МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО”

КАФЕДРА КОНСТРУЮВАННЯ ЕОА

Лабораторна робота №1 з предмету «Аппаратні прискорювачі»

Виконав:

Студент гр. ДК-01

Суханов Д.О.

Перевірив:

доц. Короткий Є В.

Київ 2022

**Завдання**

В Simulink реалізувати підсистему, що розраховує функцію:

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

Типи даних входів: int8

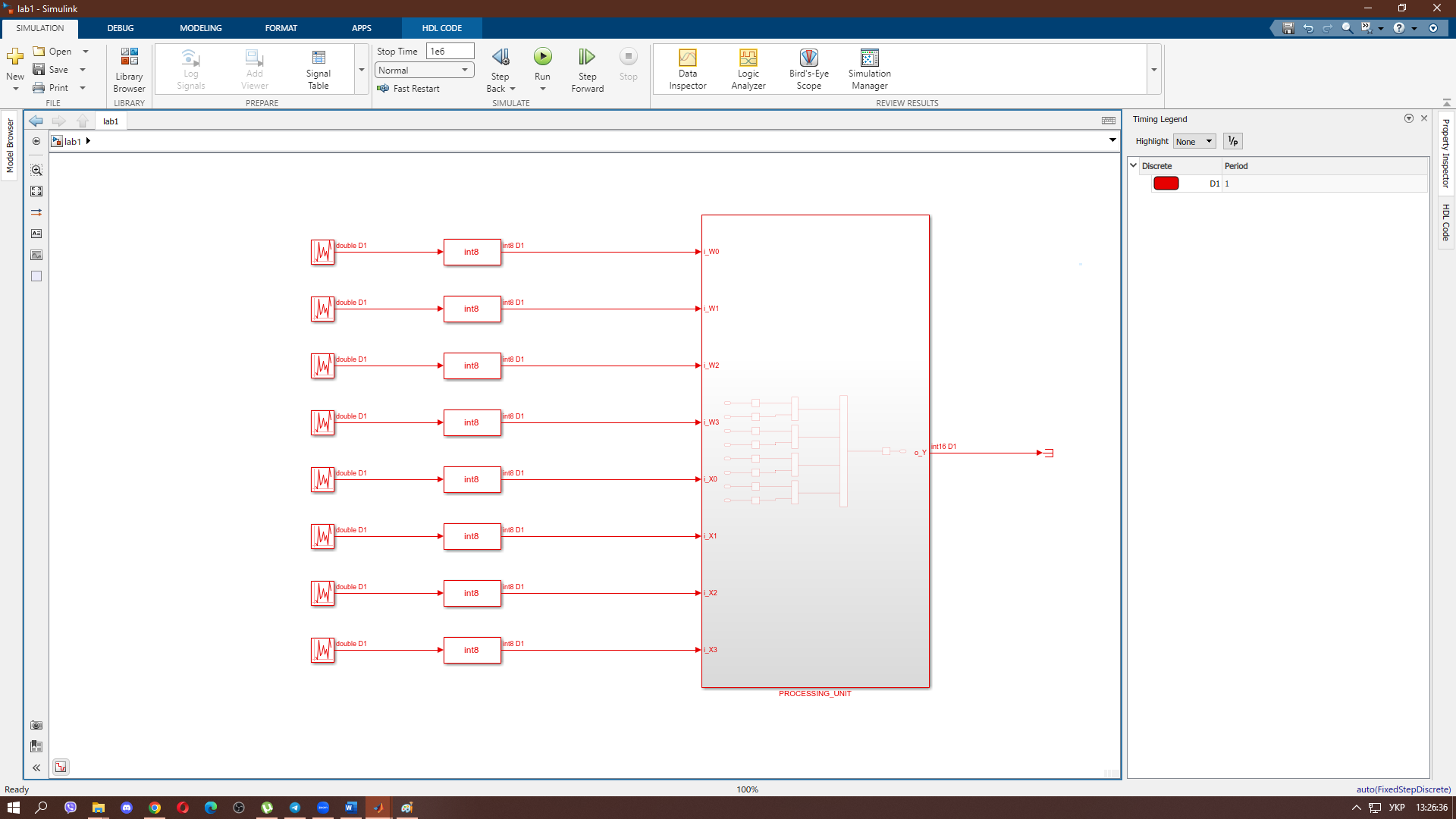
Тип даних виходу: int16

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

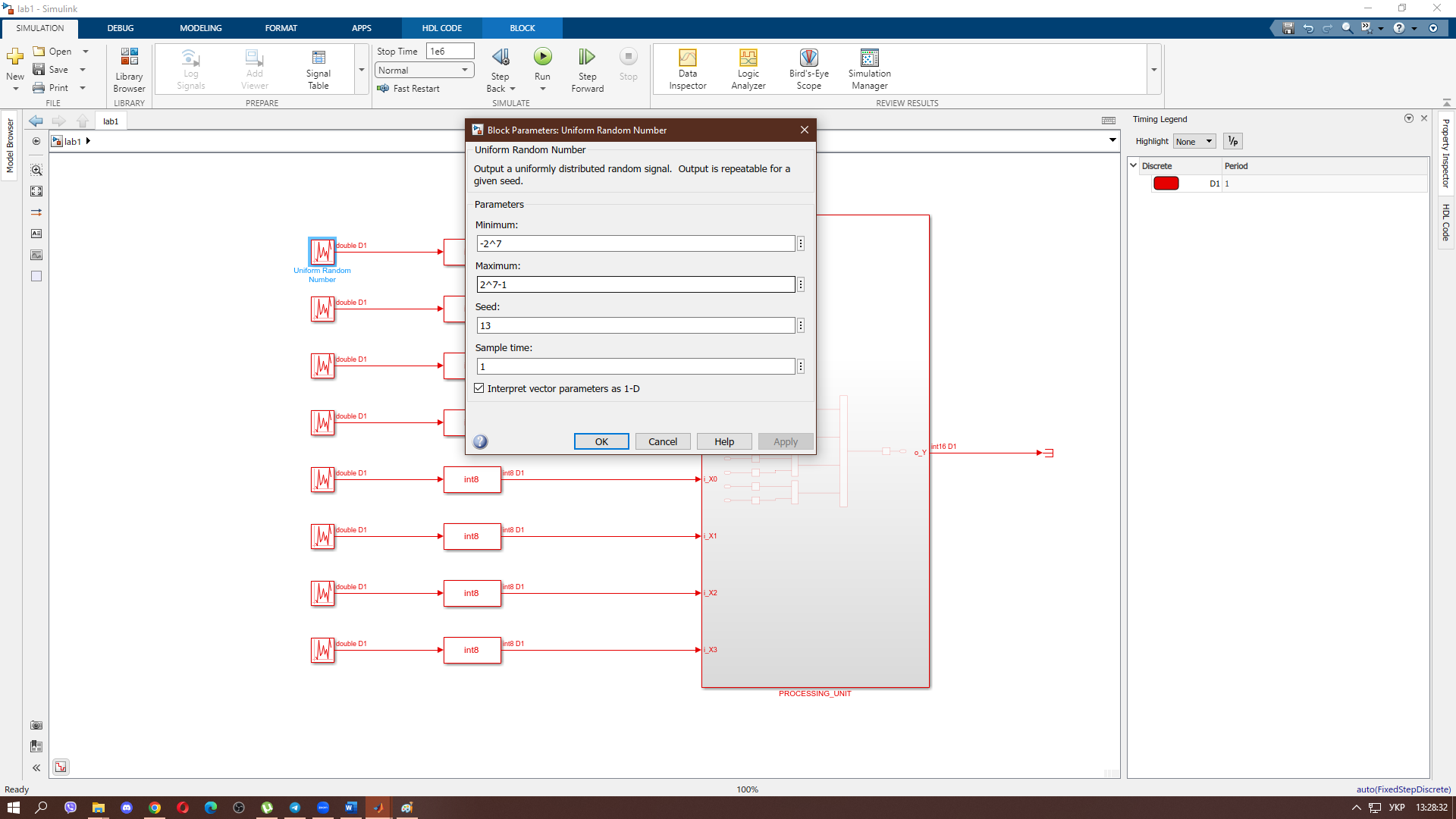
Можна назвати підсистему PROCESSING\_UNIT

