Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут»

Кафедра електронно обчислювальної апаратури

Лабораторна робота №1

з курсу «Апаратні прискорювачі на мікросхемах програмованої логіки»

Виконав:

Студент 3 курсу

Групи ДК-02

Садко Вячеслав

Варіант 10

Київ 2022

ЗМІСТ

[1. Реалізація в Simulink підсистеми, що розраховує функцію: 3](#_Toc122155998)

[2. Перегляд в логічному аналазаторі даних на входах і на виході створеної підсистеми у знаковому десятковому поданні 8](#_Toc122155999)

[3. Генерування коду на Verilog та синтез згенерованого коду в Quartus для створеної підсистеми 9](#_Toc122156000)

[4.Результат синтезу в RTL Viewer у квартусі та визначення апаратних витрат: 16](#_Toc122156001)

[5. Створення тестбенч файлу в Matlab для створеної підсистеми і результат симуляції тестбенча в Modelsim 18](#_Toc122156002)

[Висновок 20](#_Toc122156003)

Завдання

# 1. Реалізація в Simulink підсистеми, що розраховує функцію:

Y = W0\*X0 + W1\*X1 + W2\*X2 + W3\*X3

Типи даних входів: int8

Тип даних виходу: int16

На входах і виході поставити регістри (блок затримки на 1 такт)

Налаштування блоків “Uniform Random Number” згідно номеру варіанту:

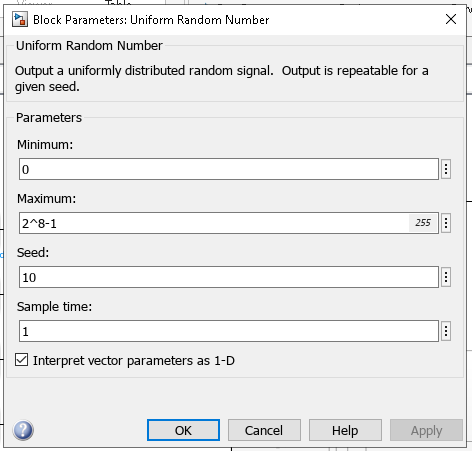


Рис. 1 Налаштування першого по порядку блоку

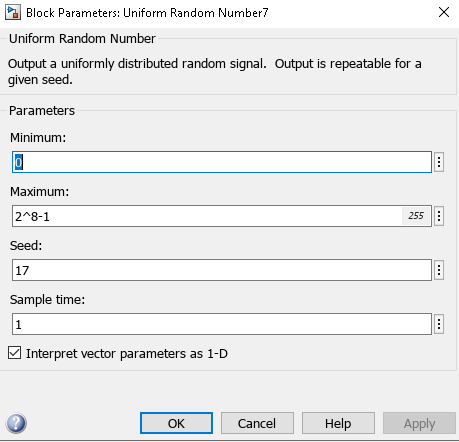


Рис. 2 Налаштування останнього по порядку блоку

Створюємо в Simulink загальну схему згідно умові:

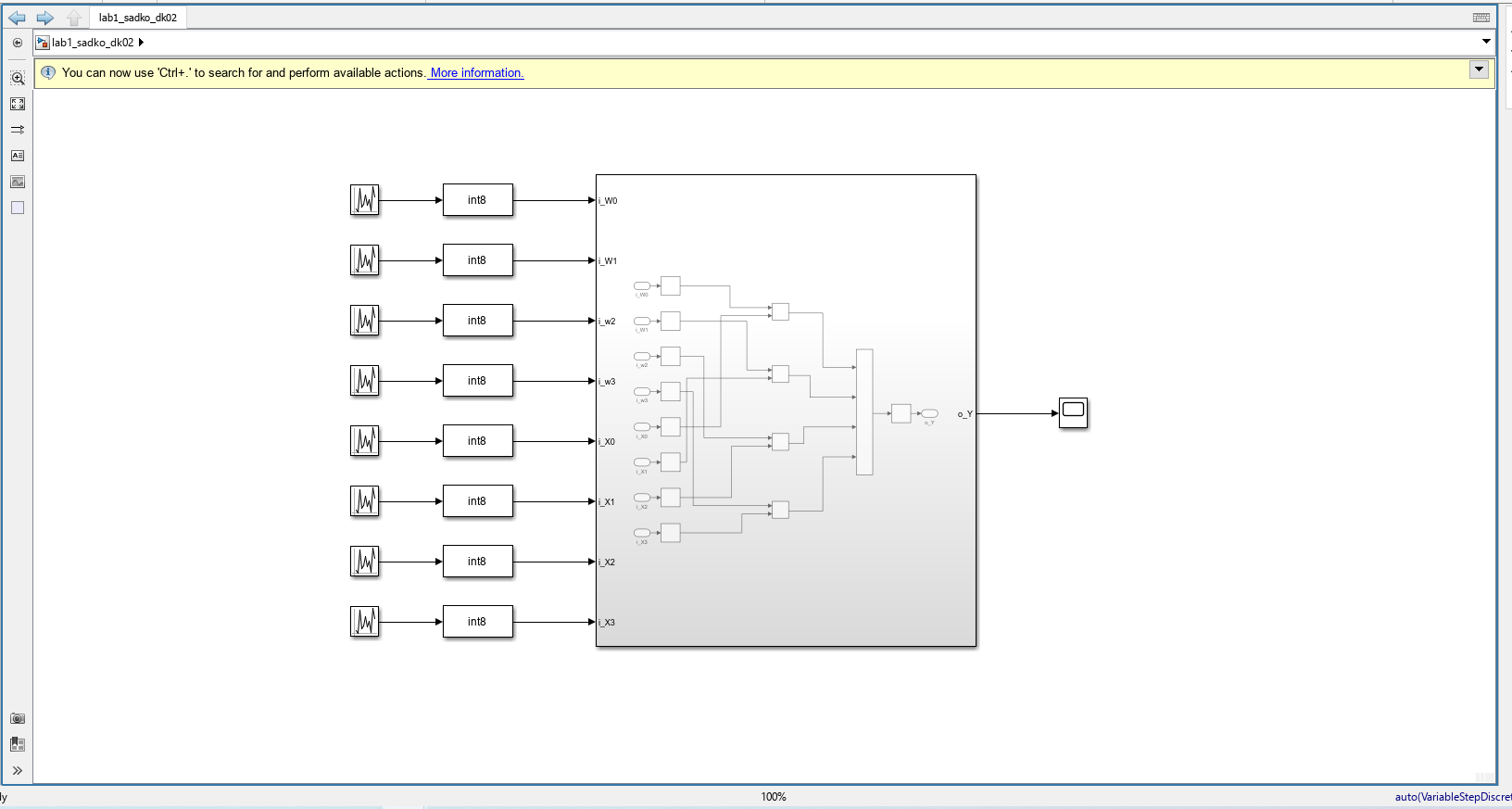


Рис. 3 Загальна схема

З схеми, яка проводить обчислення заданої функції, утворюємо підсистему:

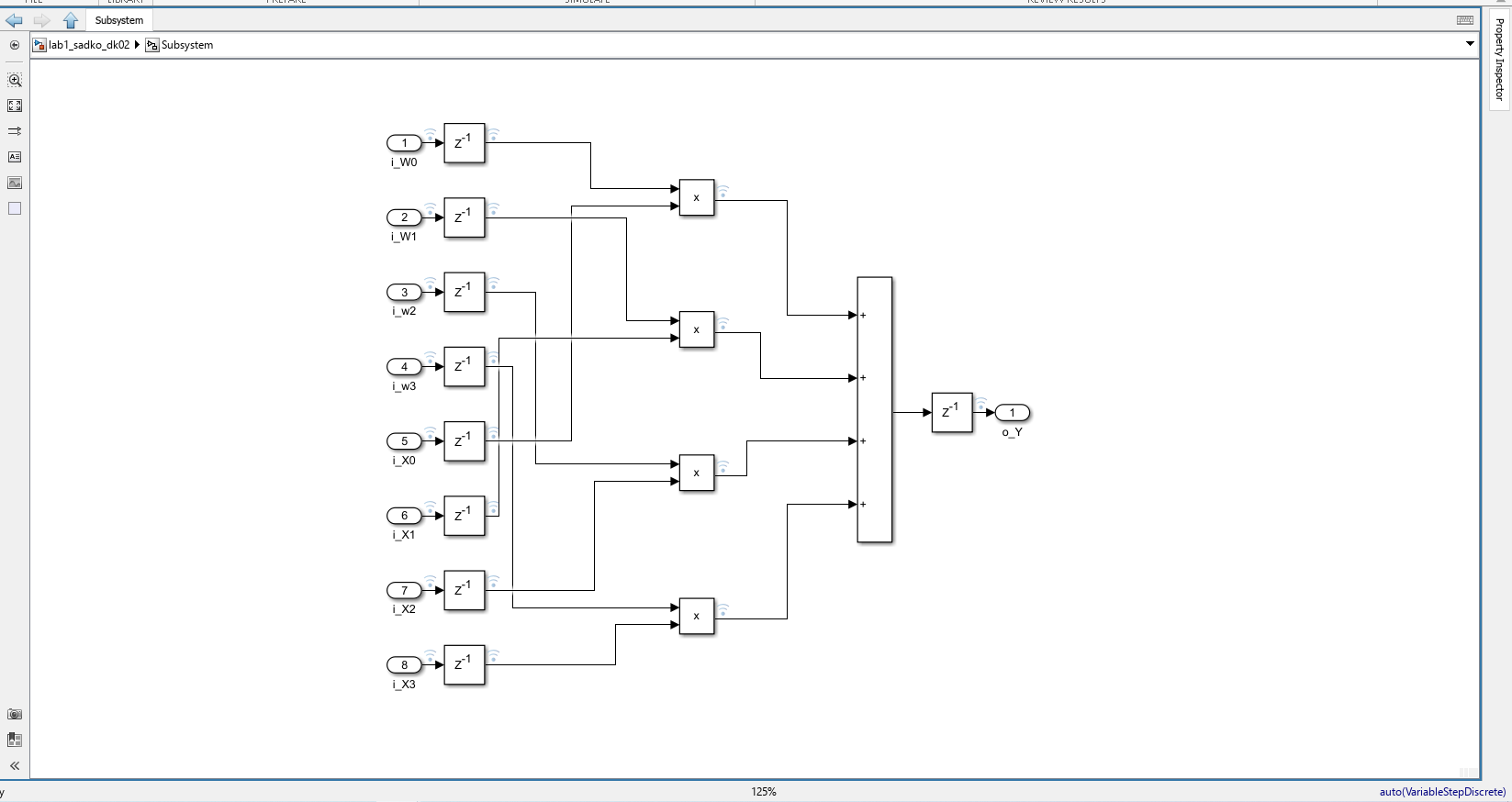


Рис. 4 Підсистема загальної схеми

Налаштування компонентів підсистеми:

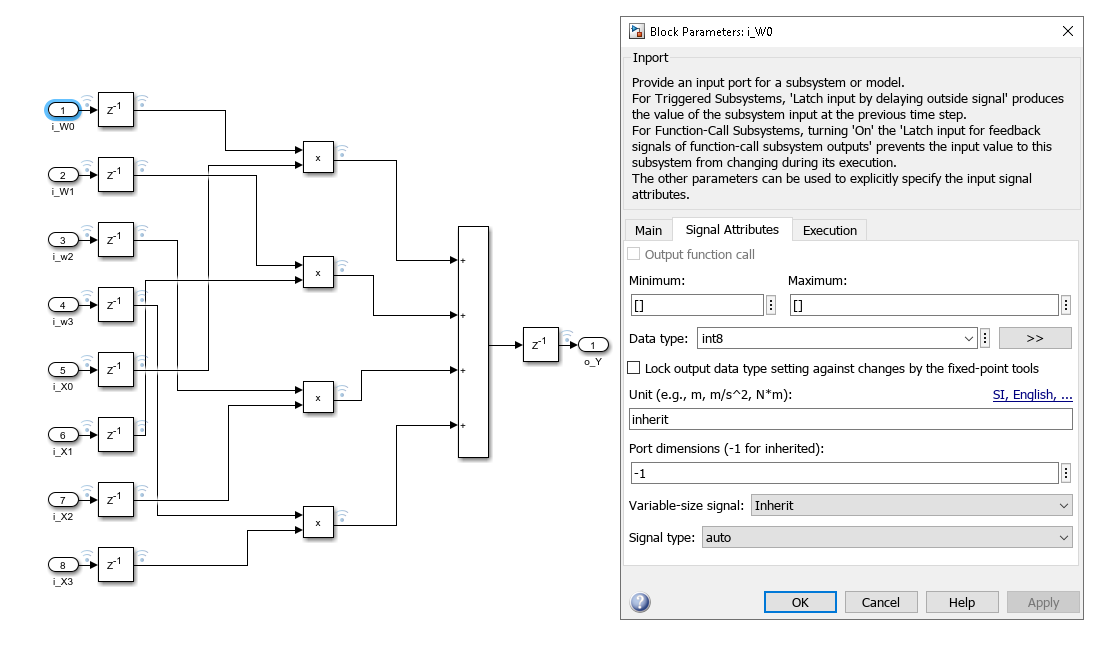


Рис. 5 Налаштування для входів i\_W

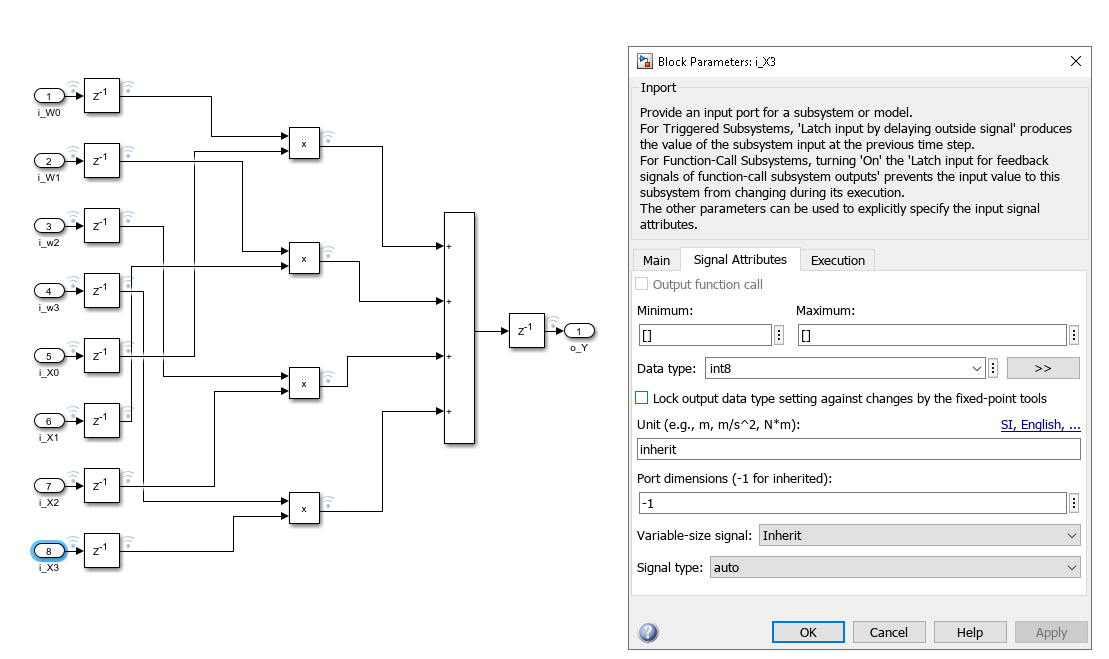


Рис. 6 Налаштування для входів i\_X

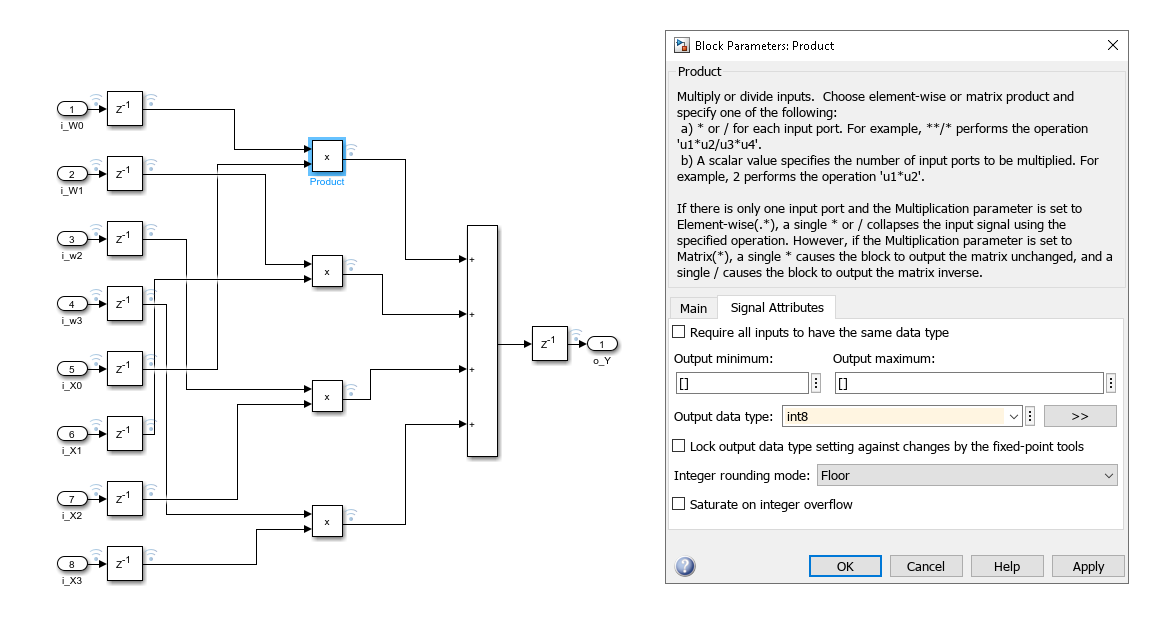


Рис. 7 Налаштування для блоків Product

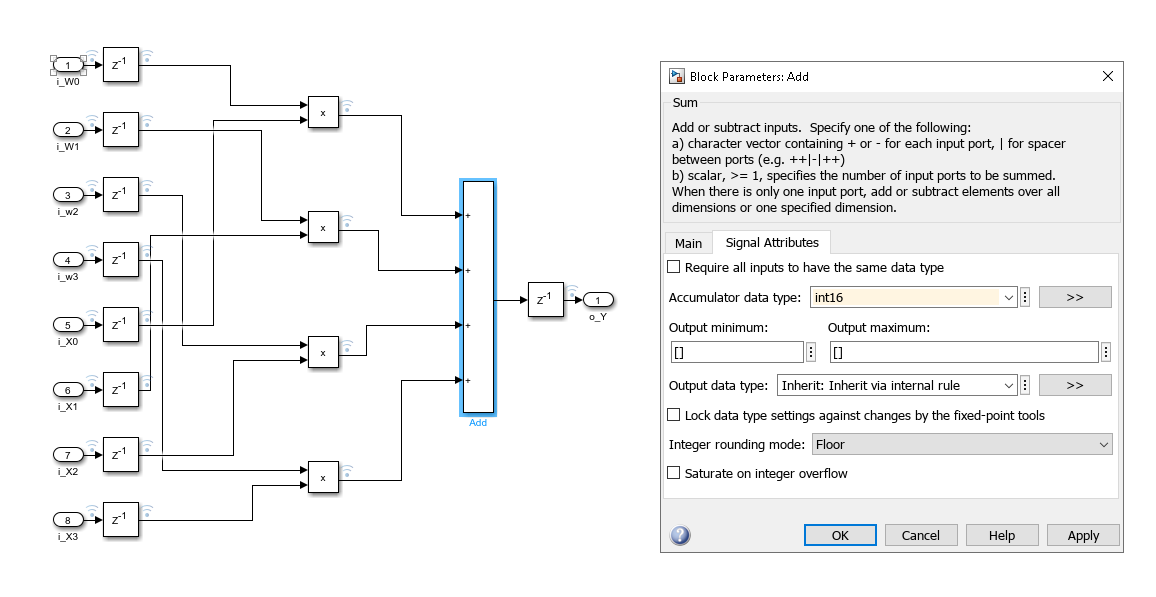


Рис. 8 Налаштування для блоків Add

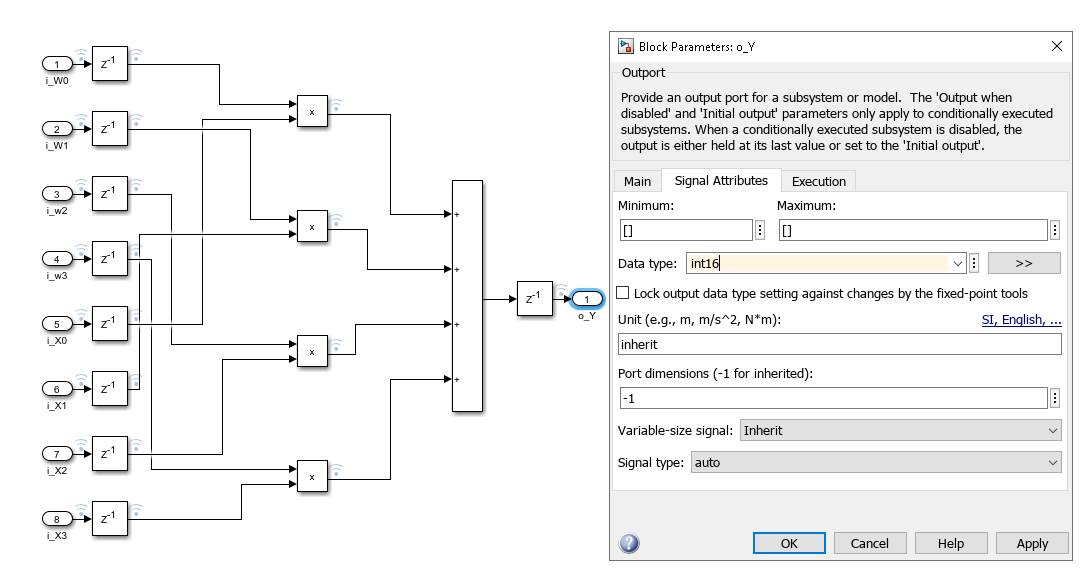


Рис. 9 Налаштування для виходу o\_Y

# 2. Перегляд в логічному аналазаторі даних на входах і на виході створеної підсистеми у знаковому десятковому поданні

Після компіляції системи в логічному аналізаторі можна відстежити результат. На рис. 10 можна побачити, що Delay0 утворює затримку на два такти.



