BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HỒ CHÍ MINH

BÁO CÁO THỰC TẬP

Học kỳ I (2022-2023)

THIẾT KẾ HỆ THỐNG VÀ VI MẠCH TÍCH HỢP

Thứ 7(Tiết 7 – 11)

GVHD: Lê Minh Thành

Sinh viên thực hiện

Nguyễn Văn Hữu Lộc

MSSV : 20161336

TP. Thủ Đức, ngày 30 tháng 10 năm 2022

YÊU CẦU BÁO CÁO

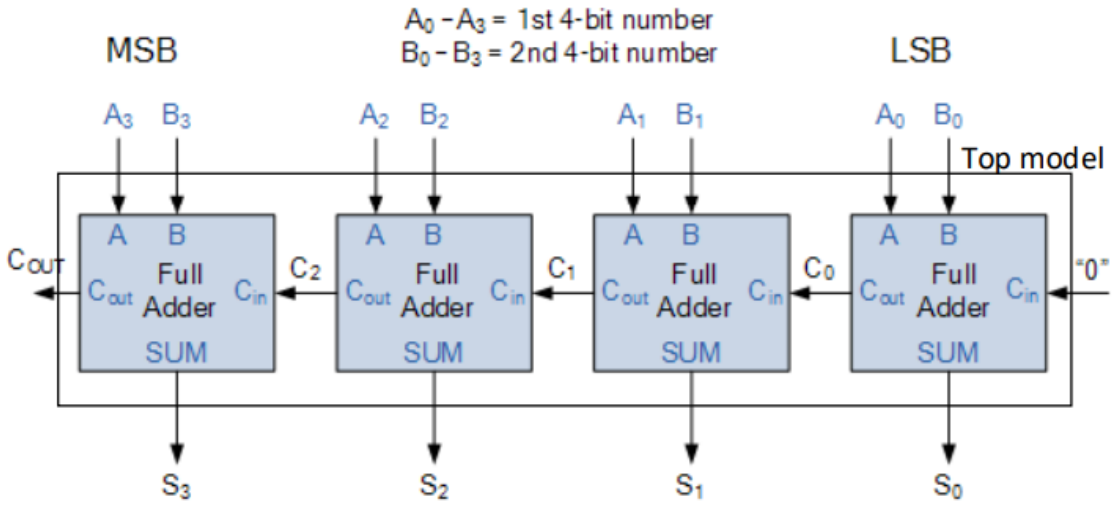
Thiết kế các đề tài bên dưới bằng ngôn ngữ mô tả phần cứng (Verilog, VHDL) sử dụng phần mềm Xilinx ISE Design Suite.

1. Thiết kế mạch cộng 4 bit theo 2 cách
   1.1. Dùng mô hình cấu trúc
   1.2. Dùng toán tử
2. Thiết kế mạch trừ 4 bit theo 3 cách
   2.1. Dùng mô hình cấu trúc
   2.2. Dùng toán tử
   2.3. Dùng phương pháp cộng bù 2
   3. Mạch giải mã 2 sang 4 ngõ ra tích cực mức cao
   4. Mạch giải mã 2 sang 4 đường ngõ ra tích cực mức thấp
   5. Mạch giải mã 2 sang 4 ngõ ra tích cực mức cao có chân enable mức thấp
   6. Mạch giải mã 2 sang 4 ngõ ra tích cực mức thấp có chân enable mức thấp
   7. Mạch giải mã 2 sang 4 ngõ ra tích cực mức cao bằng lệnh if else
   8. Mạch giải mã 3 sang 8 ngõ ra tích cực mức cao
   9. Mạch giải mã 3 sang 8 ngõ ra tích cực mức cao bằng cách ghép 2 mạch giải mã 2 sang 4
   10. Mạch giải mã 3 sang 8 ngõ ra tích cực mức cao có chân enable mức thấp
   11. Mạch giải mã 3 sang 8 có chân s cho phép ngõ ra tích cực thấp hoặc cao và chân enable mức thấp
   12. Thiết kế mạch chia xung với ngõ vào 50Mhz, 4 xung ngõ ra với tần số f, 2f, 4f, 8f, trong đó lựa chọn f ~ 1Hz.
   13. Thiết kế mạch tạo xung 1Hz.
   14. Thiết kế mạch tạo 4 xung ngõ ra với tần số lần lượt là 0.1Hz, 1Hz, 10Hz, 100Hz.
   15. Thiết kế mạch đếm đồng bộ, sử dụng phương pháp cài đặt các Flip – Flop. Xung đếm 1Hz được lấy từ mạch chia xung.
   16. Thiết kế mạch đếm lên 4 bit như baỉ 4, sử dụng phương pháp thiết kế đồng bộ.
   17. Thiết kế mạch đếm lên 8 bit, lựa chọn tần số đếm, lựa chọn đếm lên hoặc đếm xuống.
   18. Thiết kế mạch đếm lên 8 bit, lựa chọn 8 tần số đếm khác nhau, lựa chọn đếm lên hoặc đếm xuống, có tính hiệu cho phép dừng đếm (Pause), có tín hiệu đảo trạng thái ngõ ra.

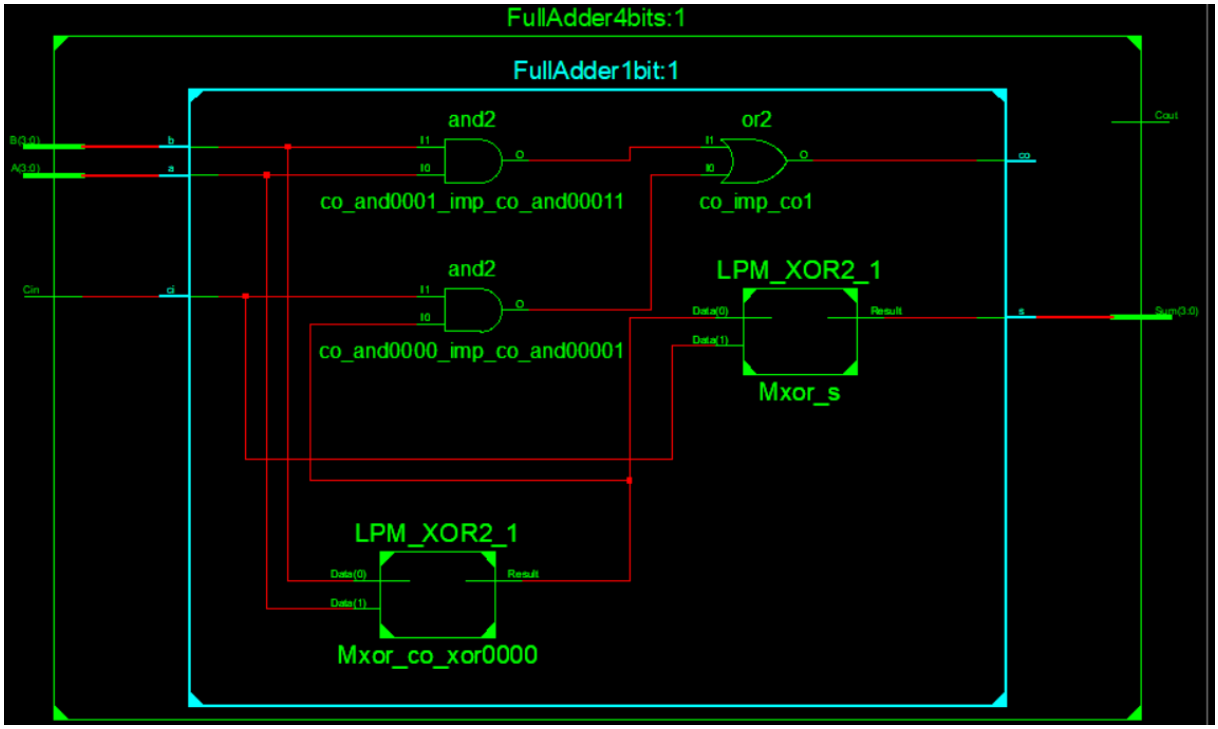
MỤC LỤC

1. Thiết kế mạch cộng 4 bit theo 2 cách

1.1. Cách 1: Mô hình cấu trúc



Hình 1.1.a. Sơ đồ khối mạch cộng toàn phần 4 bit



Hình 1.1.b. Mạch RTL

1.1.1. Code module mạch cộng toàn phần 1 bit

```
module FullAdder1bit(
  input wire   a,b,ci,
  output wire s,co);

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

endmodule
```

1.1.2. Top module

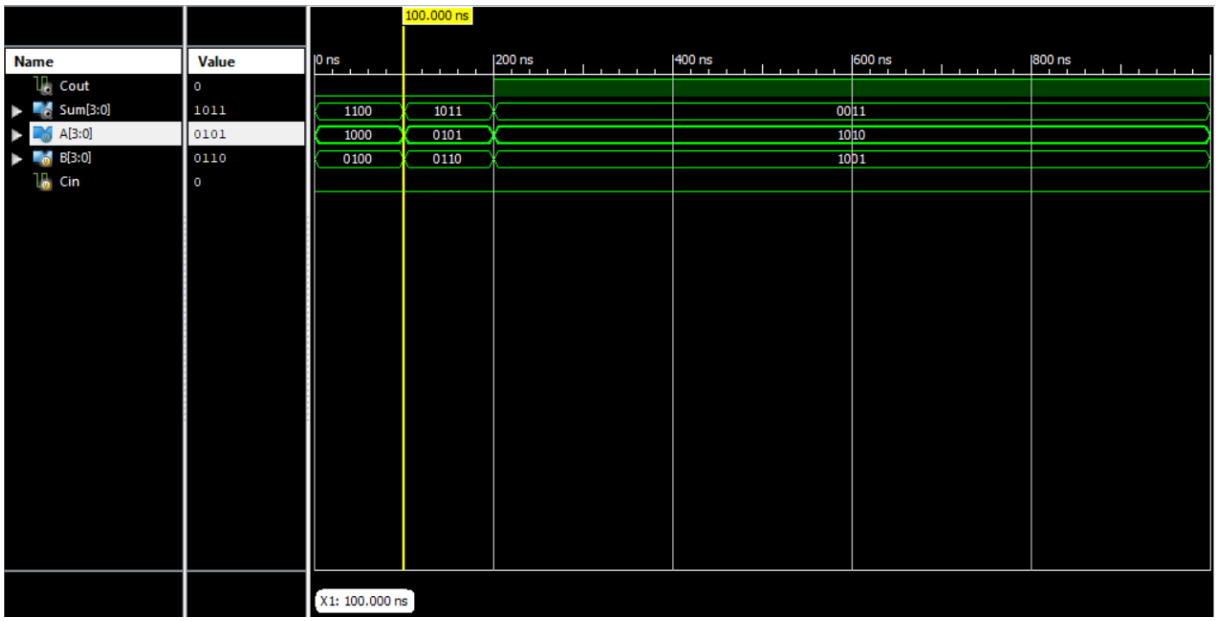
```
module FullAdder4bits(
 input wire [3:0] A,
 input wire [3:0] B,
 input wire Cin,
 output wire Cout,
 output wire [3:0] Sum
   );
        wire c1, c2, c3;
FullAdder1bit F0 (A[0], B[0], Cin ,Sum[0],c1 );
FullAdder1bit F1 (A[1], B[1], c1 ,Sum[1],c2 );
FullAdder1bit F2 (A[2], B[2], c2 ,Sum[2],c3 );
FullAdder1bit F3 (A[3], B[3], c3 ,Sum[3],Cout );

endmodule
```

1.1.3. Code test

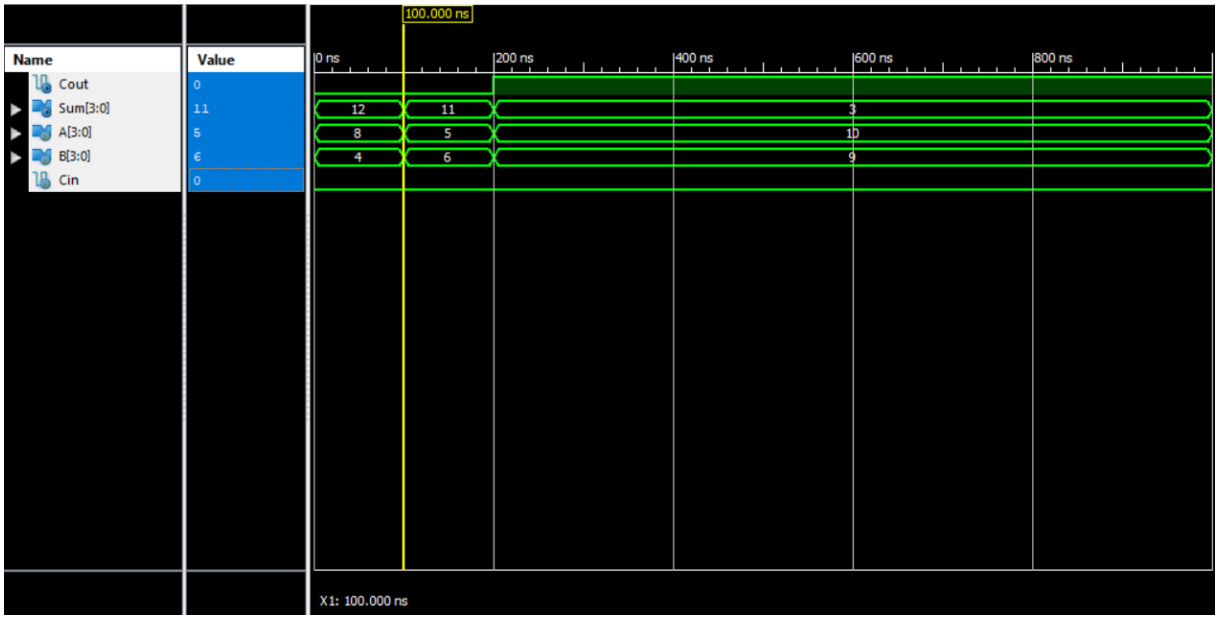
```
module test;
    reg [3:0] A;
    reg [3:0] B;
    reg Cin;
    wire Cout;
    wire [3:0] Sum;
    FullAdder4bits uut (
        .A(A),
        .B(B),
        .Cin(Cin),
        .Cout(Cout),
        .Sum(Sum)
    );

    initial begin
        A = 0;
        B = 0;
        Cin = 0;
        A=8;
        B=4;
        #100;
        A=5;
        B=6;
        #100;
        A=10;
```

```
        B=9;
        #1000;
        $stop;
    end
endmodule
```

1.1.4. Kết quả



Hình 1.1.c. Kết quả mô phỏng dưới dạng nhị phân



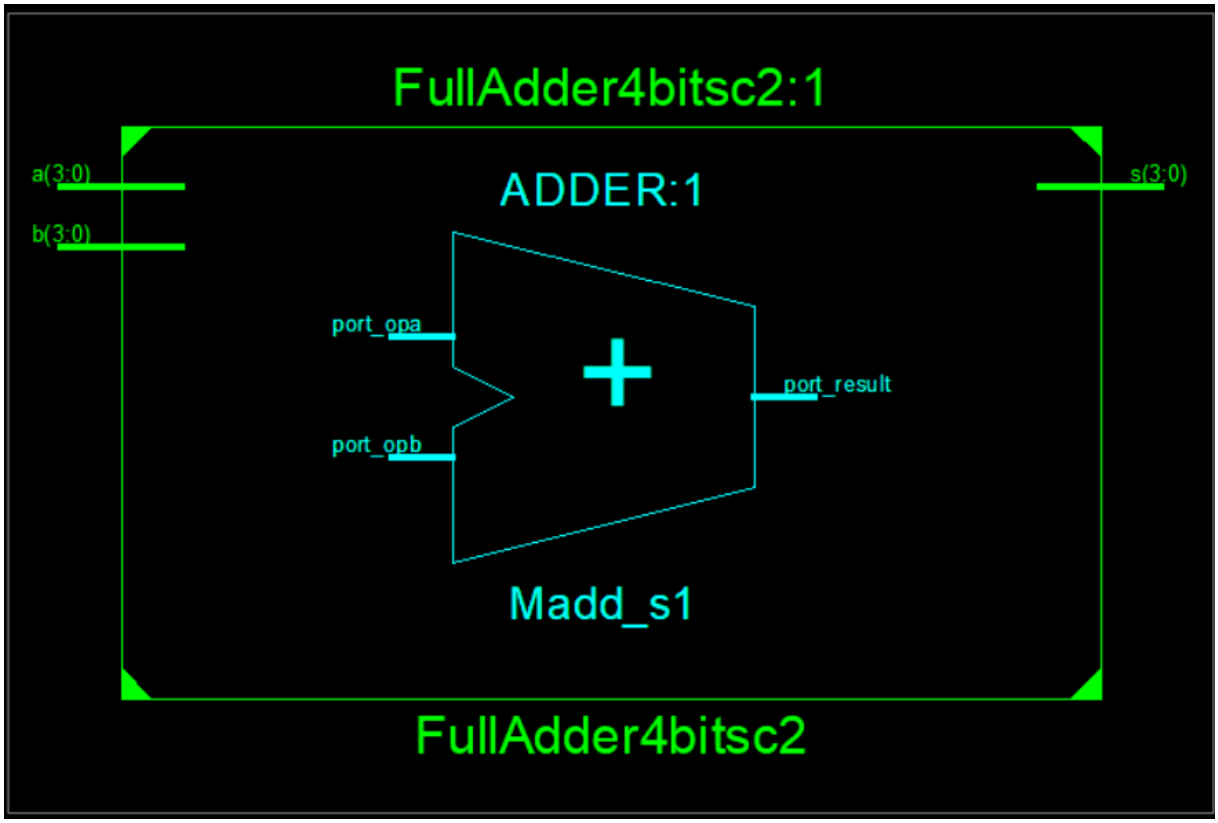
Hình 1.1.d. Kết quả mô phỏng dưới dạng thập phân

Nhận xét:

Hiển thị đúng phép tính ở các phép tính có độ lớn dưới 4 bit, với số có độ lớn trên 4 bit như 19 là 5 bit:10011 thì vẫn hiển thị 4 bit với trọng số nhỏ là 0011 hệ thập phân là 3.

