Họ và tên :Dương Minh Vương

MSV: 22022156

BÁO CÁO TUẦN 3+4

Mục Lục

I.Thiết kế mạch NAND 2
1.Đặc tả chức năng 2
2.Mô phỏng bằng ModelSim 2
3.Mô phỏng 3
4.Chạy trên quartus 3
5.Kiểm thử trên bo mạch 3
II.Mux 4:1 5
1.Đặc tả chức năng 5
2.Mô phỏng trên ModelSim 7
3.Mô phỏng 7
4.Chạy trên Quartus 8
5.Kiểm thử trên bo mạch 8
III.Tri State Buffer 11
1.Đặc tả chức năng 11
2.Mô phỏng trên ModelSim 11
3.Mô phỏng 12
4.Chạy trên Quartus 13
5.Kiểm thử trên bo mạch 13
IV.Encoder 4:2 14
1.Đặc tả chức năng 15
2.Mô phỏng trên Model Sim 15
3.Mô phỏng 16
4.Chạy trên Quartus 16
5.Kiểm thử trên bo mạch 16
V. Decode 2:4 19
1.Đặc tả chức năng 19
2.Mô phỏng trên ModelSim 20
3.Mô phỏng 20
4.Chạy trên Quartus 20
5.Kiểm thử trên bo mạch 21
VI.4 bit ripple carry full adder 23
1.Đặc tả chức năng 23
2.Chạy trên ModelSim 24
3.Mô phỏng 25
4.Chạy trên Quartus 25
5.Kiểm thử trên bo mạch 24

I.Thiết kế mạch NAND

1.Đặc tả chức năng

Inputs		Outputs
A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

$Z = \overline{A.B}$

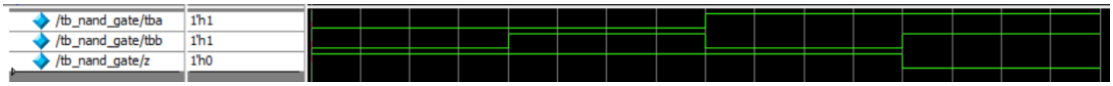
2.Mô phỏng bằng ModelSim

nand.sv	nand_tb.sv

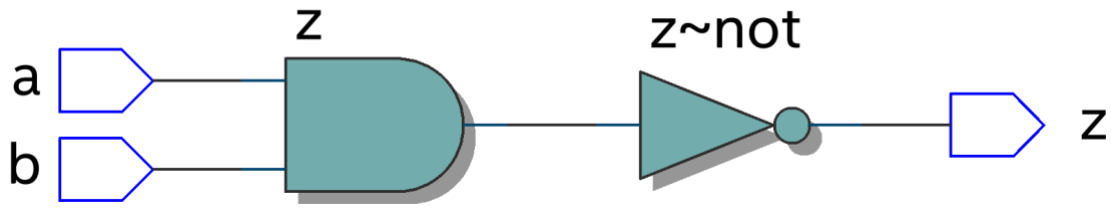
```
module nand_gate (
    input a, b,
    output z
);
    assign z = ~(a & b);
endmodule
```

```
`timescale 1ns/1ns

module tb_nand_gate();
    reg tba, tbb;
    wire z;

    nand_gate test (
        .a(tba),
        .b(tbb),
        .z(z)
    );

    initial begin
        tba = 0; tbb = 0; #20;
        tba = 0; tbb = 1; #20;
        tba = 1; tbb = 0; #20;
        tba = 1; tbb = 1; #20;

        $finish;
    end
endmodule
```

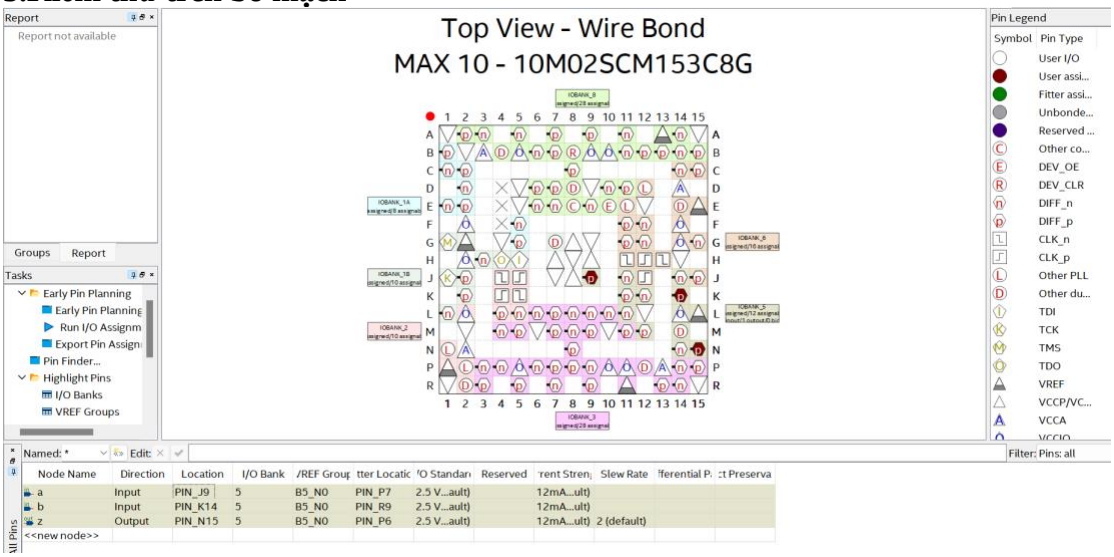
3.Mô phỏng



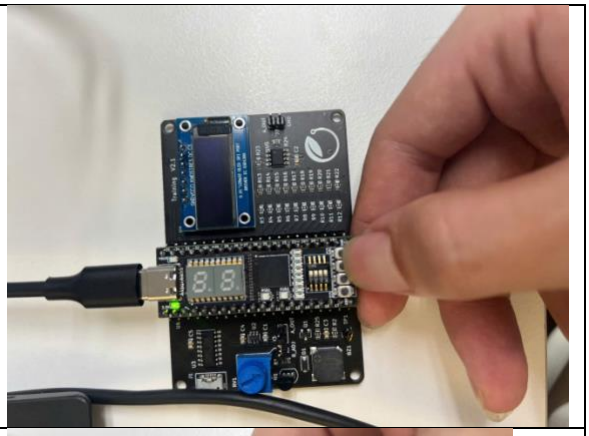
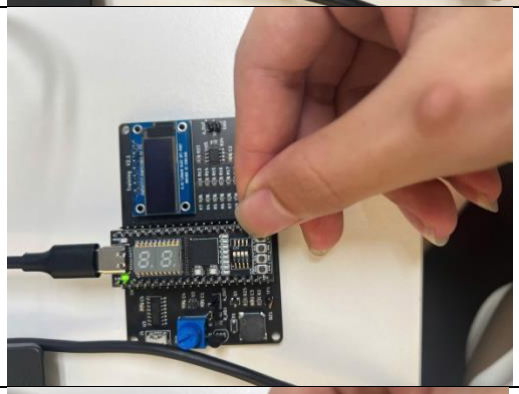
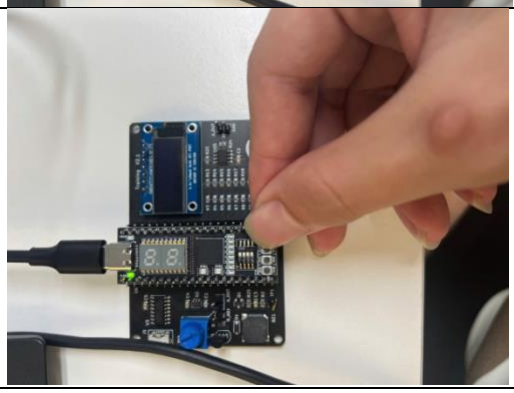
4.Chạy trên quartus