Рис. 10 Результат симуляції схеми

# 3. Генерування коду на Verilog та синтез згенерованого коду в Quartus для створеної підсистеми

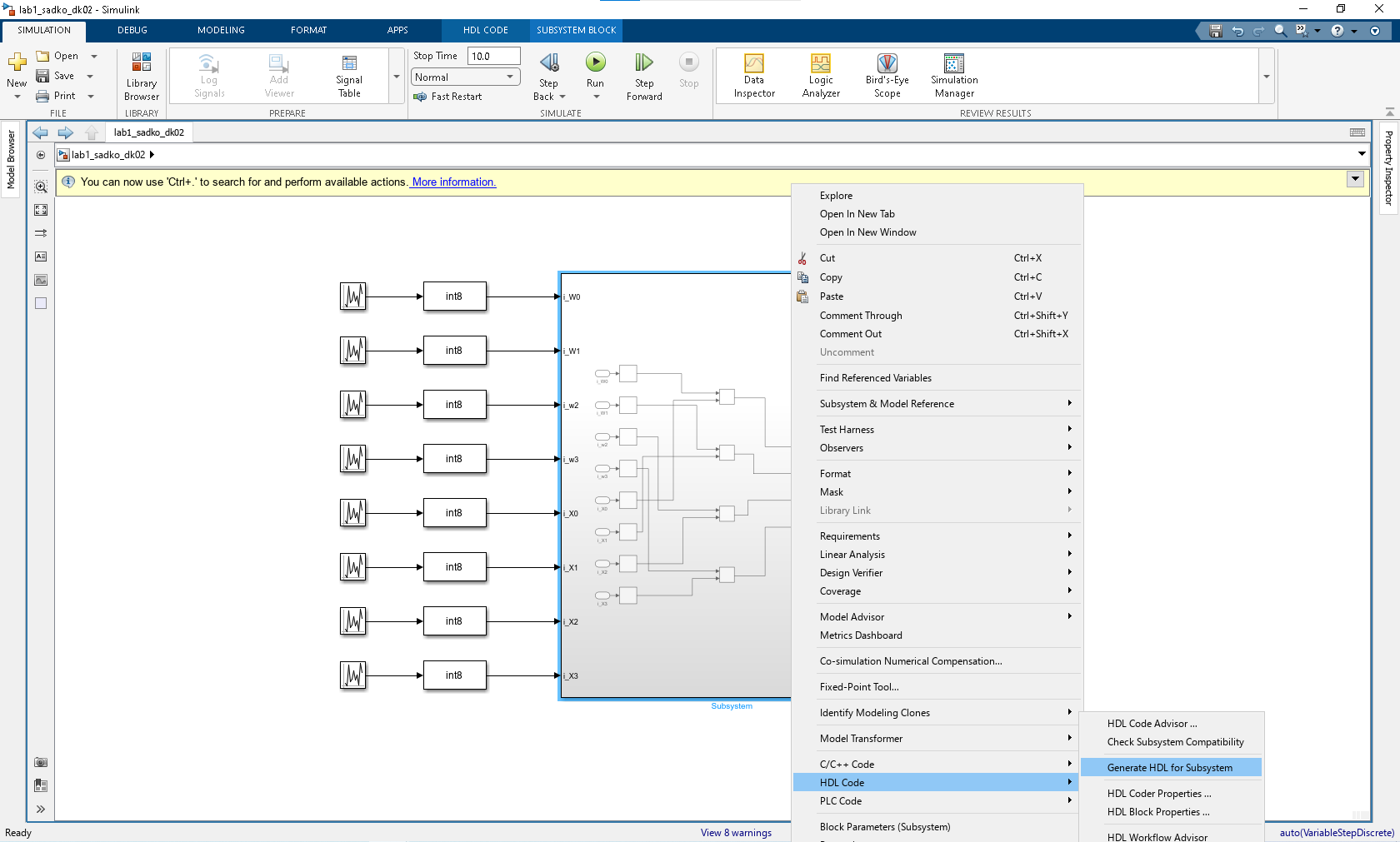


Рис. 11 Генерація коду на Verilog для підсистеми

Лістинг 3.1

// коментарі прибрав, але вони, як і сам .v файл будуть у директорії, де розміщенні будуть всі файли проєкту

`timescale 1 ns / 1 ns

module Subsystem

(clk,

reset,

clk\_enable,

i\_W0,

i\_W1,

i\_w2,

i\_w3,

i\_X0,

i\_X1,

i\_X2,

i\_X3,

ce\_out,

o\_Y);

input clk;

input reset;

input clk\_enable;

input signed [7:0] i\_W0; // int8

input signed [7:0] i\_W1; // int8

input signed [7:0] i\_w2; // int8

input signed [7:0] i\_w3; // int8

input signed [7:0] i\_X0; // int8

input signed [7:0] i\_X1; // int8

input signed [7:0] i\_X2; // int8

input signed [7:0] i\_X3; // int8

output ce\_out;

output signed [15:0] o\_Y; // int16

wire enb;

reg signed [7:0] Delay\_out1; // int8

reg signed [7:0] Delay1\_out1; // int8

reg signed [7:0] Delay2\_out1; // int8

reg signed [7:0] Delay3\_out1; // int8

reg signed [7:0] Delay4\_out1; // int8

wire signed [15:0] Product\_mul\_temp; // sfix16

wire signed [7:0] Product\_out1; // int8

reg signed [7:0] Delay5\_out1; // int8

wire signed [15:0] Product1\_out1; // int16

wire signed [15:0] Add\_stage2\_add\_temp; // sfix16

wire signed [15:0] Add\_stage2\_1; // sfix16

wire signed [16:0] Add\_op\_stage1; // sfix17

reg signed [7:0] Delay6\_out1; // int8

wire signed [15:0] Product2\_out1; // int16

wire signed [15:0] Add\_stage3\_add\_cast; // sfix16

wire signed [15:0] Add\_stage3\_add\_temp; // sfix16

wire signed [17:0] Add\_op\_stage2; // sfix18

reg signed [7:0] Delay7\_out1; // int8

wire signed [15:0] Product3\_out1; // int16

wire signed [15:0] Add\_stage4\_add\_cast; // sfix16

wire signed [15:0] Add\_out1; // int16

reg signed [15:0] Delay8\_out1; // int16

assign enb = clk\_enable;

always @(posedge clk or posedge reset)

begin : Delay\_process

if (reset == 1'b1) begin

Delay\_out1 <= 8'sb00000000;

end

else begin

if (enb) begin

Delay\_out1 <= i\_W0;

end

end

end

always @(posedge clk or posedge reset)

begin : Delay1\_process

if (reset == 1'b1) begin

Delay1\_out1 <= 8'sb00000000;