1.2. Cách 2: Dùng toán tử

S = A + B



Hình 2.1.a. Mô hình RTL

1.2.1. Code

```
module FullAdder4bitsc2(
input[3:0] a,
input[3:0] b,
output  [3:0] s


);
        assign s =  {a + b};
endmodule
```

1.2.2. Code Test

```
module test;
	reg [3:0] a;
	reg [3:0] b;
	wire [3:0] s;
	FullAdder4bitsc2 uut (
		.a(a),
		.b(b),
		.s(s)
	);
	initial begin

		a = 0;b = 0;
		#100;
		a = 8;b = 4;
		#100;
		a = 5;b = 6;
		#100;
		a = 10;b = 9;
		#100;
		$stop;
	end
endmodule
```

1.2.3. Kết quả

Hình 1.2.b. Kết quả hiển thị dạng nhị phân

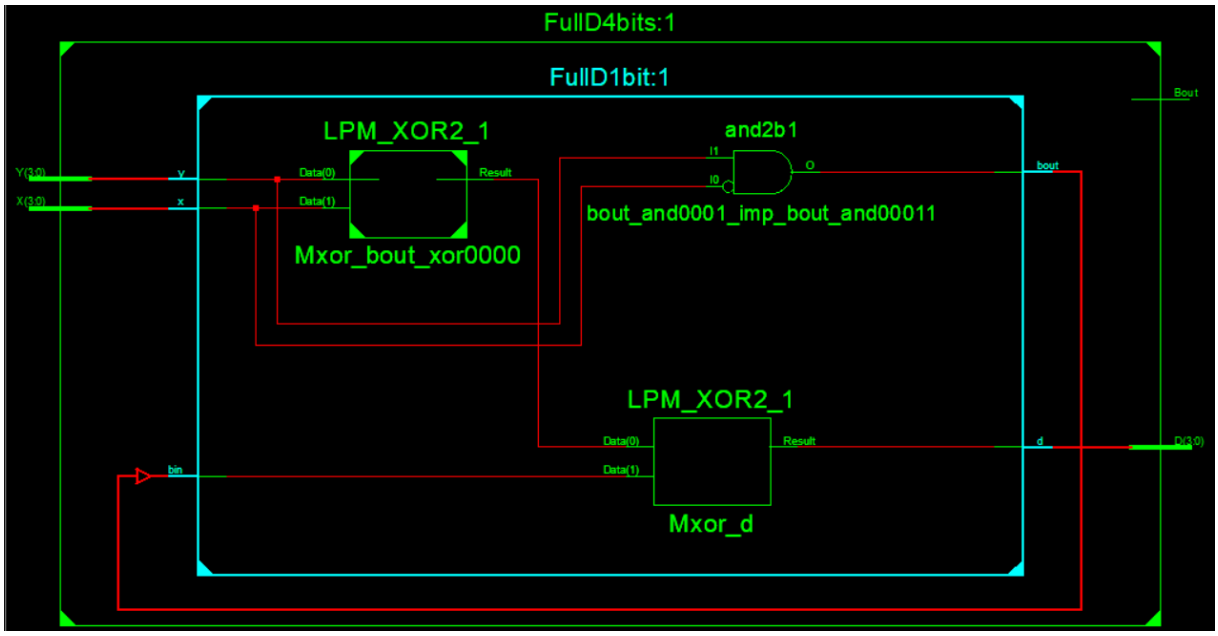
Hình 1.2.c. Kết quả hiển thị ở số thập phân

Nhận xét:

Hiển thị đúng phép tính ở các phép tính có độ lớn dưới 4 bit, với số có độ lớn trên 4 bit như 19 là 5 bit:10011 thì vẫn hiển thị 4 bit với trọng số nhỏ là 0011 hệ thập phân là 3.

2. Mạch trừ 4 bit theo 3 cách

2.1. Cách 1 : Mô hình cấu trúc



Hình 2.1.a. Sơ đồ RTL

2.1.1. Code module trừ toàn phần 1 bit

```
module FullD1bit(
input x,y,bin,
output d,bout
   );
        assign d= x^y^bin;
        assign bout=((~(x^y))&bin) | ((~x)&y);
endmodule
```

2.1.2. Top module

```
module FullD4bits(
 input wire [3:0] X,
 input wire [3:0] Y,
 output wire Bout,
 output wire [3:0] D
```

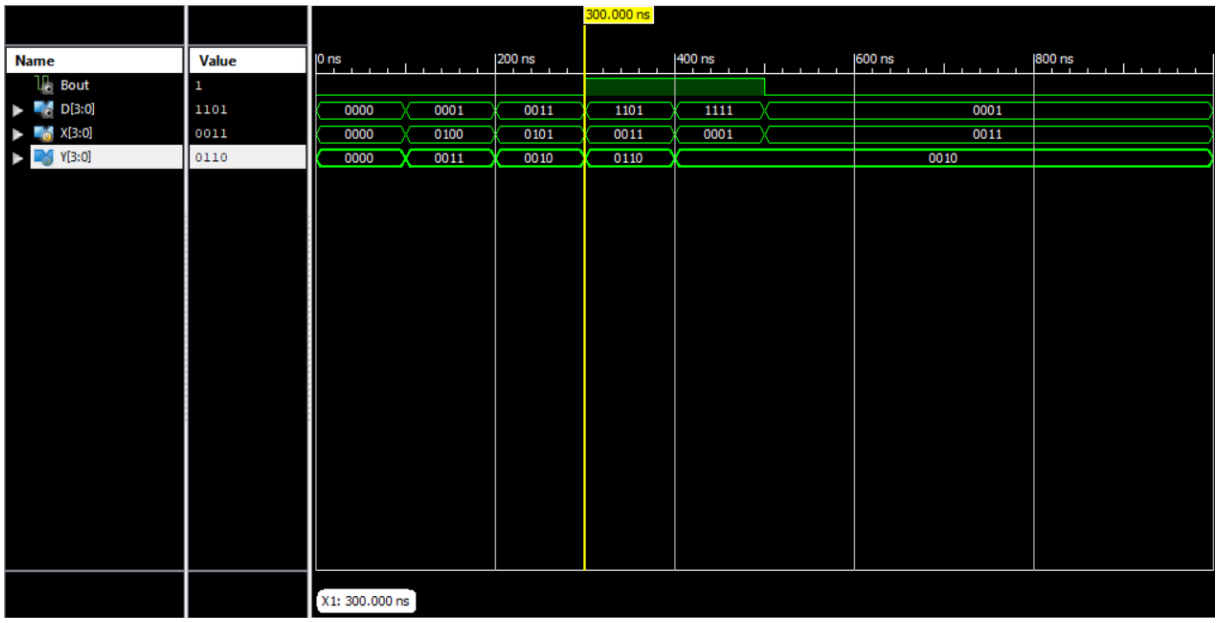
```
    );
        wire b1, b2, b3;
    FullD1bit FD0 (X[0], Y[0], 1'b0 ,D[0],b1 );// 1'b0 nghia là đưa vào bit 0
    FullD1bit FD1 (X[1], Y[1], b1 ,D[1],b2 );
    FullD1bit FD2 (X[2], Y[2], b2 ,D[2],b3 );
    FullD1bit FD3 (X[3], Y[3], b3 ,D[3],Bout );
Endmodule
```

2.1.3. Code test

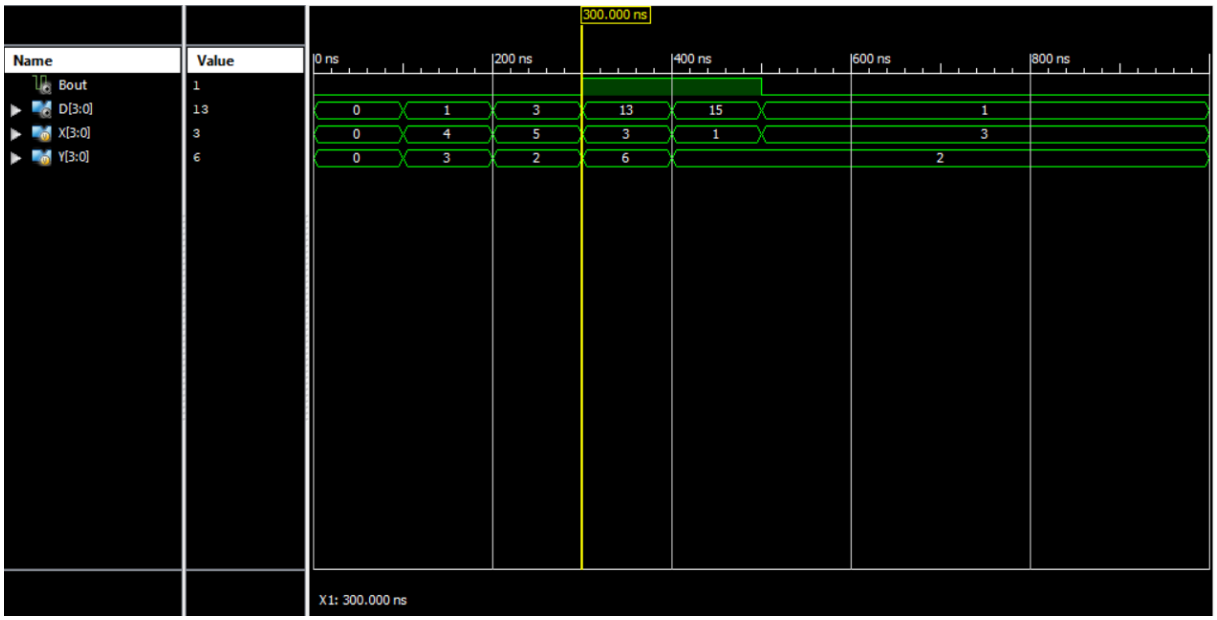
```
module TEST;
        reg [3:0] X;
        reg [3:0] Y;
        wire Bout;
        wire [3:0] D;
        FullD4bits uut (
                .X(X),
                .Y(Y),
                .Bout(Bout),
                .D(D)
        );
        initial begin
                X = 0;Y = 0;
                #100;
                X = 4;Y = 3;
                #100;
                X = 5;Y = 2;
                #100;
```

```
        X = 3;Y = 6;

        #100;

        X = 17;Y = 2;

        #100;

        X = 19;Y = 18;

        #1000;

        $stop;

    end

endmodule
```

2.1.4. Kết quả



Hình 2.1.b. Kết quả hiện thị dạng nhị phân



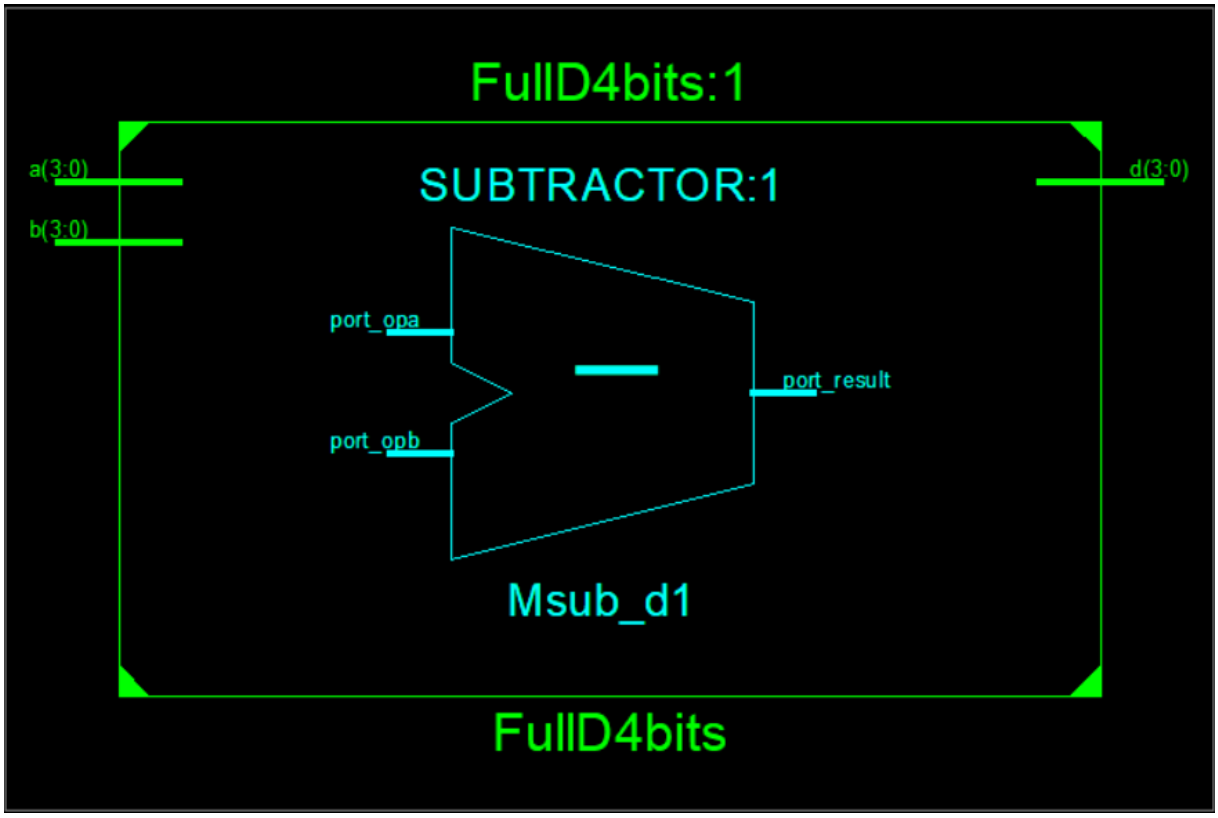
Hình 2.1.c. Kết quả hiện thị dạng thập phân

Nhận xét:

- Cho kết quả chính xác các phép tính có độ lớn từ 4 bit trở xuống và giá trị thập phân lớn hơn hoặc bằng 0,
- Cho kết quả sai ghi kết quả phép trừ bé hơn 0,
- Các thành phần trong phép tính có độ lớn hơn 4 bit thì kết quả cho ra chỉ hiển thị 4 bit ở trọng số nhỏ.

2.2. Cách 2 : Dùng toán tử

D = A - B



Hình 2.2.a. Mô hình RTL

2.2.1. Code

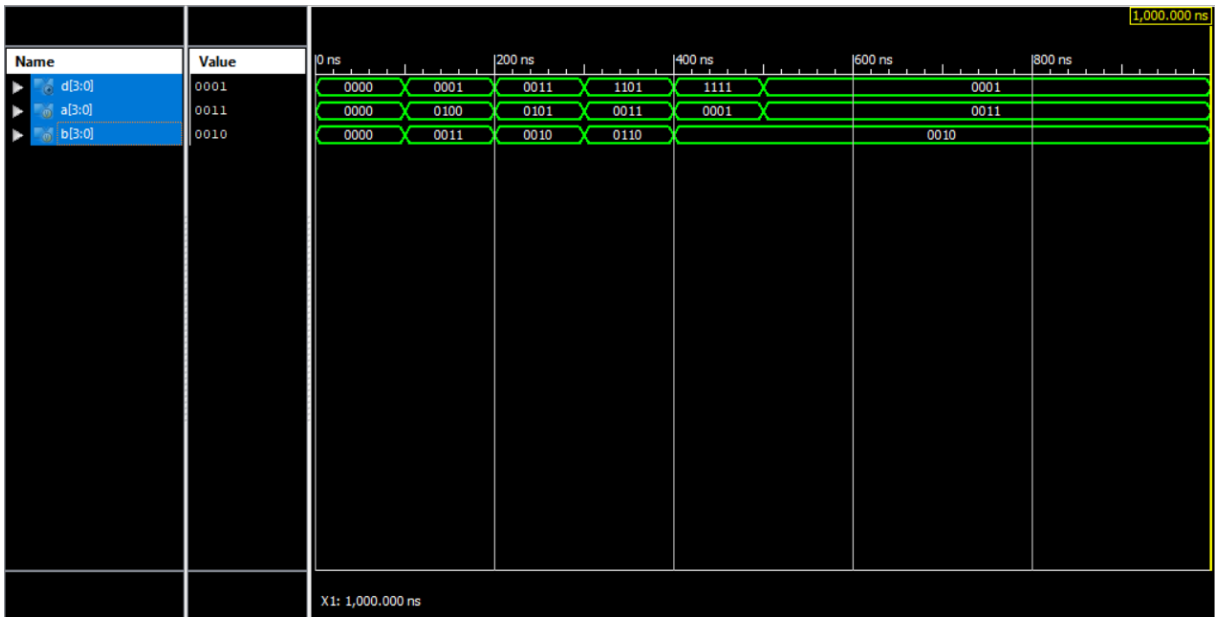
```
module FullD4bits(
input [3:0] a,b,
output [3:0] d
    );
assign d=a-b;
endmodule
```

2.2.2. Code test

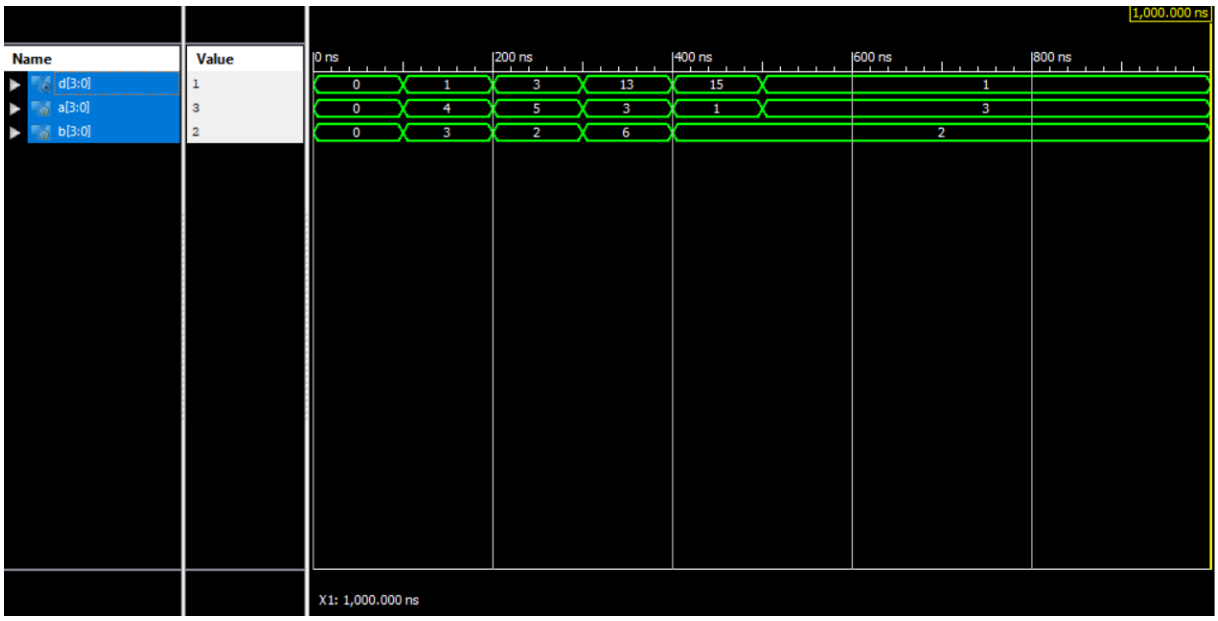
```
module t;
        reg [3:0] a;
        reg [3:0] b;
```

```
        wire [3:0] d;
        FullD4bits uut (
                .a(a),
                .b(b),
                .d(d)
        );
        initial begin
                a = 0;b = 0;
                #100;
                a=4;b=3;
                #100;
                a=5;b=2;
                #100;
                a=3;b=6;
                #100;
                a=17;b=2;
                #100;
                a=19;b=18;
                #100;
        end
endmodule
```