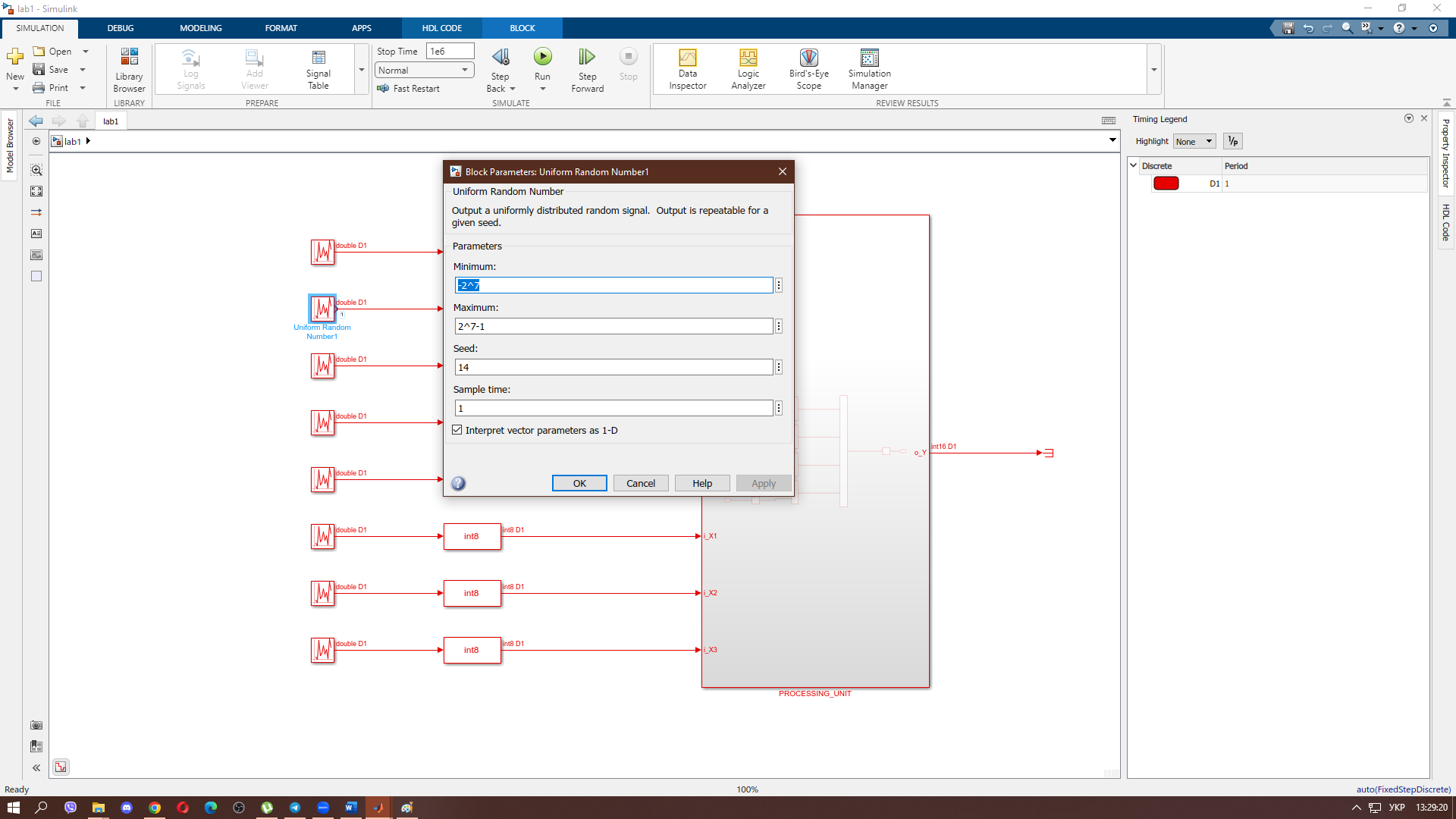
**Виконання**

Результат виконання завдання:

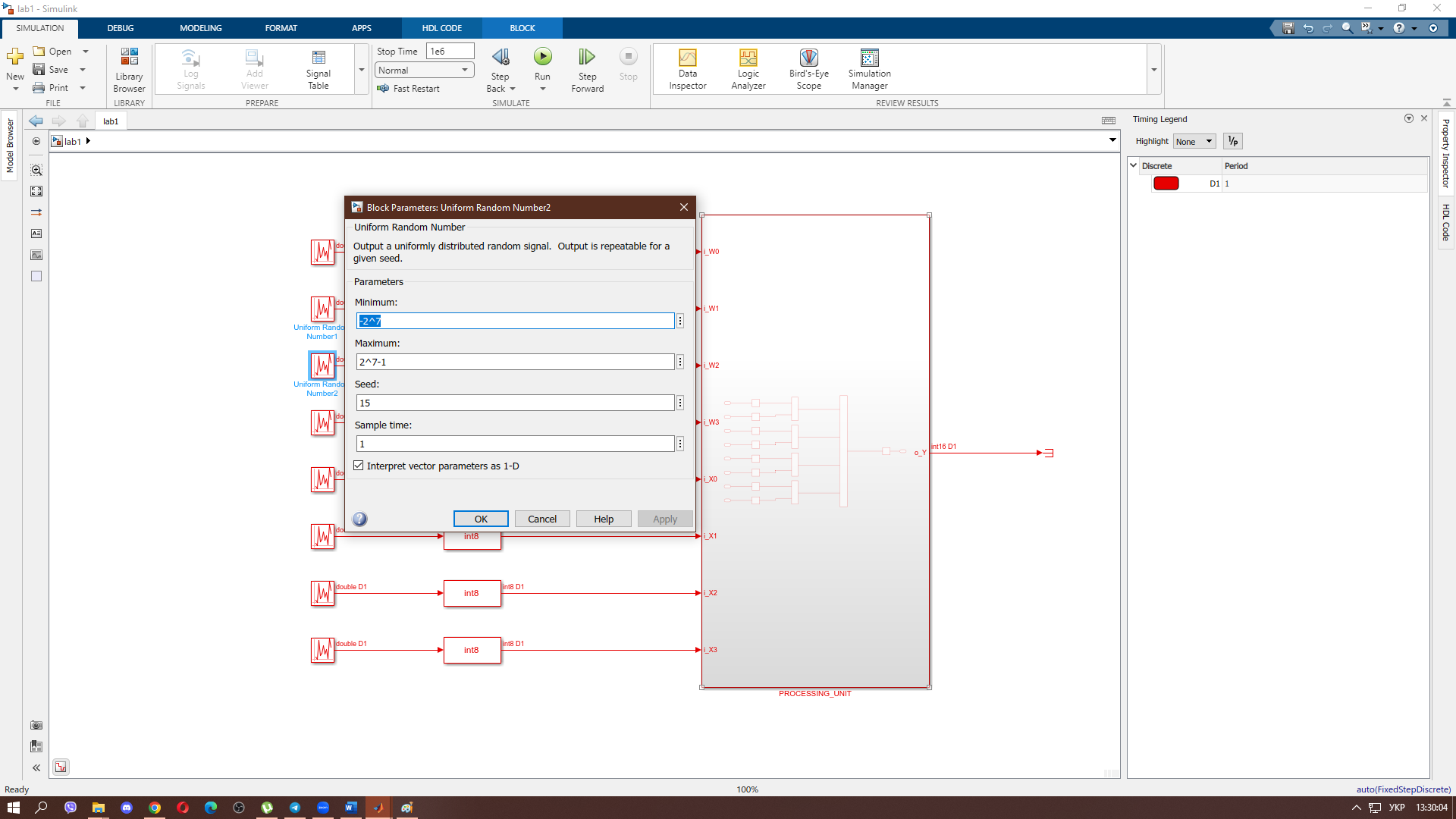


Налаштування Uniform Random Number:

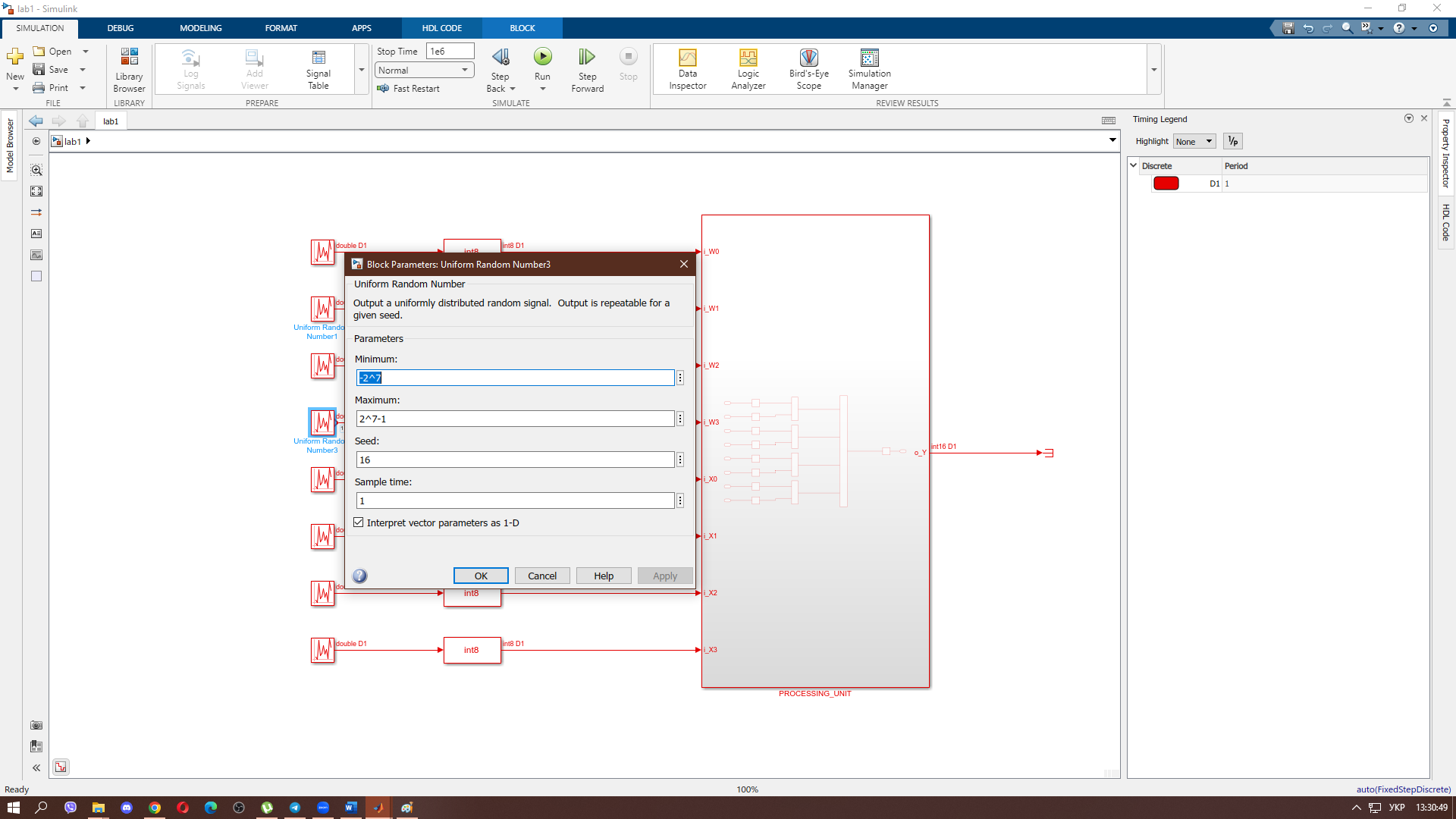
 - перший



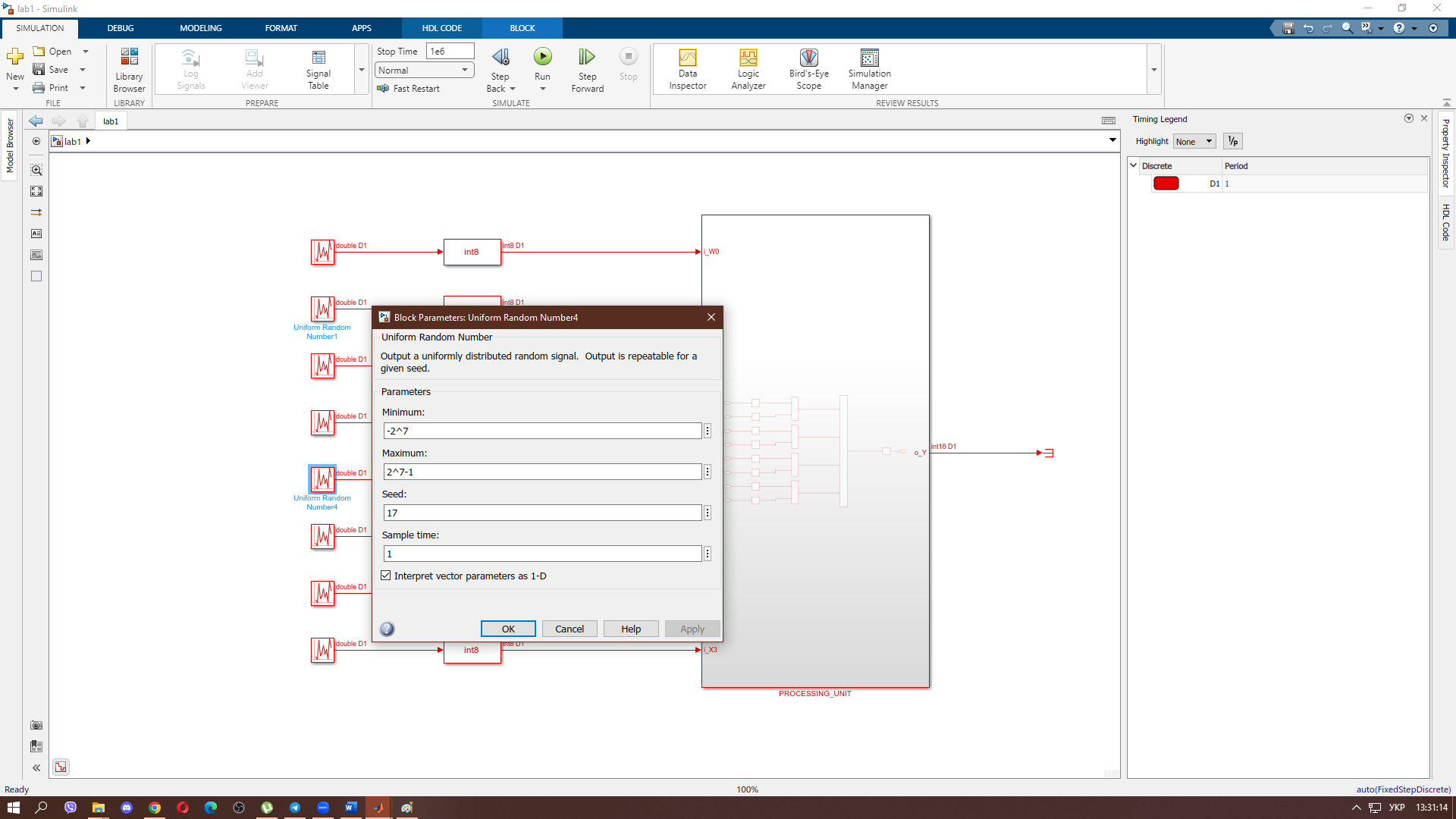
- другий