end

else begin

if (enb) begin

Delay1\_out1 <= i\_W1;

end

end

end

always @(posedge clk or posedge reset)

begin : Delay2\_process

if (reset == 1'b1) begin

Delay2\_out1 <= 8'sb00000000;

end

else begin

if (enb) begin

Delay2\_out1 <= i\_w2;

end

end

end

always @(posedge clk or posedge reset)

begin : Delay3\_process

if (reset == 1'b1) begin

Delay3\_out1 <= 8'sb00000000;

end

else begin

if (enb) begin

Delay3\_out1 <= i\_w3;

end

end

end

always @(posedge clk or posedge reset)

begin : Delay4\_process

if (reset == 1'b1) begin

Delay4\_out1 <= 8'sb00000000;

end

else begin

if (enb) begin

Delay4\_out1 <= i\_X0;

end

end

end

assign Product\_mul\_temp = Delay\_out1 \* Delay4\_out1;

assign Product\_out1 = Product\_mul\_temp[7:0];

always @(posedge clk or posedge reset)

begin : Delay5\_process

if (reset == 1'b1) begin

Delay5\_out1 <= 8'sb00000000;

end

else begin

if (enb) begin

Delay5\_out1 <= i\_X1;

end

end

end

assign Product1\_out1 = Delay1\_out1 \* Delay5\_out1;

assign Add\_stage2\_1 = {{8{Product\_out1[7]}}, Product\_out1};

assign Add\_stage2\_add\_temp = Add\_stage2\_1 + Product1\_out1;

assign Add\_op\_stage1 = {Add\_stage2\_add\_temp[15], Add\_stage2\_add\_temp};

always @(posedge clk or posedge reset)

begin : Delay6\_process

if (reset == 1'b1) begin

Delay6\_out1 <= 8'sb00000000;

end

else begin

if (enb) begin

Delay6\_out1 <= i\_X2;

end

end

end

assign Product2\_out1 = Delay2\_out1 \* Delay6\_out1;

assign Add\_stage3\_add\_cast = Add\_op\_stage1[15:0];

assign Add\_stage3\_add\_temp = Add\_stage3\_add\_cast + Product2\_out1;

assign Add\_op\_stage2 = {{2{Add\_stage3\_add\_temp[15]}}, Add\_stage3\_add\_temp};

always @(posedge clk or posedge reset)

begin : Delay7\_process

if (reset == 1'b1) begin

Delay7\_out1 <= 8'sb00000000;

end

else begin

if (enb) begin

Delay7\_out1 <= i\_X3;

end

end

end

assign Product3\_out1 = Delay3\_out1 \* Delay7\_out1;

assign Add\_stage4\_add\_cast = Add\_op\_stage2[15:0];

assign Add\_out1 = Add\_stage4\_add\_cast + Product3\_out1;

always @(posedge clk or posedge reset)

begin : Delay8\_process

if (reset == 1'b1) begin

Delay8\_out1 <= 16'sb0000000000000000;

end

else begin

if (enb) begin

Delay8\_out1 <= Add\_out1;

end

end

end

assign o\_Y = Delay8\_out1;

assign ce\_out = clk\_enable;

endmodule // Subsystem

# 4.Результат синтезу в RTL Viewer у квартусі та визначення апаратних витрат:

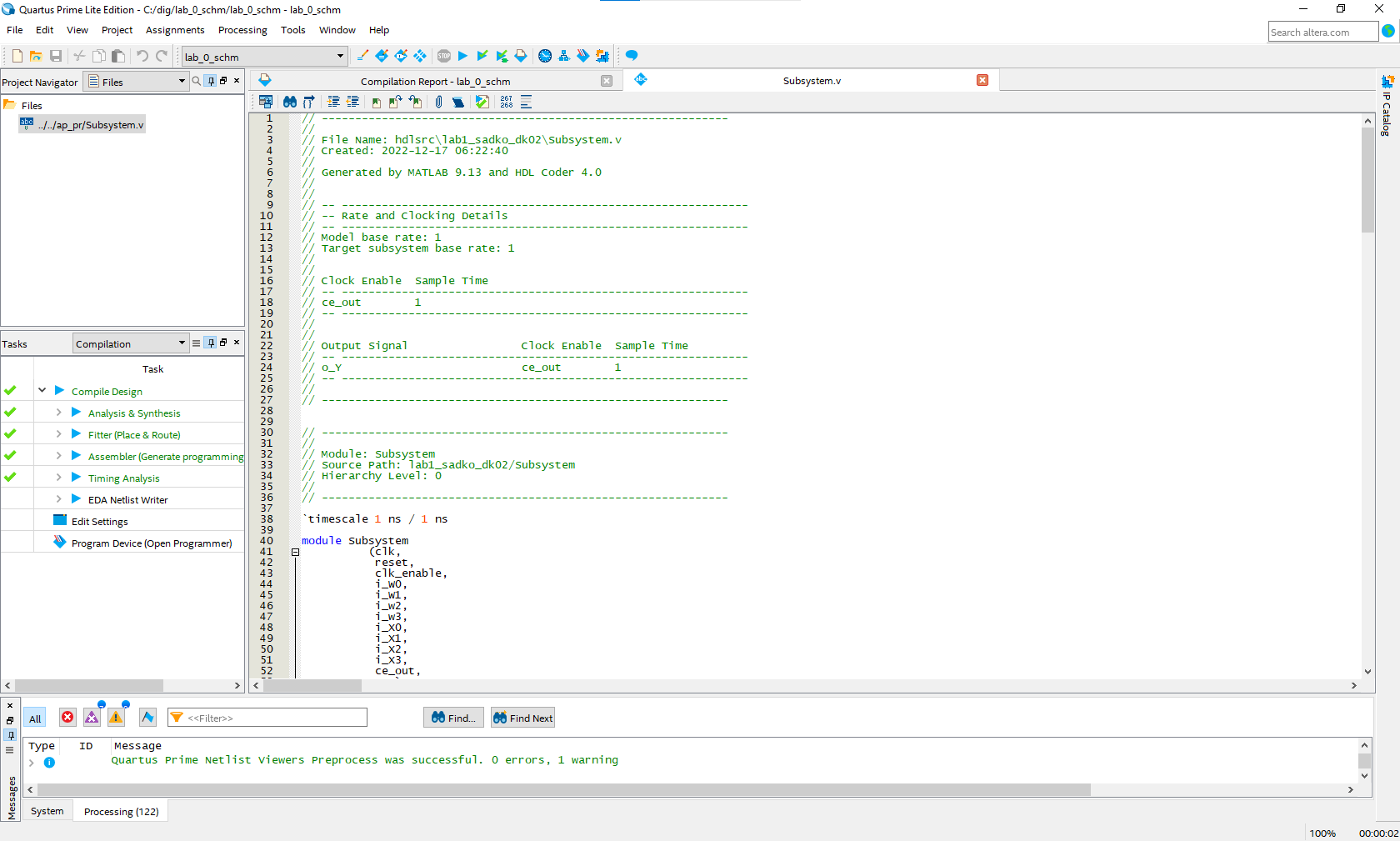


Рис. 12 Результат компіляції згенерованого коду у Quartus Prime Lite

За рис. 13 можна побачити, що у нас все правильно було згенеровано, особлива увага на розрядності, на входах по 8-біт, на Mult, Add та виході 16-біт, як і було задано в умові. На рис. 14 після компіляції файлу, як результат, можемо відстежити які апаратні витрати на цю підсистему.

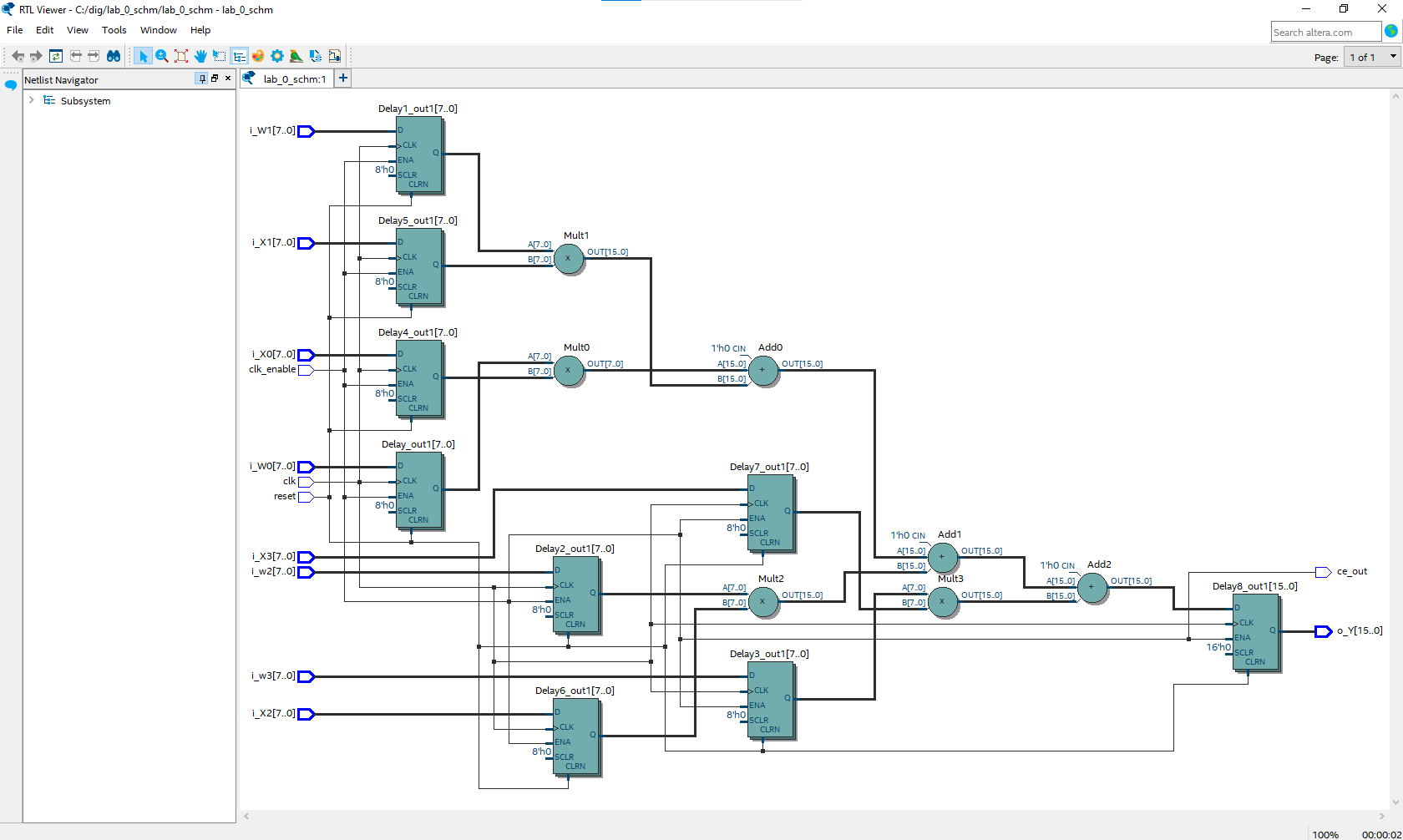


Рис. 13 Синтез згенерованого коду через RTL viewer

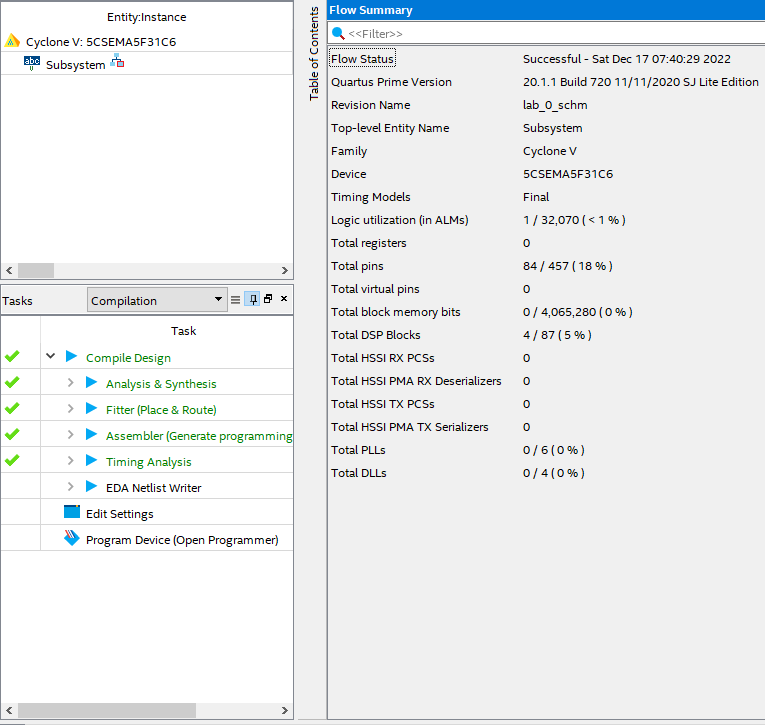


Рис. 14 Результат компіляції та дані про апаратні витрати підсистеми

# 5. Створення тестбенч файлу в Matlab для створеної підсистеми і результат симуляції тестбенча в Modelsim

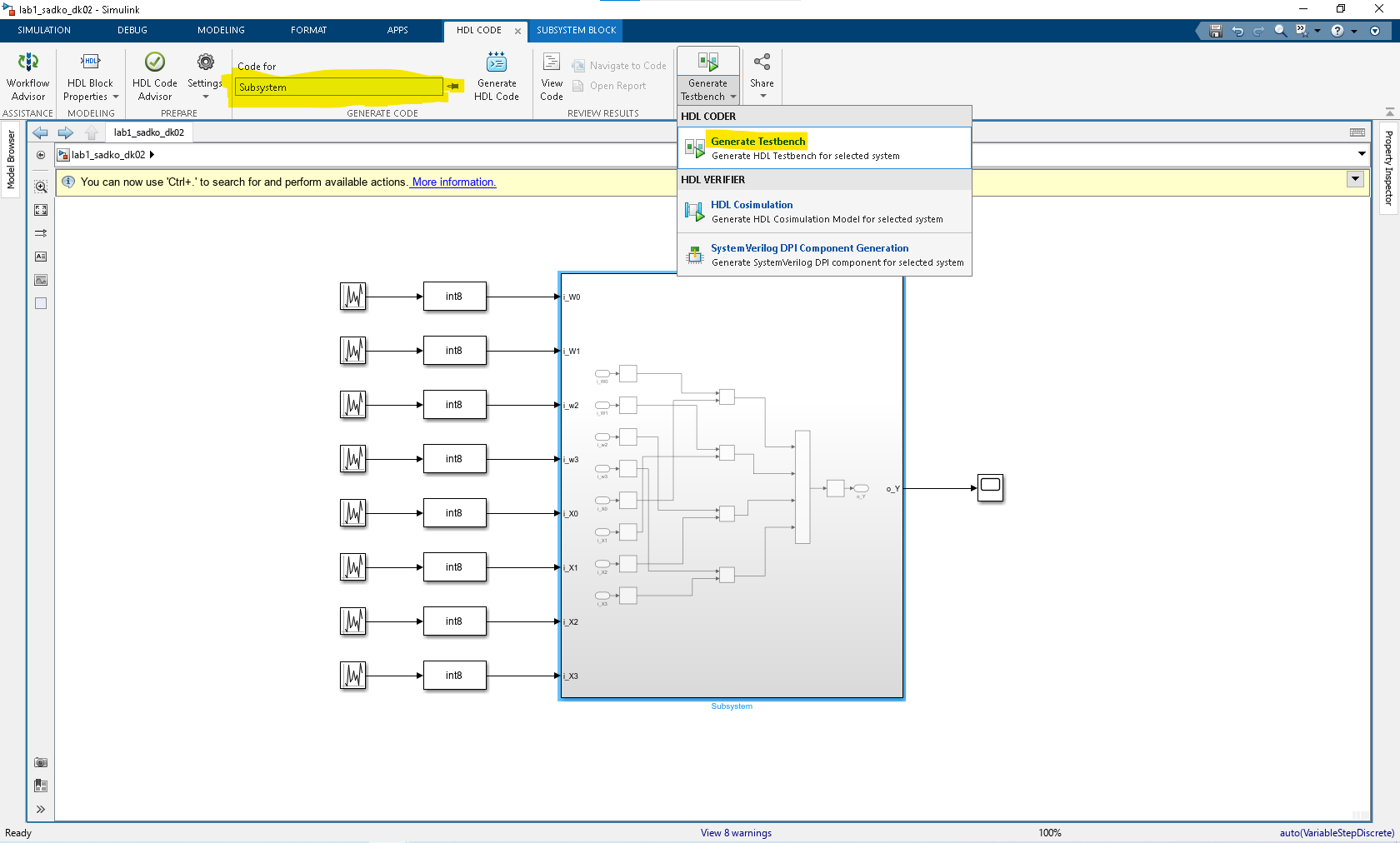


Рис. 15 Генерування тестбенч файлу для згенерованого коду

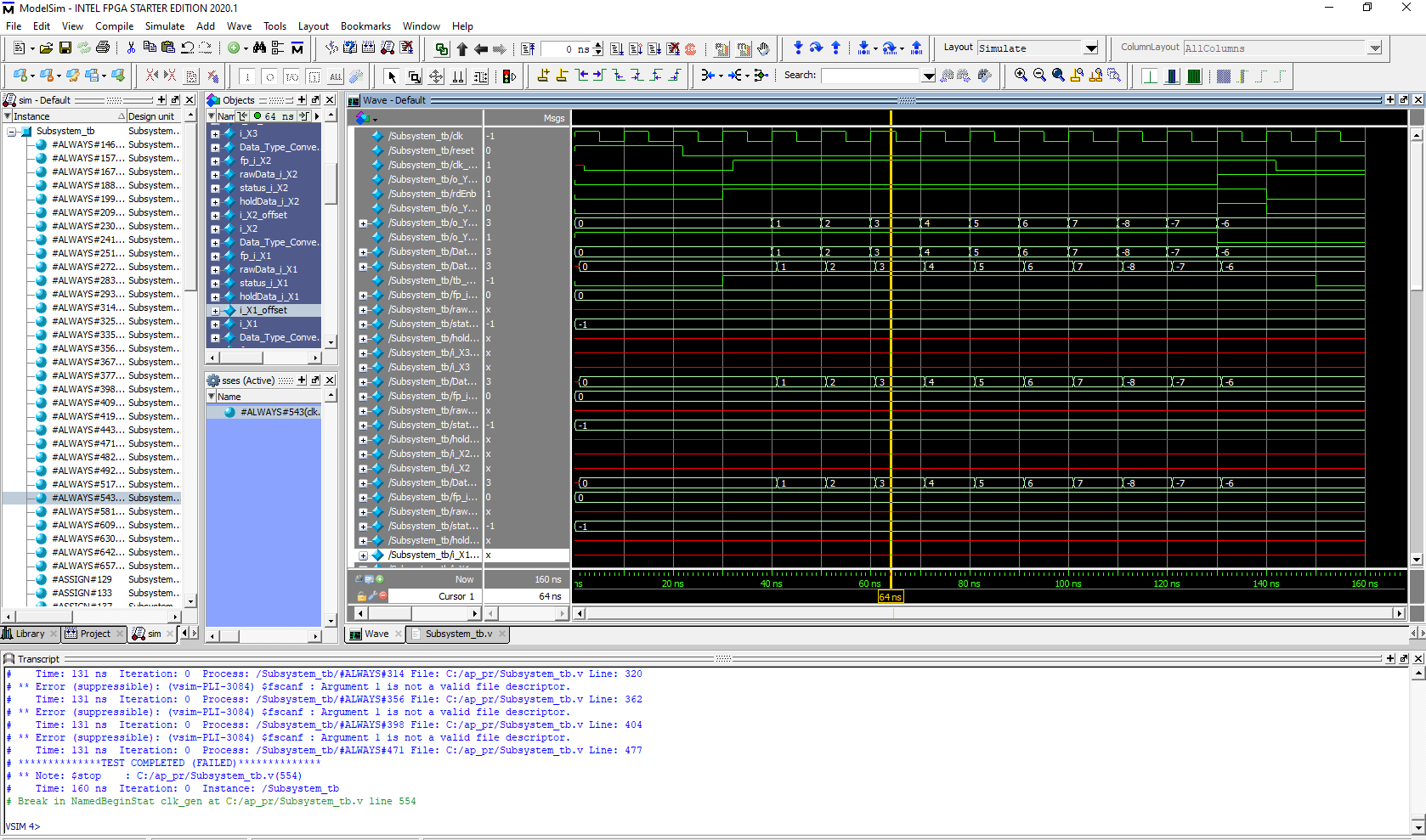


Рис. 16 Результат компіляції та запуску тестбенч файлу

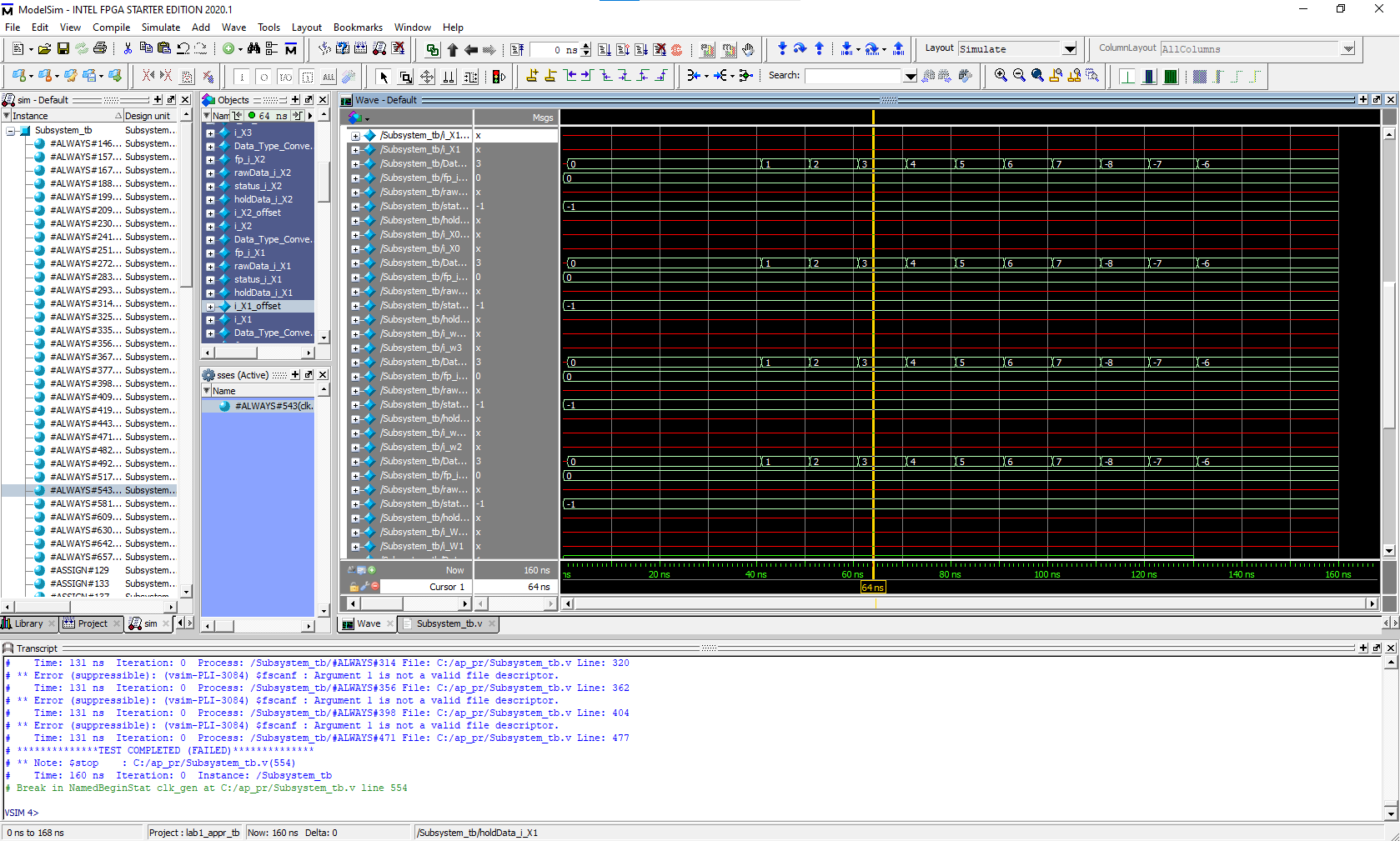


Рис. 17 Результат компіляції та запуску тестбенч файлу

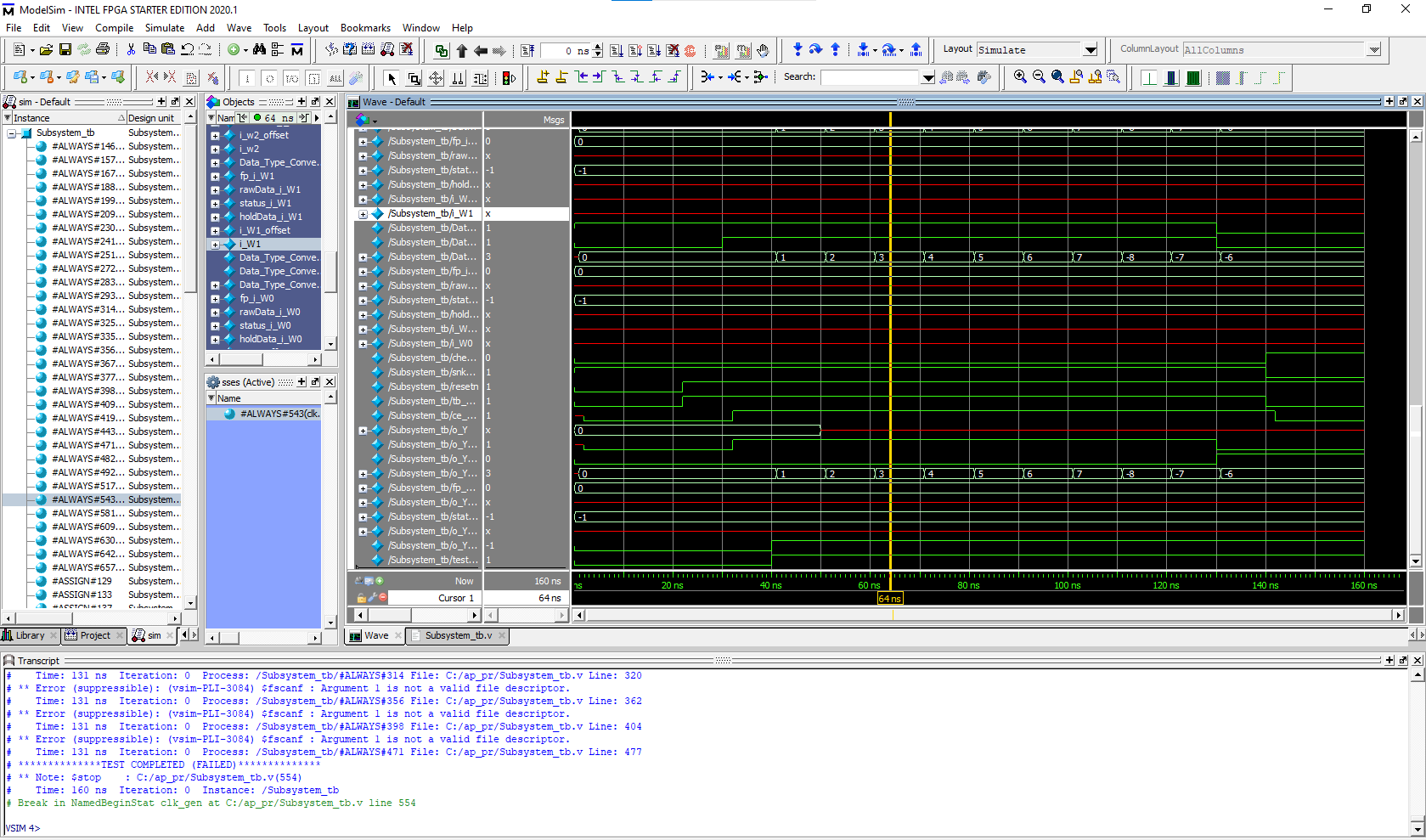


Рис. 18 Результат компіляції та запуску тестбенч файлу

# Висновок:

В процесі виконання лабораторної роботи ознайомився з налаштуванням Matlab та Simulink, склавши систему наведену на рис. 3 та підсистему на рис. 4. В результаті зкомпілюваваши цю систему отримав результати обчислень на рис. 10.

Додатково провів окреме дослідження, згенерувавши Verilog код підсистеми і в Quartus отримав досить корисні дані про апартні витрати підсистеми на рис. 14 , це дуже важливо для проведень оптимізацій різних систем в процесі розробки і тестування.

Також був згенерований ще тестбенч файл для вищезгаданого коду, це дало змогу окремо протестувати підсистему користуючись іншим програмним забезпеченням.