2.2.3. Kết quả



Hình 2.2.b. Kết quả hiển thị dạng nhị phân



Hình 2.2.c. Kết quả hiển thị dạng thập phân

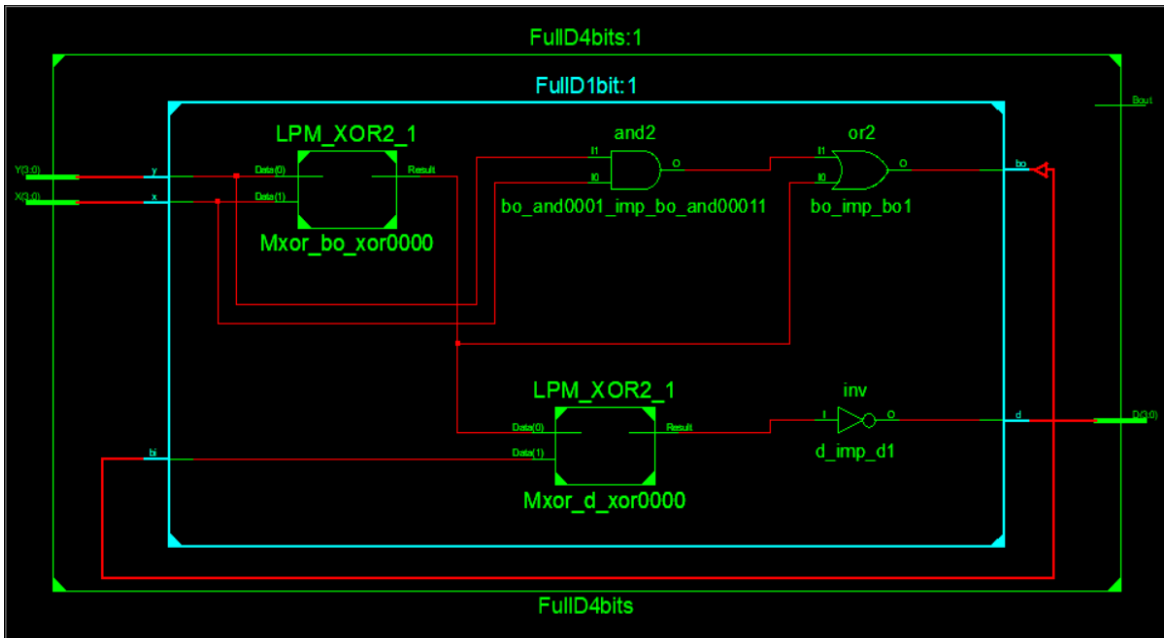
Nhận xét:

- Cho kết quả chính xác các phép tính có độ lớn từ 4 bit trở xuống và giá trị thập phân lớn hơn hoặc bằng 0,
- Cho kết quả sai ghi kết quả phép trừ bé hơn 0,
- Các thành phần trong phép tính có độ lớn hơn 4 bit thì kết quả cho ra chỉ hiển thị 4 bit ở trọng số nhỏ.
-

2.3. Cách 3: dùng phương pháp cộng bù 2

Mô tả lý thuyết mạch cộng bù 2

D = A – B = A + (~B+1)



Hình 2.3.a. Mô hình RTL

2.3.1. Code module Cộng bù 1 bit

```
module FullD1bit(
  input wire x,y,bi,
  output wire d,bo);
assign d = (x^((~y)^bi)) ;
assign bo = ((x^(~y))&bi) | (x&(~y)) ;
endmodule
```

2.3.2. Top module

```
module FullD4bits(
 input wire [3:0] X,
 input wire [3:0] Y,
 output wire Bout,
 output wire [3:0] D
```

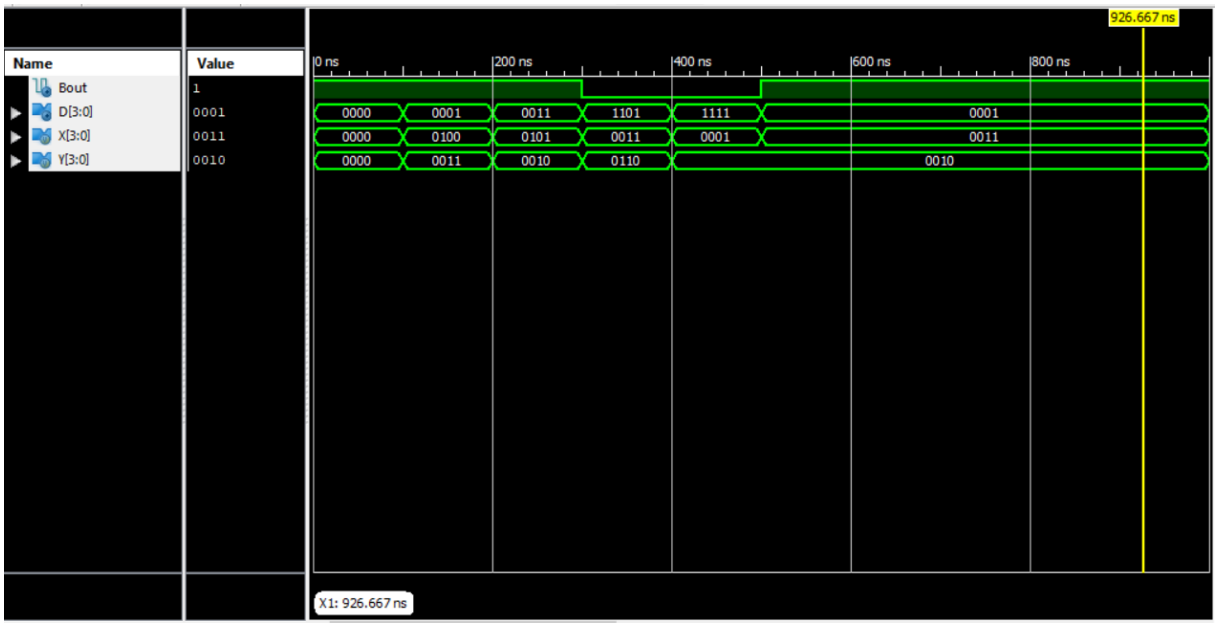
```
    );
        wire b1, b2, b3;
    FullD1bit FD0 (X[0], Y[0], 1'b1 ,D[0],b1 );
    FullD1bit FD1 (X[1], Y[1], b1 ,D[1],b2 );
    FullD1bit FD2 (X[2], Y[2], b2 ,D[2],b3 );
    FullD1bit FD3 (X[3], Y[3], b3 ,D[3],Bout );
endmodule
```

2.3.3. Code test

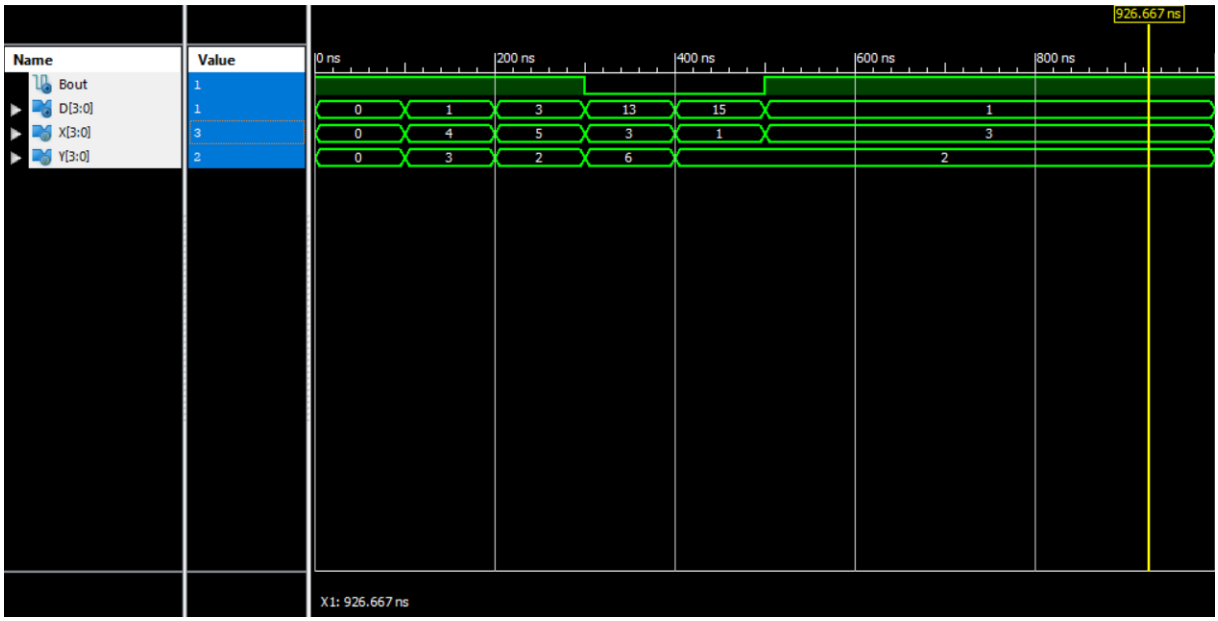
```
module test;
        reg [3:0] X;
        reg [3:0] Y;
        wire Bout;
        wire [3:0] D;
        FullD4bits uut (
                .X(X),
                .Y(Y),
                .Bout(Bout),
                .D(D)
        );
        initial begin
                X = 0;Y = 0;
                #100;
                X = 4;Y = 3;
                #100;
                X = 5;Y = 2;
                #100;
```

```
        X = 3; Y = 6;
        #100;
        X = 17; Y = 2;
        #100;
        X = 19; Y = 18;
        #1000;
        $stop;
    end
endmodule
```

2.3.4. Kết quả



Hình 2.3.b. Kết quả dạng nhị phân

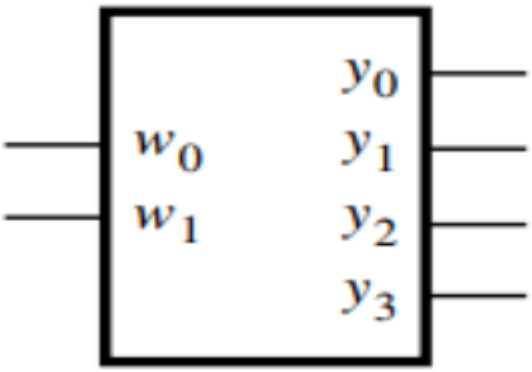


Hình 2.3.c. Hiển thị ở dạng số thập phân

Nhận xét

- Hoạt động tốt ở các phép tính có độ lớn từ 4 bit trở xuống và giá trị thập phân lớn hơn hoặc bằng 0,
- Cho kết quả sai ghi kết quả phép trừ bé hơn 0,
- Các thành phần trong phép tính có độ lớn hơn 4 bit thì kết quả cho ra chỉ hiển thị 4 bit ở trọng số nhỏ.

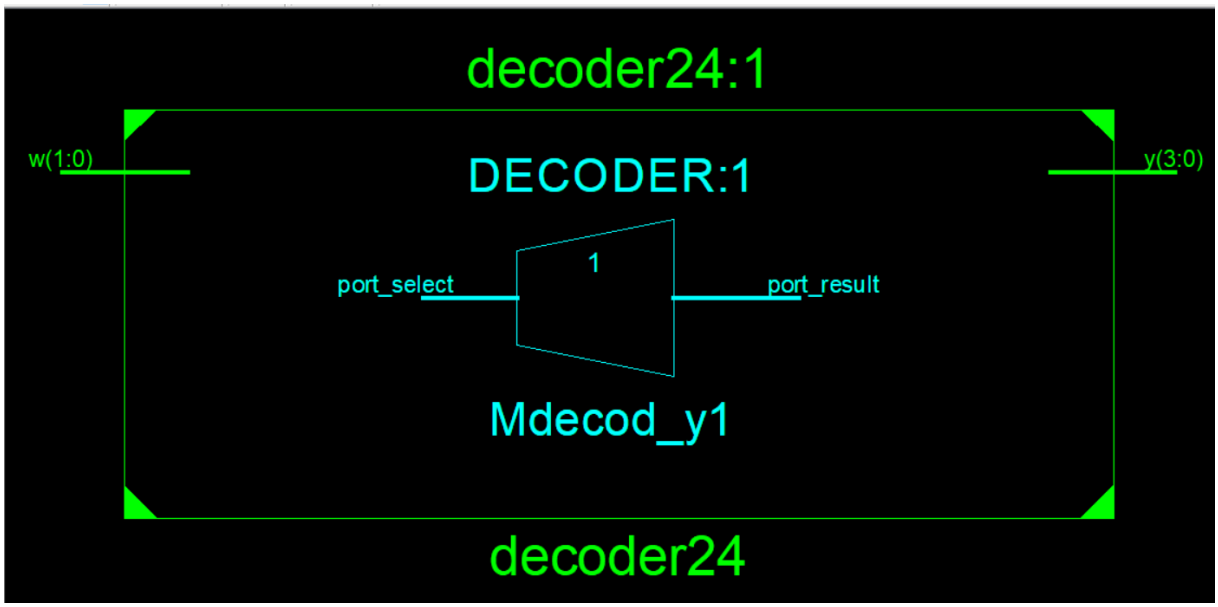
3. Mạch giải mã 2 sang 4 ngõ ra tích cực mức cao



Hình 3.1. Sơ đồ khối

w1	w0	y3	y2	y1	y0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Hình 3.2. Bảng trạng thái



Hình 3.3. Mạch RTL

Code

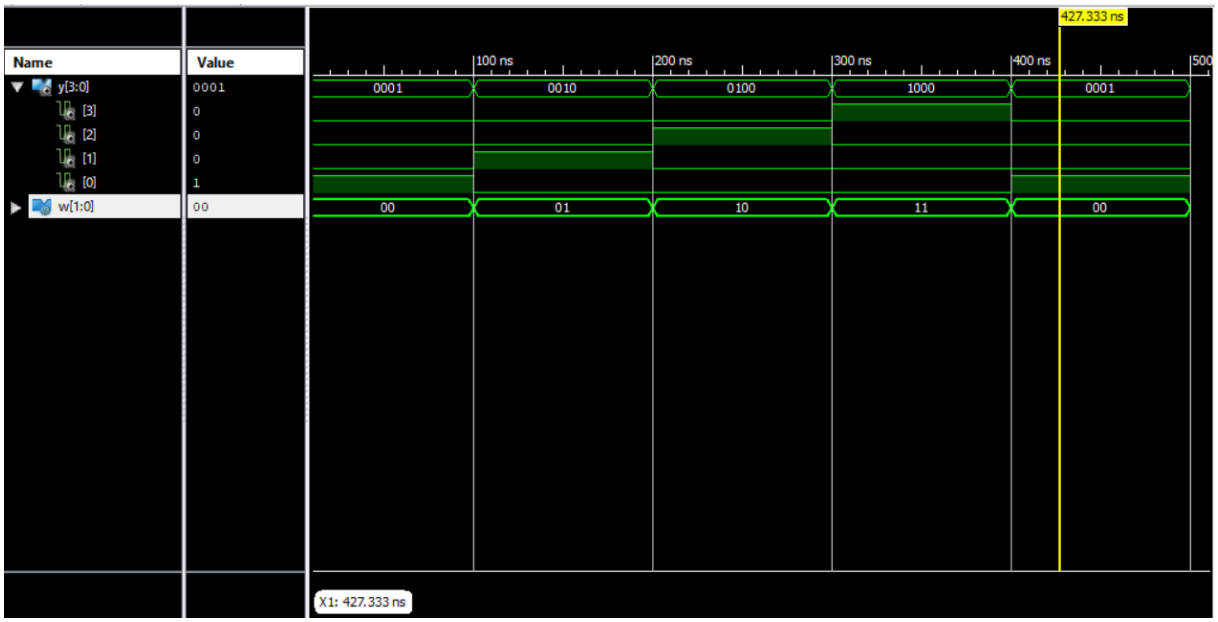
```
module decoder24(
 input wire [1:0] w,
 output reg [3:0] y
   );
 always @(w)
  case (w)
   0: y = 4'b0001;
   1: y = 4'b0010;
   2: y = 4'b0100;
   3: y = 4'b1000;
  endcase
endmodule
```

Code test

```
module test;
        reg [1:0] w;
        wire [3:0] y;
        decoder24 uut (
                .w(w),
                .y(y)
        );
        initial begin
                w = 0;
                #100;
                w = 1;
                #100;
                w = 2;
                #100;
                w = 3;
```

```
        #100;
        w = 4;
        #100;
        $stop;
    end
endmodule
```

Kết quả

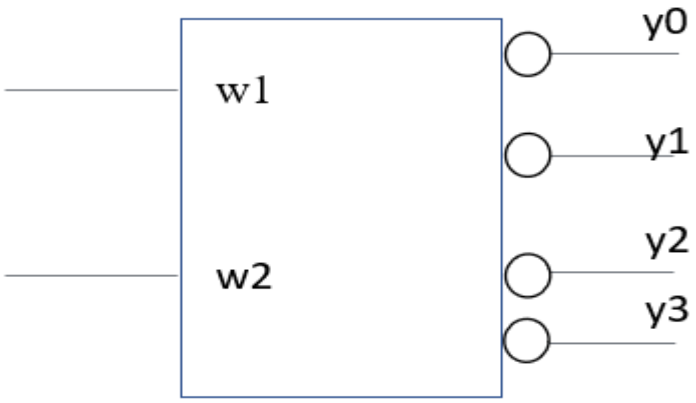


Hình 3.4. Biểu đồ dạng sóng

<u>Nhận xét:</u>

Từ 0-400 ns mạch hoạt động đúng bảng trạng thái khi nhập đúng đầu vào của 2 ngõ vào, khi nhập dữ liệu lớn hơn 2 bit thì đầu vào cũng chỉ hiện 2 bit trọng số nhỏ và đưa ra kết quả tương ứng (thời điểm hơn 400ns).