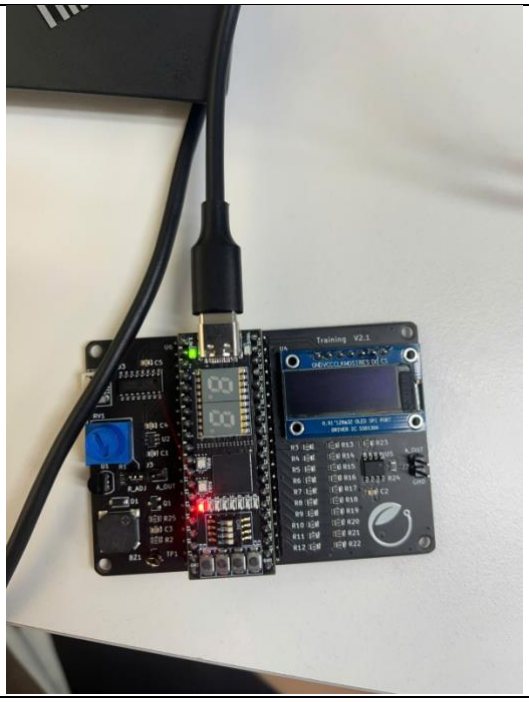


5.Kiểm thử trên bo mạch



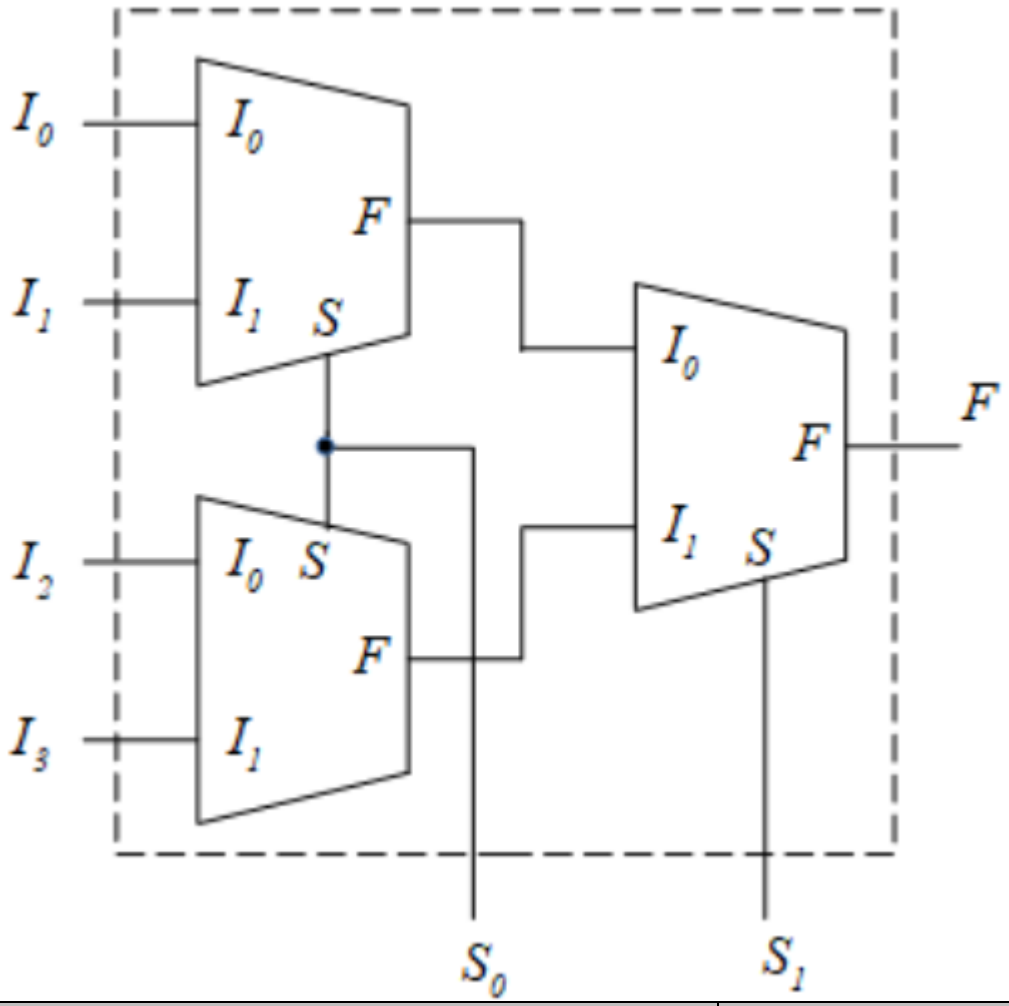
Trường hợp	Ảnh

a=0,b=0	
a=0,b=1	
a=1,b=0	

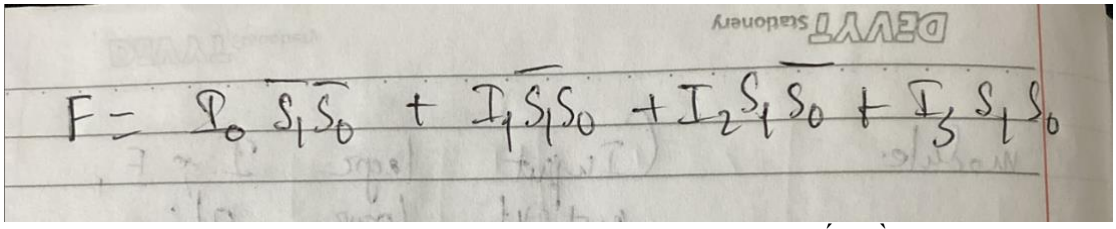
a=1,b=1	

II.Mux 4:1

1.Đặc tả chức năng



Inputs		Outputs
S1	S0	F
0	0	I0
0	1	I1
1	0	I2
1	1	I3



Bộ MUX 4-to-1 từ MUX 2-to-1 có chức năng chọn một trong bốn đầu vào dữ liệu (I0, I1, I2, I3) dựa trên hai tín hiệu chọn (S1, S0), và xuất đầu ra (F). Cụ thể:

MUX 4-to-1 sử dụng ba MUX 2-to-1 để chọn đầu vào:

+ Hai MUX 2-to-1 đầu tiên nhận cặp đầu vào (I0, I1) và (I2, I3) và dựa vào tín hiệu chọn thấp (S0) để chọn một trong hai đầu vào của mỗi cặp.

+ MUX 2-to-1 thứ ba nhận kết quả từ hai MUX trước và sử dụng tín hiệu chọn cao (S1) để quyết định đầu ra cuối cùng.

2.Mô phỏng trên ModelSim

mux_2to1.sv

```
module mux_2to1 (
    input wire I0, I1,
    input wire S,
    output wire F
);

    assign F = S ? I1 : I0;

endmodule
```

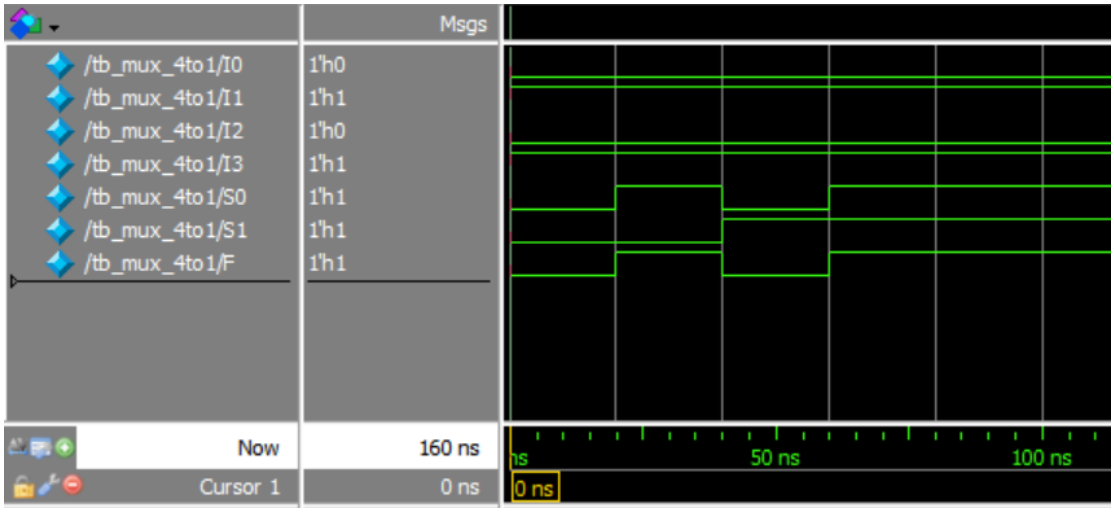
mux_4to1.sv

```
module mux_4to1 (
    input wire I0, I1, I2, I3,
    input wire S0, S1,
    output wire F
);

    wire F1, F2;

    mux_2to1 mux1 (.I0(I0), .I1(I1), .S(S0), .F(F1));
    mux_2to1 mux2 (.I0(I2), .I1(I3), .S(S0), .F(F2));
    mux_2to1 mux3 (.I0(F1), .I1(F2), .S(S1), .F(F));

endmodule
```