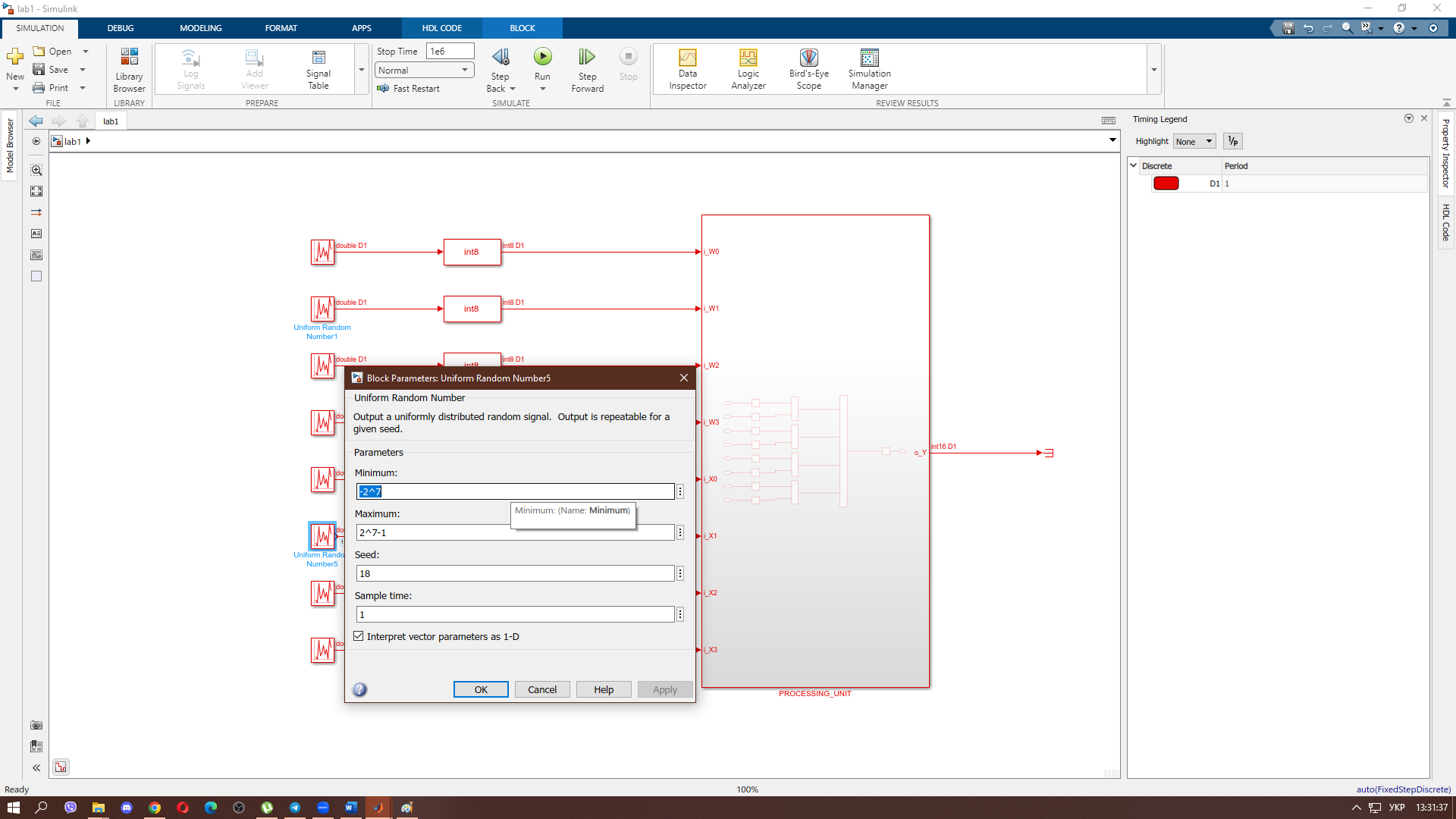
- третій



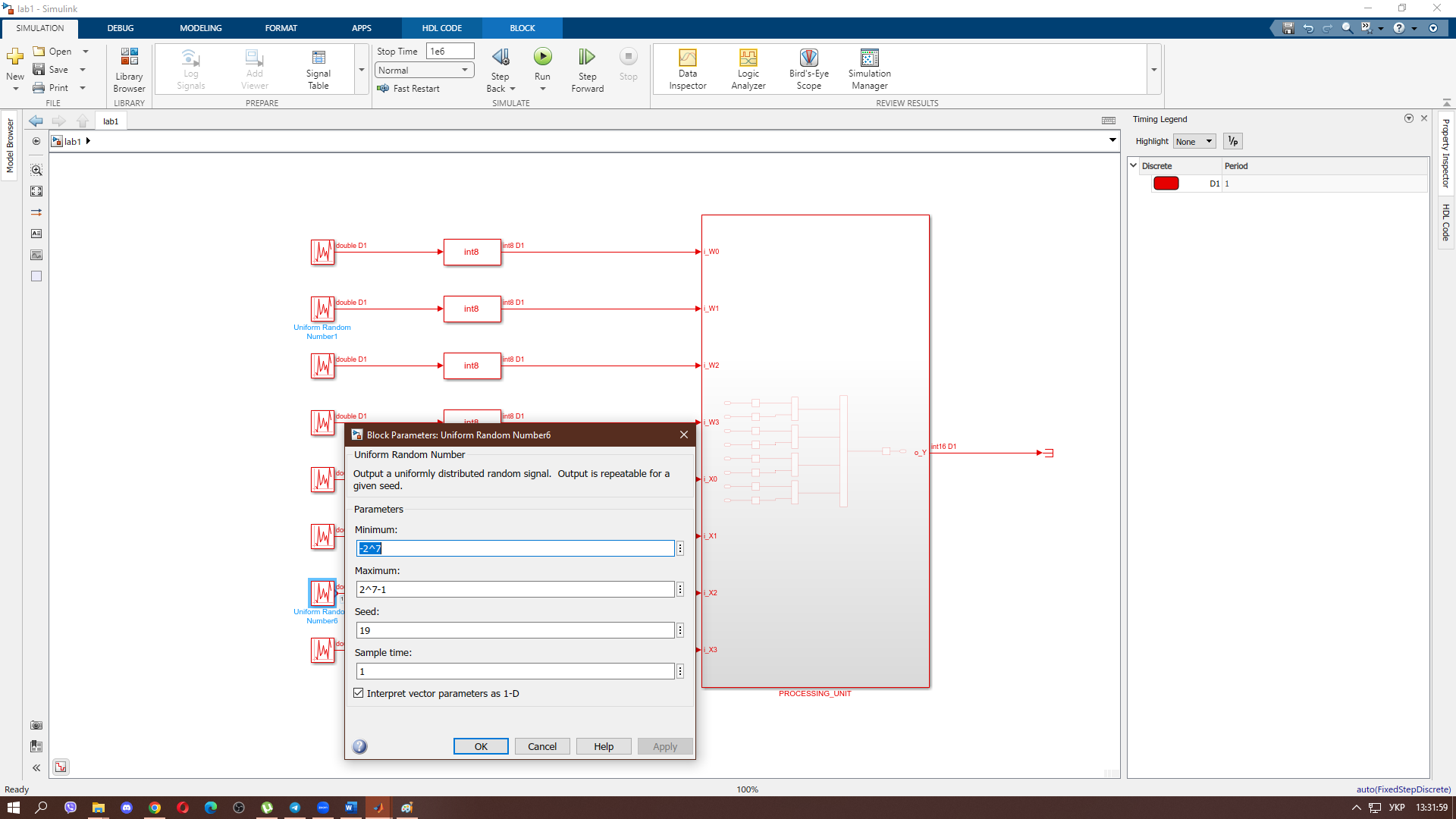
- четвертий



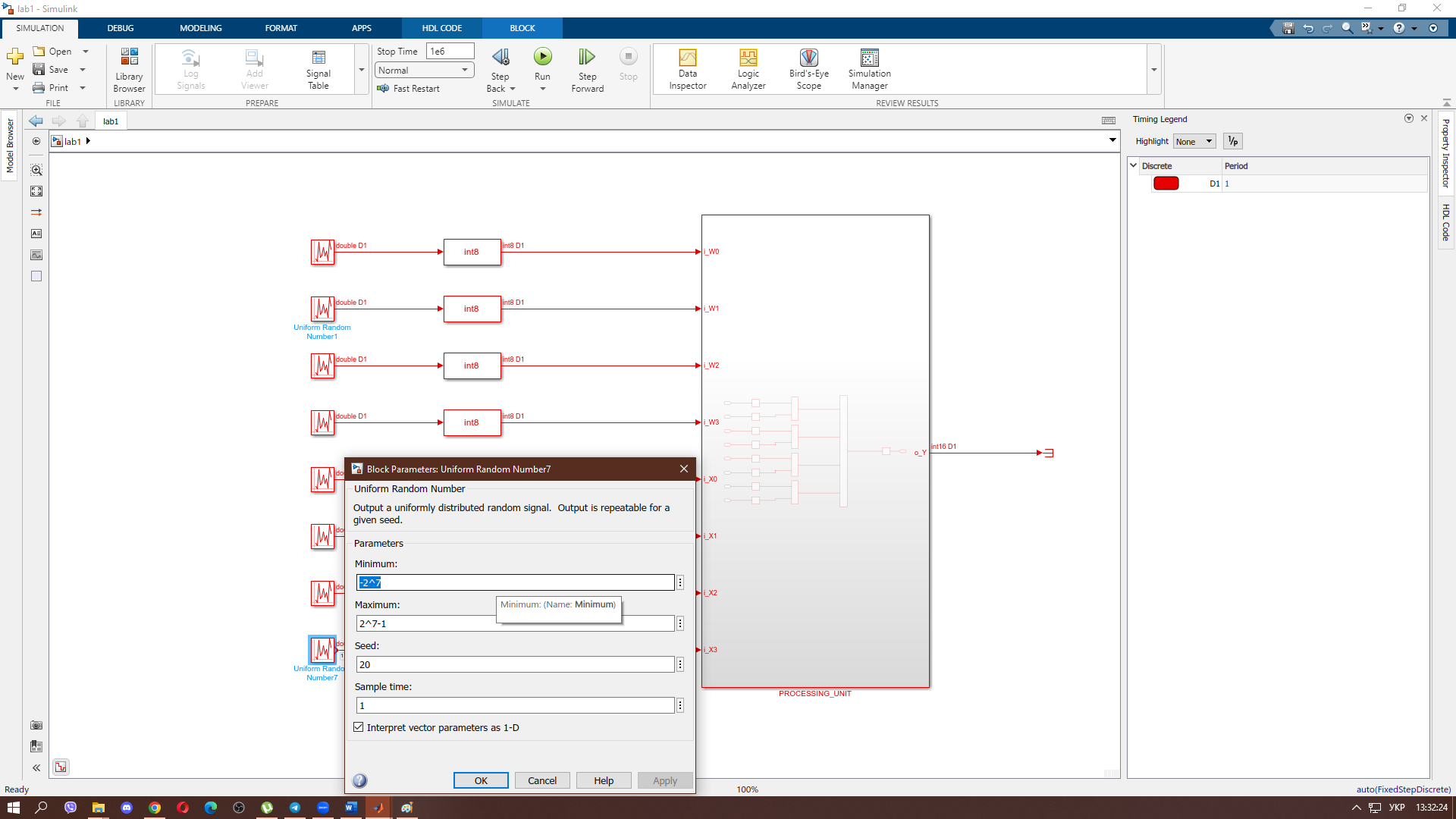
- п’ятий



- шостий

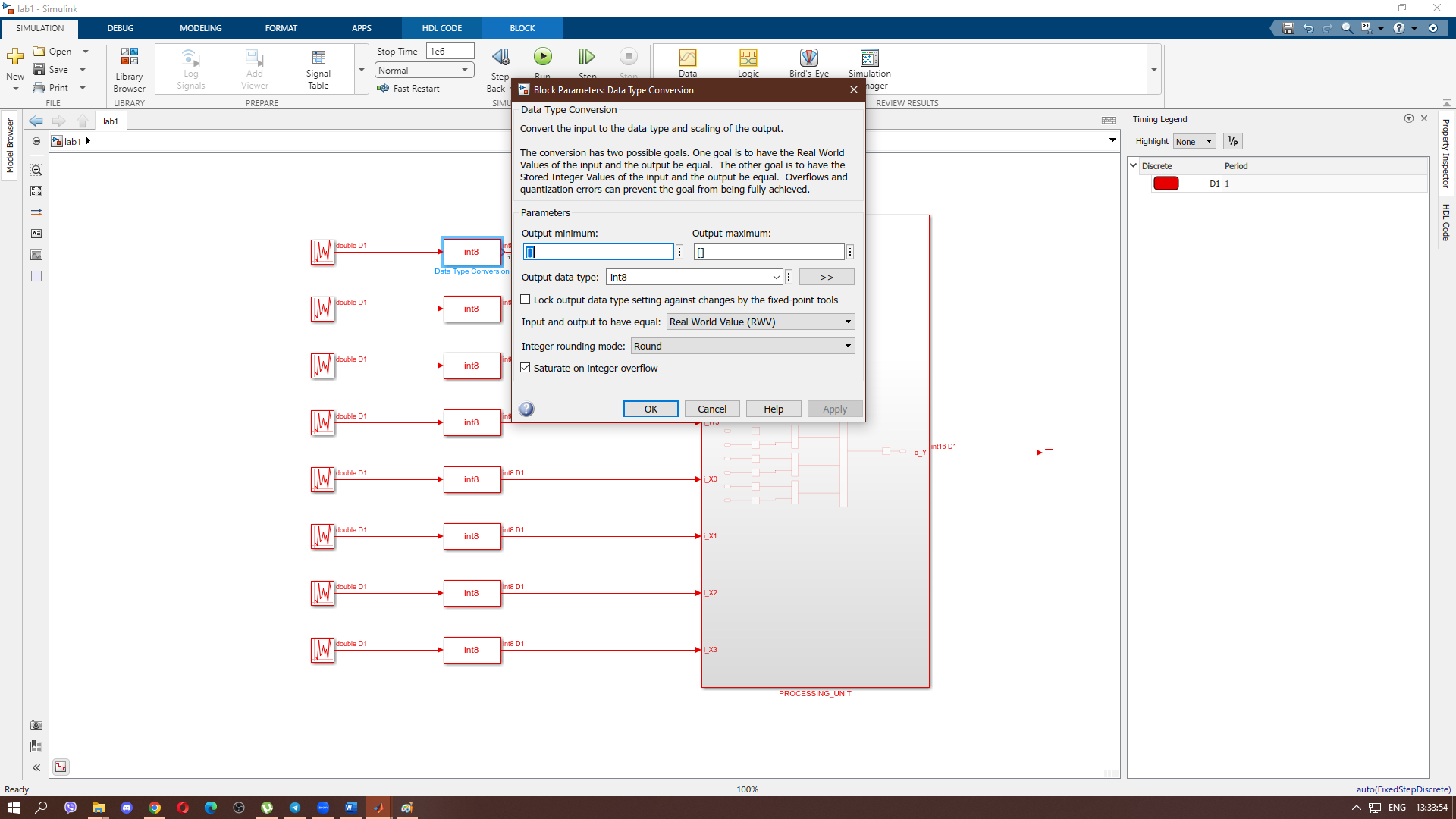


- сьомий



- восьмий

Налаштування Data Type Conversion:

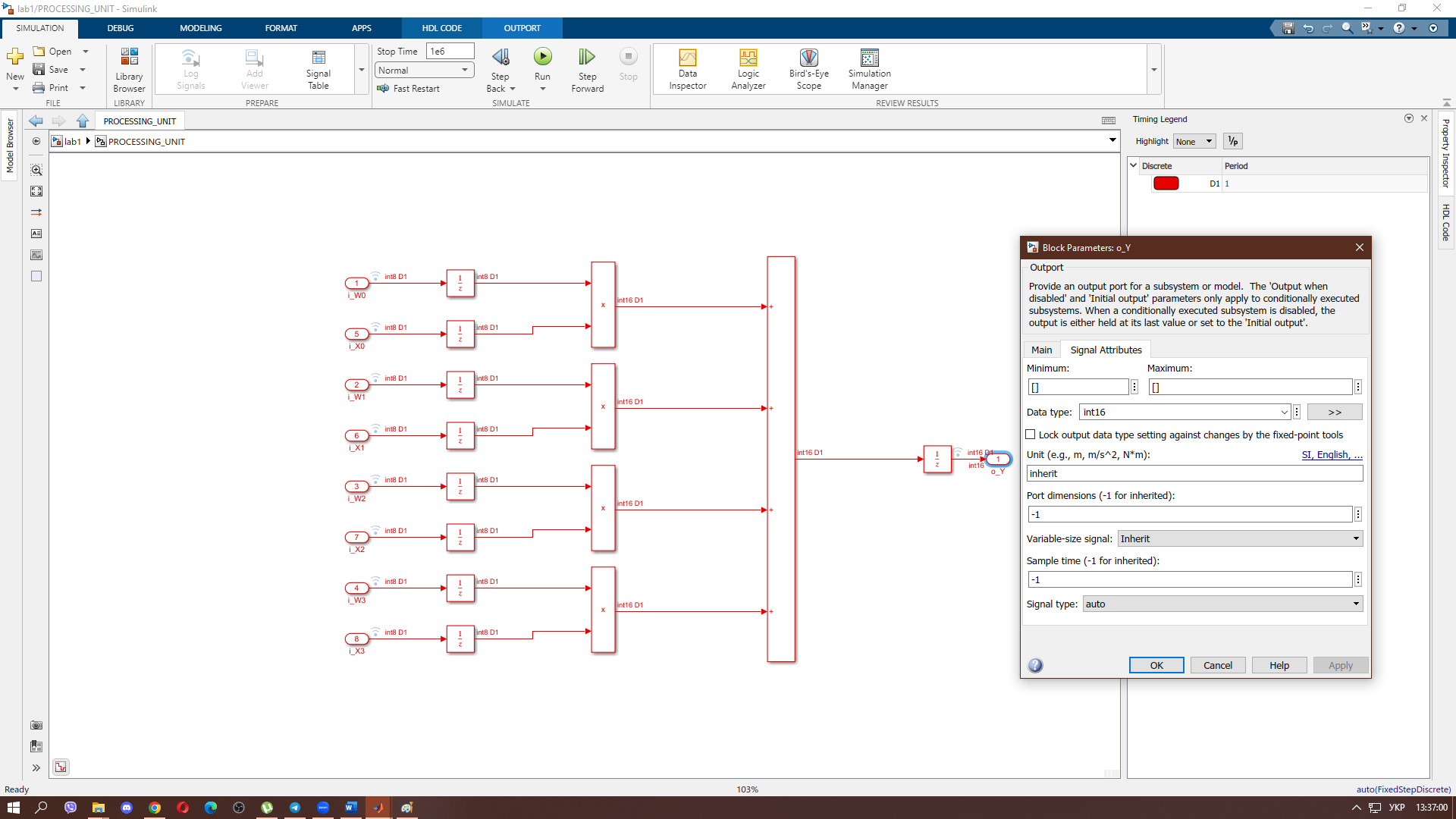


- такі налаштування у всіх восьми елементів

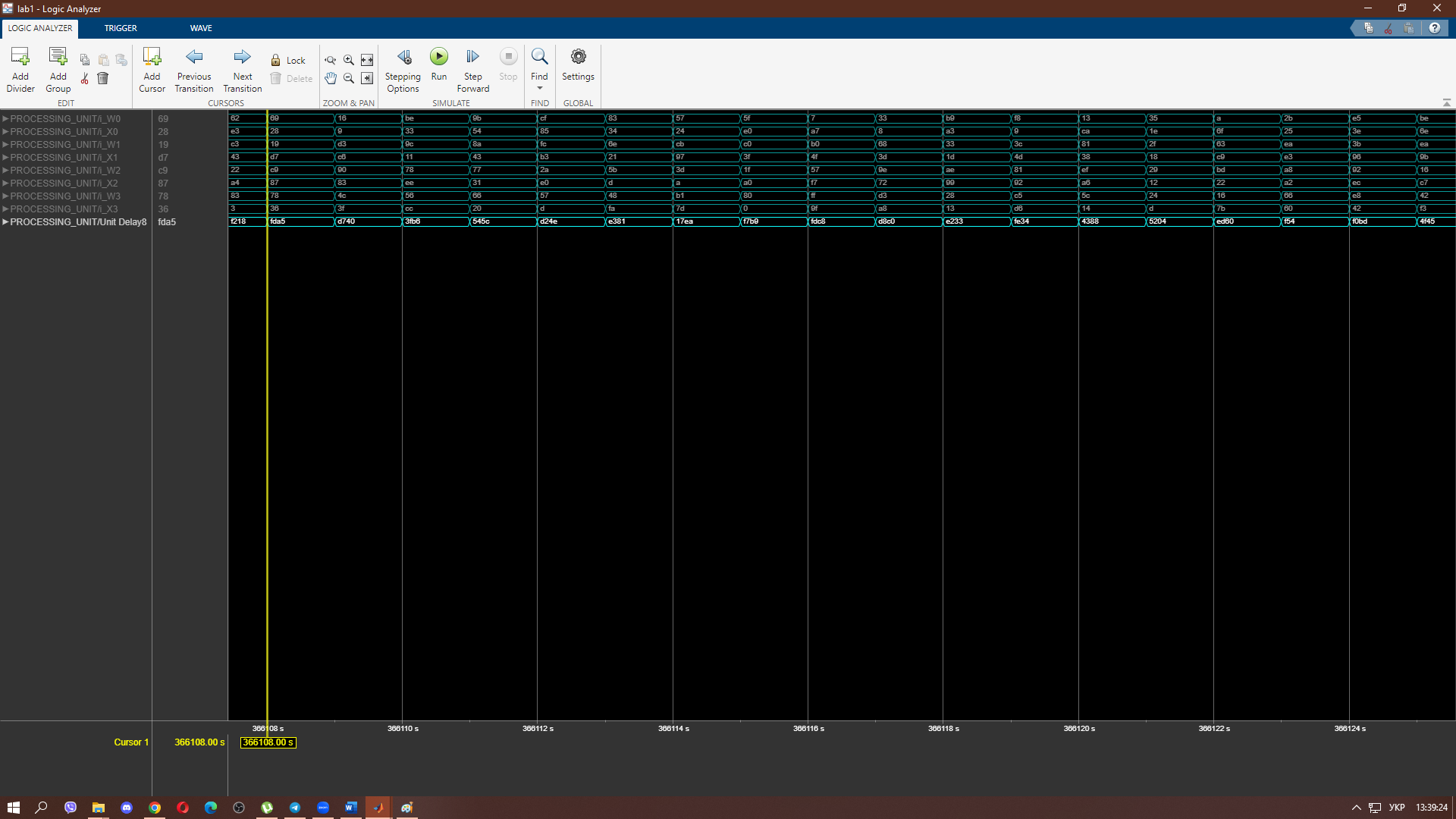
PROCESSING\_UNIT:



Налаштування виходу:



Результати у Logic Analyzer:

 Згенерований код для квартуса:

// -------------------------------------------------------------

//

// File Name: hdlsrc\lab1\PROCESSING\_UNIT.v

// Created: 2022-11-21 23:02:30

//

// Generated by MATLAB 9.12 and HDL Coder 3.20

//

//

// -- -------------------------------------------------------------

// -- Rate and Clocking Details

// -- -------------------------------------------------------------

// Model base rate: 1

// Target subsystem base rate: 1

//

//

// Clock Enable Sample Time

// -- -------------------------------------------------------------

// ce\_out 1

// -- -------------------------------------------------------------

//

//

// Output Signal Clock Enable Sample Time

// -- -------------------------------------------------------------

// o\_Y ce\_out 1

// -- -------------------------------------------------------------

//

// -------------------------------------------------------------

// -------------------------------------------------------------

//

// Module: PROCESSING\_UNIT

// Source Path: lab1/PROCESSING\_UNIT

// Hierarchy Level: 0

//

// -------------------------------------------------------------

`timescale 1 ns / 1 ns

module PROCESSING\_UNIT

(i\_CLK,

i\_RST\_N,

i\_CLK\_EN,

i\_W0,

i\_W1,

i\_W2,

i\_W3,

i\_X0,

i\_X1,

i\_X2,

i\_X3,

ce\_out,

o\_Y);

input i\_CLK;

input i\_RST\_N;

input i\_CLK\_EN;

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] Unit\_Delay\_out1; // int8

reg signed [7:0] Unit\_Delay2\_out1; // int8

reg signed [7:0] Unit\_Delay4\_out1; // int8

reg signed [7:0] Unit\_Delay6\_out1; // int8

reg signed [7:0] Unit\_Delay1\_out1; // int8

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

reg signed [7:0] Unit\_Delay3\_out1; // int8

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

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

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

reg signed [7:0] Unit\_Delay5\_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] Unit\_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] Unit\_Delay8\_out1; // int16

assign enb = i\_CLK\_EN;

always @(posedge i\_CLK or negedge i\_RST\_N)

begin : Unit\_Delay\_process

if (i\_RST\_N == 1'b0) begin

Unit\_Delay\_out1 <= 8'sb00000000;

end

else begin

if (enb) begin

Unit\_Delay\_out1 <= i\_W0;

end

end

end

always @(posedge i\_CLK or negedge i\_RST\_N)

begin : Unit\_Delay2\_process

if (i\_RST\_N == 1'b0) begin

Unit\_Delay2\_out1 <= 8'sb00000000;

end

else begin

if (enb) begin

Unit\_Delay2\_out1 <= i\_W1;

end

end

end

always @(posedge i\_CLK or negedge i\_RST\_N)

begin : Unit\_Delay4\_process

if (i\_RST\_N == 1'b0) begin

Unit\_Delay4\_out1 <= 8'sb00000000;

end

else begin

if (enb) begin

Unit\_Delay4\_out1 <= i\_W2;

end

end

end

always @(posedge i\_CLK or negedge i\_RST\_N)

begin : Unit\_Delay6\_process

if (i\_RST\_N == 1'b0) begin

Unit\_Delay6\_out1 <= 8'sb00000000;

end

else begin

if (enb) begin

Unit\_Delay6\_out1 <= i\_W3;

end

end

end

always @(posedge i\_CLK or negedge i\_RST\_N)

begin : Unit\_Delay1\_process

if (i\_RST\_N == 1'b0) begin

Unit\_Delay1\_out1 <= 8'sb00000000;

end

else begin

if (enb) begin

Unit\_Delay1\_out1 <= i\_X0;

end

end

end

assign Product\_out1 = Unit\_Delay\_out1 \* Unit\_Delay1\_out1;

always @(posedge i\_CLK or negedge i\_RST\_N)

begin : Unit\_Delay3\_process

if (i\_RST\_N == 1'b0) begin

Unit\_Delay3\_out1 <= 8'sb00000000;

end

else begin

if (enb) begin

Unit\_Delay3\_out1 <= i\_X1;

end

end

end

assign Product1\_out1 = Unit\_Delay2\_out1 \* Unit\_Delay3\_out1;

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

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

always @(posedge i\_CLK or negedge i\_RST\_N)

begin : Unit\_Delay5\_process

if (i\_RST\_N == 1'b0) begin

Unit\_Delay5\_out1 <= 8'sb00000000;

end

else begin

if (enb) begin

Unit\_Delay5\_out1 <= i\_X2;

end

end

end

assign Product2\_out1 = Unit\_Delay4\_out1 \* Unit\_Delay5\_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 i\_CLK or negedge i\_RST\_N)

begin : Unit\_Delay7\_process

if (i\_RST\_N == 1'b0) begin

Unit\_Delay7\_out1 <= 8'sb00000000;

end

else begin

if (enb) begin

Unit\_Delay7\_out1 <= i\_X3;

end

end

end

assign Product3\_out1 = Unit\_Delay6\_out1 \* Unit\_Delay7\_out1;

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

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

always @(posedge i\_CLK or negedge i\_RST\_N)

begin : Unit\_Delay8\_process

if (i\_RST\_N == 1'b0) begin

Unit\_Delay8\_out1 <= 16'sb0000000000000000;

end

else begin

if (enb) begin

Unit\_Delay8\_out1 <= Add\_out1;

end

end

end

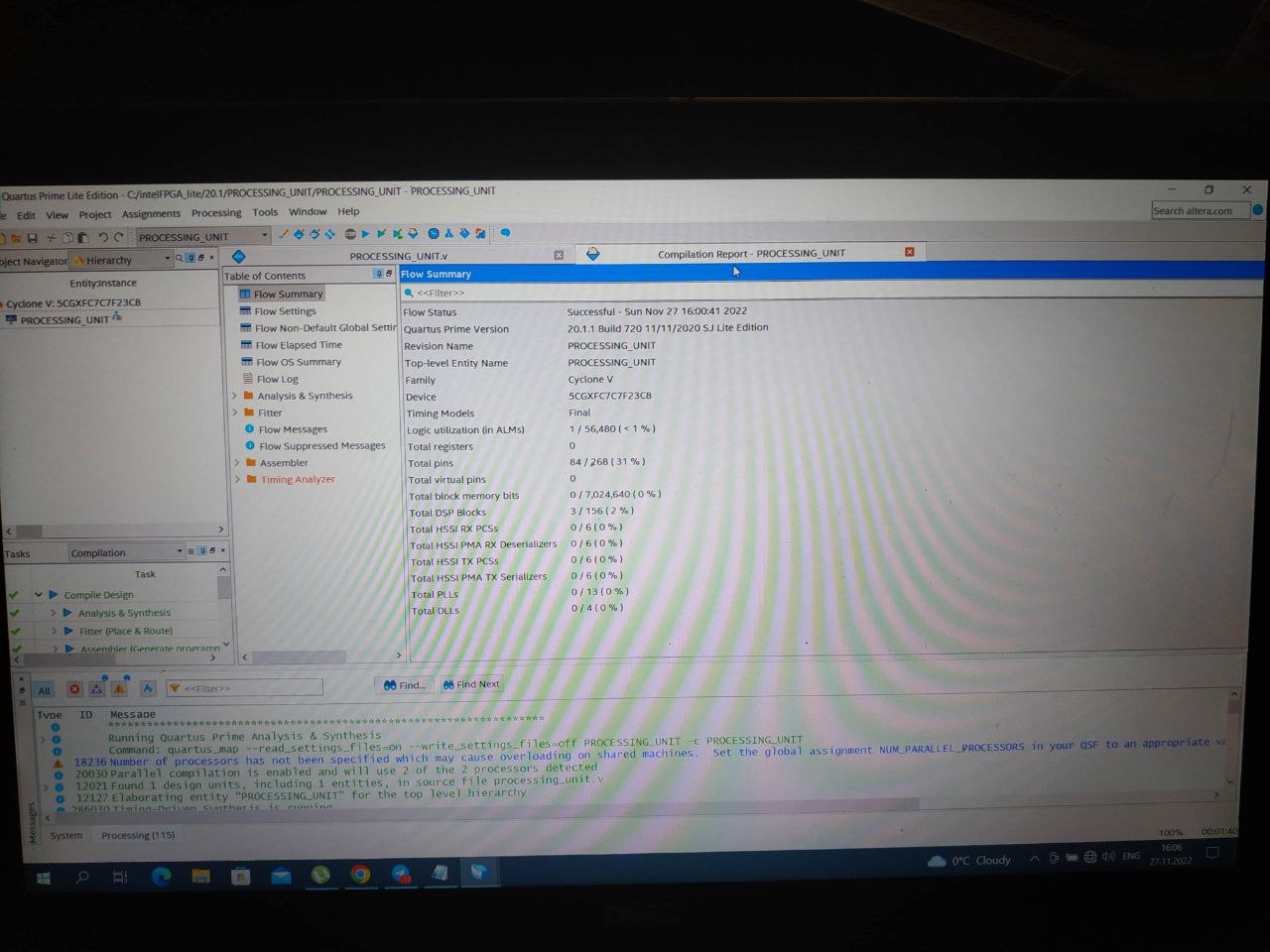
assign o\_Y = Unit\_Delay8\_out1;

assign ce\_out = i\_CLK\_EN;

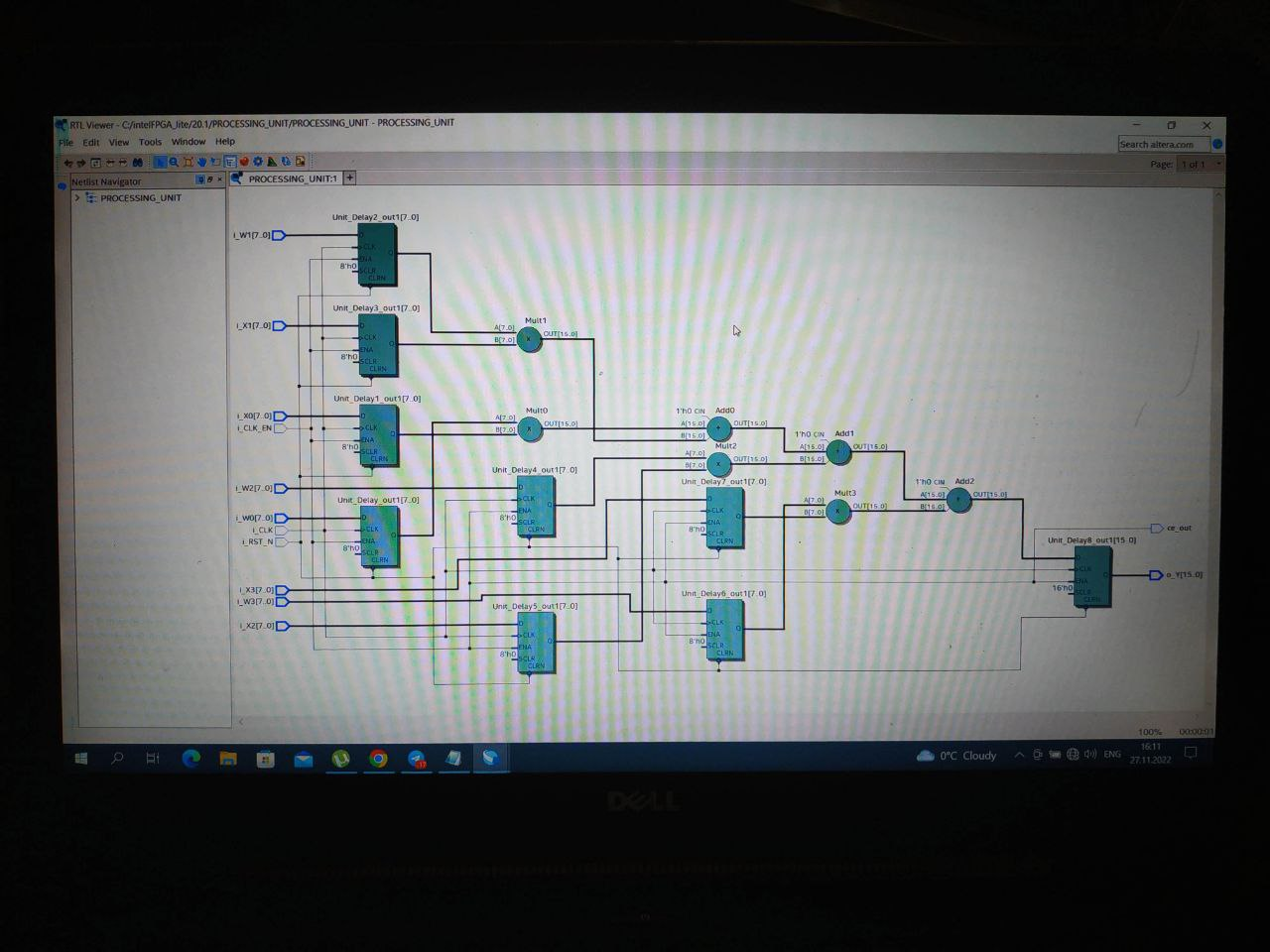
endmodule // PROCESSING\_UNIT

Результат синтезу згенерованого коду в Quartus для створеної підсистеми

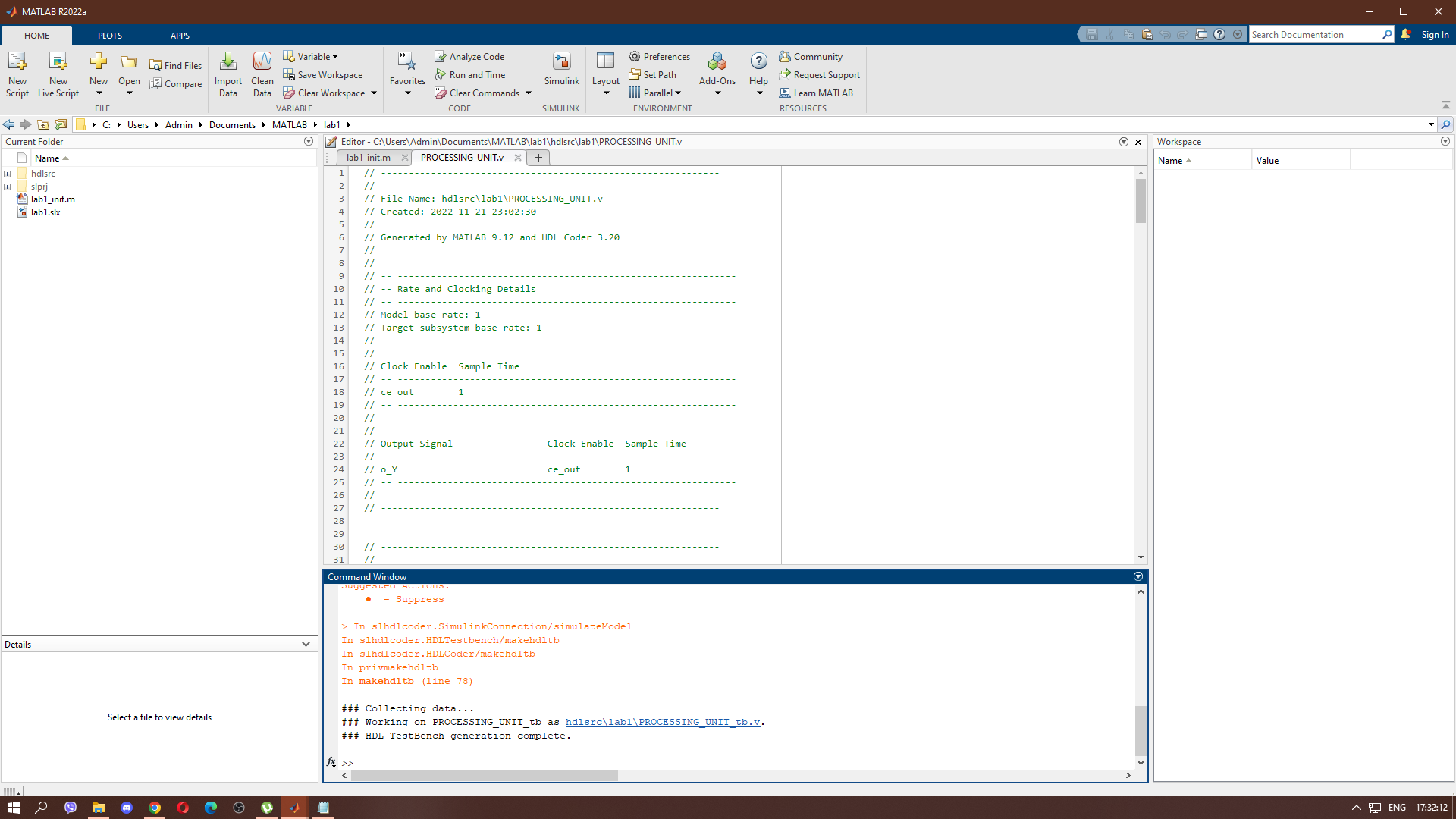
Звіт по апаратним витратам:



Результат виклику RTL Viewer:



Тестбенч:



Висновок: у ході лабораторної роботи, мені вдалося реалізувати програму, яка реалізує функцію Y = W0\*X0 + W1\*X1 + W2\*X2 + W\*X3. Виставив в матлабі такі налаштування, які були вказані в методичці, створив програму, підсистему, протестував її в logic analyzer, все спрацювало, програма працює нормально, також був протестований код верілог у квартусі, який згенерував матлаб, в результаті все спрацювало, і RTL Viewer показав схожу схему, також був успішно сроблений тестбенч у матлабі.