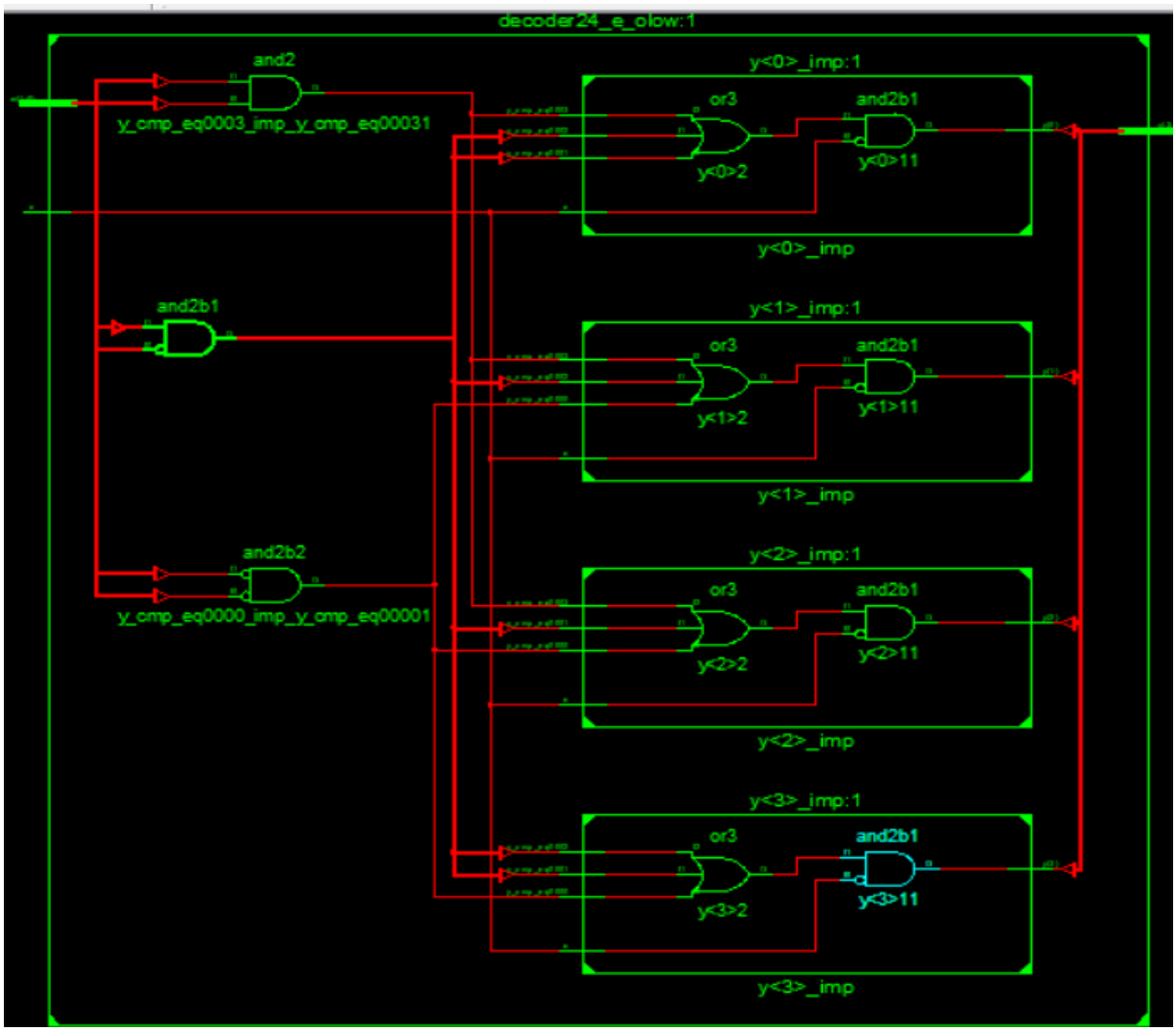
4. Mạch giải mã 2 sang 4 đường ngõ ra tích cực mức thấp.



Hình 4.1. Sơ đồ khối

w1	w0	y3	y2	y1	y0
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

Hình 4.2. Bảng trạng thái



Hình 4.3. Mạch RTL

Code

```
module decoder24_e_olow(
 input wire [1:0] w,
 input wire e,
 output reg [3:0] y
    );
always @(w , e)
 if ( e==0)
  case (w)
    0: y = 4'b1110;
```

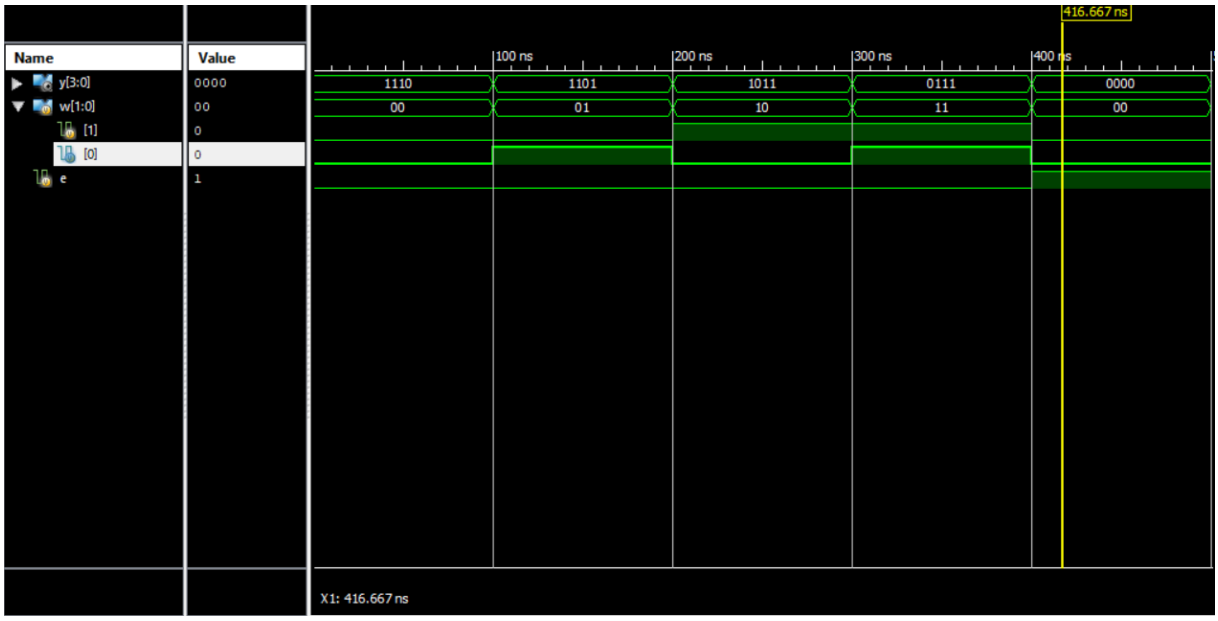
```
    1: y = 4'b1101;
    2: y = 4'b1011;
    3: y = 4'b0111;
  endcase
 else y=15;
endmodule
```

Code test

```
module text;
        reg [1:0] w;
        reg e;
        wire [3:0] y;
        decoder24_e_olow uut (
                .w(w),
                .e(e),
                .y(y)
        );
        initial begin
                w = 0;
                e = 0;
                #100;
                w=1;
                e=0;
                #100;
                w=2;
                e=0;
                #100;
```

```
        w=3;
        e=0;
        #100;
        w=0;
        e=1;
        #100;
        $stop;
    end
endmodule
```

Kết quả

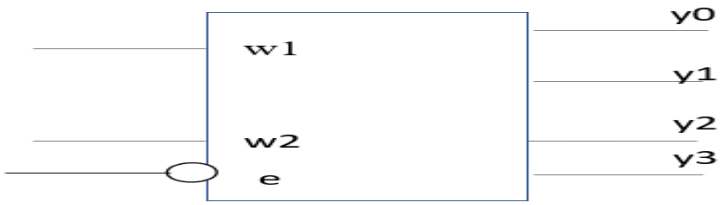


Hình 4.4. Biểu đồ dạng sóng

Nhận xét:

Từ 0-400 ns mạch hoạt động đúng bảng trạng thái khi nhập đúng đầu vào của 2 ngõ vào, khi nhập dữ liệu lớn hơn 2 bit thì đầu vào cũng chỉ hiện 2 bit trọng số nhỏ và đưa ra kết quả tương ứng (thời điểm hơn 400ns).

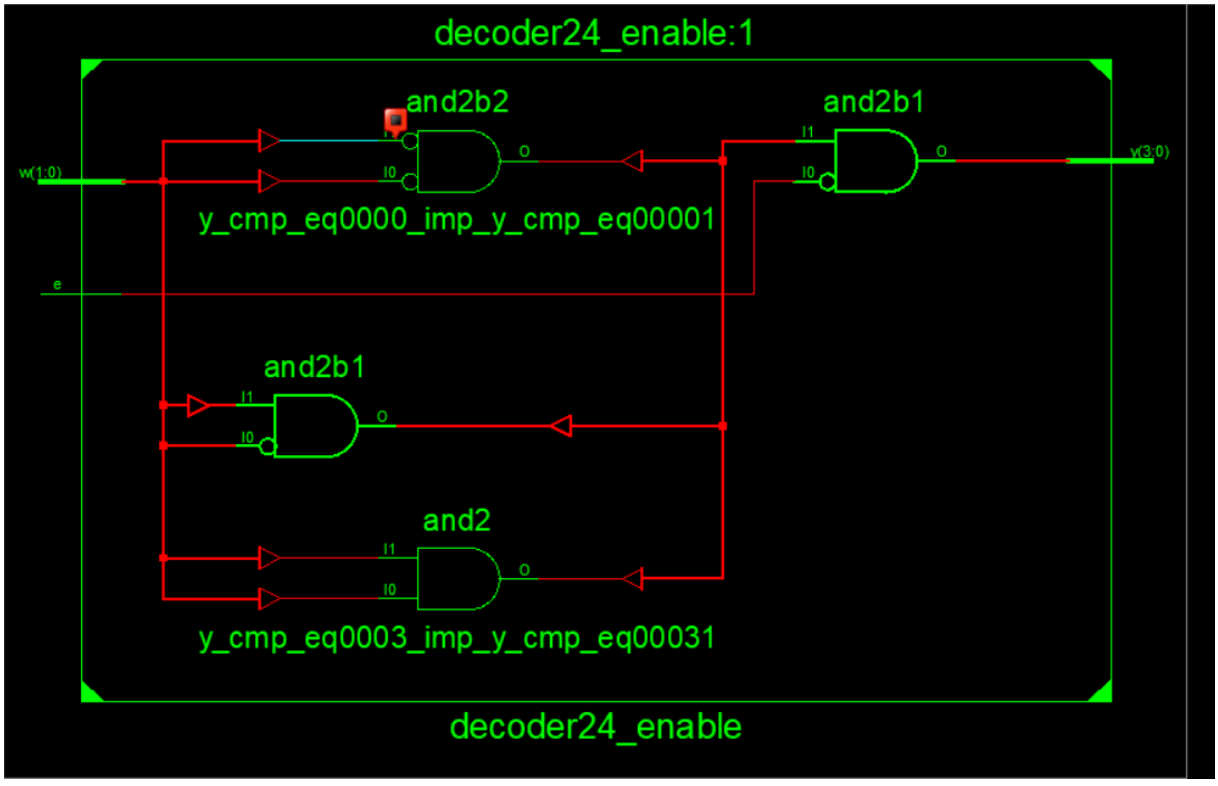
5. Mạch giải mã 2 sang 4 ngõ ra tích cực mức cao có chân enable mức thấp



Hình 5.1. Sơ đồ khối

e	w1	w0	y3	y2	y1	y0
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

Hình 5.2. Bảng giá trị



Hình 5.3. Mạch RTL

Code

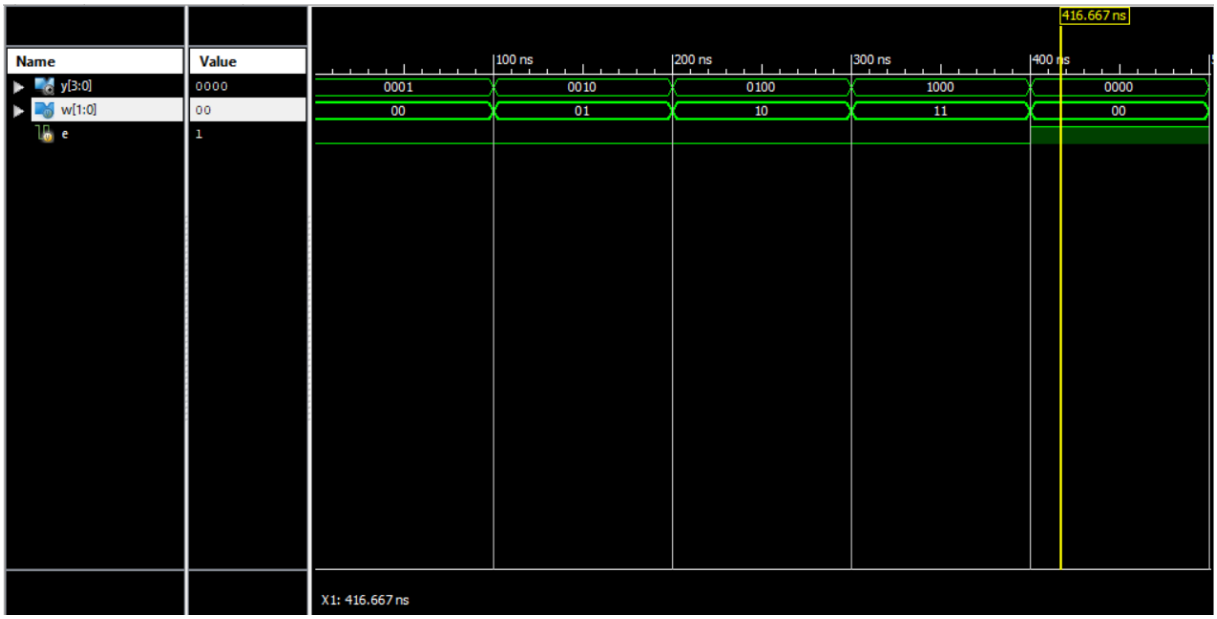
```
module decoder24_enable(
 input wire [1:0] w,
 input wire e,
 output reg [3:0] y
   );
 always @(w , e)
 if ( e==0)
  case (w)
    0: y = 4'b0001;
    1: y = 4'b0010;
    2: y = 4'b0100;
    3: y = 4'b1000;
  endcase
 else y=0;
endmodule
```

Code test

```
module test;
        reg [1:0] w;
        reg e;
        wire [3:0] y;
        decoder24_enable uut (
                .w(w),
                .e(e),
                .y(y)
        );
```

```
        initial begin
                w = 0;
                e = 0;
                #100;
                w=1;
                e=0;
                #100;
                w=2;
                e=0;
                #100;
                w=3;
                e=0;
                #100;
                w=0;
                e=1;
                #100;
                $stop;
        end
endmodule
```

Kết quả

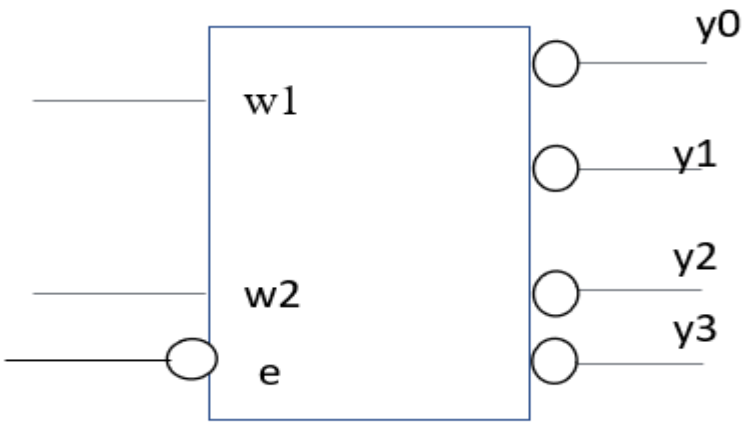


Hình 5.4. Kết quả dạng sóng

Nhận xét:

Từ 0-400 ns mạch hoạt động đúng bảng trạng thái khi nhập đúng đầu vào của 2 ngõ vào và chân e ở mức 0, khi nhập dữ liệu e=1 thì trả lại kết quả 0000 (thời điểm hơn 400ns).

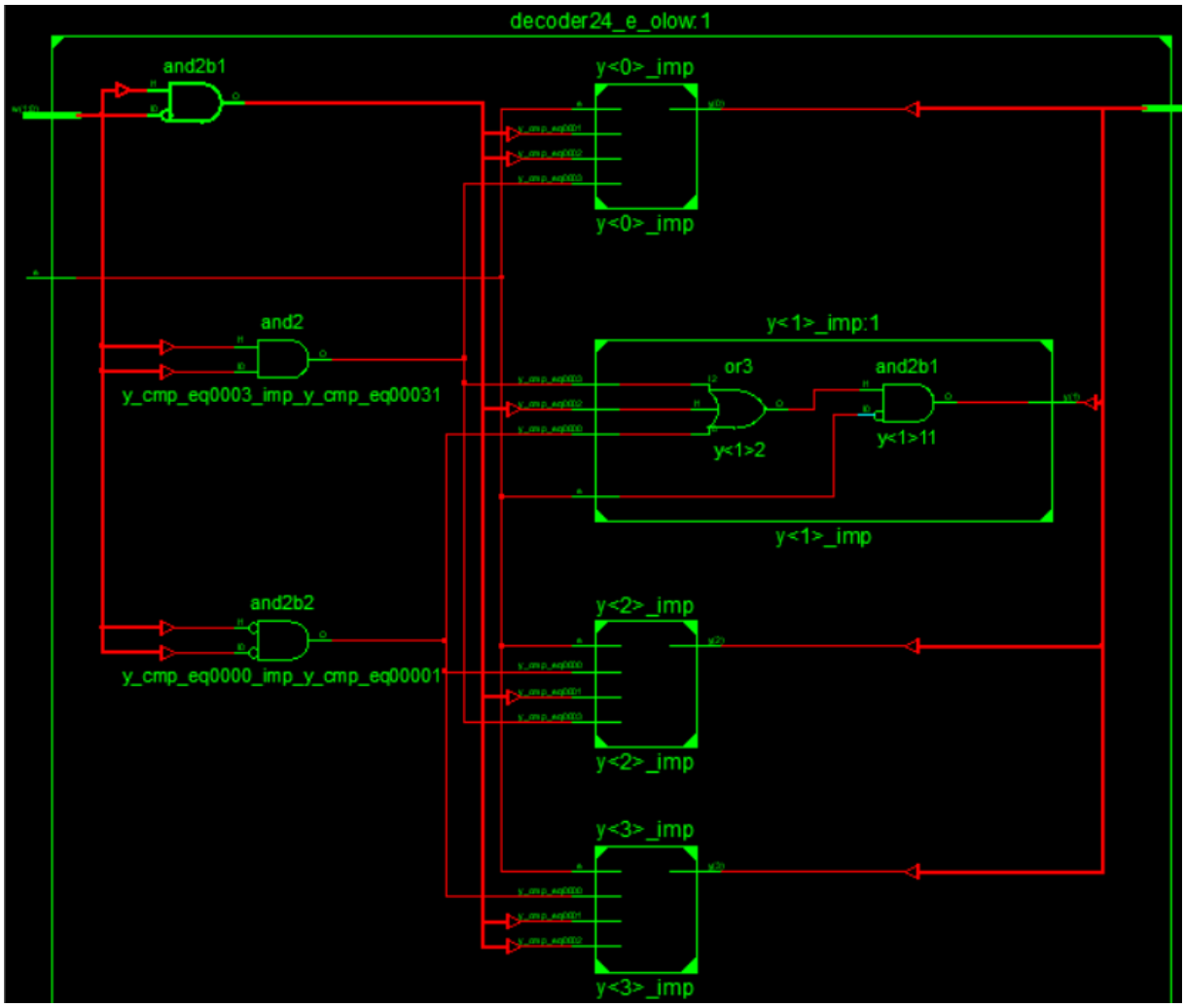
6. Mạch giải mã 2 sang 4 ngõ ra tích cực mức thấp có chân enable mức thấp.



