

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
2. Перегляд в логічному аналізаторі даних на входах і на виході створеної підсистеми у знаковому десятковому поданні.....	8
3. Генерування коду на Verilog та синтез згенерованого коду в Quartus для створеної підсистеми	9
4.Результат синтезу в RTL Viewer у квартусі та визначення апаратних витрат:	16
5. Створення тестбенч файлу в Matlab для створеної підсистеми і результат симуляції тестбенча в Modelsim	18
Висновок.....	20

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:

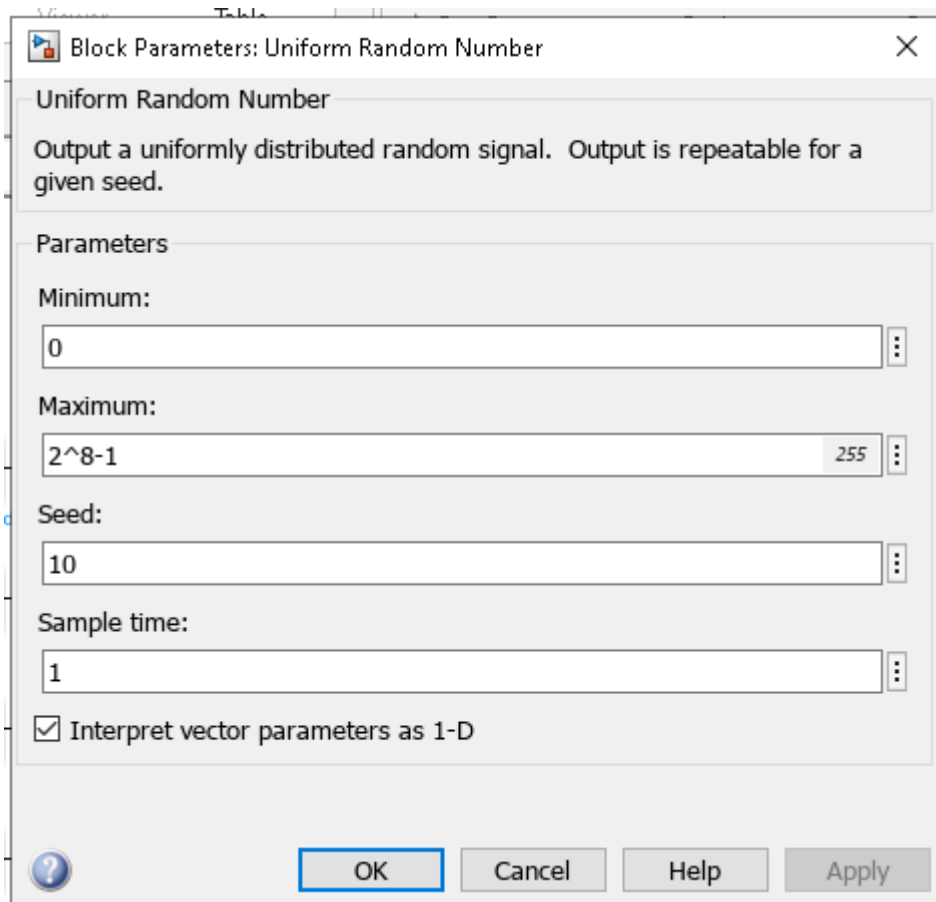


Рис. 1 Налаштування першого по порядку блоку

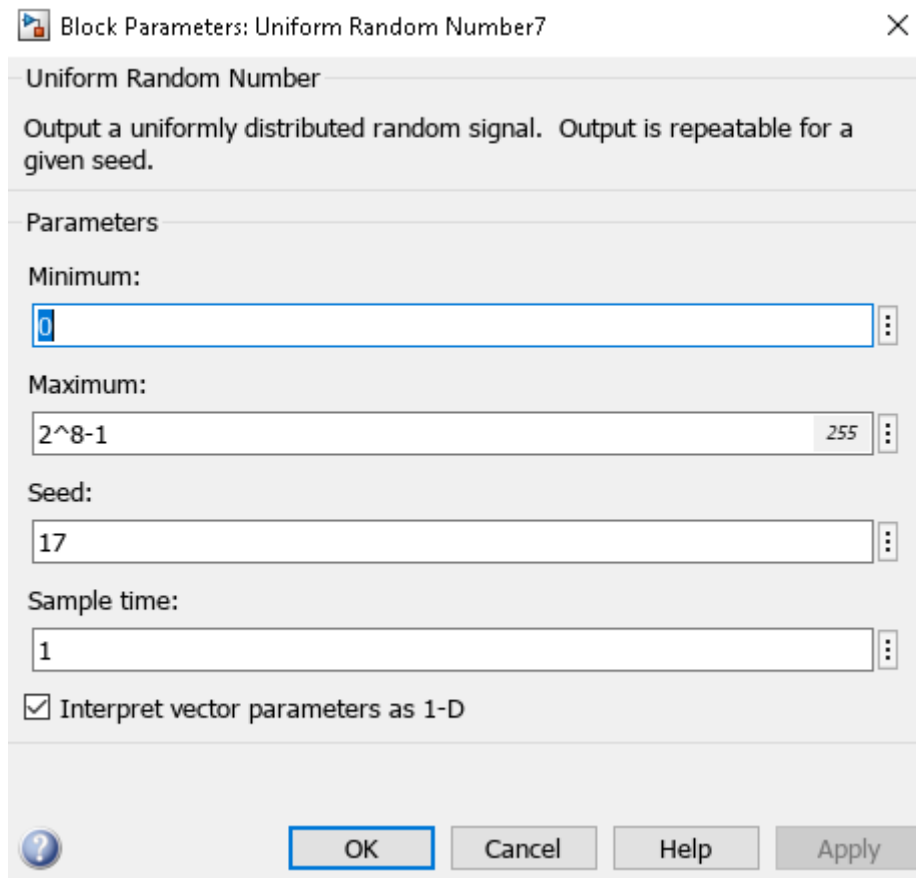


Рис. 2 Налаштування останнього по порядку блоку

Створюємо в Simulink загальну схему згідно умові:

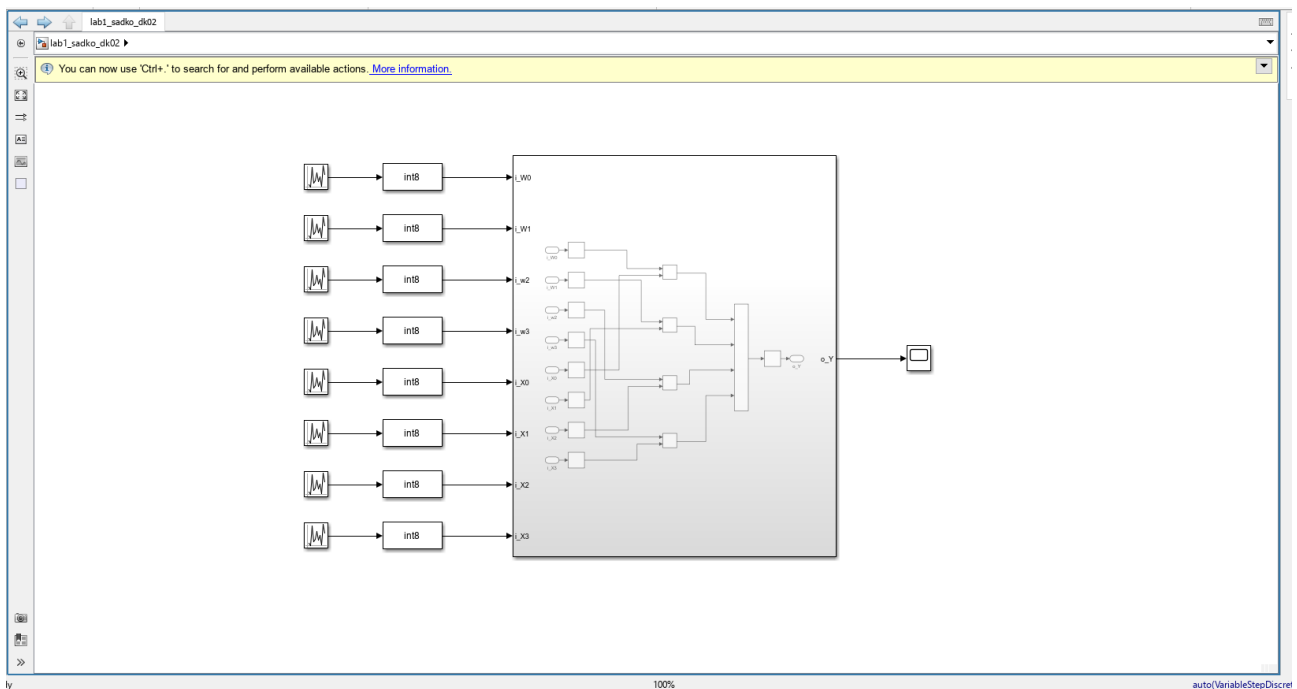


Рис. 3 Загальна схема

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

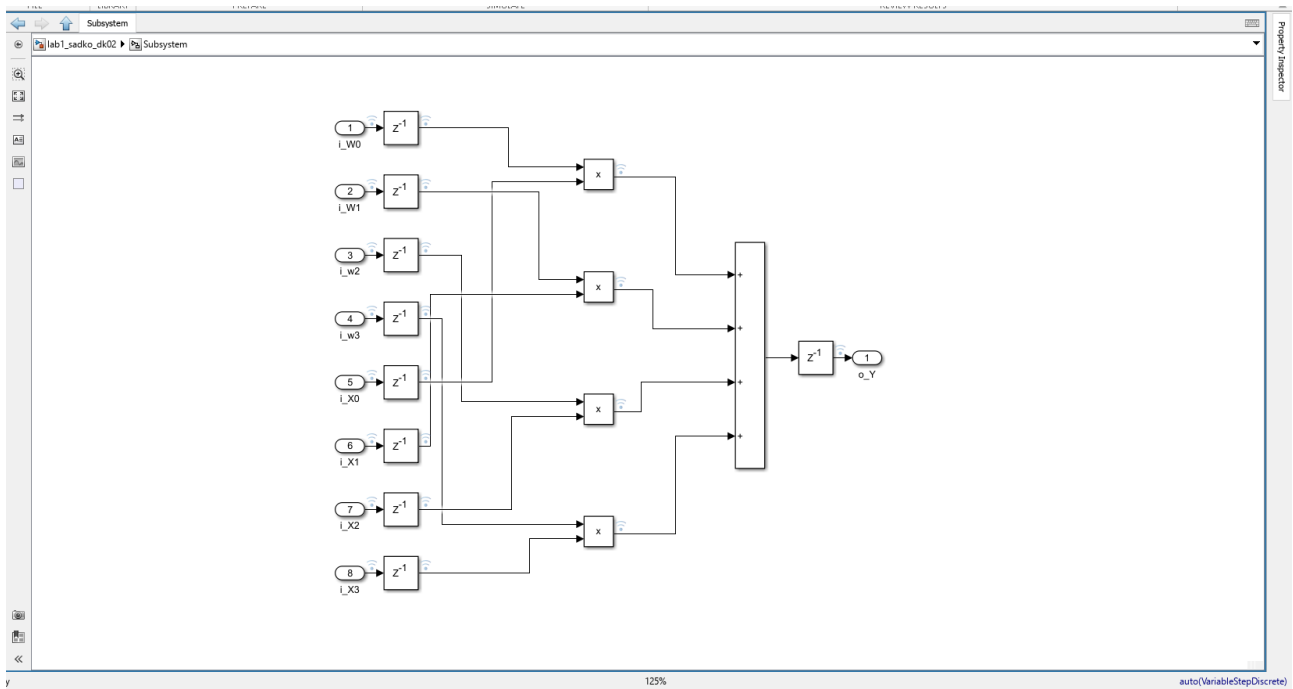


Рис. 4 Підсистема загальної схеми

Налаштування компонентів підсистеми:

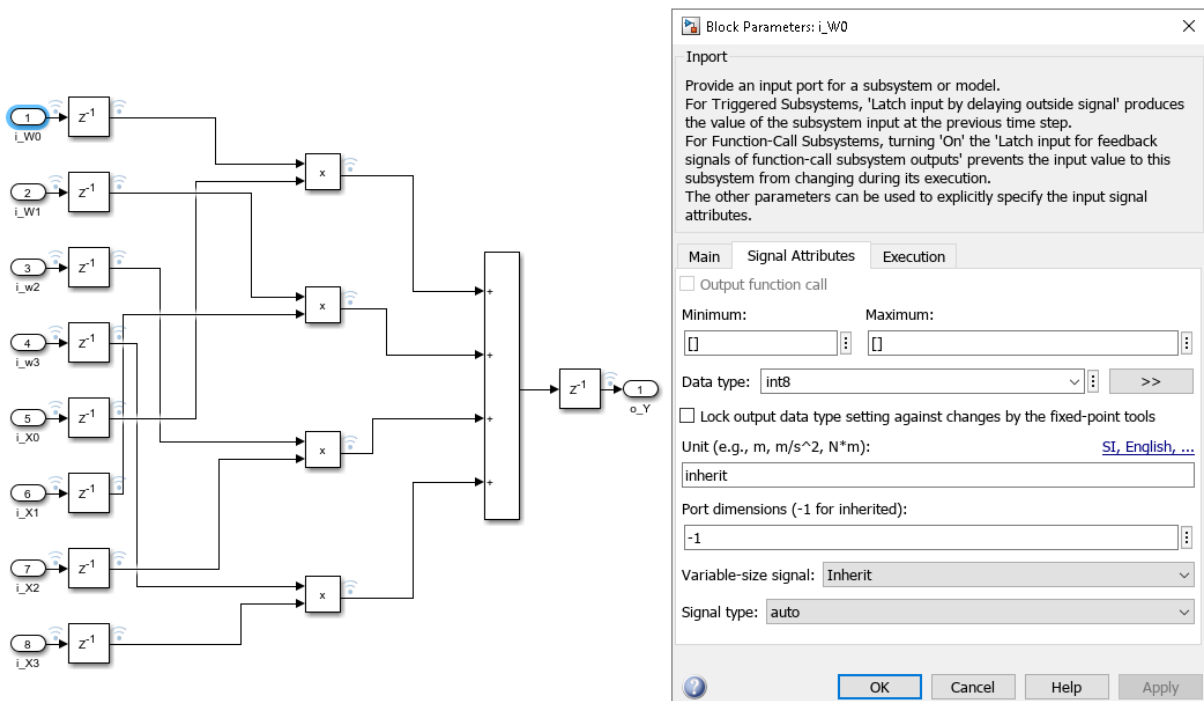


Рис. 5 Налаштування для входів i_W

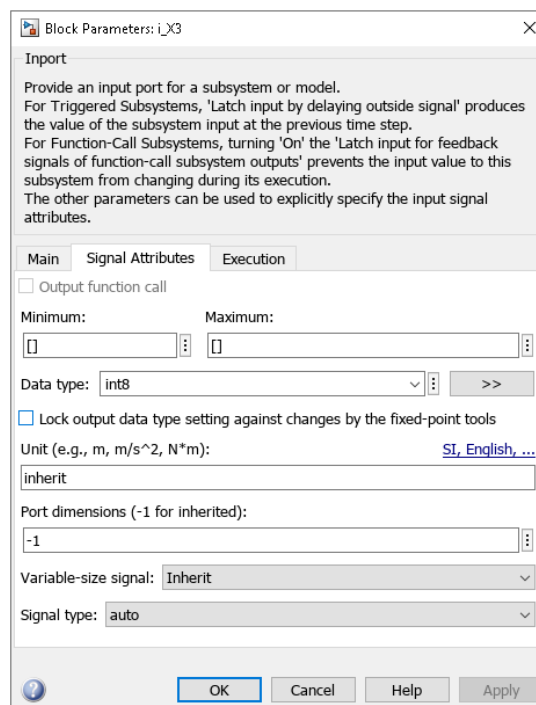
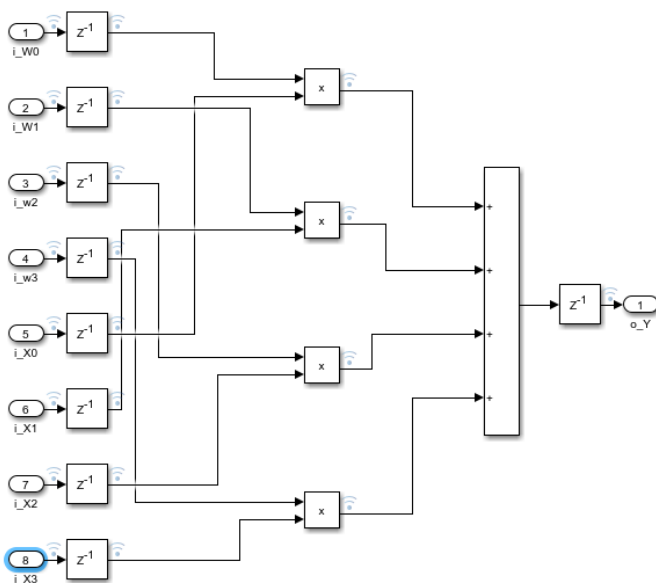


Рис. 6 Налаштування для входів i_X

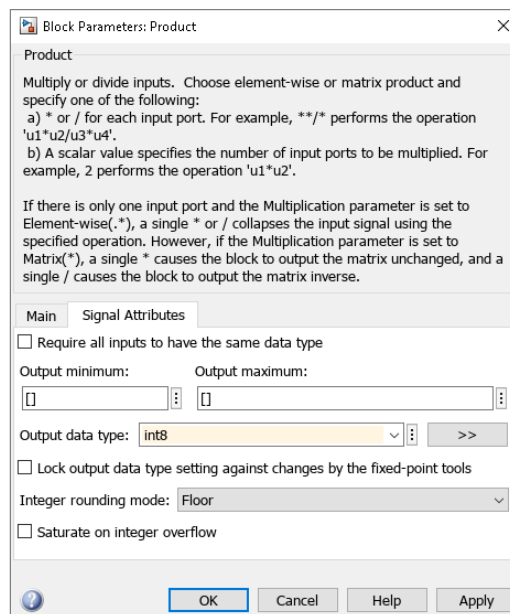
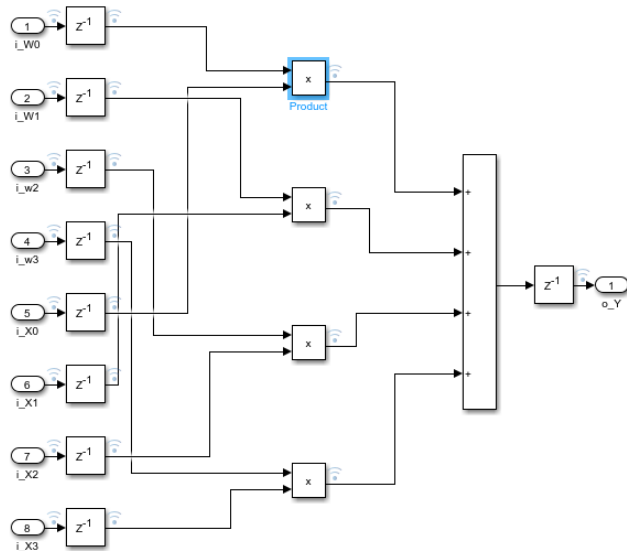


Рис. 7 Налаштування для блоків Product

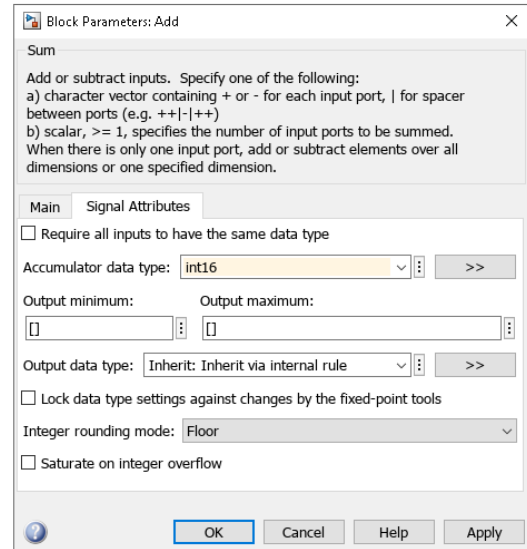
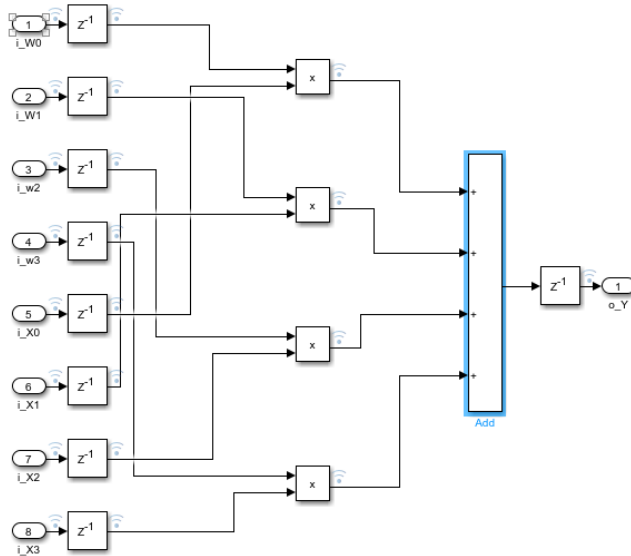


Рис. 8 Налаштування для блоків Add

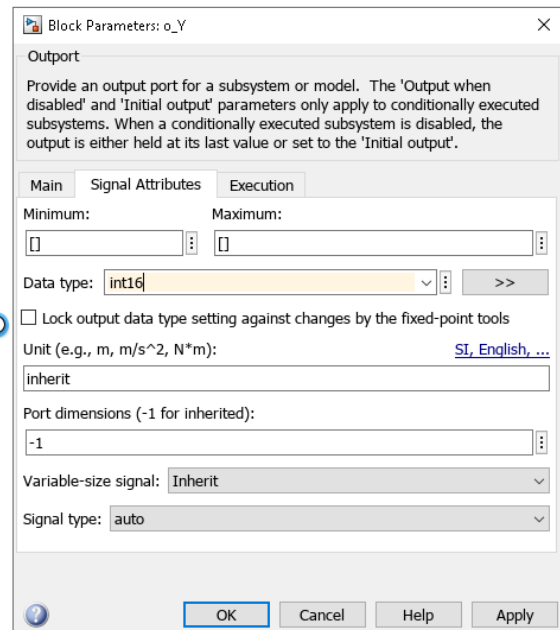
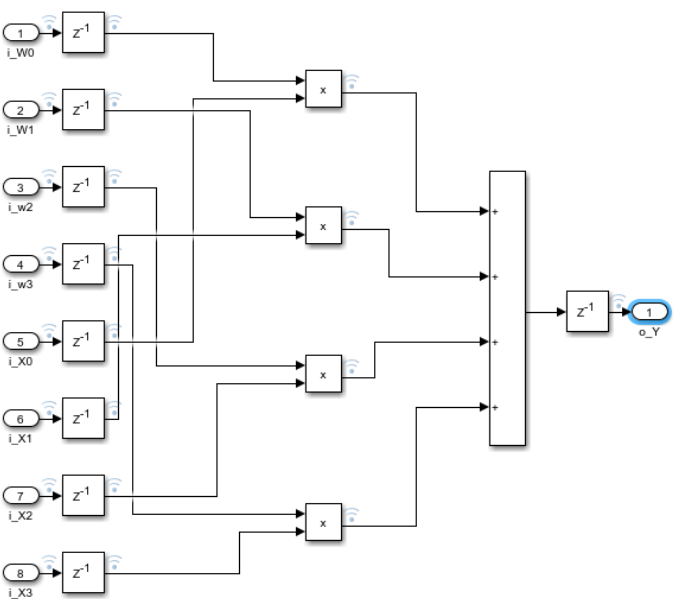


Рис. 9 Налаштування для виходу o_Y

2. Перегляд в логічному аналізаторі 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.

Signal	Value	Signal	Value	Signal	Value	Signal	Value	Signal	Value	Signal	Value	Signal	Value	Signal	Value	Signal	Value	Signal	Value
Subsystem_V0	-22	32	-102	-128	-14	85	23	40	57	80	88								
Subsystem_Delay0	-8262	0	15582	-11527	5344	5988	2046	5354	7419	2507									
Subsystem_V1	28	32	16	32	11	60	62	70	37	16	94								
Subsystem_w2	77	32	-122	62	35	62	-126	98	-18	102									
Subsystem_w3	126	38	-5	-115	80	94	56	-127	3	82	111								
Subsystem_V0	-81	48	113	-26	84	18	-20	27	23	43	120								
Subsystem_X1	-32	30	-25	65	108	-128	34	58	43	-119	-128								
Subsystem_X2	18	62	52	-102	-124	-17	58	-39	83	25	-119								
Subsystem_X3	87	32	-25	13	32	94	14	3	83	10	181								
Subsystem_Delay	85	0	22	-102	-138	-14	85	23	40	57	80								
Subsystem_Delay1	94	0	-93	16	-39	11	-20	-52	70	-37	16								
Subsystem_Delay2	102	0	39	-122	62	35	62	-126	98	-18	109								
Subsystem_Delay7	-111	0	-73	-46	-13	-99	94	14	9	83	70								
Subsystem_Delay2	111	0	-38	-5	-135	60	24	58	-127	3	-52								
Subsystem_Delay4	120	0	18	113	35	84	18	32	27	23	43								
Subsystem_Delay5	-128	0	-80	-25	65	108	-128	34	-68	43	-119								
Subsystem_Delay6	-119	0	62	93	-102	-124	-17	88	-38	83	25								
Subsystem_Product	-40	0	-84	-5	0	104	-5	62	-40	-31	-112								
Subsystem_Product1	-12032	0	7340	-400	-2535	1188	7660	4888	-4760	-1291	-1785								
Subsystem_Product2	-12138	0	2038	-11336	-5304	-4380	-484	-11088	3762	-11154	2225								
Subsystem_Product3	-12321	0	6278	225	1485	2690	3838	784	1143	248	3840								

Рис. 10 Результат симуляції схеми

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

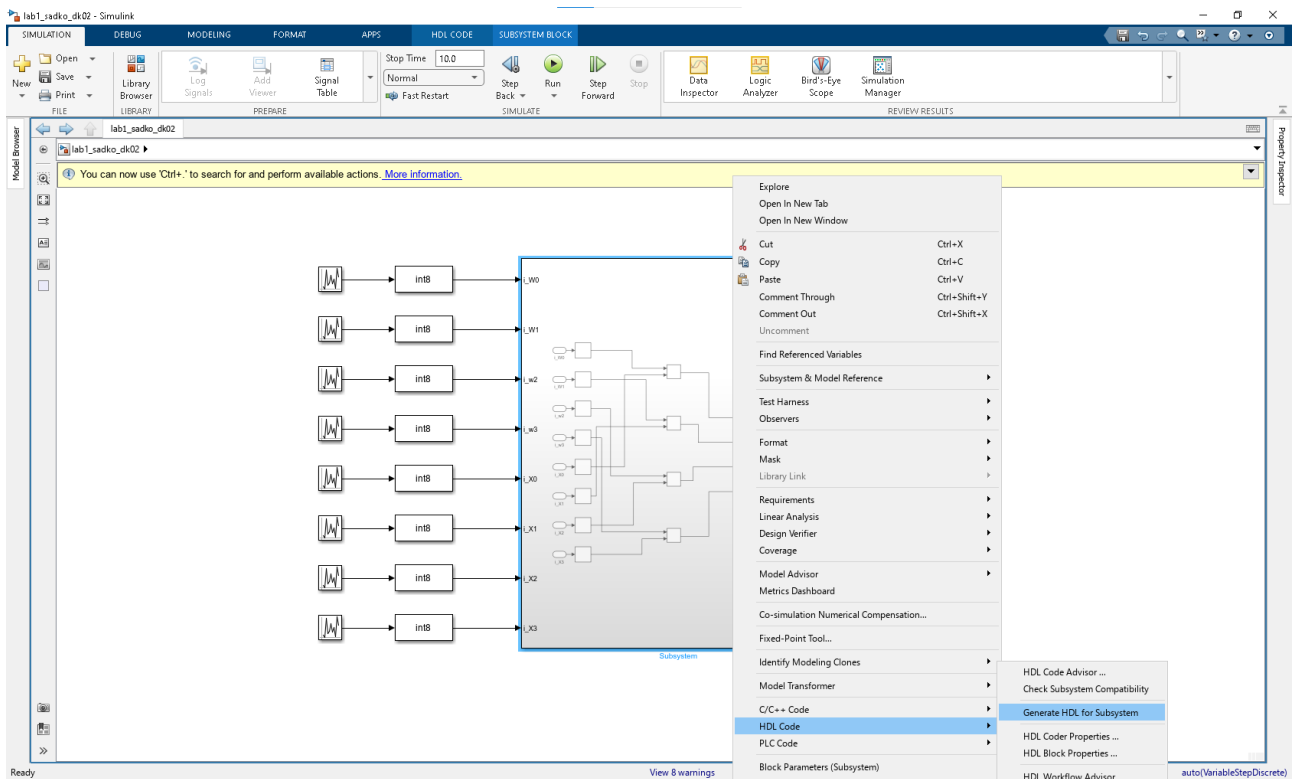


Рис. 11 Генерація коду на Verilog для підсистеми

Лістинг 3.1

```
// коментарі прибрав, але вони, як і сам .v файл будуть у директорії, де розміщенні  
будуть всі файли проекту  
  
`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;  // sfixed16

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; // sfixed16

wire signed [15:0] Add_stage2_1; // sfixed16

wire signed [16:0] Add_op_stage1; // sfixed17

reg signed [7:0] Delay6_out1; // int8

wire signed [15:0] Product2_out1; // int16

wire signed [15:0] Add_stage3_add_cast; // sfixed16

wire signed [15:0] Add_stage3_add_temp; // sfixed16

wire signed [17:0] Add_op_stage2; // sfixed18

reg signed [7:0] Delay7_out1; // int8

wire signed [15:0] Product3_out1; // int16

wire signed [15:0] Add_stage4_add_cast; // sfixed16

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

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

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

```

```

        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

    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:

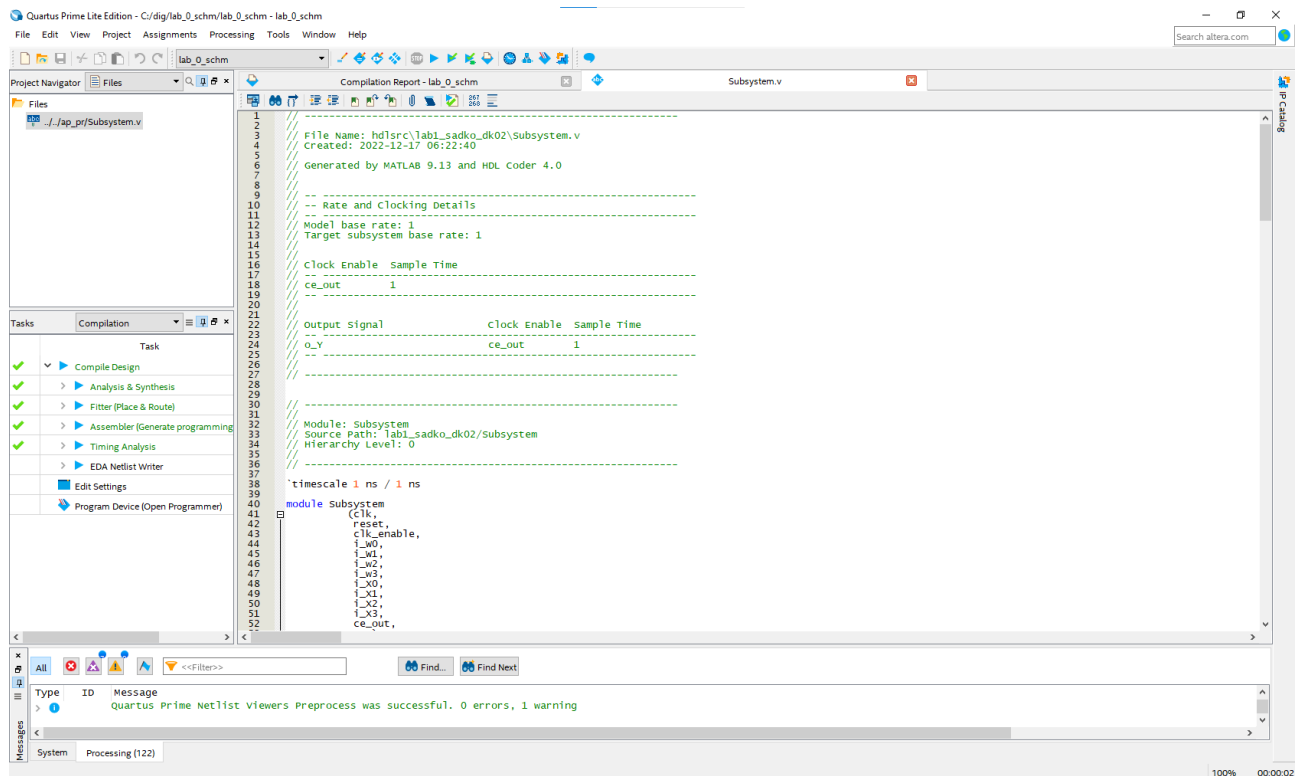


Рис. 12 Результат компіляції згенерованого коду у Quartus Prime Lite

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.

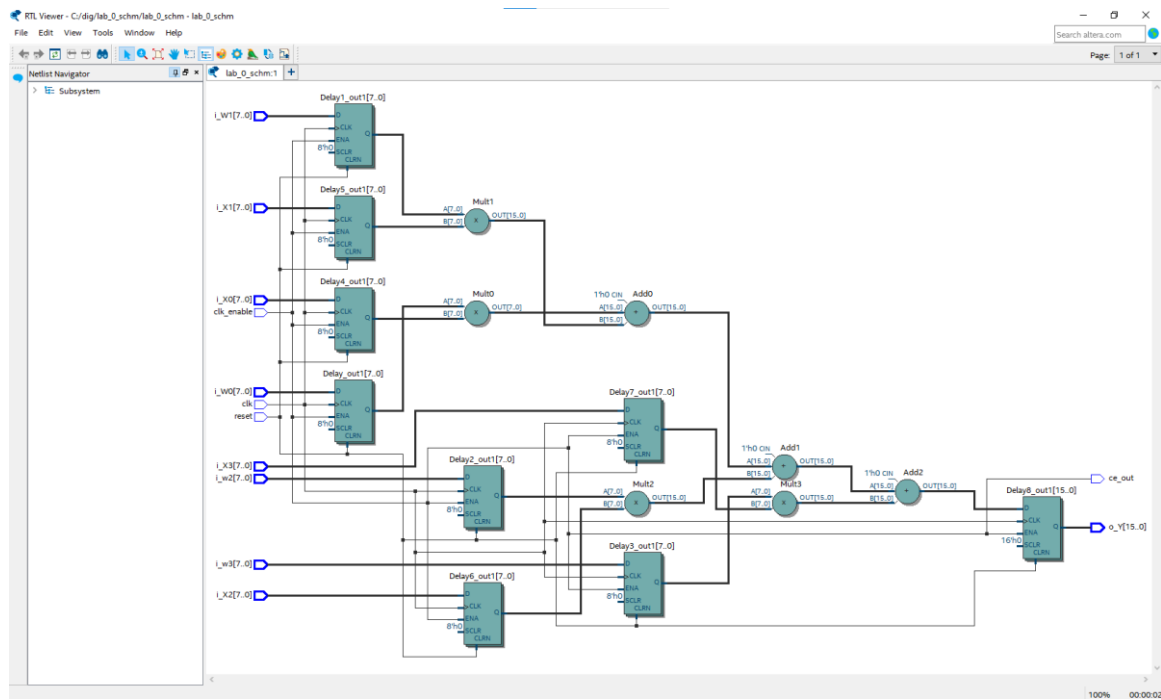


Рис. 13 Синтез згенерованого коду через RTL viewer

Entity:Instance

Cyclone V: 5CSEMA5F31C6

Subsystem

Tasks Compilation

Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming)
Timing Analysis
EDA Netlist Writer
Edit Settings
Program Device (Open Programmer)

Flow Summary

Flow Status: Successful - Sat Dec 17 07:40:29 2022

Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	lab_0_schm
Top-level Entity Name	Subsystem
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	1 / 32,070 (< 1 %)
Total registers	0
Total pins	84 / 457 (18 %)
Total virtual pins	0
Total block memory bits	0 / 4,065,280 (0 %)
Total DSP Blocks	4 / 87 (5 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)

Рис. 14 Результат компіляції та дані про апаратні витрати підсистеми

The screenshot displays the ModelSim - INTEL FPGA Starter Edition 2020.1 interface. The top menu bar includes File, Edit, View, Compile, Simulate, Add, Wave, Tools, Layout, Bookmarks, Window, and Help. Below the menu is a toolbar with various icons for file operations, simulation control, and viewing. The main window is divided into three panes: Design Unit on the left, Objects in the middle, and Wave on the right. The Design Unit pane shows a project tree with 'Subsystem_B' selected. The Objects pane shows a list of objects including 'LX3', 'Data_Type_Conv', 'fb_j2', 'rawData_j2', 'status_j2', 'holdData_j2', 'LX2_offset', 'LX2', 'Data_Type_Conv', 'fb_j1', 'rawData_j1', 'status_j1', 'holdData_j1', 'LX1_offset', 'LX1', and 'Data_Type_Conv'. The Wave pane shows a timing diagram with multiple signals. A yellow cursor is positioned at 64 ns. The bottom status bar shows 'Library', 'Project', 'Sim', and 'Wave' tabs. The bottom pane displays a transcript of simulation messages, including errors and warnings.

18

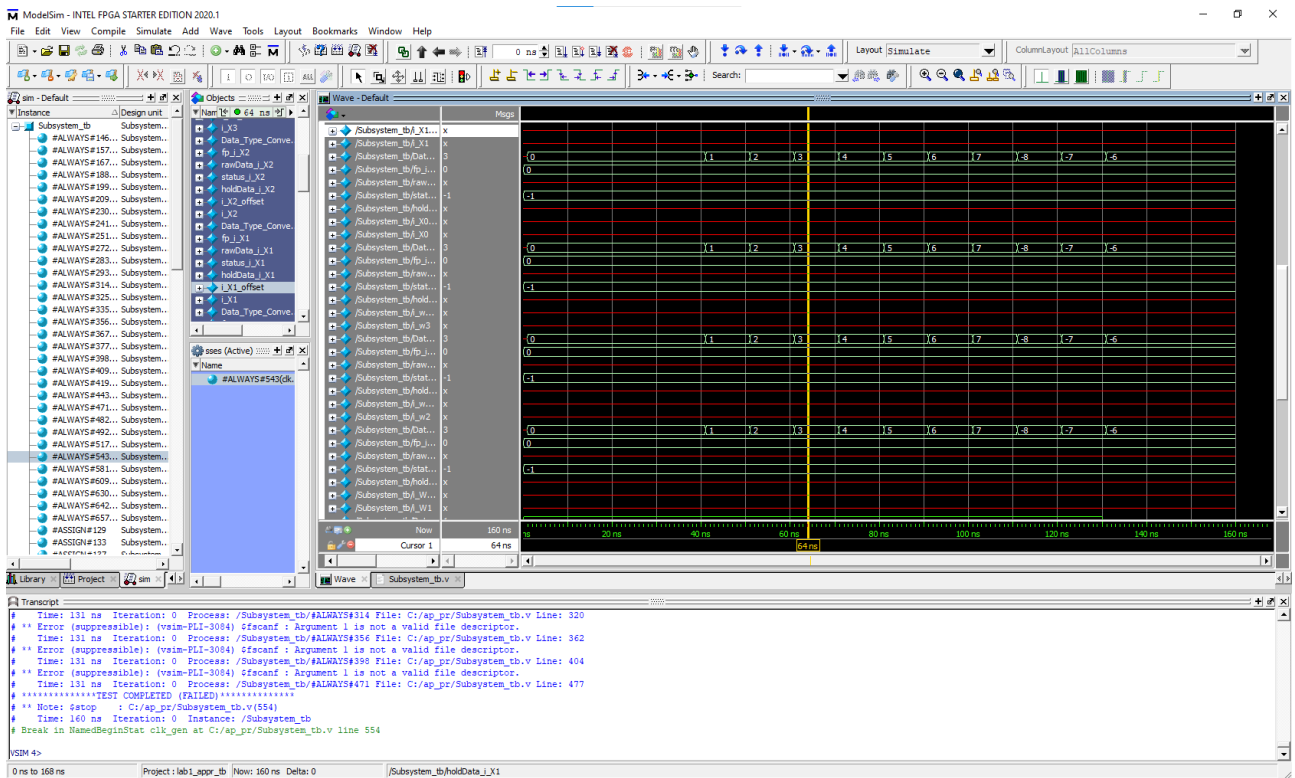


Рис. 17 Результат компіляції та запуску тестбенч файлу

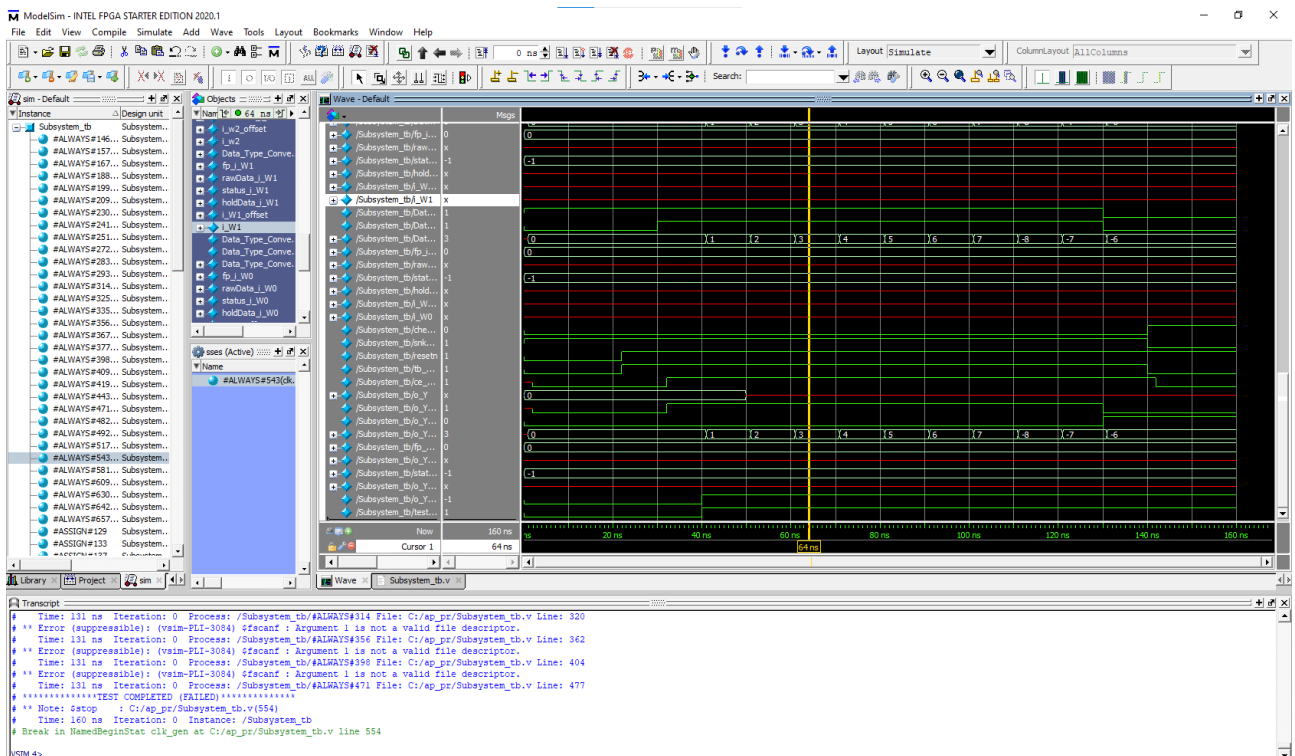


Рис. 18 Результат компіляції та запуску тестбенч файлу

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 зкомпільовавши цю систему отримав результати обчислень на рис. 10.

Додатково провів окреме дослідження, згенерувавши Verilog код підсистеми і в Quartus отримав досить корисні дані про апаратні витрати підсистеми на рис. 14, це дуже важливо для проведення 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.