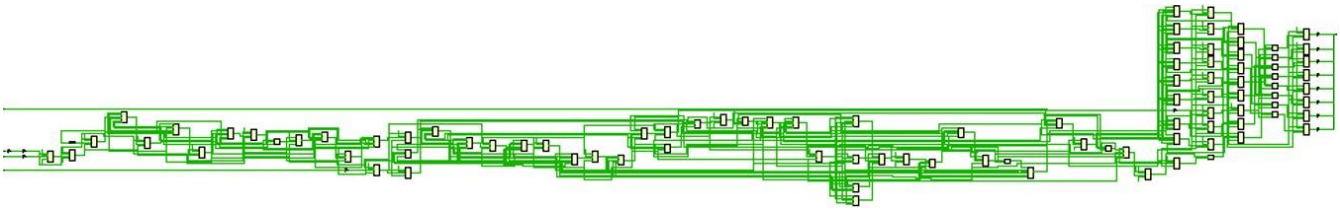


Рис. 17. Проект перетворений у плоский

3.5.10. У вікні *Flow Navigator* вибрати *Synthesis* клацнути на *Open Synthesized Design* і клацнути на **Report Utilization** та побачити, що ієрархічне використання більше не доступне. Також зауважимо, що число **Slice Registers** є **48**.

3.6. Записати контрольну точку для того, щоб аналізувати результати без виконання реального процесу синтезу.

3.6.1. Вибрати **File > Write Checkpoint…** для збереження обробленого проекту так, щоб його можна було відкрити пізніше для подальшого аналізу.

3.6.2. Появиться діалогове вікно, яке буде показувати ім'я файлу за замовчуванням (checkpoint_2.dcp) у каталозі поточного проекту.

3.6.3. Клацнути **OK**.

3.6.4. Закрити проект вибором **File > Close Project.**

Крок 4. Читання контрольних точок

4.1. Прочитати попередньо збережену контрольну точку (checkpoint_1) для аналізу результатів без дійсного виконання процесу синтезу.

4.1.1. Вибрати **File > Open Checkpoint…** на екрані *Getting Started.*

4.1.2. Переглянути **<2015_2_artix7_labs>\lab2** і вибрати **checkpoint_1**.

4.1.3. Клацнути **OK**.

4.1.4. Якщо схема не відкривається за замовчуванням, у закладці *Netlist*, вибрати екземпляр верхнього рівня, **uart_led**, клацнути правою клавішею миші і вибрати **Schematic**.

Можна буде побачити ієрархічні блоки. Можна подвійно клацнути на любому з блоків верхнього рівня і побачити блоки нижчого рівня. Можна також клацнути правою кнопкою миші на любий блок нижчого і вибравши **Schematic** побачити проект відповідного рівня.

4.1.5. У закладці *Netlist*, вибрати екземпляр модуля верхнього рівня **uart_led**, клацнути на ньому правою клавішею і вибрати **Show Hierarchy**.

Можна побачити ієрархічне з'єднання блоків.

4.1.6. Вибрати елемент меню **Tools > Timing > Report Timing Summary** і клацнути **OK**

щоб побачити звіт, який переглядався раніше.

4.1.7. Вибрати **Tools > Report > Report Utilization…** і клацнути **OK** щоб побачити звіт використання ресурсів, який переглядався раніше.

4.1.8. Вибрати **File > Open Checkpoint**, знайти **<2015_2_artix7_labs>\lab2** і вибрати **checkpoint_2**.

4.1.9. Клацнути **No** для збереження відкритим Checkpoint_1. Це запустить друге середовище

Vivado GUI.

4.1.10. Якщо схема не відкривається за замовчуванням, у закладці *Netlist* вибрати екземпляр верхнього рівня, **uart_led**, клацнути правою кнопкою миші і вибрати **Schematic**.

Можна буде побачити плоский проект.

4.1.11. При бажанні можна згенерувати потрібний звіт на цій контрольній точці.

4.1.12. Закрити програму **Vivado** вибором **File > Exit** і клацнути **OK**.

Завдання до лабораторної роботи №6

Реалізуйте прийомо-передавач із виводом даних на модулі виводу навчальної плати Nexys 4DDR.

Номер варіанту	Модуль виводу
1.	Семисегментний індикатор 1 секції
2.	Семисегментний індикатор 2 секції
3.	LED-світлодіоди
4.	RGB-світлодіоди
5.	VGA-дисплей

Зміст звіту

1. Титульна сторінка.
2. Назва та мета лабораторної роботи.
3. Навести RTL-схеми проекту та їх VHDL-код.
4. Подати звіти часових та електричних характеристик проекту.
5. Висновки.

ЛАБОРАТОРНА РОБОТА №7

Тема роботи: технології моделювання складних схем засобами *ModelSim* фірми *Mentor Graphics*.

Мета роботи: засвоїти навики роботи в середовищі *ModelSim* та навчитися виконувати симуляцію та відладку розроблених проектів на мовах опису апаратних засобів.

Короткі теоретичні відомості

ModelSim – це потужний засіб для симуляції та відлагодження програм написаних з використанням мов опису апаратних засобів (HDL-мови). Іноді ModelSim використовують як платформо-незалежне середовище для вивчення мов опису апаратних засобів VHDL та Verilog. В лабораторній роботі буде використовуватися програмне середовище ModelSim Altera Starter Edition версії 10.1b компанії Альтера (Інтел) інформація про використану версію подана на рис. 1.

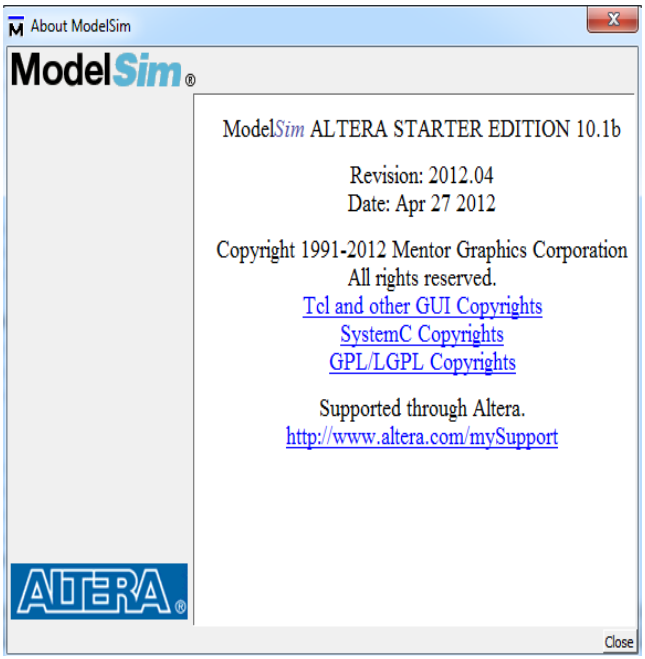
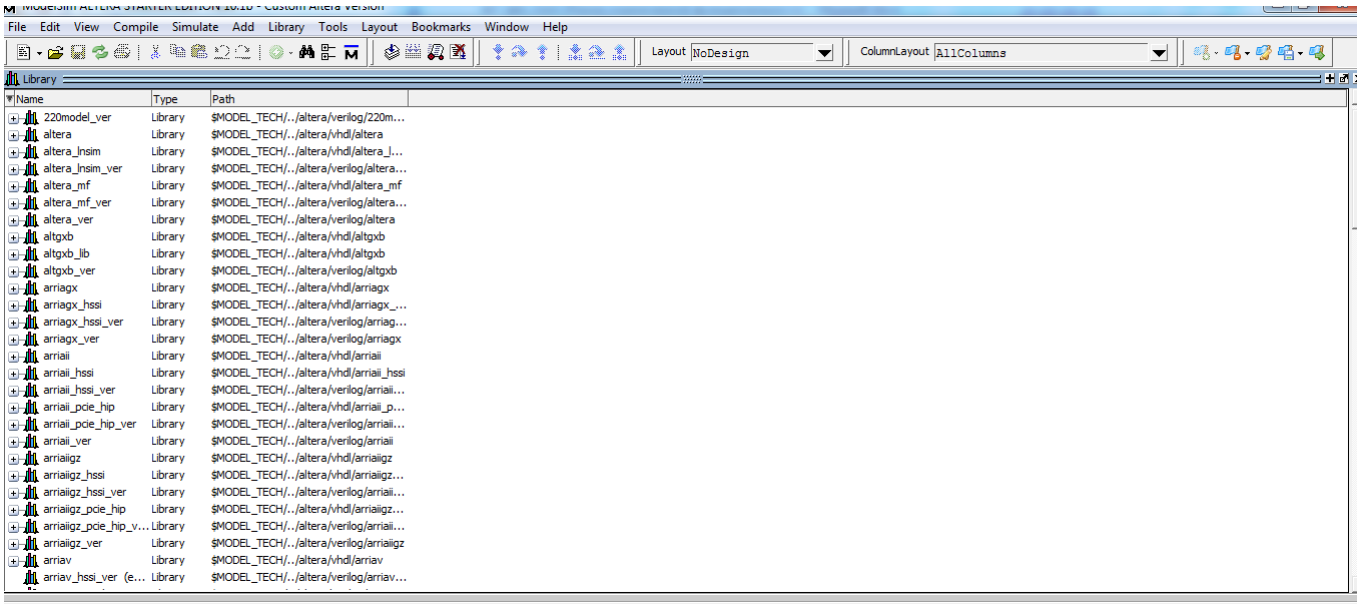


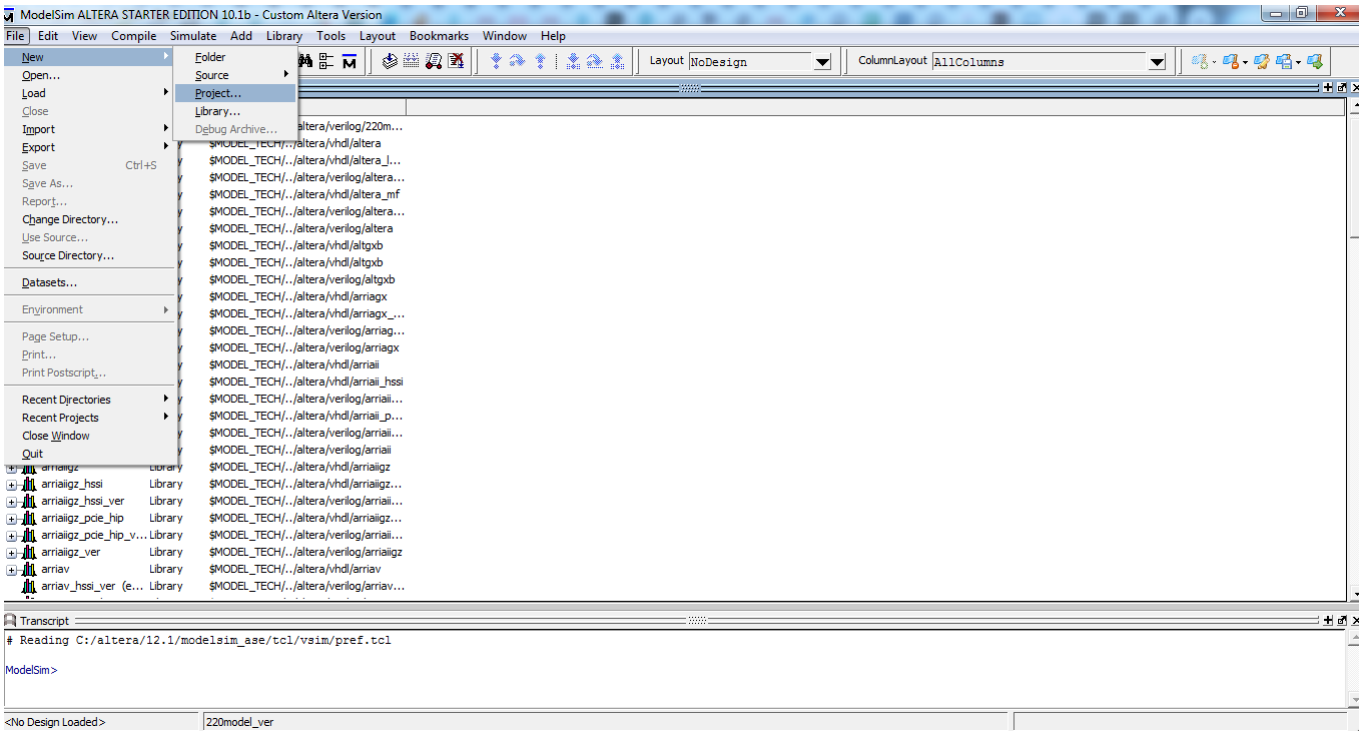
Рис. 1. Програмний продукт ModelSim Altera Starter Edition фірми Mentor Graphics права на який має компанія Альтера (входить в компанію Intel).

Порядок виконання роботи

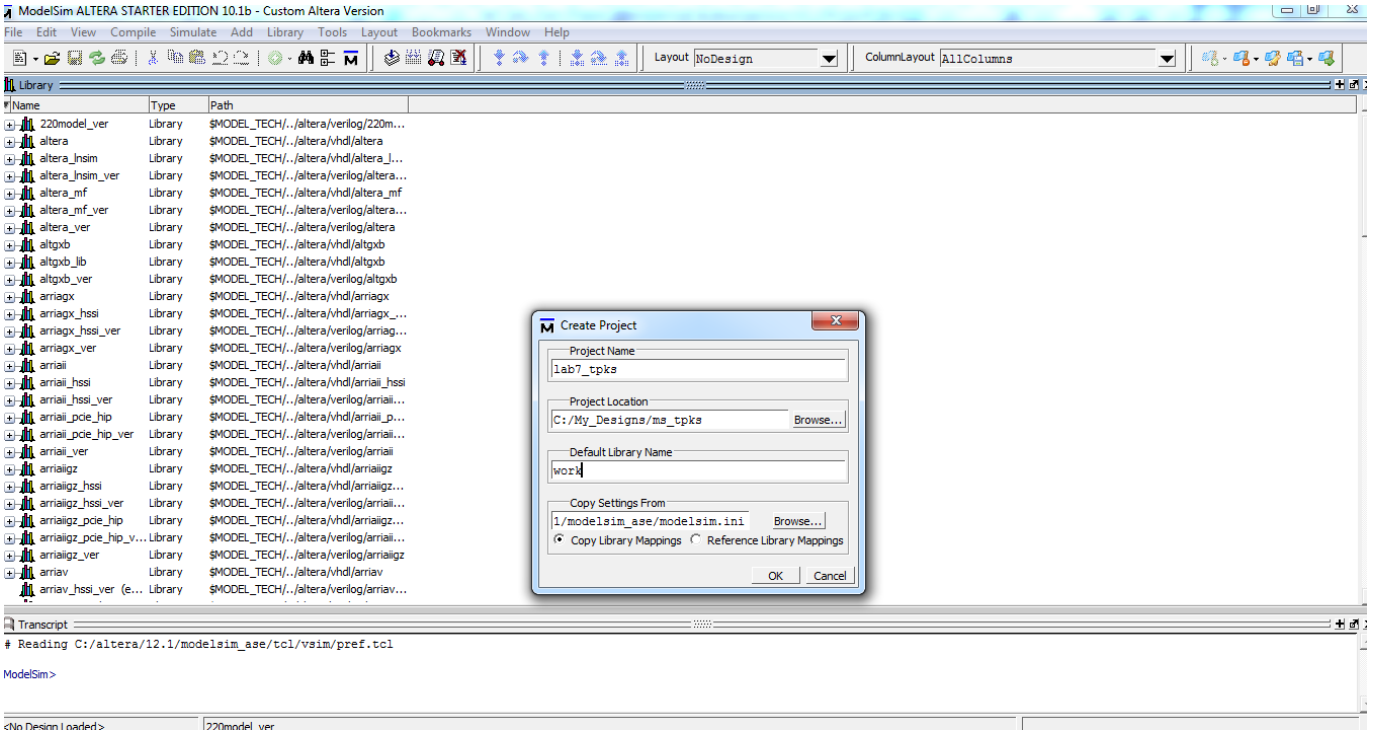
1. Запустити програмне середовище симуляції і відладки ModelSim.



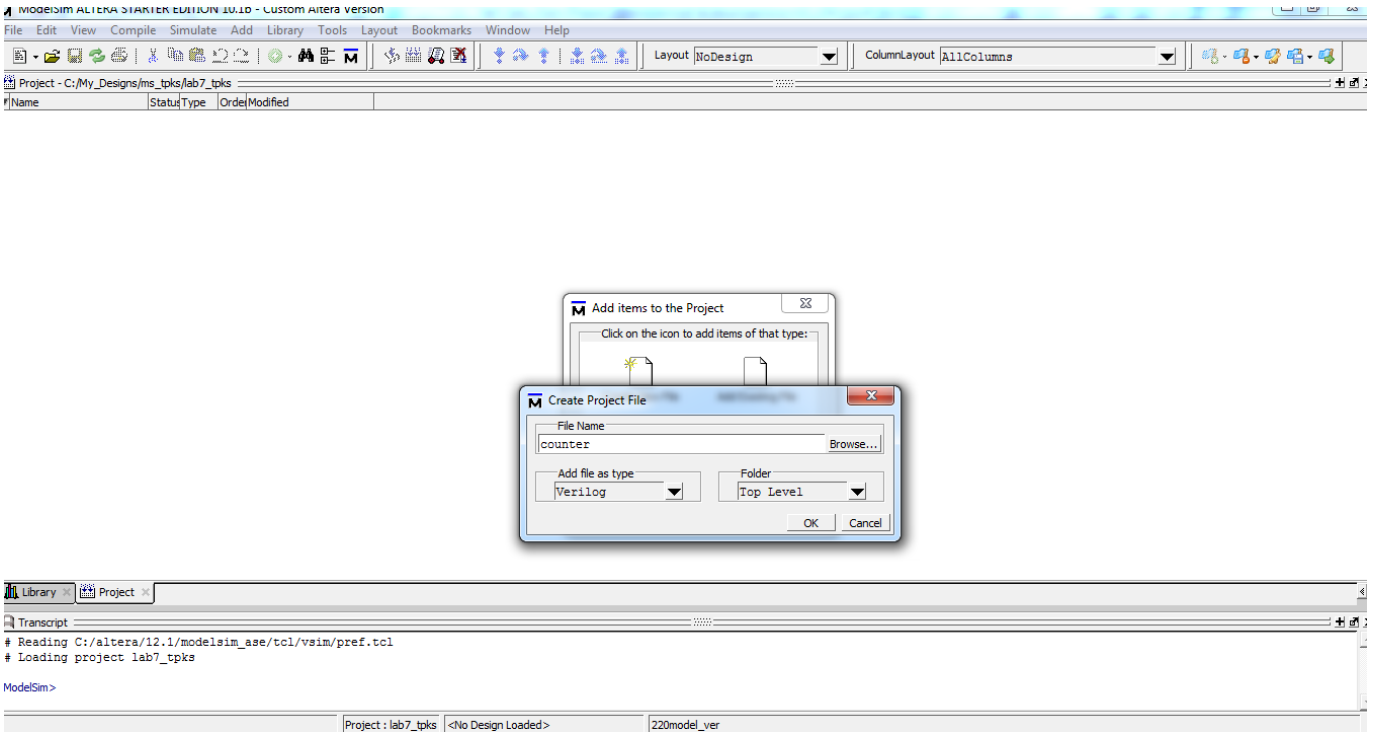
2. Створити новий проект натиснувши File->New->Project...



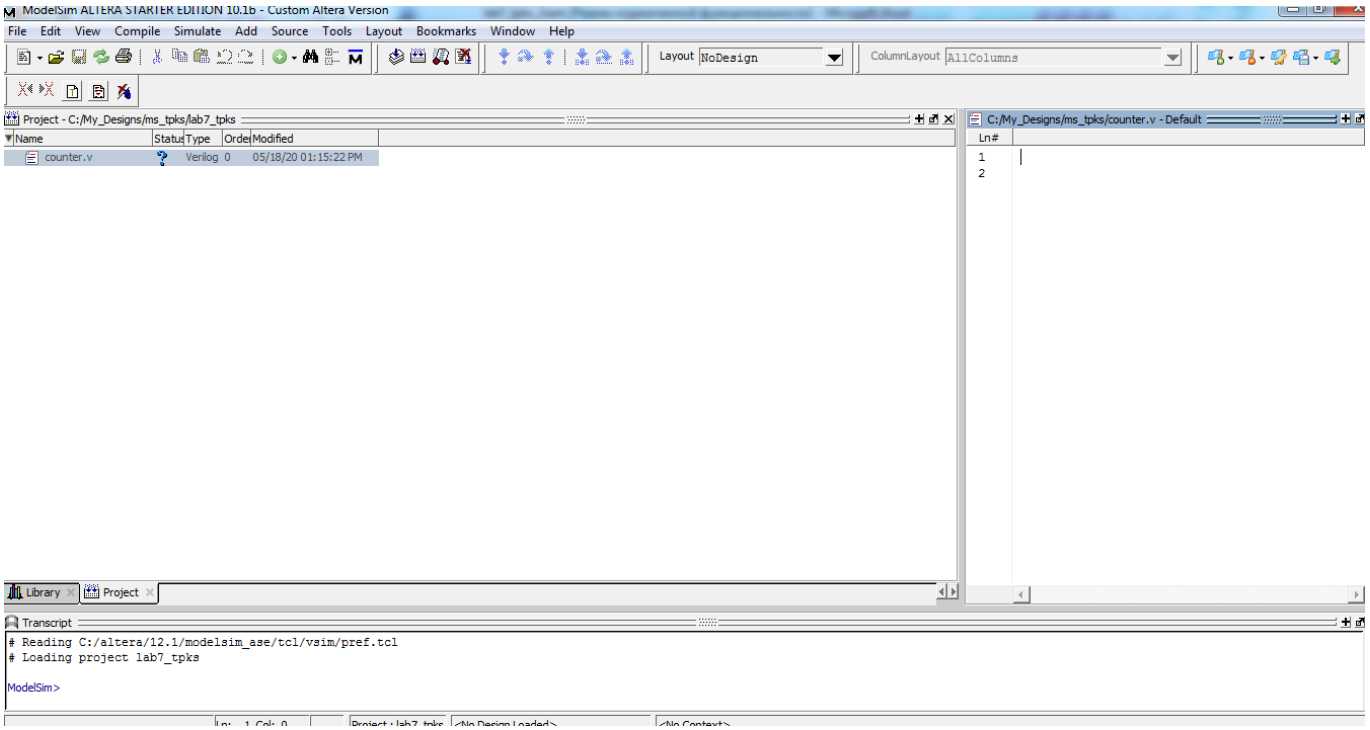
3. Ввести назву проекту, вибрати шлях його розміщення та за задати ім’я бібліотеки та натиснути Ок.



4. Введіть назву основного файлу counter та виберіть тип його мови опису Verilog та натисніть Ок.



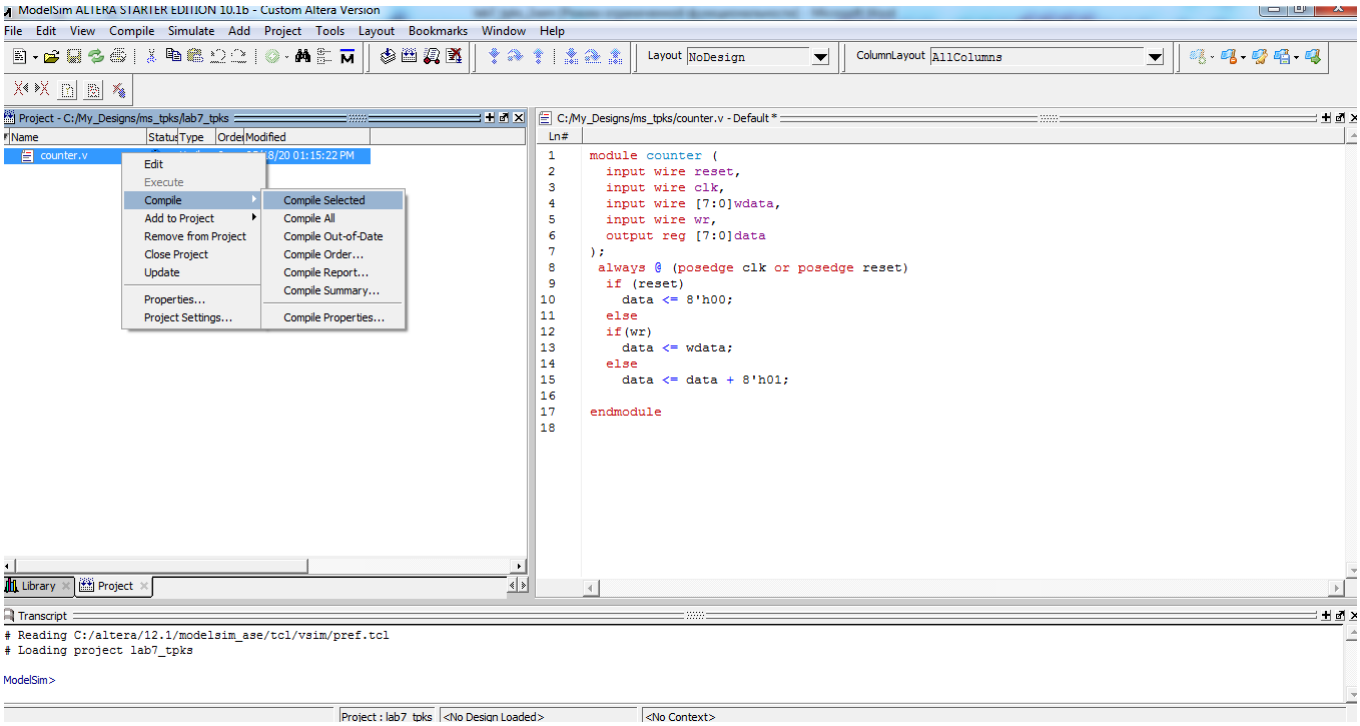
5. Двічі клацніть на файл counter.v



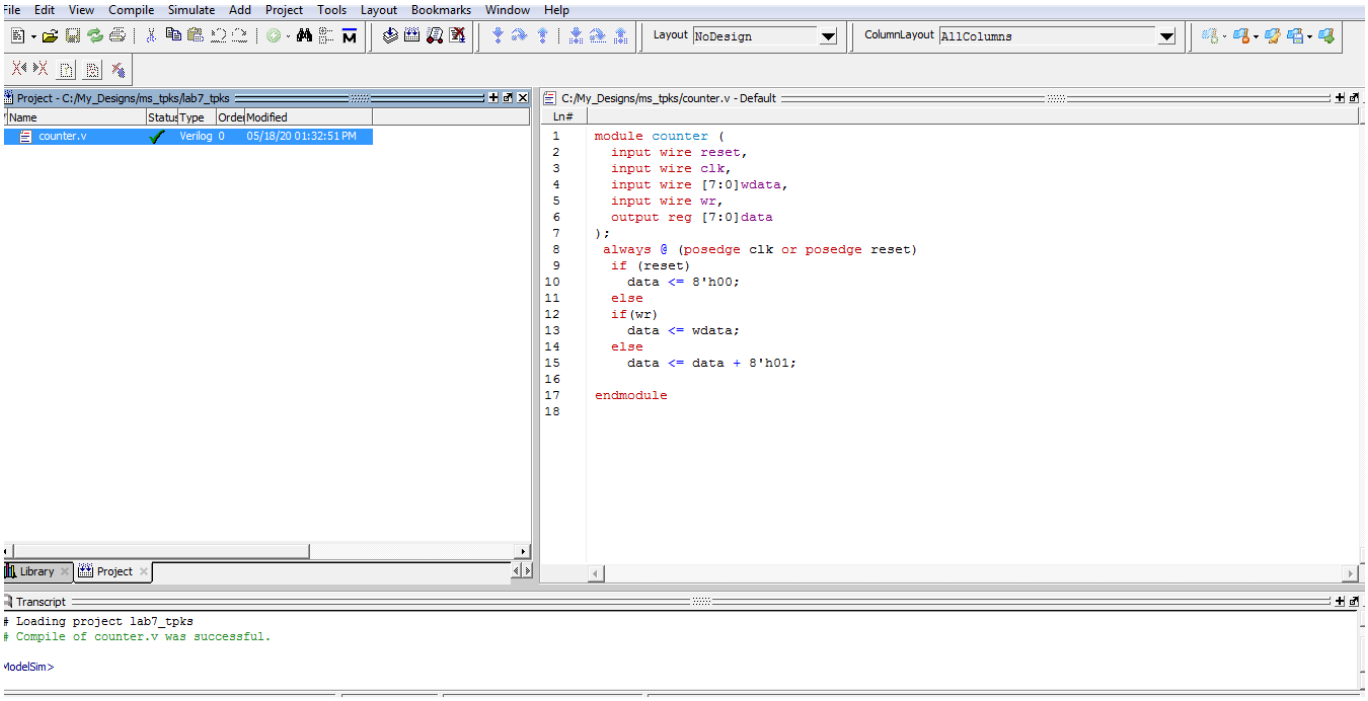
6. Вставте поведінковий опис лічильника на мові Verilog у поле файлу counter.v

```
module counter (
  input wire reset,
  input wire clk,
  input wire [7:0]wdata,
  input wire wr,
  output reg [7:0]data
);
 always @ (posedge clk or posedge reset)
  if (reset)
    data <= 8'h00;
  else
  if(wr)
    data <= wdata;
  else
    data <= data + 8'h01;

endmodule
```

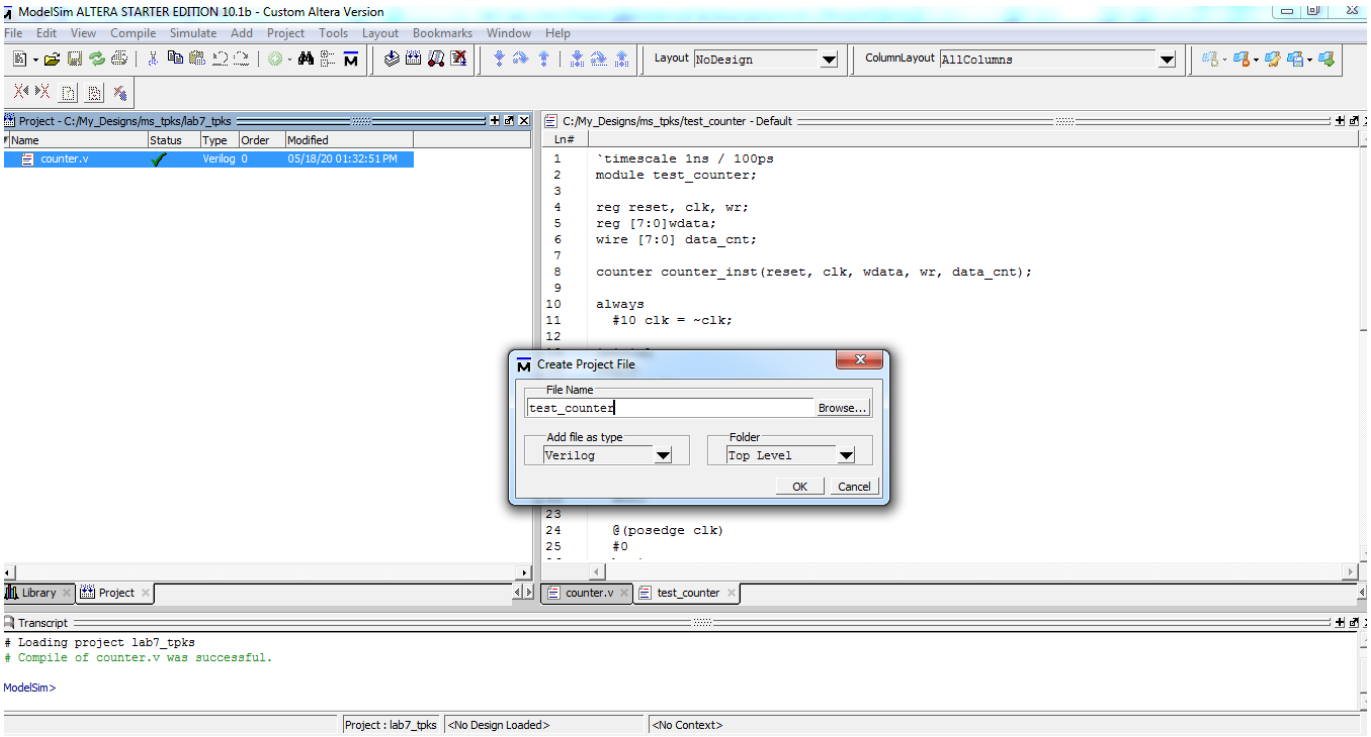
7. Виконайте компіляцію даного файлу виділивши його та вибравши Compile -> Compile Selected.



8. Результат компіляції відображається у вікні Transcript, у разі успішної компіляції файл буде відмічений зеленою галочкою.



9. У ModelSim є багато шаблонів для створення файлу автоматичної генерації тестової послідовності сигналів (testbench), а також є можливість додати у проект створений окремо тестовий файл вручну. Для цього виділяємо файл counter.v натискаємо праву клавішу миші та вибираємо Add to Project -> New File. Задаємо назву файл test_counter та вибираємо його тип Verilog.

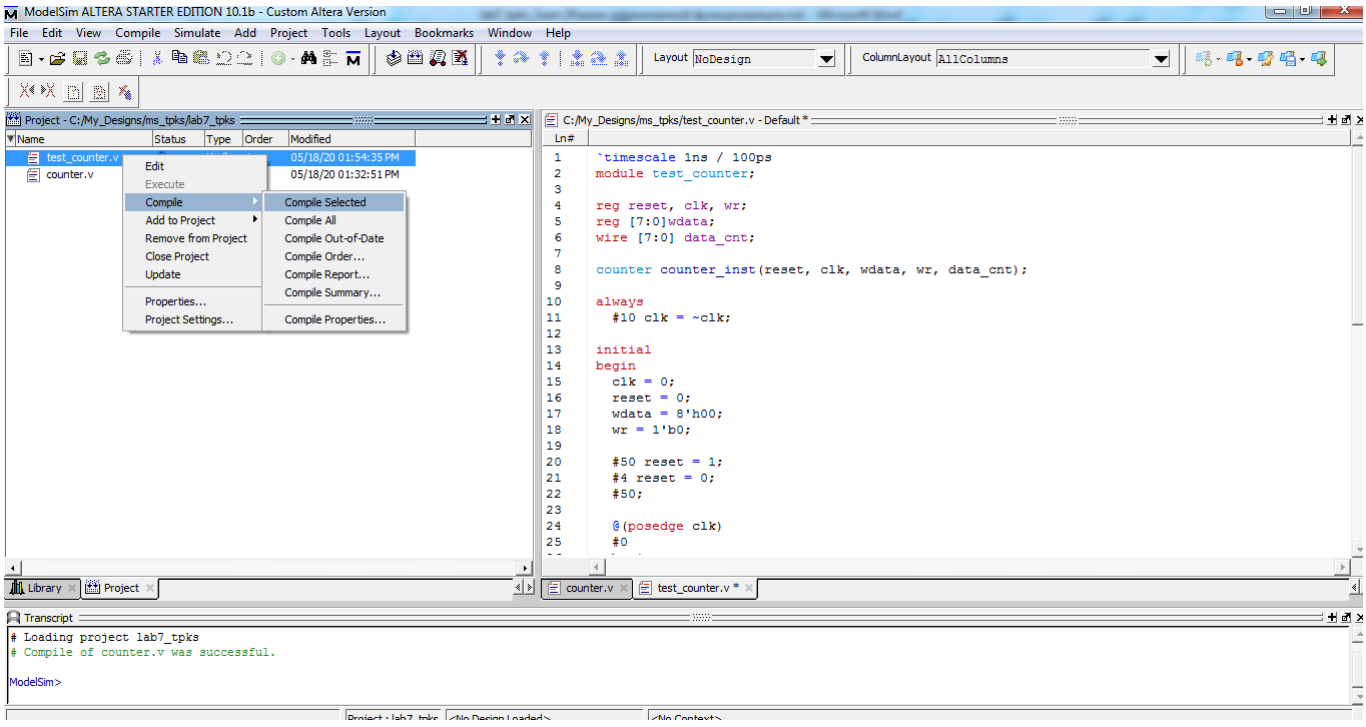


10. Відкривши файл test_counter.v вставляємо в нього готову тестову послідовність.

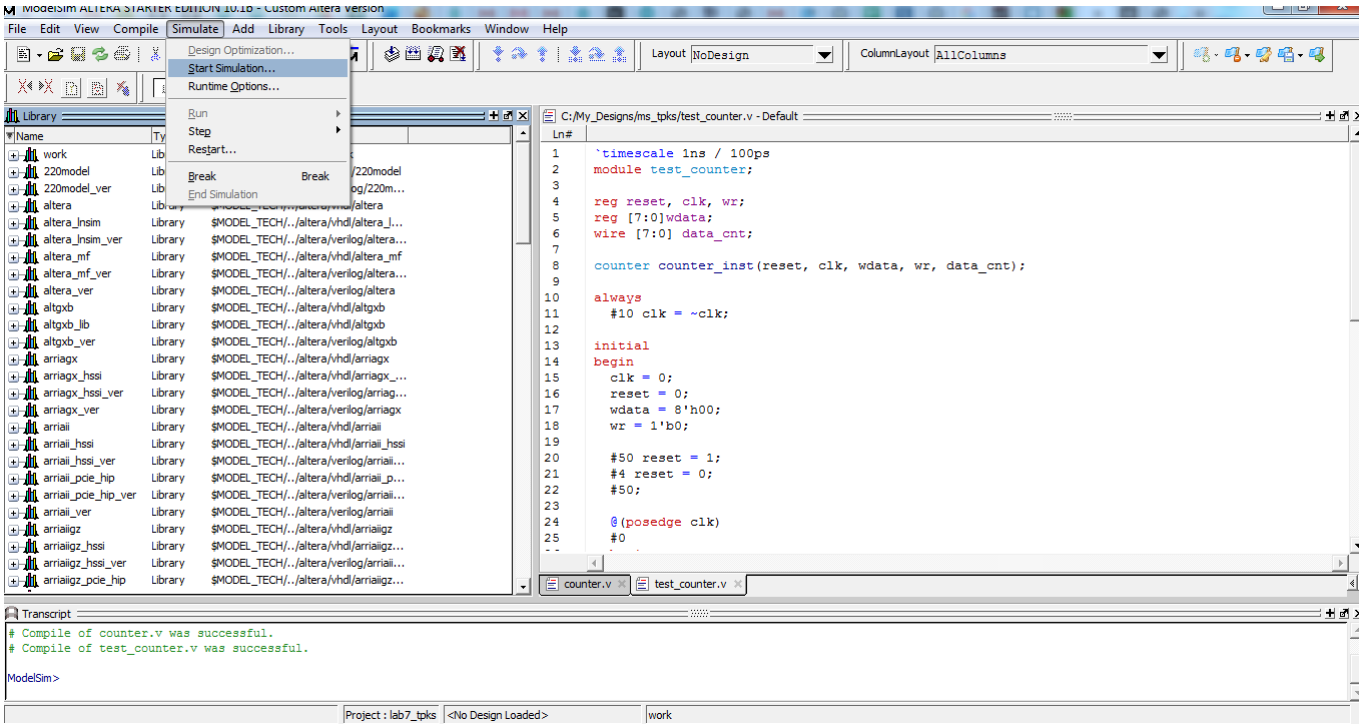
```
`timescale 1ns / 100ps
module test_counter;

reg reset, clk, wr;
reg [7:0]wdata;
wire [7:0] data_cnt;

counter counter_inst(reset, clk, wdata, wr, data_cnt);

always
  #10 clk = ~clk;

initial
begin
  clk = 0;
  reset = 0;
  wdata = 8'h00;
  wr = 1'b0;

  #50 reset = 1;
  #4 reset = 0;
  #50;

  @(posedge clk)
  #0
  begin
    wdata = 8'h55;
```

```
    wr = 1'b1;
  end

  @(posedge clk)
  #0
  begin
    wdata = 8'h00;
    wr = 1'b0;
  end
end

initial
begin
  #400 $finish;
end

initial
begin
  $dumpfile("out.vcd");
  $dumpvars(0,test_counter);
  $dumpvars(0,counter_inst);
end

initial
  $monitor($stime,, reset,, clk,,, wdata,, wr,, data_cnt);

endmodule
```

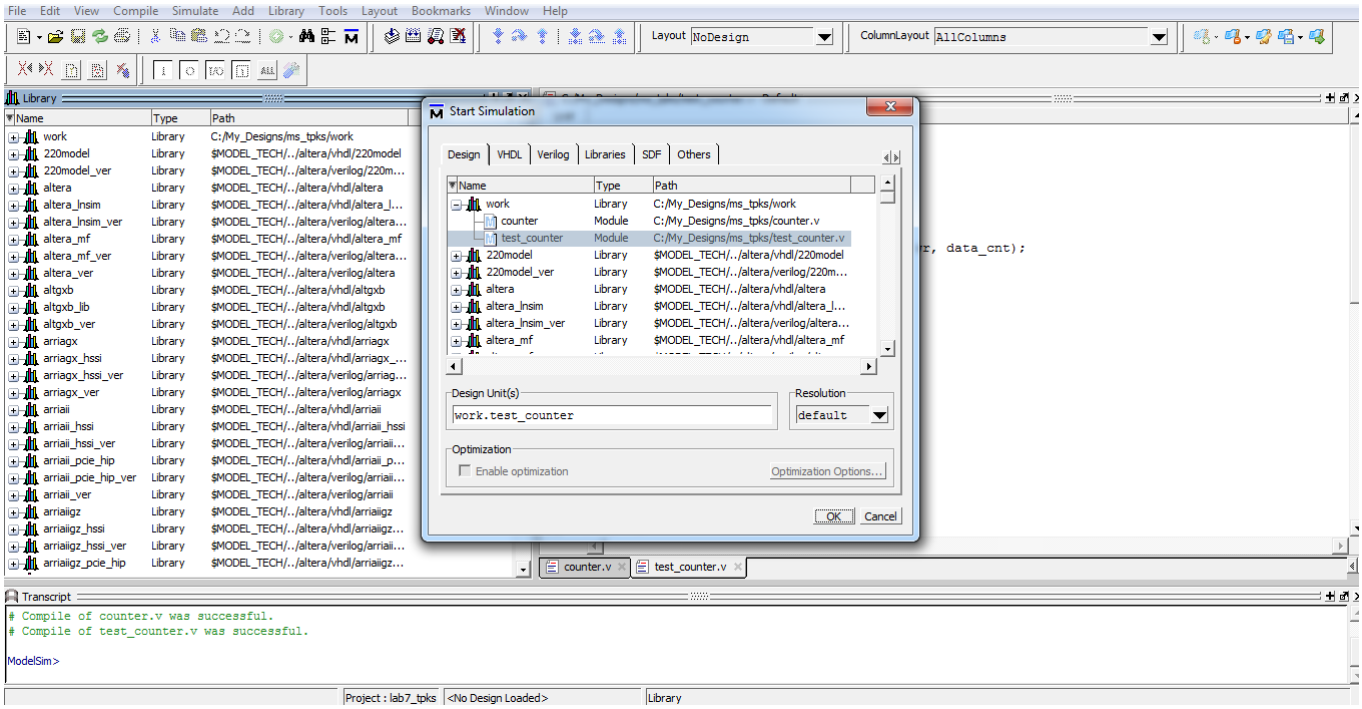
11. Виконуємо компіляцію даного файлу виділивши його та вибравши Compile -> Compile Selected.



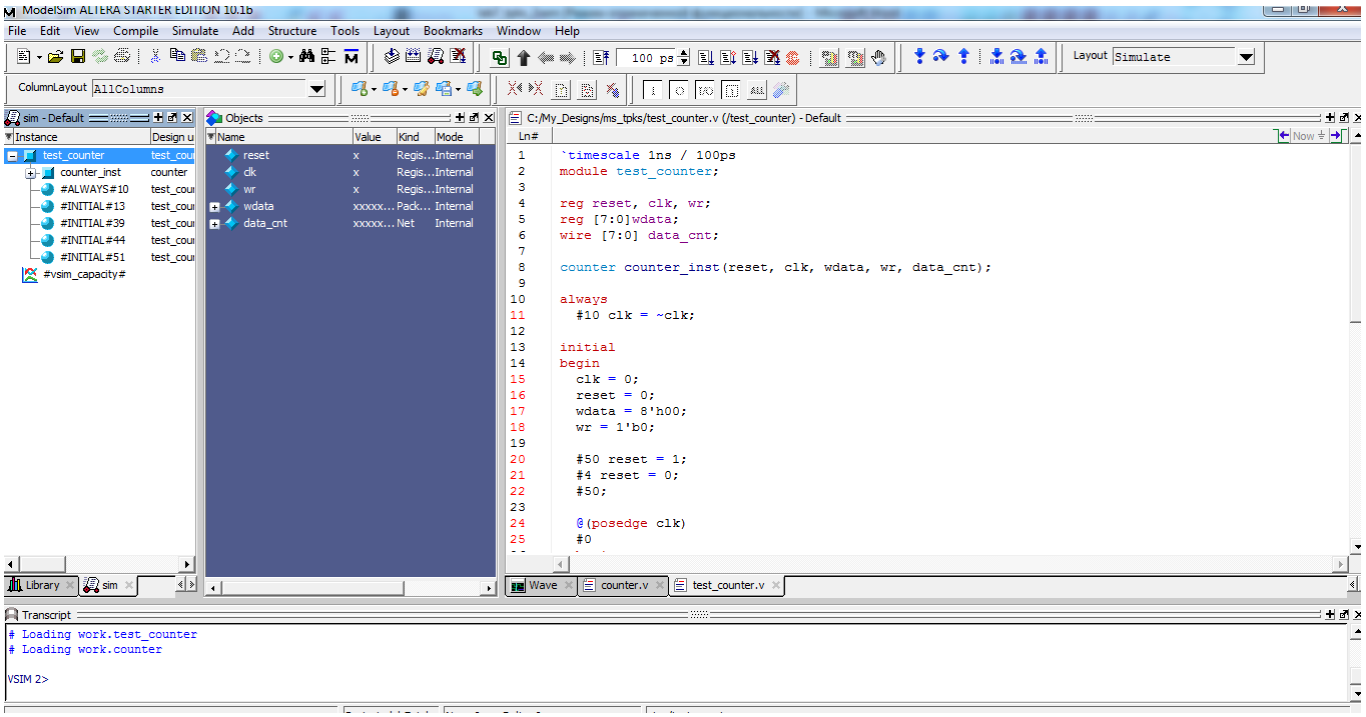
12. Наступним кроком виконуємо симуляцію (моделювання) проекту вибравши в основному меню Simulate -> Start Simulation…



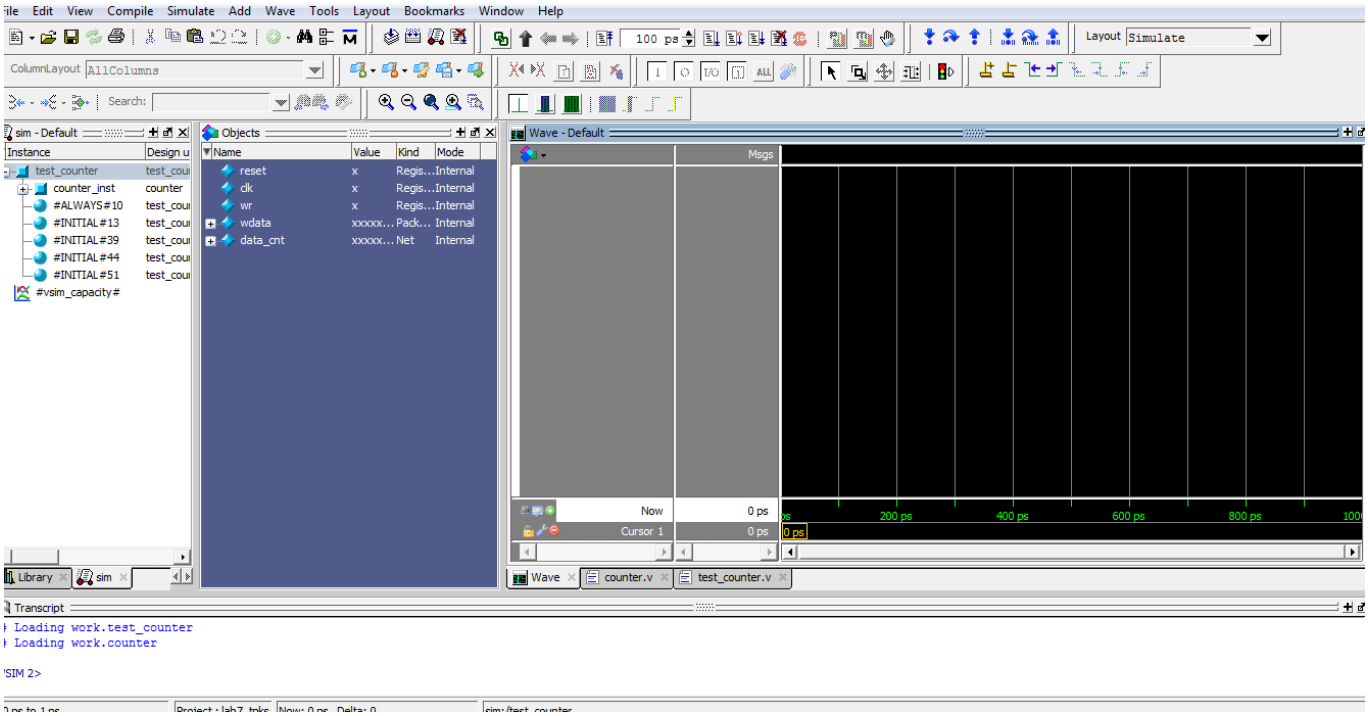
13. Вибираємо назву створеної бібліотеки (work) та натискаємо Ок.



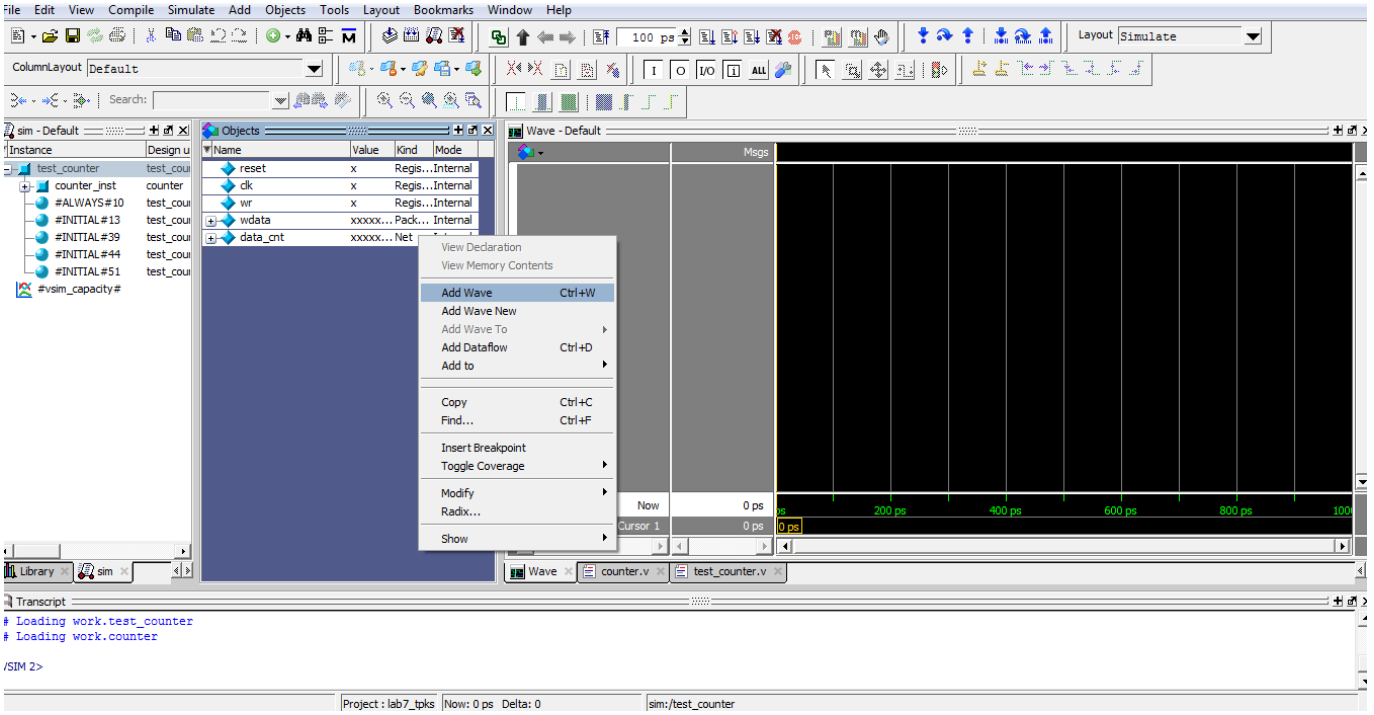
14. Бачимо відображення вхідних та вихідних сигналів у вкладці Objects.



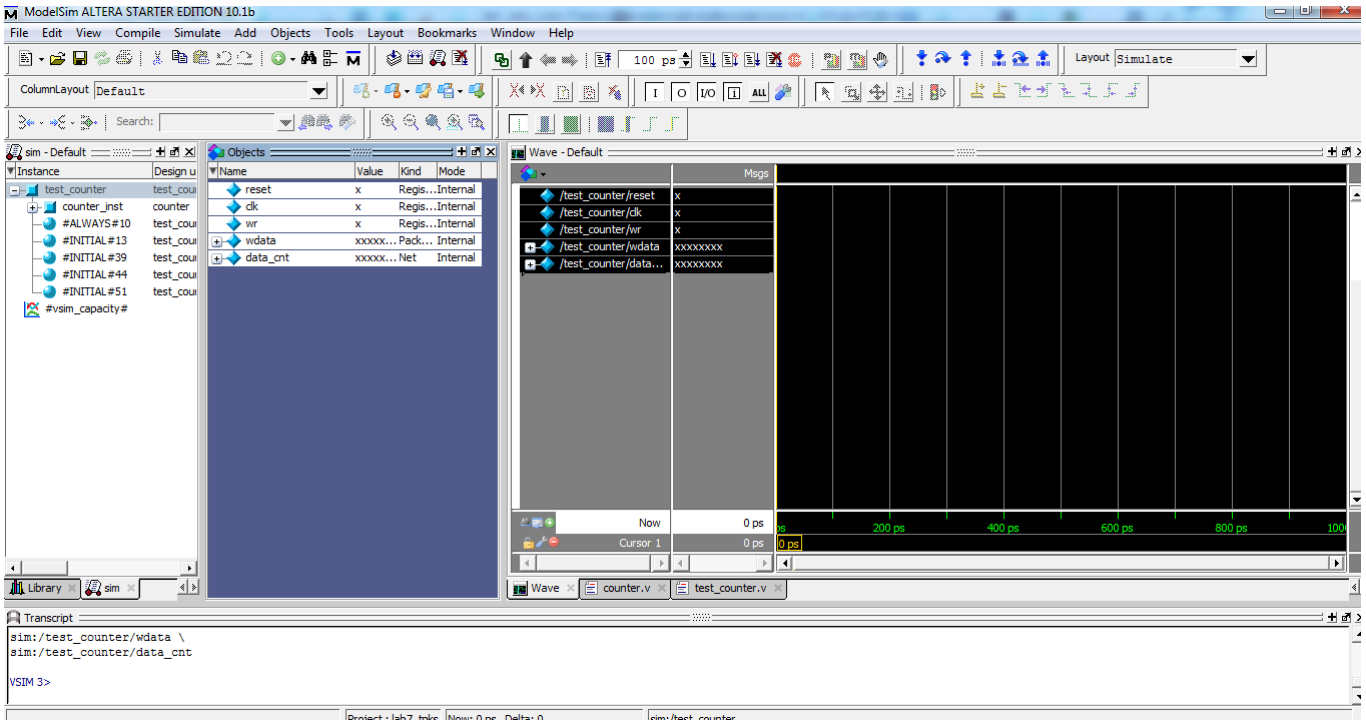
15. Переходимо на вкладку Wave.



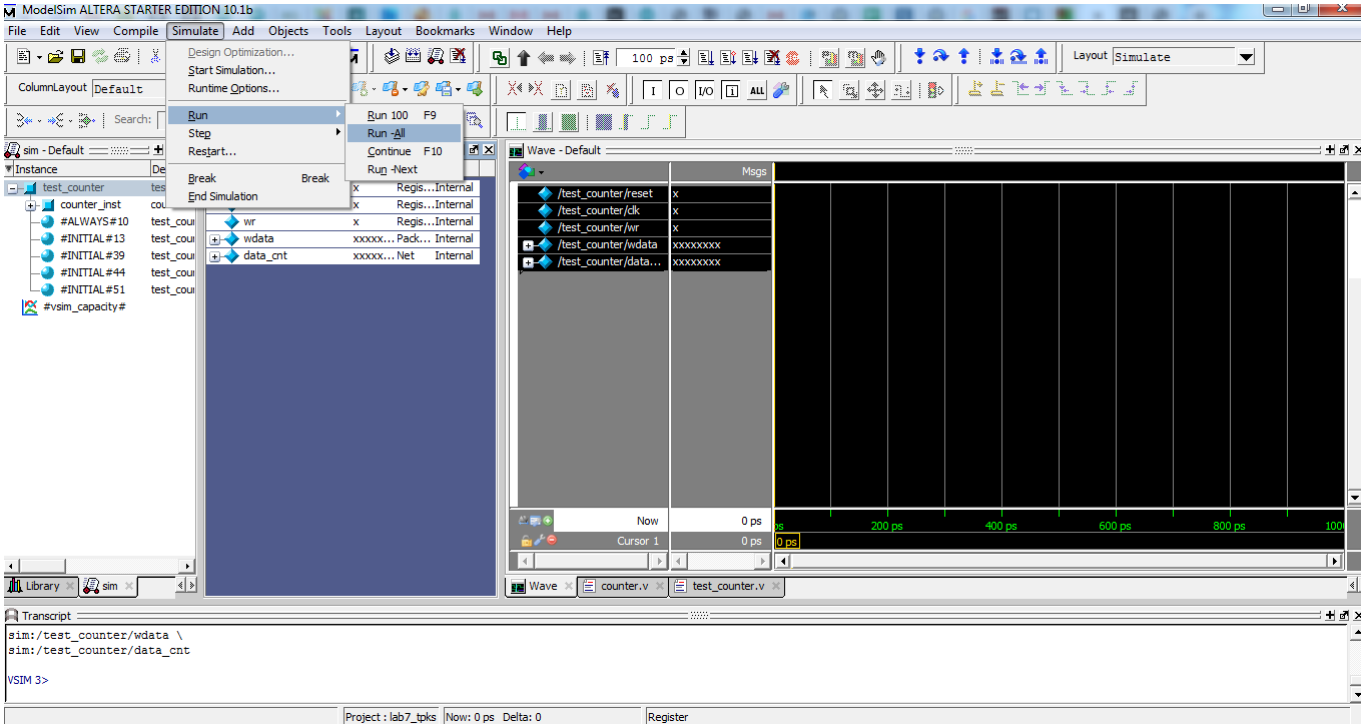
16. Додаємо сигнали на діаграму, виділивши всі сигнали у вкладці Objects та натиснувши правою кнопку миші вибираємо Add Wave.



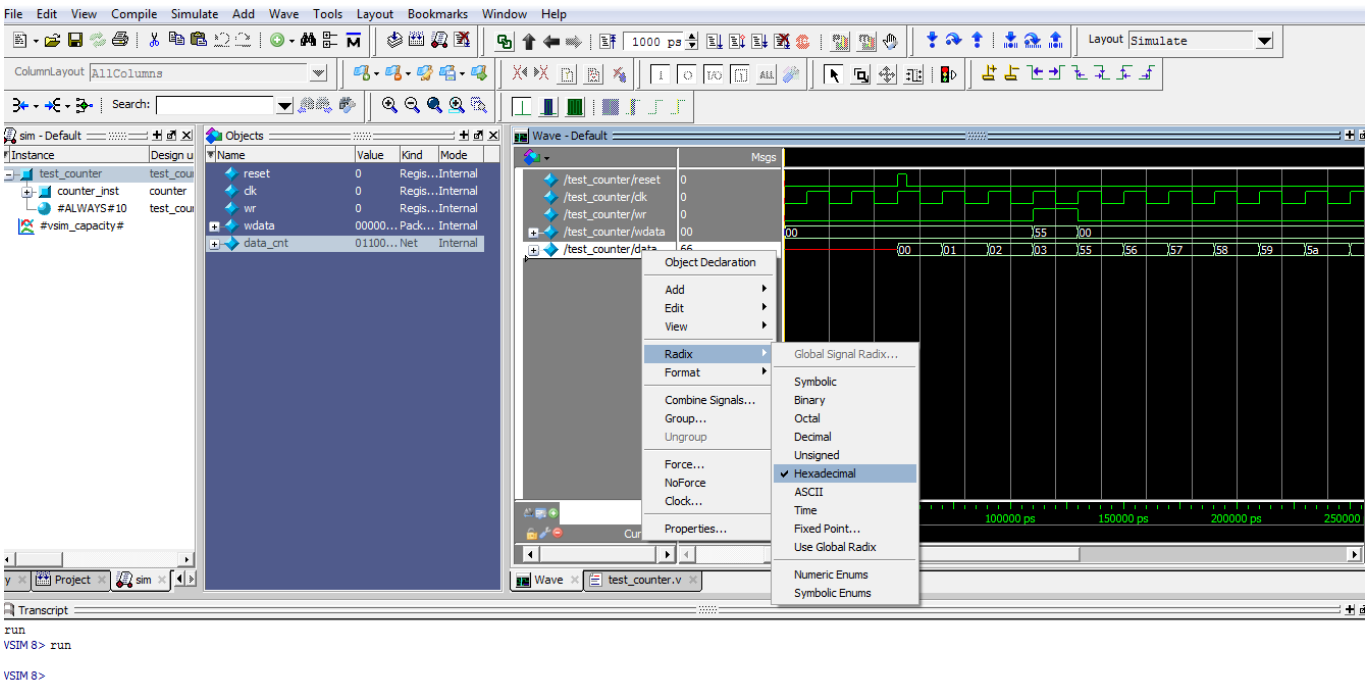
17. Сигнали додані на вкладку Wave.



18. Запускаємо процес симуляції роботи синхронного 8-розрядного двійкового лічильника задавши час 1000ps на часовій панелі середовища симуляції та вибраючи в основному меню Simulate -> Run -> Run 100 або натиснувши декілька разів клавішу F9.



19. Змінюємо відображення вхідної шини запису даних wdata та вихідної шини лічильника.



На кінцевій функціональній діаграмі роботи двійкового синхронного лічильника (рис. 2) бачимо формування сигналу тактової синхронізації clk, сигналу скиду rst, сигналу дозволу запису даних wr, вхідної шини wdata лічильника та вихідної шини лічильника data.

Після скиду сигналу reset в стан лог. ”0”, починається видача вихідних даних лічильника у порядку зростання починаючи з нуля. При встановленні сигналу дозволу запису лічильника wr в стан лог. “1” відбувається запис

вхідних даних із шини wdata (записується число 55). Після скиду сигналу дозволу запису лічильника wr в стан лог. “0”, на вихідній шині формуються зростаючі на одиницю послідовності двійкових кодів починаючи із числа 55.

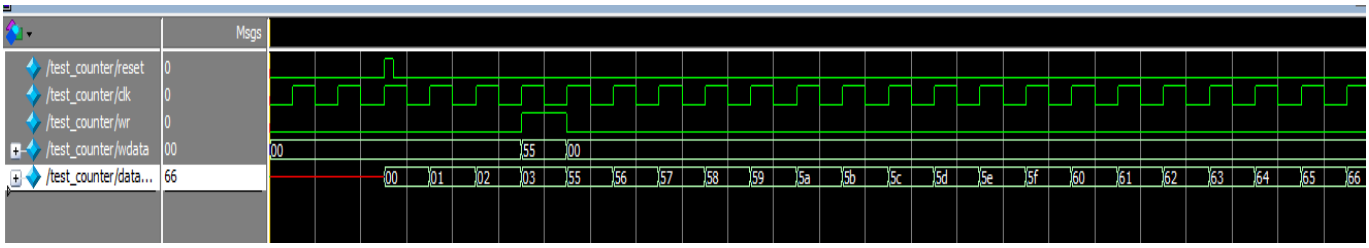


Рис. 2. Функціональна діаграма роботи 8-розрядного синхронного двійкового лічильника.

Завдання до лабораторної роботи №7

Реалізувати та дослідити синхронний двійковий лічильник на мові Verilog, створити тестовий файл для перевірки правильності його роботи (testbench) та виконати його функціональну симуляцію в ModelSim згідно заданого варіанту.

№	Назва пристрою	Тип підрахунку	Розрядність
1	Двійковий лічильник	up (по зростанню)	16
2	Двійковий лічильник	down (по спаданню)	20
3	Двійковий лічильник	up	24
4	Двійковий лічильник	down	28
5	Двійковий лічильник	up	32
6	Двійковий лічильник	down	16
7	Двійковий лічильник	up	20
8	Двійковий лічильник	down	24
9	Двійковий лічильник	up	28
10	Двійковий лічильник	down	32
11	Двійковий лічильник	up	16
12	Двійковий лічильник	down	20

Зміст звіту

1. Титульна сторінка.
2. Назва та мета лабораторної роботи.
3. Створити простий проект згідно заданого варіанту.
4. Навести програмний код основних файлів проекту.
5. Навести функціональну діаграму симуляції роботи реалізованого синхронного двійкового лічильника.
6. Висновки.

ЛАБОРАТОРНА РОБОТА №8

Назва роботи: технології автоматичної генерації, компіляції та моделювання програмних моделей електронних пристроїв.

Мета роботи: оволодіння засобами системи генерування програмних моделей обчислювальних пристроїв IP Core Generator в середовищі Active-HDL версії 9.1 по VHDL-опису, компіляції та функціональній симуляції згенерованого пристрою.

Теоретичні відомості

Система генерування програмних моделей обчислювальних пристроїв *IP Core Generator*, розроблена фірмою *ALDEC*, вбудована в середовище *Active-HDL*.

Система призначена для автоматичного генерування широкого діапазону програмних моделей цифрових елементів мовами *VHDL* та *Verilog* на основі їх конфігурованих моделей. Генеровані моделі є технологічно незалежними і можуть бути синтезовані засобами синтезу різних виробників.

Для запуску генератора необхідно вибрати опцію меню *Tools/IP Core Generator* (рис. 1).

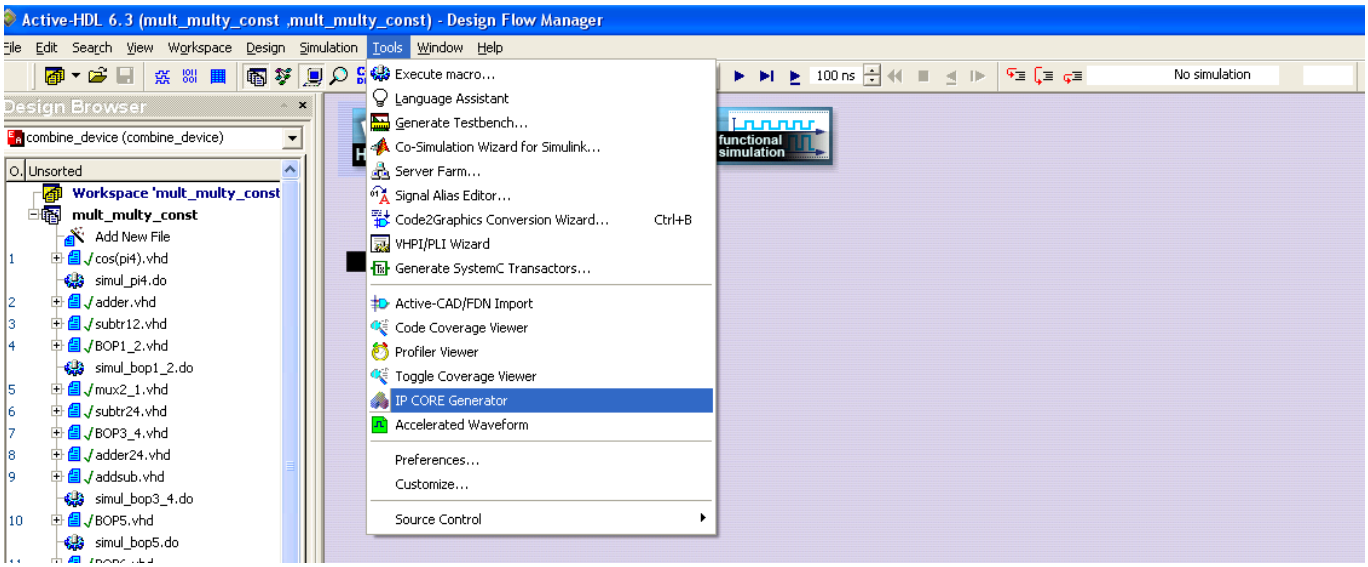


Рис. 1. Меню *Tools* з опцією запуску генератора в середовищі *Active-HDL*.

Початкове вікно системи *IP Core Generator* має наступний вигляд (рис. 2).

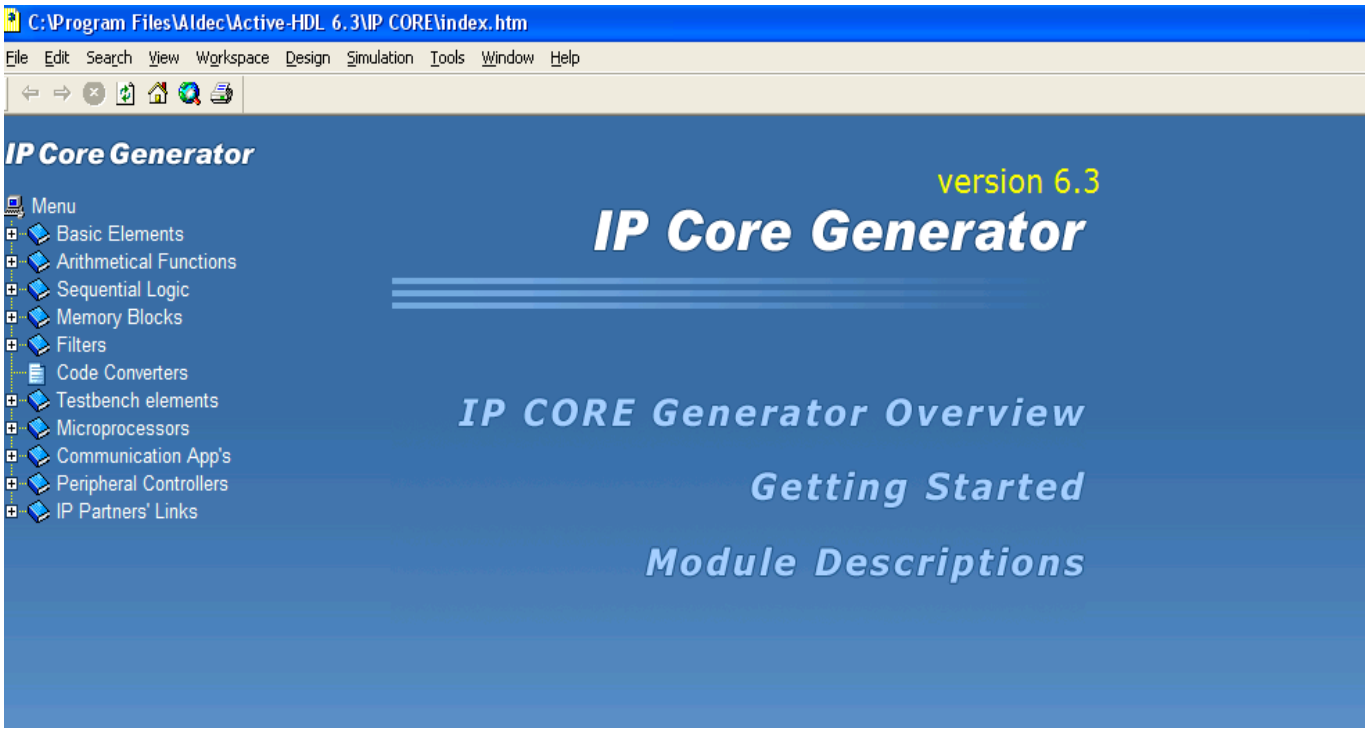


Рис. 2. Початкове вікно системи *IP Core Generator* в середовищі *Active-HDL*.

Дане вікно містить меню, яке складається з трьох секцій:

- Огляд системи (*IP Core Generator Overview*), в якій знаходиться загальна інформація про систему;
- "Приступаючи до роботи" (*Getting Started*), де описано основи роботи системи і показано, як генерувати програмні моделі;
- Опис модулів (*Modules Description*), в якій пояснюється поведінка, специфікація та діапазон зміни розмірів портів для кожного типу генерованих елементів.

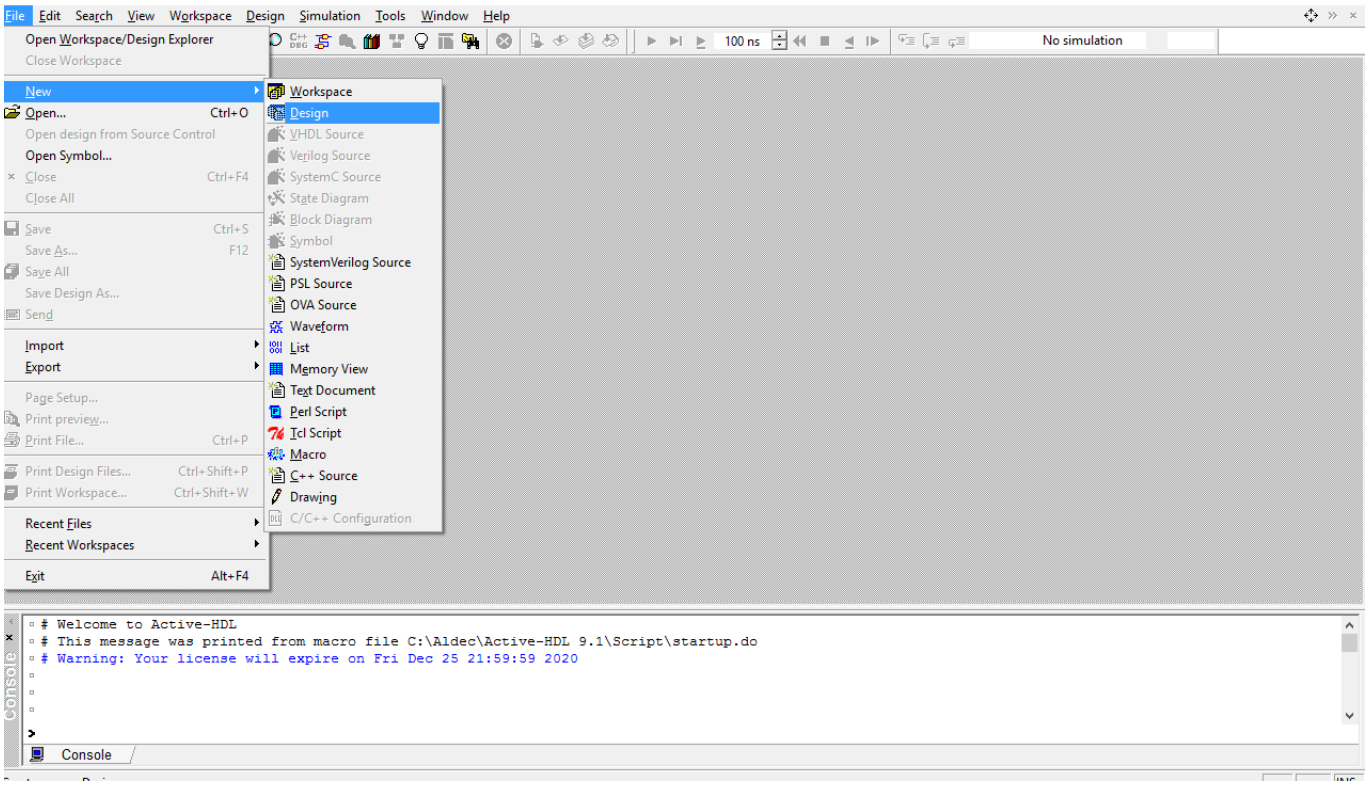
Деревоподібне меню навігації, розміщене в лівій частині вікна, надає доступ до елементів, програмні моделі яких можна згенерувати, і поділене на вісім тематичних груп а саме:

- Базові елементи: мультиплексори, демультиплексори, тригери, вентилі, вхідні та вихідні буфери, генератори констант і деякі інші;
- Модулі виконання арифметичних функцій: арифметико-логічні пристрої, суматори/віднімачі, компаратори, акумулятори, перемножувачі, подільники;

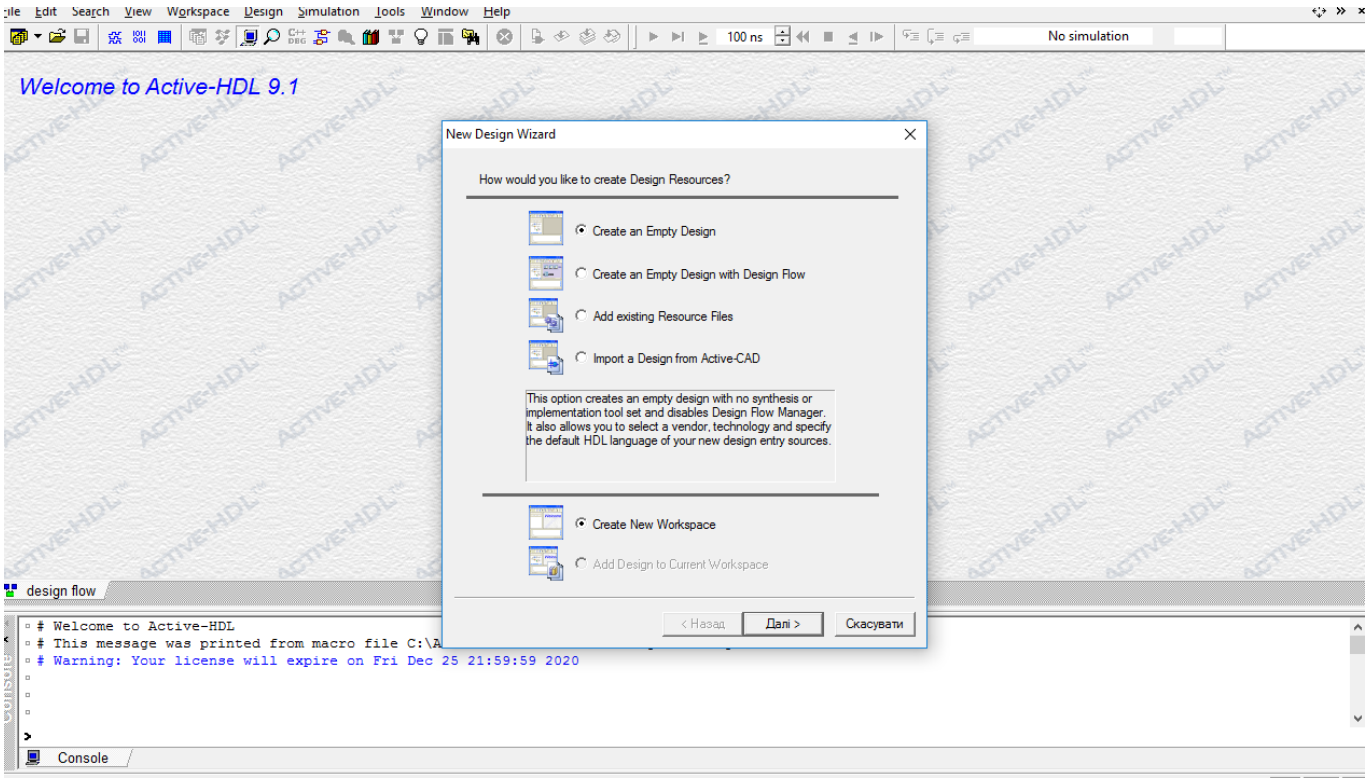
- Елементи послідовної логіки: лічильники, регістри зсуву, синхронні і асинхронні тригери;
- Пам'ять: асоціативна, з довільним доступом (оперативна, постійна), з послідовним доступом (*FIFO/LIFO*);
- Фільтри: послідовні та паралельні КІХ фільтри, сортувальні фільтри;
- Перетворювачі кодів: HOT -> BIN, HOT -> GRAY, HOT -> JOHNSON,
- Елементи систем тестування: осцилятори, генератори аналогових сигналів (синусоїдальні, прямокутні, трикутні та ін.), підпрограми роботи з текстовими файлами;
- Мікроконтролери *Microchip PIC16C5x*, *Intel 8051* та деякі інші. Необхідно зазначити, що програмні моделі цих мікроконтролерів доступні на замовлення за окрему плату, натомість генератор надає можливість переглянути рекламні листи про ці мікроконтролери;
- Контролери інтерфейсів;
- Програмні моделі спеціалізованих процесорів: прямого та інверсного дискретного косинусного перетворення, шифрування за алгоритмом *DES*, автентифікації за алгоритмом *SHA-1*, декодер Вітербі, контролери інтерфейсів *USB* та *USB2*. Всі зазначені програмні моделі доступні на замовлення, однак надається інформація про них у вигляді рекламних листів.

Хід виконання лабораторної роботи

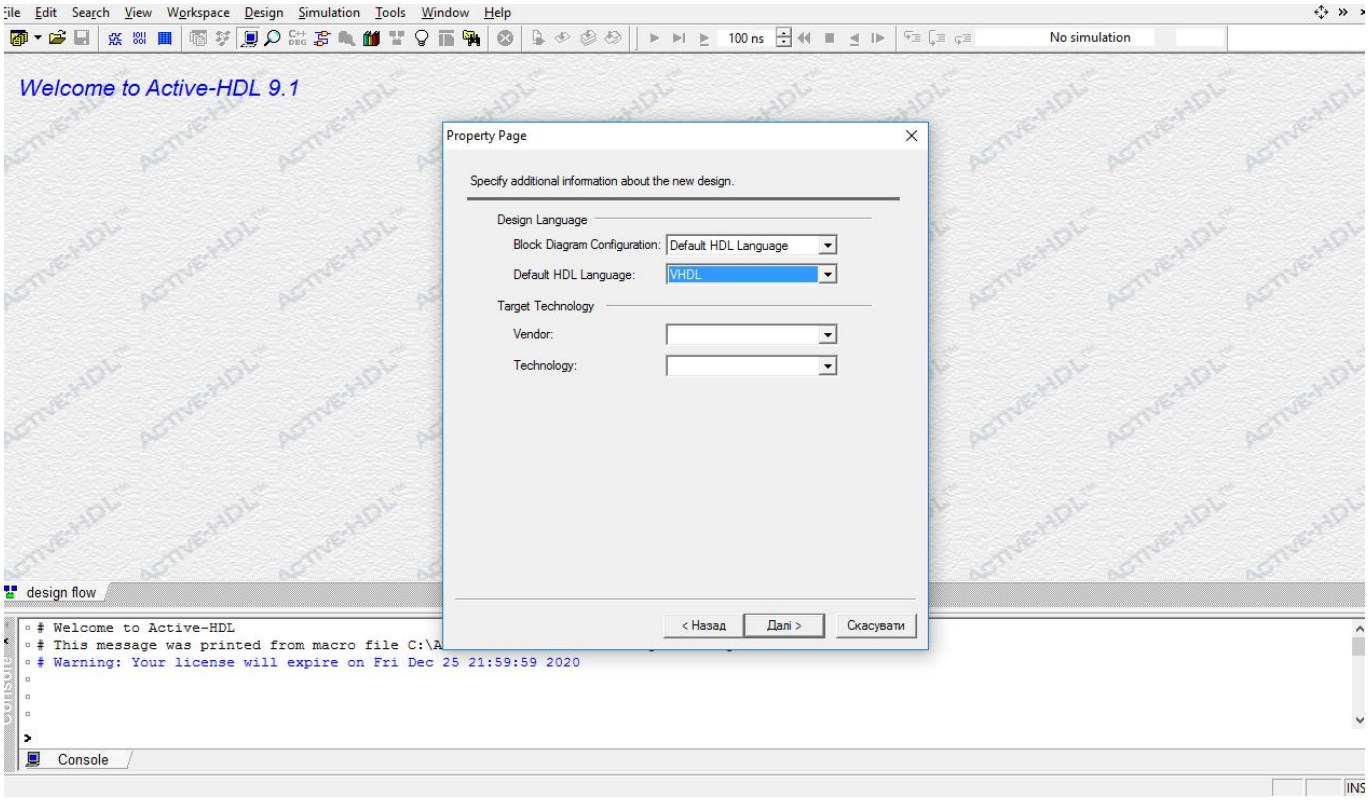
1. Запустити інтегроване середовище розробки Active-HDL 9.1.
2. Створити новий проект натиснувши File->New->Design.



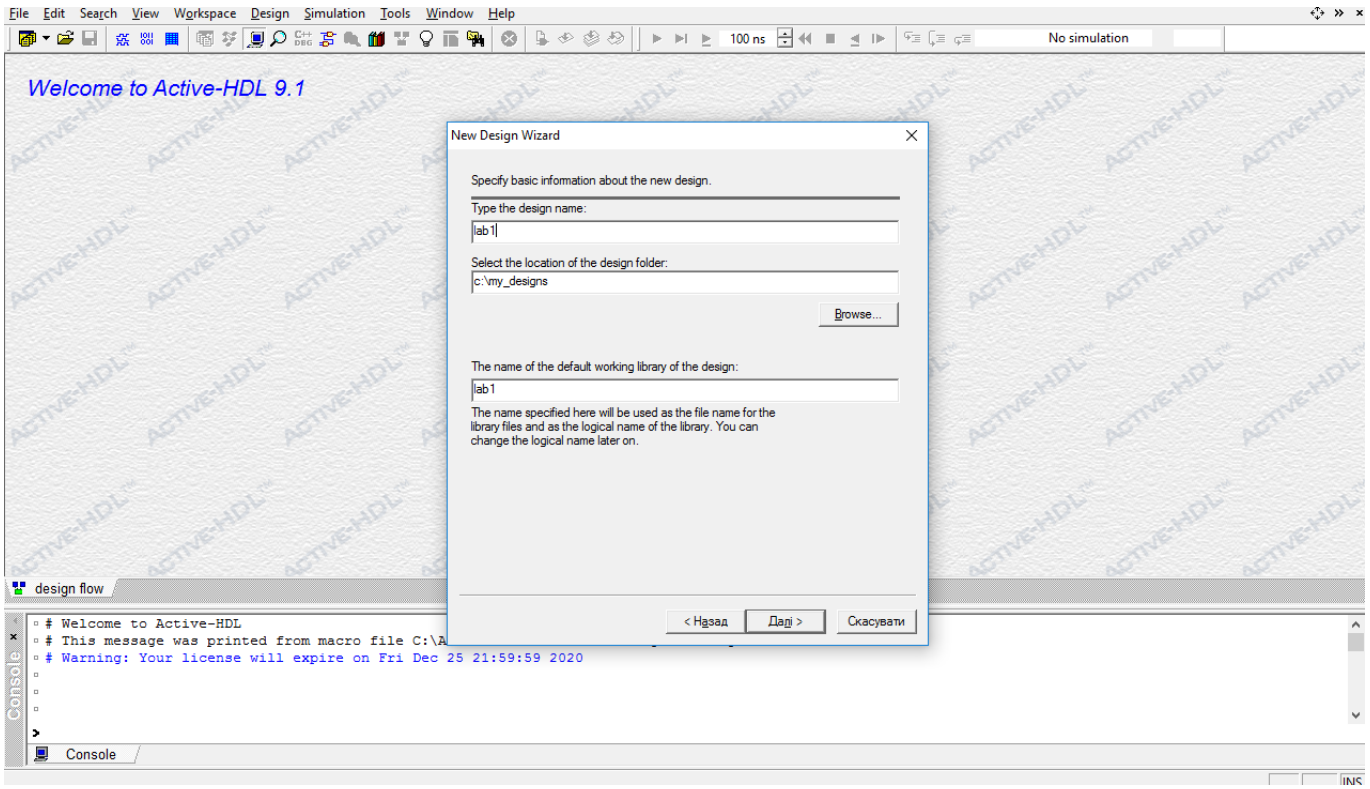
3. У вікні New Design Wizard зробити активною опцію Create Empty Design та натиснути Далі.



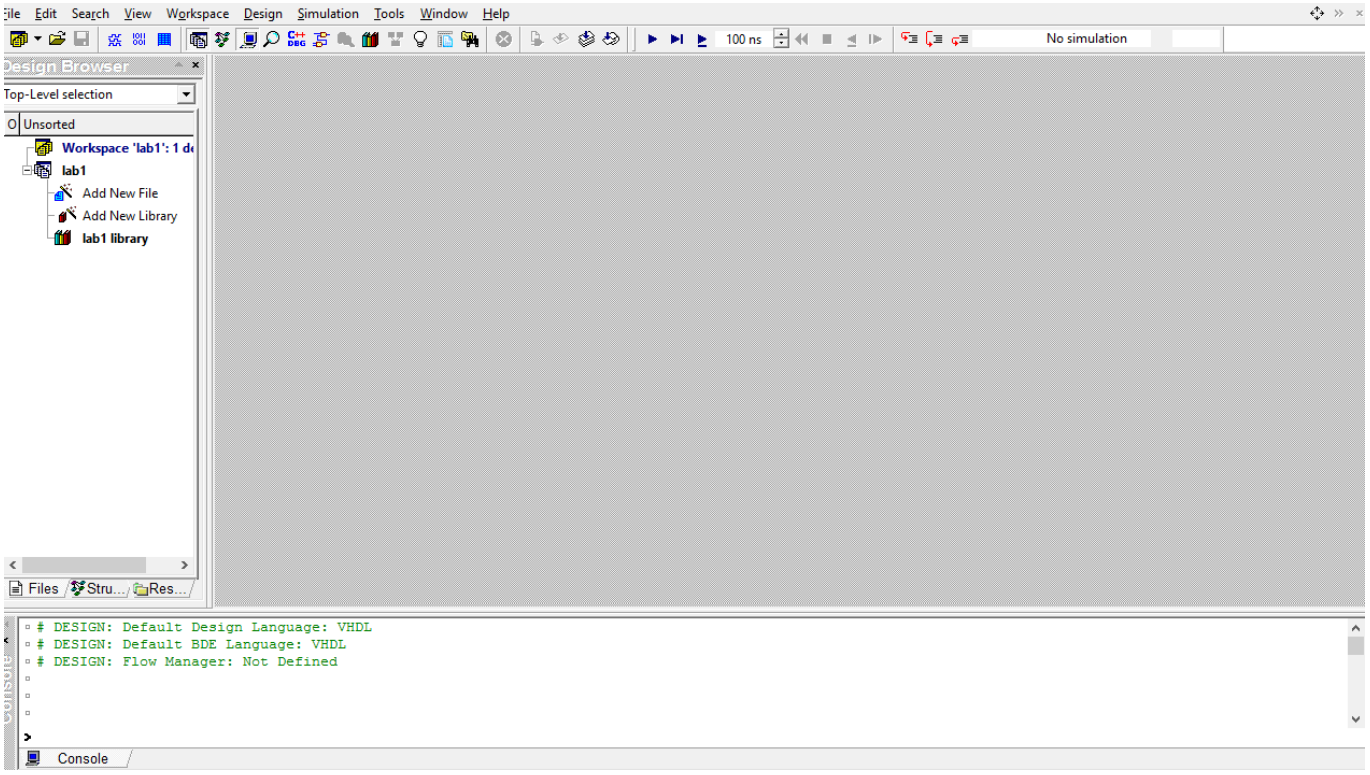
4. У вікні Property Page потрібно, щоб опція вибору мови HDL по замовчуванню (Default HDL Language) була вибрана мова VHDL і тиснемо кнопку Далі.



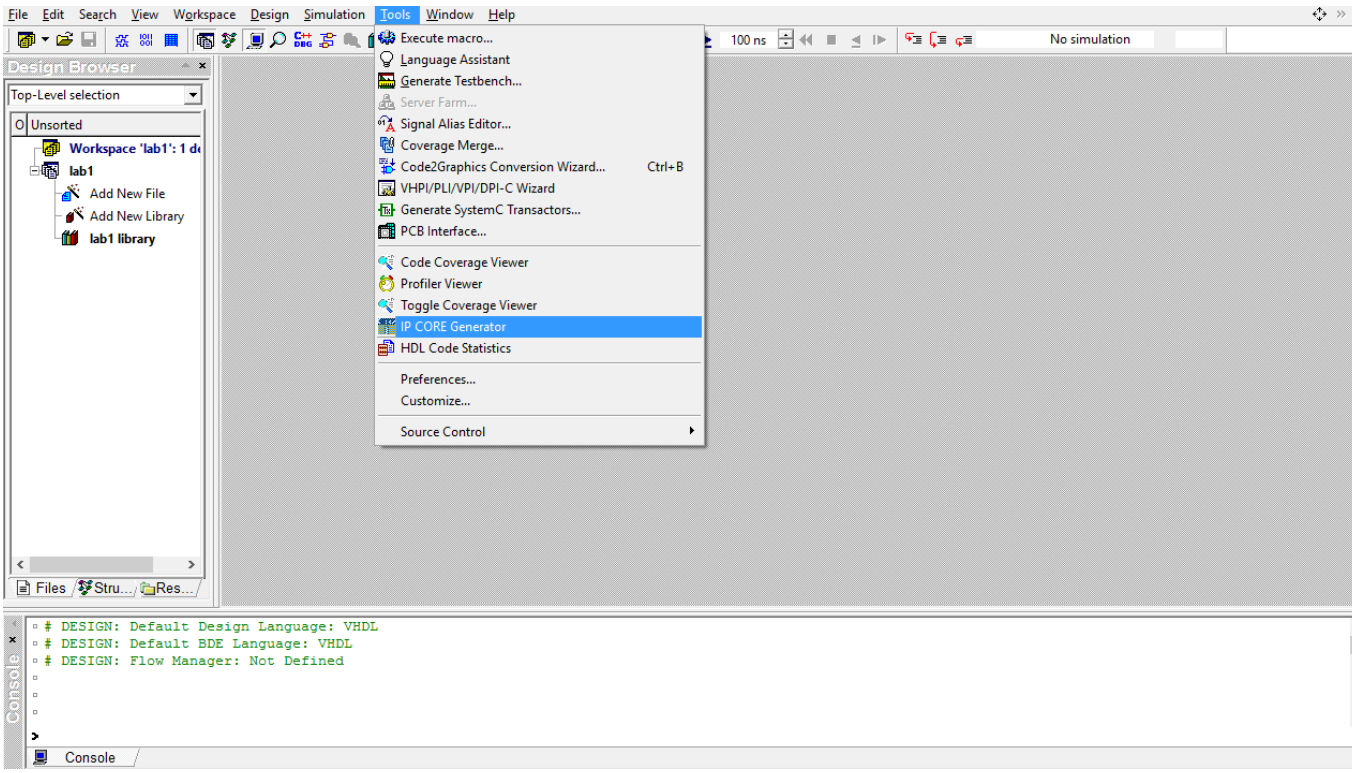
5. У вікні New Design Wizard вводимо назву проекту і тиснемо кнопку Далі.



6. Натиснувши кнопку Готово зліва появляється панель Design Browser створеного проекту (lab8).

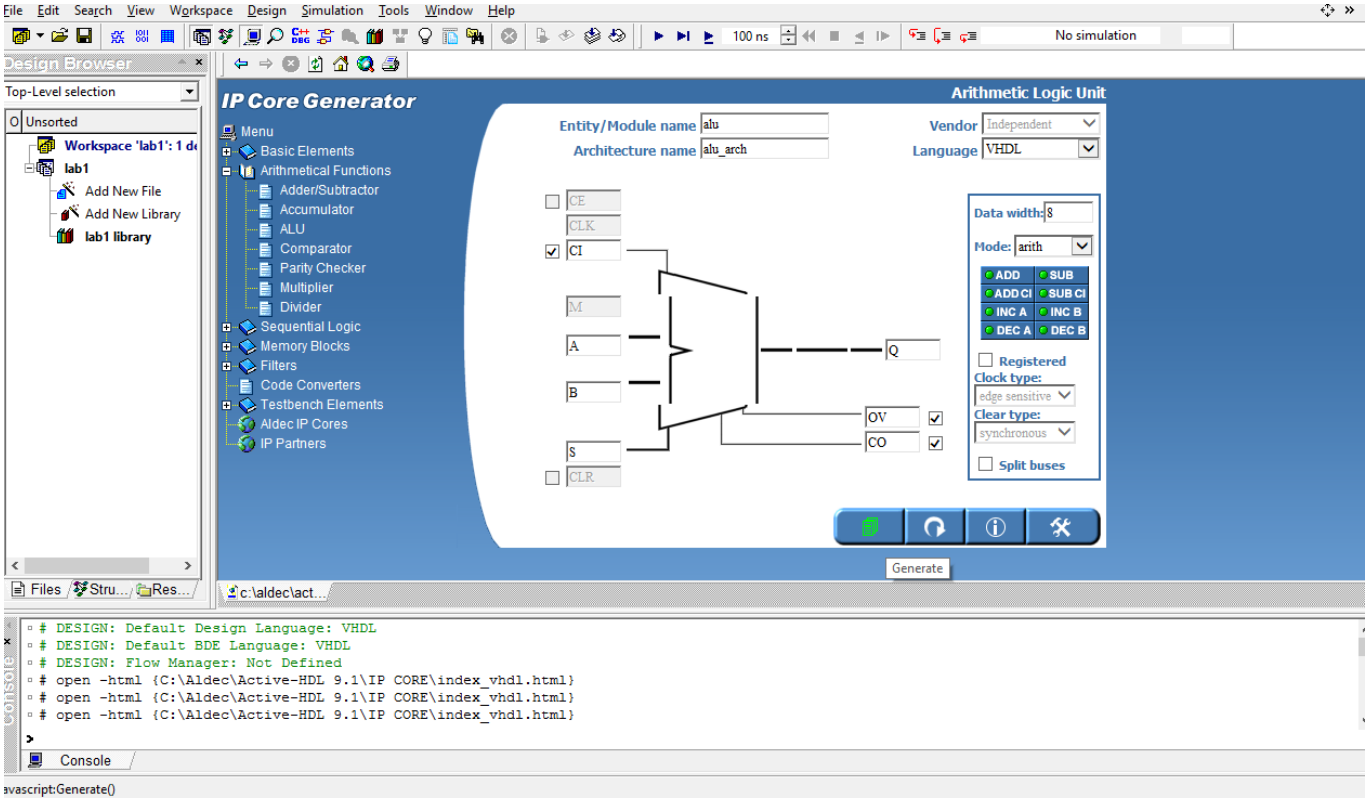


7. Наступним кроком заходимо в меню Tools і запускаємо опцію IP CORE Generator.

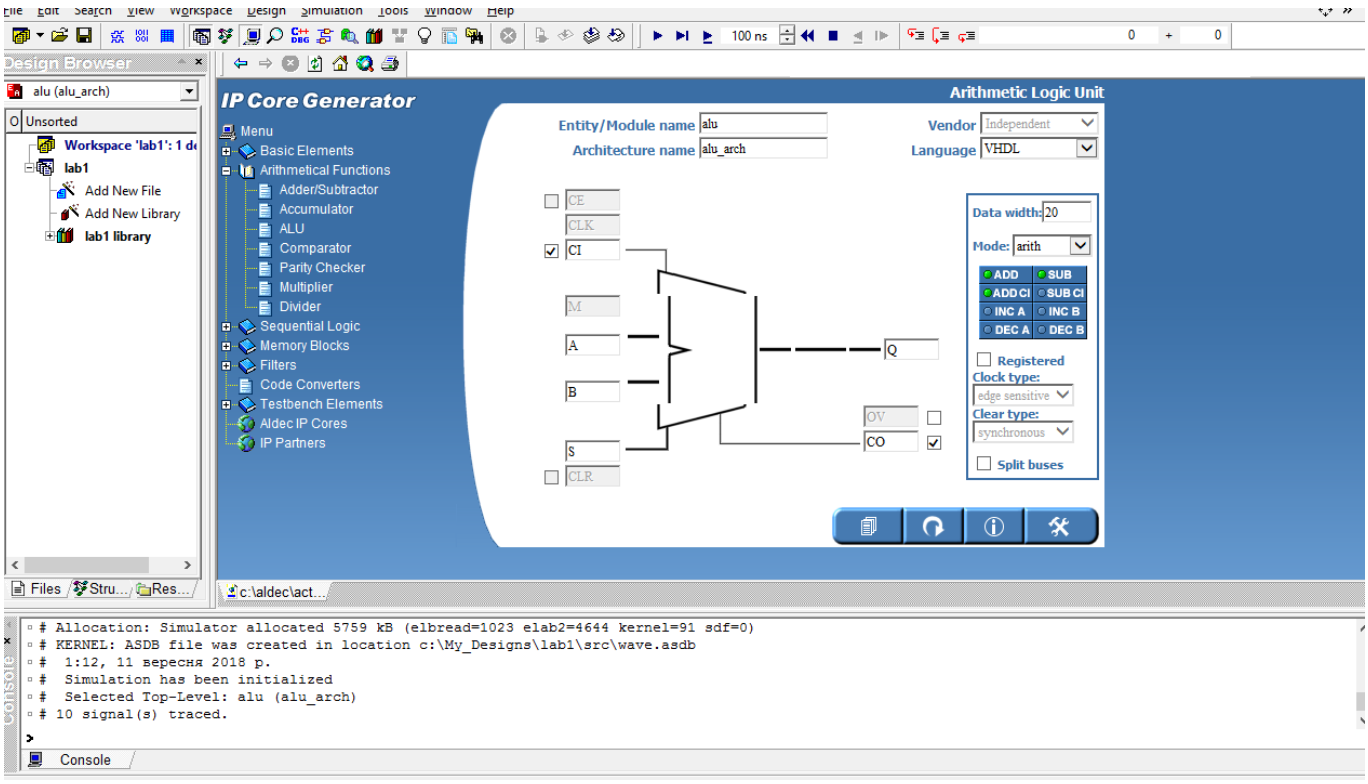


8. Коли запустився IP CORE Generator в меню вибираємо Arithmetical Functions

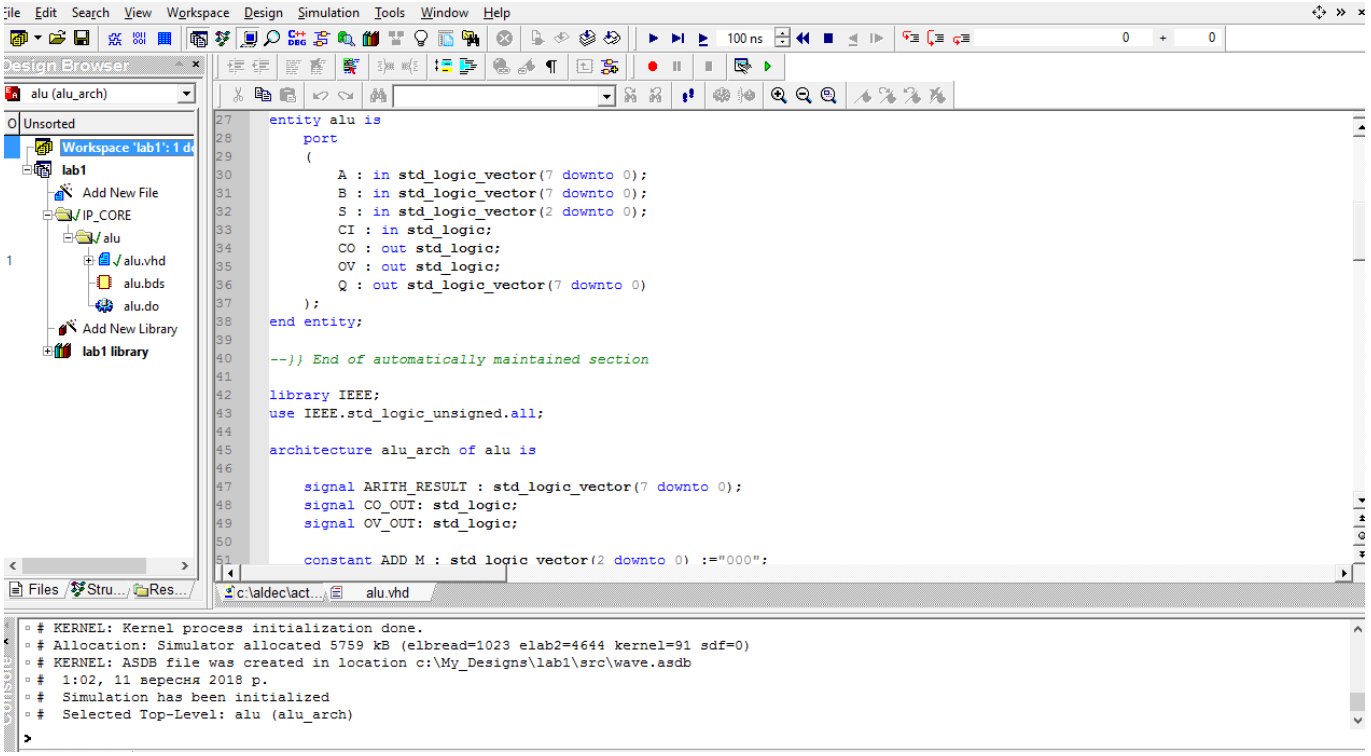
   -> ALU (Арифметико-логічний пристрій).



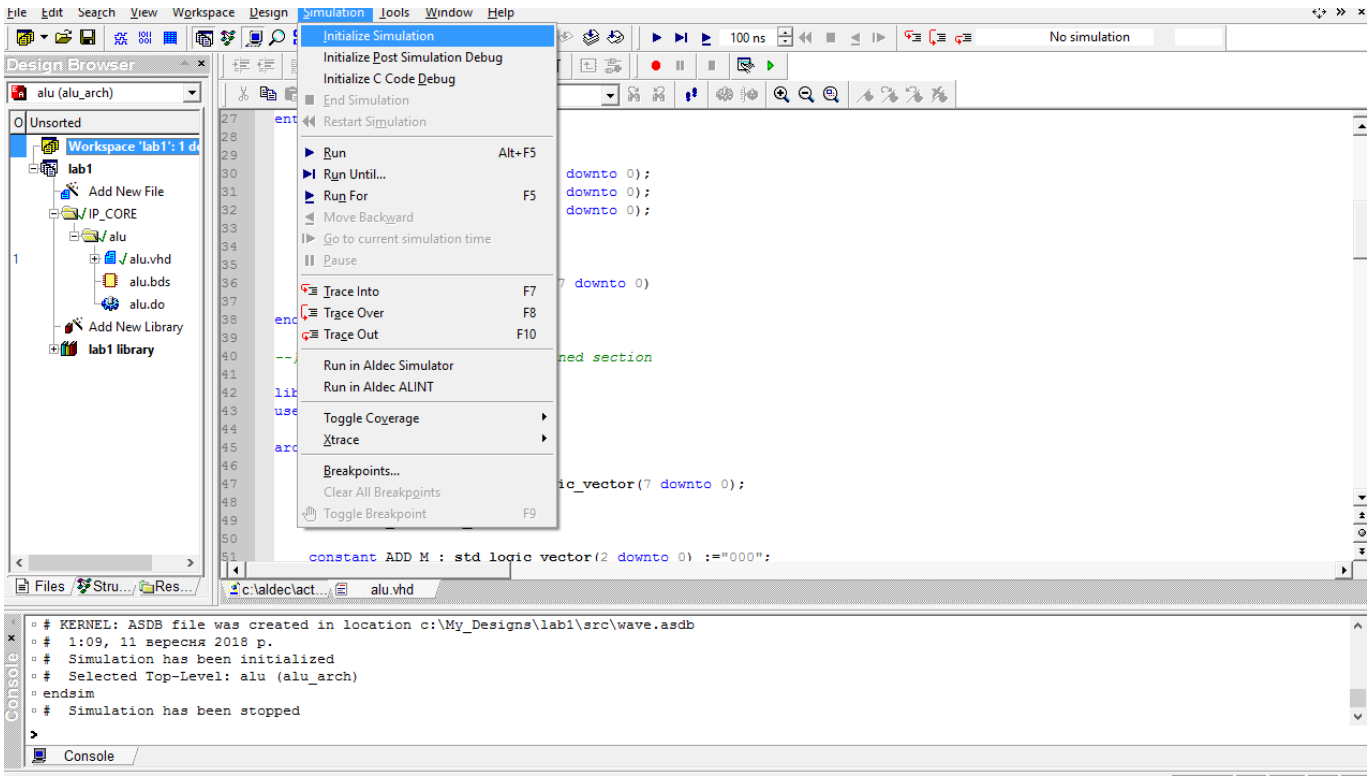
9. Коли запустився IP CORE Generator в меню ALU (розглядається варіант №26) знімаємо галочку з Registered Clock type, виставляємо розрядність даних (Data width = 20) і вибираємо команди згідно варіанту (add, add_ci, sub) і знімаємо галочку з вихідного сигналу OV і тиснемо знизу крайню ліву кнопку Generate.



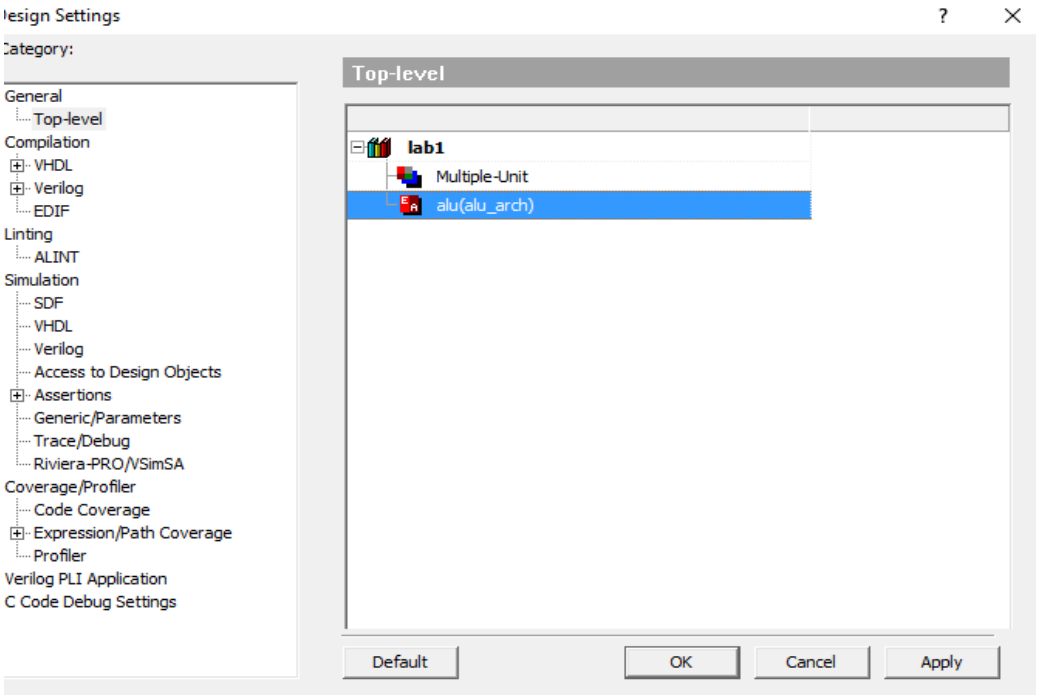
10. У вікні Design Browser створюється папка IP CORE а в даній папці вкладена папка alu, яка містить файл alu.vhd з описом АЛП на мові VHDL. Натиснувши клавішу F11 можна скомпілювати дану програму.



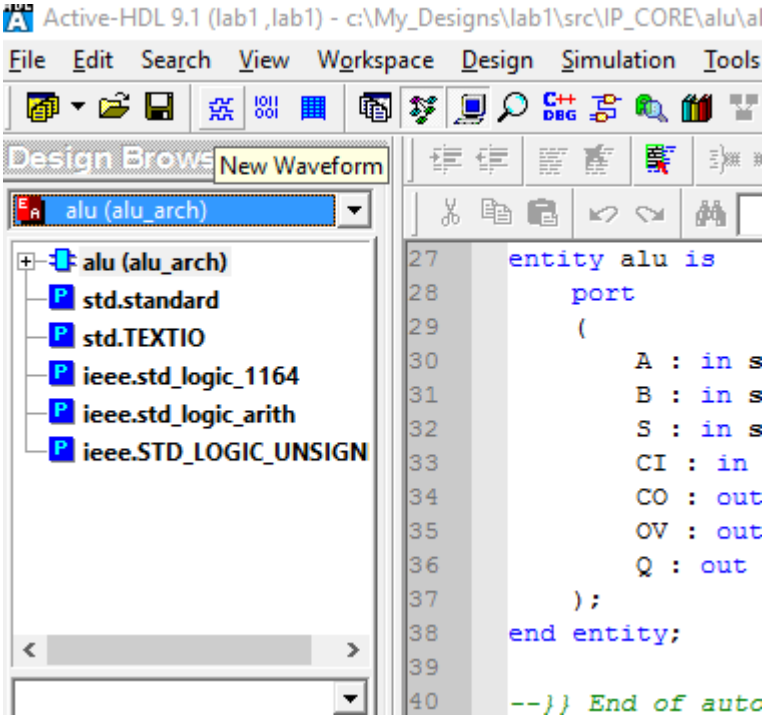
11. Далі переходимо до функціональної симуляції пристрою у меню середовища натискаємо опції Simulation -> Initialize Simulation.



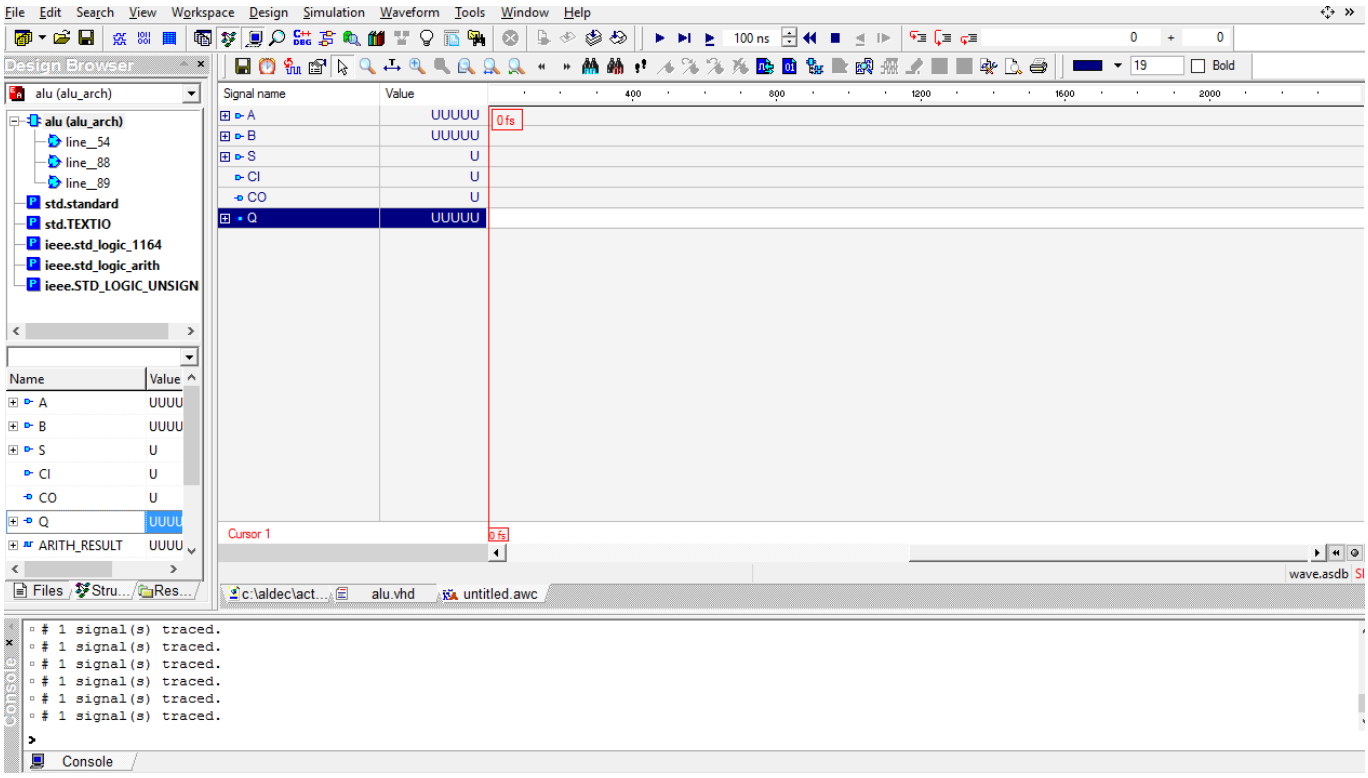
12. При появі вікна Design Settings вибираємо в Top-level -> alu(alu_arch) і тиснемо кнопку ОК.



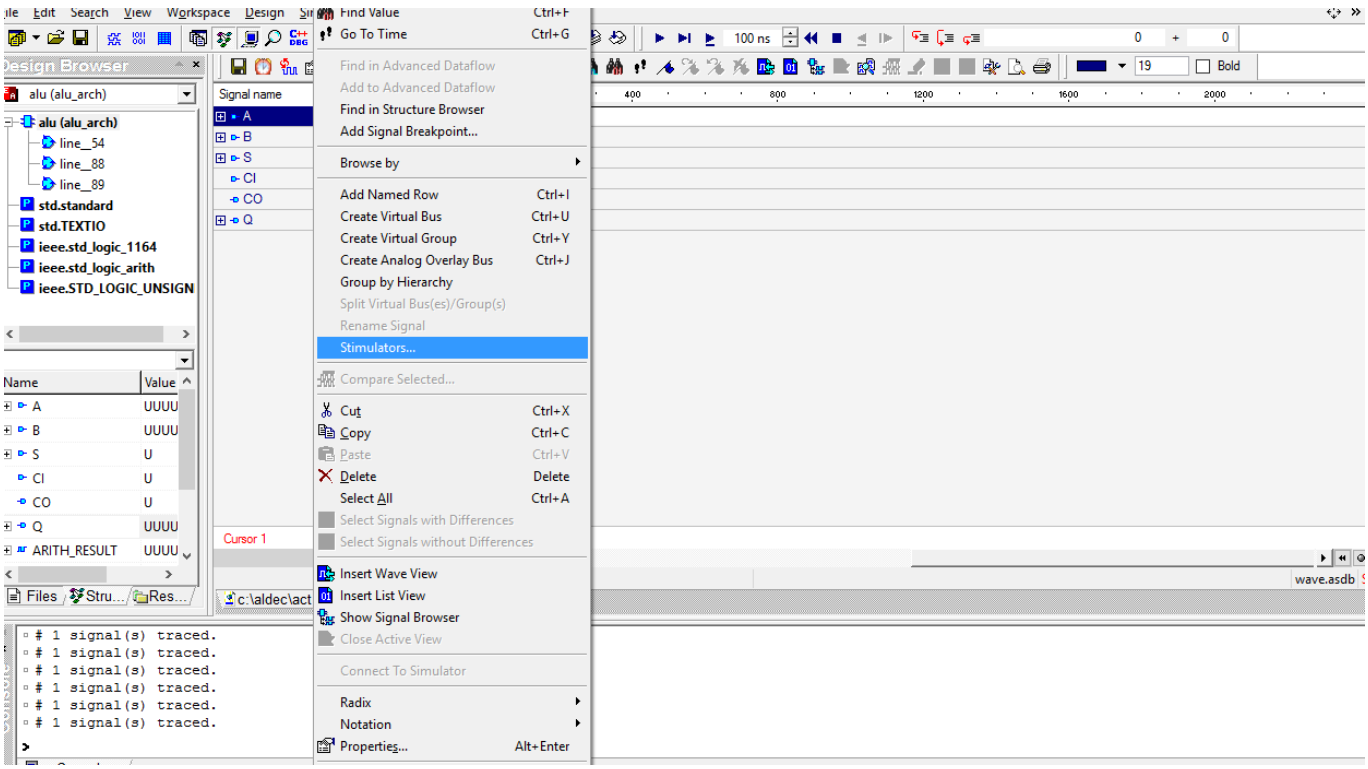
13. Далі на панелі інструментів інтегрованого середовища натискаємо кнопку New Waveform.



14. Далі перетягуємо за допомогою миші сигнали, які відображаються внизу вікна Design Browser на діаграму.



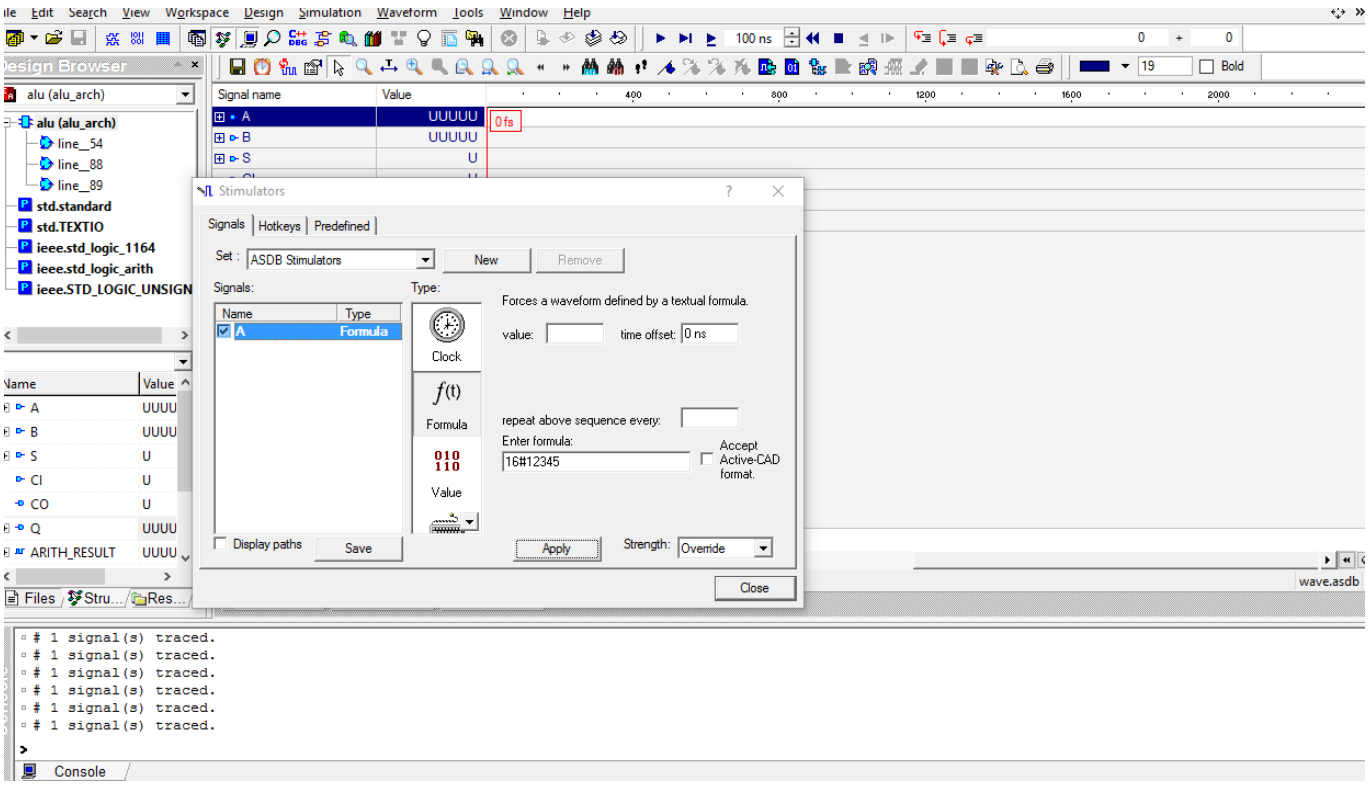
15. Функціональну симуляцію можна проводити двома способами (ручним і автоматичним). Для ручної симуляції потрібно зробити активним той сигнал на який ми хочемо подати дані і натиснувши праву кнопку миші вибрати Stimulators.



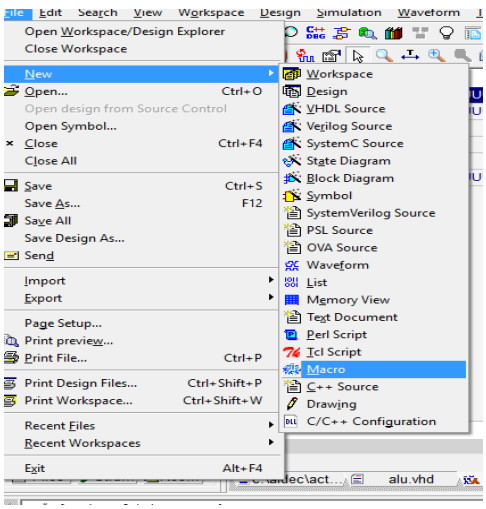
16. Відкривши вікно Stimulators обираємо клавішу Formula та в полі Enter Formula вводимо вхідне значення для подачі на шину A у

шістнадцятковій системі числення 16# вводимо 20-розрядні дані у 16-вому вигляді 12345 де кожна цифра представляється чотирма розрядами (0001 0010 0011 0100 0101).

Для подачі даних на однобітові сигнали використовуємо кнопку Value. Коли всі вхідні значення подані для запуску симуляції натискаємо клавішу F5.



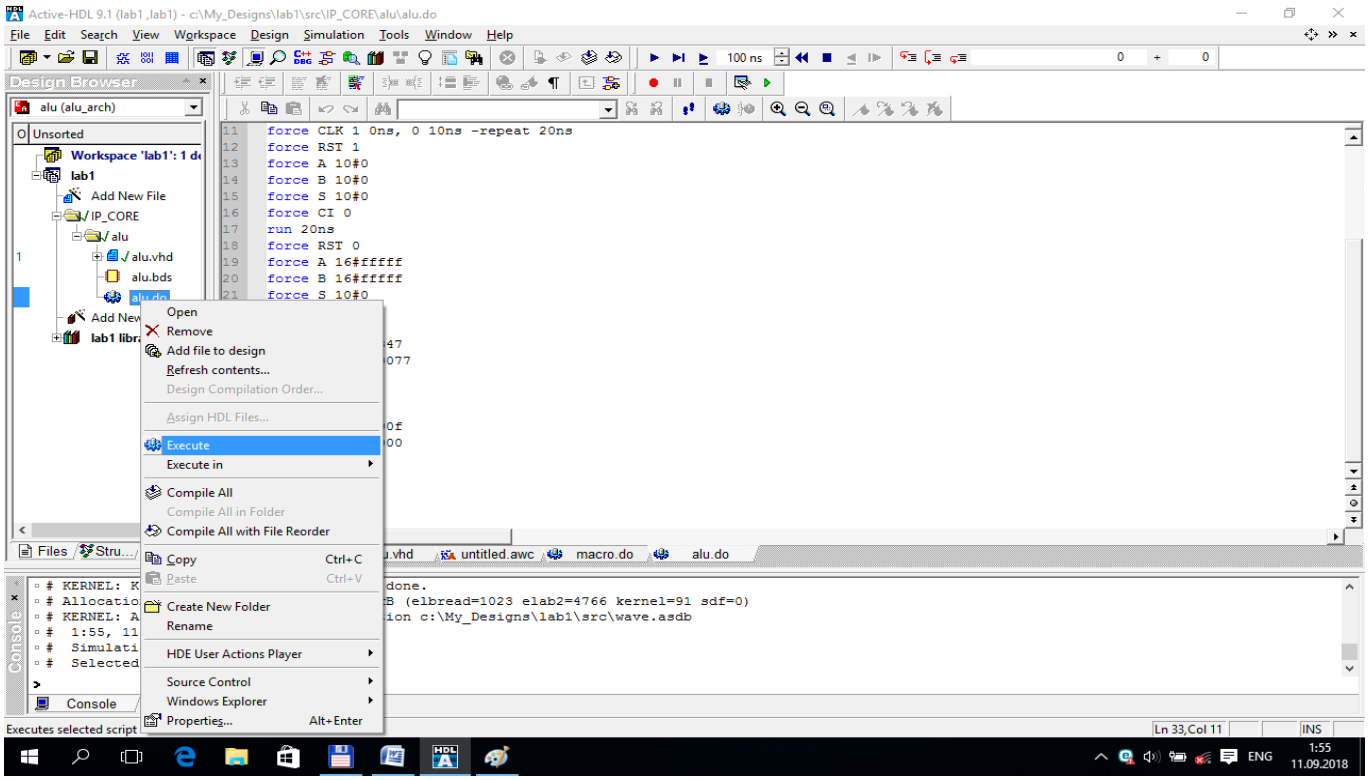
17. Для автоматичної симуляції потрібно створити відповідний виконавчий макрос (скрипт-файл), який має розширення *.do. Потрібно натиснути File -> New -> Macro або використати готовий в папці alu файл alu.do.



18. У даний макро-файл alu.do потрібно вставити відповідний код і зберегти його.

```
restart
wave
wave CLK
wave RST
wave A
wave B
wave S
wave CI
wave CO
wave Q
force CLK 1 0ns, 0 10ns -repeat 20ns
force A 16#0000f
force A 10#0
force B 10#0
force S 10#0
force CI 0
run 20ns
force RST 0
force A 16#fffff
force B 16#fffff
force S 10#0
force CI 1
run 20ns
force A 16#12347
force  B 16#00077
force  S 10#1
force  CI 1
run 20ns
force A 16#0000f
force B 16#00000
force S 10#2
force  CI 1
run   140ns
```

19. Далі ініціалізувавши симуляцію (Simulation -> Initialize Simulation) клікнувши на файлі alu.do правою кнопки миші потрібно запустити процес симуляції натиснувши опцію Execute. В результаті повинна згенеруватися функціональна діаграма роботи пристрою.



Результат функціональної симуляції пристрою.

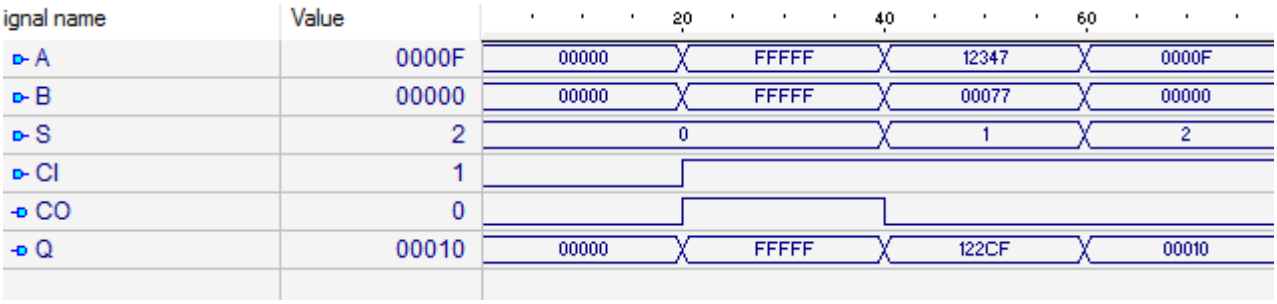


Рис. 1. Функціональна діаграма роботи 20-ти розрядного АЛП.

Зміст звіту

1. Титульна сторінка.
2. Назва та мета лабораторної роботи.
3. Завдання до лабораторної роботи.
4. Навести VHDL-код згенерованого пристрою згідно варіанту завдання.
5. Навести текст командного файлу, який запускає функціональну симуляцію згенерованого пристрою.
6. Навести функціональну діаграму (результат симуляції) роботи завершеного пристрою.
7. Висновки.

Завдання до лабораторної роботи №8

	Назва пристрою	Розрядність вхідних даних
1	Comparator(unsigned, <,=,>)	20
2	Parity checker	20
3	Adder (+)	20
4	Subtractor (-)	20
5	Adder, CI-з вхідним переносом (+)	20
6	Subtractor, CI-з вхідним переносом (-)	20
7	Adder/Subtractor (+/-)	20
8	Adder/Subtractor ,CI-з вхідним переносом (+/-)	20
9	Accumulator (+)	20
10	Accumulator (-)	20
11	Accumulator (+/-)	20
12	ALU (ADD_CI, SUB, SUB_CI)	20

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з дисципліни
“Комп’ютерні технології в наукових дослідженнях”
для студентів спеціальності “Електроніка”

Укладач: Грига В.М., к. т. н., доцент.

Редактор Грига В.М.
Технічне редагування і коректура Данилюк О.М.

Підписано до друку __.__.20__ р. Формат 60×84/8
Папір офсетний. Гарнітура Times New Roman
Умов. друк. арк. 2,7. Тираж ___. Зам. ___

…….....

76025, м. Івано-Франківськ, вул. Шевченка, 57
Прикарпатський національний університет імені Василя Стефаника
Розтиражовано ВДВ ЦІТ