Hình 6.1. Sơ đồ khối

e	w1	w0	y3	y2	y1	y0
1	x	x	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

Hình 6.2 Bảng giá trị



Hình 6.3. Mạch RTL

Code

```
module decoder24_e_olow(
 input wire [1:0] w,
 input wire e,
 output reg [3:0] y
```

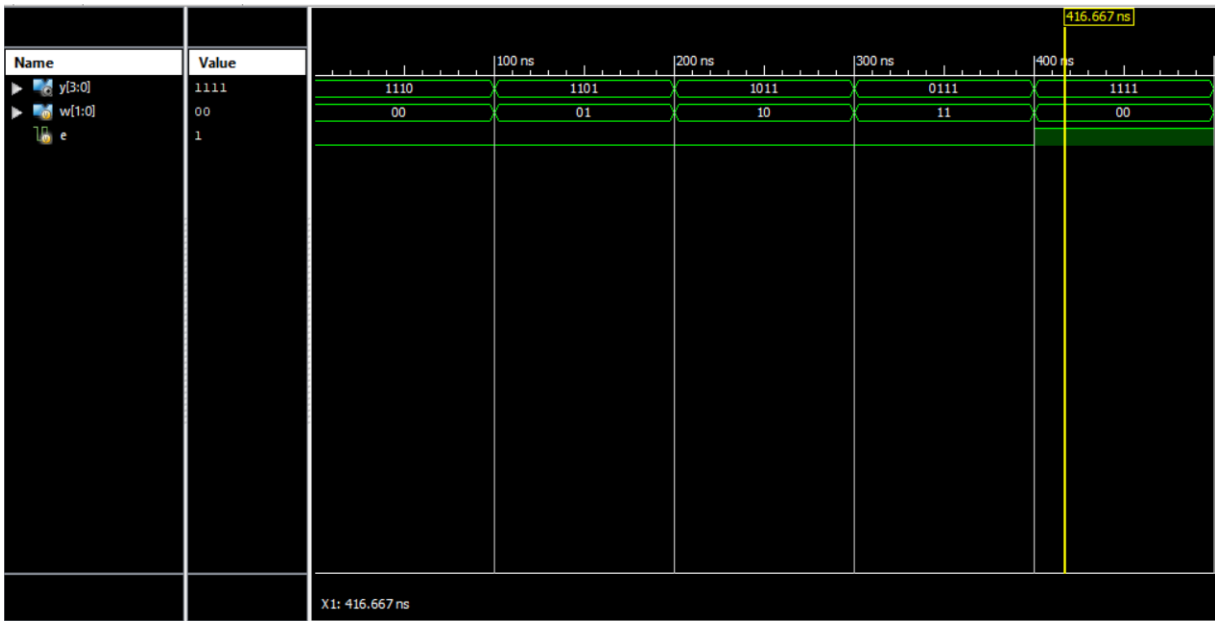
```
    );
always @(w , e)
 if ( e==0)
  case (w)
    0: y = 4'b1110;
    1: y = 4'b1101;
    2: y = 4'b1011;
    3: y = 4'b0111;
  endcase
 else y=15;
endmodule
```

Code test

```
module text;
        reg [1:0] w;
        reg e;
        wire [3:0] y;
        decoder24_e_olow uut (
                .w(w),
                .e(e),
                .y(y)
        );
        initial begin
                w = 0;
                e = 0;
                #100;
                w=1;
```

```
        e=0;
        #100;
        w=2;
        e=0;
        #100;
        w=3;
        e=0;
        #100;
        w=0;
        e=1;
        #100;
        $stop;
    end
endmodule
```

Kết quả

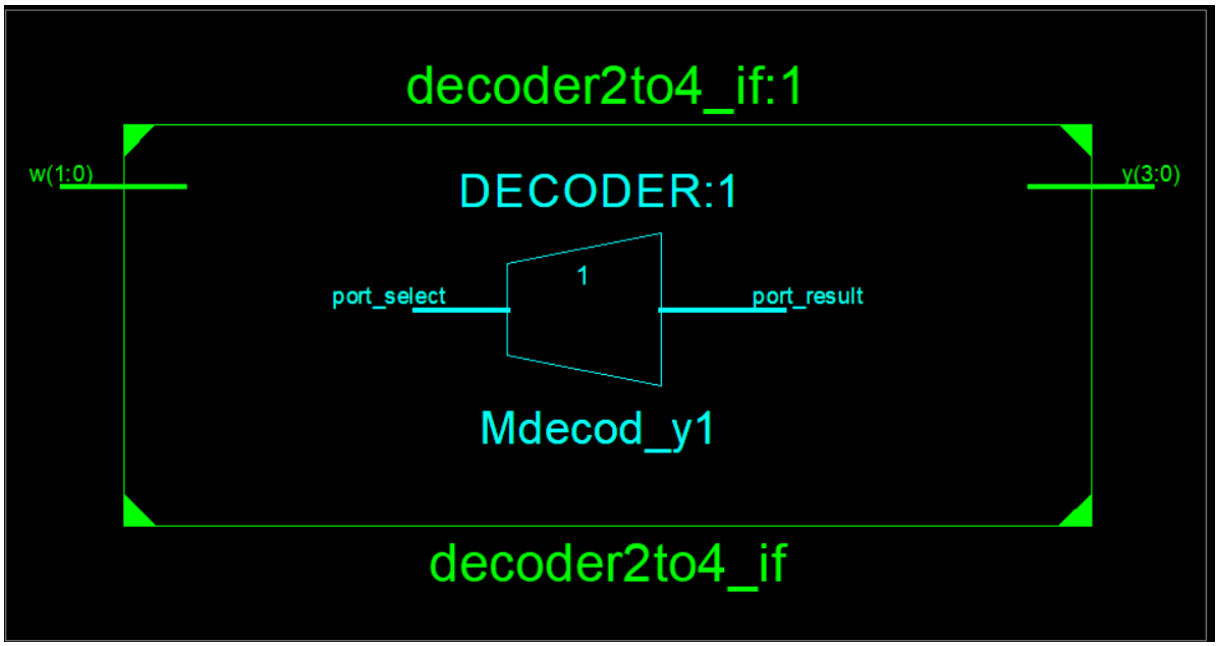


Hình 6.4. Biểu đồ dạng sóng

Nhận xét:

Từ 0-400 ns mạch hoạt động đúng bảng trạng thái khi nhập đúng đầu vào của 2 ngõ vào và chân e ở mức 0, khi nhập dữ liệu e=1 thì trả lại kết quả 1111 (thời điểm hơn 400ns).

7. **Mạch giải mã 2 sang 4 ngõ ra tích cực mức cao bằng lệnh if else.**



Hình 7.1. Mạch RTL

Code

```
module decoder2to4_if(
input wire [1:0] w,
output reg [3:0] y
   );
always @(w)
begin
 if(w==0)
 y=4'b0001;
 else if(w==1)
 y=4'b0010;
 else if(w==2)
```

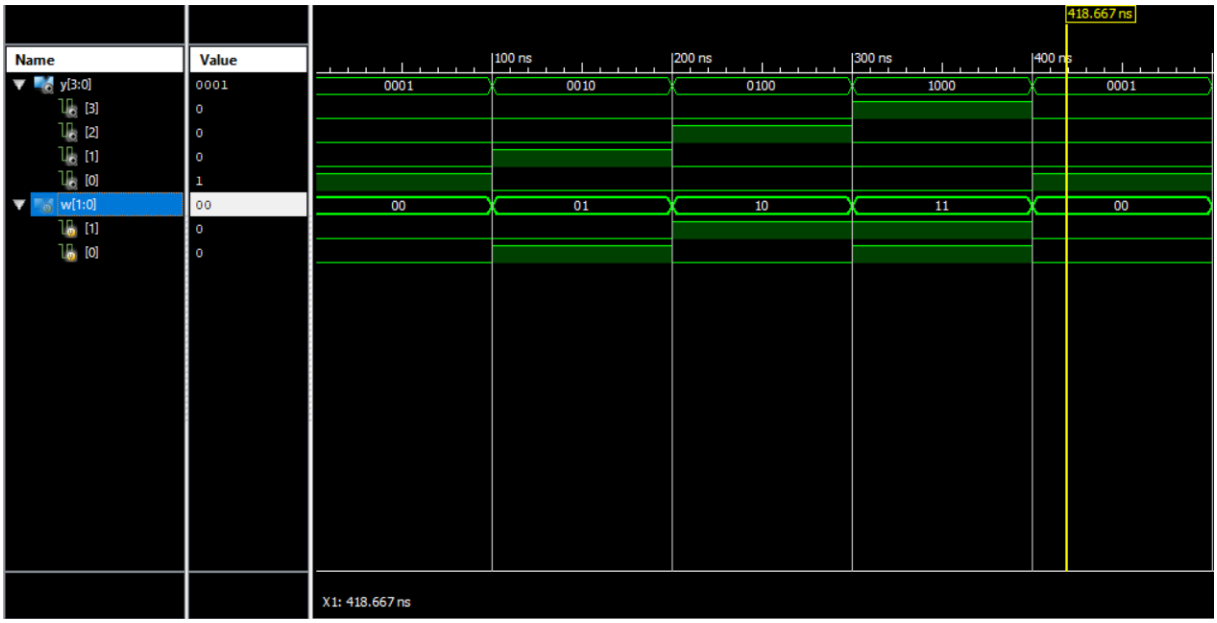
```
 y=4'b0100;
 else
 y=4'b1000;
end
endmodule
```

Code test

```
module test;
	reg [1:0] w;
	wire [3:0] y;
	decoder2to4_if uut (
		.w(w),
		.y(y)
	);
	initial begin
		w = 0;
		#100;
		w = 1;
		#100;
		w = 2;
		#100;
		w = 3;
		#100;
		w = 4;
		#100;
		$stop;
	end
```

```
endmodule
```

Kết quả

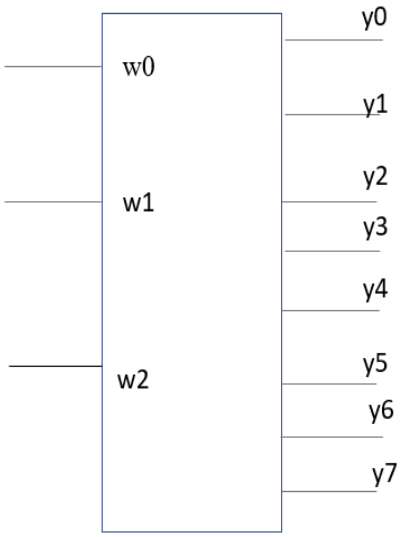


Hình 7.2. Kết quả dạng sóng

<u>Nhận xét:</u>

Từ 0-400 ms mạch hoạt động đúng bảng trạng thái khi nhập đúng đầu vào của 2 ngõ vào, khi nhập dữ liệu lớn hơn 2 bit thì đầu vào cũng chỉ hiện 2 bit trọng số nhỏ và đưa ra kết quả tương ứng (thời điểm hơn 400ms).

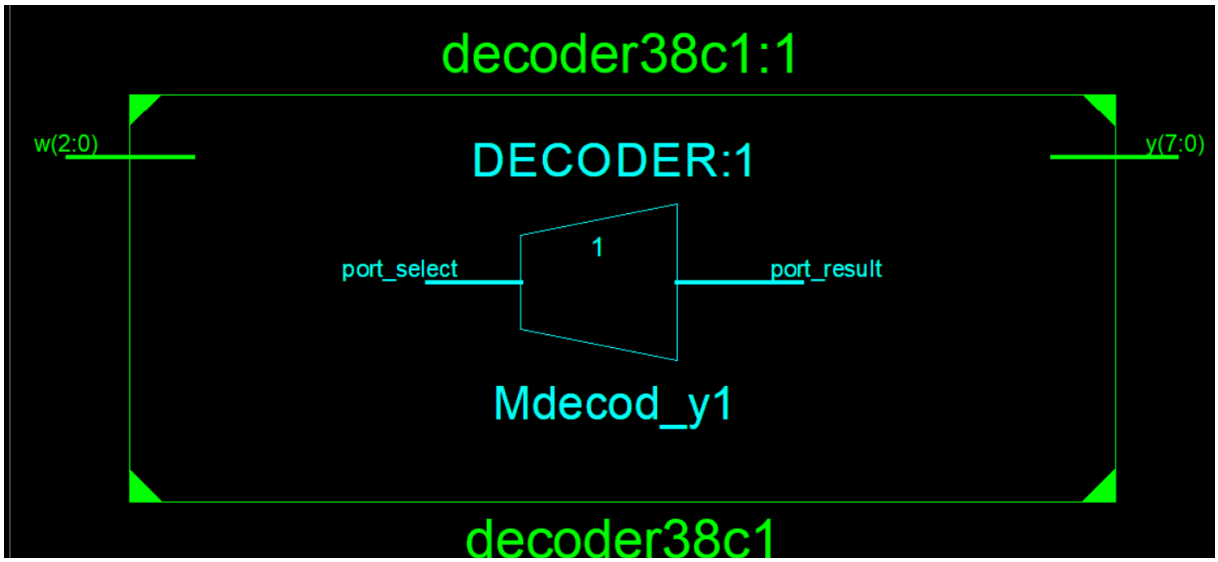
8. Mạch giải mã 3 sang 8 ngõ ra tích cực mức cao



Hình 8.1. Sơ đồ khối

w2	w1	w0	y7	y6	y5	y4	y3	y2	y1	y0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Hình 8.2. Bảng gía trị



Hinh 8.3. Mạch RTL

Code

```
module decoder38c1(
 input wire [2:0] w,
 output reg [7:0] y
  );
always @(w)
case (w)
 0: y = 8'b00000001;
 1: y = 8'b00000010;
 2: y = 8'b00000100;
 3: y = 8'b00001000;
 4: y = 8'b00010000;
 5: y = 8'b00100000;
 6: y = 8'b01000000;
 7: y = 8'b10000000;
endcase
endmodule
```

Code test

```
module test;
	reg [2:0] w;
	wire [7:0] y;
	decoder38c1 uut (
		.w(w),
		.y(y)
	);
	initial begin
		w = 0;
		#100;
    w = 1;
    #100;
    w = 2;
    #100;
    w = 3;
    #100;
    w = 4;
    #100;
    w = 5;
    #100;
    w = 6;
    #100;
    w = 7;
    #100;
    w = 8;
```

```
    #100;
    $stop;
        end
endmodule
```

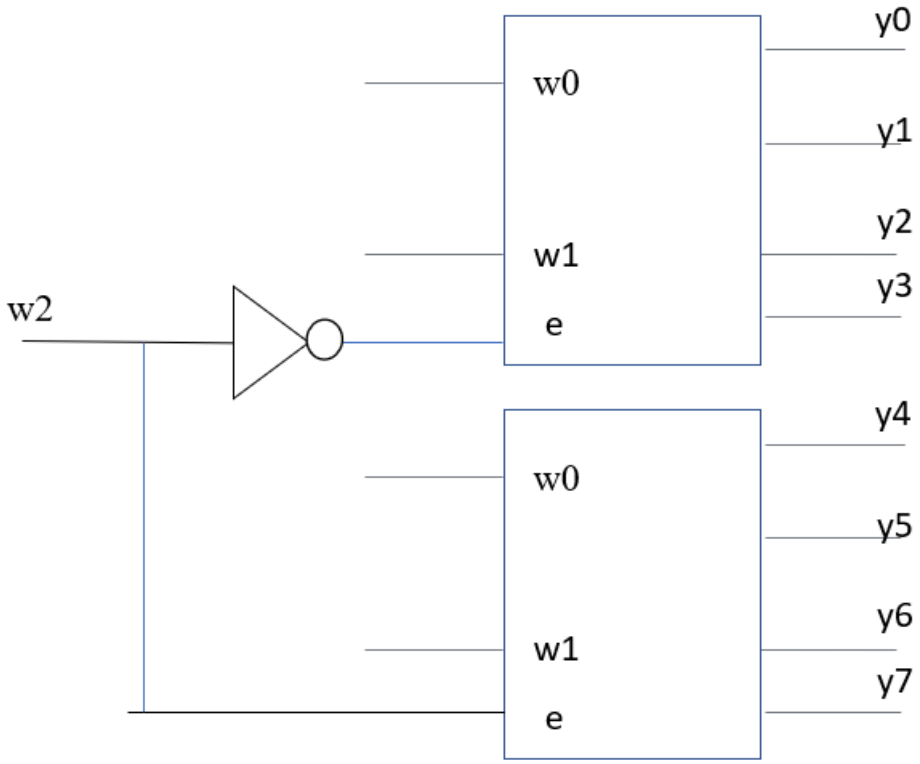
Kết quả

Hình 8.4. Kết quả dạng sóng

<u>Nhận xét:</u>

Từ 0-800 ns mạch hoạt động đúng bảng trạng thái khi nhập đúng đầu vào của 3 ngõ vào, khi nhập dữ liệu lớn hơn 3 bit thì đầu vào cũng chỉ hiện 3 bit trọng số nhỏ và đưa ra kết quả tương ứng (thời điểm hơn 800ns).

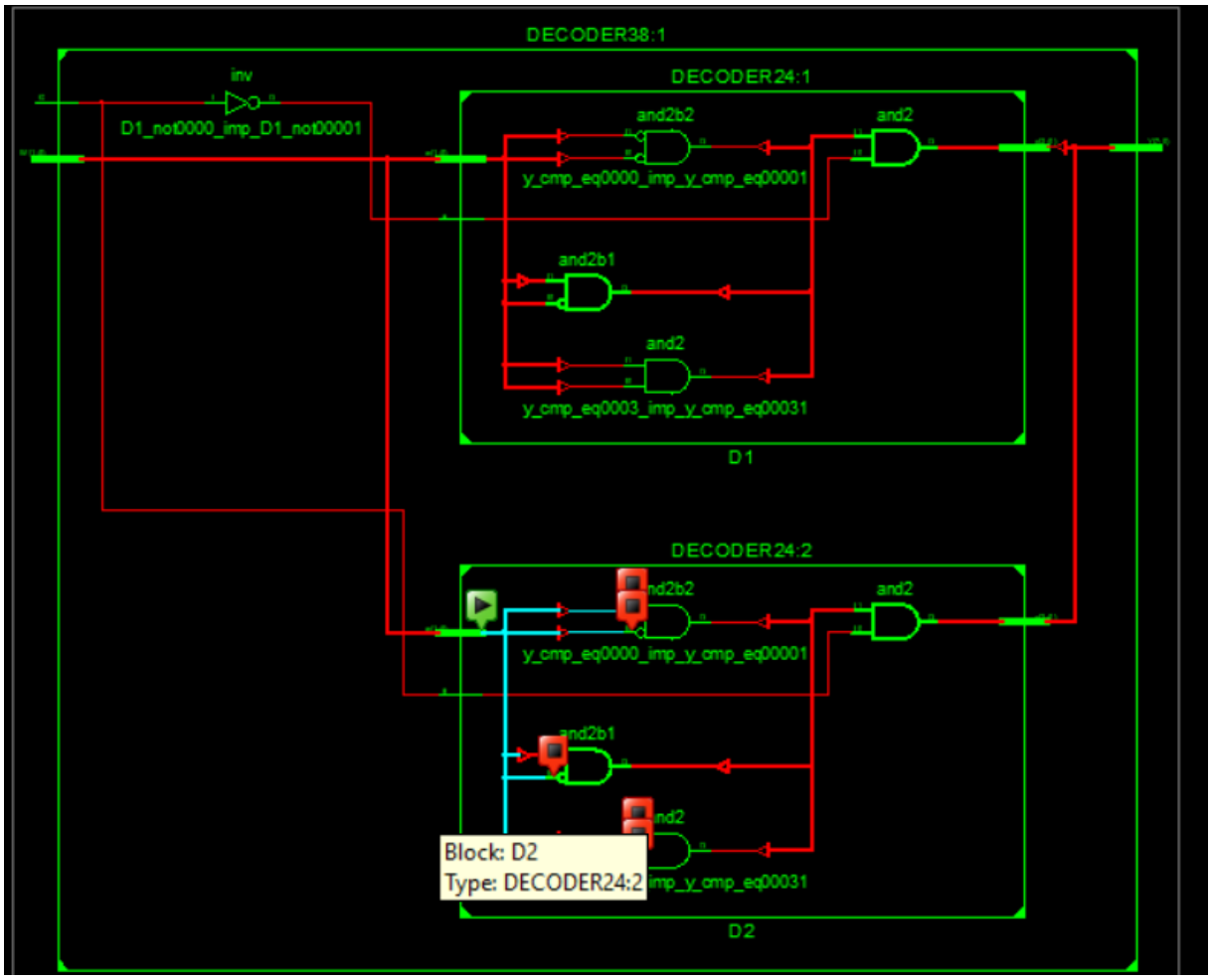
9. Mạch giải mã 3 sang 8 ngõ ra tích cực mức cao bằng cách ghép 2 mạch giải mã 2 sang 4



Hình 9.1. Sơ đồ khối

w2	w1	w0	y7	y6	y5	y4	y3	y2	y1	y0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Hình 9.2. Bảng giá trị



Hình 9.3. Mạch RTL

Code module decoder 2to4

```
module DECODER24(
input wire [1:0] w,
input g,
output reg [3:0] y
    );
          always @(w or g)
if(g==1)
 case (w)
0: y = 4'b0001;
1: y = 4'b0010;
```

```
2: y = 4'b0100;
3: y = 4'b1000;
 endcase
else y = 0;
endmodule
```

Code top module decoder3to8

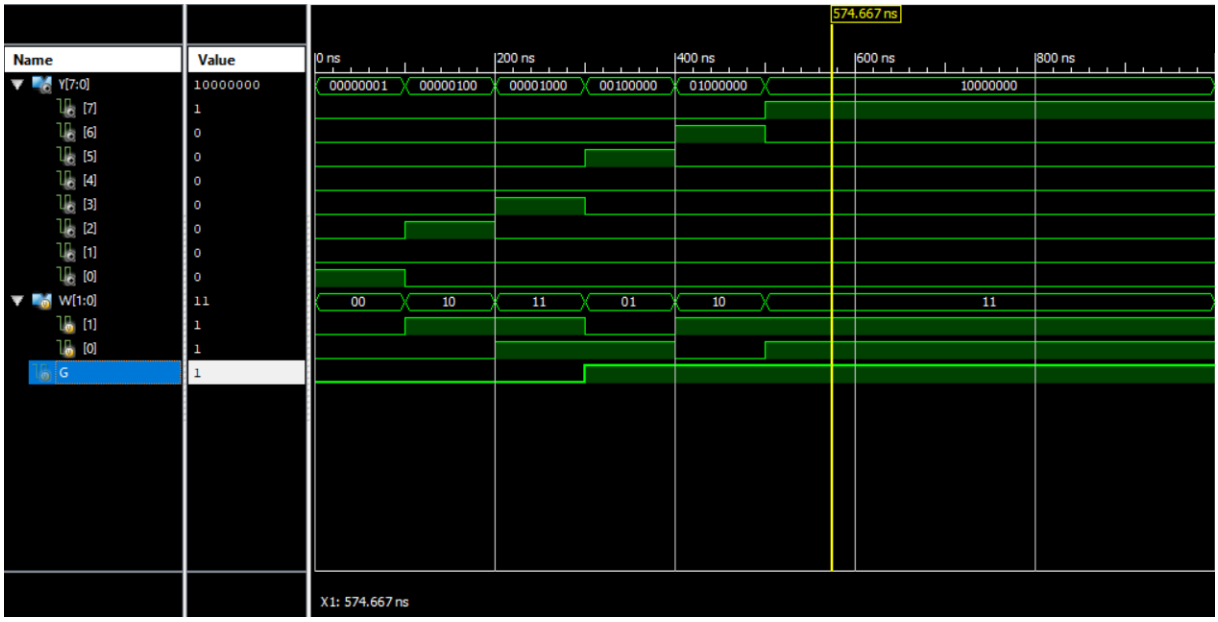
```
module DECODER38(
input [1:0] W,
input G,
output  [7:0] Y
   );
DECODER24 D1 (W[1:0],~G,Y[3:0]);
DECODER24 D2 (W[1:0],G,Y[7:4]);
endmodule
```

Code test

```
module test;
        reg [1:0] W;
        reg G;
        wire [7:0] Y;
        DECODER38 uut (
                .W(W),
                .G(G),
                .Y(Y)
        );
        initial begin
                W = 0;
```

```
                G = 0;
                #100;
                W = 2;
                G = 0;
                #100;
                W = 3;
                G = 0;
                #100;
                W = 1;
                G = 1;
                #100;
                W = 2;
                G = 1;
                #100;
                W = 3;
                G = 1;
                #600;
                $stop;
        end
endmodule
```

Kết quả

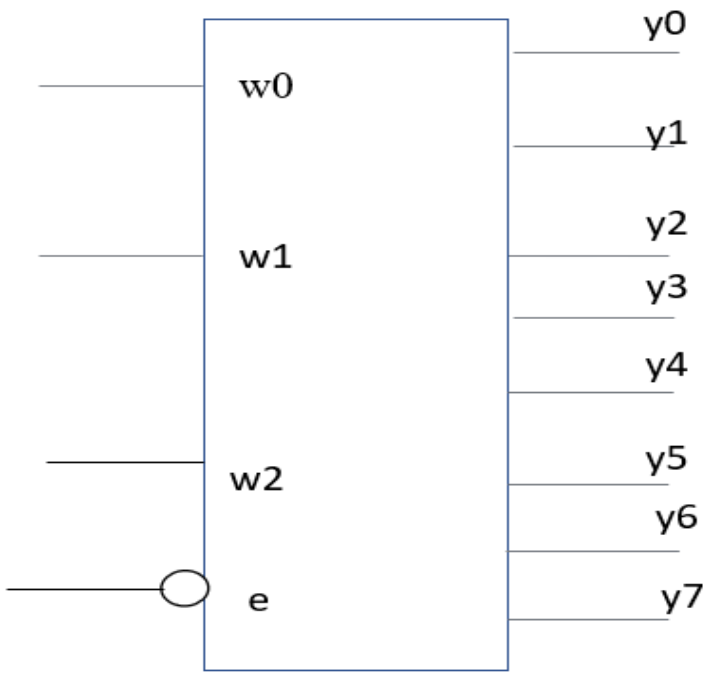


Hình 9.4. Biểu đồ dạng sóng

Nhận xét:

Từ 0-800 ns mạch hoạt động đúng bảng trạng thái khi nhập đúng đầu vào của 3 ngõ vào, khi nhập dữ liệu lớn hơn 3 bit thì đầu vào cũng chỉ hiện 3 bit trọng số nhỏ và đưa ra kết quả tương ứng (thời điểm hơn 800ns).

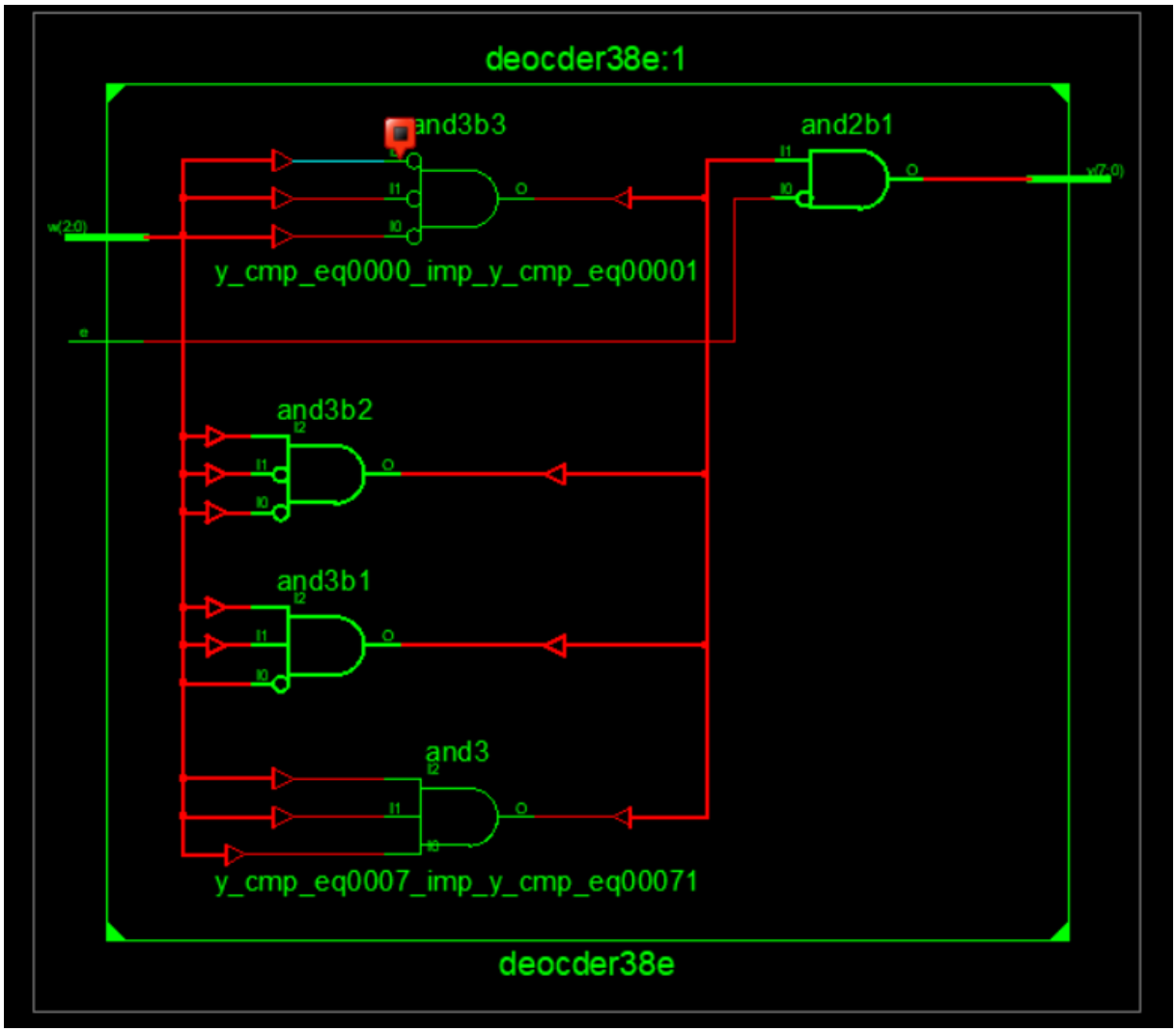
10. Mạch giải mã 3 sang 8 ngõ ra tích cực mức cao có chân enable mức thấp.



Hình 10.1. Sơ đồ khối

e	w2	w1	w0	y7	y6	y5	y4	y3	y2	y1	y0
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

Hình 10.2. Bảng trạng thái



Hình 10.3. Mạch RTL

Code

```
module deocder38e(
 input wire [2:0] w,
 input wire e,
 output reg [7:0] y
   );
          always @(w , e)
    if ( e==0)
     case (w)
      0: y = 8'b00000001;
```

```
    1: y = 8'b00000010;
    2: y = 8'b00000100;
    3: y = 8'b00001000;
    4: y = 8'b00010000;
    5: y = 8'b00100000;
    6: y = 8'b01000000;
    7: y = 8'b10000000;
 endcase
 else y=0;
endmodule
```

Code test

```
module text;
        reg [2:0] w;
        reg e;
        wire [7:0] y;
        deocder38e uut (
                .w(w),
                .e(e),
                .y(y)
        );
        initial begin
                w = 0;
                e = 0;
                #100;
    w = 1;
                e = 0;
```

```
    #100;
    w = 2;
                e = 0;
    #100;
    w = 3;
                e = 0;
    #100;
    w = 4;
                e = 0;
    #100;
    w = 5;
                e = 0;
    #100;
                e = 0;
    w = 6;
    #100;
    w = 7;
                e = 0;
    #100;
                w = 7;
                e = 1;
    #100;
    $stop;
        end
endmodule
```

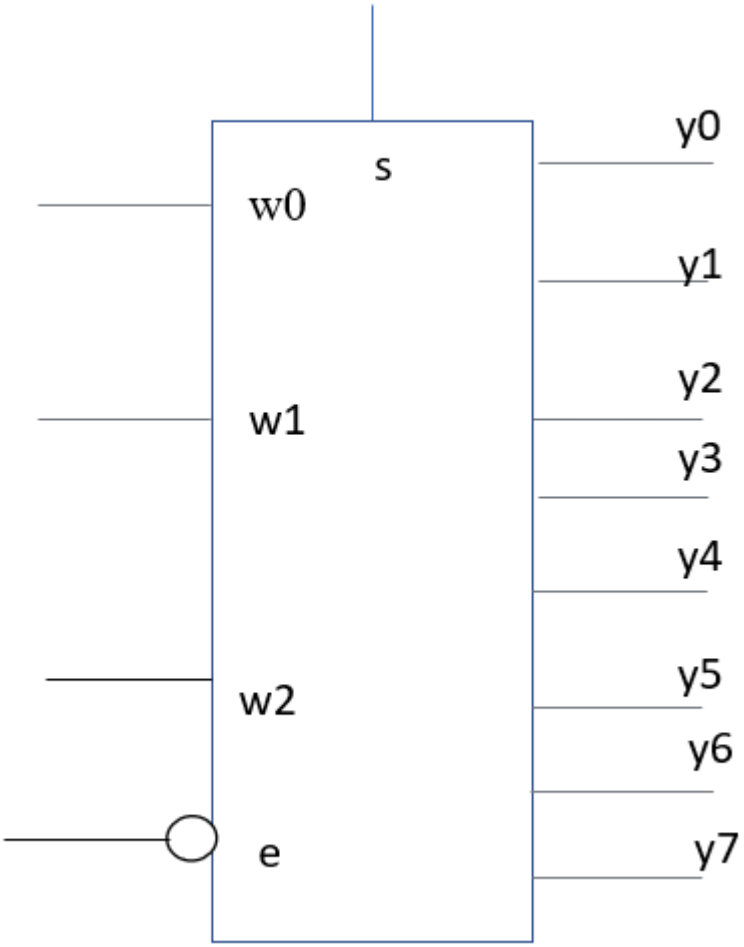
Kết quả

Hình 10.4. Biểu đồ dạng sóng

Nhận xét:

Từ 0-800 ns mạch hoạt động đúng bảng trạng thái khi nhập đúng đầu vào của 3 ngõ vào và chân e ở mức 0, khi nhập dữ liệu e=1 thì trả lại kết quả 00000000 (thời điểm hơn 800ns).

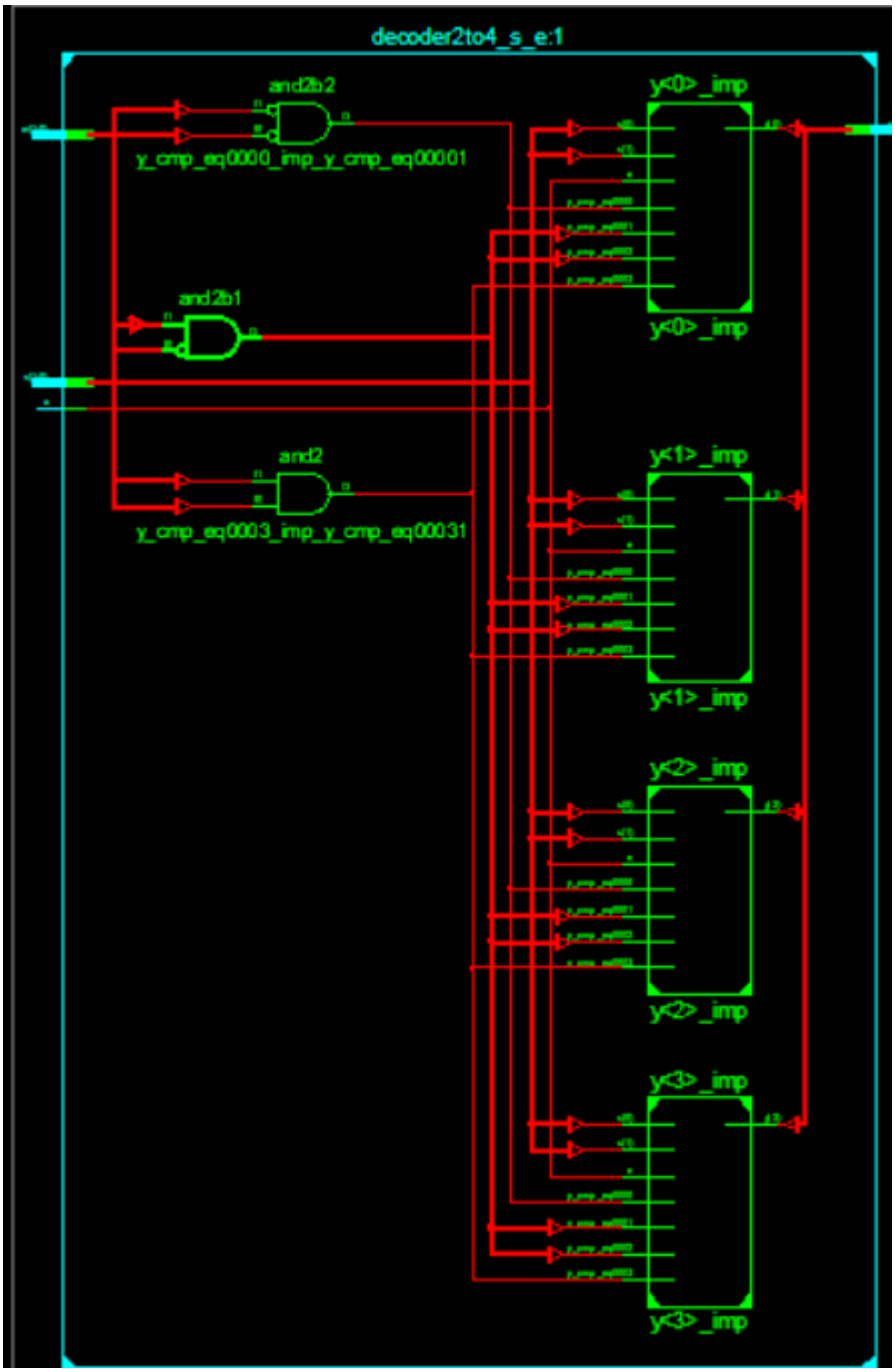
11. Mạch giải mã 3 sang 8 có chân s cho phép ngõ ra tích cực thấp hoặc cao và chân enable mức thấp.



Hình 11.1. Sơ đồ khối

s	e	w2	w1	w0	y7	y6	y5	y4	y3	y2	y1	y0
1	1	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	1	0	0
1	0	0	1	1	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	1	0	0	1	0	0	0	0	0
1	0	1	1	0	0	1	0	0	0	0	0	0
1	0	1	1	1	1	0	0	0	0	0	0	0
0	1	x	x	x	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1	1	1	0
0	0	0	0	1	1	1	1	1	1	1	0	1
0	0	0	1	0	1	1	1	1	1	0	1	1
0	0	0	1	1	1	1	1	1	0	1	1	1
0	0	1	0	0	1	1	0	0	1	1	1	1
0	0	1	0	1	1	1	0	1	1	1	1	1
0	0	1	1	0	1	0	1	1	1	1	1	1
0	0	1	1	1	0	1	1	1	1	1	1	1

Hình 11.2. Bảng trạng thái



Hình 11.3. Mạch RTL

Code

```
module deocder3to8_s_e(
 input [2:0] w,
 input s,e,
 output reg [7:0] y
   );
```

```
always @(s,e,w)
 if(s==1)
  if(e==0)
        case(w)
        0: y=8'b00000001;
        1: y=8'b00000010;
        2: y=8'b00000100;
        3: y=8'b00001000;
        4: y=8'b00010000;
        5: y=8'b00100000;
        6: y=8'b01000000;
        7: y=8'b10000000;
         endcase
         else y=0;
        else if (s==0)
        if(e==0)
         case(w)
        0: y=8'b11111110;
        1: y=8'b11111101;
        2: y=8'b11111011;
        3: y=8'b11110111;
        4: y=8'b11101111;
        5: y=8'b11011111;
        6: y=8'b10111111;
        7: y=8'b01111111;
         endcase
```

```
                else y=255;
endmodule
```

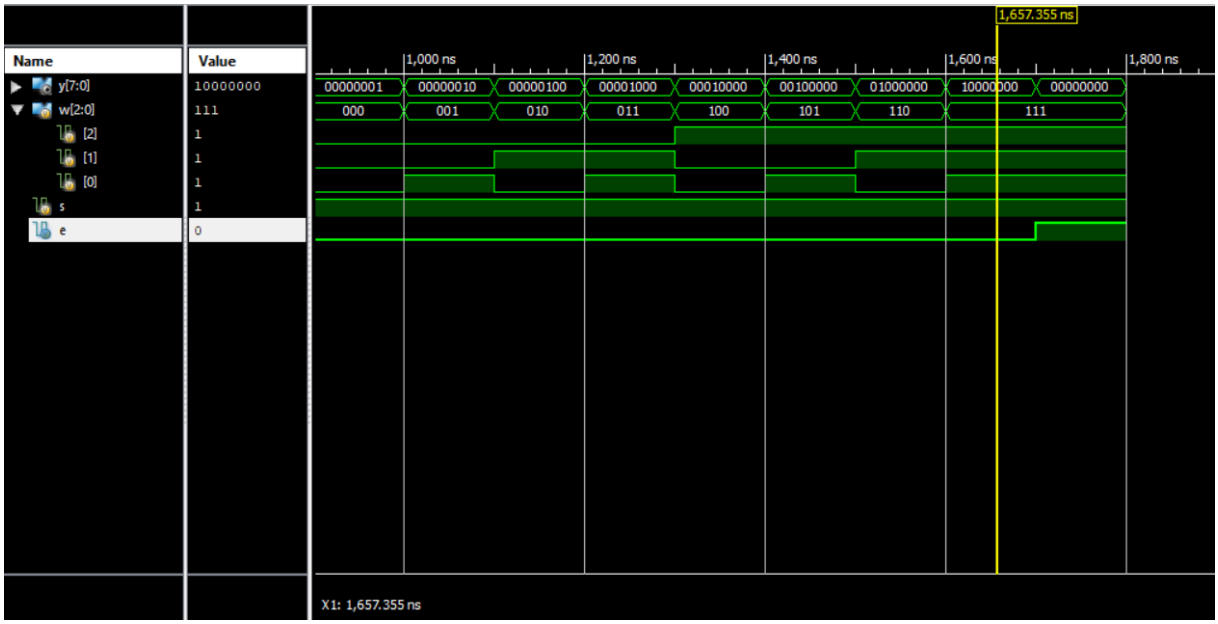
Code test

```
module test;
        reg [2:0] w;
        reg s;
        reg e;
        wire [7:0] y;
        deocder3to8_s_e uut (
                .w(w),
                .s(s),
                .e(e),
                .y(y)
        );

        initial begin
                w = 0;s = 0;e = 0;#100;
                w = 1;s = 0;e = 0;#100;
                w = 2;s = 0;e = 0;#100;
                w = 3;s = 0;e = 0;#100;
                w = 4;s = 0;e = 0;#100;
                w = 5;s = 0;e = 0;#100;
                w = 6;s = 0;e = 0;#100;
                w = 7;s = 0;e = 0;#100;
                w = 7;s = 0;e = 1;#100;
                w = 0;s = 1;e = 0;#100;
```

```
        w = 1;s = 1;e = 0;#100;
        w = 2;s = 1;e = 0;#100;
        w = 3;s = 1;e = 0;#100;
        w = 4;s = 1;e = 0;#100;
        w = 5;s = 1;e = 0;#100;
        w = 6;s = 1;e = 0;#100;
        w = 7;s = 1;e = 0;#100;
        w = 7;s = 1;e = 1;#100;
        $stop;
    end
endmodule
```

Kết quả

Hình 11.4. Kết quả dạng sóng thời điểm từ 0 đến 900 ns



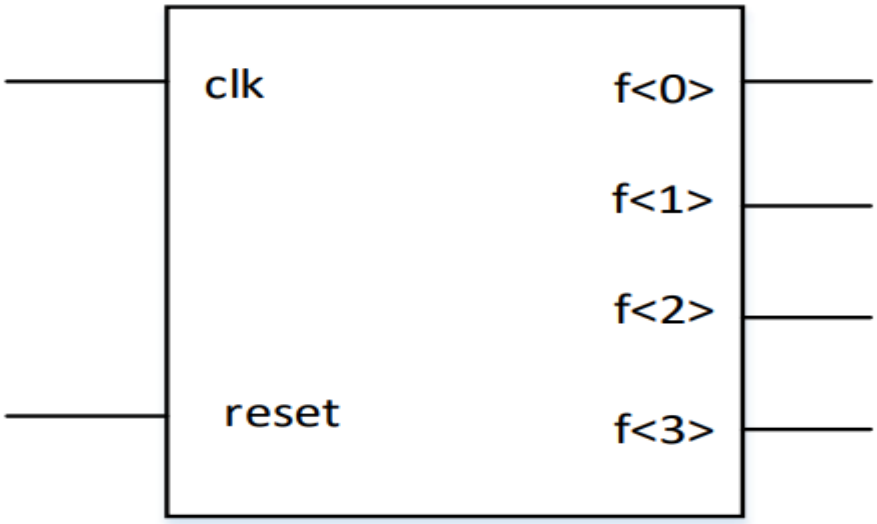
Hình 11.5. Kết quả dạng sóng thời điểm từ 900 đến 1800 ns

Nhận xét:

Mạch hoạt động đúng theo bảng trạng thái, khi chân s mức 0 chân e mức 0 mạch hoạt động như mạch 3 sang 8 ngõ ra tích cực mức thấp, khi chân e lên 1 trả về 11111111.

Mạch hoạt động đúng theo bảng trạng thái, khi chân s mức 1 chân e mức 0 mạch hoạt động như mạch 3 sang 8 ngõ ra tích cực mức cao, khi chân e lên 1 trả về 00000000.

12. Thiết kế mạch chia xung với ngõ vào 50Mhz, 4 xung ngõ ra với tần số f, 2f, 4f, 8f, trong đó lựa chọn f ~ 1Hz.



Hình 12.1. Sơ đồ khối bài 1

Module chia xung

```
module CLKdiv
#(parameter M=50000000)
(
 input clki,
 output 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
```

Module TOP

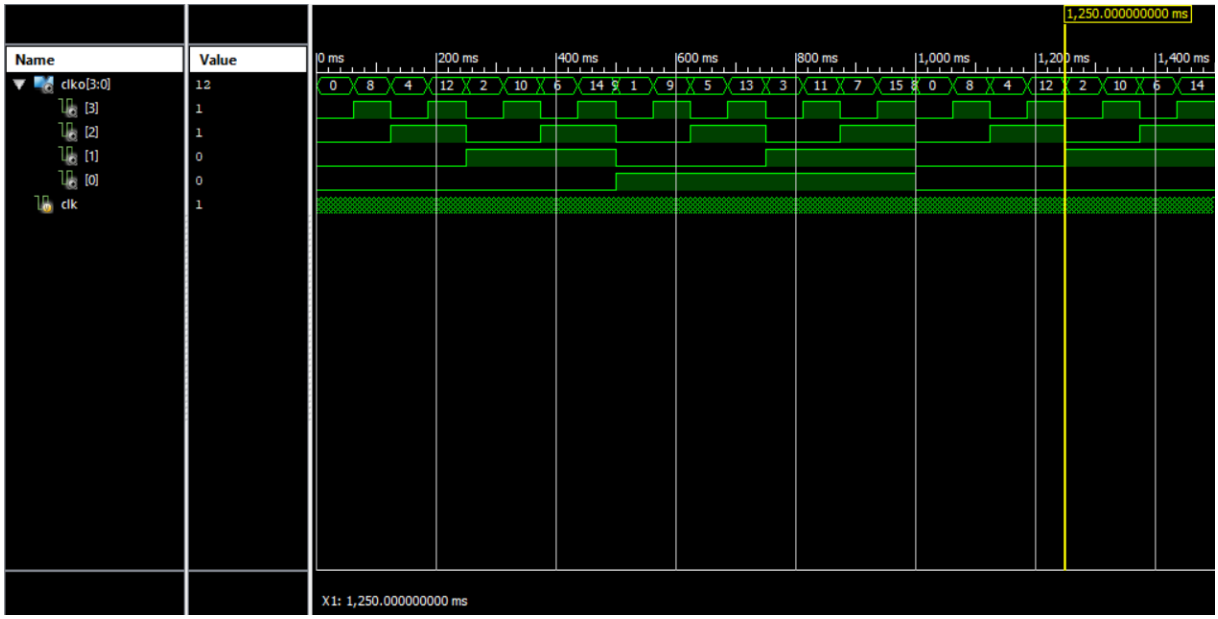
```
module CLKdevision(
input clk,
output [3:0] clko
    );
   CLKdiv #(50000000) cd0  (clk,clko[0]);
   CLKdiv #(25000000) cd1  (clk,clko[1]);
   CLKdiv #(12500000) cd2  (clk,clko[2]);
   CLKdiv #(6250000)   cd3  (clk,clko[3]);
endmodule
```

Test

```
module test;
        reg clk;
        wire [3:0] clko;
        top uut (
                .clk(clk),
                .clko(clko)
        );
        initial begin
                clk = 0;
                #100;
        end
    always
                begin
```

```
        clk=~clk;
        #10;
    end
endmodule
```

Kết quả



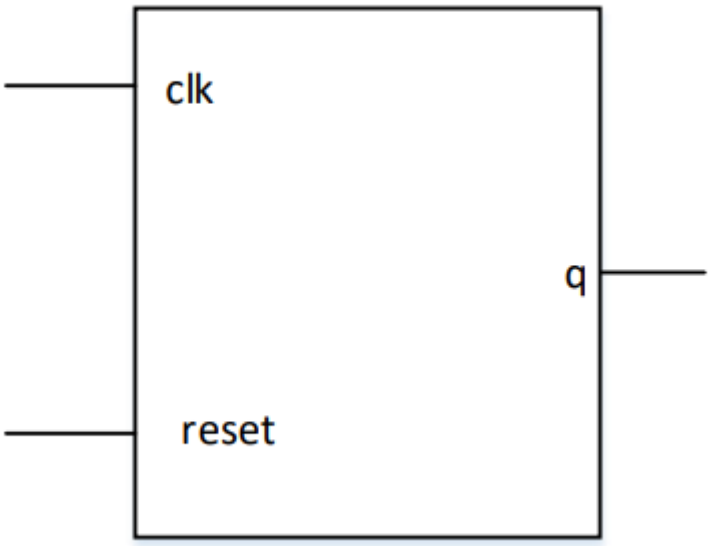
Hình 12.2. Kết quả dạng sóng bài 1

Nhận xét:

Mạch hoạt động với 4 tần số khác nhau với 4 ngõ ra q tương ứng:

- Q3 với tần số 8 Hz.
- Q2 với tần số 4 Hz.
- Q1 với tấn số 2 Hz.
- Q0 với tần số 1 Hz.

13. Thiết kế mạch tạo xung 1Hz.



Hình 13.1. Sơ đồ khối bài 2

Code

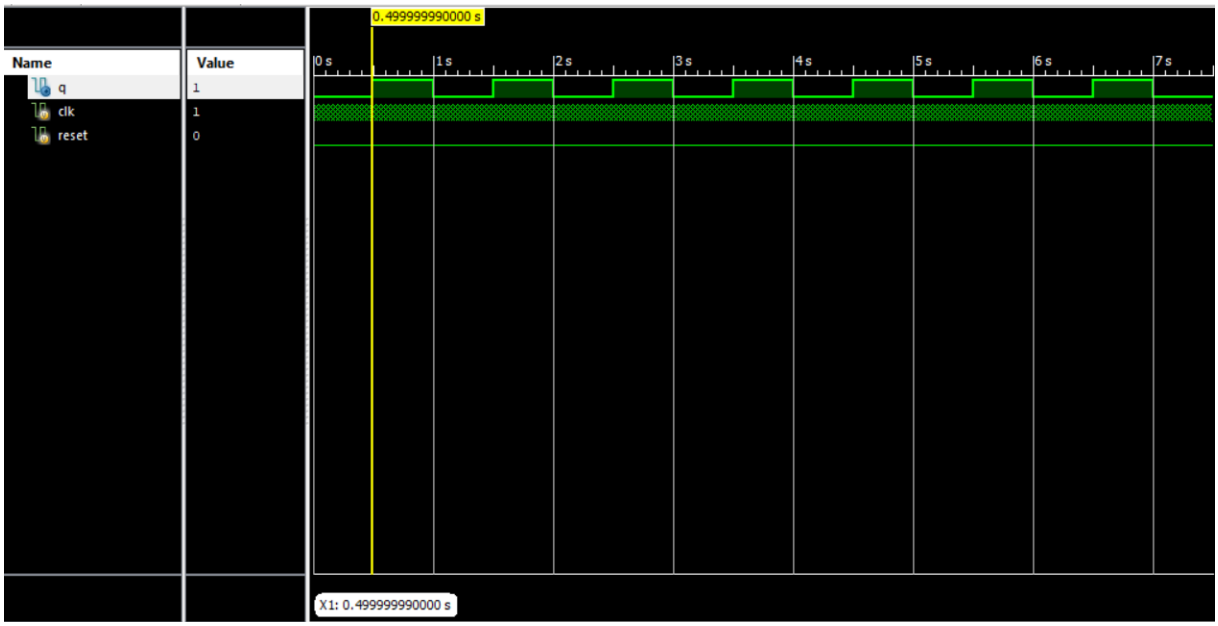
```
module Counter
#(parameter N= 26, M = 50000000)
( input wire clk, reset,
  output wire q
 );
        reg [N-1:0] r_reg;
        wire [N-1:0] r_next;
                always @(posedge clk, posedge reset)
                        if (reset)
                                r_reg <= 0;
                        else
                                r_reg<=r_next;
                assign r_next = (r_reg==M)?0:r_reg + 1;
                assign q=(r_reg<M/2)?0:1;
```

```
endmodule
```

Test

```
module hz;
        reg clk;
        reg reset;
        wire q;
        Counter uut (
                .clk(clk),
                .reset(reset),
                .q(q)
        );
        initial begin
                clk = 0;
                reset = 1;
                #10;
                reset=0;
                #100;
        end
        always
          begin
            #10;
            clk=~clk;
        end
endmodule
```

Kết quả

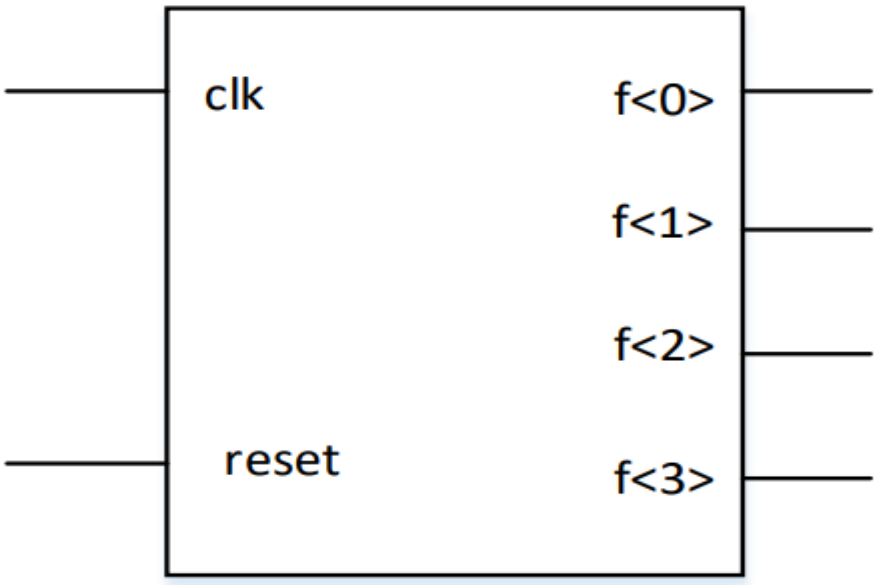


Hình 13.2. Biểu đồ dạng sóng bài 2

Nhận xét:

Mạch đếm đúng tần số 1 Hz.

14. Thiết kế mạch tạo 4 xung ngõ ra với tần số lần lượt là 0.1Hz, 1Hz, 10Hz, 100Hz.



Hình 14.1. Sơ đồ khối bài 3

Module chia xung

```
module Counter
#(parameter M=500000000)
(
 input clki,
 output 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
```

Module TOP

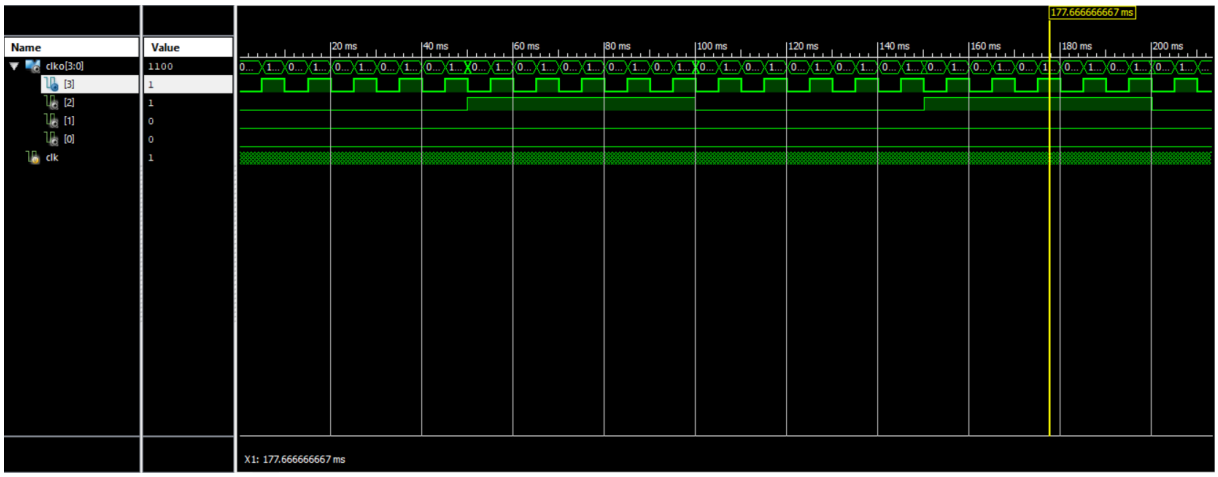
```
module top(
 input clk,
 output [3:0] clko


   );
  Counter #(500000000) cd0  (clk,clko[0]);
  Counter #(50000000)   cd1  (clk,clko[1]);
  Counter #(5000000)    cd2  (clk,clko[2]);
  Counter #(500000)     cd3  (clk,clko[3]);
endmodule
```

Test

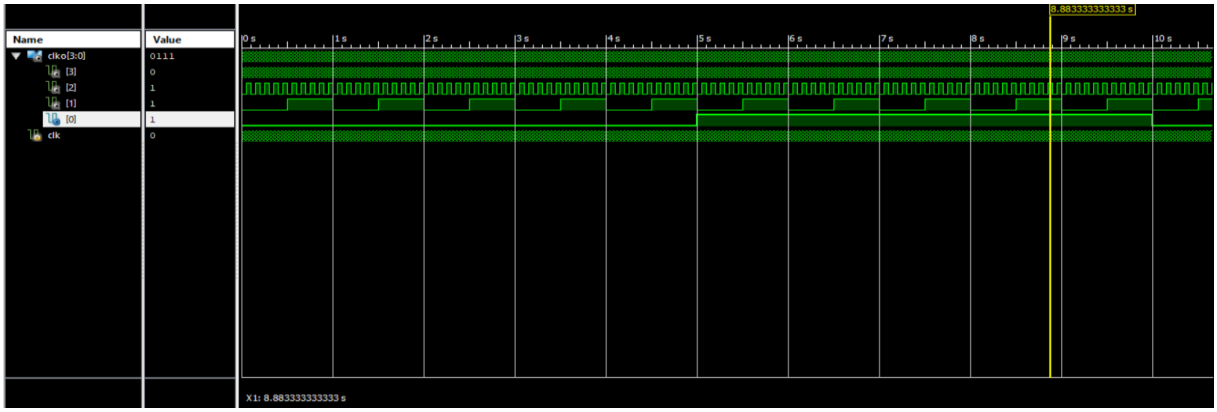
```
module test;
        reg clk;
        wire [3:0] clko;
        top uut (
                .clk(clk),
                .clko(clko)
        );
        initial begin
                clk = 0;
                #100;
        end
    always
```

```
        begin
         clk=~clk;
         #10;
        end
endmodule
```

Kết quả



Hình 14.2. thời điểm từ 0 đến hơn 200 ms, thấy được tần số 100 Hz ở q3 và 10 Hz ở q2.



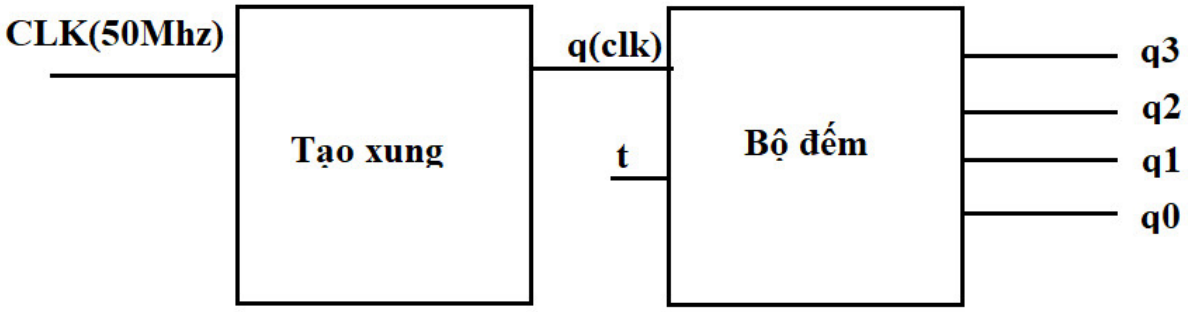
Hình 14.3. Thời điểm từ 0 đến hơn 10 s, thấy được tần số 1 Hz ở q1 và 0,1 Hz ở q0.

<u>Nhận xét:</u>

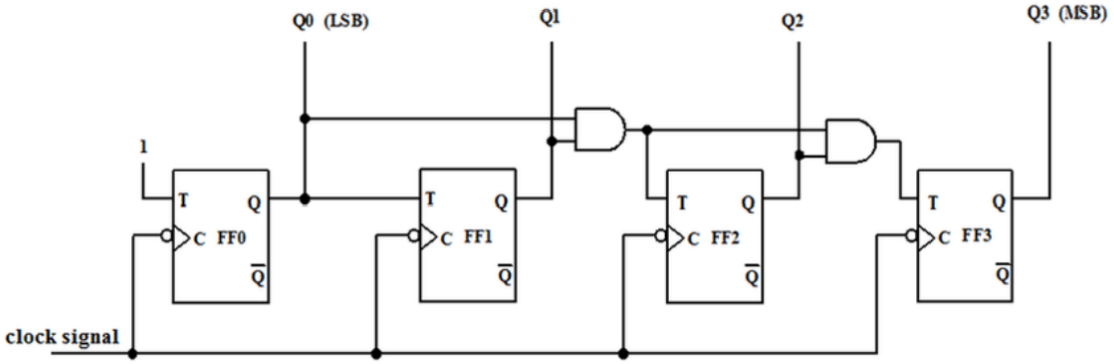
Mạch hoạt động với 4 tần số khác nhau với 4 ngõ ra q tương ứng:

- Q3 với tần số 100 Hz.
- Q2 với tần số 10 Hz.
- Q1 với tấn số 1 Hz.
- Q0 với tần số 0,1 Hz.

15. Thiết kế mạch đếm đồng bộ, sử dụng phương pháp cài đặt các Flip – Flop. Xung đếm 1Hz được lấy từ mạch chia xung.



Hình 15.1. Sơ đồ khối mạch đếm đồng bộ 1 Hz.



Hình 15.2. Bộ đếm

Module tạo xung 1 Hz.

```
module module_counter
#(parameter N= 26, M = 50000000)
( input wire clk,
  output wire q
 );
  reg  [N-1:0] r_reg;
  wire [N-1:0] r_next;
  initial
  r_reg = 0;
   always @(posedge clk)
```

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

Bộ đếm

```
module ff_t(
 input t,clk,
 output reg [3:0] q
);
initial q=4'b0000;
always @(posedge clk)
 begin
  if(t==1)
   q=~q;
  else
   q=q;
 end
endmodule
```

Module TOP.

```
module dem_dong_bo(
input  t,
output reg [3:0] q,
input clk
    );
```

```
 wire [3:0] t0,t1,t2,t3;
 module_counter module_counter(clk, clk1);
  ff_t fft0 ( t, clk1, t0 );
  ff_t fft1 ( t0,clk1, t1 );
  ff_t fft2 ( t0&&t1, clk1, t2);
  ff_t fft3 ( t0&&t1&&t2, clk1, t3);
 always @(posedge clk)
 begin
   q[0]=t0;
   q[1]=t1;
   q[2]=t2;
   q[3]=t3;
 end
endmodule
```

Test

```
module test;
        reg t;
        reg clk;
        wire [3:0] q;
        dem_dong_bo uut (
                .t(t),
                .q(q),
                .clk(clk)
        );
        initial begin
                t = 0;
```

```
        clk = 0;
        end
        always #1 clk=~clk;
        always #1 t=~t;
endmodule
```

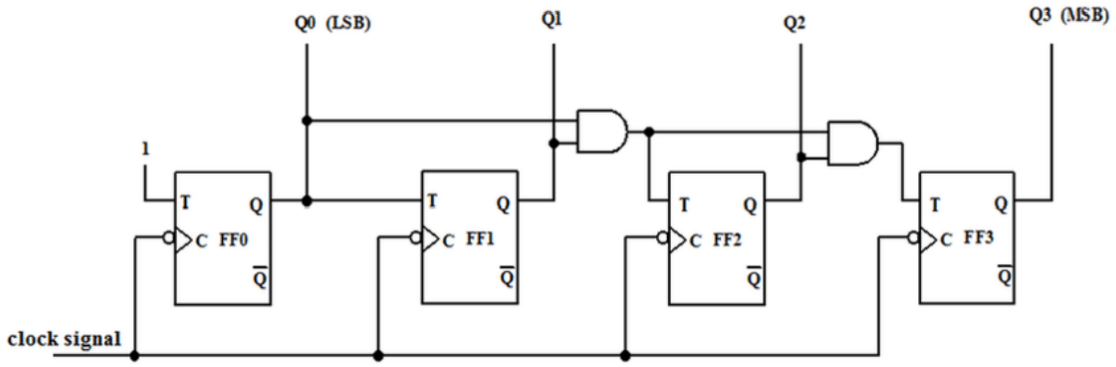
Kết quả

Hình 15.3. Kết quả dạng sóng ứng với test.

<u>Nhận xét:</u>

Mạch đếm đúng với tần số 1 Hz.

16. Thiết kế mạch đếm lên 4 bit như bài 4, sử dụng phương pháp thiết kế đồng bộ.



Hình 16.1. Sơ đồ khối bài 5

Module đếm

```
module ff_t(
 input t,clk,
 output reg [3:0] q);
  initial q=4'b0000;
  always @(posedge clk)
   begin
    if(t==1)
     q=~q;
    else
     q=q;
  end
endmodule
```

Module TOP

```
module ddb(
input  t,
output reg [3:0] q,
```

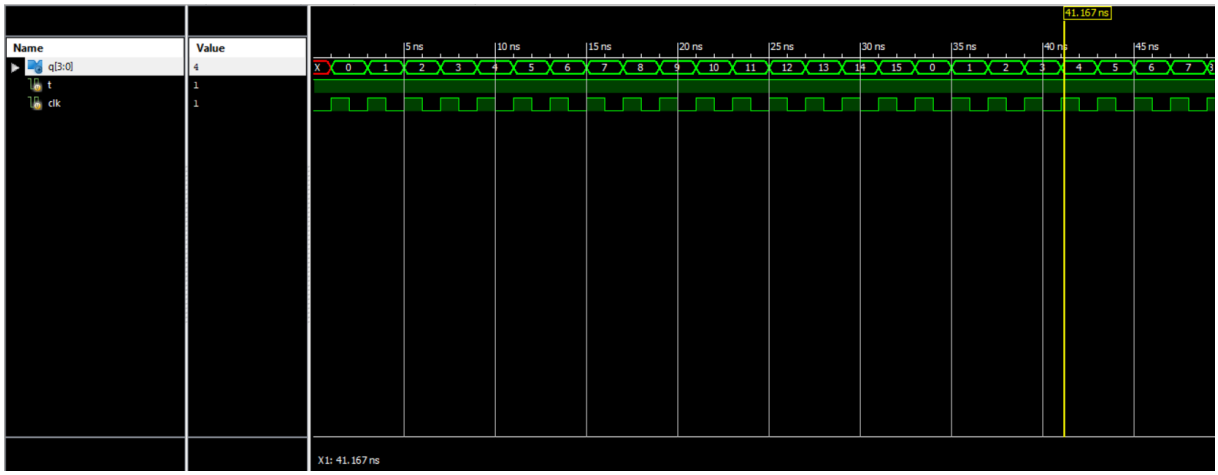
```
input clk
    );
wire [3:0] t0,t1,t2,t3;
ff_t fft0 ( t, clk, t0 );
ff_t fft1 ( t0,clk, t1 );
ff_t fft2 ( t0&&t1, clk, t2);
ff_t fft3 ( t0&&t1&&t2, clk, t3);
always @(posedge clk)
  begin
   q[0]=t0;
   q[1]=t1;
   q[2]=t2;
   q[3]=t3;
  end
endmodule
```

Test.

```
module test;
        reg t;
        reg clk;
        wire [3:0] q;
        ddb uut (
                .t(t),
                .q(q),
                .clk(clk)
        );
        initial begin
```

```
        t = 1;
        clk = 0;
        end
        always #1 clk=~clk;
endmodule
```

Kết quả.

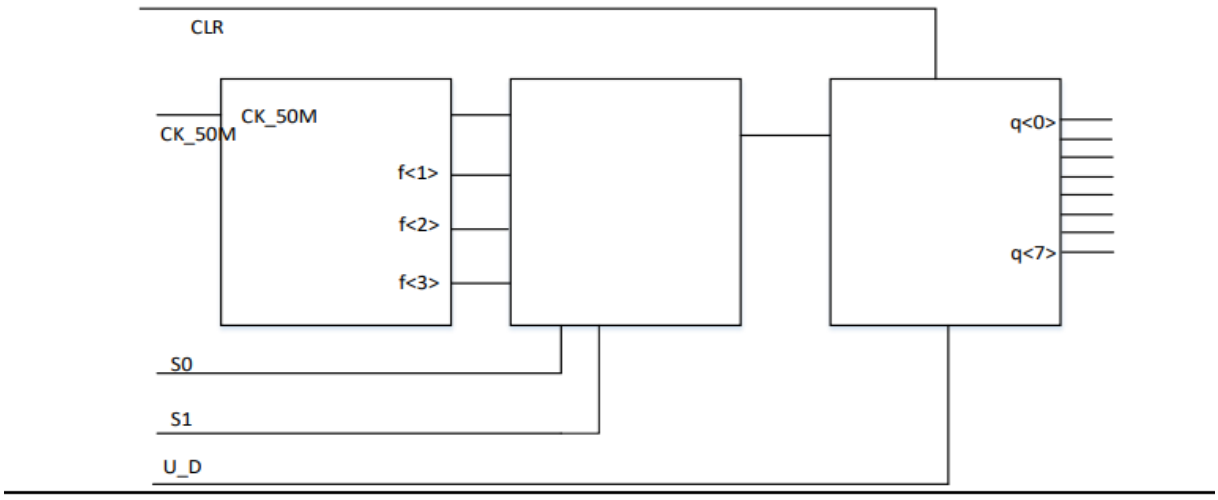


Hình 16.2. Biểu đồ dạng sóng tương ứng test.

<u>Nhận xét:</u>

Mạch đếm lên 4 bit khi đạt tới trạng thái 1111 tương ứng 15 hệ thập phân sẽ trở lại về giá trị 0000 tương ứng 0 hệ thập phân và đếm lên lại.

17. