Міністерство освіти і науки України Національний технічний університет України «Київський Політехнічний Інститут імені Ігоря Сікорського» Кафедра конструювання електронно-обчислювальної апаратури

**Лабораторна робота №1**

**з дисципліни “ Апаратні прискорювачі ”**

**Виконав:** студент групи ДК-02 Дубовик В. Ю.

**Перевірив:** доц. Короткий Є В.

***Київ – 2023***

1. Реалізувати в simulink підсистему, що реалізує наступну функцію:

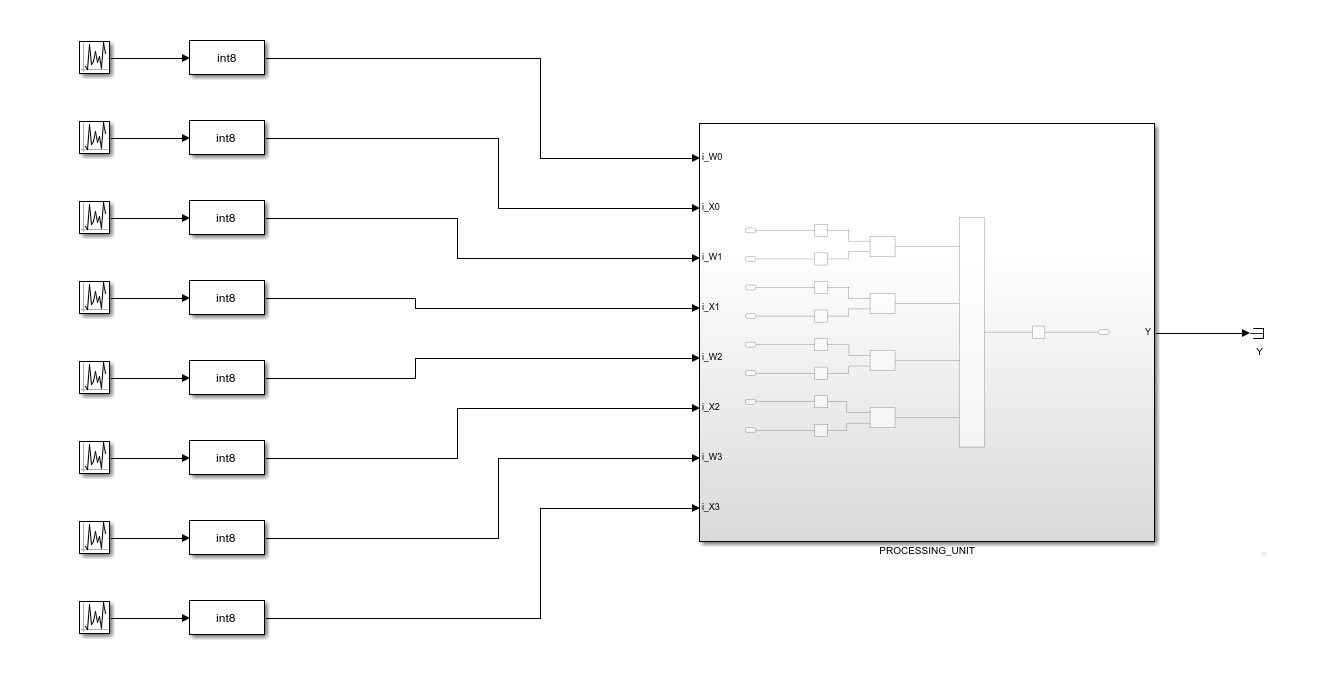
Y = W0\*X0 + W1\*X1 + W2\*X2 + W\*X3

Типи даних входів: int8

Тип даних виходу: int16

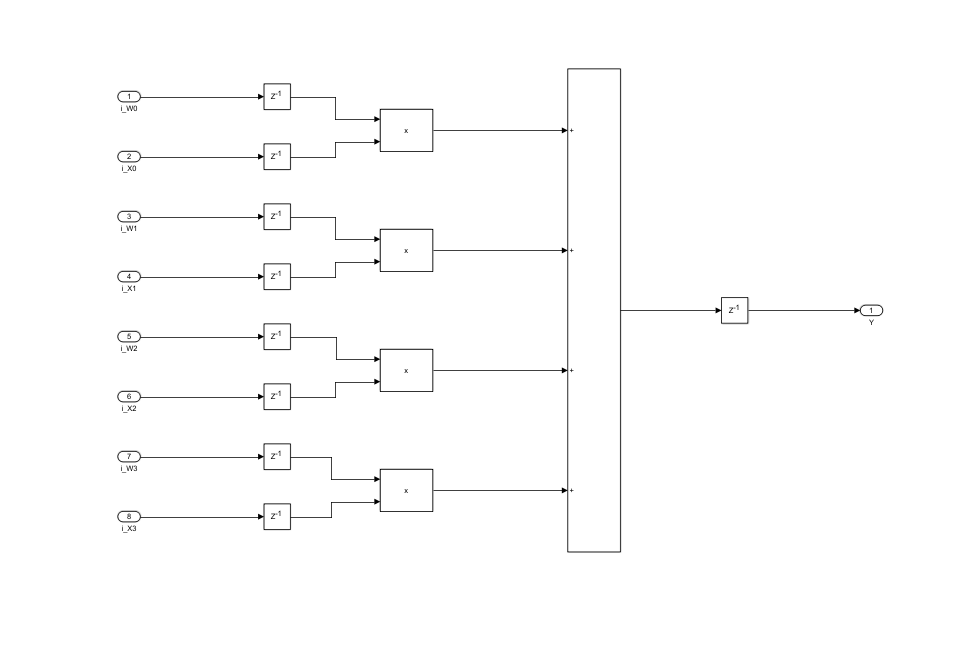
На входах і виході поставити регістри (блок затримки на 1 такт).

Створимо схему у Simulink-у:

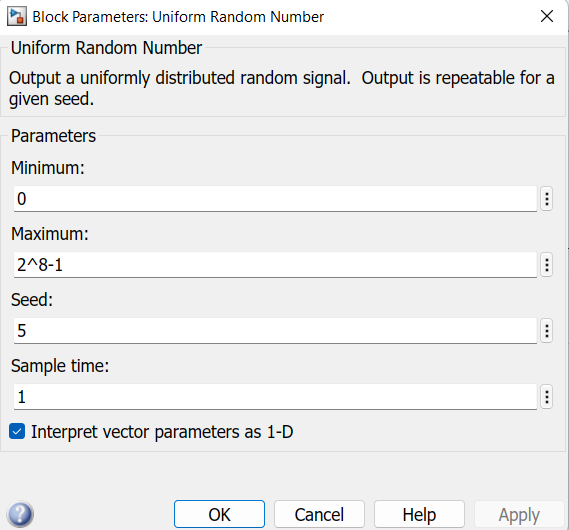


*Мал.*1 Основна схема

Маємо підсистеми PROCESSING\_UNIT:



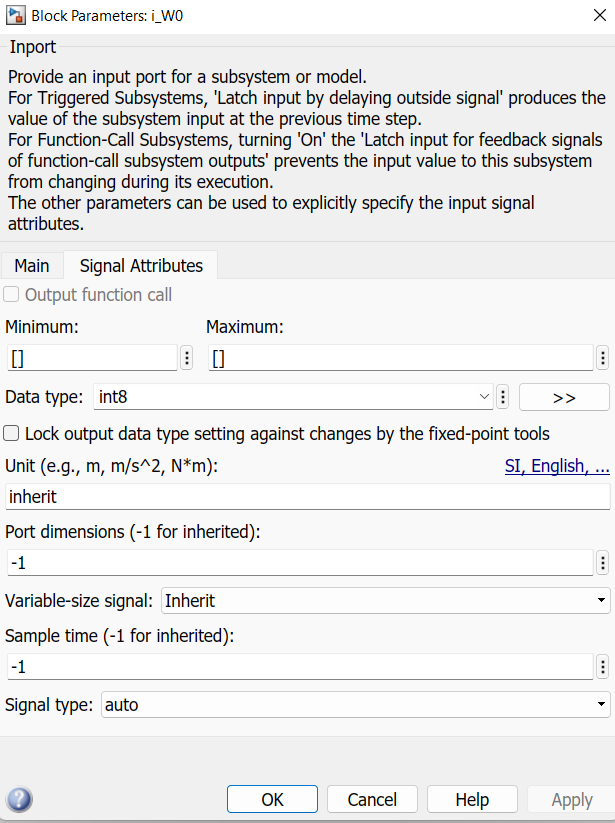
*Мал.*2 Підсистема



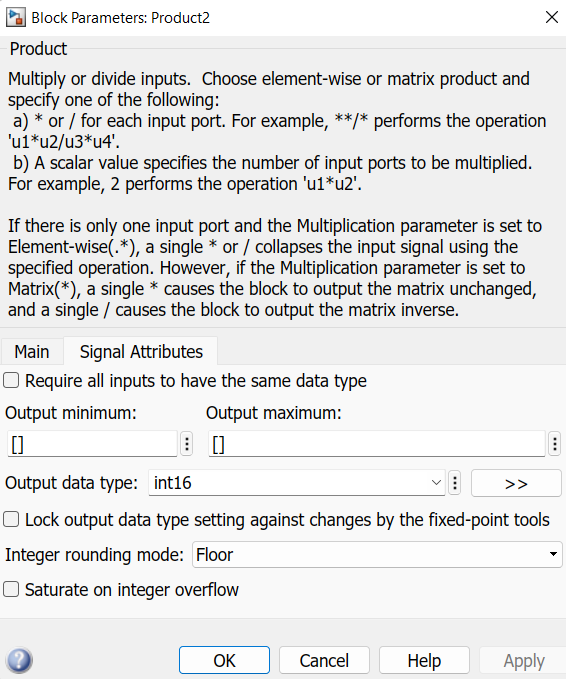
Наступні блоки Uniform Random Number робимо з таким же максимумом та мінімумом, але кожен наступний блок з кроком сіда на 1 більший.

**Далі розглянемо налаштування підсистем:**

Налаштуємо входи i\_W0 - i\_W3 та i\_X0 – i\_X3:

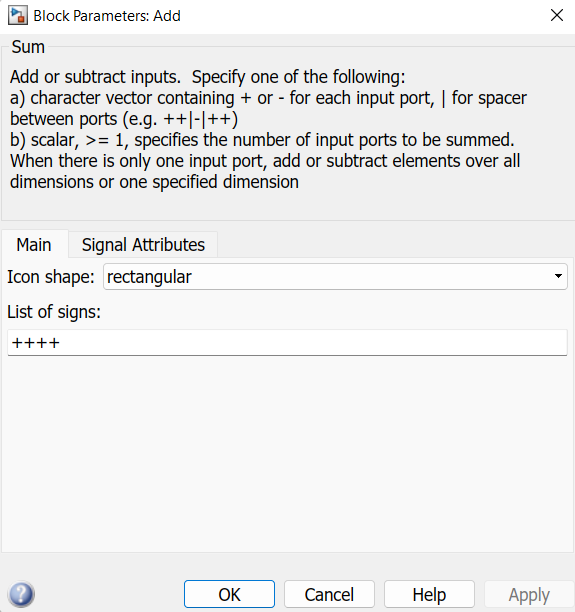


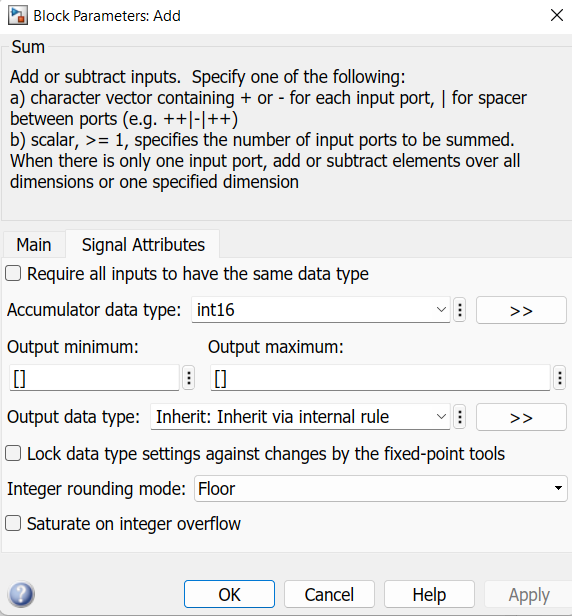
Налаштуємо блоки Product:



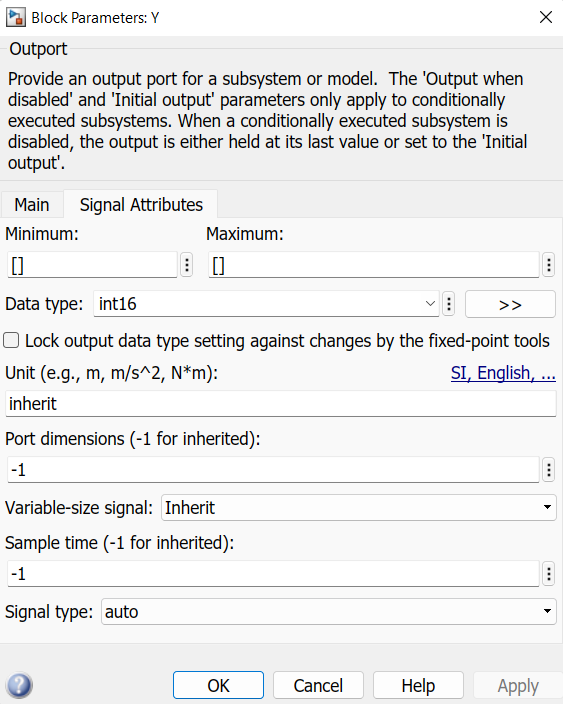
Тут зрозуміло, що при перемноженні тип даних на виході має стати int16.

Налаштування блока ADD:

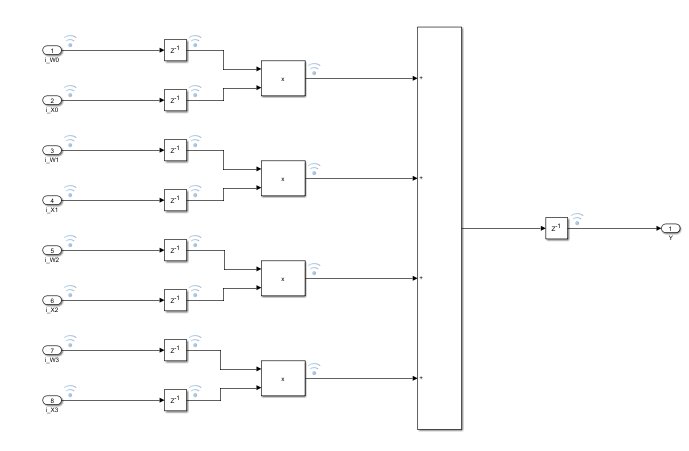




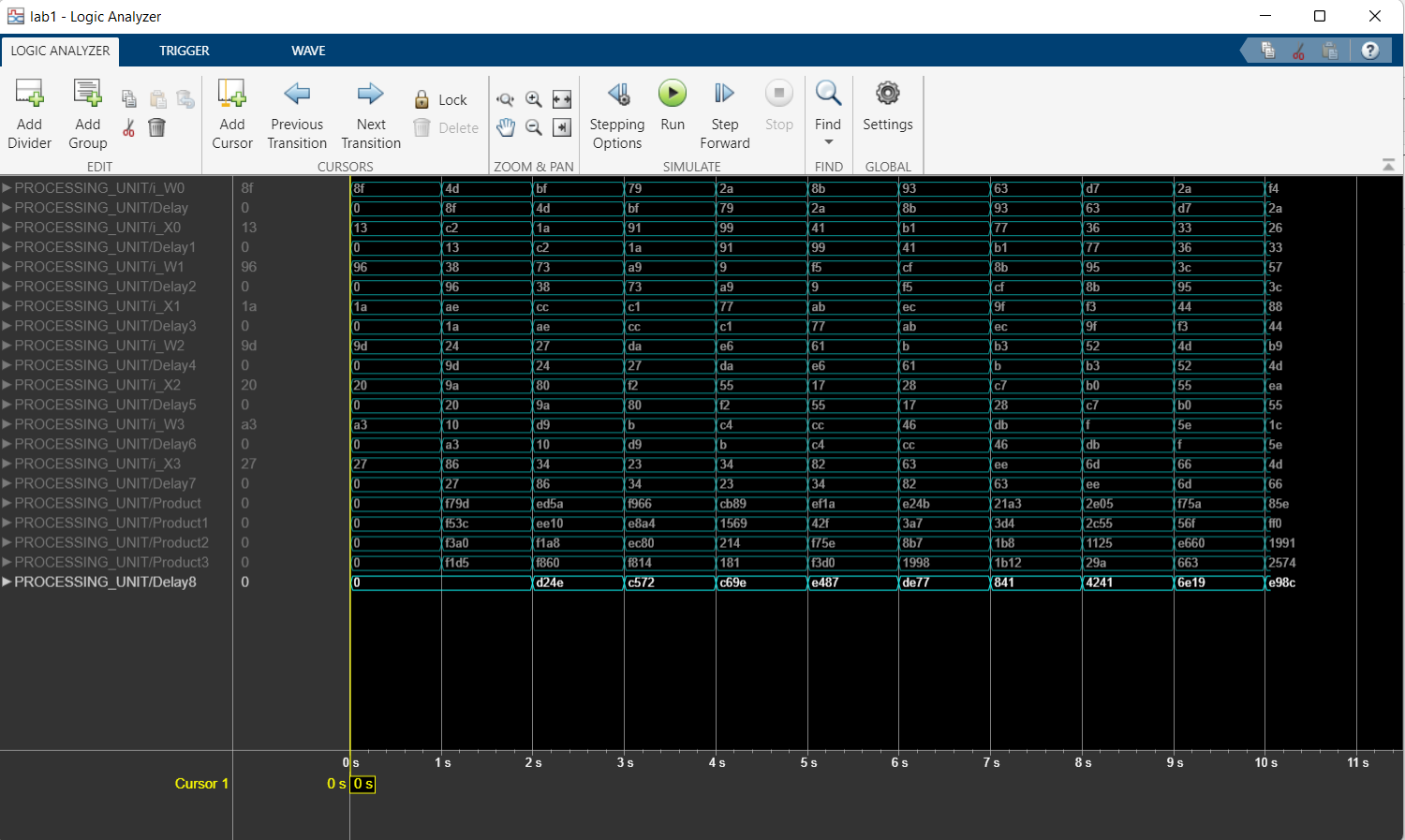
Налаштуємо Output Y:



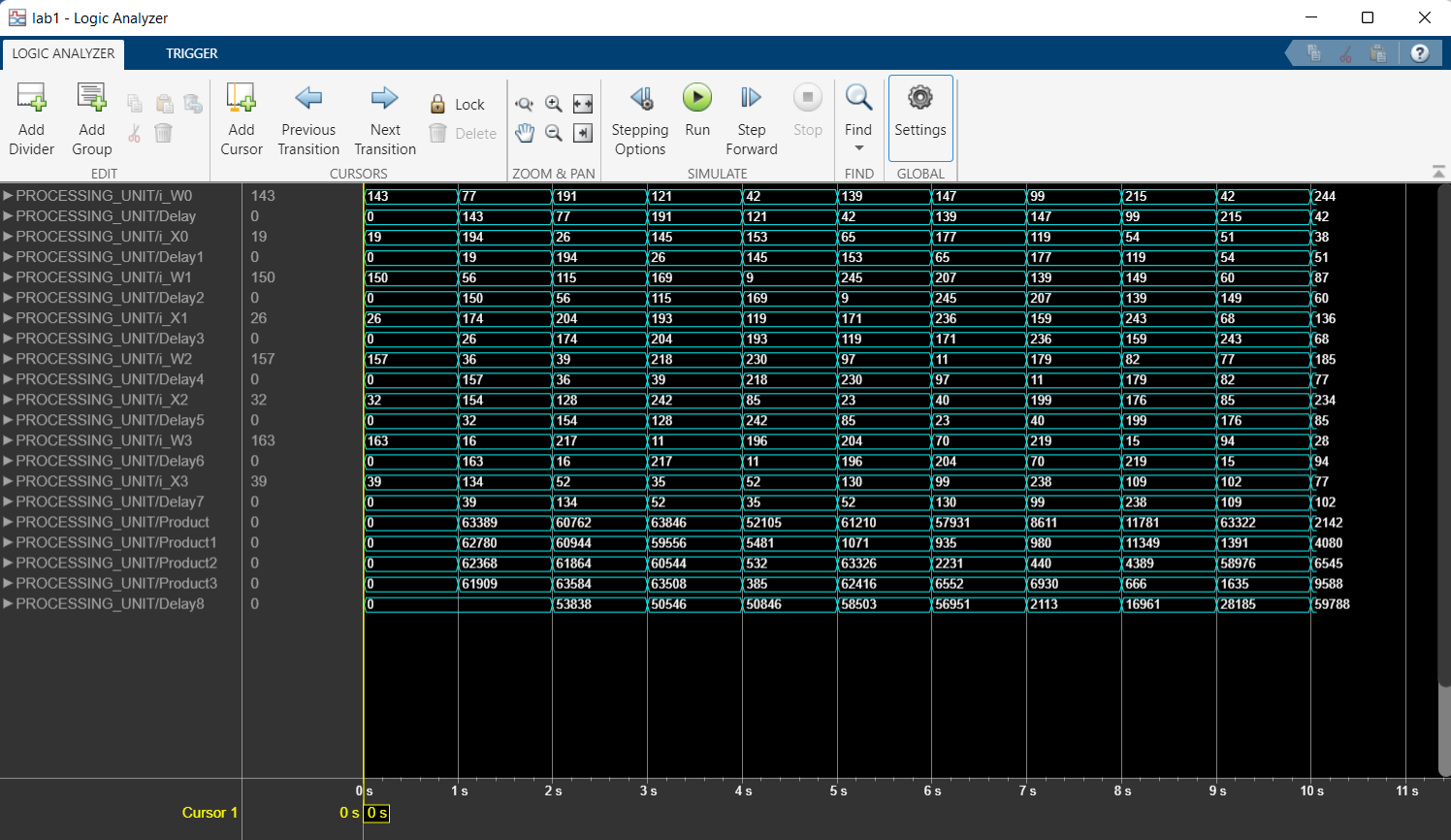
1. **Далі можемо розглянемо результат симуляції в логічному аналізаторі. Виберемо елементи для візуалізації:**



Отримуємо симуляцію:



Налаштуємо відображення у десятковій беззнаковій системі числення:



1. **Далі згенеруємо Verilog-код та за допомогою коду побачимо апаратні витрати програми та схему, згенеровану RTL Viewer-ом.**

*Код:*

`timescale 1 ns / 1 ns

module PROCESSING\_UNIT

(clk,

reset,

clk\_enable,

i\_W0,

i\_X0,

i\_W1,

i\_X1,

i\_W2,

i\_X2,

i\_W3,

i\_X3,

ce\_out,

Y);

input clk;

input reset;

input clk\_enable;

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

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

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

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

input signed [7:0] i\_W2; // int8

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

input signed [7:0] i\_W3; // int8

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

output ce\_out;

output signed [15:0] Y; // int16

wire enb;

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

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

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

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

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

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

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

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

wire signed [15:0] Add\_op\_stage1; // sfix16

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

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

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

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

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

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 : Delay2\_process

if (reset == 1'b1) begin

Delay2\_out1 <= 8'sb00000000;

end

else begin

if (enb) begin

Delay2\_out1 <= i\_W1;

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\_W2;

end

end

end

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\_W3;

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\_X0;

end

end

end

assign Product\_out1 = Delay\_out1 \* Delay1\_out1;

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\_X1;

end

end

end

assign Product1\_out1 = Delay2\_out1 \* Delay3\_out1;

assign Add\_op\_stage1 = Product\_out1 + Product1\_out1;

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\_X2;

end

end

end

assign Product2\_out1 = Delay4\_out1 \* Delay5\_out1;

assign Add\_op\_stage2 = Add\_op\_stage1 + Product2\_out1;

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 = Delay6\_out1 \* Delay7\_out1;

assign Add\_out1 = Add\_op\_stage2 + 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

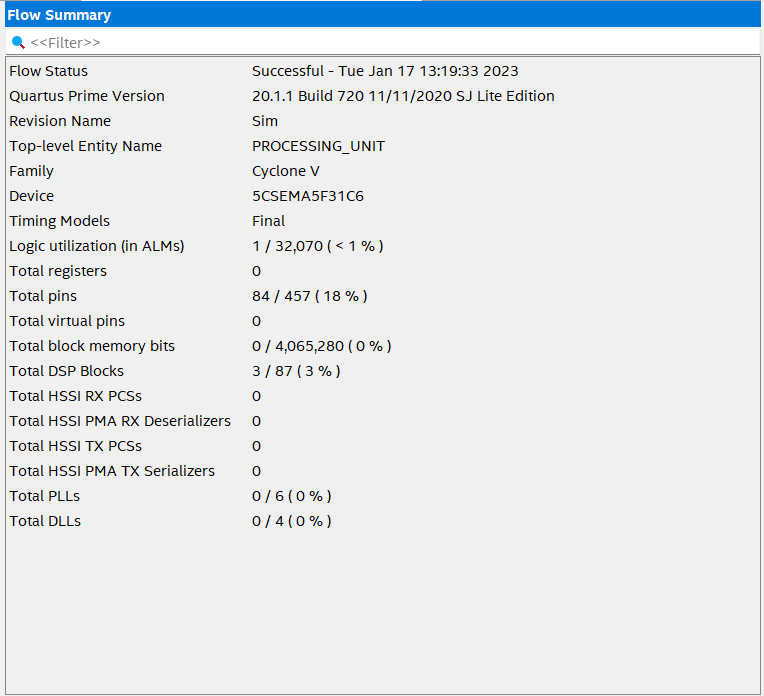
end

assign Y = Delay8\_out1;

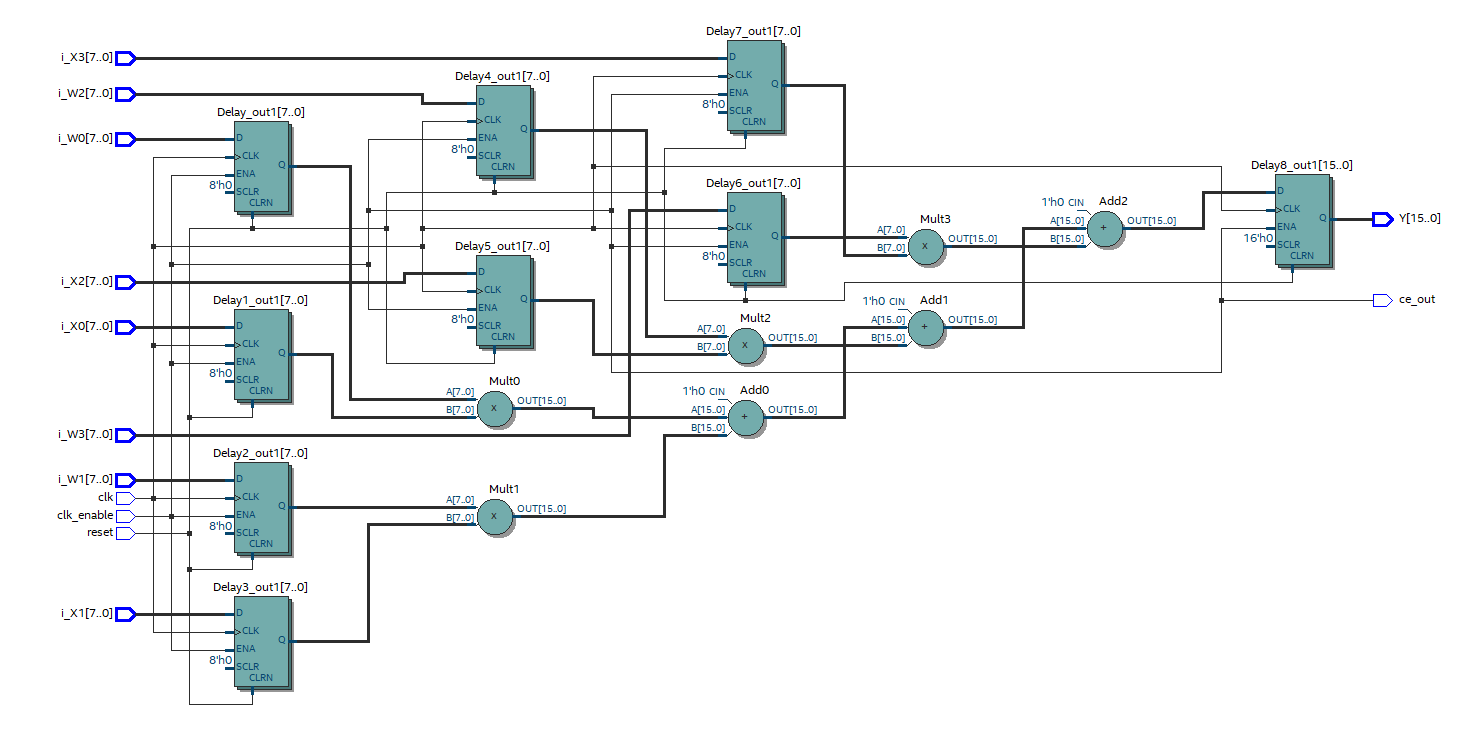
assign ce\_out = clk\_enable;

endmodule // PROCESSING\_UNIT

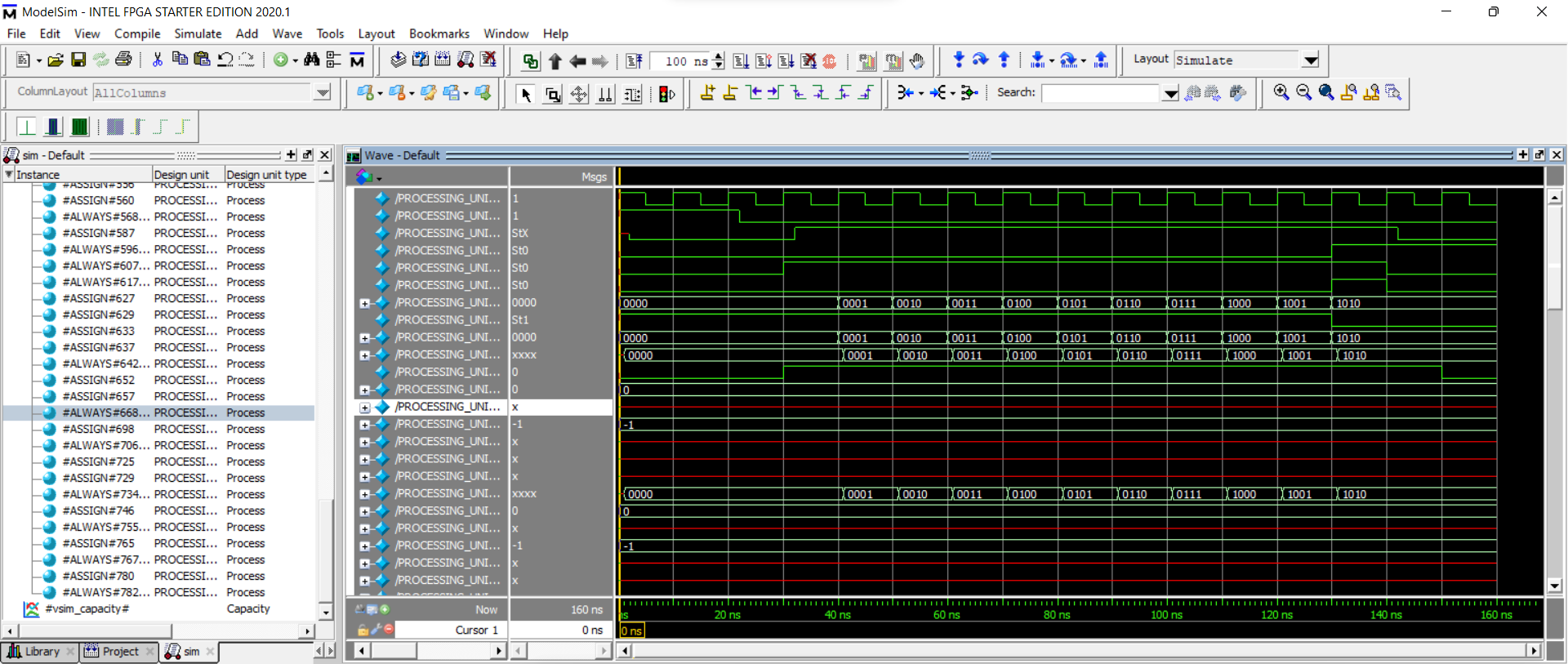
Після генерації проекту у Quartus маємо такі результати витрат:



Тепер синтезуємо код за допомогою RTL Viewer-а для наглядної демонстрації нашої схеми:



За допомогою згенерованого коду та тестбенчу, можна переконатися у працездатності ще раз і переглянути симуляцію у моделсімі:



**Висновок**

У результаті виконання лабораторної роботи, було успішно побудовано схему, що реалізує задану функцію за допомогою Matlab-Simulink. Навчився створювати схема та підсистеми у середовищі Matlab-Simulink. Завдяки автоматично згенерованому коду, вдалося переглянути симуляцію схеми у програмі ModelSim, а за допомогою Quartus та інструмента RTL Viewer отримав реалізацію схеми і її апаратні витрати. Загалом, можна сказати, що навички роботи з Matlab-Simulink дозволять досить швидко генерувати і симулювати код безпосередньо для того, щоб протестувати його у симуляції, або на FPGA.

Тестбенч та згенерований код можна знайти у відповідному репозиторії на ГітХаб: <https://github.com/0XXX0TNIK/Matlab-KEOA/tree/main/lab1>