WEEKLY REPORT – DIGITAL SYSTEM DESIGN USING VERILOG & FPGA

1. General Information

Project Title: Adder_4bits

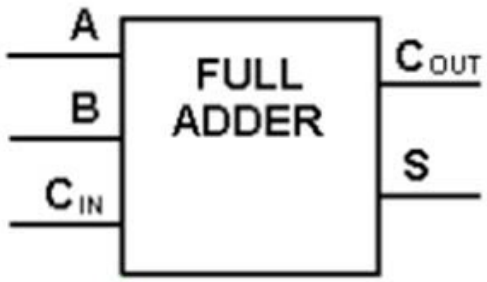
Students: Lê Quang Minh Nhật

Students ID: 23139031

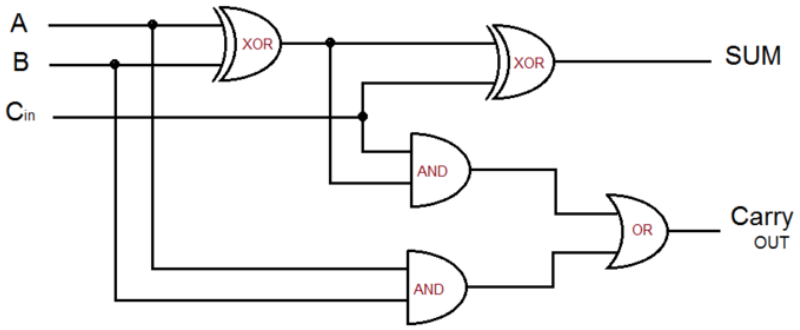
Date: 29/08/2025

2. Block Diagram

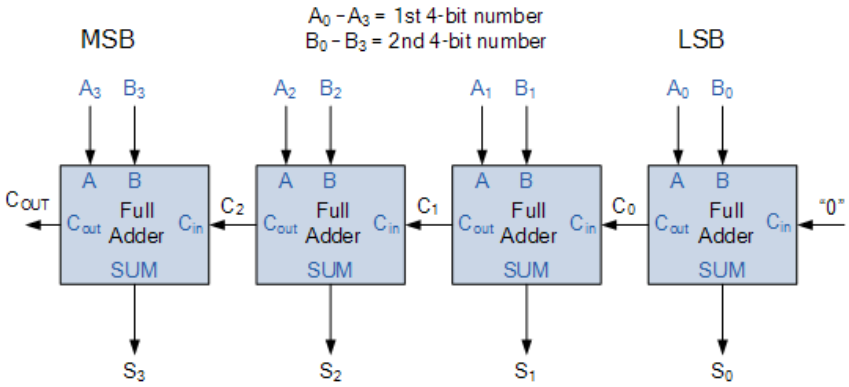
Draw block diagram here.



Hình 1: Sơ đồ khối mạch cộng toàn phần



Hình 2: Mô hình chính trong modum mạch cộng toàn phần 1-bit



Hình 3: Mô hình mạch cộng 4 -bit

3. State Transition Diagram

Describe Truth table or Design requirements or the FSM state diagram, etc

Input			Output	
A	**B**	**Cin**	**S**	**Cout**
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Mô đum FullAdder

```
`timescale 1ns / 1ps

module FullAdder(

        input wire a, b, ci,

        output wire s, co);



        assign s = a^b^ci;

        assign co = ((a^b)&ci) | (a&b);

endmodule
```

Mô đum Adder_4bits

```
`timescale 1ns / 1ps
module Adder_4bits(
    input [3:0] A,
    input [3:0] B,
    input Cin,
    output Cout,
    output [3:0] Sum
    );

wire c1, c2, c3;

FullAdder add0 (A[0], B[0], Cin, Sum[0], c1);
FullAdder add1 (A[1], B[1], c1, Sum[1], c2);
FullAdder add2 (A[2], B[2], c2, Sum[2], c3);
FullAdder add3 (A[3], B[3], c3, Sum[3], Cout);

endmodule
```

4. Test Cases Overview

```
`timescale 1ns / 1ps

module Testbench_Adder_4bits;

        // Inputs
        reg [3:0] A;
        reg [3:0] B;
```

```
reg Cin;

// Outputs
wire Cout;
wire [3:0] Sum;

// Instantiate the Unit Under Test (UUT)
Adder_4bits uut (
        .A(A),
        .B(B),
        .Cin(Cin),
        .Cout(Cout),
        .Sum(Sum)
);

initial begin
        // Initialize Inputs
        A = 4'b0000;
        B = 4'b0000;
        Cin = 1'b0;

        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
```

```
        end


        always forever #10 A = A + 1;

        always forever #20 B = B + 1;


endmodule
```

TC ID	Purpose	Input	Expected Output	Result (Pass/Fail)
TC1	Kiểm tra cộng thông thường	Cin = 1'b0 = 0 A = 4'b1010 = 10 B = 4'b0101 = 5	Sum = 4'b1111 = 15 Cout = 1'b0 = 0	Pass
TC2	Kiểm tra tràn	Cin = 1'b0 = 0 A = 4'b1011 = 11 B = 4'b0101 = 5	Sum = 4'b0000 = 0 Cout = 1b1 = 1	Pass
TC3	Kiểm tra mạch khi không còn tràn	Cin = 1'b0 = 0 A = 4'b0000 = 0 B = 4'b1000 = 8	Sum = 4'b1000 = 8 Cout = 1b0 = 0	Pass

5. Testcase Details

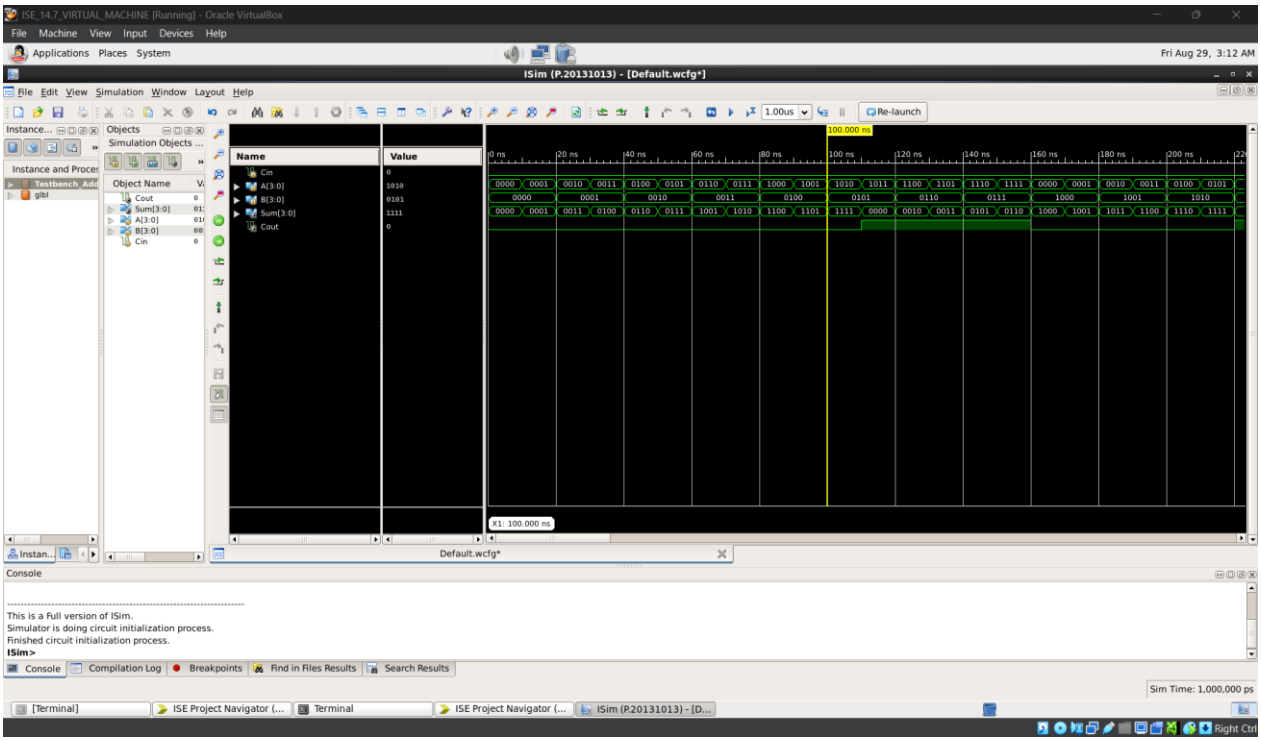
Testcase 1: TC1

Purpose: Kiểm tra cộng thông thường

Input/Stimulus: Cin = 1'b0 = 0, A = 4'b1010 = 10, B = 4'b0101 = 5

Expected Output: Sum = 4'b1111 = 15 Cout = 1'b0 = 0

Waveform Simulation (attach/insert image):



Hình 4: t = 100ns

Analysis:

Tại 100 ns, tín hiệu vào đã ổn định. Phép cộng 1010(10) + 0101(5) + Cin (0)

⇒ 1111(15), không tràn nên Cout = 0. Waveform hiển thị Sum = 1111, Cout = 0, khớp kỳ vọng, không thấy glitch/X.

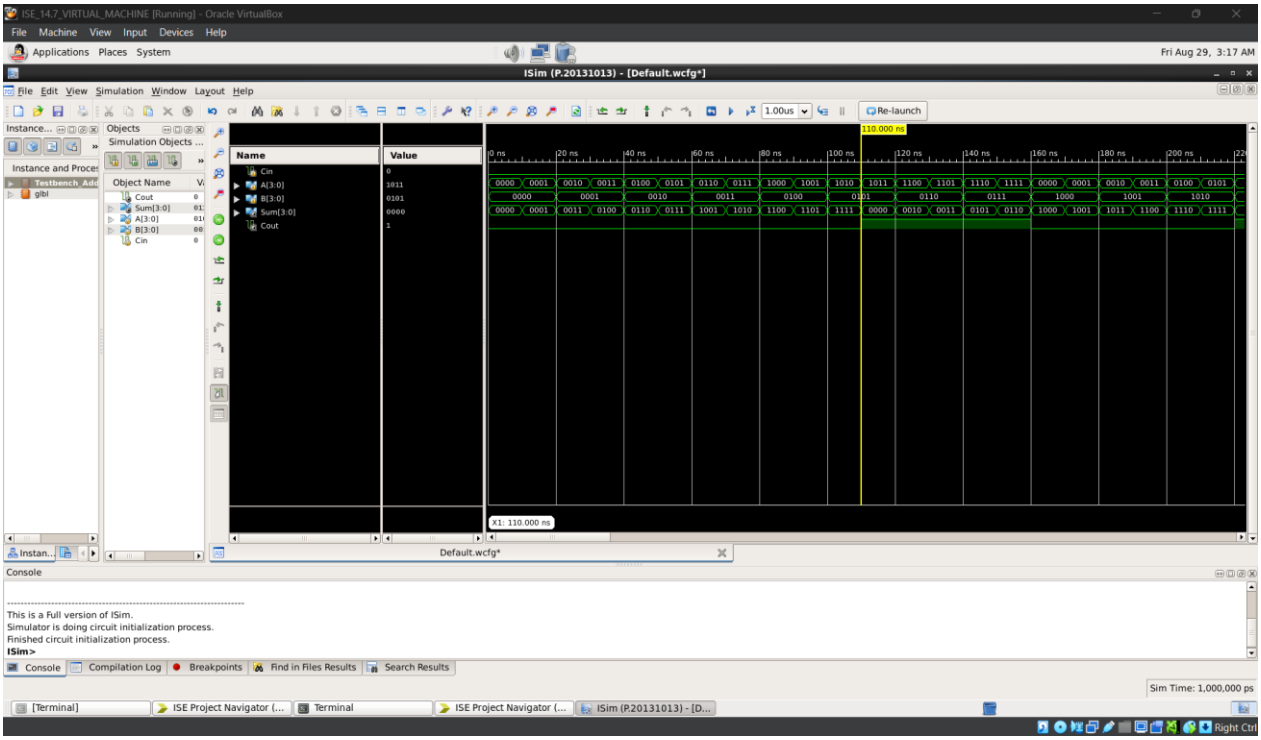
Testcase 2: TC2

Purpose: Kiểm tra tràn

Input/Stimulus: Cin = 1'b0 = 0, A = 4'b1011 = 11, B = 4'b0101 = 5

Expected Output: Sum = 4'b0000 = 0, Cout = 1b1 = 1

Waveform Simulation (attach/insert image):



Hình 5: t = 110ns

Analysis:

Tại 110 ns, Cin = 0, A = 11, B = 5

⇒ 11 + 5 = 16 = $1 \cdot 2^4 + 0$ (mod 16). Vì là bộ cộng 4-bit, Sum = 0000, Cout = 1. Waveform khớp kỳ vọng

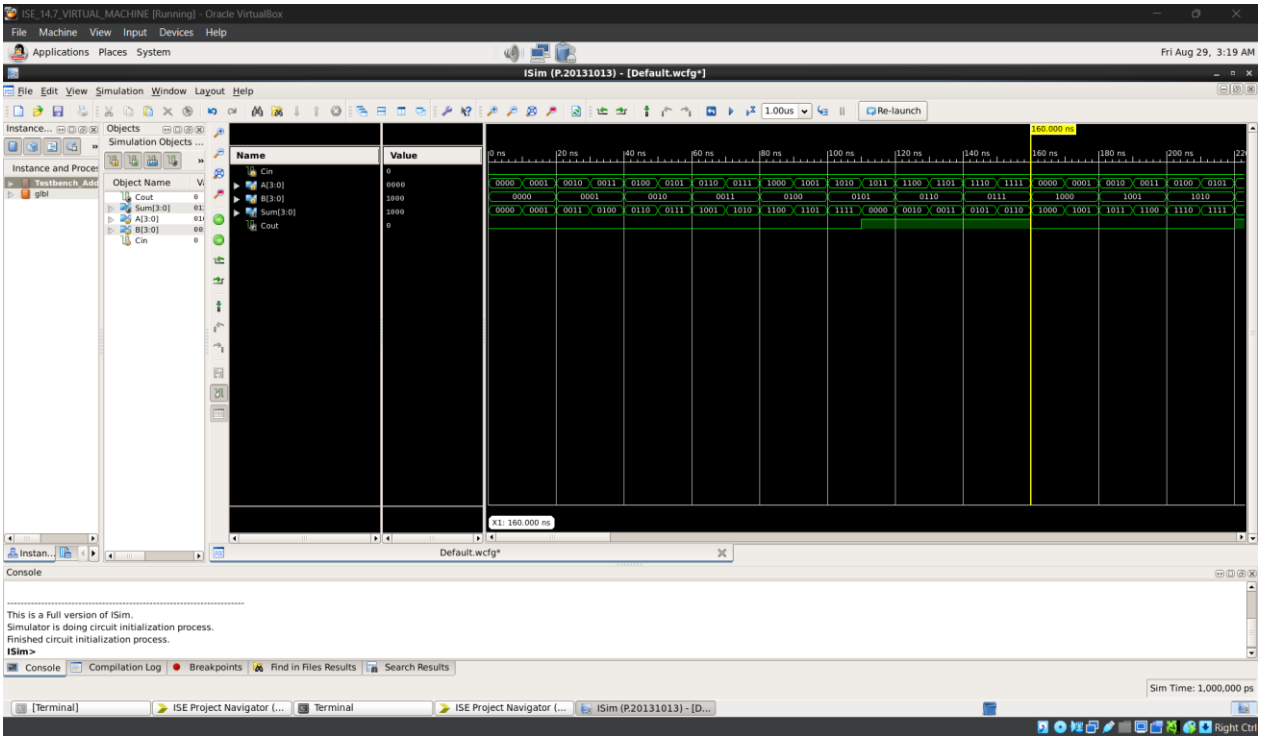
Testcase 3: TC3

Purpose: Kiểm tra mạch khi không còn tràn

Input/Stimulus: Cin = 1'b0 = 0, A = 4'b0000 = 0, B = 4'b1000 = 8

Expected Output: Sum = 4'b1000 = 8, Cout = 1b0 = 0

Waveform Simulation (attach/insert image):

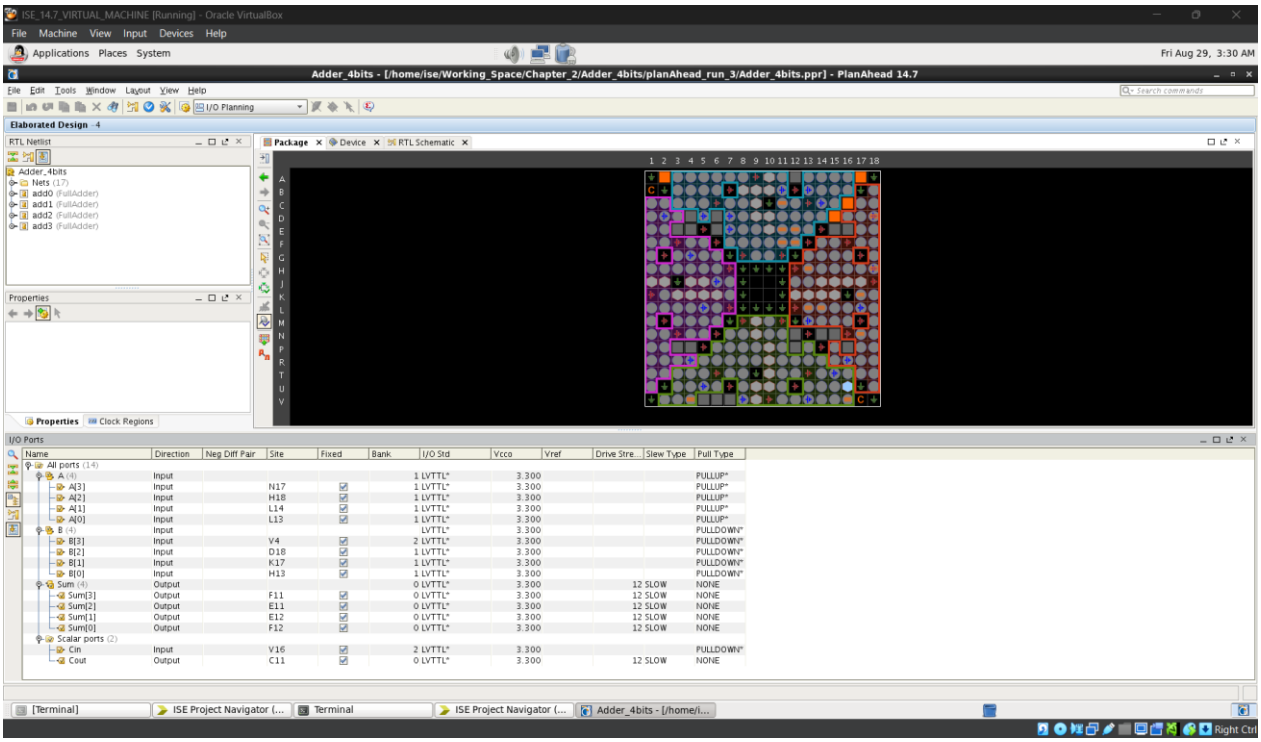


Hình 6: t = 160ns

Analysis:

Tại 160 ns, A = 0, B = 8, Cin = 0 ⇒ 0 + 8 = 8, không phát sinh carry. Waveform cho Sum = 1000, Cout = 0 đúng kỳ vọng

6. Summary



Hình 7: Cấu hình bằng phần mềm

```
NET "A[3]" IOSTANDARD = LVTTL;

NET "A[2]" IOSTANDARD = LVTTL;



NET "A[3]" LOC = N17;

NET "A[2]" LOC = H18;



NET "A[3]" PULLUP;

NET "A[2]" PULLUP;

NET "A[1]" PULLUP;

NET "A[0]" PULLUP;
```

```
NET "B[3]" PULLDOWN;
NET "B[2]" PULLDOWN;
NET "B[1]" PULLDOWN;
NET "B[0]" PULLDOWN;



NET "A[1]" LOC = L14;
NET "A[0]" LOC = L13;



NET "A[0]" IOSTANDARD = LVTTL;
NET "A[1]" IOSTANDARD = LVTTL;
NET "B[3]" IOSTANDARD = LVTTL;
NET "B[2]" IOSTANDARD = LVTTL;
NET "B[1]" IOSTANDARD = LVTTL;
NET "B[0]" IOSTANDARD = LVTTL;
NET "Sum[3]" IOSTANDARD = LVTTL;
NET "Sum[2]" IOSTANDARD = LVTTL;
NET "Sum[1]" IOSTANDARD = LVTTL;
NET "Sum[0]" IOSTANDARD = LVTTL;
NET "Cin" IOSTANDARD = LVTTL;
NET "Cout" IOSTANDARD = LVTTL;



NET "B[3]" LOC = V4;
```

```
NET "B[2]" LOC = D18;

NET "B[1]" LOC = K17;

NET "B[0]" LOC = H13;

NET "Sum[3]" LOC = F11;

NET "Sum[2]" LOC = E11;

NET "Sum[1]" LOC = E12;

NET "Sum[0]" LOC = F12;

NET "Cin" LOC = V16;

NET "Cout" LOC = C11;


# PlanAhead Generated IO constraints


NET "Cin" PULLDOWN;
```

WEEKLY REPORT – DIGITAL SYSTEM DESIGN USING VERILOG & FPGA

1. General Information

Project Title: Decoder2_4

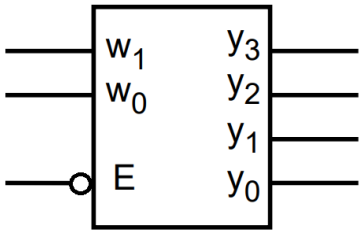
Students: Lê Quang Minh Nhật

Students ID: 23139031

Date: 29/08/2025

2. Block Diagram

Draw block diagram here.



Hình 8: Sơ đồ khối mạch 2 sang 4 E tích cực thấp

3. State Transition Diagram

Describe Truth table or Design requirements or the FSM state diagram, etc

Input			Output			
E	**w_1**	**w_0**	**y_3**	**y_2**	**y_1**	**y_0**
1	x	x	0	0	0	0
0	0	0	0	0	0	1
0	0	1	0	0	1	0
0	1	0	0	1	0	0
0	1	1	1	0	0	0

Mô đum mạch Decoder2_4

```
`timescale 1ns / 1ps


module Decoder2_4(
    input wire [1:0] w,
          input wire E,
    output reg [3:0] y
    );
always @* begin
          y = 4'b0000;
          if (E == 1'b0) begin
                    case (w)
                              0 : y = 4'b0001;
```

```
                1 : y = 4'b0010;
                2 : y = 4'b0100;
                3 : y = 4'b1000;
                default: y = 4'b0000;
            endcase
        end
end
endmodule
```

4. Test Cases Overview

```
`timescale 1ns / 1ps


module Test_bench_Decoder2_4;


    // Inputs
    reg [1:0] w;
    reg E;


    // Outputs
    wire [3:0] y;


    // Instantiate the Unit Under Test (UUT)
    Decoder2_4 uut (
        .w(w),
        .E(E),
        .y(y)
```

```
);

initial begin
        // Initialize Inputs
        w = 0;
        E = 0;

        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
        w = 2'b00; #100;
        w = 2'b01; #100;
        w = 2'b10; #100;
        w = 2'b11; #100;

        E = 1;
        w = 2'b00; #100;
        w = 2'b01; #100;
        w = 2'b10; #100;
        w = 2'b11; #100;


end
```

```
endmodule
```

TC ID	Purpose	Input	Expected Output	Result (Pass/Fail)
TC1	Test tại mức E tích cực thấp	E = 0 w = 2’b01 = 1	y = 4’b 0010	Pass
TC2	Test tại mức E tích cực cao	E = 1 w = 2’b01 = 1	y = 4’b 0000	Pass

5. Testcase Details

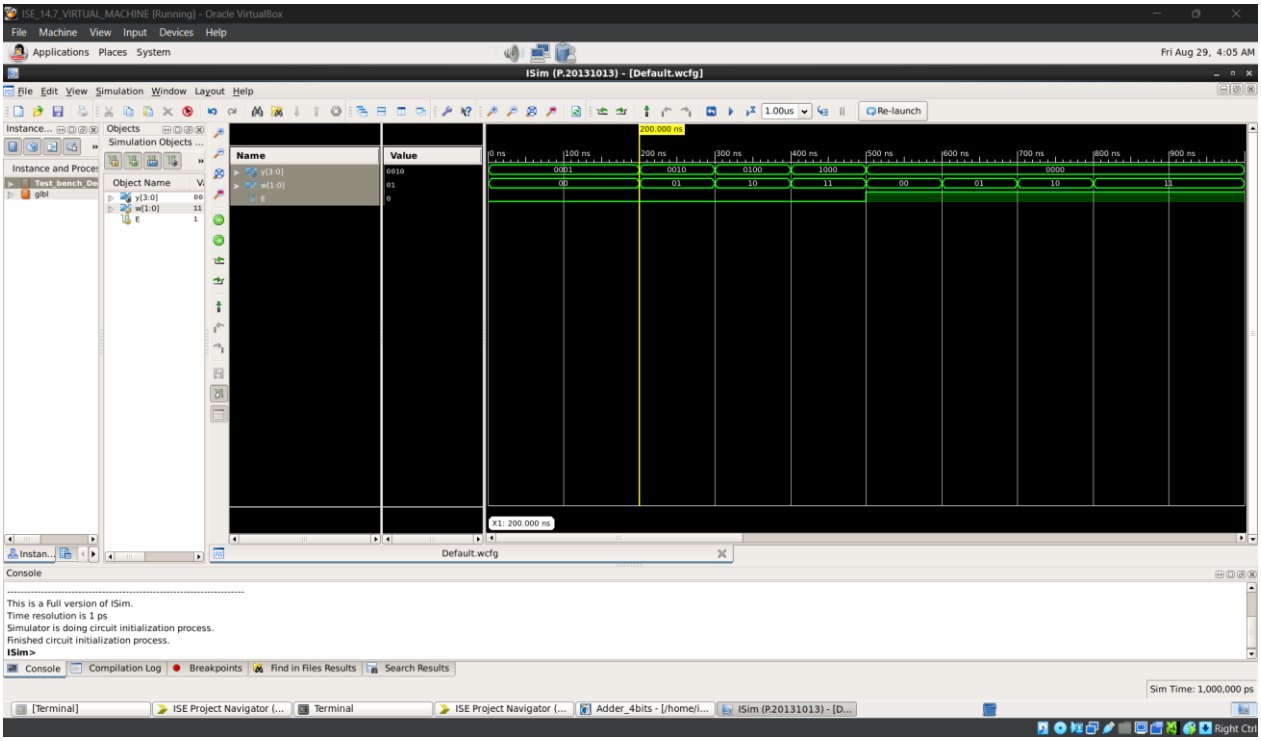
Testcase 1: TC1

Purpose: Test tại mức E tích cực thấp

Input/Stimulus: E = 0, w = 2’b01 = 1

Expected Output: y = 4’b0010

Waveform Simulation (attach/insert image):



Hình 9: t = 200ns

Analysis:

Tại 200 ns, E = 0 ⇒ mạch được kích hoạt. Với w = 1'b01, ngõ ra chọn y1 = 1 và các bit khác = 0

⇒ y = 0010. Waveform cho thấy giá trị ổn định, khớp kỳ vọng

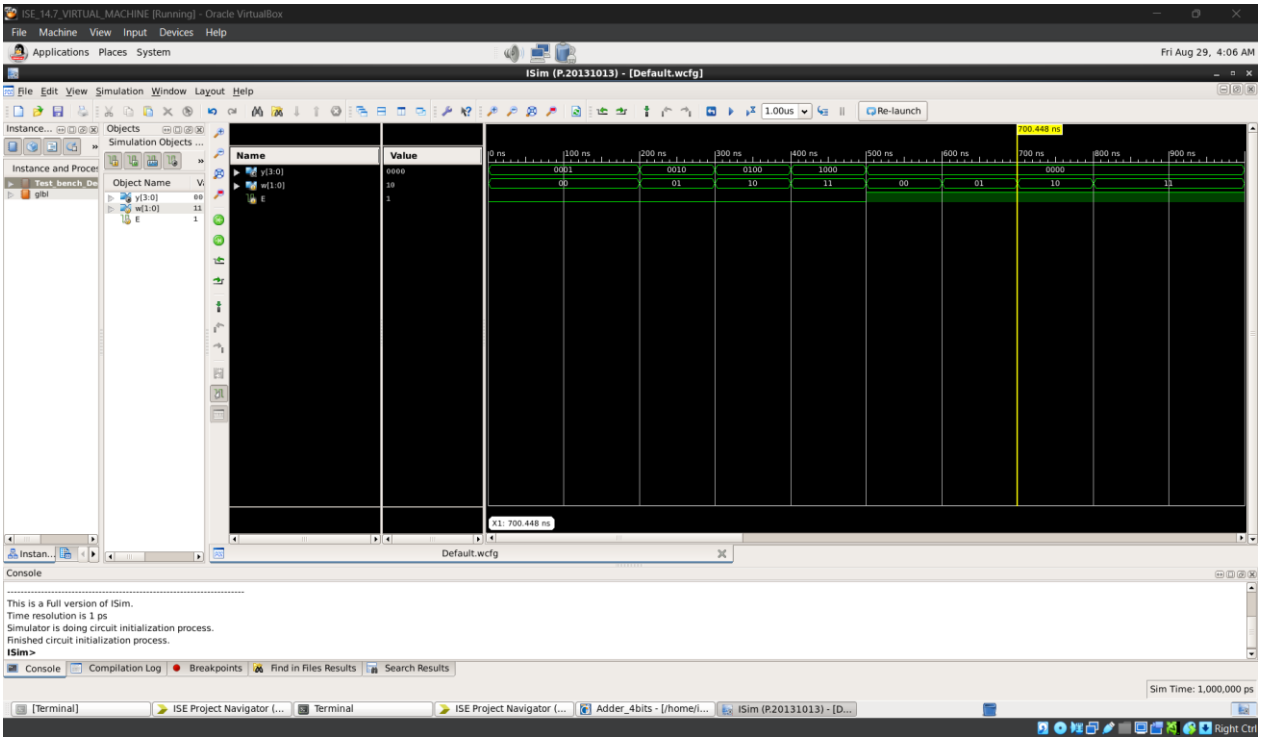
Testcase 2: TC2

Purpose: Test tại mức E tích cực cao

Input/Stimulus: E = 1, w = 2'b01 = 1

Expected Output: y = 4'b0000

Waveform Simulation (attach/insert image):

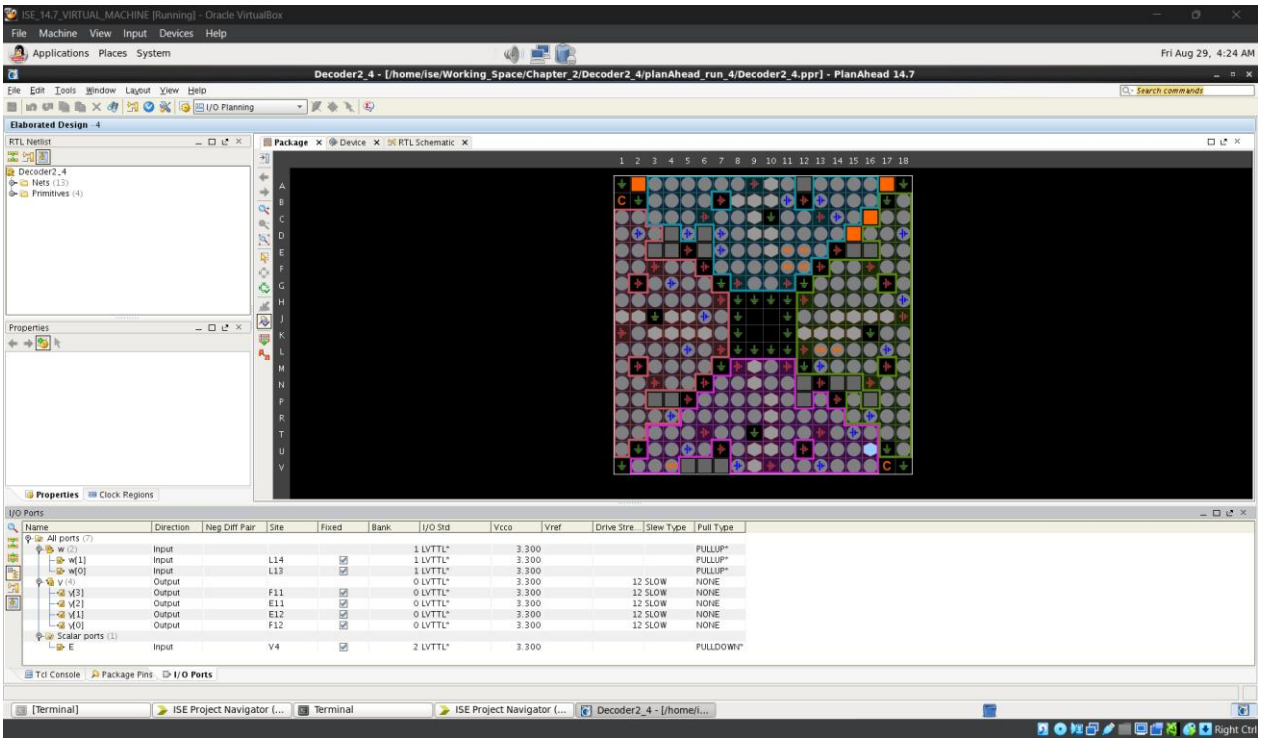


Hình 10: t = 700 ns

Analysis:

E = 1 (enable mức thấp ⇒ vô hiệu) nên y = 4'b0000 bất kể w = 2'b01. Waveform tại 700 ns cho thấy y giữ 0000 ổn định

6. Summary



Hình 11: Cấu hình bằng phần mềm

```
NET "w[1]" IOSTANDARD = LVTTL;

NET "w[0]" IOSTANDARD = LVTTL;

NET "w[1]" PULLUP;

NET "w[0]" PULLUP;



NET "w[1]" LOC = L14;

NET "w[0]" LOC = L13;



NET "y[3]" IOSTANDARD = LVTTL;

NET "y[2]" IOSTANDARD = LVTTL;
```

```
NET "y[1]" IOSTANDARD = LVTTL;
NET "y[0]" IOSTANDARD = LVTTL;



NET "y[3]" LOC = F11;
NET "y[2]" LOC = E11;
NET "y[1]" LOC = E12;
NET "y[0]" LOC = F12;

# PlanAhead Generated physical constraints

NET "E" LOC = V4;

# PlanAhead Generated IO constraints

NET "E" IOSTANDARD = LVTTL;
NET "E" PULLDOWN;
```

WEEKLY REPORT – DIGITAL SYSTEM DESIGN USING VERILOG & FPGA

1. General Information

Project Title: Decoder3_8

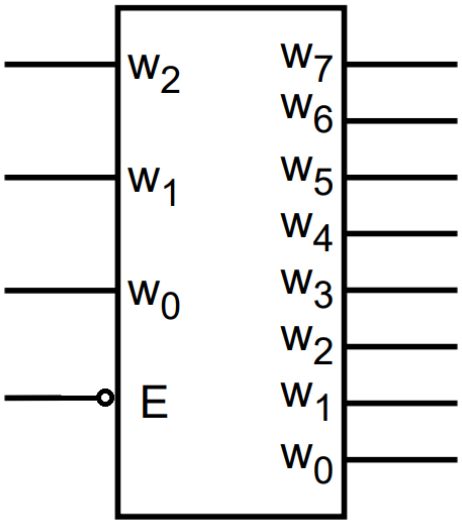
Students: Lê Quang Minh Nhật

Students ID: 23139031

Date: 29/08/2025

2. Block Diagram

Draw block diagram here.



Hình 12: Sơ đồ khối mạch 3 sang 8 E tích cực thấp

3. State Transition Diagram

Describe Truth table or Design requirements or the FSM state diagram, etc

Input				Output							
E	**w_2**	**w_1**	**w_0**	**y_7**	**y_6**	**y_5**	**y_4**	**y_3**	**y_2**	**y_1**	**y_0**
1	x	x	x	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	1	0	0	0	0	0
0	1	1	0	0	1	0	0	0	0	0	0

0	1	1	1	1	0	0	0	0	0	0	0

Mô đum mạch Decoder3_8

```
`timescale 1ns / 1ps


module Decoder3_8(
    input wire E,
    input wire [2:0] w,
    output reg [7:0] y
    );


always @* begin
        y = 8'b0000_0000;
        if (E == 1'b0) begin
                case (w)
                        0 : y = 8'b0000_0001;
                        1 : y = 8'b0000_0010;
                        2 : y = 8'b0000_0100;
                        3 : y = 8'b0000_1000;
                        4 : y = 8'b0001_0000;
                        5 : y = 8'b0010_0000;
                        6 : y = 8'b0100_0000;
                        7 : y = 8'b1000_0000;
                        default: y = 8'b0000_0000;
                endcase
        end
```

```
end

endmodule
```

4. Test Cases Overview

```
`timescale 1ns / 1ps


module Test_bench_Decoder3_8;


	// Inputs
	reg E;
	reg [2:0] w;

	// Outputs
	wire [7:0] y;

	// Instantiate the Unit Under Test (UUT)
	Decoder3_8 uut (
		.E(E),
		.w(w),
		.y(y)
	);

	initial begin
		// Initialize Inputs
		E = 0;
		w = 0;
```

```
        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
        w = 3'b000; #100;
        w = 3'b001; #100;
        w = 3'b010; #100;
        w = 3'b011; #100;
        w = 3'b100; #100;
        w = 3'b101; #100;
        w = 3'b110; #100;
        w = 3'b111; #100;

        E = 1;
        w = 3'b000; #100;
        w = 3'b001; #100;
        w = 3'b010; #100;
        w = 3'b011; #100;
        w = 3'b100; #100;
        w = 3'b101; #100;
        w = 3'b110; #100;
        w = 3'b111; #100;

end
```

endmodule

TC ID	Purpose	Input	Expected Output	Result (Pass/Fail)
TC1	Kiểm tra E tích cực thấp Ngõ ra chọn kênh 0	E = 0 w = 3'b000	y = 8'b0000_0001	Pass
TC2	Kiểm tra E tích cực thấp Ngõ ra chọn kênh 1	E = 0 w = 3'b001	y = 8'b0000_0010	Pass
TC3	Kiểm tra E tích cực cao	E = 1 w = 3'b000	y = 8'b0000_0000	Pass

5. Testcase Details

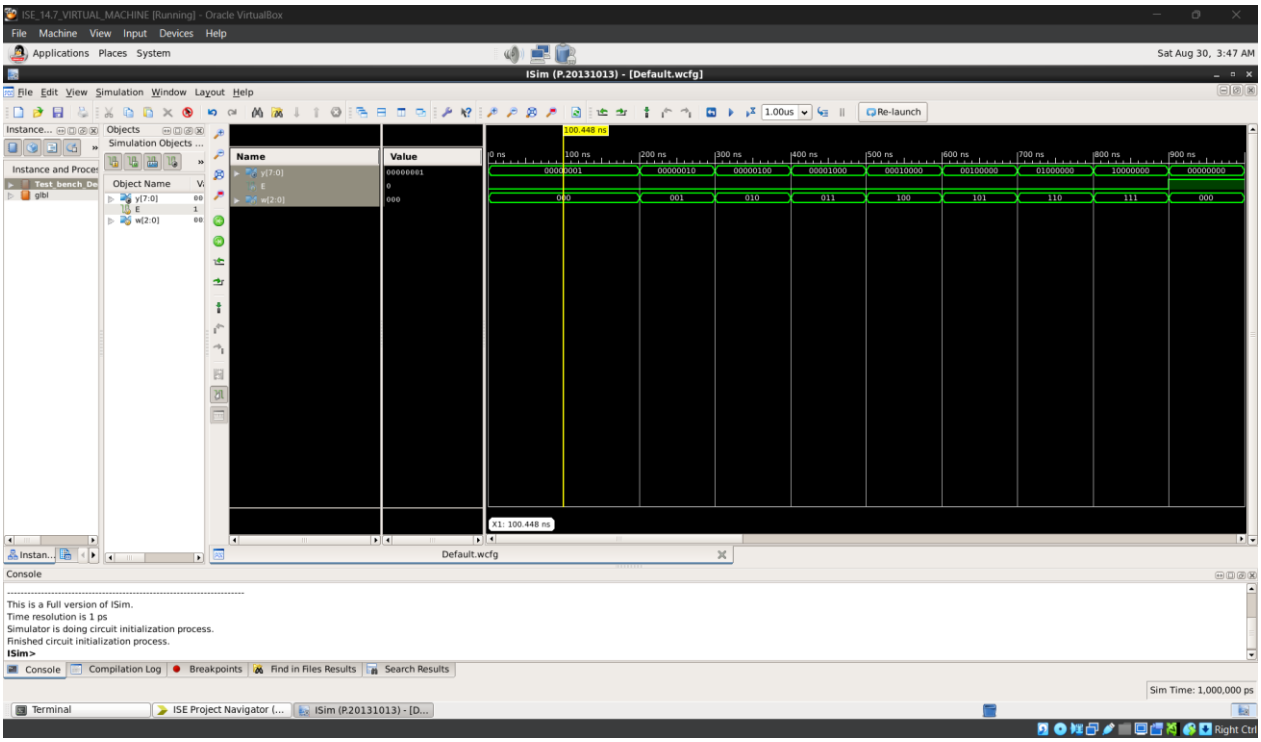
Testcase 1: TC1

Purpose: Kiểm tra E tích cực thấpm, ngõ ra chọn kênh 0.

Input/Stimulus: E = 0, w = 3'b000

Expected Output: y = 8'b0000_0001

Waveform Simulation (attach/insert image):



Hình 13: t = 100ns

Analysis:

Tại 100 ns, E = 0 (enable), w = 3’b000:

⇒ ngõ ra chọn Y_0: y = 8’b0000_0001. Waveform cho thấy y[0]=1, các bit khác = 0, ổn định.

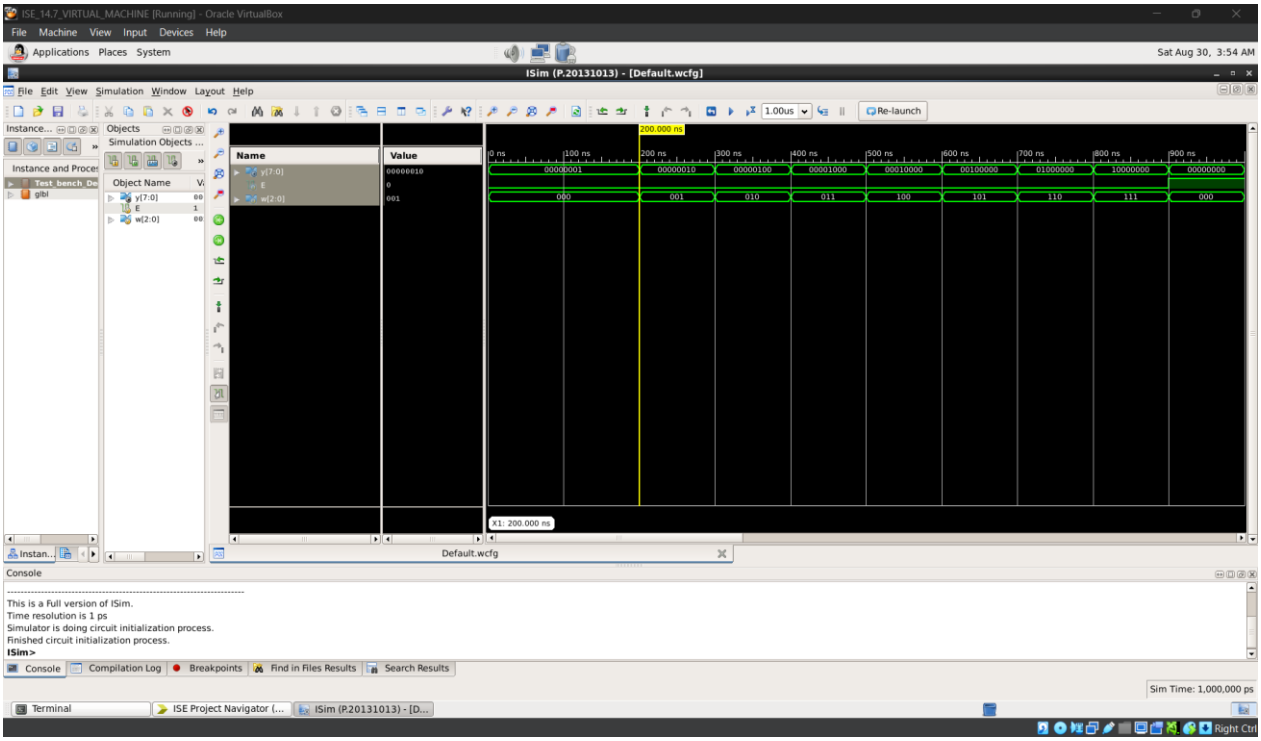
Testcase 2: TC2

Purpose: Kiểm tra E tích cực thấp, ngõ ra chọn kênh.

Input/Stimulus: E = 0, w = 3’b001

Expected Output: y = 8’b0000_0010

Waveform Simulation (attach/insert image):



Hình 14: t = 200ns

Analysis:

Tại 200 ns, E = 0 (enable). Với w = 3’b001, DEMUX chọn kênh 1.

⇒ y[1]=1, các bit khác = 0, y = 8’b0000_0010. Waveform khớp kỳ vọng, không thấy glitch/X.

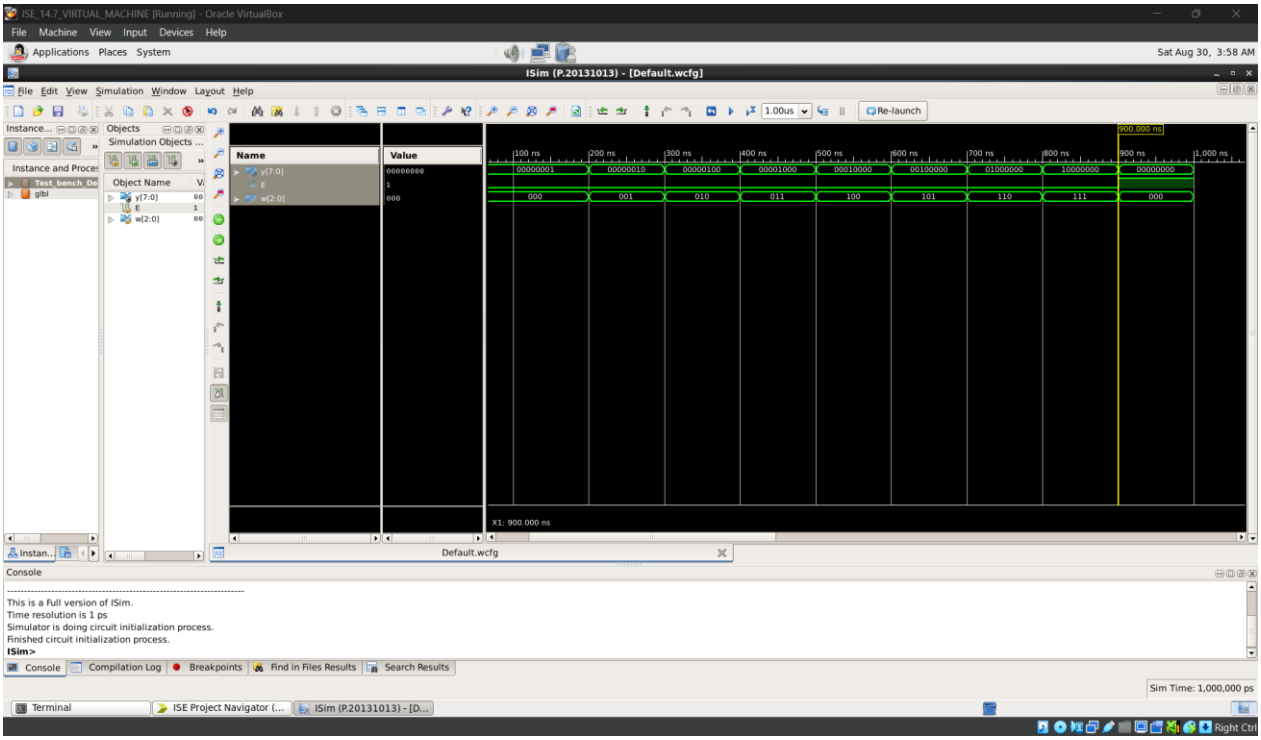
Testcase 3: TC3

Purpose: Kiểm tra E tích cực cao

Input/Stimulus: E = 1, w = 3’b000

Expected Output: y = 8’b0000_0000

Waveform Simulation (attach/insert image):



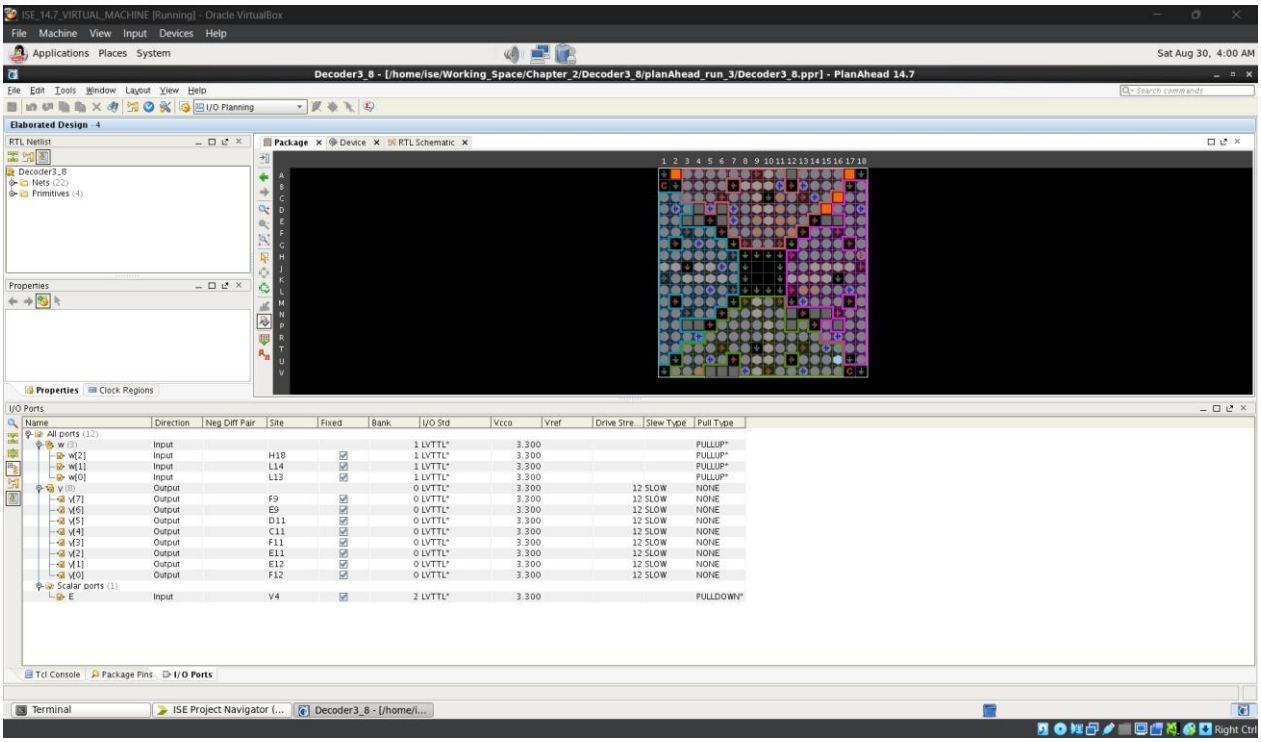
Hình 15: t = 900ns

Analysis:

E = 1 (enable mức thấp ⇒ disable) nên y = 8’b0000_0000 bất kể w = 3’b000.

Waveform tại mốc kiểm tra cho thấy toàn bộ bit y = 0 ổn định.

6. Summary



Hình 16: Cấu hình bằng phần mềm

```
NET "w[2]" IOSTANDARD = LVTTL;

NET "w[1]" IOSTANDARD = LVTTL;

NET "w[0]" IOSTANDARD = LVTTL;

NET "y[7]" IOSTANDARD = LVTTL;

NET "y[6]" IOSTANDARD = LVTTL;

NET "y[5]" IOSTANDARD = LVTTL;

NET "y[4]" IOSTANDARD = LVTTL;

NET "y[3]" IOSTANDARD = LVTTL;

NET "y[2]" IOSTANDARD = LVTTL;

NET "y[1]" IOSTANDARD = LVTTL;

NET "y[0]" IOSTANDARD = LVTTL;

NET "E" IOSTANDARD = LVTTL;
```

```
NET "w[2]" LOC = H18;
NET "w[1]" LOC = L14;
NET "w[0]" LOC = L13;
NET "y[7]" LOC = F9;
NET "y[6]" LOC = E9;
NET "y[5]" LOC = D11;
NET "y[4]" LOC = C11;
NET "y[3]" LOC = F11;
NET "y[2]" LOC = E11;
NET "y[1]" LOC = E12;
NET "y[0]" LOC = F12;
NET "E" LOC = V4;

# PlanAhead Generated IO constraints

NET "w[2]" PULLUP;
NET "w[1]" PULLUP;
NET "w[0]" PULLUP;
NET "E" PULLDOWN;
```

WEEKLY REPORT – DIGITAL SYSTEM DESIGN USING VERILOG & FPGA

1. General Information

Project Title: Mux 4_1 có E tích cực cao

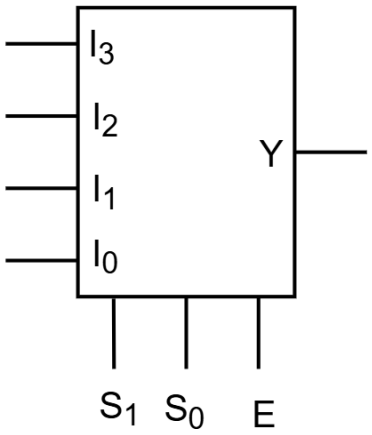
Students: Lê Quang Minh Nhật

Students ID: 23139031

Date: 9/9/2005

2. Block Diagram

Draw block diagram here.



3. State Transition Diagram

Describe Truth table or Design requirements or the FSM state diagram, etc

Input							Output
I_3	**I_2**	**I_1**	**I_0**	**S_1**	**S_0**	**E**	**Y**
x	x	x	x	x	x	0	0
x	x	x	I_0	0	0	1	I_0
x	x	I_1	x	0	1	1	I_1
x	I_2	x	x	1	0	1	I_2
I_3	x	x	x	1	1	1	I_3

Mô đum Mux4_1_E_High

```
`timescale 1ns / 1ps
```

```
module Mux4_1_E_High(
    input wire [3:0] I,
    input wire [1:0] S,
    input wire E,
    output reg Y
    );
always @* begin
        if (E == 1'b0) begin
                Y = 1'b0;
        end
        else begin
                case(S)
                        2'b00: Y = I[0];
                        2'b01: Y = I[1];
                        2'b10: Y = I[2];
                        2'b11: Y = I[3];
                        default: Y = 0;
                endcase
        end
end
endmodule
```

4. Test Cases Overview

TC ID	Purpose	Input	Expected Output	Result (Pass/Fail)
TC1	Kiểm tra mức E	E = 0	Y = 0	Pass
TC2	Kiểm tra ngõ ra S = 2’b00	E = 0 S = 2’b00	Y = I[0]	Pass

TC3	Kiểm tra ngõ ra S = 2'b10	E = 0 S = 2'b10	Y = I[2]	Pass

Mô đum testbench

```
`timescale 1ns / 1ps


module Testbench_Mux4_1_E_High;


	// Inputs
	reg [3:0] I;
	reg [1:0] S;
	reg E;

	// Outputs
	wire Y;

	// Instantiate the Unit Under Test (UUT)
	Mux4_1_E_High uut (
		.I(I),
		.S(S),
		.E(E),
		.Y(Y)
	);

	initial begin
		// Initialize Inputs
		I = 0;
```

```
        S = 0;
        E = 0;

        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
        E = 1;

    end
    always #10    I[0] = ~I[0];
    always #20    I[1] = ~I[1];
    always #30    I[2] = ~I[2];
    always #40    I[3] = ~I[3];

    always #100 S = S + 1'b1;

endmodule
```

5. Testcase Details

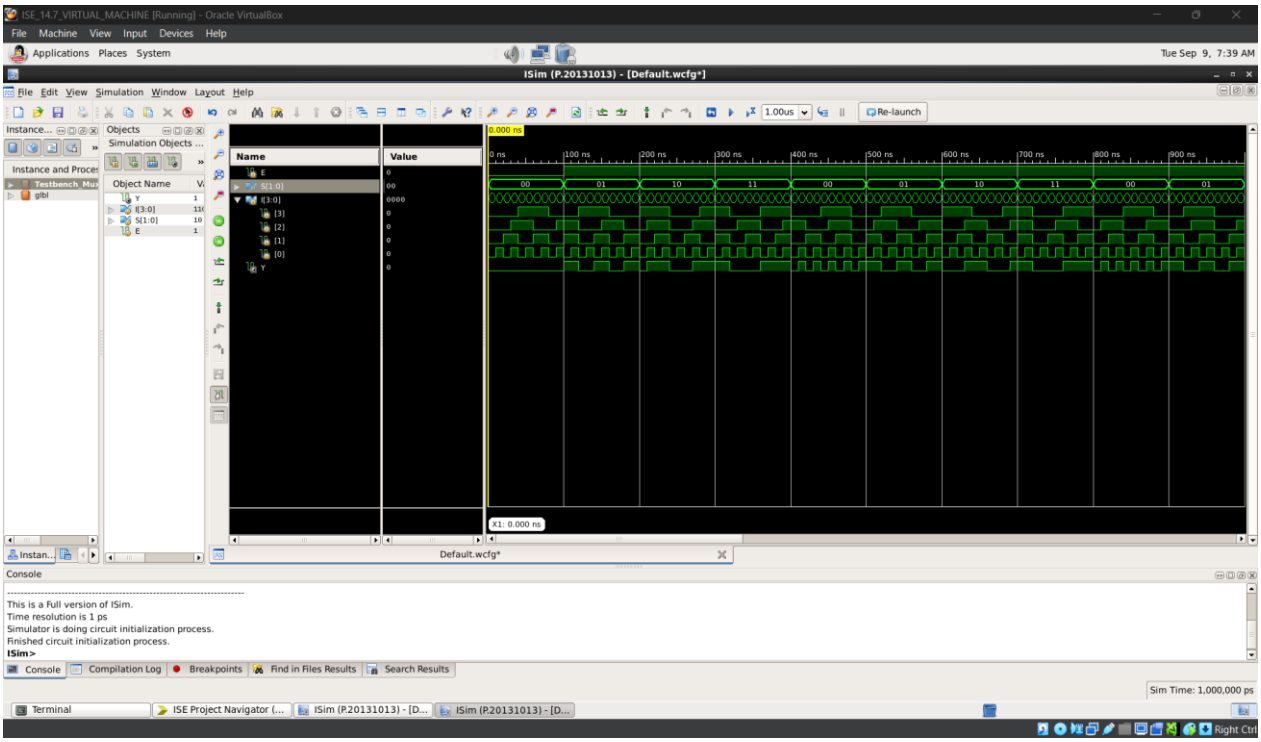
Testcase 1: TC1

Purpose: Kiểm tra mức E

Input/Stimulus: E = 0

Expected Output: Y = 0

Waveform Simulation (attach/insert image):



Analysis:

Trong khoảng thời gian 0ns → 100ns, E = 0 nên Y = 0 (Theo mặc định của bảng trạng thái)

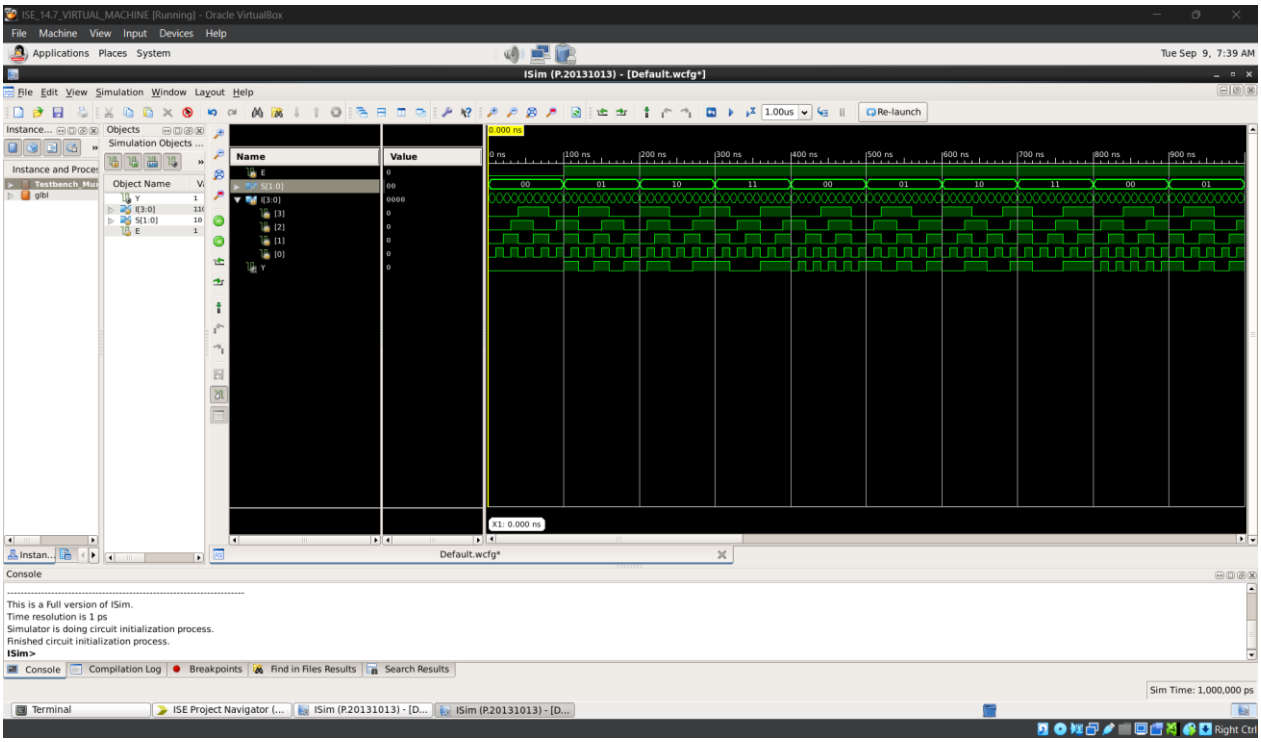
Testcase 2: TC2

Purpose: Kiểm tra ngõ ra S = 2'b00

Input/Stimulus: E = 0, S = 2'b00

Expected Output: Y = I[0]

Waveform Simulation (attach/insert image):



Analysis:

Trong khoảng thời gian này, E = 1 nên Y = I[0] . Vậy nên dạng sóng ngõ ra trong khoảng thời gian đó, trùng với dạng sóng của I[0]

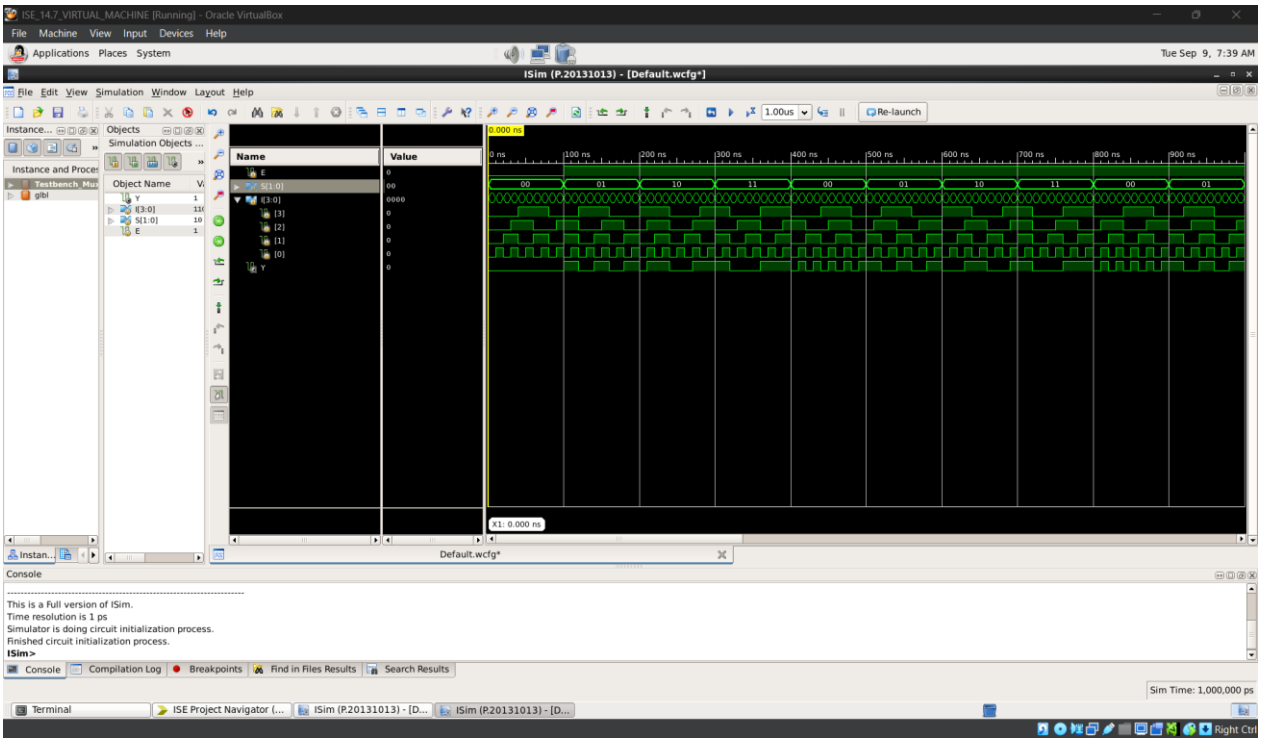
Testcase 3: TC3

Purpose: Kiểm tra ngõ ra S = 2'b10

Input/Stimulus: E = 0, S = 2'b10

Expected Output: Y = I[2]

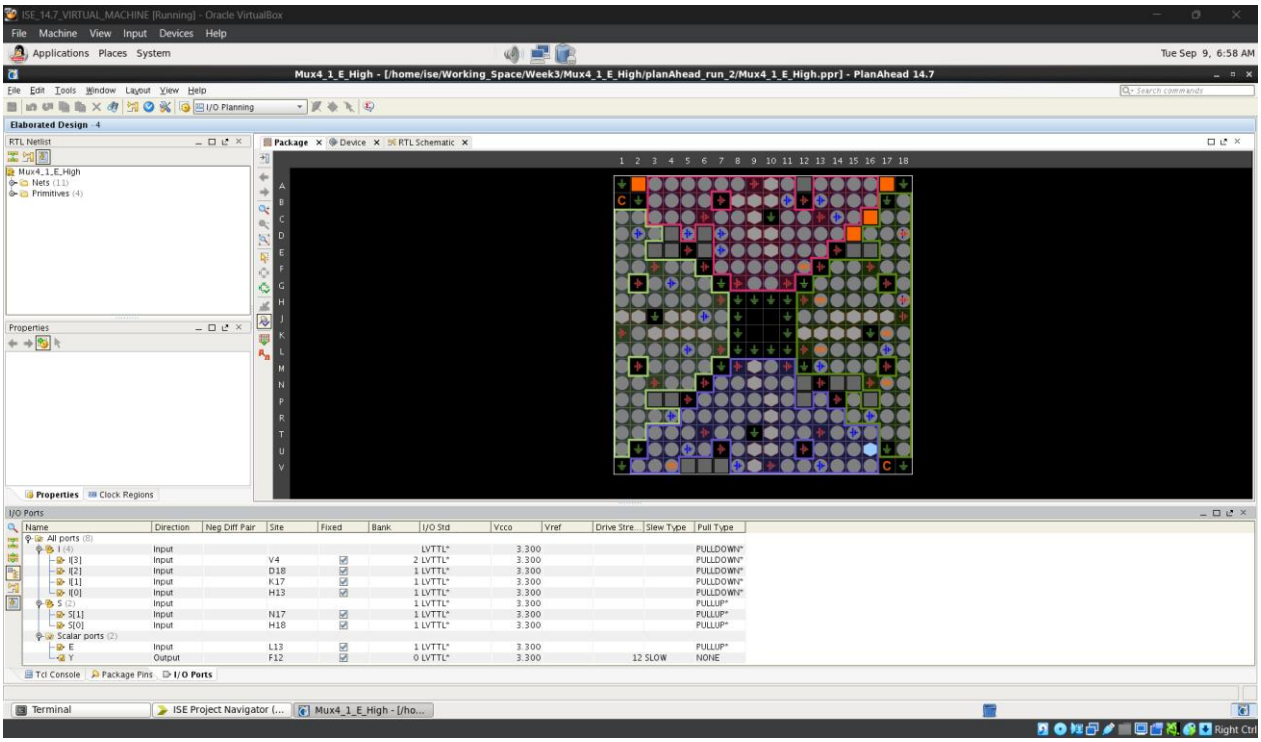
Waveform Simulation (attach/insert image):



Analysis:

Trong khoảng thời gian này, E = 1 nên Y = I[2] . Vậy nên dạng sóng ngõ ra trong khoảng thời gian đó, trùng với dạng sóng của I[2]

6. Summary



Hình 17: Cấu hình bằng phần mềm

```
# PlanAhead Generated IO constraints


NET "I[3]" IOSTANDARD = LVTTL;

NET "I[2]" IOSTANDARD = LVTTL;

NET "I[1]" IOSTANDARD = LVTTL;

NET "I[0]" IOSTANDARD = LVTTL;

NET "S[1]" IOSTANDARD = LVTTL;

NET "S[0]" IOSTANDARD = LVTTL;

NET "E" IOSTANDARD = LVTTL;

NET "Y" IOSTANDARD = LVTTL;


# PlanAhead Generated physical constraints
```

```
NET "S[0]" LOC = H18;

NET "E" LOC = L13;

NET "Y" LOC = F12;

NET "S[1]" LOC = N17;


# PlanAhead Generated IO constraints


NET "S[1]" PULLUP;

NET "S[0]" PULLUP;

NET "I[3]" PULLDOWN;

NET "I[2]" PULLDOWN;

NET "I[1]" PULLDOWN;

NET "I[0]" PULLDOWN;


# PlanAhead Generated physical constraints


NET "I[3]" LOC = V4;

NET "I[2]" LOC = D18;

NET "I[1]" LOC = K17;

NET "I[0]" LOC = H13;


# PlanAhead Generated IO constraints


NET "E" PULLUP;
```

WEEKLY REPORT – DIGITAL SYSTEM DESIGN USING VERILOG & FPGA

1. General Information

Project Title: Mux10_1_by_4_1

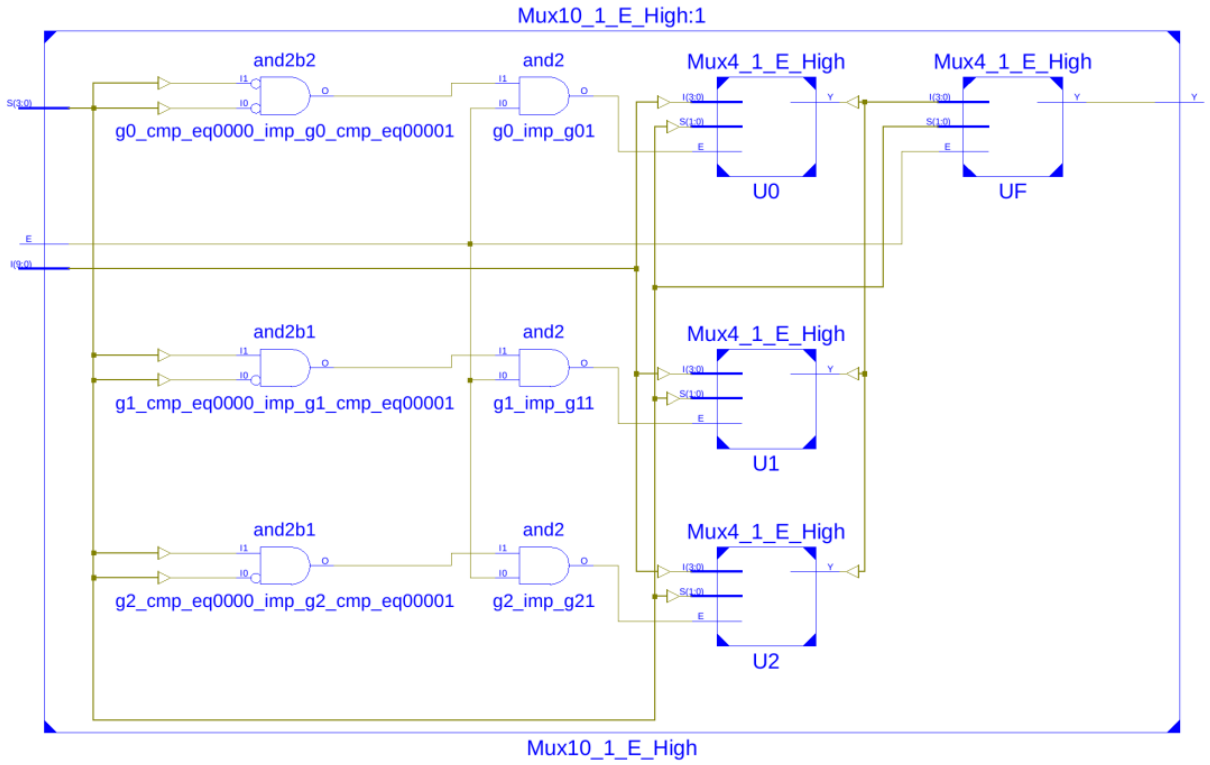
Students: Lê Quang Minh Nhật

Students ID: 2319031

Date: 9/9/2005

2. Block Diagram

Draw block diagram here.



3. State Transition Diagram

Describe Truth table or Design requirements or the FSM state diagram, etc

Input					Output
S_3	**S_2**	**S_1**	**S_0**	**E**	**Y**
x	x	x	x	0	0
0	0	0	0	1	I_0
0	0	0	1	1	I_1
0	0	1	0	1	I_2
0	0	1	1	1	I_3

0	1	0	0	1	I_4
0	1	0	1	1	I_5
0	1	1	0	1	I_6
0	1	1	1	1	I_7
1	0	0	0	1	I_8
1	0	0	1	1	I_9
1	0	1	0	1	0
1	0	1	1	1	0

Mô đum Mux10_1

```
`timescale 1ns / 1ps


module Mux10_1_E_High(
    input wire [9:0] I,
    input wire [3:0] S,
    input wire E,
    output wire Y
    );


wire y0, y1, y2;


wire g0 = E & (S[3:2] == 2'b00);
wire g1 = E & (S[3:2] == 2'b01);
wire g2 = E & (S[3:2] == 2'b10);


Mux4_1_E_High U0 (.I(I[3:0]), .S(S[1:0]), .E(g0), .Y(y0));
Mux4_1_E_High U1 (.I(I[7:4]), .S(S[1:0]), .E(g1), .Y(y1));
Mux4_1_E_High U2 (.I({1'b0,1'b0,I[9],I[8]}), .S(S[1:0]), .E(g2), .Y(y2));
```

```
Mux4_1_E_High UF (.I({1'b0, y2, y1, y0}), .S(S[3:2]), .E(E), .Y(Y));


endmodule
```

4. Test Cases Overview

TC ID	Purpose	Input	Expected Output	Result (Pass/Fail)
TC1	Xét ngõ ra mức E = 0	E = 0	Y = 1’b0	Pass
TC2	Xét ngõ ra chọn E = 1, I[0]	E = 1, S = 4’b0000	Y = I[0]	Pass
TC3	Xét ngõ ra chọn E = 1, I[4]	E = 1 S = 4’b0100	Y = I[4]	Pass

Mô đum testbench

```
`timescale 1ns / 1ps


module Testbench_Mux10_1_E_High;


    // Inputs
    reg [9:0] I;
    reg [3:0] S;
    reg E;


    // Outputs
    wire Y;


    // Instantiate the Unit Under Test (UUT)
    Mux10_1_E_High uut (
        .I(I),
        .S(S),
```

```
        .E(E),
        .Y(Y)
);

initial begin
        // Initialize Inputs
        I = 0;
        S = 0;
        E = 0;

        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
        E = 1;

        //00
        S = 4'b0000;
        #600;
        S = 4'b0001;
        #500;
        S = 4'b0010;
        #500;
        S = 4'b0011;
        #500;
```

```
        //01
        S = 4'b0100;
        #500;
        S = 4'b0101;
        #500;
        S = 4'b0110;
        #500;
        S = 4'b0111;
        #500;

        //10
        S = 4'b1000;
        #500;
        S = 4'b1001;
        #500;

end
always #5 I[9] = ~I[9];
always #10 I[8] = ~I[8];
always #15 I[7] = ~I[7];
always #20 I[6] = ~I[6];
always #25 I[5] = ~I[5];
always #30 I[4] = ~I[4];
always #35 I[3] = ~I[3];
```

```
        always #40 I[2] = ~I[2];

        always #45 I[1] = ~I[1];

        always #50 I[0] = ~I[0];


endmodule
```

5. Testcase Details

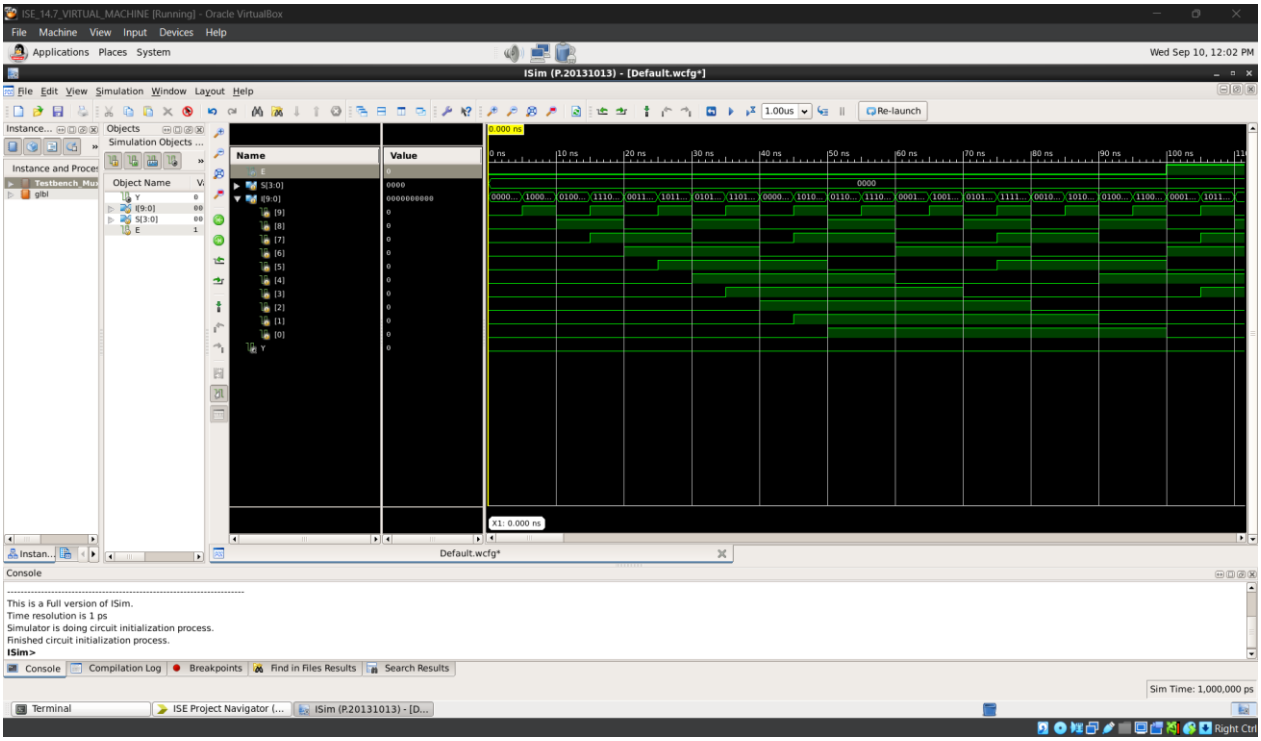
Testcase 1: TC1

Purpose: Xét ngõ ra mức E = 0

Input/Stimulus: E = 0

Expected Output: Y = 1'b0

Waveform Simulation (attach/insert image):



Analysis:

Khi ngõ vào E = 0, mạch không cho phép ngõ vào nào hết, do đó ngõ ra luôn ở mức 0.

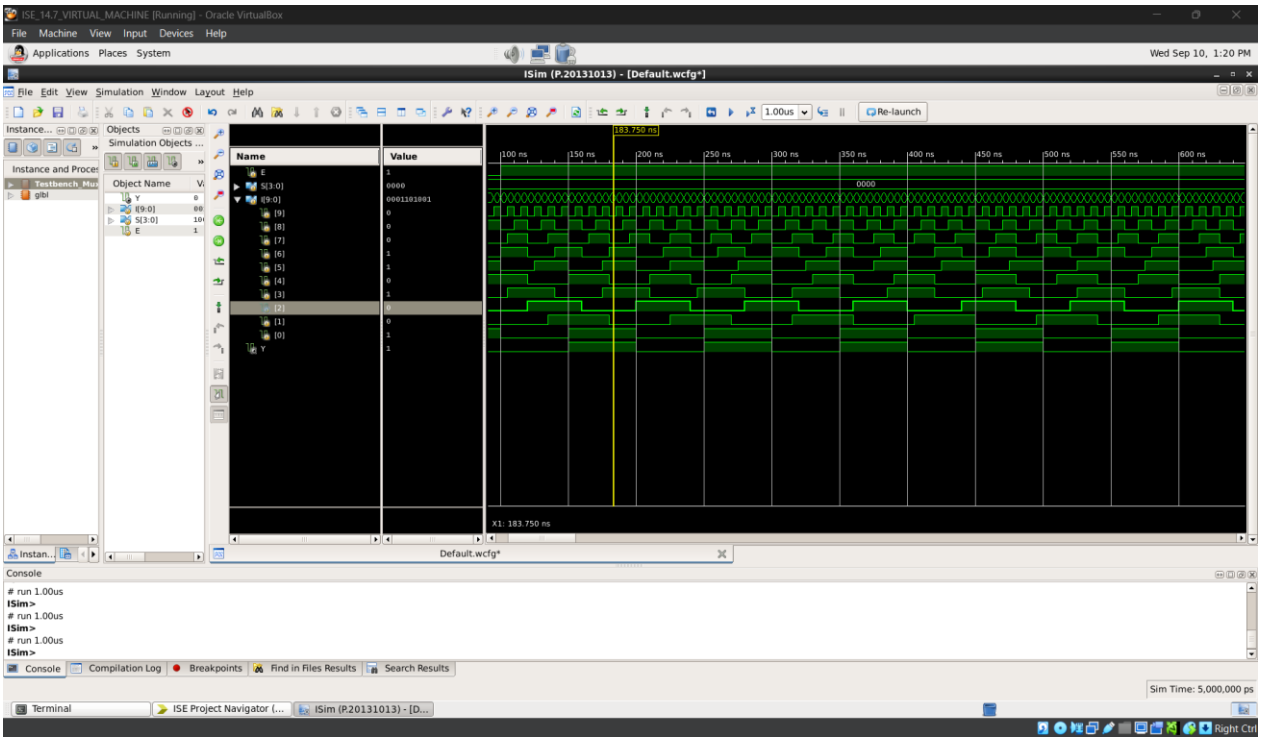
Testcase 2: TC2

Purpose: Xét ngõ ra chọn E = 1, I[0]

Input/Stimulus: E = 1, S = 4'b0000

Expected Output: Y = I[0]

Waveform Simulation (attach/insert image):



Analysis:

Khi ngõ vào cho phép E = 1, và mạch chọn S = 4'b0000, thì mạch có ngõ ra theo tần số của ngõ vào I[0], đúng theo yêu cầu của đề và bảng trạng thái.

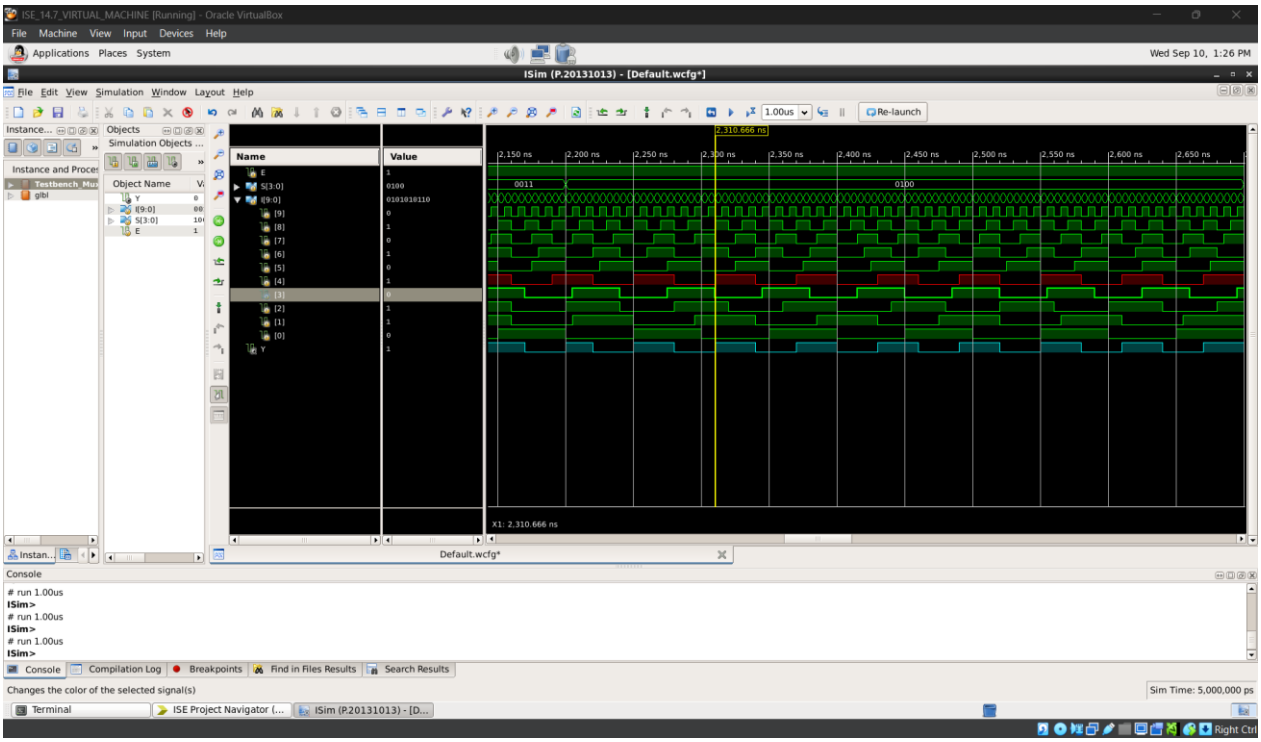
Testcase 3: TC3

Purpose: Xét ngõ ra chọn E = 1, I[4]

Input/Stimulus: E = 1, S = 4'b0100

Expected Output: Y = I[4]

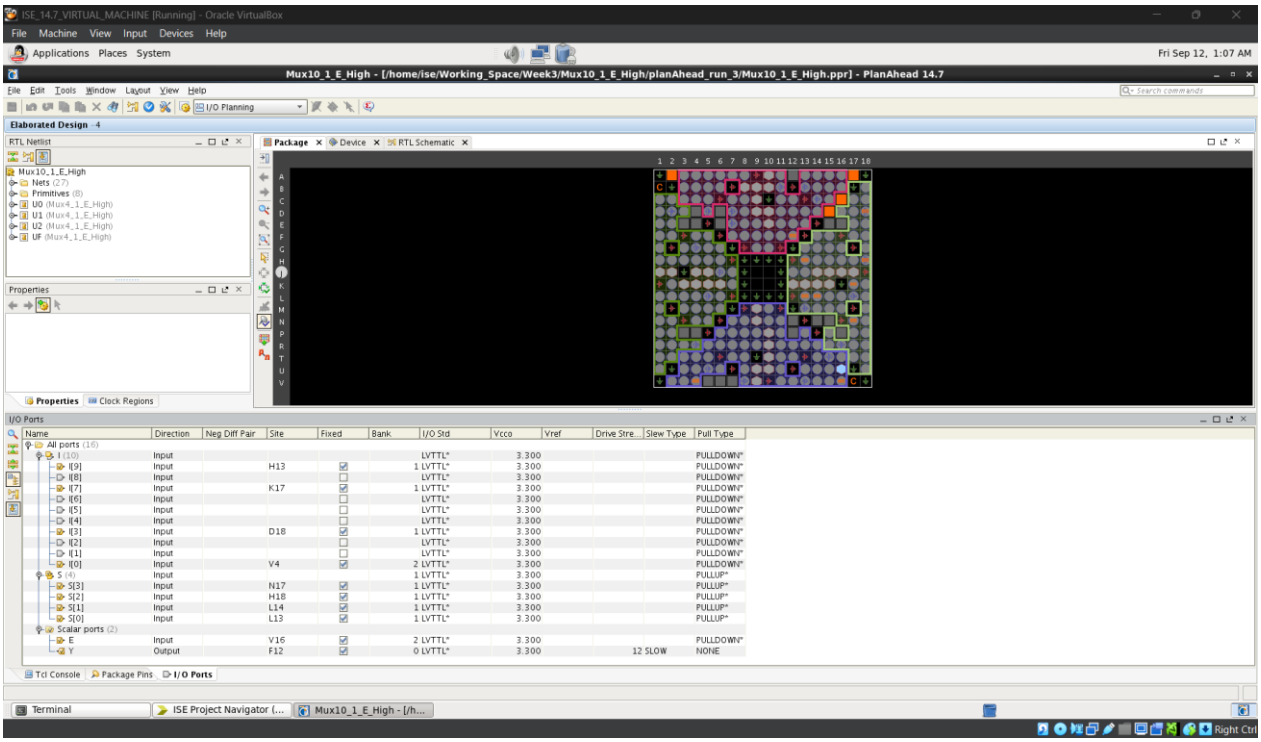
Waveform Simulation (attach/insert image):



Analysis:

Khi ngõ vào cho phép E = 1, và mạch chọn S = 4’b0100, thì mạch có ngõ ra theo tần số của ngõ vào I[4], đúng theo yêu cầu của đề và bảng trạng thái.

6. Summary



Hình 18: Cấu hình bằng phần mềm

```
# PlanAhead Generated IO constraints


NET "E" IOSTANDARD = LVTTL;

NET "Y" IOSTANDARD = LVTTL;

NET "S[3]" IOSTANDARD = LVTTL;

NET "S[2]" IOSTANDARD = LVTTL;

NET "S[1]" IOSTANDARD = LVTTL;

NET "S[0]" IOSTANDARD = LVTTL;

NET "I[9]" IOSTANDARD = LVTTL;

NET "I[8]" IOSTANDARD = LVTTL;

NET "I[7]" IOSTANDARD = LVTTL;
```

```
NET "I[6]" IOSTANDARD = LVTTL;
NET "I[5]" IOSTANDARD = LVTTL;
NET "I[4]" IOSTANDARD = LVTTL;
NET "I[3]" IOSTANDARD = LVTTL;
NET "I[2]" IOSTANDARD = LVTTL;
NET "I[1]" IOSTANDARD = LVTTL;
NET "I[0]" IOSTANDARD = LVTTL;
NET "I[9]" PULLDOWN;
NET "I[8]" PULLDOWN;
NET "I[7]" PULLDOWN;
NET "I[6]" PULLDOWN;
NET "I[5]" PULLDOWN;
NET "I[4]" PULLDOWN;
NET "I[3]" PULLDOWN;
NET "I[2]" PULLDOWN;
NET "I[1]" PULLDOWN;
NET "I[0]" PULLDOWN;
NET "S[3]" PULLUP;
NET "S[2]" PULLUP;
NET "S[1]" PULLUP;
NET "S[0]" PULLUP;
NET "E" PULLDOWN;

# PlanAhead Generated physical constraints
```

```
NET "Y" LOC = F12;

NET "S[3]" LOC = N17;

NET "S[2]" LOC = H18;

NET "S[1]" LOC = L14;

NET "S[0]" LOC = L13;

NET "E" LOC = V16;

NET "I[0]" LOC = V4;

NET "I[3]" LOC = D18;

NET "I[7]" LOC = K17;

NET "I[9]" LOC = H13;
```

WEEKLY REPORT – DIGITAL SYSTEM DESIGN USING VERILOG & FPGA

1. General Information

Project Title: MUX1_8

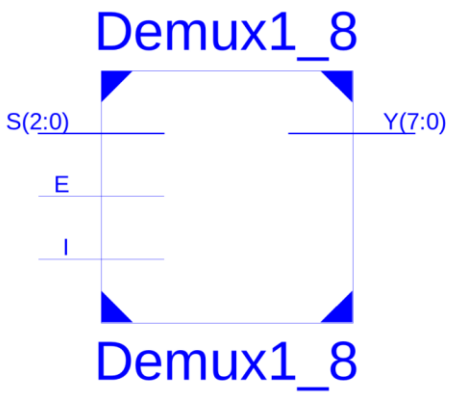
Students: Lê Quang Minh Nhật

Students ID: 23139031

Date: 12/09/2005

2. Block Diagram

Draw block diagram here.



3. State Transition Diagram

Describe Truth table or Design requirements or the FSM state diagram, etc

Input					Output							
I	**S2**	**S1**	**S0**	**E**	**Y7**	**Y6**	**Y5**	**Y4**	**Y3**	**Y2**	**Y1**	**Y0**
x	x	x	x	0	0	0	0	0	0	0	0	0
I	0	0	0	1	0	0	0	0	0	0	0	I
I	0	0	1	1	0	0	0	0	0	0	I	0
I	0	1	0	1	0	0	0	0	0	I	0	0
I	0	1	1	1	0	0	0	0	I	0	0	0
I	1	0	0	1	0	0	0	I	0	0	0	0
I	1	0	1	1	0	0	I	0	0	0	0	0
I	1	1	0	1	0	I	0	0	0	0	0	0
I	1	1	1	1	I	0	0	0	0	0	0	0

Mô đum Demux 1_8

```
`timescale 1ns / 1ps
```

```
module Demux1_8(
input wire I,
input wire [2:0] S,
input wire E,
output wire [7:0] Y
);
assign Y = (E && I) ? (8'b0000_0001 << S) : 8'b0000_0000;
endmodule
```

4. Test Cases Overview

TC ID	Purpose	Input	Expected Output	Result (Pass/Fail)
TC1	Test ngõ cho phép	E = 0	Y[7:0] = 8'b0000_0000	Pass
TC2	Test khi chọn Y[1]	E = 1 S = 3'b001	Y[1] = I[1]	Pass
TC3	Test khi chọn Y[4]	E = 1 S = 3'b100	Y[4] = I[4]	Pass

Mô đum testbench

```
`timescale 1ns / 1ps
module Testbecnh_Demux1_8;
	// Inputs
	reg I;
	reg [2:0] S;
	reg E;
	// Outputs
	wire [7:0] Y;
	// Instantiate the Unit Under Test (UUT)
	Demux1_8 uut (
		.I(I),
		.S(S),
```

```
        .E(E),
        .Y(Y)
    );
    initial begin
        // Initialize Inputs
        I = 0;
        S = 0;
        E = 0;
        // Wait 100 ns for global reset to finish
        #100;
        // Add stimulus here
        I = 1;
        E = 1;
    end

    always forever #200   S = S + 1'b1;
    always forever #25    I = ~I;
endmodule
```

5. Testcase Details

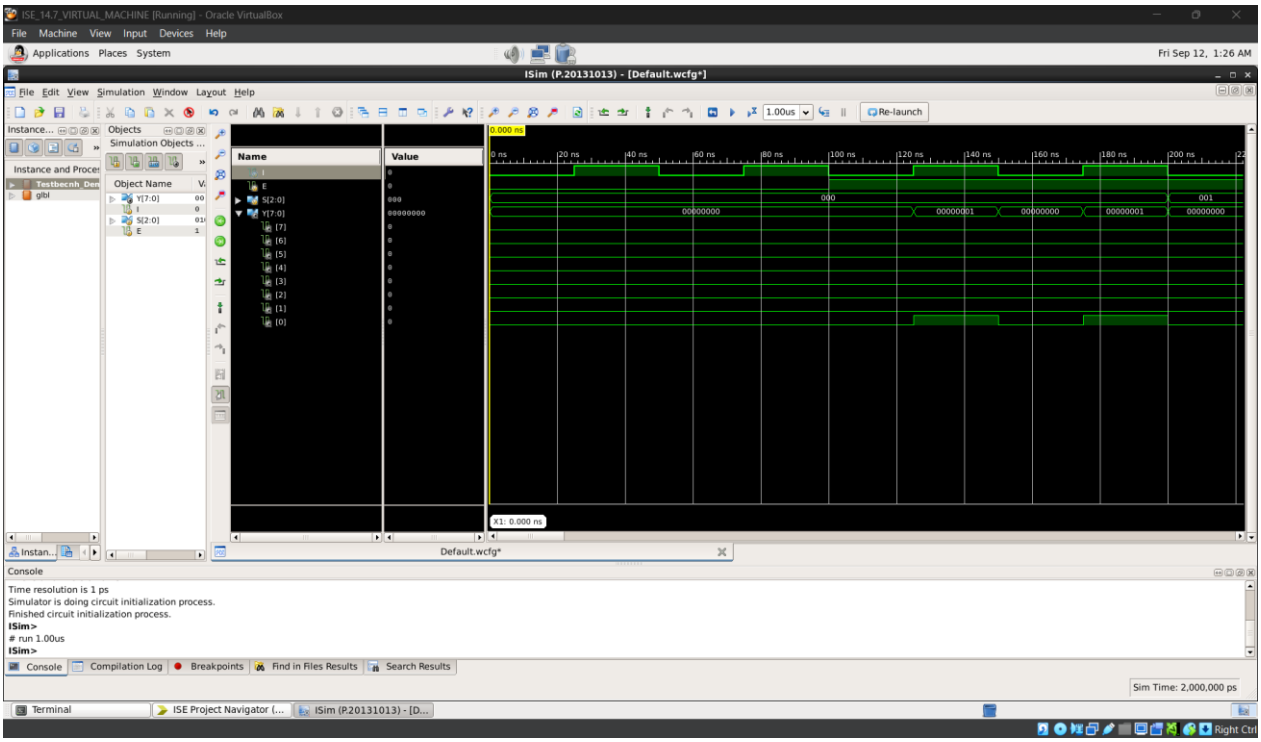
Testcase 1: TC1

Purpose: Test ngõ cho phép

Input/Stimulus: E = 0

Expected Output: Y[7:0] = 8’b0000_0000

Waveform Simulation (attach/insert image):



Analysis:

Khi ngõ vào không cho phép E = 0, ngõ ra sẽ không có tín hiệu

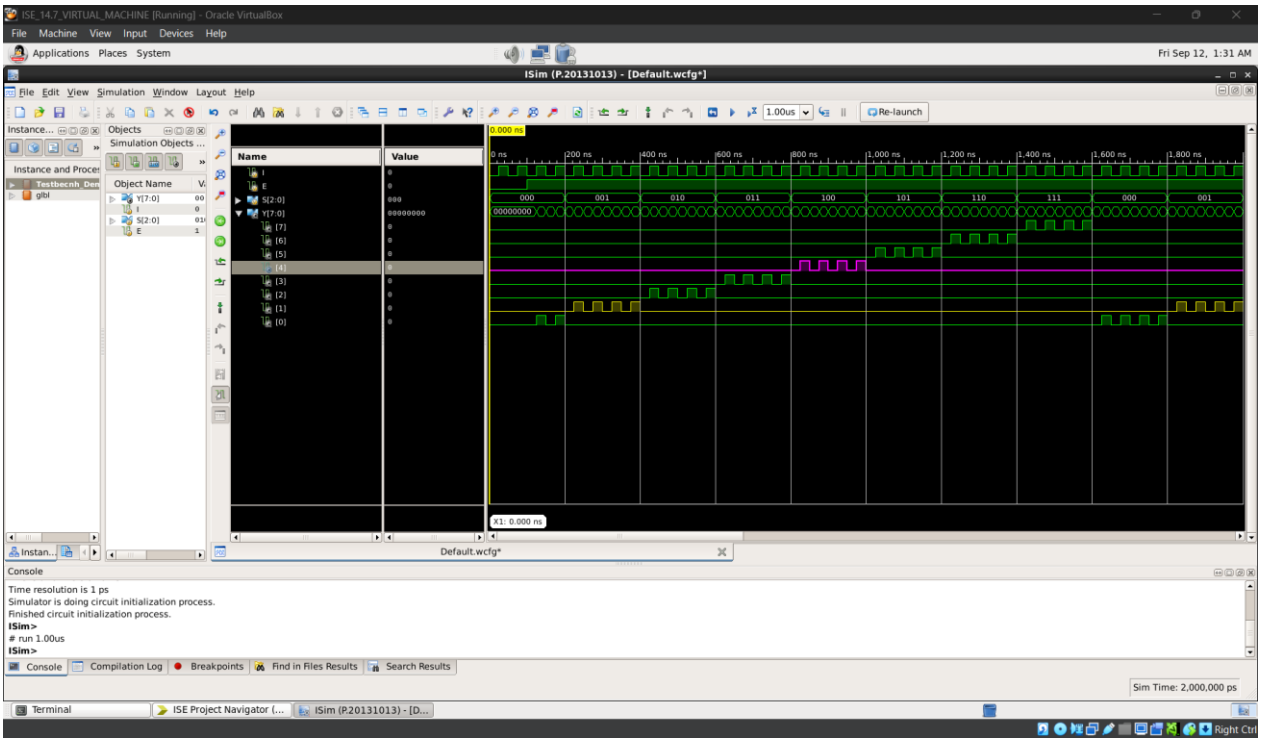
Testcase 2: TC2

Purpose: Test khi chọn Y[1]

Input/Stimulus: E = 1, S = 3’b001

Expected Output: Y[1] = I

Waveform Simulation (attach/insert image):



Analysis:

Khi ngõ vào cho phép E = 1, S = 3'b001, ngõ ra chọn ngõ vào là Y[1] = I

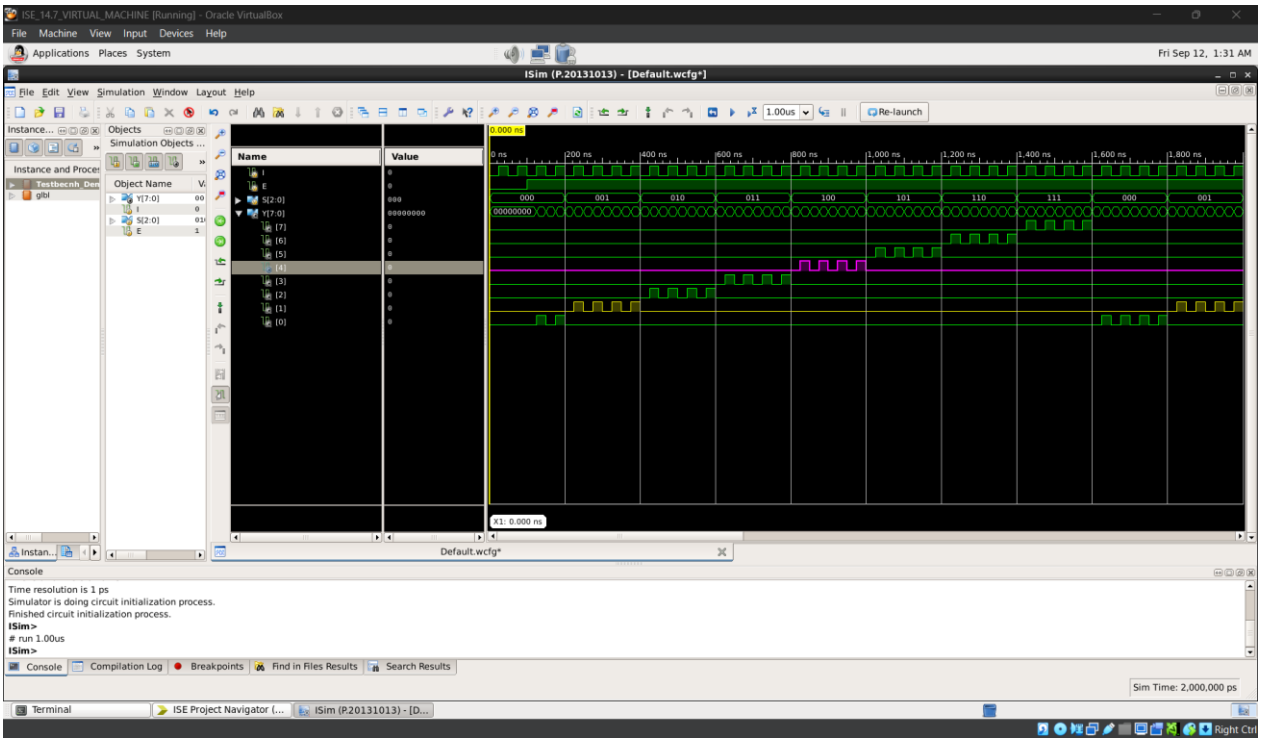
Testcase 3: TC3

Purpose: Test khi chọn Y[4]

Input/Stimulus: E = 1, S = 3'b100

Expected Output: Y[4] = I[4]

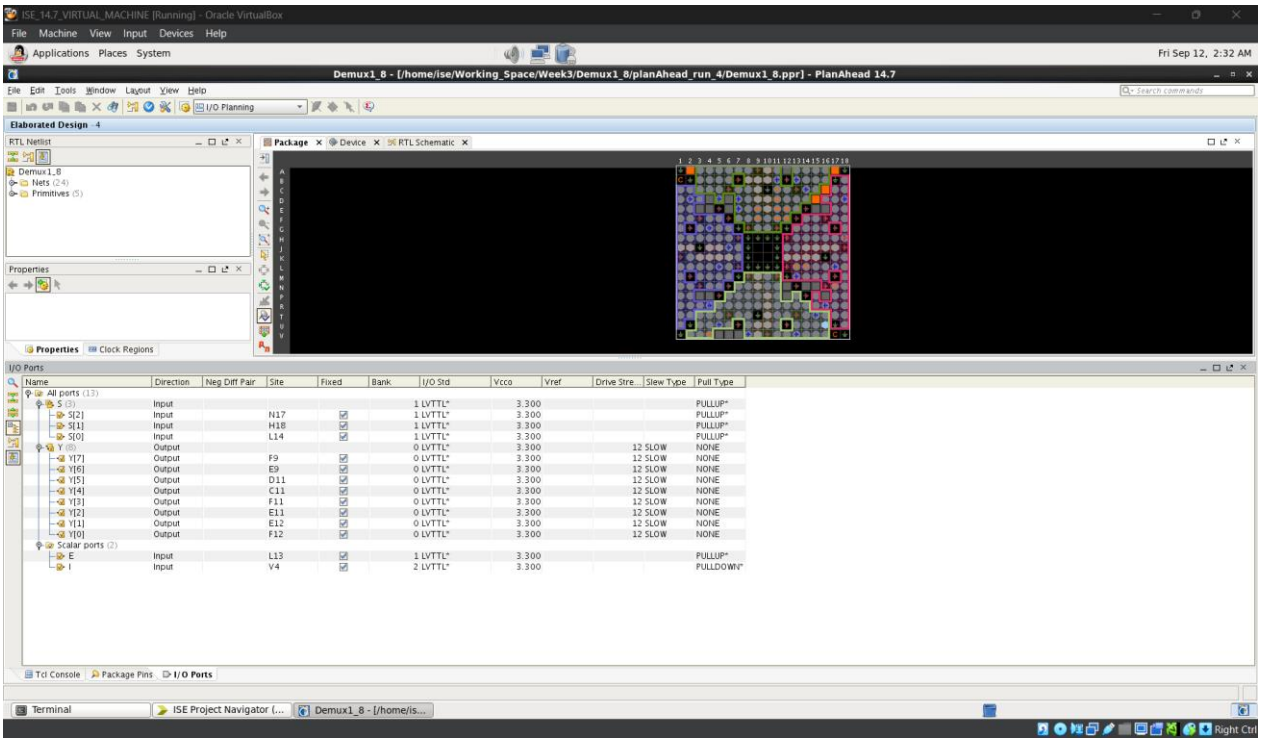
Waveform Simulation (attach/insert image):



Analysis:

Khi ngõ vào cho phép E = 1, S = 3'b100, ngõ ra chọn ngõ vào là Y[4] = I

6. Summary



Hình 19: Cấu hình bằng phần mềm

```
# PlanAhead Generated IO constraints


NET "E" IOSTANDARD = LVTTL;

NET "I" IOSTANDARD = LVTTL;

NET "Y[7]" IOSTANDARD = LVTTL;

NET "Y[6]" IOSTANDARD = LVTTL;

NET "Y[5]" IOSTANDARD = LVTTL;

NET "Y[4]" IOSTANDARD = LVTTL;

NET "Y[3]" IOSTANDARD = LVTTL;

NET "Y[2]" IOSTANDARD = LVTTL;

NET "Y[1]" IOSTANDARD = LVTTL;

NET "Y[0]" IOSTANDARD = LVTTL;
```

```
NET "S[2]" IOSTANDARD = LVTTL;
NET "S[1]" IOSTANDARD = LVTTL;
NET "S[0]" IOSTANDARD = LVTTL;

# PlanAhead Generated physical constraints

NET "S[2]" LOC = N17;
NET "S[1]" LOC = H18;
NET "S[0]" LOC = L14;
NET "E" LOC = L13;

# PlanAhead Generated IO constraints

NET "I" PULLDOWN;

# PlanAhead Generated physical constraints

NET "I" LOC = V4;
NET "Y[7]" LOC = F9;
NET "Y[6]" LOC = E9;
NET "Y[5]" LOC = D11;
NET "Y[4]" LOC = C11;
NET "Y[3]" LOC = F11;
NET "Y[2]" LOC = E11;
NET "Y[1]" LOC = E12;
```

```
NET "Y[0]" LOC = F12;


# PlanAhead Generated IO constraints


NET "S[2]" PULLUP;

NET "S[1]" PULLUP;

NET "S[0]" PULLUP;

NET "E" PULLUP;
```

WEEKLY REPORT – DIGITAL SYSTEM DESIGN USING VERILOG & FPGA

1. General Information

Project Title: Led_7_Anode

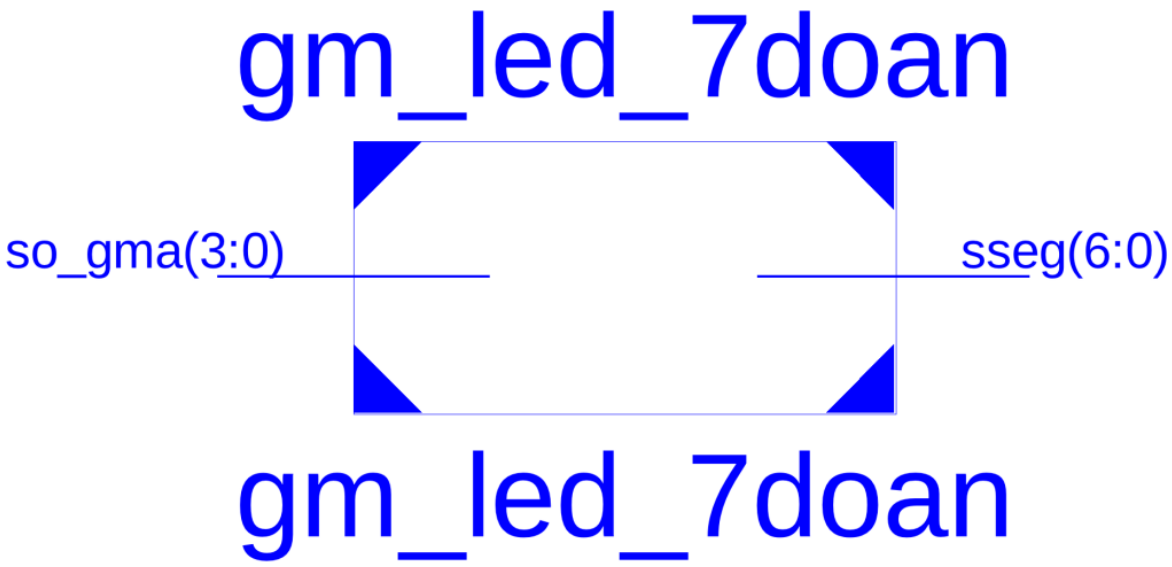
Students: Lê Quang Minh Nhật

Students ID: 23139031

Date: 12/09/2005

2. Block Diagram

Draw block diagram here.



3. State Transition Diagram

Describe Truth table or Design requirements or the FSM state diagram, etc

S3	S2	S1	S0	Số	6	5	4	3	2	1	0	HEX
					g	f	e	d	c	b	a	
0	0	0	0	0	1	0	0	0	0	0	0	C0
0	0	0	1	1	1	1	1	1	0	0	1	F9
0	0	1	0	2	0	1	0	0	1	0	0	A4

0	0	1	1	3	0	1	1	0	0	0	0	B0
0	1	0	0	4	0	0	1	1	0	0	1	99
0	1	0	1	5	0	0	1	0	0	1	0	92
0	1	1	0	6	0	0	0	0	0	1	0	82
0	1	1	1	7	0	0	0	1	1	1	1	8F
1	0	0	0	8	0	0	0	0	0	0	0	80
1	0	0	1	9	0	0	1	0	0	0	0	90
1	0	1	0	A	0	0	0	1	0	0	0	88
1	0	1	1	B	0	0	0	0	0	1	1	83
1	1	0	0	C	1	0	0	0	1	1	0	C6
1	1	0	1	D	0	1	0	0	0	0	1	A1
1	1	1	0	E	0	0	0	0	1	1	0	86
1	1	1	1	F	0	0	0	1	1	1	0	8E

Mô đum gm_led_7doan

```
`timescale 1ns / 1ps
module gm_led_7doan(
input wire [3:0] so_gma,
output reg [6:0] sseg
);
        always @* begin
                case (so_gma)
                        4'b0000: sseg = 7'b1000000;
                        4'b0001: sseg = 7'b1111001;
                        4'b0010: sseg = 7'b0100100;
                        4'b0011: sseg = 7'b0110000;
```

```
                4'b0100: sseg = 7'b0011001;
                4'b0101: sseg = 7'b0010010;
                4'b0110: sseg = 7'b0000010;
                4'b0111: sseg = 7'b1111000;
                4'b1000: sseg = 7'b0000000;
                4'b1001: sseg = 7'b0010000;
                4'b1010: sseg = 7'b0001000;
                4'b1011: sseg = 7'b0000011;
                4'b1100: sseg = 7'b1000110;
                4'b1101: sseg = 7'b0100001;
                4'b1110: sseg = 7'b0000110;
                4'b1111: sseg = 7'b0001110;
                default: sseg = 7'b1111111;
        endcase
    end
endmodule
```

4. Test Cases Overview

TC ID	Purpose	Input	Expected Output	Result (Pass/Fail)
TC1	Kiểm tra số 0	so_gma = 4’b0000	sseg = 7’b100_0000	Pass
TC2	Kiểm tra số 8	so_gma = 4’b1000	sseg = 7’b000_0000	Pass
TC3	Kiểm tra chữ F	so_gma = 4’b1111	sseg = 7’b000_1110	Pass

Mô đum testbench

```
`timescale 1ns / 1ps
module Testbench_Led_7_Anode;
    // Inputs
    reg [3:0] so_gma;
    // Outputs
```

```
	wire [6:0] sseg;
	// Instantiate the Unit Under Test (UUT)
	gm_led_7doan uut (
		.so_gma(so_gma),
		.sseg(sseg)
	);
	initial begin
		// Initialize Inputs
		so_gma = 0;
		// Wait 100 ns for global reset to finish
		#100;
		// Add stimulus here
	end

	always forever #50 so_gma = so_gma + 1'b1;
endmodule
```

5. Testcase Details

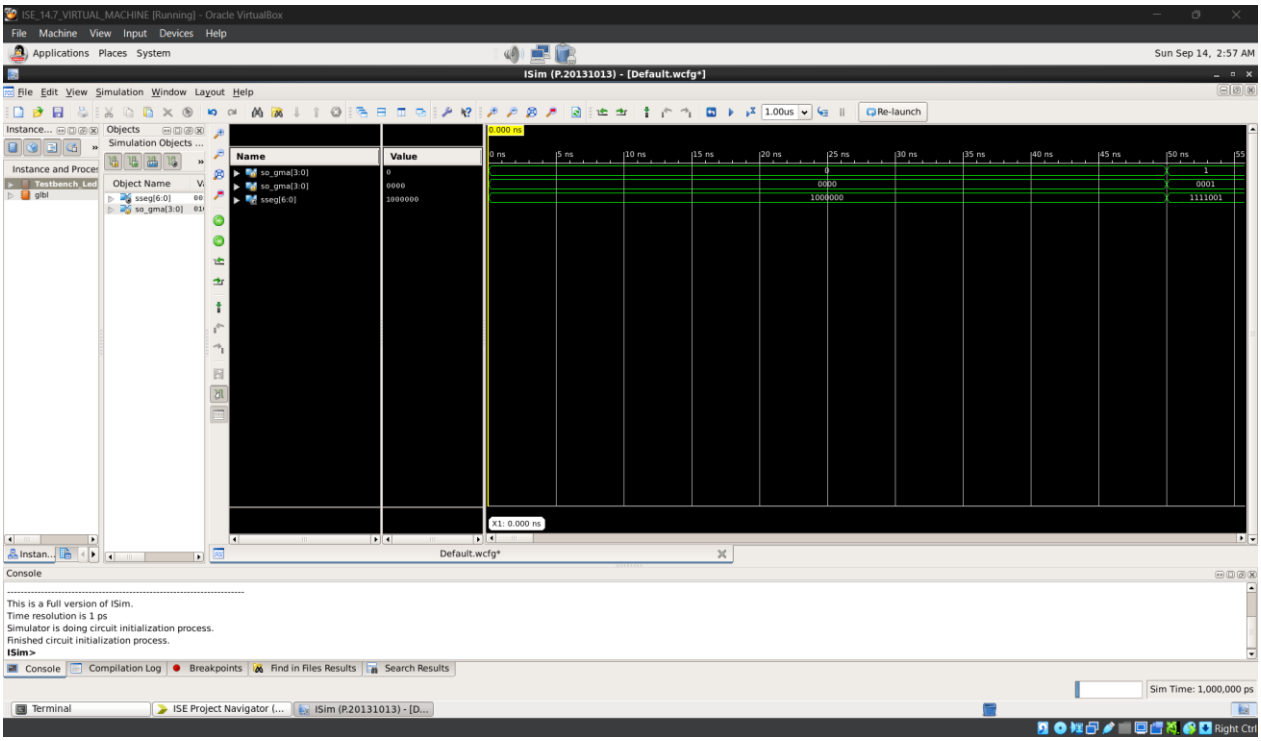
Testcase 1: TC1

Purpose: Kiểm tra số 0

Input/Stimulus: so_gma = 4’b0000

Expected Output: sseg = 7’b100_0000

Waveform Simulation (attach/insert image):



Analysis:

Khi chọn so_gma = 3’b000, giải mã ra kết quả đúng yêu cầu của bảng trạng thái

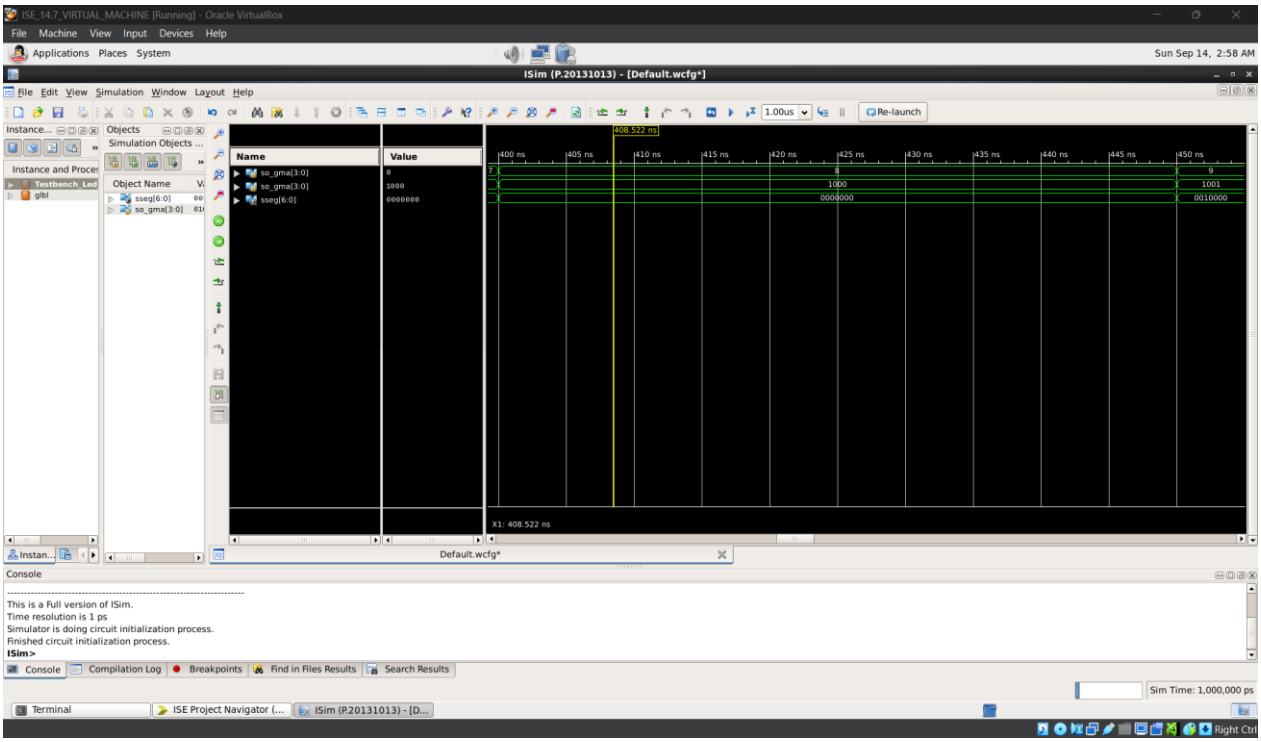
Testcase 2: TC2

Purpose: Kiểm tra số 8

Input/Stimulus: so_gma = 4’b1000

Expected Output: sseg = 7’b000_0000

Waveform Simulation (attach/insert image):



Analysis:

Khi chọn so_gma = 3'b100, giải mã ra kết quả đúng yêu cầu của bảng trạng thái

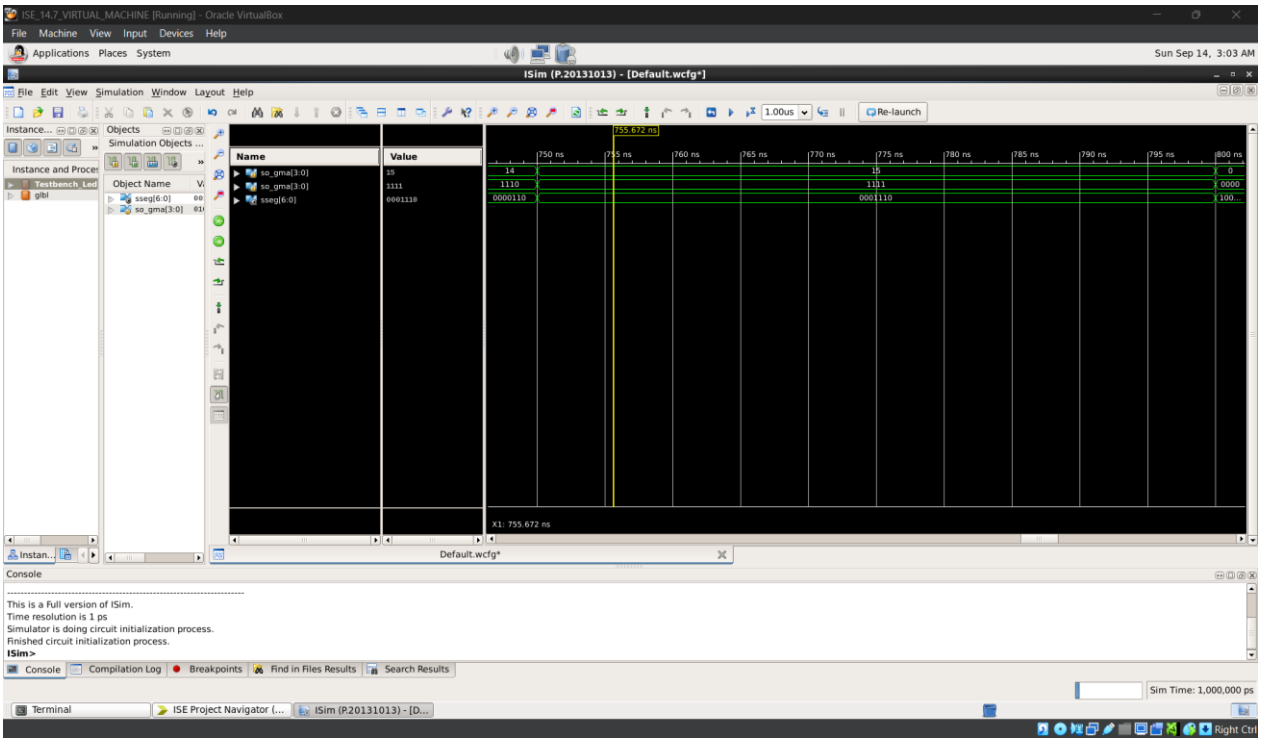
Testcase 3: TC3

Purpose: Kiểm tra chữ F

Input/Stimulus: so_gma = 4'b1111

Expected Output: sseg = 7'b000_1110

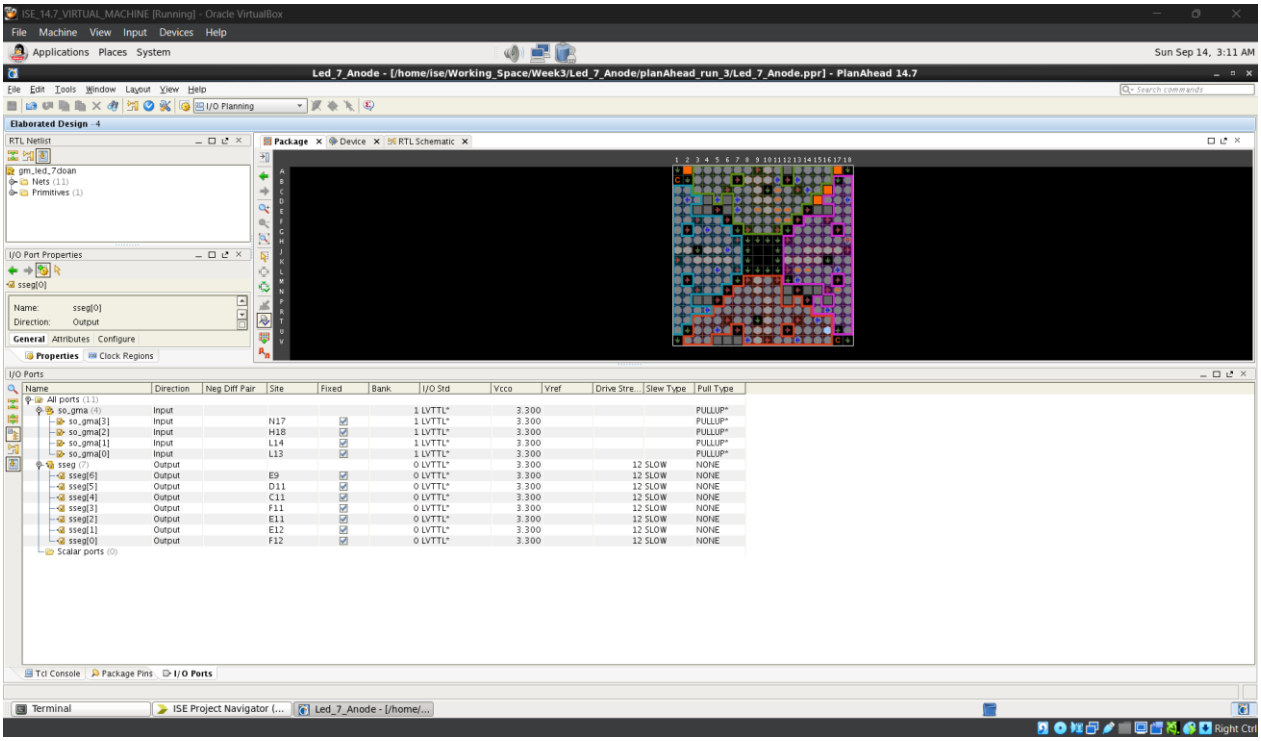
Waveform Simulation (attach/insert image):



Analysis:

Khi chọn so_gma = 3'b111, giải mã ra kết quả đúng yêu cầu của bảng trạng thái

6. Summary



```
# PlanAhead Generated IO constraints


NET "so_gma[3]" IOSTANDARD = LVTTL;

NET "so_gma[2]" IOSTANDARD = LVTTL;

NET "so_gma[1]" IOSTANDARD = LVTTL;

NET "so_gma[0]" IOSTANDARD = LVTTL;

NET "sseg[6]" IOSTANDARD = LVTTL;

NET "sseg[5]" IOSTANDARD = LVTTL;

NET "sseg[4]" IOSTANDARD = LVTTL;

NET "sseg[3]" IOSTANDARD = LVTTL;

NET "sseg[2]" IOSTANDARD = LVTTL;

NET "sseg[1]" IOSTANDARD = LVTTL;

NET "sseg[0]" IOSTANDARD = LVTTL;

NET "so_gma[3]" PULLUP;

NET "so_gma[2]" PULLUP;

NET "so_gma[1]" PULLUP;

NET "so_gma[0]" PULLUP;


# PlanAhead Generated physical constraints


NET "so_gma[3]" LOC = N17;

NET "so_gma[2]" LOC = H18;

NET "so_gma[1]" LOC = L14;

NET "so_gma[0]" LOC = L13;

NET "sseg[6]" LOC = E9;
```

```
NET "sseg[5]" LOC = D11;
NET "sseg[4]" LOC = C11;
NET "sseg[3]" LOC = F11;
NET "sseg[2]" LOC = E11;
NET "sseg[1]" LOC = E12;
NET "sseg[0]" LOC = F12;
```

WEEKLY REPORT – DIGITAL SYSTEM DESIGN USING VERILOG & FPGA

1. General Information

Project Title: clockDiv_1Hz

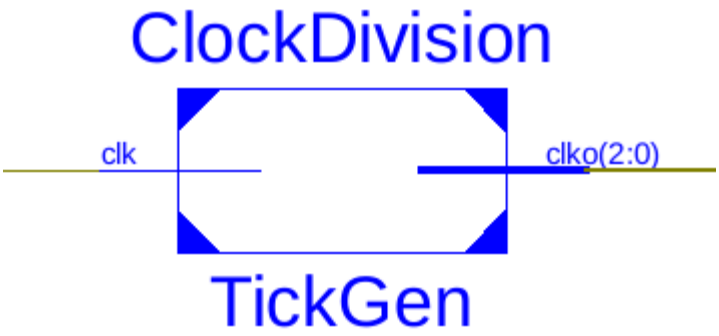
Students: Lê Quang Minh Nhật

Students ID: 23139031

Date: 19/09/2005

2. Block Diagram

Draw block diagram here.



3. State Transition Diagram

Describe Truth table or Design requirements or the FSM state diagram, etc

r_reg hiện tại	r_next	clko
0 … ⌊M/2⌋	r_reg + 1	0
⌊M/2⌋ + 1 … M−1	r_reg + 1	1
M	0	1

Mô đum clockDiv

```
`timescale 1ns / 1ps


module clockDiv

#(parameter M=50000000)

(
```

```
    input wire clki,
    output wire clko
    );


        wire [30:0] r_next;
        reg [30:0] r_reg ;


        initial r_reg =0;


        always @(posedge clki)
                r_reg <=r_next;


        assign r_next =(r_reg==M)?0:r_reg+1;
        assign clko = (r_reg<=M/2)?0:1 ;
endmodule
```

4. Test Cases Overview

TC ID	Purpose	Input	Expected Output	Result (Pass/Fail)
TC1	Kiểm tra chu kì ngõ ra	f = 50 MHz	f = 1Hz	Pass

Testbench

```
`timescale 1ns / 1ps


module Testbench_clockDiv_1Hz;


        // Inputs
        reg clki;
```

```
	// Outputs
	wire clko;

	// Instantiate the Unit Under Test (UUT)
	clockDiv_1Hz uut (
		.clki(clki),
		.clko(clko)
	);

	initial begin
		// Initialize Inputs
		clki = 0;

		// Wait 100 ns for global reset to finish
		#100;

		// Add stimulus here

	end

	always forever #10 clki = ~clki;

endmodule
```

5. Testcase Details

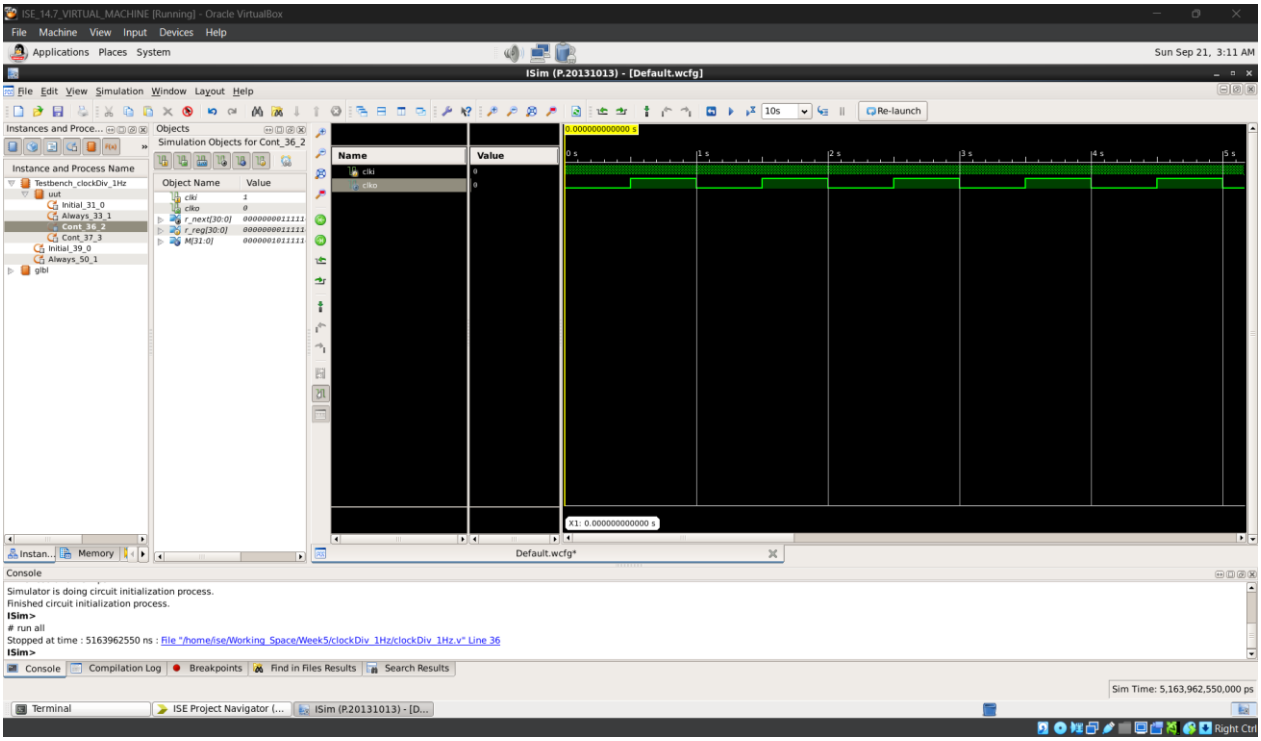
Testcase 1: TC 1

Purpose: Kiểm tra chu kì ngõ ra

Input/Stimulus: f = 50 MHz

Expected Output: f = 1 Hz

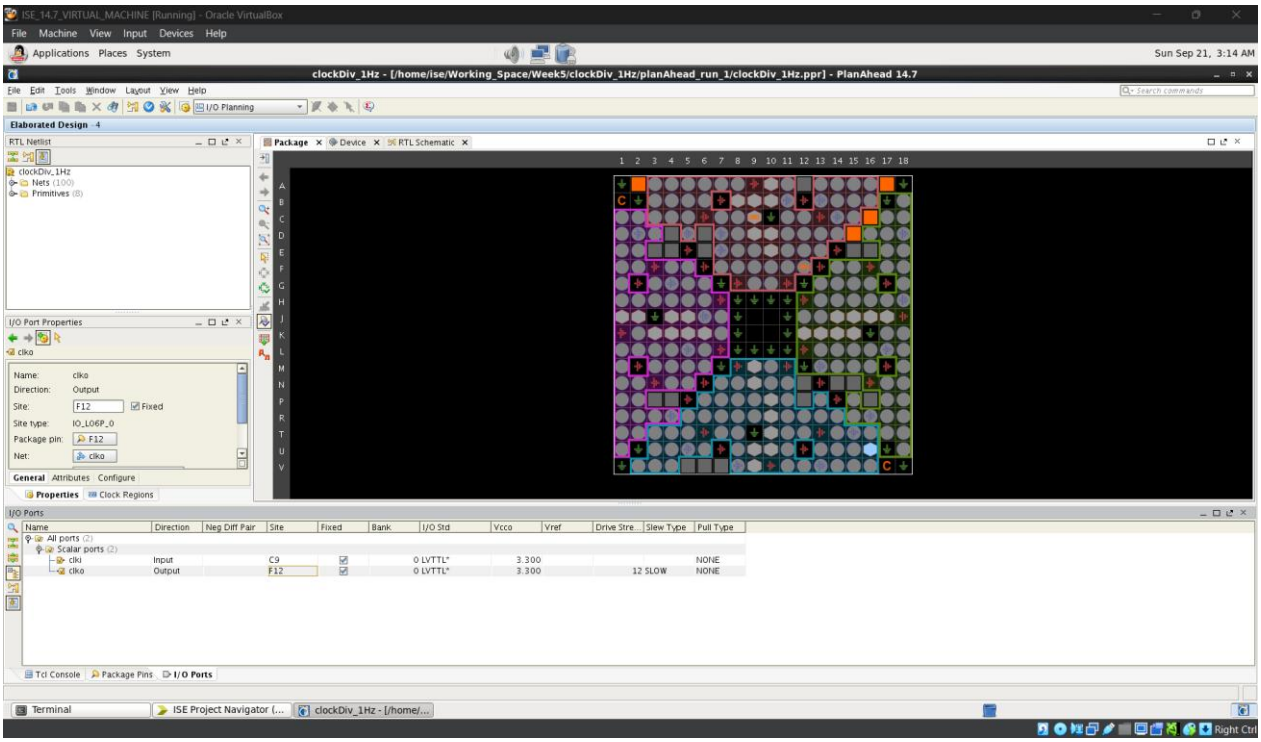
Waveform Simulation (attach/insert image):



Analysis:

Trong 1 giây dạng sóng thực hiện được 1 chu kì nên tần số ngõ ra là 1Hz

6. Summary




```
# PlanAhead Generated physical constraints

NET "clki" LOC = C9;

# PlanAhead Generated IO constraints

NET "clki" IOSTANDARD = LVTTL;
NET "clko" IOSTANDARD = LVTTL;

# PlanAhead Generated physical constraints

NET "clko" LOC = F12;
```

WEEKLY REPORT – DIGITAL SYSTEM DESIGN USING VERILOG & FPGA

1. General Information

Project Title: ClockDivision

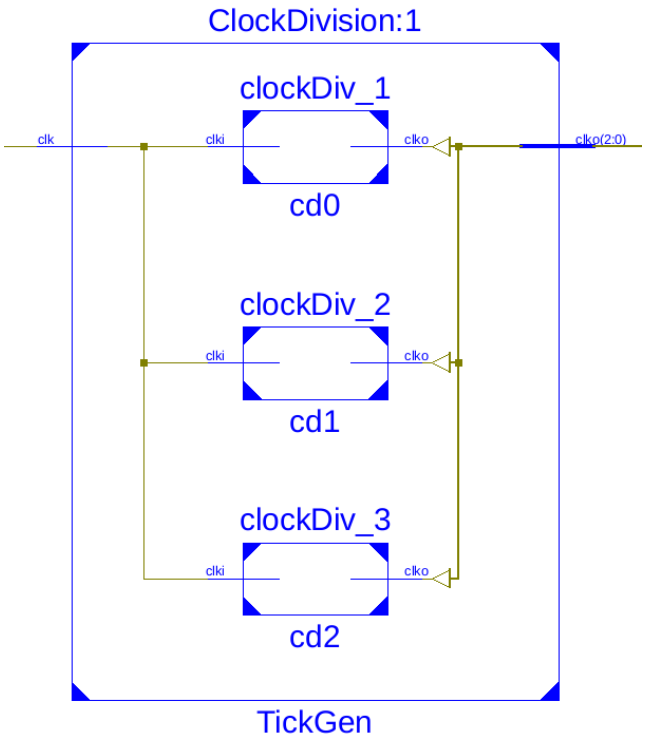
Students: Lê Quang Minh Nhật

Students ID: 23139031

Date: 19/09/2005

2. Block Diagram

Draw block diagram here.



3. State Transition Diagram

Describe Truth table or Design requirements or the FSM state diagram, etc

Bảng trạng thái cho xung Clko[0]

r_reg hiện tại	r_next	Clko[0]
0 … 50,000,000	r_reg+1	0
50,000,001 … 99,999,999	r_reg+1	1
100,000,000	0	1

Bảng trạng thái cho xung Clko[1]

r_reg hiện tại	r_next	clko
0 … 25,000,000	r_reg+1	0
25,000,001 … 49,999,999	r_reg+1	1
50,000,000	0	1

Bảng trạng thái cho xung Clko[2]

r_reg hiện tại	r_next	clko
0 … 12,500,000	r_reg+1	0
12,500,001 … 24,999,999	r_reg+1	1
25,000,000	0	1

Mô đum clockDiv

```
`timescale 1ns / 1ps


module clockDiv
#(parameter M=50000000)
(
   input wire clki,
   output wire clko
   );


        wire [30:0] r_next;
        reg [30:0] r_reg ;


        initial r_reg =0;
```

```
        always @(posedge clki)
                r_reg <=r_next;


        assign r_next =(r_reg==M)?0:r_reg+1;
        assign clko = (r_reg<=M/2)?0:1 ;


endmodule
```

Mô đum ClockDivision

```
`timescale 1ns / 1ps


module ClockDivision(
    input wire clki,
    output wire [2:0] clko
    );


        clockDiv #(100000000) cd0 (.clki(clki), .clko(clko[0]));
        clockDiv #(50000000) cd1 (.clki(clki), .clko(clko[1]));
        clockDiv #(25000000) cd2 (.clki(clki), .clko(clko[2]));


endmodule
```

4. Test Cases Overview

TC ID	Purpose	Input	Expected Output	Result (Pass/Fail)
TC1	Kiểm tra ngõ ra Clko[0]	f = 50MHz	f = 0.5 Hz	Pass
TC2	Kiểm tra ngõ ra Clko[1]	f = 50MHz	f = 1 Hz	Pass
TC3	Kiểm tra ngõ ra Clko[1]	f = 50MHz	f = 2 Hz	Pass

Testbench

```
`timescale 1ns / 1ps
```

```
module Testbench_ClockDivision;

	// Inputs
	reg [0:0] clki;

	// Outputs
	wire [2:0] clko;

	// Instantiate the Unit Under Test (UUT)
	ClockDivision uut (
		.clki(clki),
		.clko(clko)
	);

	initial begin
		// Initialize Inputs
		clki = 0;

		// Wait 100 ns for global reset to finish
		#100;

		// Add stimulus here

	end
```

```
    always forever #10 clki = ~clki;


endmodule
```

5. Testcase Details

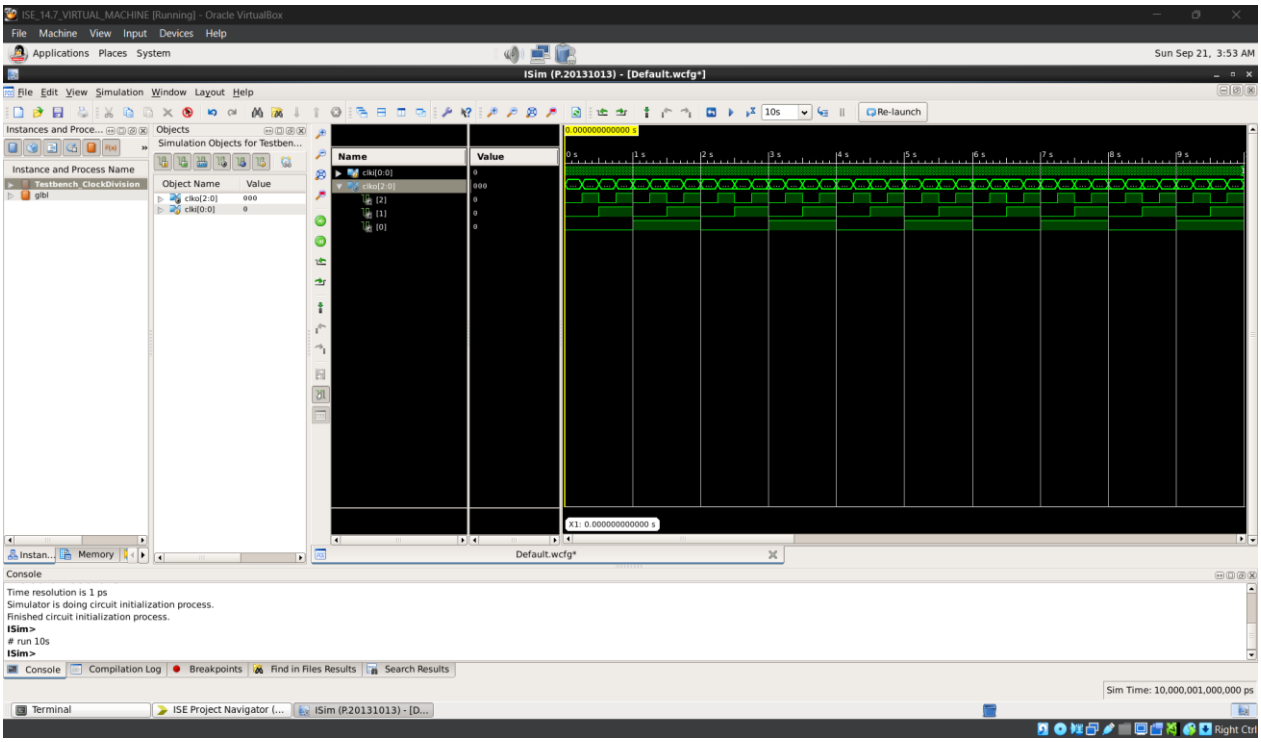
Testcase 1: TC1

Purpose: Kiểm tra ngõ ra Clko[0]

Input/Stimulus: f = 50MHz

Expected Output: f = 0.5 Hz

Waveform Simulation (attach/insert image):



Analysis:

Trong 1 giây dạng sóng thực hiện được 2 chu kì nên tần số ngõ ra là 0.5Hz

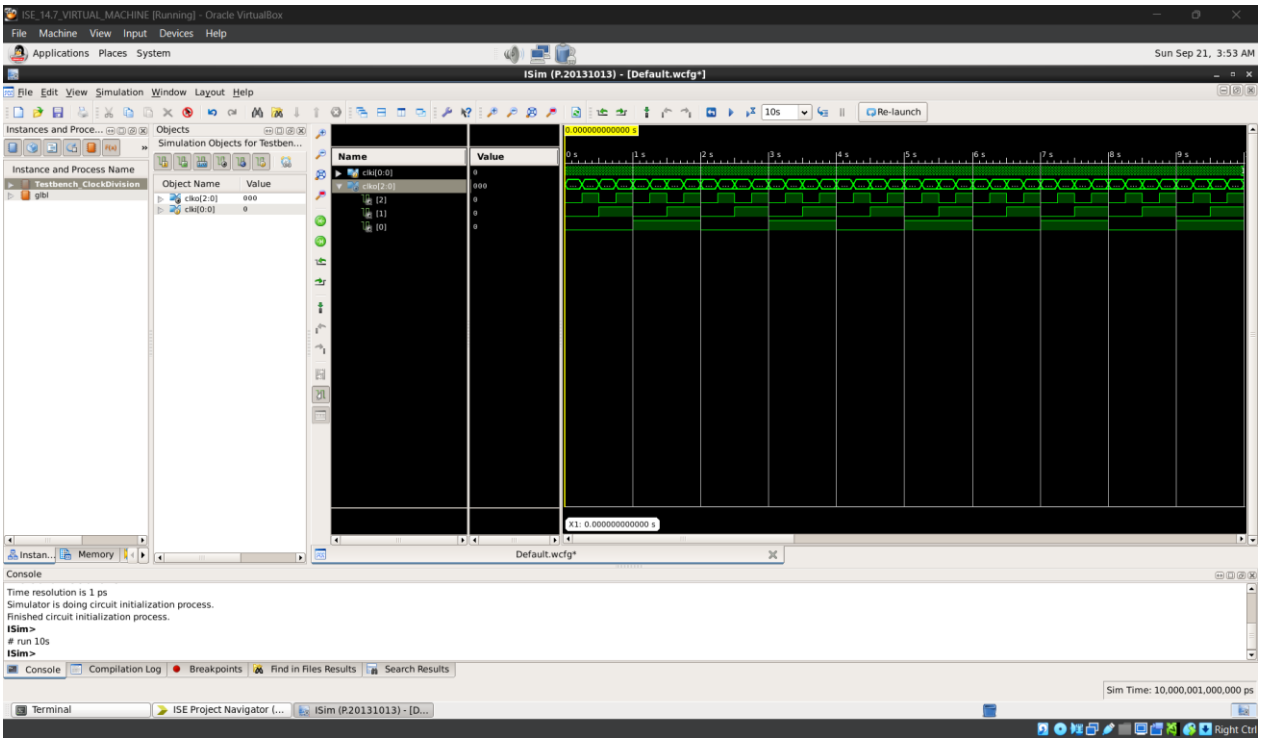
Testcase 2: TC2

Purpose: Kiểm tra ngõ ra Clko[1]

Input/Stimulus: f = 50MHz

Expected Output: f = 1 Hz

Waveform Simulation (attach/insert image):



Analysis:

Trong 1 giây dạng sóng thực hiện được 1 chu kì nên tần số ngõ ra là 1Hz

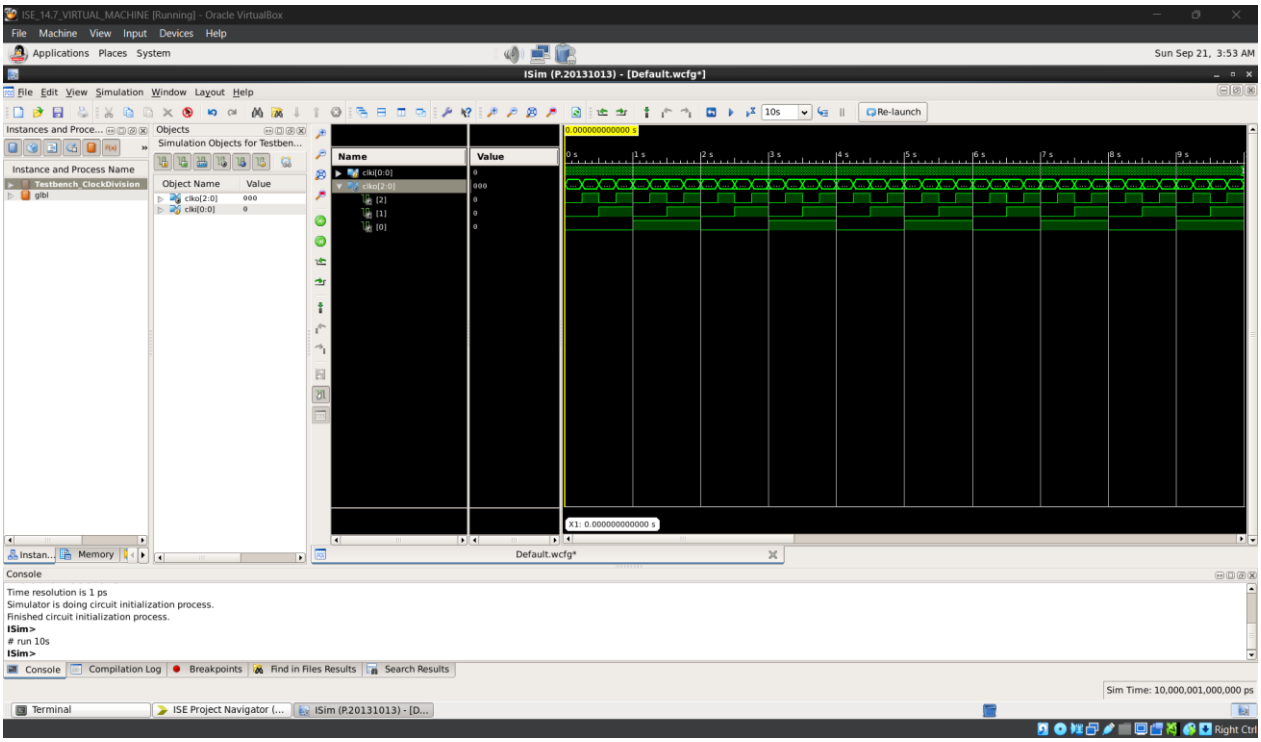
Testcase 3: TC3

Purpose: Kiểm tra ngõ ra Clko[2]

Input/Stimulus: f = 50MHz

Expected Output: f = 2 Hz

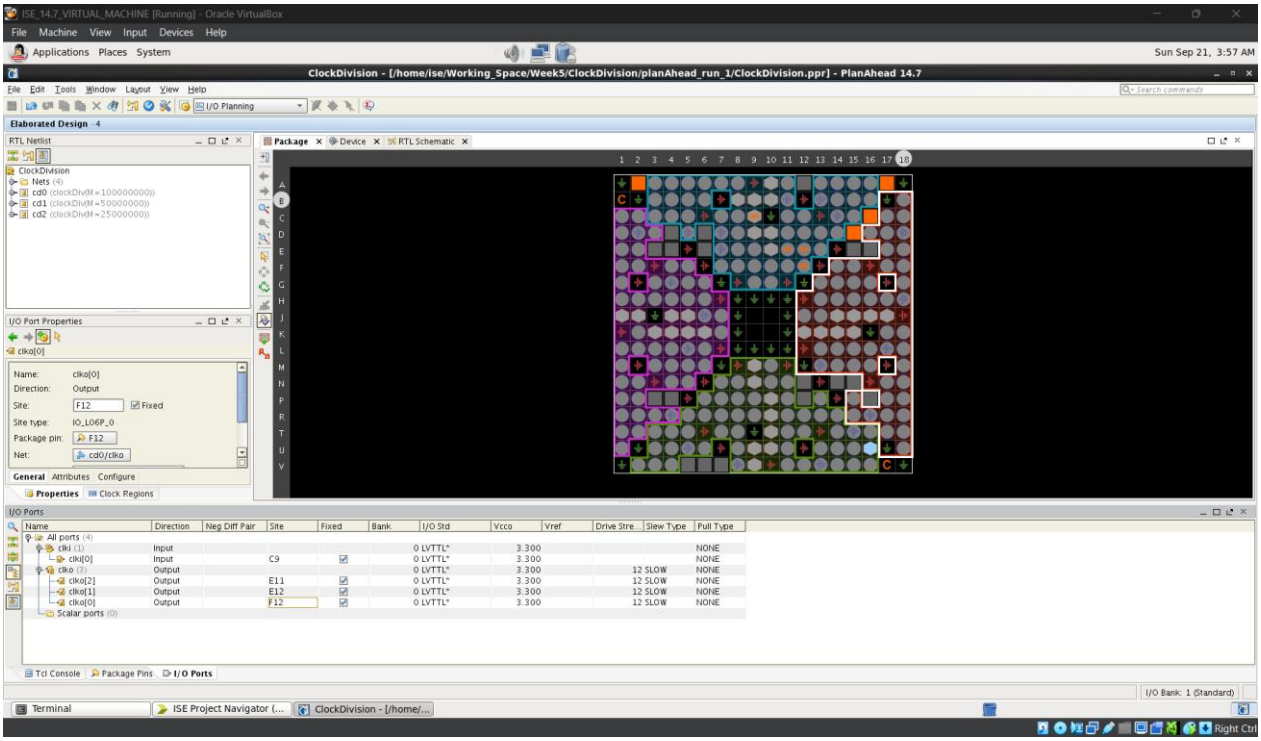
Waveform Simulation (attach/insert image):



Analysis:

Trong 2 giây dạng sóng thực hiện được 1 chu kì nên tần số ngõ ra là 2Hz

6. Summary



```
# PlanAhead Generated physical constraints


NET "clki[0]" LOC = C9;


# PlanAhead Generated IO constraints


NET "clki[0]" IOSTANDARD = LVTTL;

NET "clko[2]" IOSTANDARD = LVTTL;

NET "clko[1]" IOSTANDARD = LVTTL;

NET "clko[0]" IOSTANDARD = LVTTL;


# PlanAhead Generated physical constraints


NET "clko[2]" LOC = E11;

NET "clko[1]" LOC = E12;

NET "clko[0]" LOC = F12;
```

WEEKLY REPORT – DIGITAL SYSTEM DESIGN USING VERILOG & FPGA

1. General Information

Project Title: Chon_Xung_Ngo_Ra

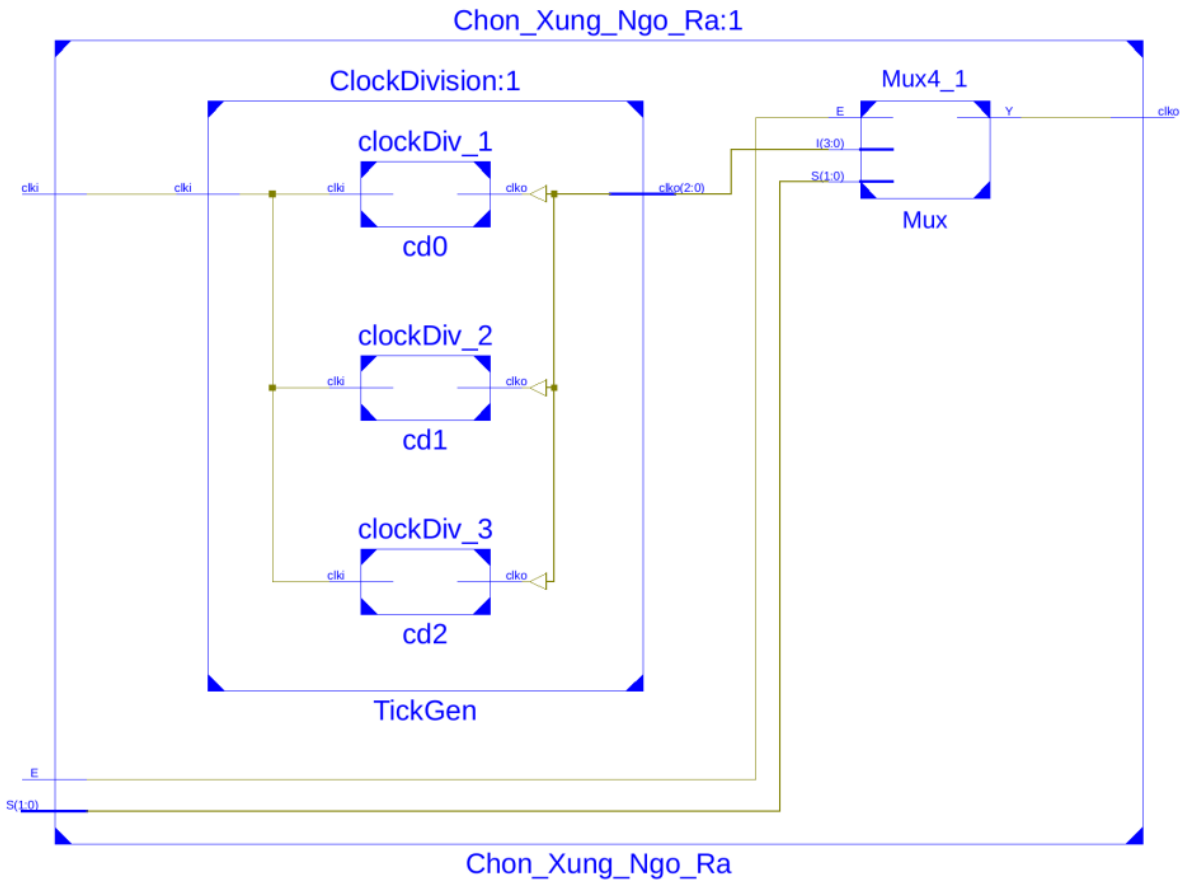
Students: Lê Quang Minh Nhật

Students ID: 23139031

Date: 19/09/2005

2. Block Diagram

Draw block diagram here.



3. State Transition Diagram

Describe Truth table or Design requirements or the FSM state diagram, etc

E	S1	S0	clko lấy từ	Tham số M
0	x	x	0	–

1	0	0	tick0p5	100,000,000
1	0	1	tick1	50,000,000
1	1	0	tick2	25,000,000
1	1	1	0	–

Mô đum clockDiv

```
`timescale 1ns / 1ps


module clockDiv
#(parameter M=50000000)
(
   input wire clki,
   output wire clko
   );


        wire [30:0] r_next;
        reg [30:0] r_reg ;


        initial r_reg =0;


        always @(posedge clki)
                r_reg <=r_next;


        assign r_next =(r_reg==M)?0:r_reg+1;
        assign clko = (r_reg<=M/2)?0:1 ;
```

```
endmodule
```

Mô đum ClockDivision

```
`timescale 1ns / 1ps


module ClockDivision(
    input wire clki,
    output wire [2:0] clko
    );


        clockDiv #(100000000) cd0 (.clki(clki), .clko(clko[0]));
        clockDiv #(50000000) cd1 (.clki(clki), .clko(clko[1]));
        clockDiv #(25000000) cd2 (.clki(clki), .clko(clko[2]));


endmodule
```

Mô đum Mux4_1

```
`timescale 1ns / 1ps


module Mux4_1(
    input wire [3:0] I,
    input wire [1:0] S,
    input wire E,
    output reg Y
    );
        always @* begin
                if (E == 1'b0) begin
```

```
                Y = 1'b0;
            end
            else begin
                case(S)
                    2'b00: Y = I[0];
                    2'b01: Y = I[1];
                    2'b10: Y = I[2];
                    2'b11: Y = I[3];
                    default: Y = 0;
                endcase
            end
        end

endmodule
```

Mô đum Chon_Xung_Ngo_Ra

```
`timescale 1ns / 1ps


module Chon_Xung_Ngo_Ra(
    input wire clki,
    input wire [1:0] S,
    input wire E,
    output wire clko
    );


        wire tick0p5, tick1, tick2;
```

```
    ClockDivision TickGen(.clki(clki), .clko({tick2, tick1, tick0p5}));

    Mux4_1 Mux (.I({1'b0, tick2, tick1, tick0p5}), .S(S), .E(E), .Y(clko));

endmodule
```

4. Test Cases Overview

TC ID	Purpose	Input	Expected Output	Result (Pass/Fail)
TC1	Chọn xung f = 0.5 Hz	E = 1 S = 2’b00 clki = 50 MHz	clko = 0.5 Hz	Pass
TC2	Chọn xung f = 1 Hz	E = 1 S = 2’b01 clki = 50 MHz	clko = 1 Hz	Pass
TC3	Chọn xung f = 2 Hz	E = 1 S = 2’b10 clki = 50 MHz	clko = 2 Hz	Pass

Testbench

```
`timescale 1ns / 1ps

module Testbench_Chon_Xung_Ngo_Ra;

    // Inputs
    reg clki;
    reg [1:0] S;
    reg E;

    // Outputs
    wire clko;
```

```
// Instantiate the Unit Under Test (UUT)
Chon_Xung_Ngo_Ra uut (
        .clki(clki),
        .S(S),
        .E(E),
        .clko(clko)
);

localparam time ONE_S  = 1_000_000_000;   // 1 s
localparam time TEN_S  = 10*ONE_S;        // 10 s

initial begin
        // Initialize Inputs
        clki = 0;
        S = 0;
        E = 0;

        // Wait 1s for global reset to finish
        #(ONE_S);

        // Add stimulus here
        E = 1;
        #(TEN_S);
        S = 1;
```

```
            #(TEN_S);
            S = 2;
            #(TEN_S);
            S = 3;
            #(TEN_S);
            S = 0;

        end

        always forever #10 clki = ~clki;

endmodule
```

5. Testcase Details

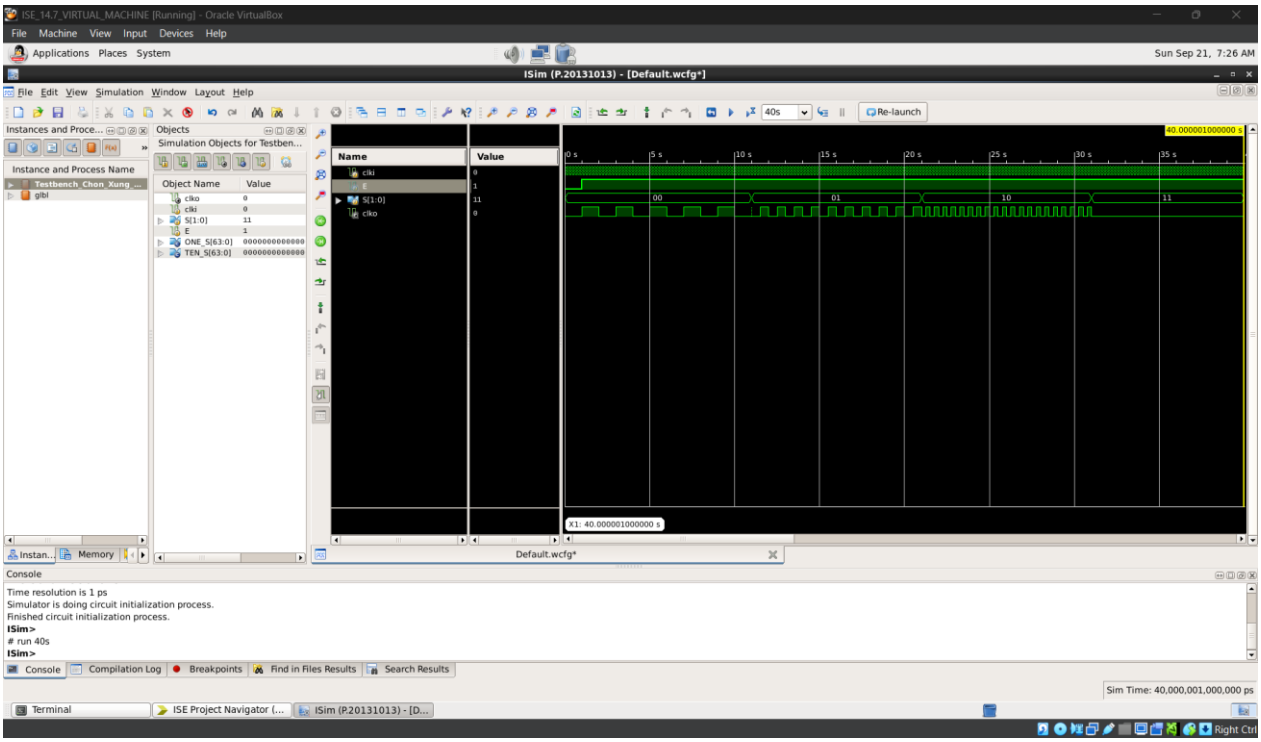
Testcase 1: TC1

Purpose: Chọn xung f = 0.5 Hz

Input/Stimulus: E = 1, S = 2’b00, clki = 50 MHz

Expected Output: clko = 0.5 Hz

Waveform Simulation (attach/insert image):



Analysis:

Ngõ vào E = 1, S = 2'b00, clki = 50 MHz đúng yêu cầu đề bài

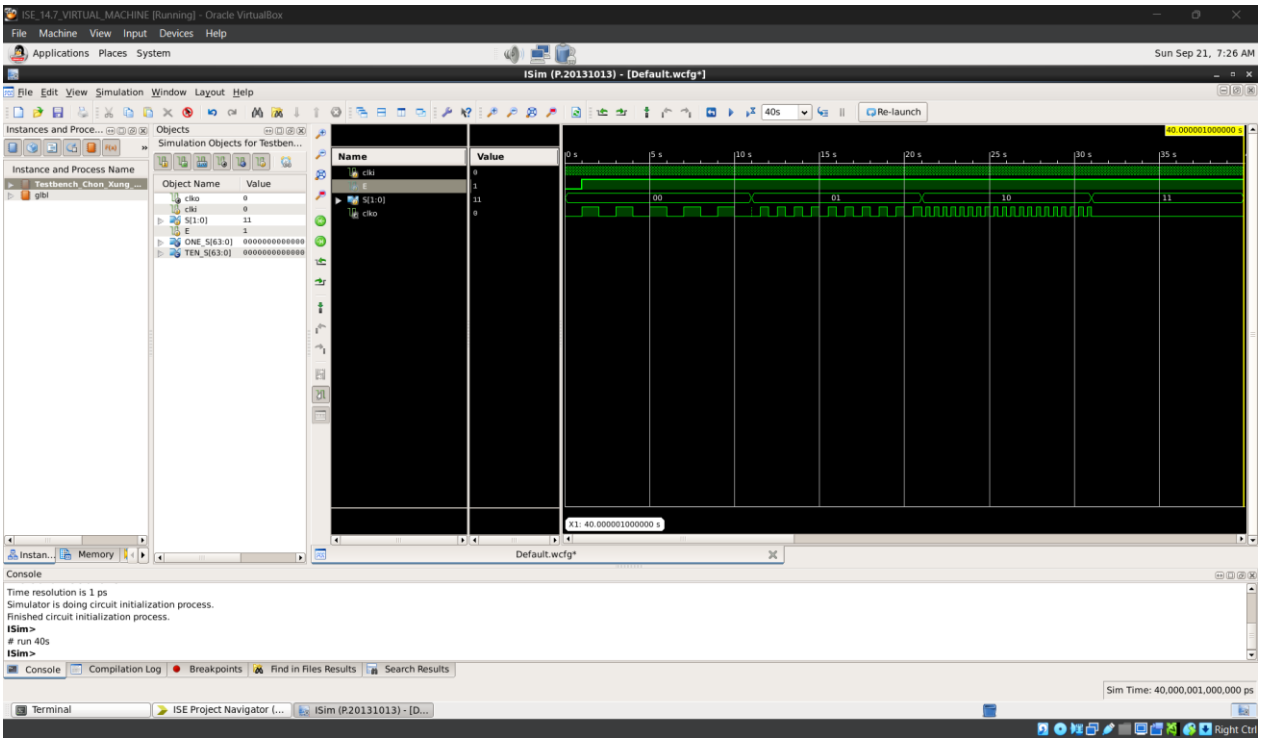
Testcase 2: TC2

Purpose: Chọn xung f = 1 Hz

Input/Stimulus: E = 1, S = 2'b01, clki = 50 MHz

Expected Output: clko = 0.5 Hz

Waveform Simulation (attach/insert image):



Analysis:

Ngõ vào E = 1, S = 2'b01, clki = 50 MHz đúng yêu cầu đề bài

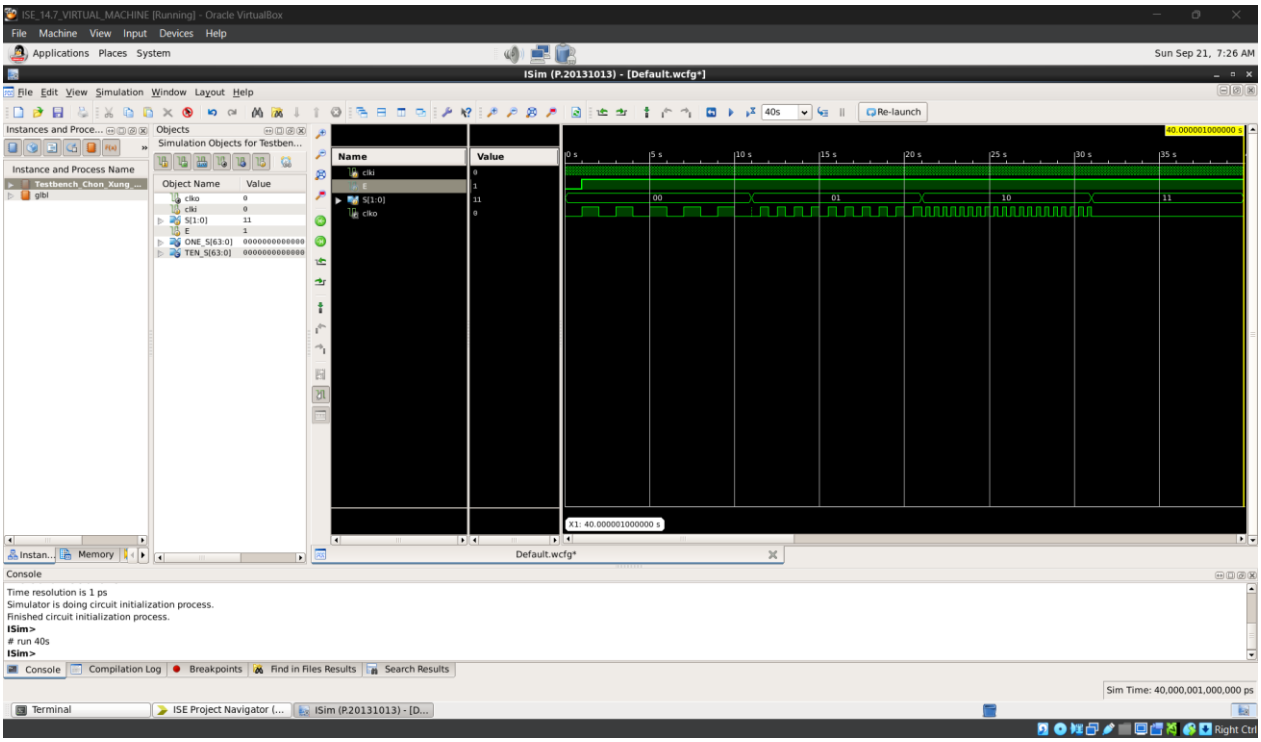
Testcase 3: TC3

Purpose: Chọn xung f = 2 Hz

Input/Stimulus: E = 1, S = 2'b10, clki = 50 MHz

Expected Output: clko = 0.5 Hz

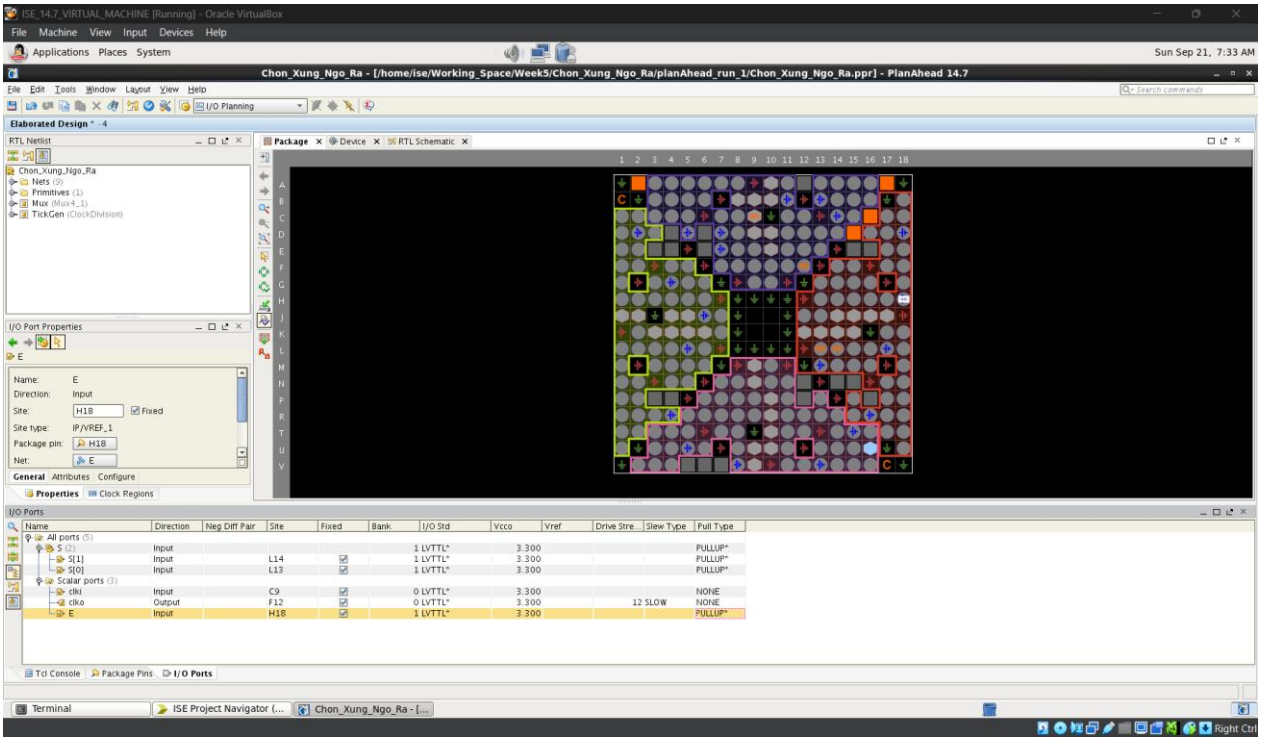
Waveform Simulation (attach/insert image):



Analysis:

Ngõ vào E = 1, S = 2'b10, clki = 50 MHz đúng yêu cầu đề bài

6. Summary



```
# PlanAhead Generated physical constraints


NET "clki" LOC = C9;


# PlanAhead Generated IO constraints


NET "clki" IOSTANDARD = LVTTL;

NET "clko" IOSTANDARD = LVTTL;

NET "E" IOSTANDARD = LVTTL;

NET "S[1]" IOSTANDARD = LVTTL;

NET "S[0]" IOSTANDARD = LVTTL;


# PlanAhead Generated physical constraints


NET "S[1]" LOC = L14;

NET "S[0]" LOC = L13;

NET "E" LOC = H18;

NET "clko" LOC = F12;


# PlanAhead Generated IO constraints


NET "S[1]" PULLUP;

NET "S[0]" PULLUP;

NET "E" PULLUP;
```

WEEKLY REPORT – DIGITAL SYSTEM DESIGN USING VERILOG & FPGA

1. General Information

Project Title: Dem_0_9

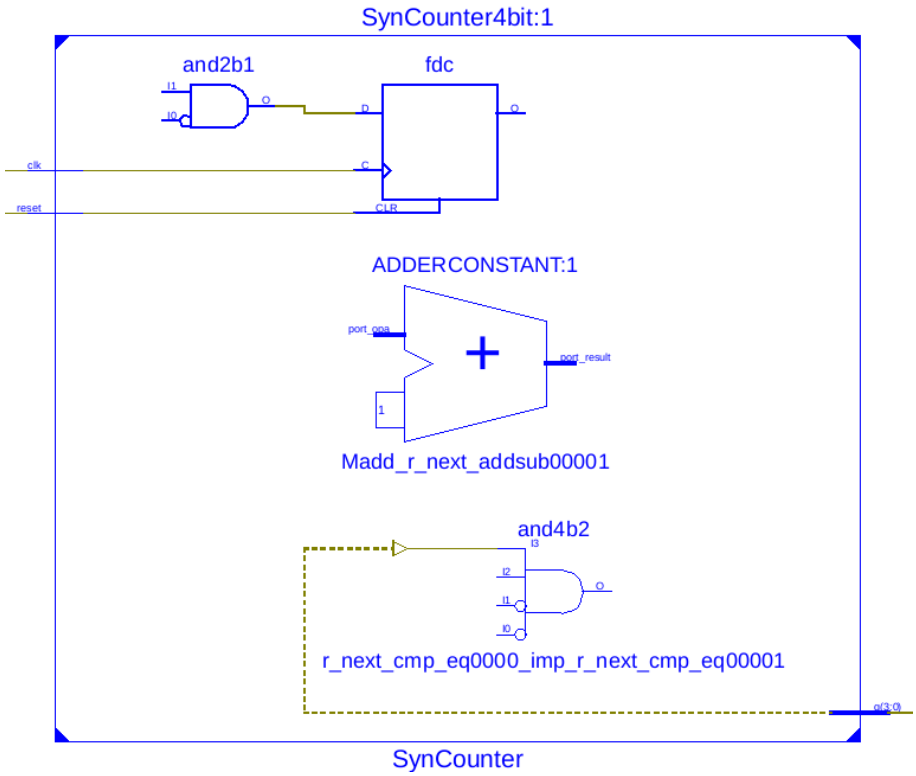
Students: Lê Quang Minh Nhật

Students ID: 23139031

Date: 19/09/2005

2. Block Diagram

Draw block diagram here.



3. State Transition Diagram

Describe Truth table or Design requirements or the FSM state diagram, etc

reset	r_reg (hiện tại)	r_next (sau posedge)	q
1	*	0	0
0	0	1	0
0	1	2	1

0	2	3	2
0	3	4	3
0	4	5	4
0	5	6	5
0	6	7	6
0	7	8	7
0	8	9	8
0	9	0	9
0	10	11	10
0	11	12	11
0	12	13	12
0	13	14	13
0	14	15	14
0	15	0	15

Mô đum clockDiv

```
module clockDiv
#(parameter M=50000000)
(
   input wire clki,
   output wire clko
   );


        wire [30:0] r_next;
        reg [30:0] r_reg ;
```

```
        initial r_reg =0;


        always @(posedge clki)
                r_reg <=r_next;


        assign r_next =(r_reg==M)?0:r_reg+1;
        assign clko = (r_reg<=M/2);


endmodule
```

Mô đum SynCounter4bit

```
`timescale 1ns / 1ps


module SynCounter4bit(
   input wire clk,
   input wire reset,
   output wire [3:0] q
   );


        // signal declaration
        reg [3:0] r_reg;
        wire [3:0] r_next;


        // body, register
        always @(posedge clk, posedge reset)
                if (reset)
```

```
                r_reg <= 0;
            else
                r_reg<=r_next; // <= is non-blocking statement

    // next state logic
    assign r_next = (r_reg == 4'd9) ? 4'd0 : (r_reg + 1);

    // output logic
    assign q = r_reg;
endmodule
```

Mô đum Dem_0_9

```
module Dem_0_9(
   input wire clk,
   input wire reset,
   output wire [3:0] q
   );

    wire clk_1Hz;

    clockDiv gen (.clki(clk) ,.clko(clk_1Hz));

    SynCounter4bit Up (.clki(clk_1Hz), .reset(reset), .q(q));

endmodule
```

4. Test Cases Overview

TC ID	Purpose	Input	Expected Output	Result (Pass/Fail)
TC1	Kiểm tra reset	E = 1 → 0 Clki	Ngõ ra bắt đầu đếm	Pass
TC2	Kiểm tra đếm lên	E = 0 Clki	Đếm lên đúng yêu cầu	Pass
TC3	Kiểm tra overflow	E = 0 Clki	Từ 9 về lại 0	Pass

Testbench

```
`timescale 1ns / 1ps


module Testbench_SynCounter4bit;


	// Inputs
	reg clki;
	reg reset;

	// Outputs
	wire [3:0] q;

	// Instantiate the Unit Under Test (UUT)
	SynCounter4bit uut (
		.clki(clki),
		.reset(reset),
		.q(q)
	);

	initial begin
```

```
        // Initialize Inputs
        clki = 0;
        reset = 0;

        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
        reset = 1;
        #100;
        reset = 0;

    end

    always forever #20 clki = ~clki;

endmodule
```

5. Testcase Details

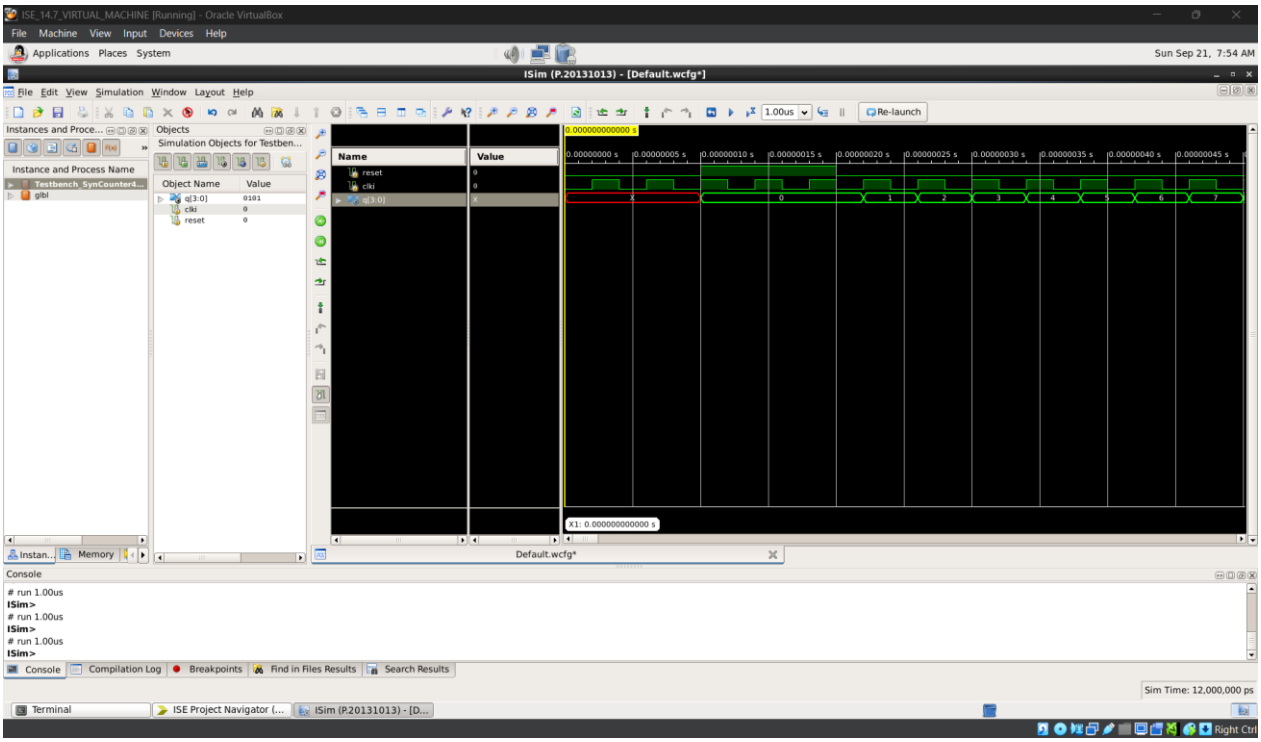
Testcase 1: TC1

Purpose: Kiểm tra reset

Input/Stimulus: E = 1 → 0, Clki

Expected Output: Ngõ ra bắt đầu đếm

Waveform Simulation (attach/insert image):



Analysis:

Ngõ ra đúng mong đợi

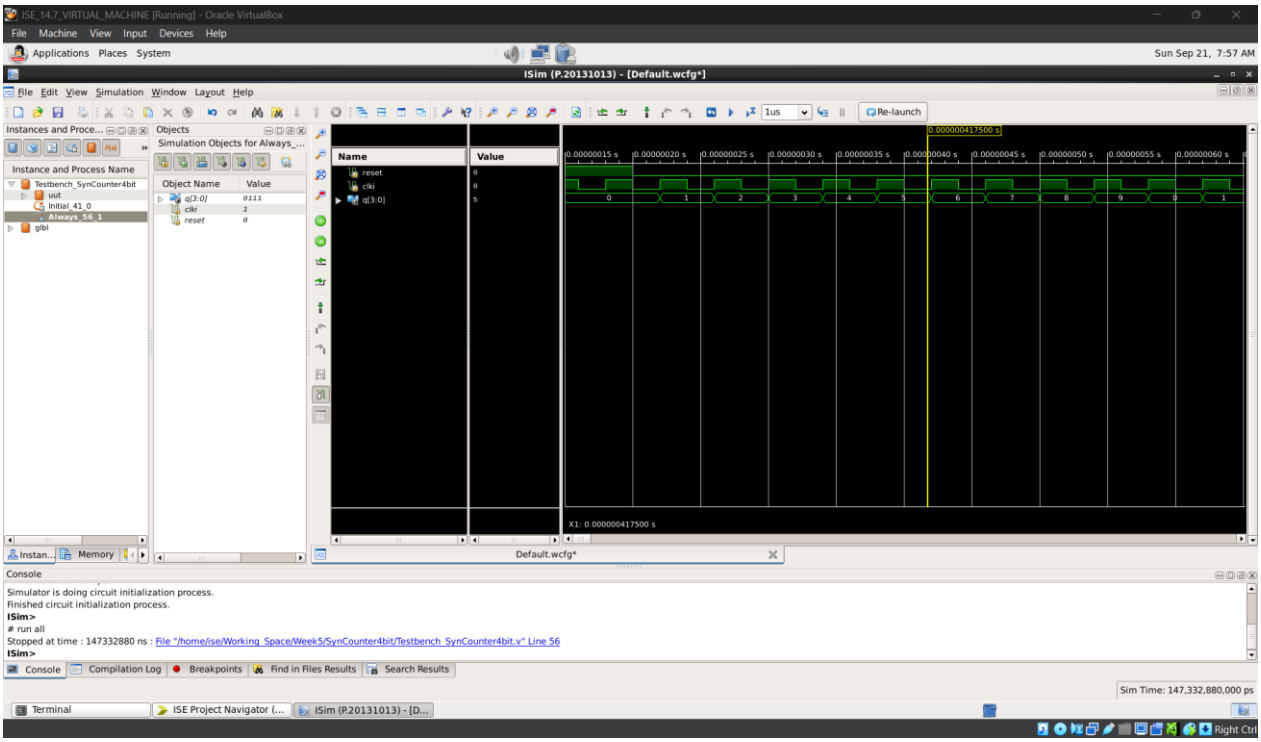
Testcase 2: TC2

Purpose: Kiểm tra đếm lên

Input/Stimulus: E = 0, Clki

Expected Output: Đếm lên đúng yêu cầu

Waveform Simulation (attach/insert image):



Analysis:

Ngõ ra đúng mong đợi

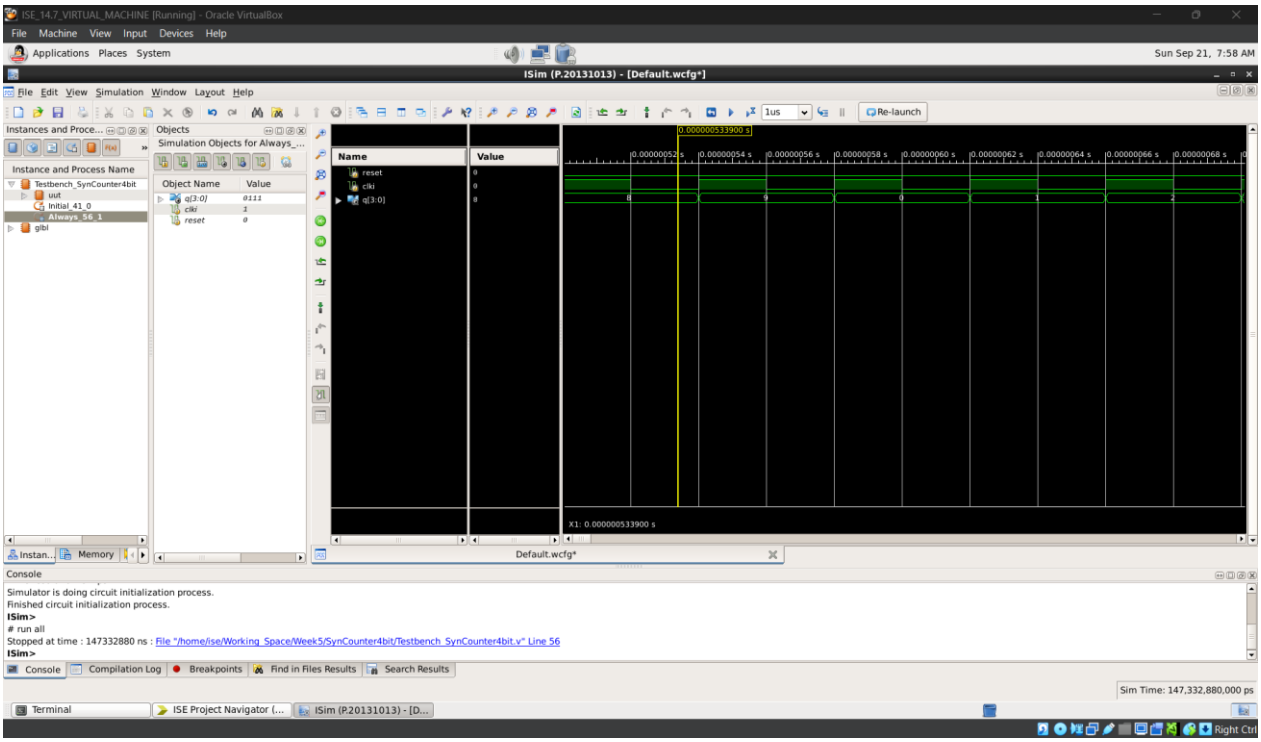
Testcase 3: TC3

Purpose: Kiểm tra đếm lên

Input/Stimulus: E = 1 → 0, Clki

Expected Output: Từ 9 về lại 0

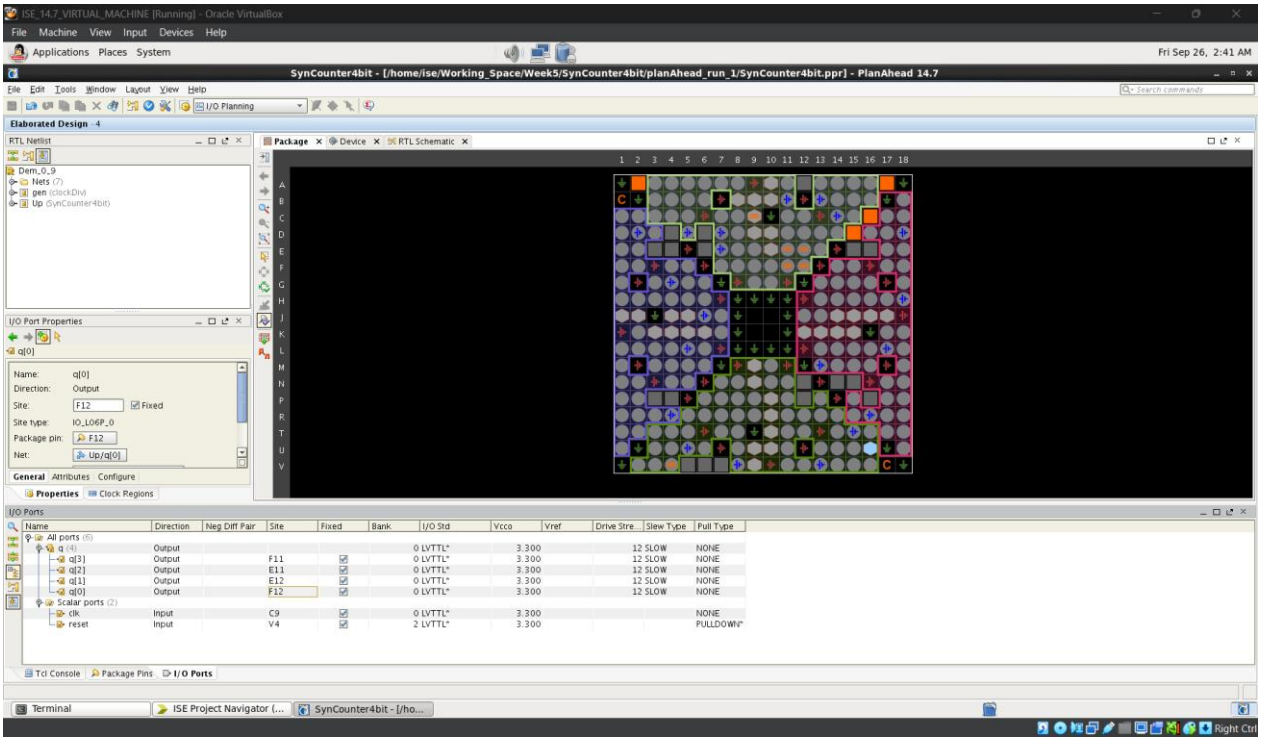
Waveform Simulation (attach/insert image):



Analysis:

Ngõ ra đúng mong đợi

6. Summary



```
# PlanAhead Generated physical constraints

NET "reset" LOC = V4;

# PlanAhead Generated IO constraints

NET "reset" PULLDOWN;

# PlanAhead Generated physical constraints

NET "clk" LOC = C9;

# PlanAhead Generated IO constraints

NET "clk" IOSTANDARD = LVTTL;
NET "reset" IOSTANDARD = LVTTL;
NET "q[3]" IOSTANDARD = LVTTL;
NET "q[2]" IOSTANDARD = LVTTL;
NET "q[1]" IOSTANDARD = LVTTL;
NET "q[0]" IOSTANDARD = LVTTL;

# PlanAhead Generated physical constraints

NET "q[3]" LOC = F11;
NET "q[2]" LOC = E11;
```

```
NET "q[1]" LOC = E12;

NET "q[0]" LOC = F12;
```

WEEKLY REPORT – DIGITAL SYSTEM DESIGN USING VERILOG & FPGA

1. General Information

Project Title: Dem_9_0

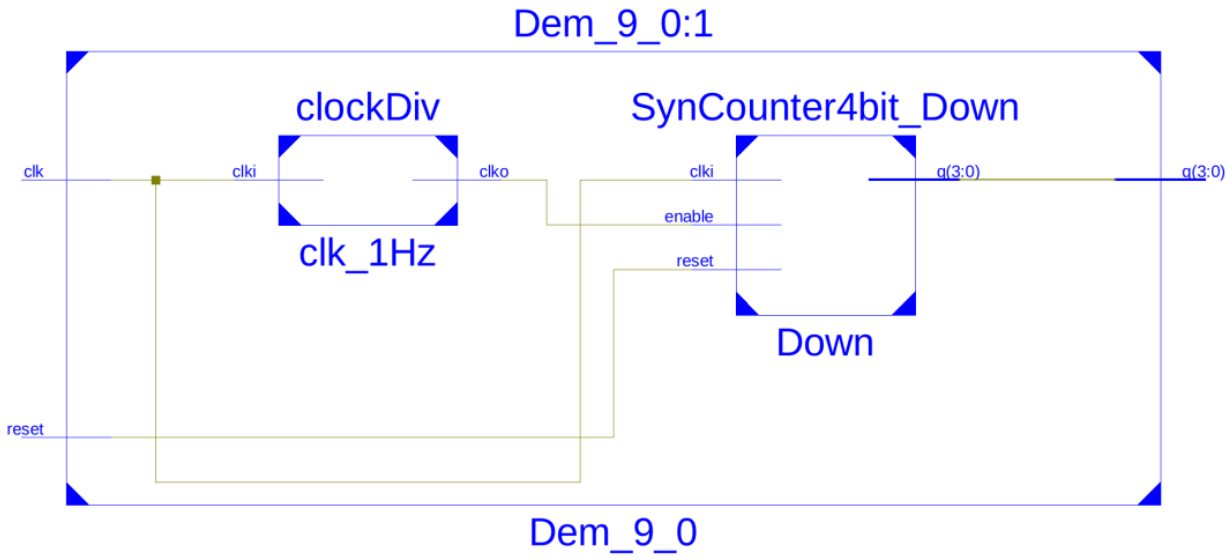
Students: Lê Quang Minh Nhật

Students ID: 23139031

Date: 26/09/2005

2. Block Diagram

Draw block diagram here.



3. State Transition Diagram

Describe Truth table or Design requirements or the FSM state diagram, etc

Bảng trạng thái – clockDiv

Cạnh clki	Trạng thái hiện tại r_reg	Ngõ ra clko	Trạng thái kế r_reg
↑	0 … M-2	0	r_reg + 1
↑	M-1	1	0

Bảng trạng thái – SynCounter4bit_Down

Sự kiện	reset	enable	Trạng thái hiện tại r_reg	Trạng thái kế r_reg	Ngõ ra q

↑ reset	1	X	X	9	9
↑ clki	0	0	x	giữ nguyên x	x
↑ clki	0	1	1…9	r_reg - 1	r_reg - 1
↑ clki	0	1	0	9	9

Bảng trạng thái gộp – toàn mạch Dem_9_0

Cạnh clk	reset	clk_1 (từ clockDiv)	q hiện tại	q kế tiếp
↑	1	X	X	9
↑	0	0	k	k (giữ)
↑	0	1	9…1	k-1
↑	0	1	0	9

Mô đum clockDiv

```
module clockDiv
#(parameter M=50000000)
(
   input wire clki,
   output wire clko
   );

        reg [30:0] r_reg ;

        initial r_reg =0;

        always @(posedge clki) begin
                if (r_reg == M-1)
                        r_reg <= 0;
```

```
                else
                        r_reg <= r_reg + 1;
                end

        assign clko = (r_reg == M-1);

endmodule
```

Mô đum SynCounter4bit_Down

```
module SynCounter4bit_Down(
   input wire clki,
   input wire reset,
         input wire enable,
   output wire [3:0] q
   );

   // signal declaration
   reg [3:0] r_reg;
   wire [3:0] r_next;

         // body, register
   always @(posedge clki, posedge reset)
      if (reset)
         r_reg <= 4'd9;        // reset ve9
      else if (enable)
         r_reg <= r_next;      // cap nhat gia tri moi
```

```
        // next state logic
    assign r_next = (r_reg == 4'd0) ? 4'd9 : (r_reg - 1);

    // output logic
    assign q = r_reg;

endmodule
```

Mô đum Dem_9_0

```
module Dem_9_0(
    input wire clk,
    input wire reset,
    output wire [3:0] q
    );

        wire clk_1;

        clockDiv clk_1Hz (.clki(clk), .clko(clk_1));

        SynCounter4bit_Down Down (.clki(clk), .reset(reset), .enable(clk_1),.q(q));

endmodule
```

4. Test Cases Overview

TC ID	Purpose	Input	Expected Output	Result (Pass/Fail)
TC1	Kiểm tra reset	reset = 1	Ngõ ra đếm	Pass

TC2	Kiểm tra overflow	reset = 0 clki = ~clki	Từ 0 → 9	Pass

Mô đum Testbench

```
`timescale 1ns / 1ps


module Testbench_SynCounter4bit_Down;


	// Inputs
	reg clki;
	reg reset;
	reg enable;

	// Outputs
	wire [3:0] q;

	// Instantiate the Unit Under Test (UUT)
	SynCounter4bit_Down uut (
		.clki(clki),
		.reset(reset),
		.enable(enable),
		.q(q)
	);

	initial begin
		// Initialize Inputs
		clki = 0;
```

```
        reset = 0;
        enable = 0;

        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
        reset = 1;
        #100;
        reset = 0;
    end

    always forever #10 clki = ~clki;
    always forever #10 enable = ~enable;

endmodule
```

5. Testcase Details

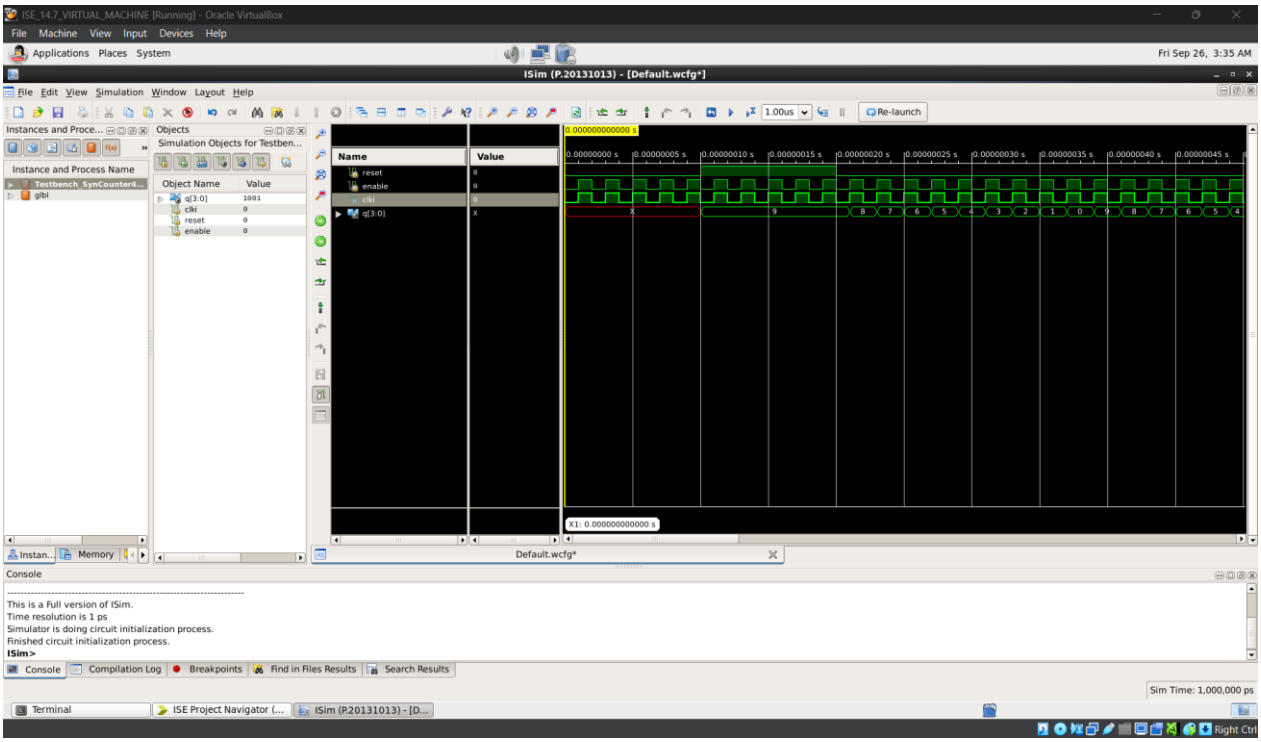
Testcase 1: TC1

Purpose: Kiểm tra reset

Input/Stimulus: reset = 1

Expected Output: Ngõ ra đếm

Waveform Simulation (attach/insert image):



Analysis:

Khi reset = 1, FlipFlop nhận giá trị, mạch bắt đầu đếm.

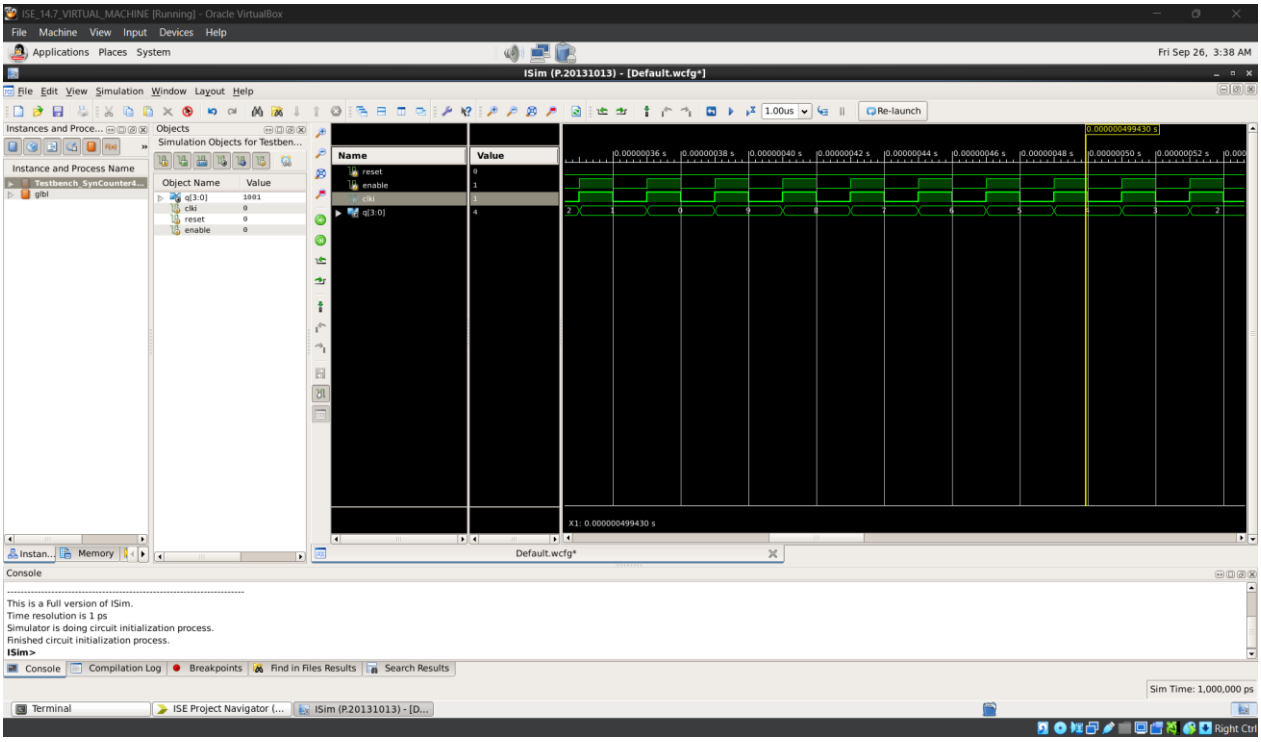
Testcase 2: TC 2

Purpose: Kiểm tra overflow

Input/Stimulus: reset = 0, clki = ~clki

Expected Output: Từ 0 → 9

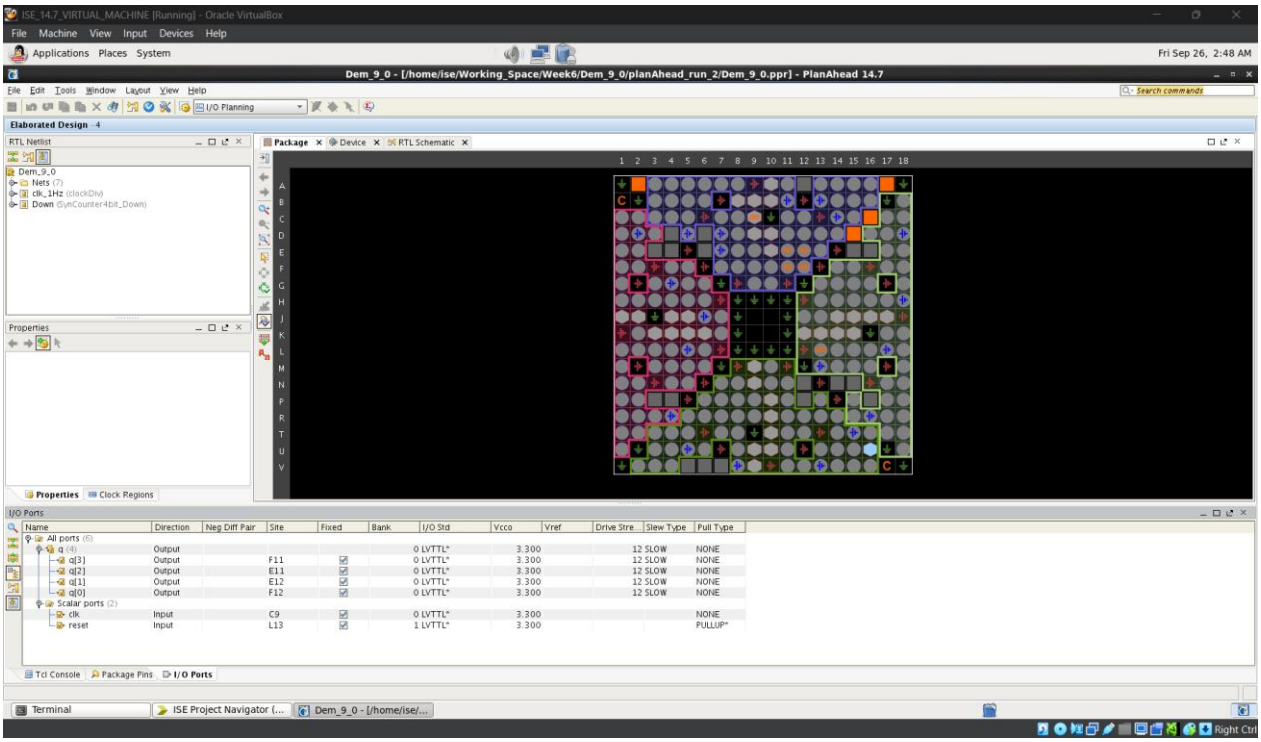
Waveform Simulation (attach/insert image):



Analysis:

Khi mạch đếm đến 9 tự trở về lại 0

6. Summary




```
# PlanAhead Generated IO constraints


NET "q[3]" IOSTANDARD = LVTTL;

NET "q[2]" IOSTANDARD = LVTTL;

NET "q[1]" IOSTANDARD = LVTTL;

NET "q[0]" IOSTANDARD = LVTTL;

NET "clk" IOSTANDARD = LVTTL;

NET "reset" IOSTANDARD = LVTTL;


# PlanAhead Generated physical constraints


NET "reset" LOC = L13;

NET "clk" LOC = C9;
```

```
NET "q[3]" LOC = F11;

NET "q[2]" LOC = E11;

NET "q[1]" LOC = E12;

NET "q[0]" LOC = F12;


# PlanAhead Generated IO constraints


NET "reset" PULLUP;
```

WEEKLY REPORT – DIGITAL SYSTEM DESIGN USING VERILOG & FPGA

1. General Information

Project Title: Dem_0_9_AND_9_0_Single

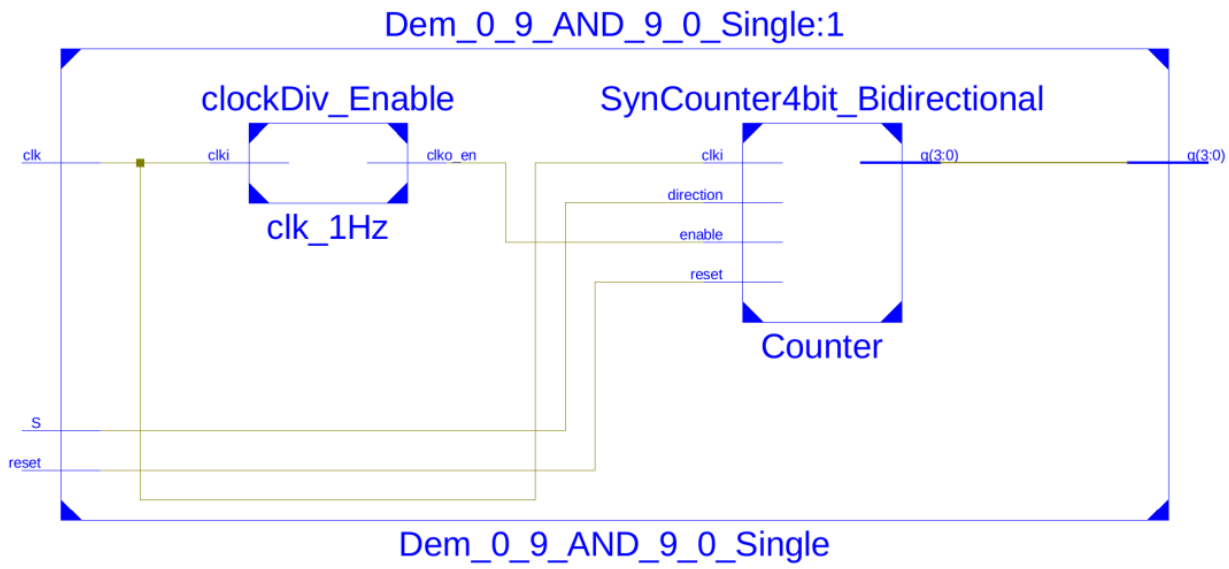
Students: Lê Quang Minh Nhật

Students ID: 23139031

Date: 26/09/2005

2. Block Diagram

Draw block diagram here.



3. State Transition Diagram

Describe Truth table or Design requirements or the FSM state diagram, etc

Bảng trạng thái clockDiv_Enable

Cạnh clki	Trạng thái hiện tại r_reg	Ngõ ra clko_en	Trạng thái kế r_reg
↑	0 … M-2	0	r_reg + 1
↑	M-1	1	0

Bảng trạng thái SynCounter4bit_Bidirectional

Sự kiện	reset	enable	direction	r_reg hiện tại	r_reg'(trạng thái kế)	q

↑reset	1	X	0	X	0	0
↑reset	1	X	1	X	9	9
↑clki	0	0	X	k	k (giữ)	k
↑clki	0	1	0 (up)	0…8	k+1	k+1
↑clki	0	1	0 (up)	9	0	0
↑clki	0	1	1 (down)	1…9	k-1	k-1
↑clki	0	1	1 (down)	0	9	9

Bảng trạng thái gộp cho toàn mạch Dem_0_9_AND_9_0_Single

Cạnh clk	reset	clk_en	S (direction)	q hiện tại	q kế tiếp
↑	1	X	0	X	0
↑	1	X	1	X	9
↑	0	0	X	k	k (giữ)
↑	0	1	0 (up)	0…8	k+1
↑	0	1	0 (up)	9	0
↑	0	1	1 (down)	1…9	k-1
↑	0	1	1 (down)	0	9

Mô đum clockDiv_Enable

```
module clockDiv_Enable

#(parameter M=50000000)

(

    input wire clki,

    output wire clko_en

);

    reg [30:0] r_reg;
```

```
    initial r_reg = 0;


    always @(posedge clki) begin
        if (r_reg == M-1)
            r_reg <= 0;
        else
            r_reg <= r_reg + 1;
    end


    assign clko_en = (r_reg == M-1);
endmodule
```

Mô đum SynCounter4bit_Bidirectional

```
module SynCounter4bit_Bidirectional(
    input wire clki,
    input wire reset,
    input wire enable,
    input wire direction,           // 0 = count up, 1 = count down
    output wire [3:0] q
    );


    reg [3:0] r_reg;
    wire [3:0] r_next;


    // Register logic
```

```
    always @(posedge clki, posedge reset) begin
        if (reset)
            r_reg <= direction ? 4'd9 : 4'd0;  // Reset to 9 if counting down, 0 if counting up
        else if (enable)
            r_reg <= r_next;
    end


            // Next state logic
    assign r_next = direction ?
                    ((r_reg == 4'd0) ? 4'd9 : (r_reg - 1)) :  // Count down: 9->8->...->1->0->9
                    ((r_reg == 4'd9) ? 4'd0 : (r_reg + 1));   // Count up: 0->1->...->8->9->0


    // Output logic
    assign q = r_reg;


endmodule
```

Mô đum Dem_0_9_AND_9_0_Single

```
module Dem_0_9_AND_9_0_Single(
    input wire S,           // Direction: 0 = count up, 1 = count down
    input wire clk,
    input wire reset,
    output wire [3:0] q
);
    wire clk_en;
```

```
    // Clock divider generates enable signal
    clockDiv_Enable clk_1Hz (.clki(clk), .clko_en(clk_en));

    // Single bidirectional counter
    SynCounter4bit_Bidirectional Counter (
        .clki(clk),
        .reset(reset),
        .enable(clk_en),
        .direction(S),      // 0 = up, 1 = down
        .q(q)
    );
endmodule
```

4. Test Cases Overview

TC ID	Purpose	Input	Expected Output	Result (Pass/Fail)
TC1	Kiểm tra reset	reset = 1	Bắt đầu đếm	Pass
TC2	Kiểm tra đếm lên	reset = 0 direction = 0	Đếm lên	Pass
TC3	Kiểm tra đếm xuống	reset = 0 direction = 1	Đếm xuống	Pass

Mô đum Testbench

```
`timescale 1ns / 1ps

module Testbench_SynCounter4bit_Bidirectional;

        // Inputs
        reg clki;
        reg reset;
        reg enable;
```

```
reg direction;

// Outputs
wire [3:0] q;

// Instantiate the Unit Under Test (UUT)
SynCounter4bit_Bidirectional uut (
        .clki(clki),
        .reset(reset),
        .enable(enable),
        .direction(direction),
        .q(q)
);

initial begin
        // Initialize Inputs
        clki = 0;
        reset = 0;
        enable = 0;
        direction = 0;

        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
```

```
                reset = 1;
                #100;
                reset = 0;
                #100;
                #1000;
                direction = 1;
                #1000;
                direction = 0;
        end

        always forever #10 clki = ~clki;
        always forever #10 enable = ~enable;

endmodule
```

5. Testcase Details

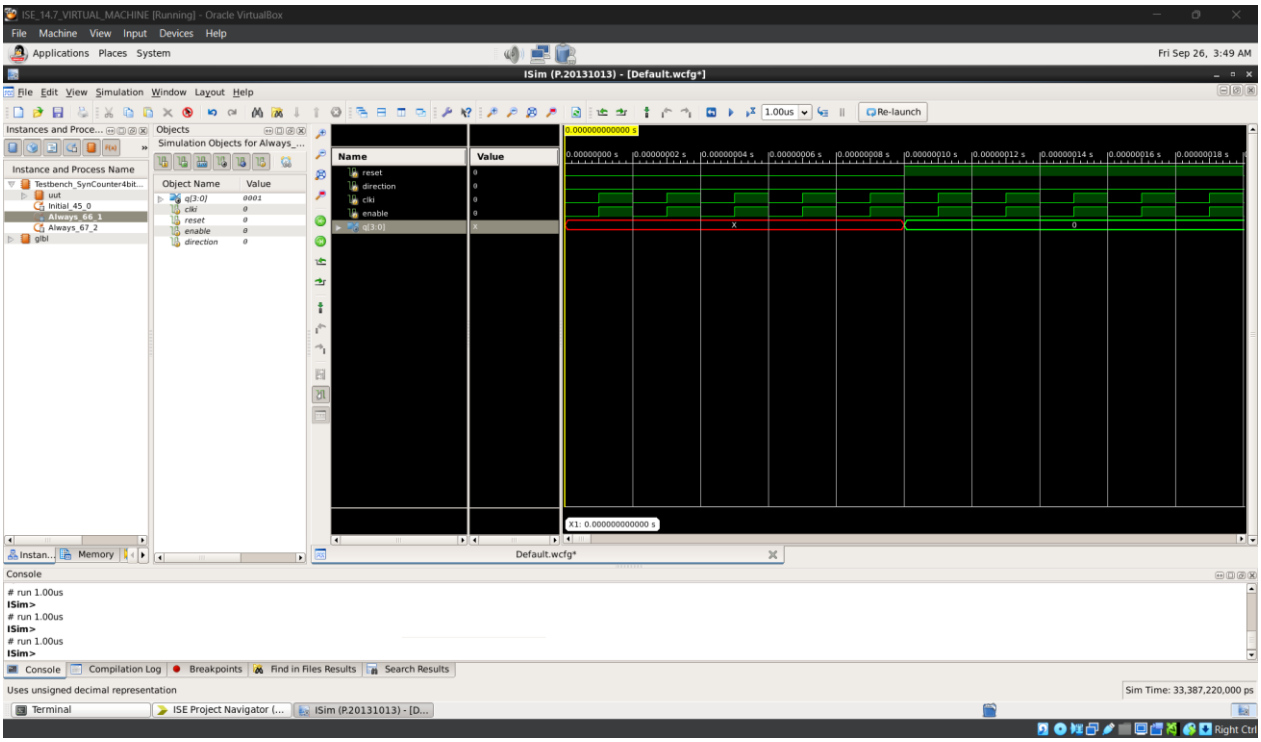
Testcase 1: TC1

Purpose: Kiểm tra reset

Input/Stimulus: reset = 1

Expected Output: Bắt đầu đếm

Waveform Simulation (attach/insert image):



Analysis:

Khi reset = 1, mạch bắt đầu đếm.

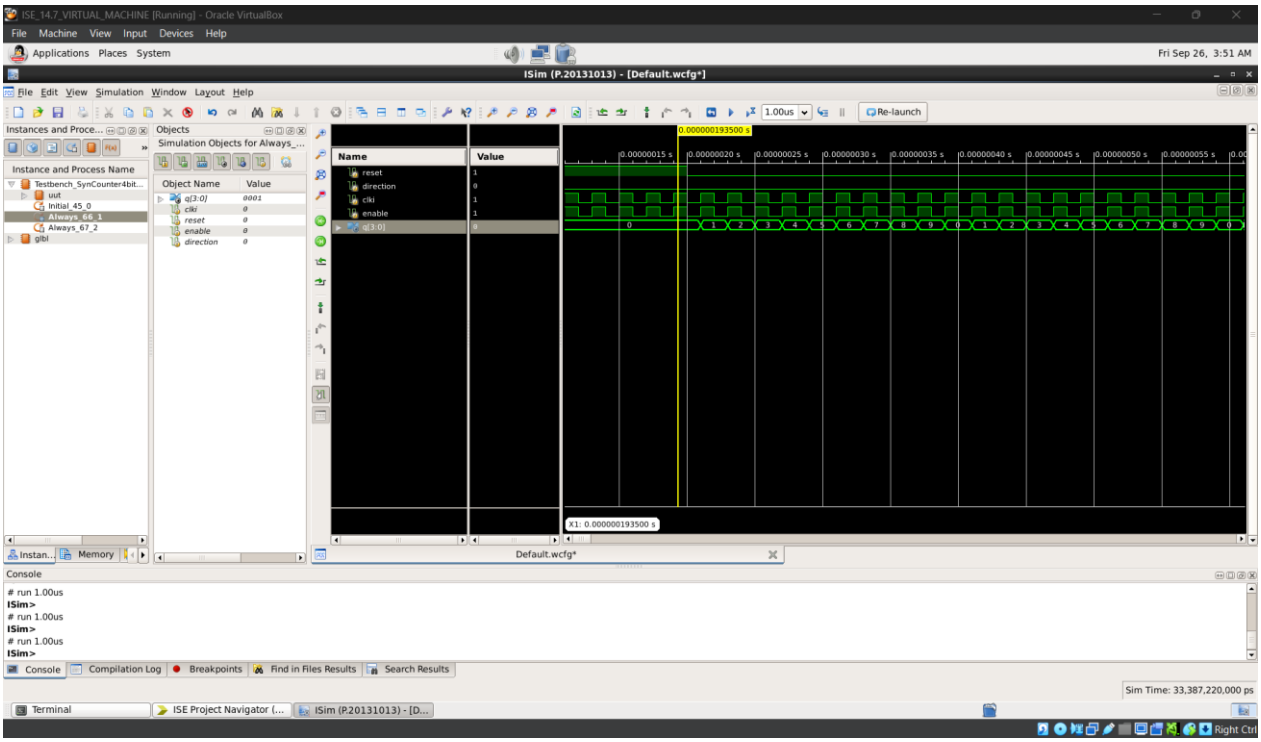
Testcase 2: TC2

Purpose: Kiểm tra đếm lên

Input/Stimulus: reset = 0, direction = 0

Expected Output: Đếm lên

Waveform Simulation (attach/insert image):



Analysis:

Mô phỏng chạy đúng yêu cầu đề bài

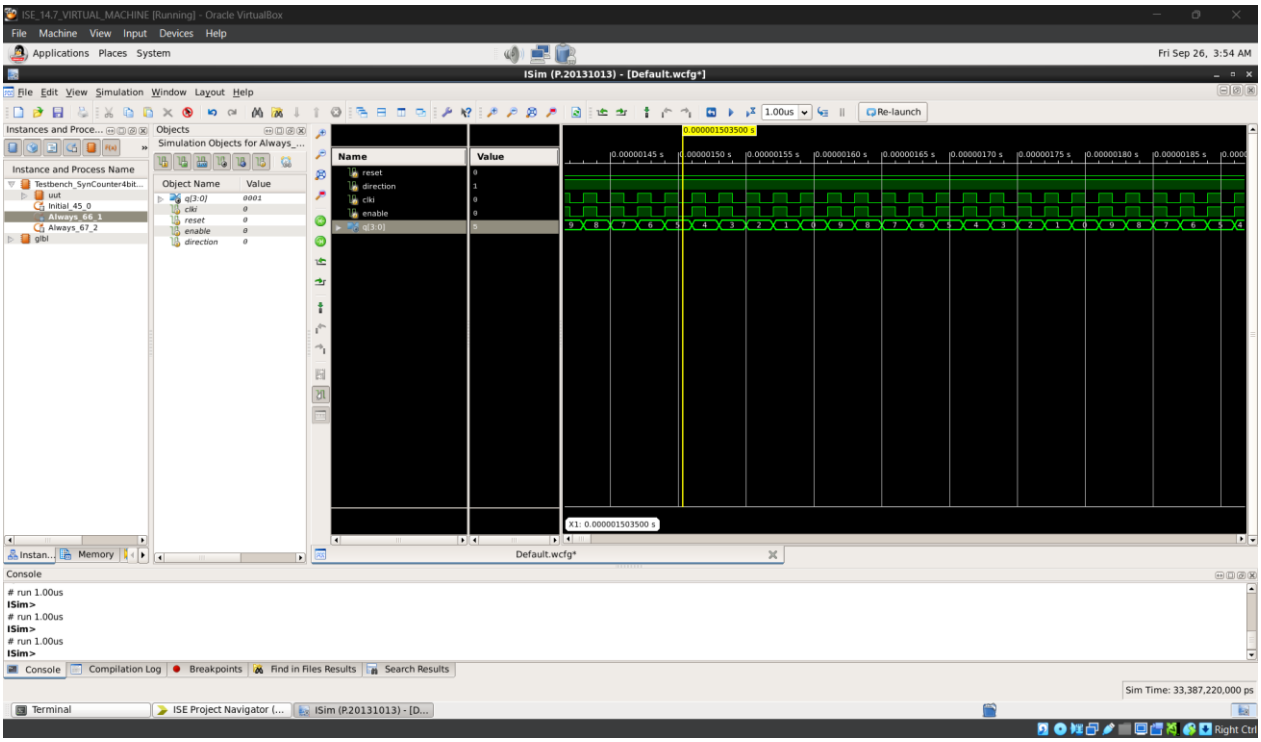
Testcase 3: TC3

Purpose: Kiểm tra đếm xuống

Input/Stimulus: reset = 0, direction = 1

Expected Output: Đếm xuống

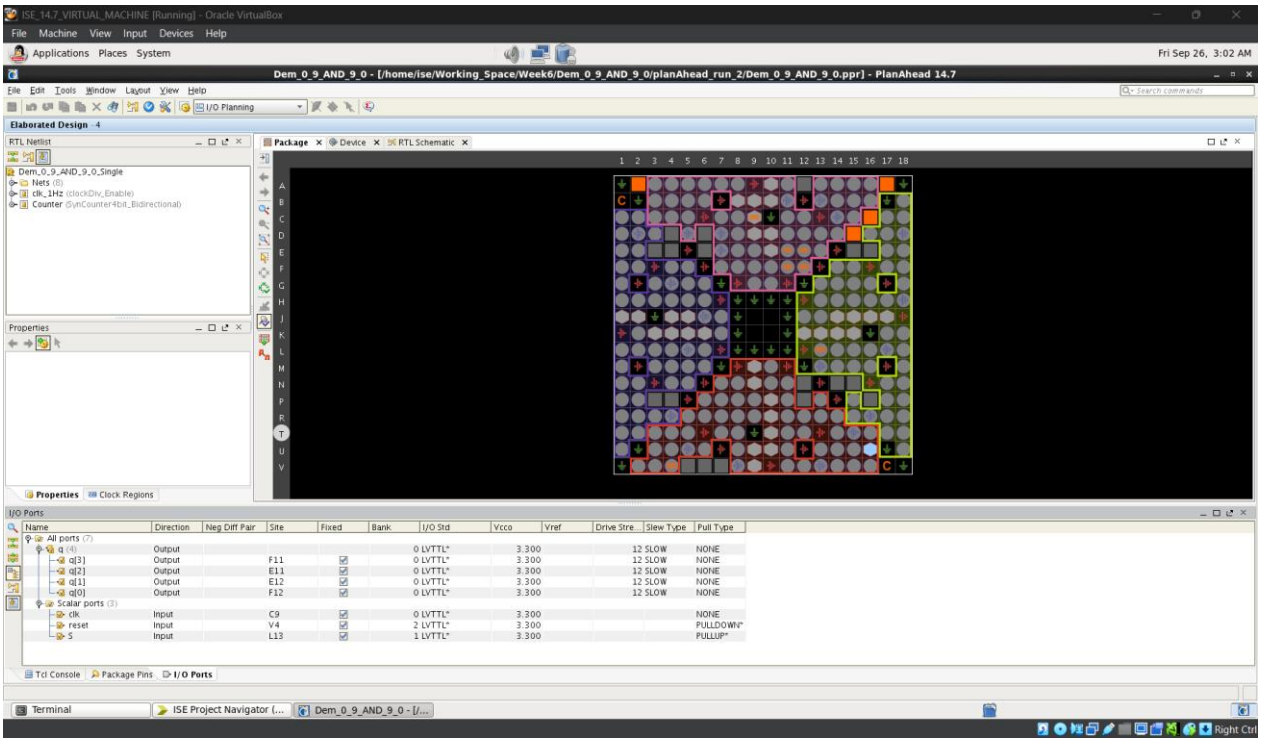
Waveform Simulation (attach/insert image):



Analysis:

Mô phỏng chạy đúng yêu cầu đề bài.

6. Summary




```
# PlanAhead Generated IO constraints


NET "q[3]" IOSTANDARD = LVTTL;

NET "q[2]" IOSTANDARD = LVTTL;

NET "q[1]" IOSTANDARD = LVTTL;

NET "q[0]" IOSTANDARD = LVTTL;

NET "clk" IOSTANDARD = LVTTL;

NET "reset" IOSTANDARD = LVTTL;

NET "S" IOSTANDARD = LVTTL;


# PlanAhead Generated physical constraints


NET "clk" LOC = C9;
```

```
NET "reset" LOC = V4;

NET "S" LOC = L13;


# PlanAhead Generated IO constraints


NET "reset" PULLDOWN;

NET "S" PULLUP;


# PlanAhead Generated physical constraints


NET "q[3]" LOC = F11;

NET "q[2]" LOC = E11;

NET "q[1]" LOC = E12;

NET "q[0]" LOC = F12;
```

WEEKLY REPORT – DIGITAL SYSTEM DESIGN USING VERILOG & FPGA

1. General Information

Project Title: Auto_Counter_0_9

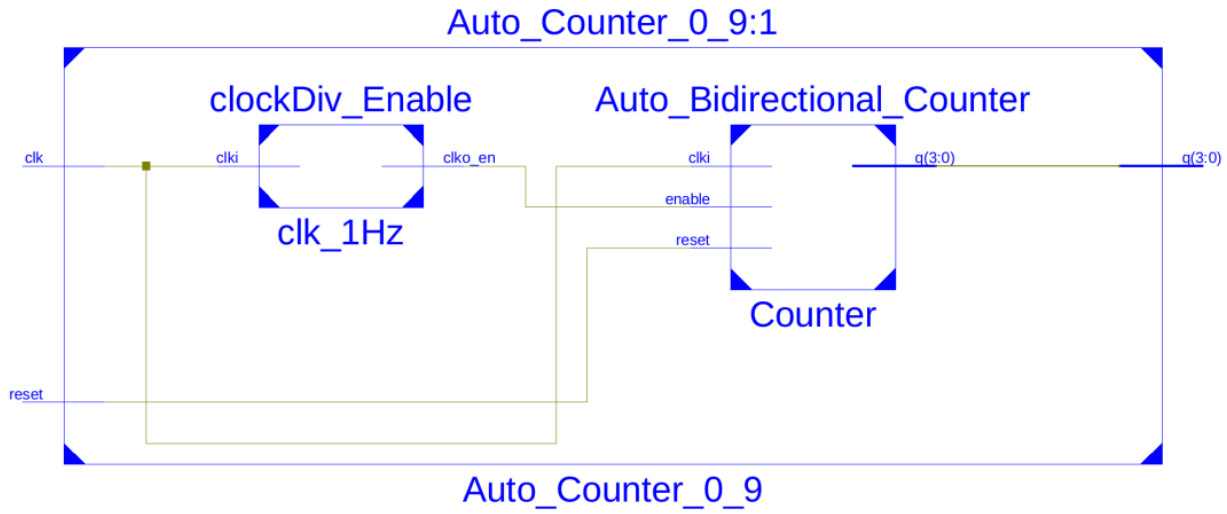
Students: Lê Quang Minh Nhật

Students ID: 23139031

Date: 26/09/2025

2. Block Diagram

Draw block diagram here.



3. State Transition Diagram

Describe Truth table or Design requirements or the FSM state diagram, etc

Bảng trạng thái clockDiv_Enable

Cạnh clki	Trạng thái hiện tại r_reg	Ngõ ra clko_en	Trạng thái kế r_reg
↑	0 … M-2	0	r_reg + 1
↑	M-1	1	0

Bảng trạng thái Auto_Bidirectional_Counter

Sự kiện	reset	enable	direction_reg	count_reg hiện tại	Trạng thái kế (count', dir')	Ghi chú
↑reset	1	X	X	X	(0, 0)	Reset về 0 và hướng Up
↑clki	0	0	X	k	(k, dir)	Không đếm khi enable=0
↑clki	0	1	0 (Up)	0…8	(k+1, 0)	Đếm tăng bình thường
↑clki	0	1	0 (Up)	9	(**8**, 1)	Chạm 9 → đổi hướng Down và nhảy về 8
↑clki	0	1	1 (Down)	2…9	(k-1, 1)	Đếm giảm bình thường
↑clki	0	1	1 (Down)	1	(0, 1)	Về 0 nhưng vẫn đang Down
↑clki	0	1	1 (Down)	0	(**1**, 0)	Chạm 0 → đổi hướng Up và nhảy về 1

Bảng trạng thái gộp cho Auto_Counter_0_9

Cạnh clk	reset	clk_en	Trạng thái (q, dir) hiện tại	Trạng thái kế (q', dir')
↑	1	X	(k, d)	(0, 0)
↑	0	0	(k, d)	(k, d) (giữ)
↑	0	1	(k, 0) & k=0…8	(k+1, 0)
↑	0	1	(9, 0)	(8, 1)
↑	0	1	(k, 1) & k=2…9	(k-1, 1)
↑	0	1	(1, 1)	(0, 1)
↑	0	1	(0, 1)	(1, 0)

Mô đum clockDiv_Enable

```
module clockDiv_Enable
```

```
#(parameter M=50000000)  // 50MHz -> 1Hz
(
	input wire clki,
	output wire clko_en
);
	reg [30:0] r_reg;

	initial r_reg = 0;

	always @(posedge clki) begin
		if (r_reg == M-1)
			r_reg <= 0;
		else
			r_reg <= r_reg + 1;
		end

		assign clko_en = (r_reg == M-1);

endmodule
```

Mô đum Auto_Bidirectional_Counter

```
module Auto_Bidirectional_Counter(
   input wire clki,
   input wire reset,
   input wire enable,
   output wire [3:0] q
```

```
);

reg [3:0] count_reg;
reg direction_reg;          //  0=up, 1=down

// Register logic
always @(posedge clki, posedge reset) begin
    if (reset) begin
        count_reg <= 4'd0;      // Reset 0
        direction_reg <= 1'b0;  // Reset up
    end
    else if (enable) begin
        // Logic counter
        if (direction_reg == 0) begin  // Up
            if (count_reg == 4'd9) begin
                count_reg <= 4'd8;      // 9 -> 0
                direction_reg <= 1'b1;  // Down
            end
            else begin
                count_reg <= count_reg + 1;  // Count continue
            end
        end
        else begin  // Down
            if (count_reg == 4'd0) begin
                count_reg <= 4'd1;      // 0 -> 9
```

```
                direction_reg <= 1'b0;  // Up
            end
            else begin
                count_reg <= count_reg - 1;  // Count continue
            end
        end
    end
    end


        // Output
    assign q = count_reg;


endmodule
```

Mô đum Auto_Counter_0_9

```
module Auto_Counter_0_9(
    input wire clk,
    input wire reset,
    output wire [3:0] q
);
    wire clk_en;

    // Clock divider tạo enable signal
    clockDiv_Enable clk_1Hz (.clki(clk), .clko_en(clk_en));

    // Auto
```

```
    Auto_Bidirectional_Counter Counter (
        .clki(clk),
        .reset(reset),
        .enable(clk_en),
        .q(q)
    );
endmodule
```

4. Test Cases Overview

TC ID	Purpose	Input	Expected Output	Result (Pass/Fail)
TC1	Kiểm tra reset	reset = 1	Bắt đầu đếm	Pass
TC2	Kiểm tra đếm tự động	reset = 0	Lên xuống tự động	Pass

5. Testcase Details

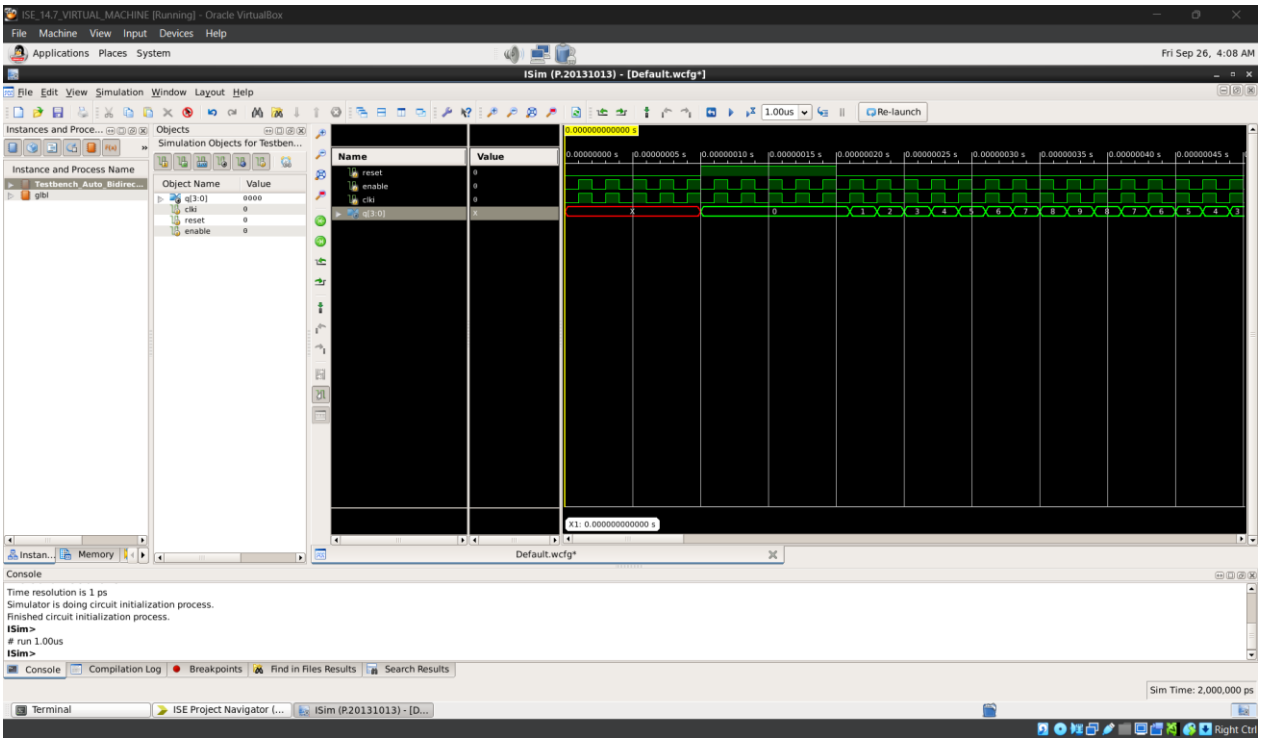
Testcase 1: TC1

Purpose: Kiểm tra reset

Input/Stimulus: reset = 1

Expected Output: Bắt đầu đếm

Waveform Simulation (attach/insert image):



Analysis:

Kiểm tra reset để đếm

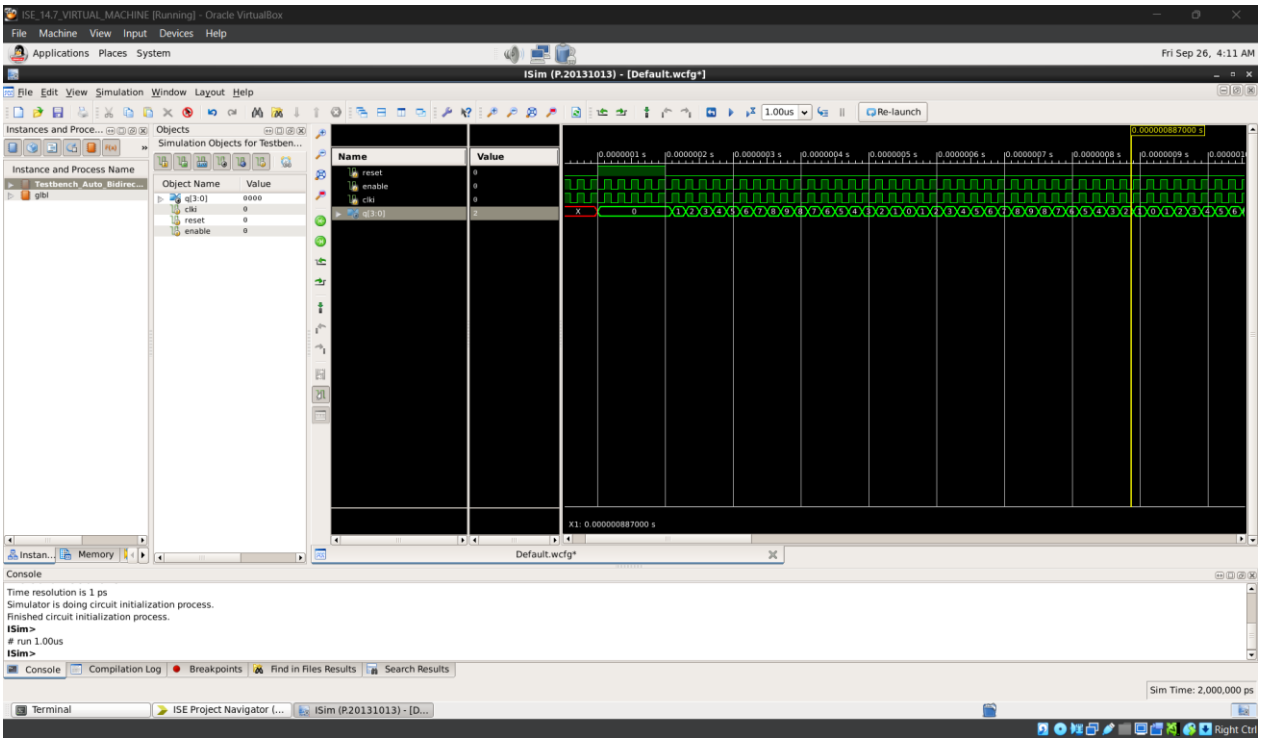
Testcase 2: TC2

Purpose: Kiểm tra đếm tự động

Input/Stimulus: reset = 0

Expected Output: Lên xuống tự động

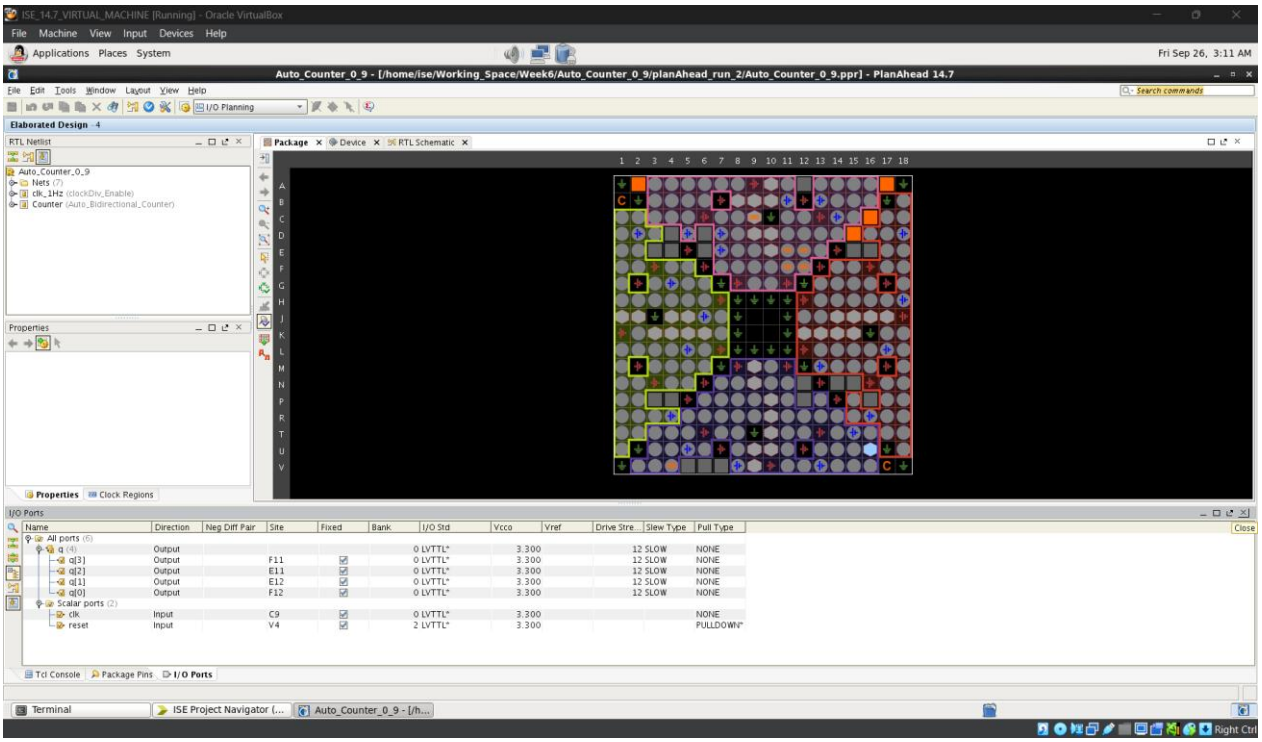
Waveform Simulation (attach/insert image):



Analysis:

Đúng yêu cầu đề bài

6. Summary




```
# PlanAhead Generated IO constraints


NET "q[3]" IOSTANDARD = LVTTL;

NET "q[2]" IOSTANDARD = LVTTL;

NET "q[1]" IOSTANDARD = LVTTL;

NET "q[0]" IOSTANDARD = LVTTL;

NET "clk" IOSTANDARD = LVTTL;

NET "reset" IOSTANDARD = LVTTL;


# PlanAhead Generated physical constraints


NET "clk" LOC = C9;

NET "reset" LOC = V4;
```

```
# PlanAhead Generated IO constraints

NET "reset" PULLDOWN;

# PlanAhead Generated physical constraints

NET "q[3]" LOC = F11;
NET "q[2]" LOC = E11;
NET "q[1]" LOC = E12;
NET "q[0]" LOC = F12;
```

WEEKLY REPORT – DIGITAL SYSTEM DESIGN USING VERILOG & FPGA

1. General Information

Project Title: Dem_BCD_0_59

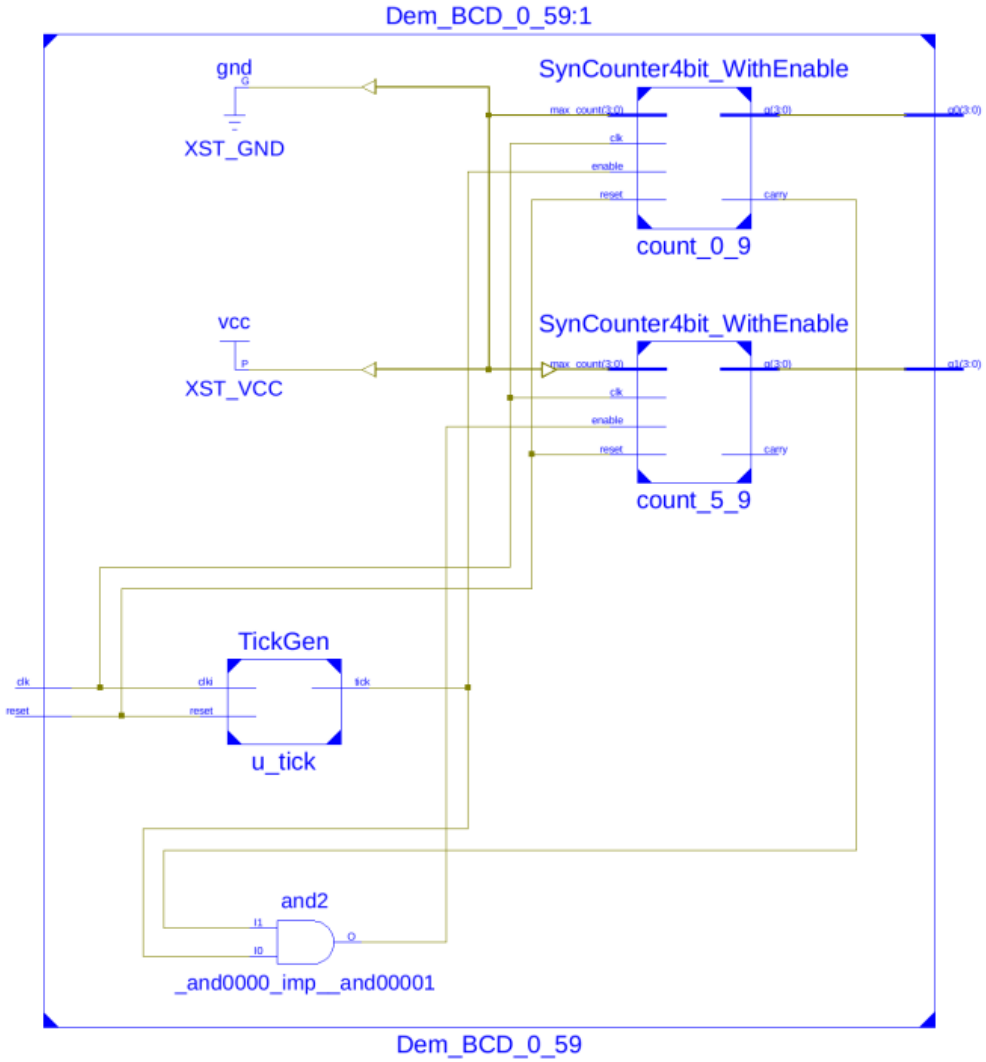
Students: Lê Quang Minh Nhật

Students ID: 23139031

Date: 3/10/2025

2. Block Diagram

Draw block diagram here.



3. State Transition Diagram

Describe Truth table or Design requirements or the FSM state diagram, etc

TickGen (M=25_000_000)

reset	r_reg (hiện tại)	tick (hiện tại)	r_reg (kế tiếp)
1	bất kỳ	0	0
0	0 … M-2	0	r_reg + 1
0	M-1	1	0

SynCounter4bit_WithEnable (đếm 0..max_count)

reset	enable	q (hiện tại)	carry	q (kế tiếp)
1	x	bất kỳ	–	0
0	0	bất kỳ	q==N	q (giữ)
0	1	0 … N-1	0	q + 1
0	1	N (=max_count)	1	0

Dem_BCD_0_59 (bộ đếm phút 00..59)

reset	tick	q1 q0 (hiện tại)	carry_q0	q1 q0 (kế tiếp)	Diễn giải
1	x	bất kỳ	–	00	Reset đồng bộ mức cao về 00
0	0	bất kỳ	q0==9	giữ nguyên	Không có tick ⇒ không đếm
0	1	q0 = 0..8	0	(q1, q0+1)	Tick làm q0 tăng
0	1	q0 = 9, q1 = 0..4	1	(q1+1, 0)	q0 tràn ⇒ q1 tăng
0	1	q0 = 9, q1 = 5	1	00	59 → 00

Mô đum TickGen

```
module TickGen
#(parameter M=50000000)
(
    input  wire clki,   // 50 MHz
    input  wire reset,
```

```
    output wire tick
);
        reg [30:0] r_reg;
        always @(posedge clki or posedge reset) begin
                if (reset)
                        r_reg <= 0;
                else if (r_reg == M-1)
                        r_reg <= 0;
                else
                        r_reg <= r_reg + 1;
        end

        assign tick = (r_reg == M-1);

endmodule
```

Mô đum SynCounter4bit_WithEnable

```
module SynCounter4bit_WithEnable(
    input wire clk,
    input wire reset,
    input wire enable,
    input wire [3:0] max_count,
    output reg [3:0] q,
    output wire carry
    );
```

```
always @(posedge clk or posedge reset) begin
        if (reset)
                q <= 4'd0;
        else if (enable) begin
                if (q == max_count)
                        q <= 4'd0;
                else
                        q <= q + 1;
        end
end

        assign carry = (q == max_count);

endmodule
```

Mô đum Dem_BCD_0_59

```
module Dem_BCD_0_59(
   input wire clk,
   input wire reset,
   output wire [3:0] q1,
   output wire [3:0] q0
   );

        wire tick_05Hz;

        TickGen #(25000000) u_tick (.clki(clk), .reset(reset), .tick(tick_05Hz));
```

```
    wire carry_q0;

    SynCounter4bit_WithEnable count_0_9 (
        .clk(clk),
        .reset(reset),
        .enable(tick_05Hz),
        .max_count(4'd9),
        .q(q0),
        .carry(carry_q0)
    );

    SynCounter4bit_WithEnable count_5_9 (
        .clk(clk),
        .reset(reset),
        .enable(tick_05Hz & carry_q0),
        .max_count(4'd5),
        .q(q1),
        .carry()
    );

endmodule
```

4. Test Cases Overview

TC ID	Purpose	Input	Expected Output	Result (Pass/Fail)
TC1	Kiểm tra reset	reset = 1	Mạch bị reset về 0	Pass
TC2	Kiểm tra đếm BCD	clk = ~clk	Mạch đếm lên	Pass

		reset = 0		
TC3	Kiểm tra khi quay về 0	clk = ~clk reset = 0	Mạch đếm về 0	Pass

Mô đum Testbecnh_Dem_BCD_0_59

```
`timescale 1ns / 1ps


module Testbecnh_Dem_BCD_0_59;


	// Inputs
	reg clk;
	reg reset;

	// Outputs
	wire [3:0] q1;
	wire [3:0] q0;

	// Instantiate the Unit Under Test (UUT)
	Dem_BCD_0_59 uut (
		.clk(clk),
		.reset(reset),
		.q1(q1),
		.q0(q0)
	);

	initial begin
		// Initialize Inputs
```

```
        clk = 0;
        reset = 0;

        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
        reset = 1;
        #100;
        reset = 0;
        #100;
    end

    always forever #10 clk = ~clk;

endmodule
```

5. Testcase Details

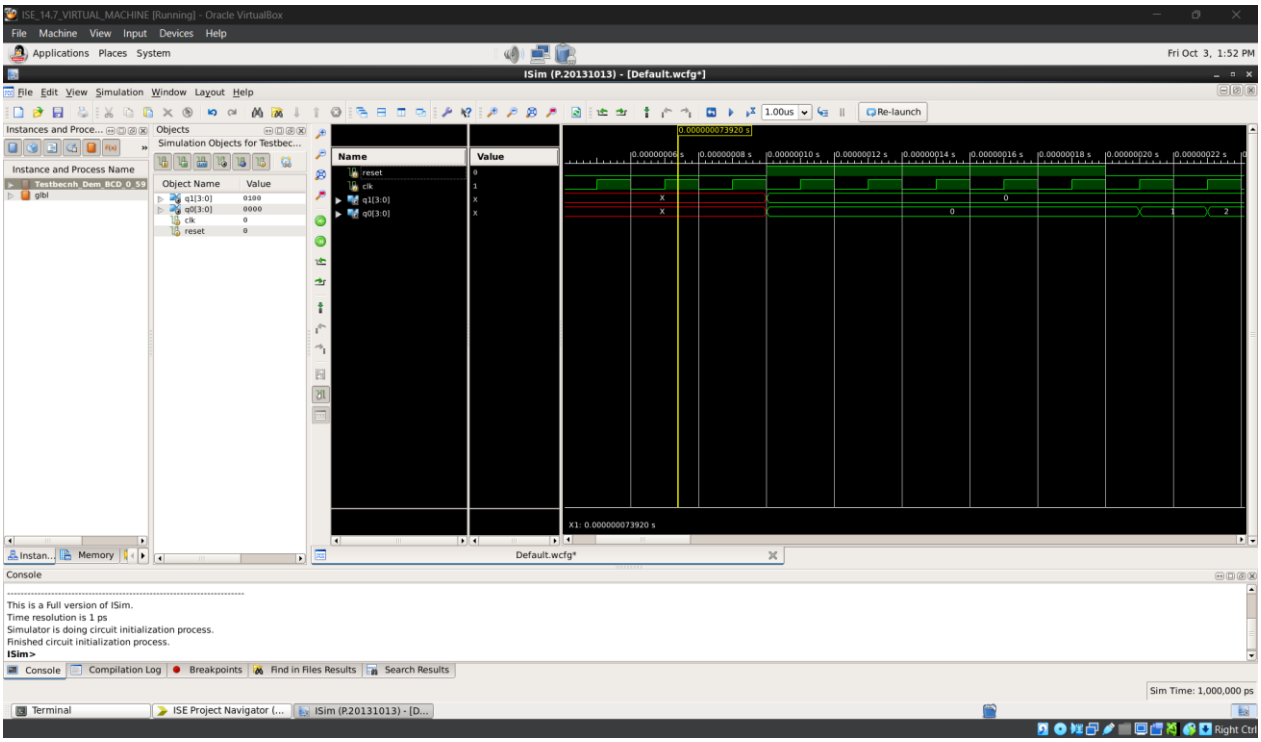
Testcase 1: TC1

Purpose: Kiểm tra reset

Input/Stimulus: reset = 1

Expected Output: Mạch bị reset về 0

Waveform Simulation (attach/insert image):



Analysis:

Mạch hoạt động đúng yêu cầu reset

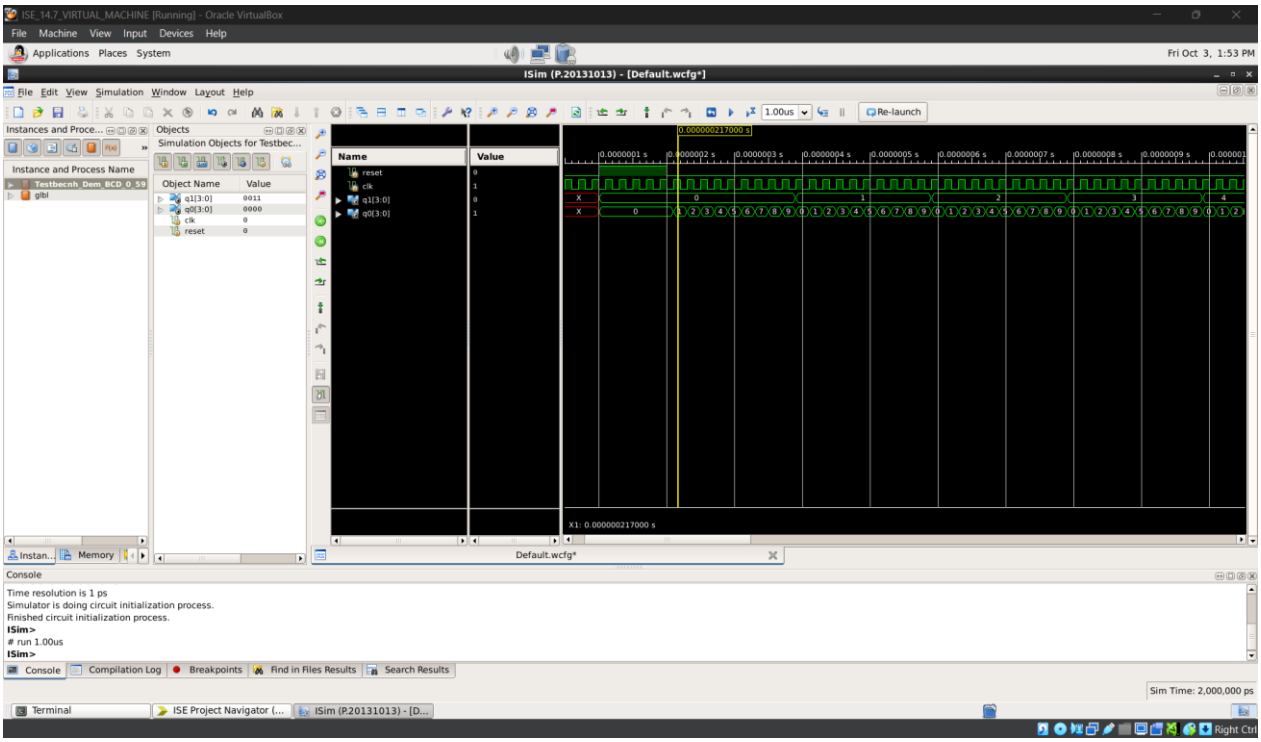
Testcase 2: TC2

Purpose: Kiểm tra đếm BCD

Input/Stimulus: clk = ~clk, reset = 0

Expected Output: Mạch đếm lên

Waveform Simulation (attach/insert image):



Analysis:

Mạch hoạt động đúng yêu cầu đếm lên BCD

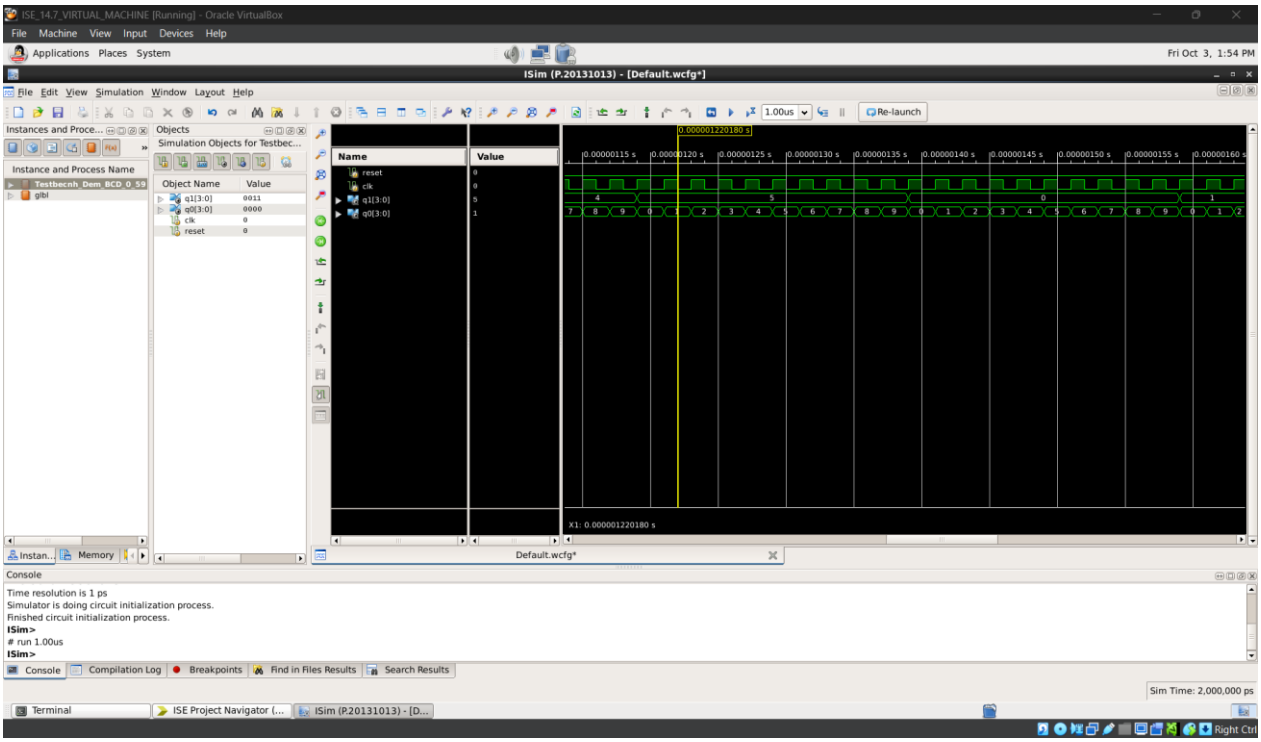
Testcase 3: TC3

Purpose: Kiểm tra khi quay về 0

Input/Stimulus: clk = ~clk, reset = 0

Expected Output: Mạch đếm về 0

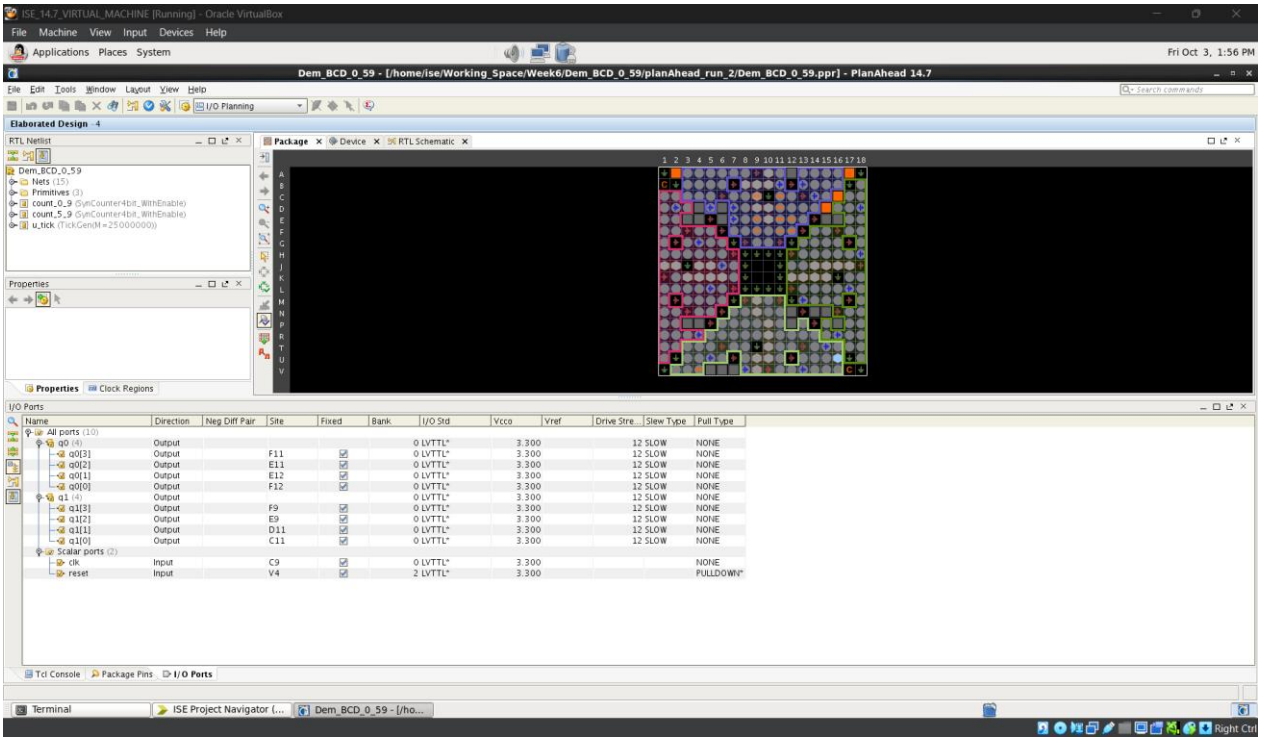
Waveform Simulation (attach/insert image):



Analysis:

Mạch từ 59 trở lại về 0.

6. Summary



```
# PlanAhead Generated IO constraints


NET "q1[3]" IOSTANDARD = LVTTL;

NET "q1[2]" IOSTANDARD = LVTTL;

NET "q1[1]" IOSTANDARD = LVTTL;

NET "q1[0]" IOSTANDARD = LVTTL;

NET "q0[3]" IOSTANDARD = LVTTL;

NET "q0[2]" IOSTANDARD = LVTTL;

NET "q0[1]" IOSTANDARD = LVTTL;

NET "q0[0]" IOSTANDARD = LVTTL;

NET "clk" IOSTANDARD = LVTTL;

NET "reset" IOSTANDARD = LVTTL;


# PlanAhead Generated physical constraints


NET "clk" LOC = C9;

NET "reset" LOC = V4;

NET "q1[3]" LOC = F9;

NET "q1[2]" LOC = E9;

NET "q1[1]" LOC = D11;

NET "q1[0]" LOC = C11;

NET "q0[3]" LOC = F11;

NET "q0[2]" LOC = E11;

NET "q0[1]" LOC = E12;

NET "q0[0]" LOC = F12;
```

```
# PlanAhead Generated IO constraints


NET "reset" PULLDOWN;
```

WEEKLY REPORT – DIGITAL SYSTEM DESIGN USING VERILOG & FPGA

1. General Information

Project Title: Dich_8_bit

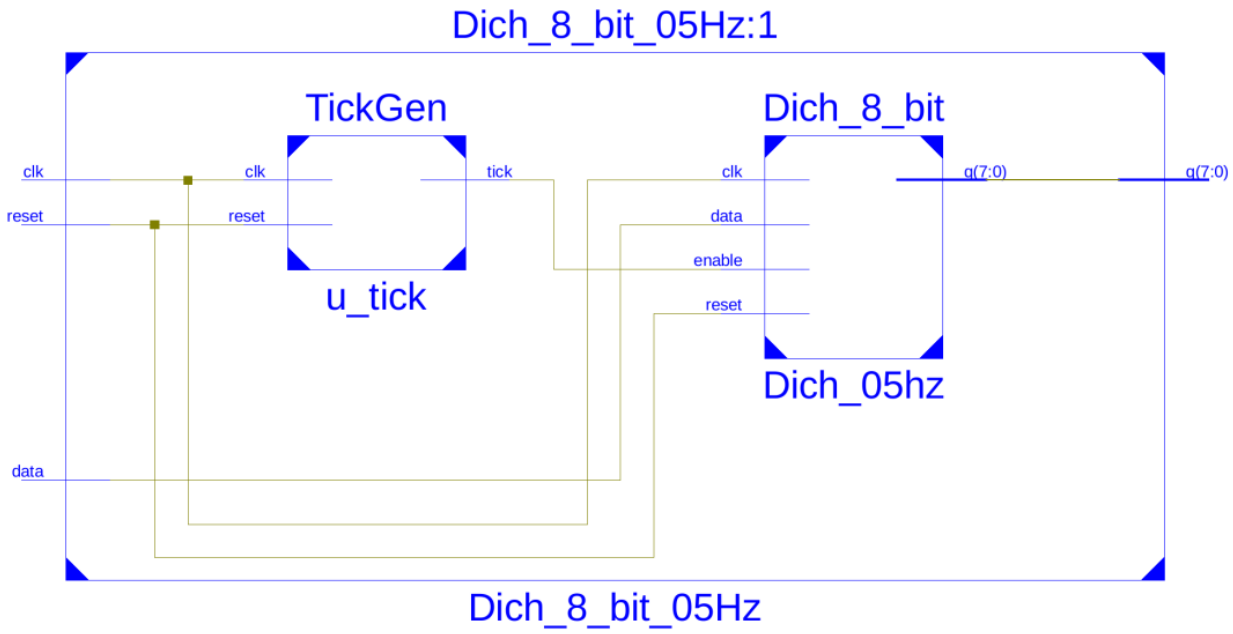
Students: Lê Quang Minh Nhật

Students ID: 23139031

Date: 3/10/2025

2. Block Diagram

Draw block diagram here.



3. State Transition Diagram

Describe Truth table or Design requirements or the FSM state diagram, etc

TickGen (đang inst với M = 25_000_000)

reset	r (hiện tại)	tick (hiện tại)	r (kế tiếp)
1	bất kỳ	0	0
0	0 … M−2	0	r + 1
0	M−1	1	0

Dich_8_bit (thanh ghi dịch trái, serial-in = data)

reset	enable	q (hiện tại)	data	q (kế tiếp)
1	x	bất kỳ	0	0000_0000
1	x	bất kỳ	1	0000_0001
0	0	bất kỳ	x	giữ nguyên (q)
0	1	q7…q0	0	{q6…q0, 0}
0	1	q7…q0	1	{q6…q0, 1}

Dich_8_bit_05Hz (TickGen → enable cho Dich_8_bit)

reset	tick_05Hz	q (hiện tại)	data	q (kế tiếp)	Ghi chú
1	x	bất kỳ	0	0000_0000	reset tải theo data
1	x	bất kỳ	1	0000_0001	
0	0	bất kỳ	x	giữ nguyên (q)	không có tick
0	1	q7…q0	0	{q6…q0, 0}	dịch trái, bơm 0
0	1	q7…q0	1	{q6…q0, 1}	dịch trái, bơm 1

Mô đum TickGen

```
module TickGen
#(parameter M = 50_000_000)
(
   input wire clk,
   input wire reset,
   output wire tick
   );


        reg [30:0] r;
        always @(posedge clk or posedge reset) begin
```

```
            if (reset)
                r <= 0;
            else if (r == M-1)
                r <= 0;
            else
                r <= r + 1;
    end

    assign tick = (r == M-1);

endmodule
```

Mô đum Dich_8_bit

```
module Dich_8_bit(
    input wire clk,
    input wire reset,
    input wire enable,
    input wire data,
    output reg [7:0] q
);

    always @(posedge clk or posedge reset) begin
        if (reset) begin
            q <= ({7'b0, data});
        end
        else if (enable) begin
```

```
                q <= {q[6:0], data};
            end
        end

endmodule
```

Mô đum Dich_8_bit_05Hz

```
module Dich_8_bit_05Hz(
    input wire clk,
    input wire reset,
    input wire data,
    output wire [7:0] q
);

    wire tick_05Hz;

    TickGen #(25_000_000) u_tick (.clk(clk), .reset(reset), .tick(tick_05Hz));

    Dich_8_bit Dich_05hz(.clk(clk),.reset(reset), .enable(tick_05Hz), .data(data), .q(q));

endmodule
```

4. Test Cases Overview

TC ID	Purpose	Input	Expected Output	Result (Pass/Fail)
TC1	Kiểm tra reset	reset = 1	Led bị reset	Pass
TC2	Kiểm tra khi nạp data = 1	clk = ~clk data = 1	Nạp bit 1 vào	Pass
TC3	Kiểm tra khi nạp data = 0	clk = ~clk data = 0	Nạp bit 0 vào	Pass

Mô đum Testbench_Dich_8_bit

```
`timescale 1ns / 1ps


module Testbench_Dich_8_bit;


	// Inputs
	reg clk;
	reg reset;
	reg enable;
	reg data;

	// Outputs
	wire [7:0] q;

	// Instantiate the Unit Under Test (UUT)
	Dich_8_bit uut (
		.clk(clk),
		.reset(reset),
		.enable(enable),
		.data(data),
		.q(q)
	);

	initial begin
		// Initialize Inputs
		clk = 0;
```

```
        reset = 0;
        enable = 0;
        data = 0;

        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
        reset = 1;
        #100;
        reset = 0;
        data = 1;
        #1000;
        data = 0;
    end

        always forever #10 clk = ~clk;
        always forever #10 enable = ~enable;

endmodule
```

5. Testcase Details

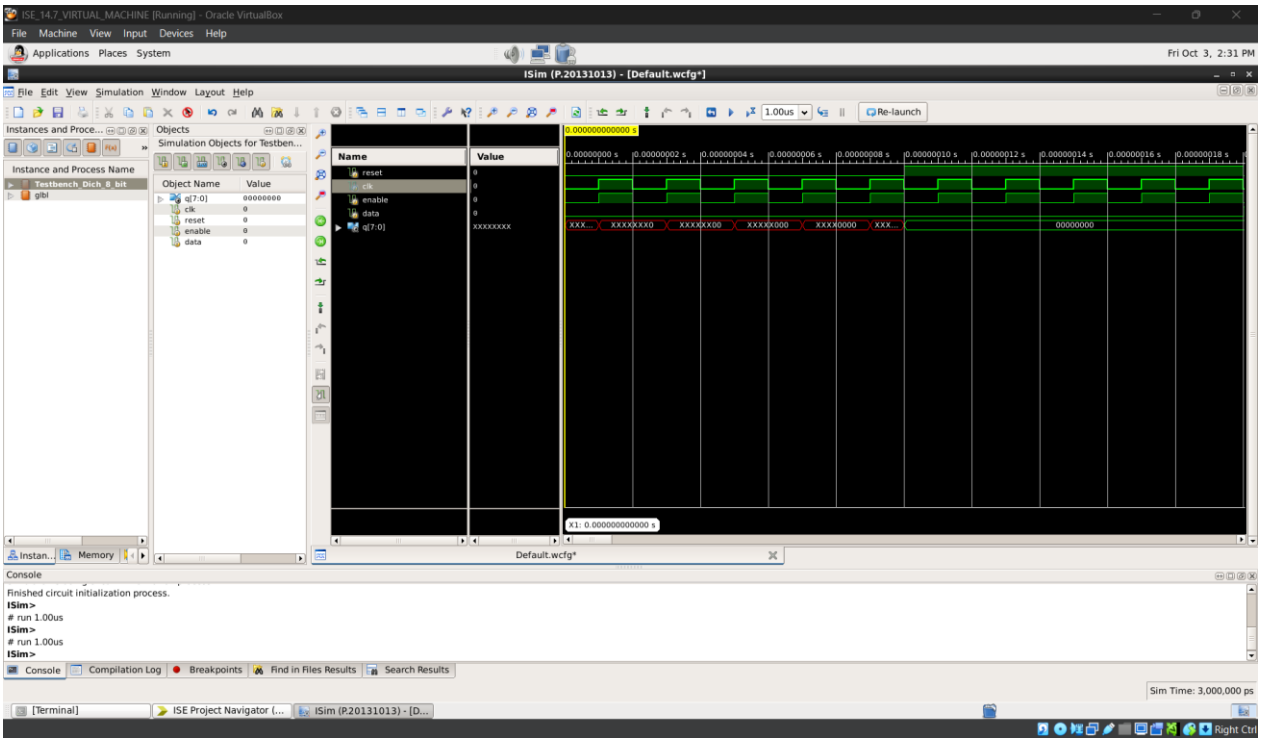
Testcase 1: TC1

Purpose: Kiểm tra reset

Input/Stimulus: reset = 1

Expected Output: Led bị reset

Waveform Simulation (attach/insert image):



Analysis:

Mạch bị reset và có thể dịch

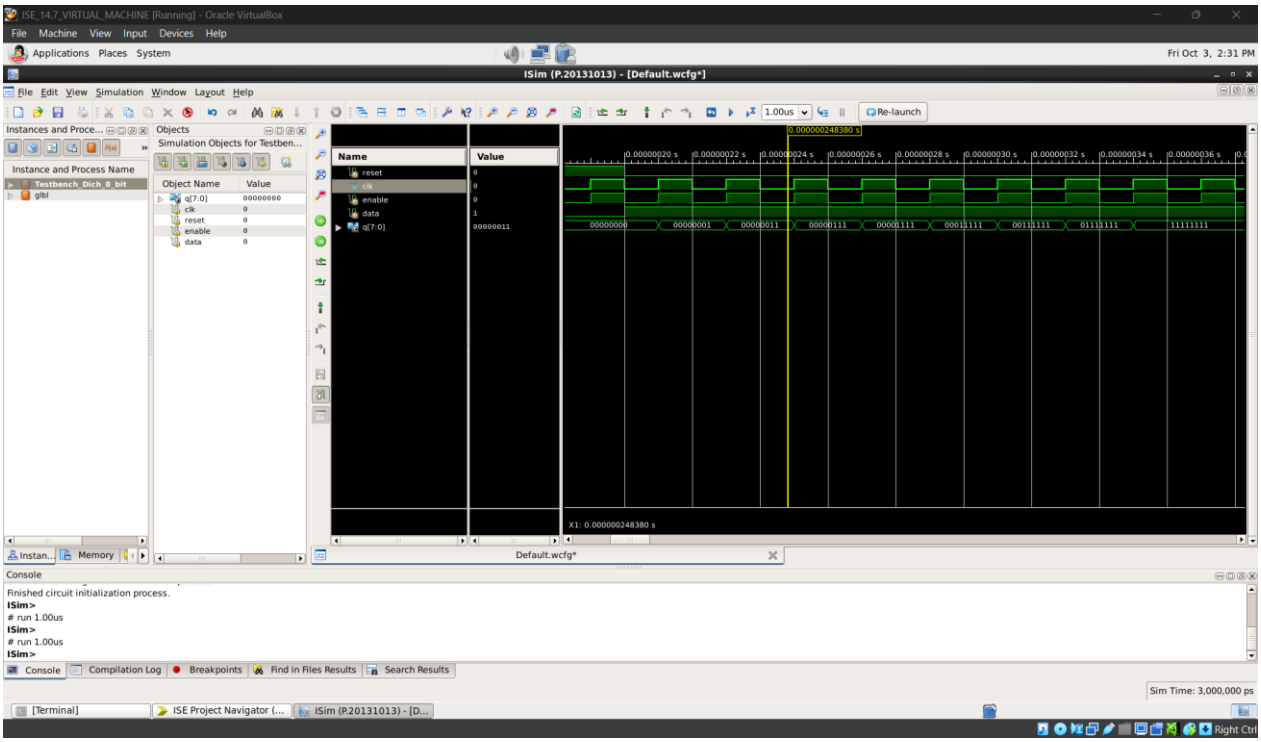
Testcase 2: TC2

Purpose: Kiểm tra khi nạp data = 1

Input/Stimulus: clk = ~clk, data = 1

Expected Output: Nạp bit 1 vào

Waveform Simulation (attach/insert image):



Analysis:

Mạch dịch bit 1 vào

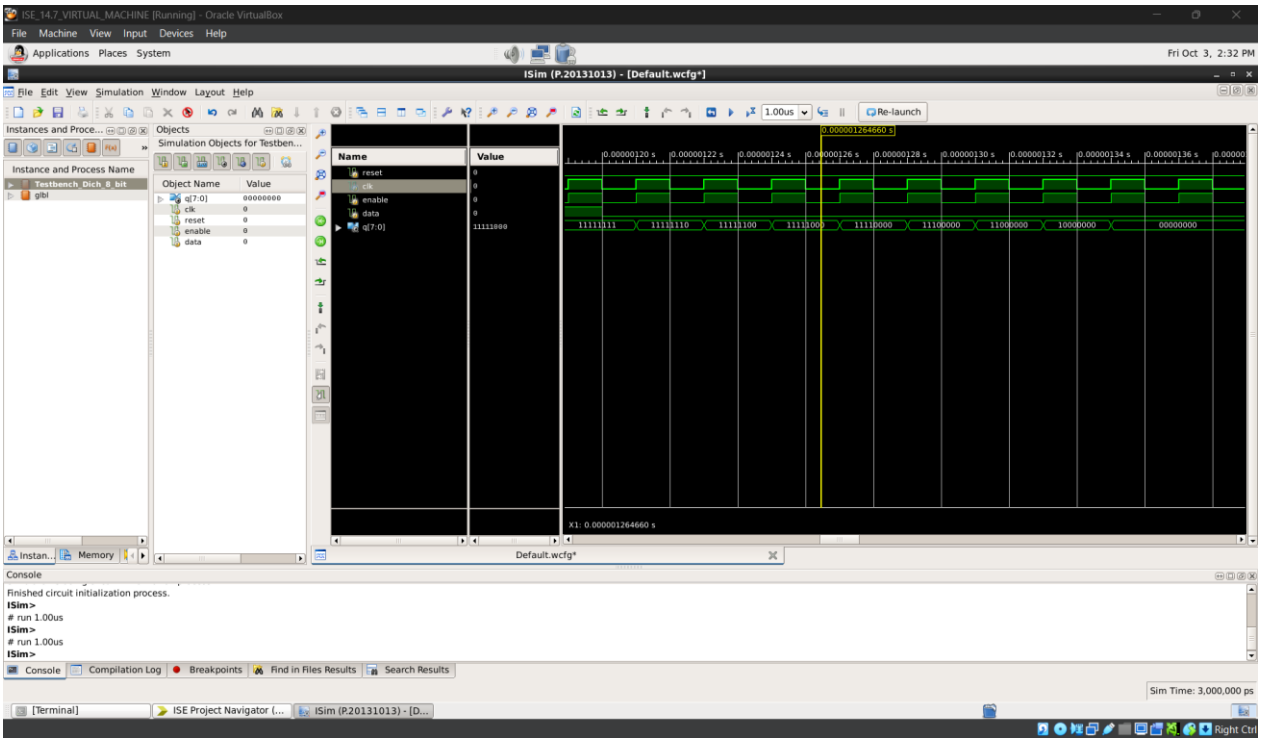
Testcase 3: TC3

Purpose: Kiểm tra khi nạp data = 0

Input/Stimulus: clk = ~clk, data = 0

Expected Output: Nạp bit 0 vào

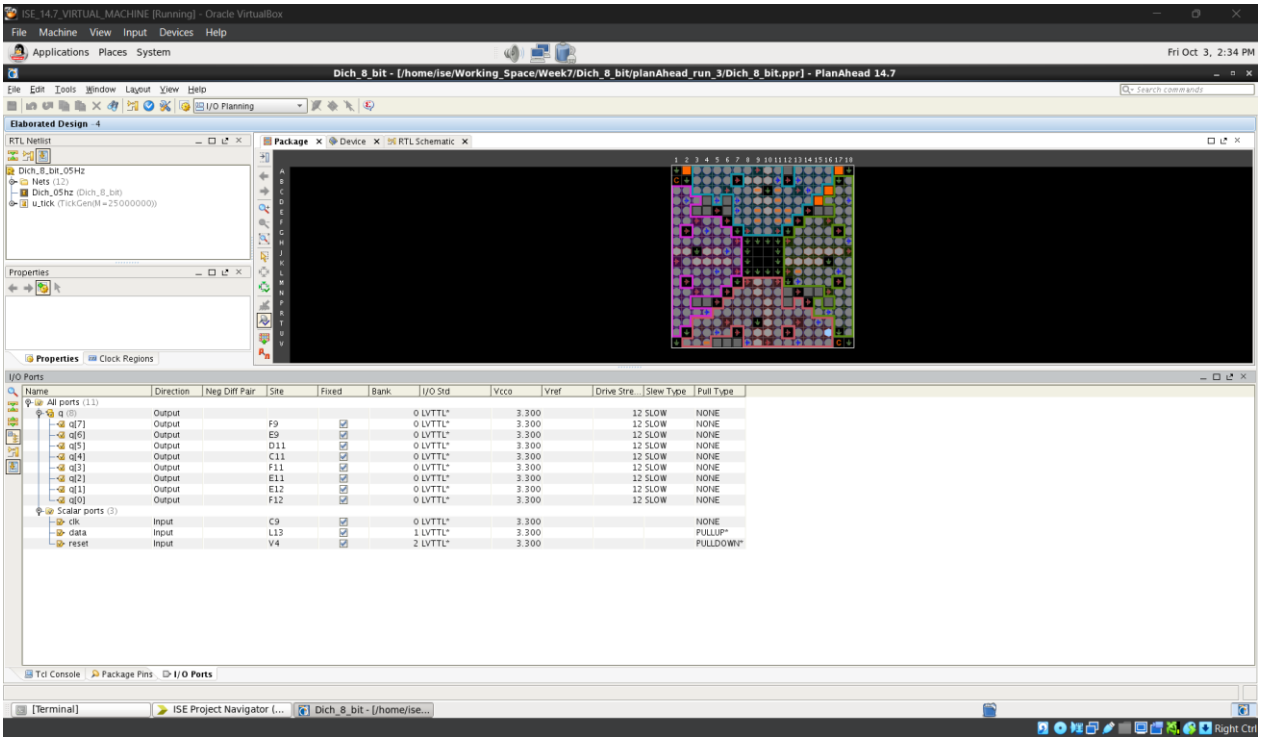
Waveform Simulation (attach/insert image):



Analysis:

Mạch dịch bit 0 vào.

6. Summary



```
NET "clk" IOSTANDARD = LVTTL;

NET "reset" IOSTANDARD = LVTTL;



NET "clk" LOC = C9;

NET "reset" LOC = V4;



NET "reset" PULLDOWN;

NET "q[7]" IOSTANDARD = LVTTL;

NET "q[6]" IOSTANDARD = LVTTL;

NET "q[5]" IOSTANDARD = LVTTL;

NET "q[4]" IOSTANDARD = LVTTL;

NET "q[3]" IOSTANDARD = LVTTL;

NET "q[2]" IOSTANDARD = LVTTL;

NET "q[1]" IOSTANDARD = LVTTL;

NET "q[0]" IOSTANDARD = LVTTL;



NET "q[7]" LOC = F9;

NET "q[6]" LOC = E9;

NET "q[5]" LOC = D11;

NET "q[4]" LOC = C11;

NET "q[3]" LOC = F11;

NET "q[2]" LOC = E11;
```

```
NET "q[1]" LOC = E12;
NET "q[0]" LOC = F12;

# PlanAhead Generated IO constraints

NET "data" IOSTANDARD = LVTTL;

# PlanAhead Generated physical constraints

NET "data" LOC = L13;

# PlanAhead Generated IO constraints

NET "data" PULLUP;
```

WEEKLY REPORT – DIGITAL SYSTEM DESIGN USING VERILOG & FPGA

1. General Information

Project Title: Led_Don_Sang_Chay

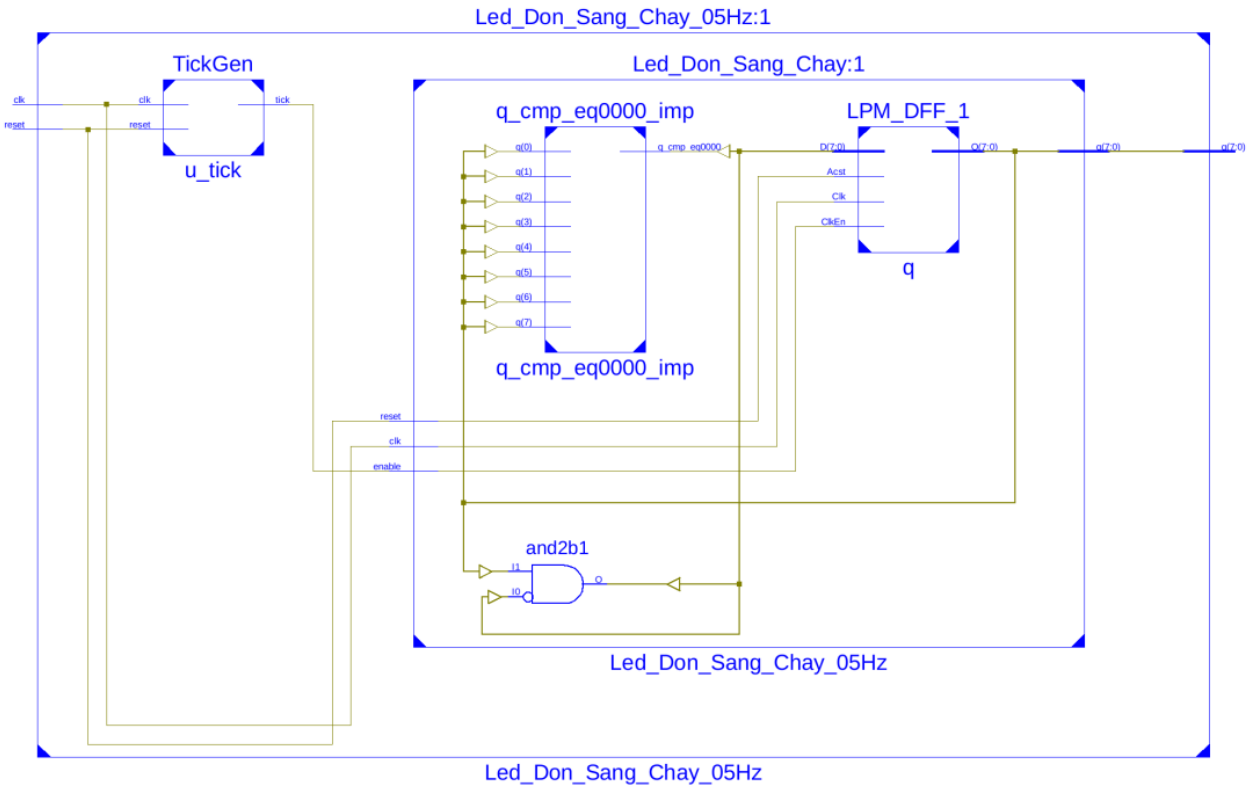
Students: Lê Quang Minh Nhật

Students ID: 23139031

Date: 3/10/2025

2. Block Diagram

Draw block diagram here.



3. State Transition Diagram

Describe Truth table or Design requirements or the FSM state diagram, etc

TickGen (instantiated với M = 25_000_000)

reset	r (hiện tại)	tick (hiện tại)	r (kế tiếp)
1	bất kỳ	0	0

0	0 … M−2	0	r + 1
0	M−1	1	0

Led_Don_Sang_Chay (walking-1, dịch trái, vòng về 1)

reset	enable	q (hiện tại)	q (kế tiếp)
1	x	bất kỳ	0000_0001
0	0	bất kỳ	giữ nguyên
0	1	0000_0001	0000_0010
0	1	0000_0010	0000_0100
0	1	0000_0100	0000_1000
0	1	0001_0000	0010_0000
0	1	0010_0000	0100_0000
0	1	0100_0000	1000_0000
0	1	1000_0000	0000_0001

Led_Don_Sang_Chay_05Hz (TickGen → enable cho LED)

reset	tick_05Hz	q (hiện tại)	q (kế tiếp)	Diễn giải
1	x	bất kỳ	0000_0001	reset đưa về LED0 sáng
0	0	bất kỳ	giữ nguyên	không có tick ⇒ đứng
0	1	0000_0001	0000_0010	dịch trái 1 bước
0	1	0000_0010	0000_0100	…
0	1	…	…	…
0	1	1000_0000	0000_0001	quay vòng

Mô đum TickGen

```
module TickGen

#(parameter M = 50_000_000)

(
```

```
    input wire clk,
    input wire reset,
    output wire tick
    );


        reg [30:0] r;
        always @(posedge clk or posedge reset) begin
                if (reset)
                        r <= 0;
                else if (r == M-1)
                        r <= 0;
                else
                        r <= r + 1;
  end


  assign tick = (r == M-1);


endmodule
```

Mô đum Led_Don_Sang_Chay

```
module Led_Don_Sang_Chay(
        input wire clk,
        input wire reset,
        input wire enable,
        output reg [7:0] q
);
```

```
    always @(posedge clk or posedge reset) begin
        if (reset) begin
            q <= 8'b0000_0001;
        end
        else if (enable) begin
            q <= (q == 8'b1000_0000) ? 8'b0000_0001 : (q << 1);
        end
    end

endmodule
```

Mô đum Led_Don_Sang_Chay_05Hz

```
module Led_Don_Sang_Chay_05Hz(
   input wire clk,
   input wire reset,
   output wire [7:0] q
);

    wire tick_05Hz;

    TickGen #(25_000_000) u_tick (.clk(clk), .reset(reset), .tick(tick_05Hz));

    Led_Don_Sang_Chay Led_Don_Sang_Chay_05Hz(.clk(clk),.reset(reset),
.enable(tick_05Hz), .q(q));
```

```
endmodule
```

4. Test Cases Overview

TC ID	Purpose	Input	Expected Output	Result (Pass/Fail)
TC1	Kiểm tra reset	reset = 1	Mạch bị reset	Pass
TC2	Kiểm tra Led chạy	reset = 0 clk = ~clk	Mạch bắt đầu chạy	Pass

Mô đum Testbench_Led_Don_Sang_Chay

```
`timescale 1ns / 1ps


module Testbench_Led_Don_Sang_Chay;


	// Inputs
	reg clk;
	reg reset;
	reg enable;

	// Outputs
	wire [7:0] q;

	// Instantiate the Unit Under Test (UUT)
	Led_Don_Sang_Chay uut (
		.clk(clk),
		.reset(reset),
		.enable(enable),
		.q(q)
	);


	initial begin
```

```
        // Initialize Inputs
        clk = 0;
        reset = 0;
        enable = 0;

        // Wait 100 ns for global reset to finish
        #100;
        reset = 1;
        #100;
        reset = 0;
    end

        always forever #10 clk = ~clk;
        always forever #10 enable = ~enable;

endmodule
```

5. Testcase Details

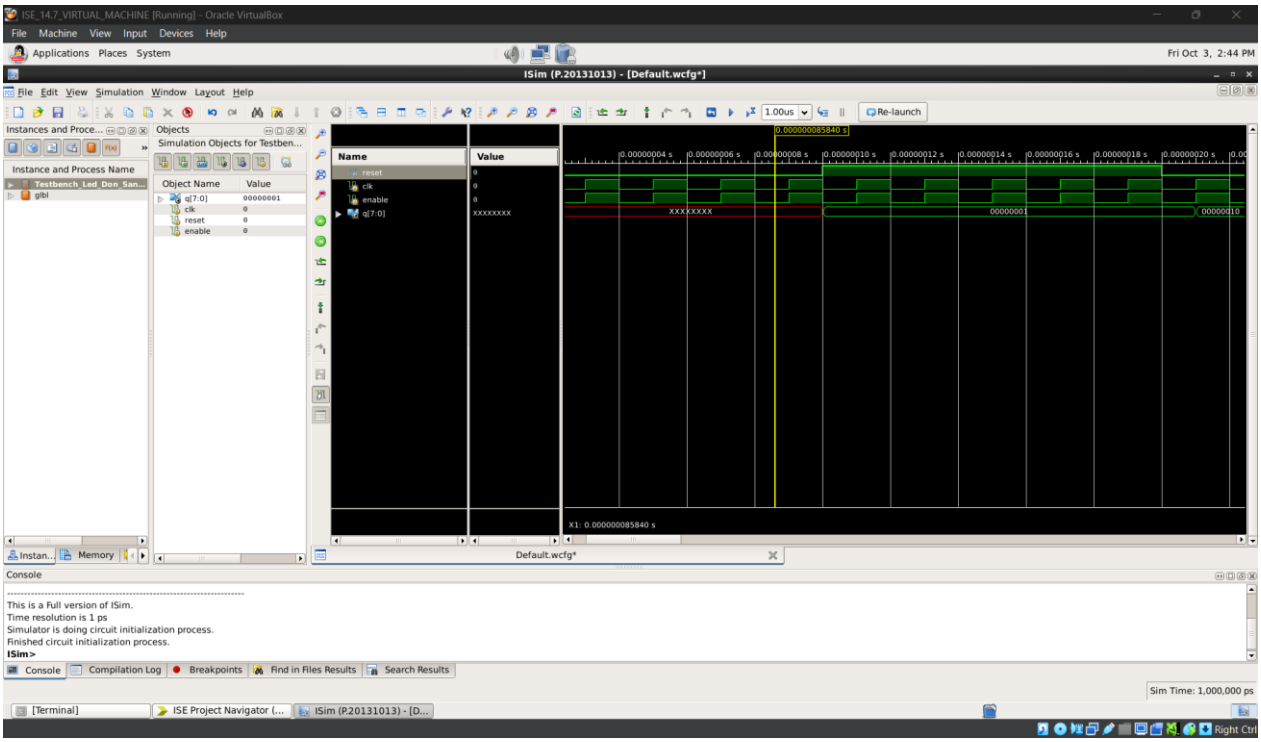
Testcase 1: TC1

Purpose: Kiểm tra reset

Input/Stimulus: reset = 1

Expected Output: Mạch bị reset

Waveform Simulation (attach/insert image):



Analysis:

Mạch bị reset về 8’b0000_0001

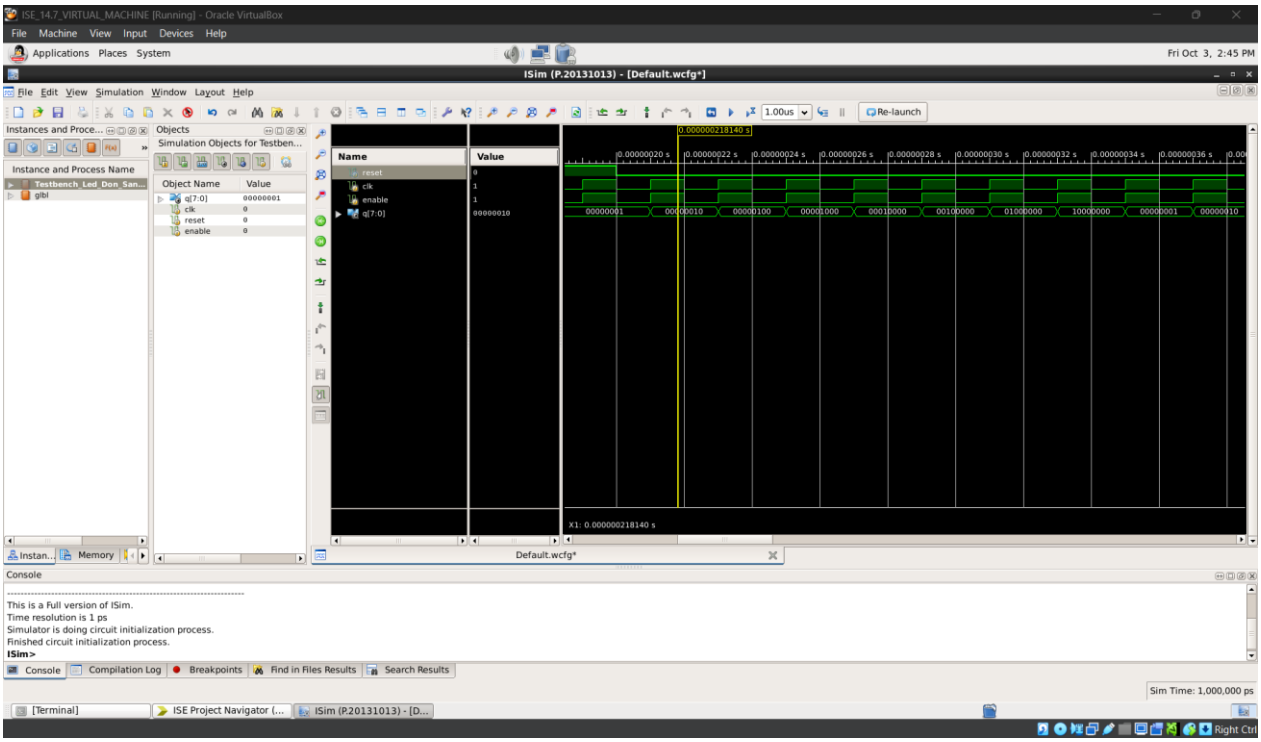
Testcase 2: TC2

Purpose: Kiểm tra Led chạy

Input/Stimulus: reset = 0, clk = ~clk

Expected Output: Mạch bắt đầu chạy

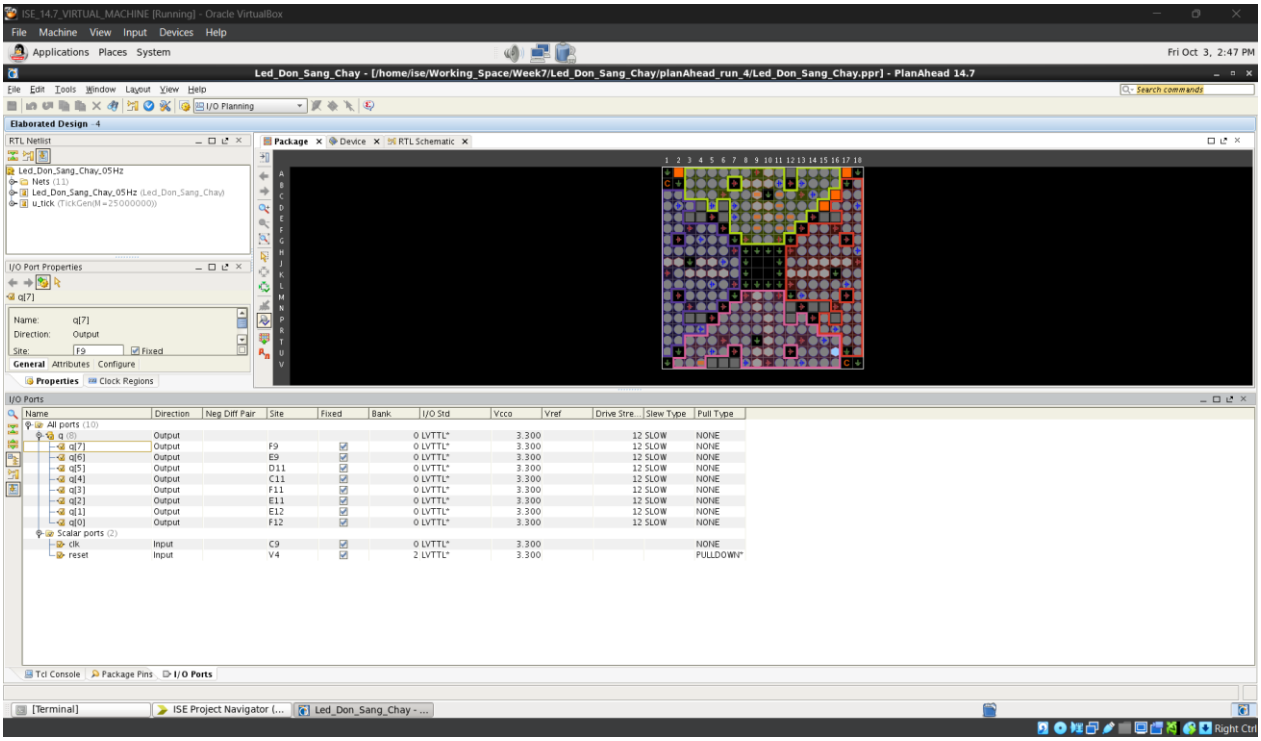
Waveform Simulation (attach/insert image):



Analysis:

Mạch chạy đúng yêu cầu bài toán.

6. Summary



```
NET "clk" IOSTANDARD = LVTTL;

NET "reset" IOSTANDARD = LVTTL;

NET "q[7]" IOSTANDARD = LVTTL;

NET "q[6]" IOSTANDARD = LVTTL;

NET "q[5]" IOSTANDARD = LVTTL;

NET "q[4]" IOSTANDARD = LVTTL;

NET "q[3]" IOSTANDARD = LVTTL;

NET "q[2]" IOSTANDARD = LVTTL;

NET "q[1]" IOSTANDARD = LVTTL;

NET "q[0]" IOSTANDARD = LVTTL;



NET "reset" LOC = V4;

NET "clk" LOC = C9;

NET "q[7]" LOC = F9;

NET "q[6]" LOC = E9;

NET "q[5]" LOC = D11;

NET "q[4]" LOC = C11;

NET "q[3]" LOC = F11;

NET "q[2]" LOC = E11;



NET "reset" PULLDOWN;


# PlanAhead Generated physical constraints
```

```
NET "q[1]" LOC = E12;
NET "q[0]" LOC = F12;
```

WEEKLY REPORT – DIGITAL SYSTEM DESIGN USING VERILOG & FPGA

1. General Information

Project Title: Sang_Dan_Tat_Dan

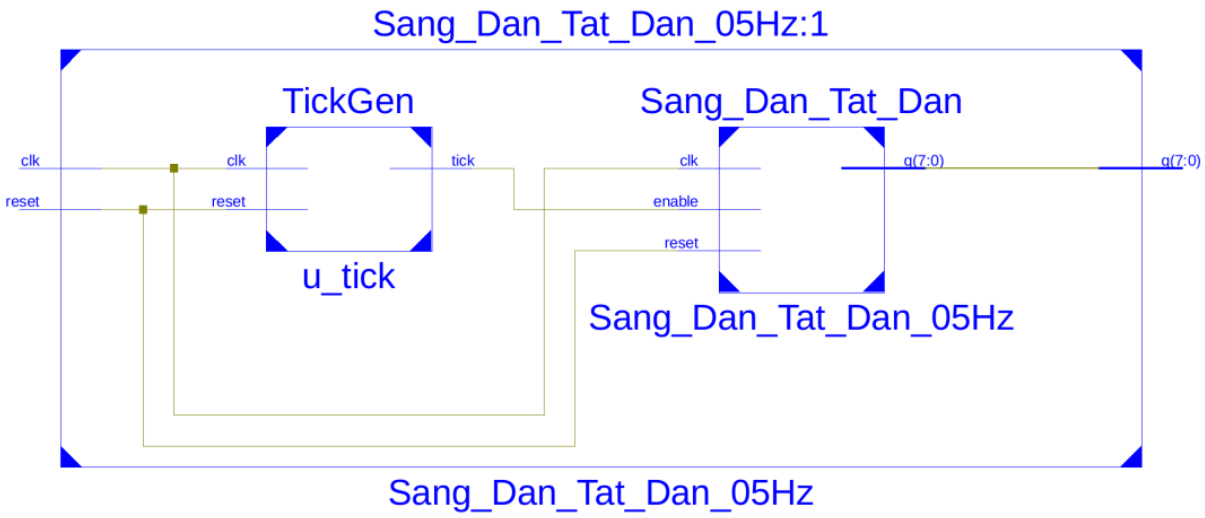
Students: Lê Quang Minh Nhật

Students ID: 23139031

Date: 3/10/2025

2. Block Diagram

Draw block diagram here.



3. State Transition Diagram

Describe Truth table or Design requirements or the FSM state diagram, etc

TickGen (instance dùng M = 25_000_000 → 2 Hz nếu clk=50 MHz)

reset	r (hiện tại)	tick	r (kế tiếp)
1	bất kỳ	0	0
0	0 … M−2	0	r + 1
0	M−1	1	0

Sang_Dan_Tat_Dan (dịch trái, bơm data; "sáng dần rồi tắt dần" cùng một hướng)

reset	enable	q (hiện tại)	data	q (kế tiếp)	data (kế tiếp)	Ghi chú

1	x	bất kỳ	x	0000_0001	1	seed
0	0	bất kỳ	x	giữ nguyên	giữ nguyên	không có tick
0	1	8'h00	0	8'h00	1	đứng 1 nhịp ở 00 rồi chuyển sang bơm 1
0	1	8'h00	1	8'h01	1	bắt đầu sáng dần
0	1	0x01..0x7F	1	`{q<<1	1}`	1
0	1	8'hFF	1	8'hFF	0	đứng 1 nhịp ở FF rồi chuyển bơm 0
0	1	8'hFF	0	8'hFE	0	bắt đầu tắt dần
0	1	0x80..0x01	0	{q>>1}	0	giảm dần số bit 1

Sang_Dan_Tat_Dan_05Hz (TickGen → enable)

enable = tick_05Hz → mỗi 0,5 s (nếu clk=50 MHz) trạng thái cập nhật một lần theo bảng trên.

q: 01 → 03 → 07 → 0F → 1F → 3F → 7F → FF → FF → FE → FC → F8 → F0 → E0 → C0 → 80 → 00 → 00 → 01 → 03 → ⋯

Mô đum TickGen

```
module TickGen
#(parameter M = 50_000_000)
(
   input wire clk,
   input wire reset,
   output wire tick
   );


        reg [30:0] r;
        always @(posedge clk or posedge reset) begin
```

```
        if (reset)
                r <= 0;
        else if (r == M-1)
                r <= 0;
        else
                r <= r + 1;
  end

  assign tick = (r == M-1);

endmodule
```

Mô đum Sang_Dan_Tat_Dan

```
module Sang_Dan_Tat_Dan(
   input wire clk,
   input wire reset,
   input wire enable,
   output reg [7:0] q
   );

        reg data;

        always @(posedge clk or posedge reset) begin
                if (reset) begin
                        q <= 8'b0000_0001;
                        data <= 1'b1;
```

```
        end else if (enable) begin
            if (q == 8'b1111_1111) begin
                data <= 1'b0;
            end else if (q == 8'b0000_0000) begin
                data <= 1'b1;
            end
            q <= {q[6:0], data};
        end
    end

endmodule
```

Mô đum Sang_Dan_Tat_Dan_05Hz

```
module Sang_Dan_Tat_Dan_05Hz(
   input wire clk,
   input wire reset,
   output wire [7:0] q
   );

    wire tick_05Hz;

    TickGen #(25_000_000) u_tick (.clk(clk), .reset(reset), .tick(tick_05Hz));

    Sang_Dan_Tat_Dan Sang_Dan_Tat_Dan_05Hz(.clk(clk),.reset(reset),
.enable(tick_05Hz), .q(q));
```

```
endmodule
```

4. Test Cases Overview

TC ID	Purpose	Input	Expected Output	Result (Pass/Fail)
TC1	Kiểm tra reset	reset = 1	Mạch bị reset	Pass
TC2	Kiểm tra sáng dần	reset = 0 clk = ~clk	Led sáng dần	Pass
TC3	Kiểm tra tắt dần	reset = 0 clk = ~clk	Led tắt dần	Pass

Mô đum Testbench_Sang_Dan_Tat_Dan

```
`timescale 1ns / 1ps


module Testbench_Sang_Dan_Tat_Dan;


	// Inputs
	reg clk;
	reg reset;
	reg enable;

	// Outputs
	wire [7:0] q;

	// Instantiate the Unit Under Test (UUT)
	Sang_Dan_Tat_Dan uut (
		.clk(clk),
		.reset(reset),
		.enable(enable),
		.q(q)
	);
```

```
	initial begin
		// Initialize Inputs
		clk = 0;
		reset = 0;
		enable = 0;

		// Wait 100 ns for global reset to finish
		#100;

		// Add stimulus here
		reset = 1;
		#100;
		reset = 0;
	end

	always forever #10 clk = ~clk;
	always forever #10 enable = ~enable;

endmodule
```

5. Testcase Details

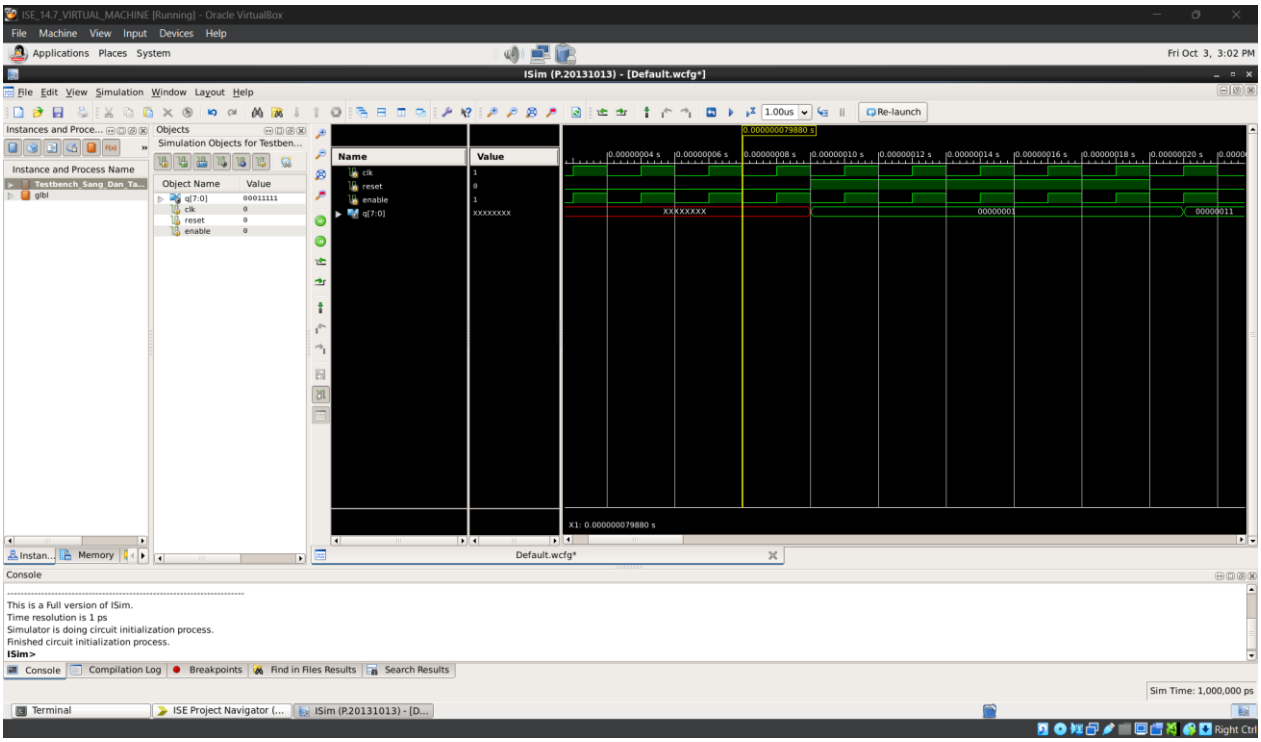
Testcase 1: TC1

Purpose: Kiểm tra reset

Input/Stimulus: reset = 1

Expected Output: Mạch bị reset

Waveform Simulation (attach/insert image):



Analysis:

Mạch bị reset do xung reset kích mức cao

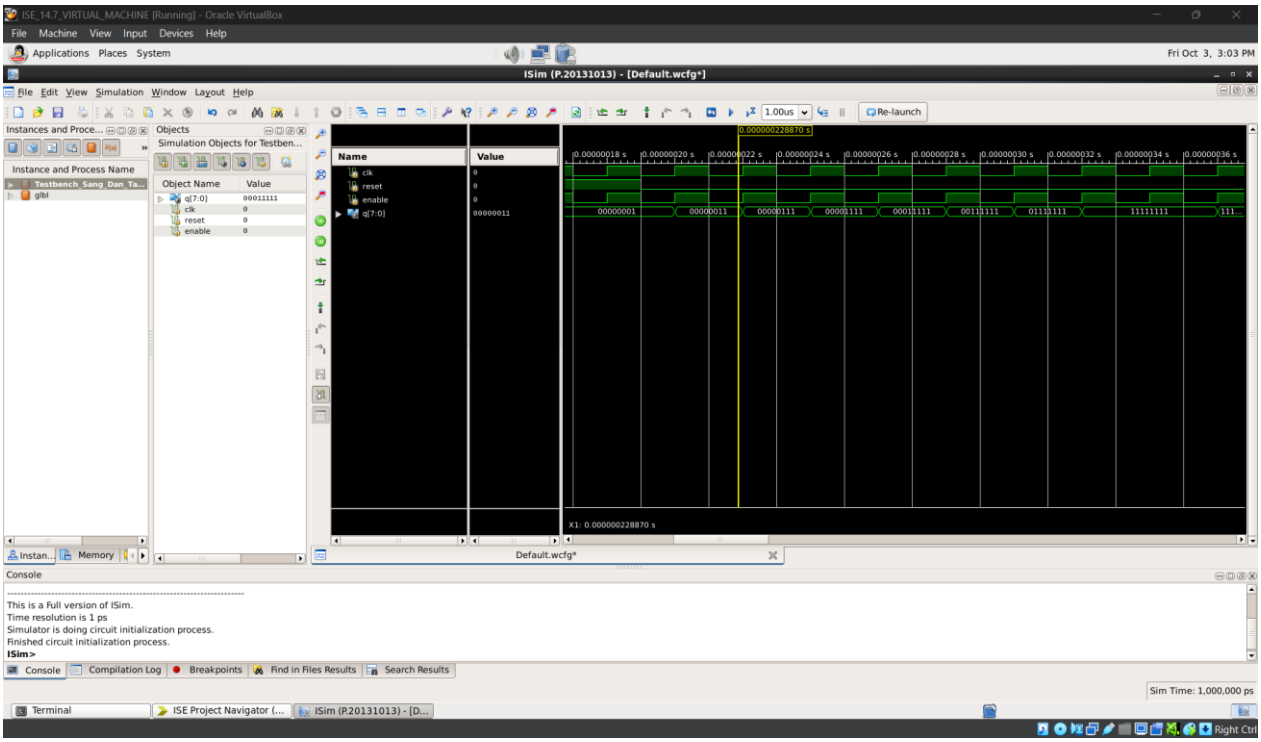
Testcase 2: TC2

Purpose: Kiểm tra sáng dần

Input/Stimulus: reset = 0, clk = ~clk

Expected Output: Mạch bị reset

Waveform Simulation (attach/insert image):



Analysis:

Mạch sáng dần từ bit trọng số thấp đến cao.

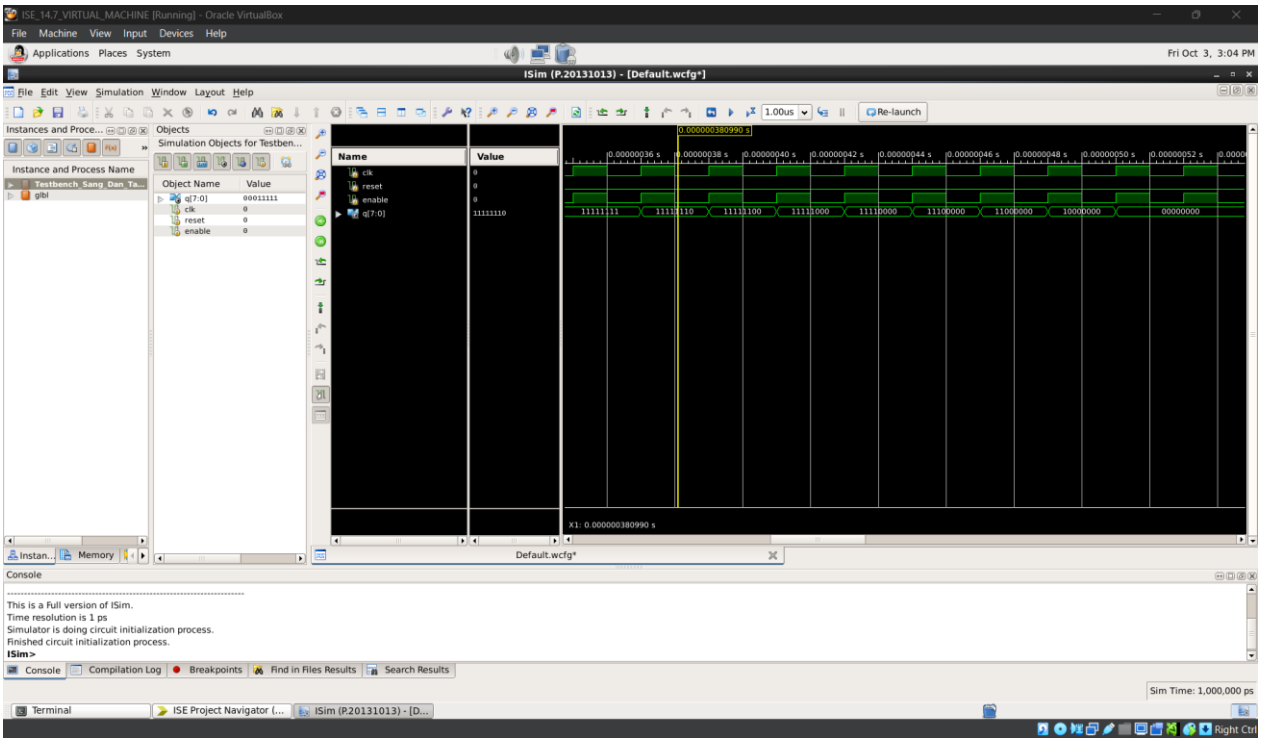
Testcase 3: TC3

Purpose: Kiểm tra tắt dần

Input/Stimulus: reset = 0, clk = ~clk

Expected Output: Led tắt dần

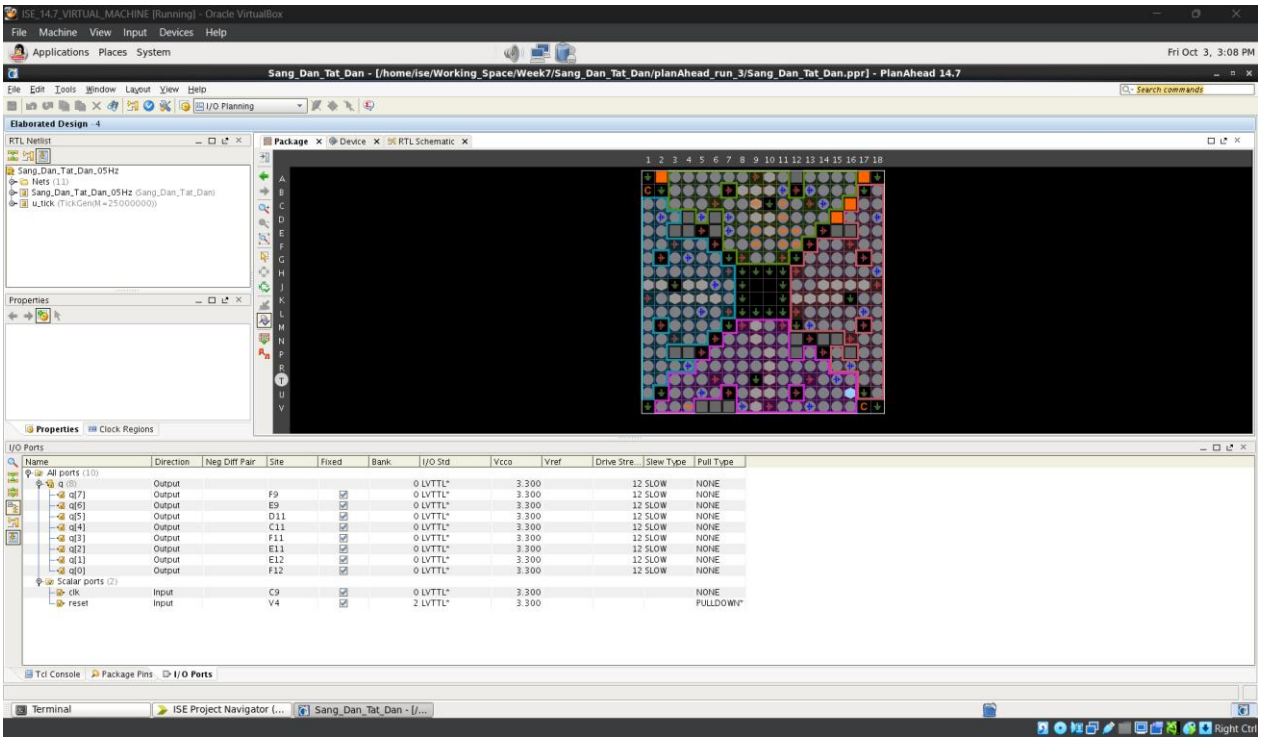
Waveform Simulation (attach/insert image):



Analysis:

Mạch tắc dần từ trọng số thấp đến cao.

6. Summary



```
# PlanAhead Generated IO constraints

NET "q[7]" IOSTANDARD = LVTTL;
NET "q[6]" IOSTANDARD = LVTTL;
NET "q[5]" IOSTANDARD = LVTTL;
NET "q[4]" IOSTANDARD = LVTTL;
NET "q[3]" IOSTANDARD = LVTTL;
NET "q[2]" IOSTANDARD = LVTTL;
NET "q[1]" IOSTANDARD = LVTTL;
NET "q[0]" IOSTANDARD = LVTTL;
NET "clk" IOSTANDARD = LVTTL;
NET "reset" IOSTANDARD = LVTTL;

# PlanAhead Generated physical constraints

NET "clk" LOC = C9;
NET "reset" LOC = V4;

# PlanAhead Generated IO constraints

NET "reset" PULLDOWN;

# PlanAhead Generated physical constraints

NET "q[7]" LOC = F9;
```

```
NET "q[6]" LOC = E9;

NET "q[5]" LOC = D11;

NET "q[4]" LOC = C11;

NET "q[3]" LOC = F11;

NET "q[2]" LOC = E11;

NET "q[1]" LOC = E12;

NET "q[0]" LOC = F12;
```