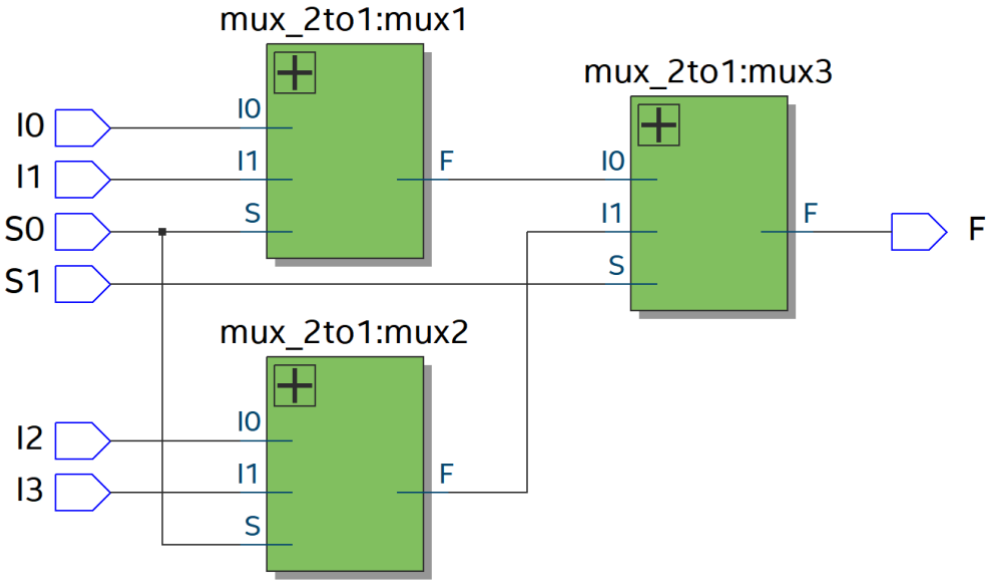
mux_4to1_tb.sv

```
`timescale 1ns/1ns

module tb_mux_4to1();
    reg I0, I1, I2, I3;
    reg S0, S1;
    wire F;

    mux_4to1 test (
        .I0(I0),
        .I1(I1),
        .I2(I2),
        .I3(I3),
        .S0(S0),
        .S1(S1),
        .F(F)
    );

    initial begin
        I0 = 0; I1 = 1; I2 = 0; I3 = 1;
        S0 = 0; S1 = 0; #20;
        S0 = 1; S1 = 0; #20;
        S0 = 0; S1 = 1; #20;
        S0 = 1; S1 = 1; #20;
        $finish;
    end

endmodule
```

3.Mô phỏng



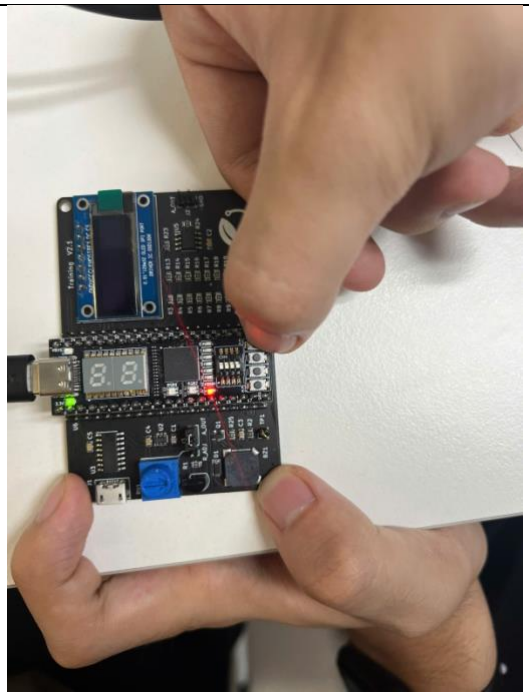
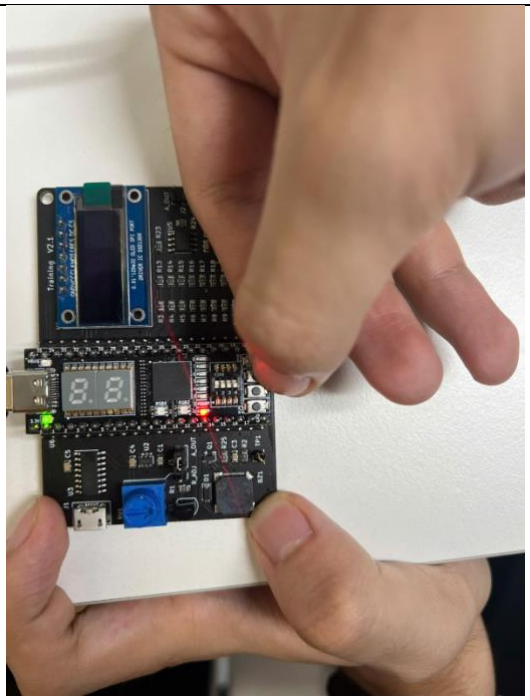
4.Chạy trên Quartus

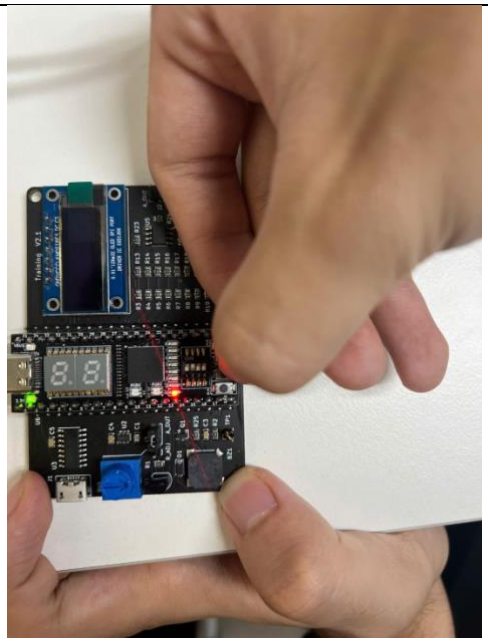
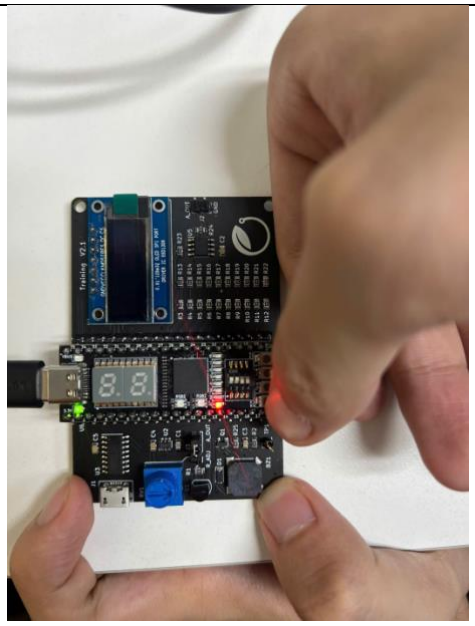


5.Kiểm thử trên bo mạch

F	Output	PIN_N15	5	B5_N0	PIN_M4	2.5 V...ault)	12mA...ult)	2 (default)
I0	Input	PIN_J9	5	B5_N0	PIN_L6	2.5 V...ault)	12mA...ult)	
I1	Input	PIN_K14	5	B5_N0	PIN_L8	2.5 V...ault)	12mA...ult)	
I2	Input	PIN_J11	5	B5_N0	PIN_R11	2.5 V...ault)	12mA...ult)	
I3	Input	PIN_J14	5	B5_N0	PIN_P15	2.5 V...ault)	12mA...ult)	
S0	Input	PIN_H11	6	B6_N0	PIN_N8	2.5 V...ault)	12mA...ult)	
S1	Input	PIN_J12	6	B6_N0	PIN_P7	2.5 V...ault)	12mA...ult)	

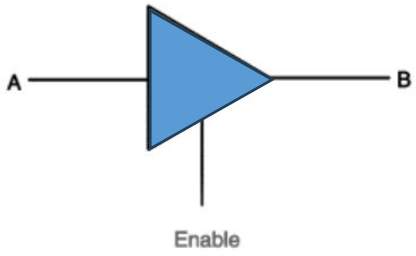
Trường hợp	Ảnh

S1S0=00, I0=0, I1=1, I2=1, I3=1	
S1S0=01, I0=1, I1=0, I2=1, I3=1	

S1S0=10, I0=1, I1=1, I2=0, I3=1	
S1S0=11, I0=1, I1=1, I2=1, I3=0	

III.Tri State Buffer

1.Đặc tả chức năng



Enable	A	B
0	0	Z
0	1	Z
1	0	0
1	1	1

B= (Enable) ? A : 1’bz

Tri state buffer sẽ có 3 trạng thái đầu ra .

Khi chân Enable = 1 thì bộ đệm sẽ chuyển đầu vào A đến đầu ra B.
Khi chân Enable = 0 thì bộ đêm sẽ chặn đầu vào và đầu ra mà nó kết nối và được coi là tải điện trở kháng cao.

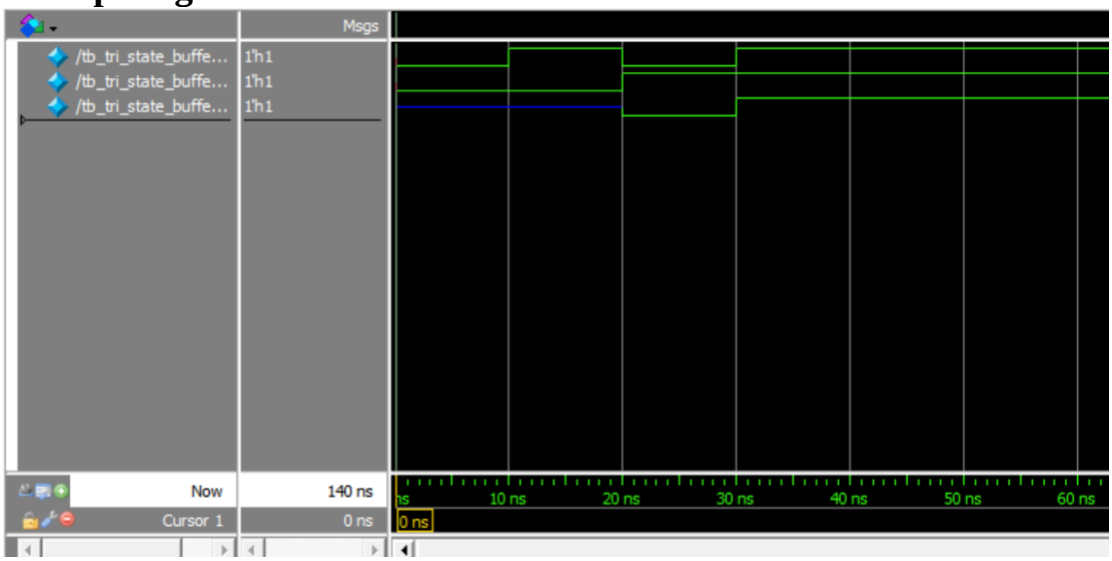
2.Mô phỏng trên ModelSim

tri_state_buffer	tri_state_buffer_tb

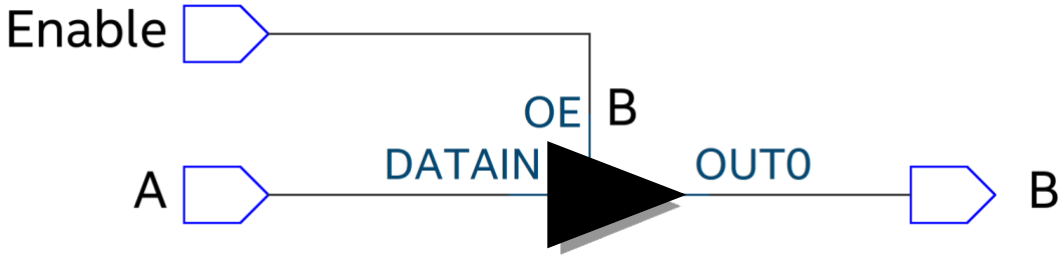
```
module tri_state_buffer (
    input wire A,
    input wire Enable,
    output wire B
);
    assign B = (Enable) ? A : 1'bz;
endmodule
```

```
`timescale 1ns/1ns

module tb_tri_state_buffer;
    reg tA;
    reg tEnable;
    wire tB;

    tri_state_buffer uut (
        .A(tA),
        .Enable(tEnable),
        .B(tB)
    );

    initial begin
        tA = 0;
        tEnable = 0;
        #10;

        tA = 1;
        tEnable = 0;
        #10;

        tA = 0;
        tEnable = 1;
        #10;

        tA = 1;
        tEnable = 1;
        #10;

        $finish;
    end
endmodule
```

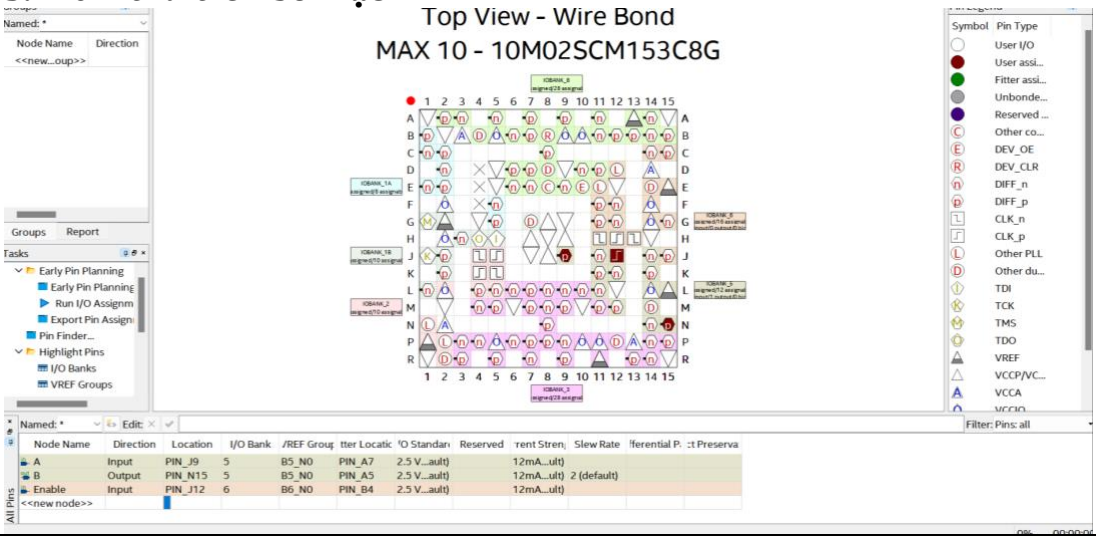
3.Mô phỏng



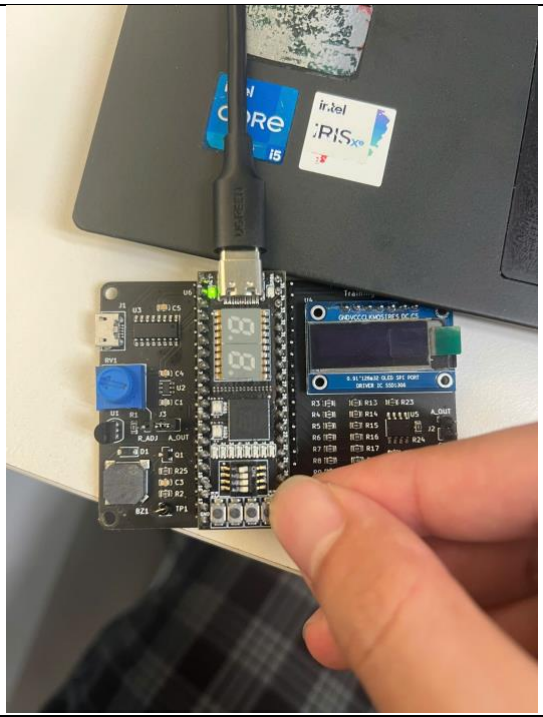
4.Chạy trên Quartus

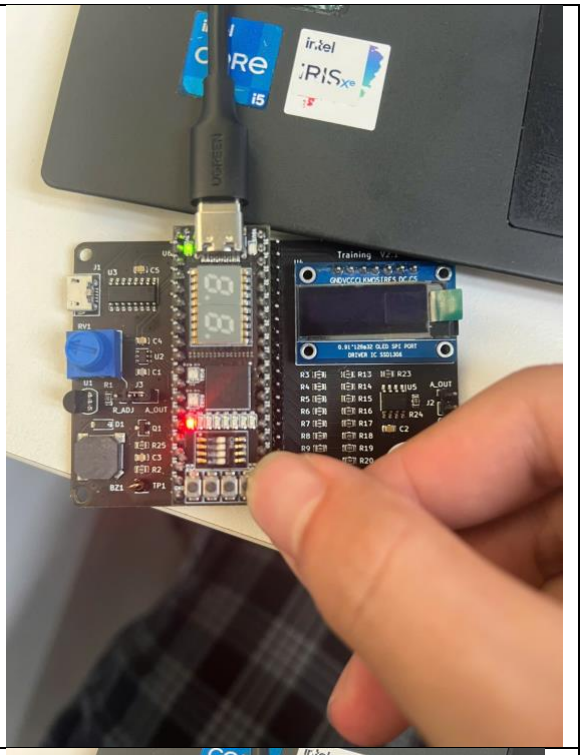


5.Kiểm thử trên bo mạch



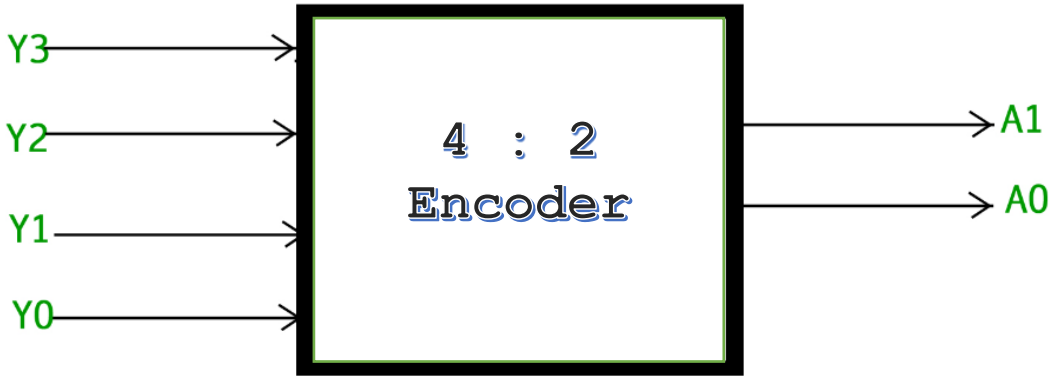
Chân	Location	Loại
A	J9	Button
B	N15	LED
Enable	J12	Switch

Trường hợp	Ảnh
A=0,Enable=0	

A=0.Enable=1	
A=1,Enable=0	

IV.Encoder 4:2

1.Đặc tả chức năng



Chức năng : Encoder 4-to-2 có chức năng chuyển đổi 4 tín hiệu đầu vào (Y0, Y1, Y2, Y3) thành 2 tín hiệu đầu ra (A1, A0). Chỉ có một đầu vào được kích hoạt tại một thời điểm, và các đầu ra sẽ biểu diễn giá trị nhị phân tương ứng của đầu vào đó. Nó được sử dụng để giảm số lượng đường dây điều khiển hoặc dữ liệu bằng cách mã hóa nhiều đầu vào thành ít đầu ra hơn.

INPUTS				OUTPUTS	
Y3	Y2	Y1	Y0	A1	A0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Từ bảng chân lí ta có
A1 = Y2 + Y3
A0 = Y1 + Y3

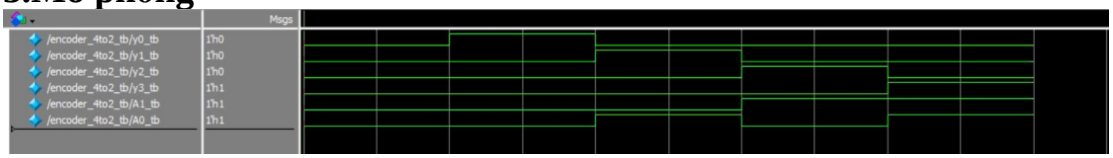
2.Mô phỏng trên Model Sim

encoder_4to2.sv	encoder_4to2_tb.sv

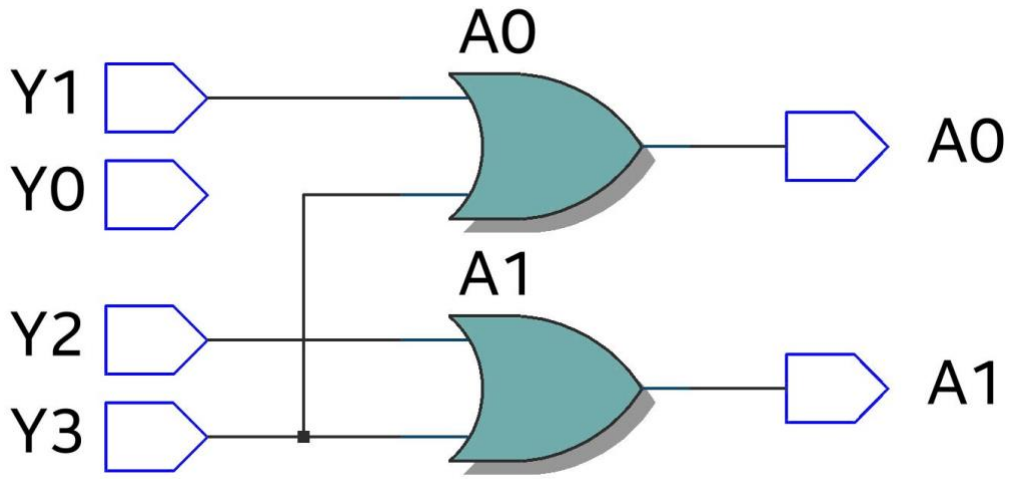
```
module encoder_4to2 (
    input wire Y0, Y1, Y2,
    output wire A1, A0
);
    assign A1 = Y2 | Y3;
    assign A0 = Y1 | Y3;
endmodule
```

```
`timescale 1ns/1ns

module encoder_4to2_tb ();
    reg y0_tb, y1_tb, y2_tb, y3_tb;
    wire A0_tb, A1_tb;

    encoder_4to2 DUT (
        .Y0(y0_tb),
        .Y1(y1_tb),
        .Y2(y2_tb),
        .Y3(y3_tb),
        .A0(A0_tb),
        .A1(A1_tb)
    );
    initial begin
        y0_tb = 0; y1_tb = 0; y2_tb = 0; y3_tb = 0;
        #20 y0_tb = 1; y1_tb = 0; y2_tb = 0; y3_tb = 0;
        #20 y0_tb = 0; y1_tb = 1; y2_tb = 0; y3_tb = 0;
        #20 y0_tb = 0; y1_tb = 0; y2_tb = 1; y3_tb = 0;
        #20 y0_tb = 0; y1_tb = 0; y2_tb = 0; y3_tb = 1;

        #20;
        $finish;
    end

endmodule
```

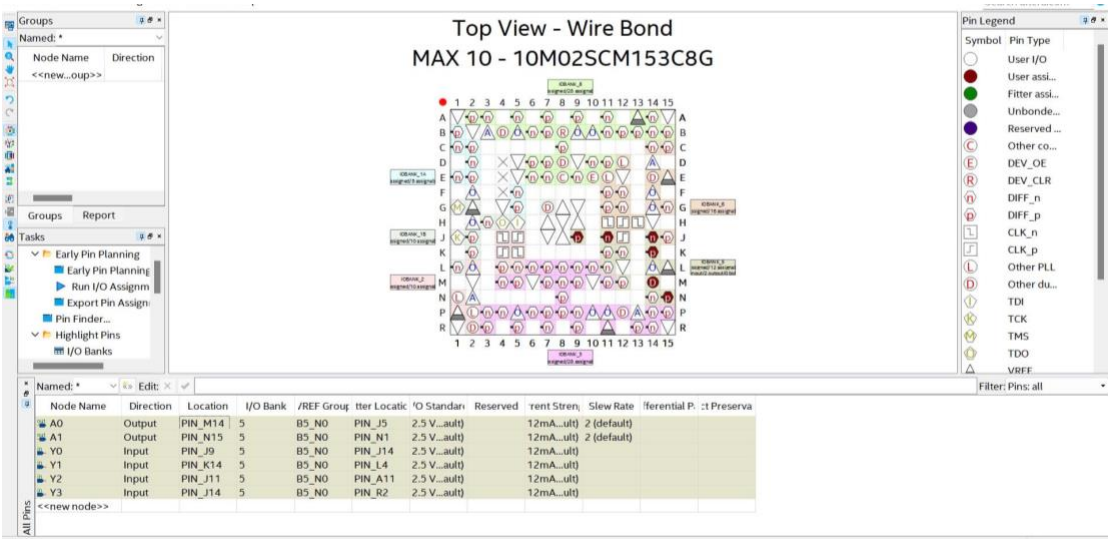
3.Mô phỏng

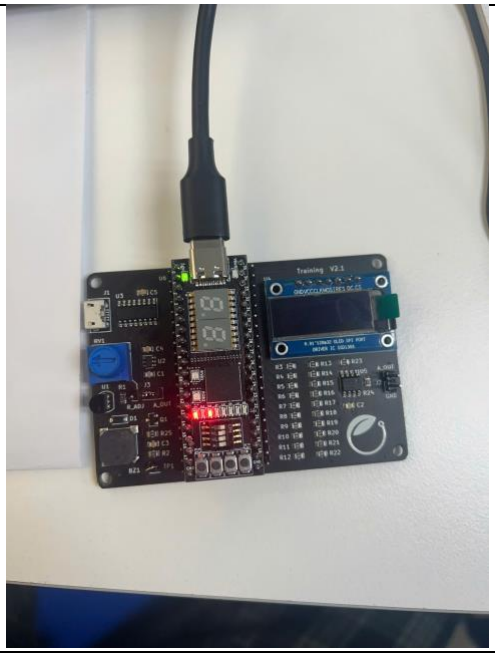


4.Chạy trên Quartus



5.Kiểm thử trên bo mạch



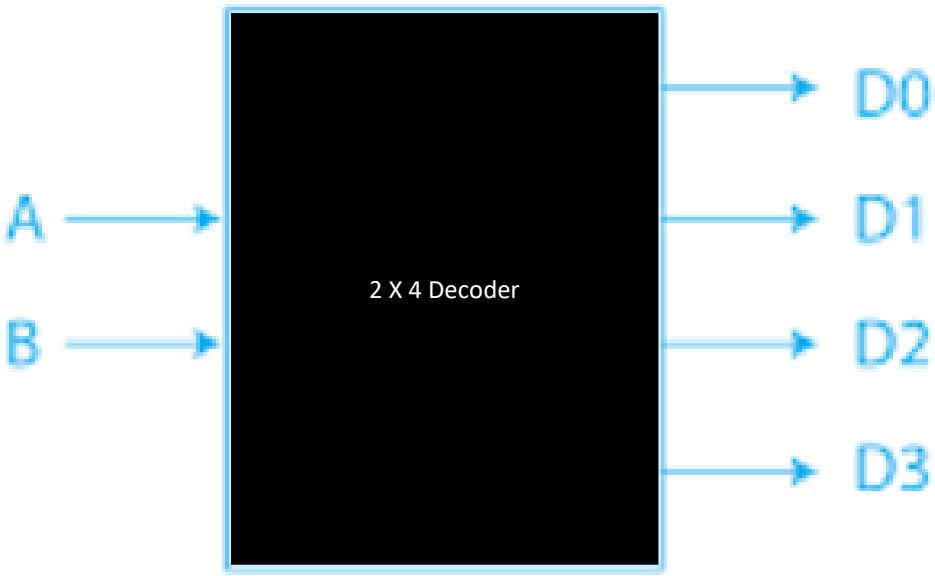
Trường hợp	Ảnh
Y0=1, Y1=0, Y2=0, Y3=0	

Y0=0, Y1=1, Y2=0, Y3=0	
Y0=0, Y1=0, Y2=1, Y3=0	

Y0=0, Y1=0, Y2=0, Y3=0	

V. Decode 2:4

1.Đặc tả chức năng



Chức năng : Bộ giải mã 2-to-4 (2x4 Decoder) có chức năng chuyển đổi 2 tín hiệu đầu vào (A, B) thành 4 tín hiệu đầu ra (D0, D1, D2, D3). Ứng với mỗi tổ hợp nhị phân của A và B, một trong bốn đầu ra sẽ được kích hoạt (bật 1), trong khi các đầu ra khác sẽ ở trạng thái 0. Bộ giải mã này thường được dùng trong các ứng dụng điều khiển đa đường hoặc chọn dữ liệu.

Inputs		Outputs			
A	B	D0	D1	D2	D3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0

1	1	0	0	0	1

2.Mô phỏng trên ModelSim

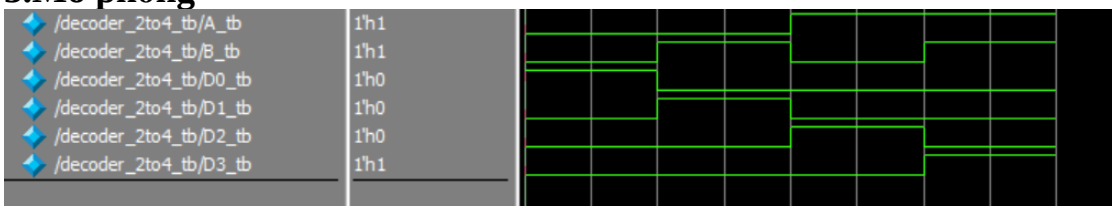
decoder_2to4.sv

```
module decoder_2to4 (
    input wire A, B,
    output wire D0, D1, D2, D3
);

    assign D0 = ~A & ~B;
    assign D1 = ~A & B;
    assign D2 = A & ~B;
    assign D3 = A & B;

endmodule
```

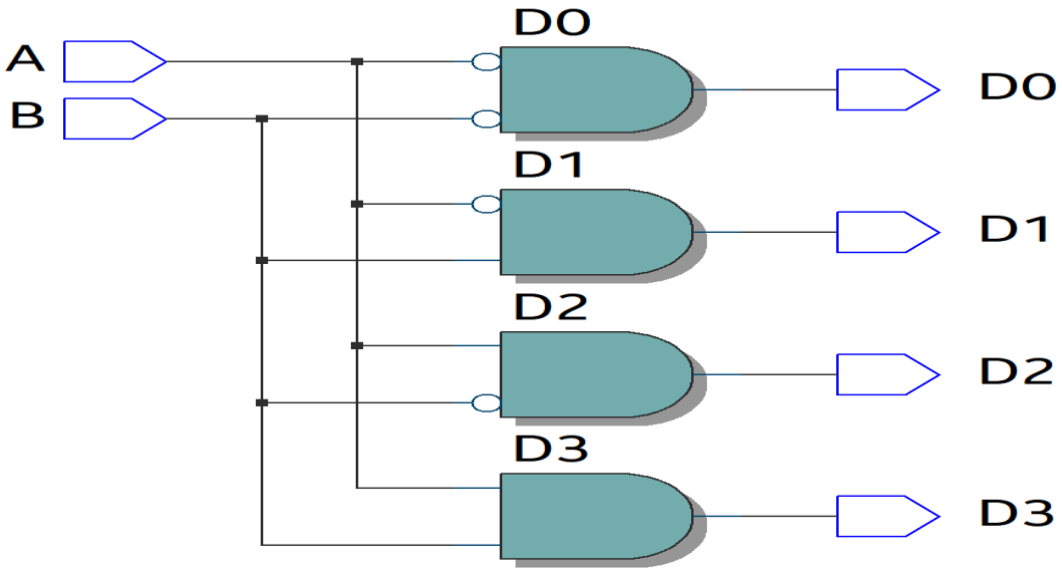
decoder_4to2_tb.sv

```
`timescale 1ns/1ns

module decoder_2to4_tb ();
    reg A_tb, B_tb;
    wire D0_tb, D1_tb, D2_tb, D3_tb;

    decoder_2to4 DUT (
        .A(A_tb),
        .B(B_tb),
        .D0(D0_tb),
        .D1(D1_tb),
        .D2(D2_tb),
        .D3(D3_tb)
    );

    initial begin
        A_tb = 0; B_tb = 0;
        #20 A_tb = 0; B_tb = 1;
        #20 A_tb = 1; B_tb = 0;
        #20 A_tb = 1; B_tb = 1;
        #20;
        $finish;
    end

endmodule
```

3.Mô phỏng

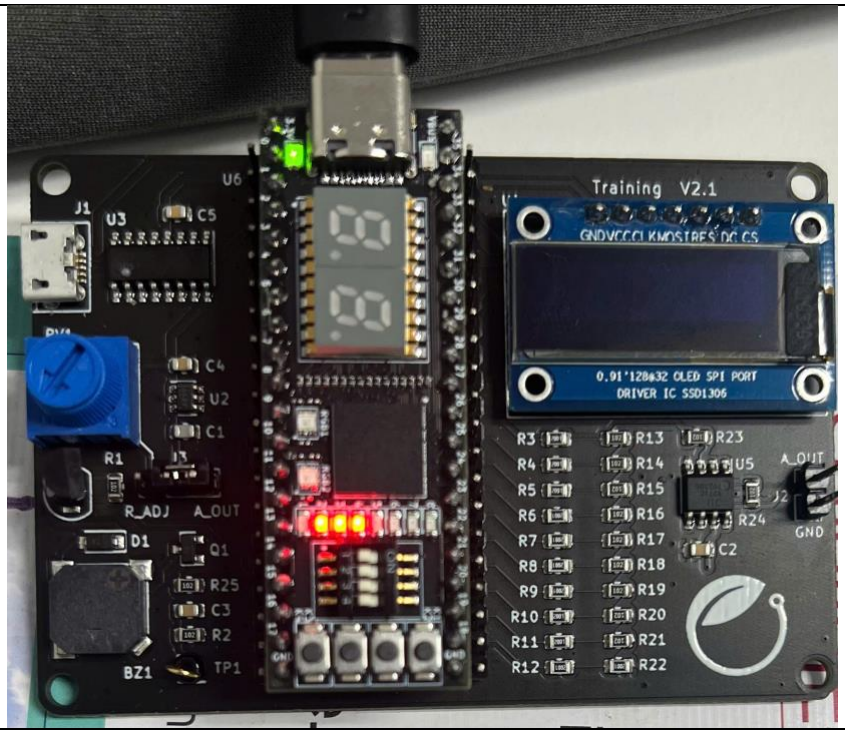


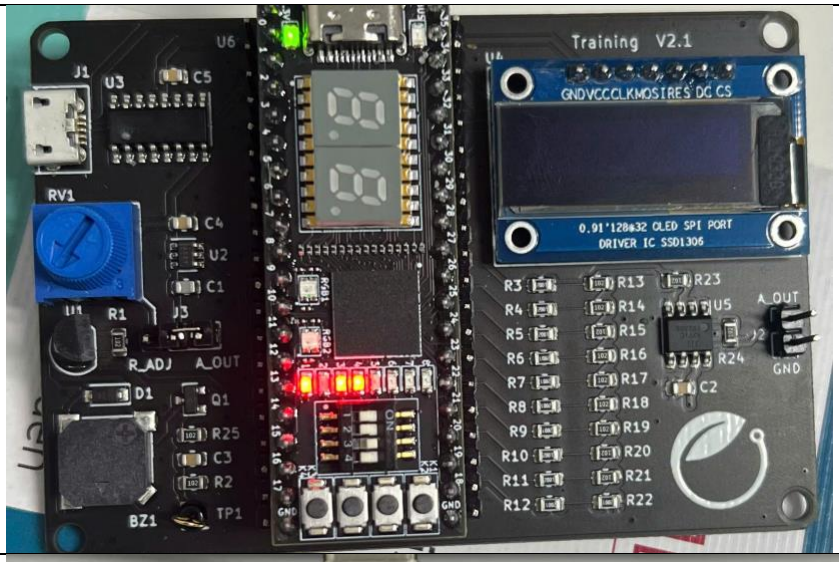
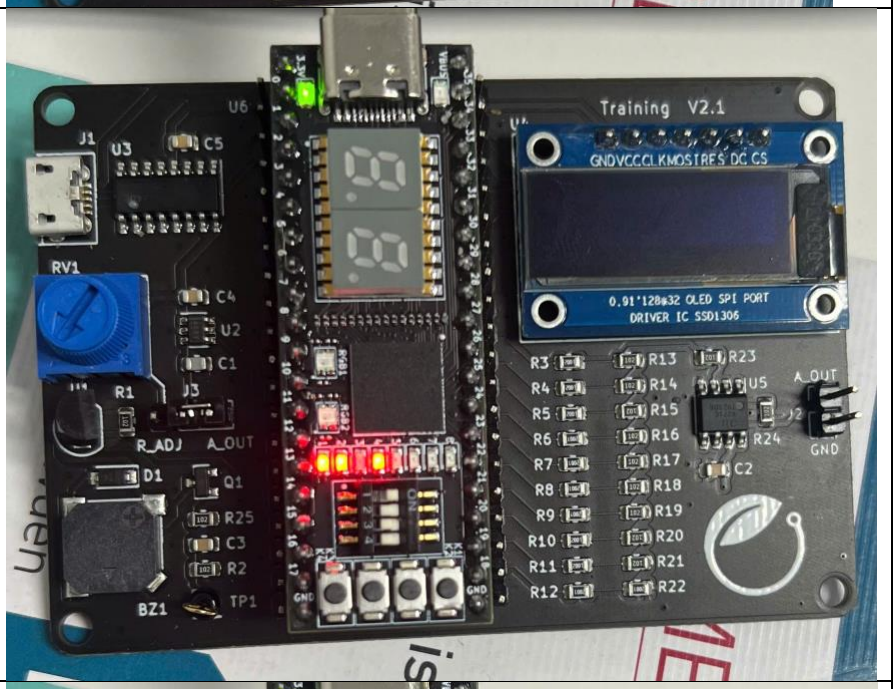
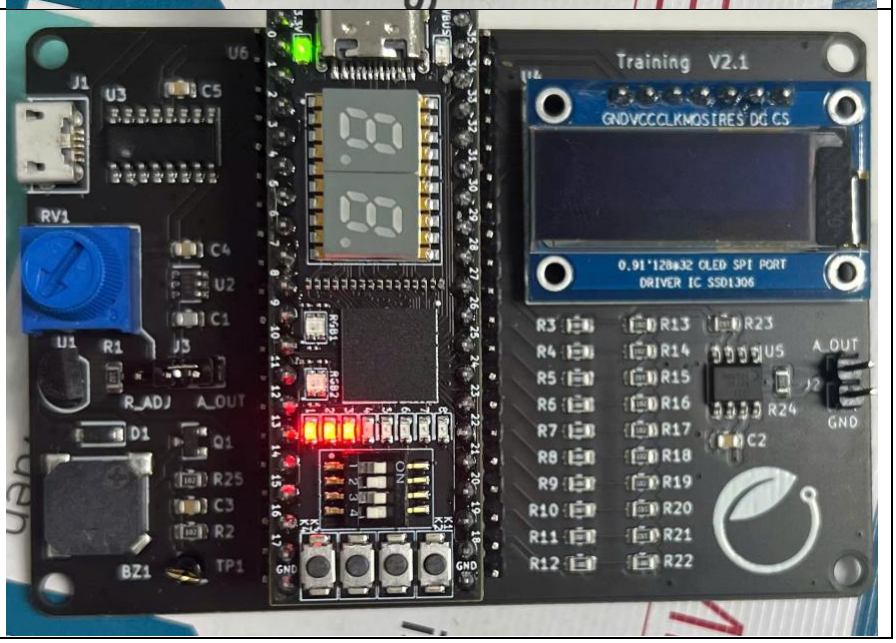
4.Chạy trên Quartus



5.Kiểm thử trên bo mạch

Node Name	Direction	Location	I/O Bank	/REF Grou	tter Locatic	'O Standar	Reserved	rent Stren	Slew Rate	fferential P	ct Preserva
A	Input	PIN_J12	6	B6_N0	PIN_J12	2.5 V		12mA...ult)			
B	Input	PIN_H12	6	B6_N0	PIN_H12	2.5 V		12mA...ult)			
D0	Output	PIN_N15	5	B5_N0	PIN_N15	2.5 V		12mA...ult)	2 (default)		
D1	Output	PIN_N14	5	B5_N0	PIN_N14	2.5 V		12mA...ult)	2 (default)		
D2	Output	PIN_M14	5	B5_N0	PIN_M14	2.5 V		12mA...ult)	2 (default)		

Trường hợp	Hình ảnh
AB =00	

AB =01	
AB =10	
AB =11	

VI.4 bit ripple carry full adder

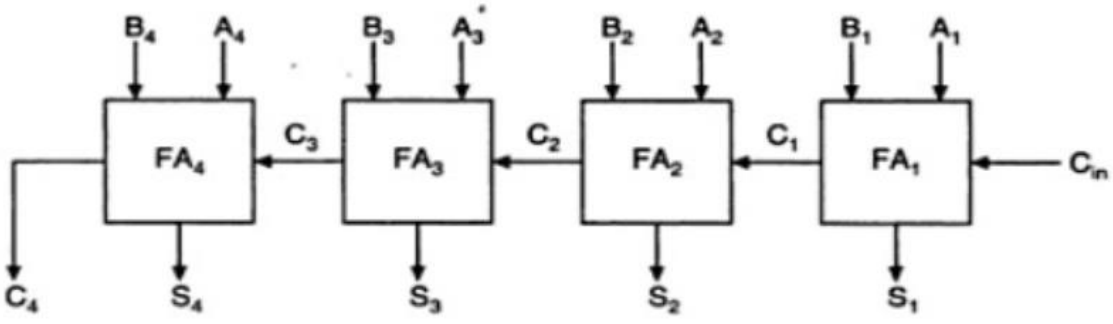
1.Đặc tả chức năng

Bộ cộng đầy đủ 1 bit

Bảng chân lí :

Inputs			Outputs	
A	B	C_in	SUM	C_out
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

Từ bộ cộng đầy đủ 1 bit ta xây dựng được bộ cộng đầy đủ 4 bit



- Đây là một mạch dùng để cộng hai số nhị phân 4-bit (A4, A3, A2, A1 và B4, B3, B2, B1) cùng với bit nhớ vào (Cin) và sinh ra kết quả tổng (S4, S3, S2, S1) cùng với bit nhớ ra cuối cùng (Cout hoặc C4).

-Bộ cộng toàn phần (Full Adder - FA): Mỗi khối FA là một bộ cộng toàn phần, có nhiệm vụ cộng hai bit đầu vào tương ứng (A[i], B[i]) và bit nhớ từ khối FA trước đó.

- Hoạt động của ripple carry:

+Bit nhớ (Cout) từ FA đầu tiên (FA1) được truyền sang FA tiếp theo (FA2), sau đó tiếp tục truyền đến FA3 và FA4.

+Quá trình này tiếp diễn từ bit có trọng số thấp nhất (LSB - A1, B1) đến bit có trọng số cao nhất (MSB - A4, B4).

2.Chạy trên ModelSim

full_adder.sv

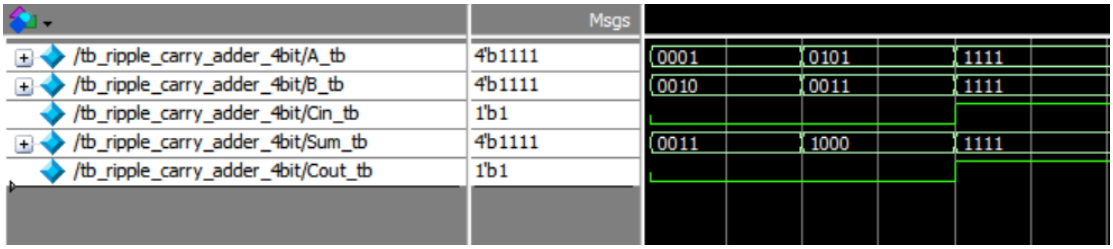
```
module full_adder (
    input wire A, B, Cin,
    output wire Sum, Cout
);

    assign Sum = A ^ B ^ Cin;
    assign Cout = (A & B) | (B & Cin) | (A & Cin);

endmodule
```

full_adder_4bit.sv

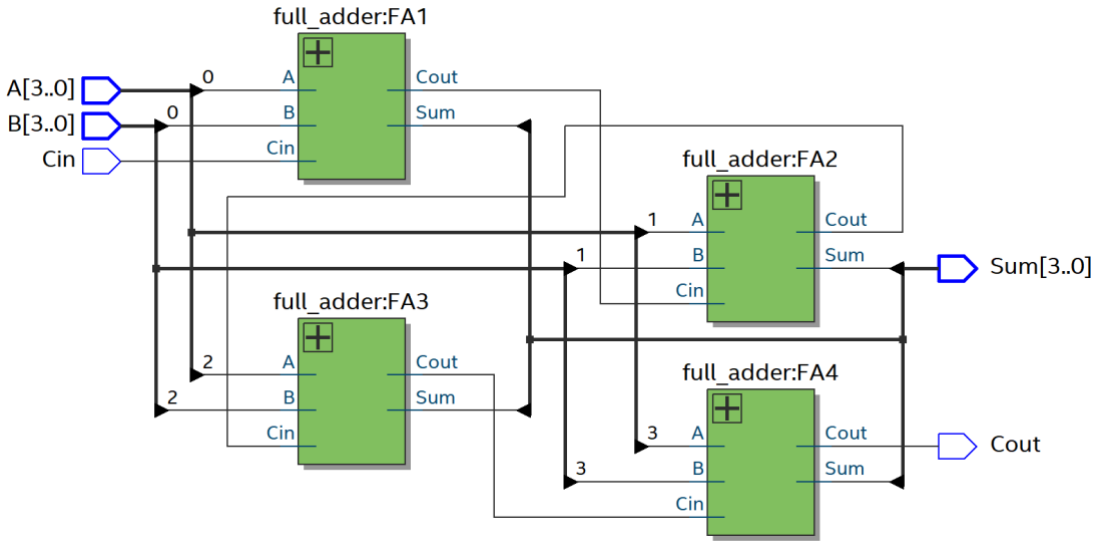
```
module ripple_carry_adder_4bit (
    input wire [3:0] A, B,
    input wire Cin,
    output wire [3:0] Sum,
    output wire Cout
);

    wire C1, C2, C3;

    full_adder FA1 (
        .A(A[0]), .B(B[0]), .Cin(Cin),
        .Sum(Sum[0]), .Cout(C1)
    );

    full_adder FA2 (
        .A(A[1]), .B(B[1]), .Cin(C1),
        .Sum(Sum[1]), .Cout(C2)
    );

    full_adder FA3 (
        .A(A[2]), .B(B[2]), .Cin(C2),
        .Sum(Sum[2]), .Cout(C3)
    );

    full_adder FA4 (
        .A(A[3]), .B(B[3]), .Cin(C3),
        .Sum(Sum[3]), .Cout(Cout)
    );

endmodule
```

full_adder_4bit_tb.sv	`timescale 1ns/1ns module tb_ripple_carry_adder_4bit (); reg [3:0] A_tb, B_tb; reg Cin_tb; wire [3:0] Sum_tb; wire Cout_tb; ripple_carry_adder_4bit DUT (.A(A_tb), .B(B_tb), .Cin(Cin_tb), .Sum(Sum_tb), .Cout(Cout_tb)); initial begin A_tb = 4'b0001; B_tb = 4'b0010; Cin_tb = 0; #20; A_tb = 4'b0101; B_tb = 4'b0011; Cin_tb = 0; #20; A_tb = 4'b1111; B_tb = 4'b1111; Cin_tb = 1; #20; $finish; end endmodule

3.Mô phỏng



4.Chạy trên Quartus


full_adder:FA1
full_adder:FA2
full_adder:FA3
full_adder:FA4
A[3..0]
B[3..0]
Cin
A
B
Cin
Cout
Sum
Sum[3..0]
Cout
0
1
2
3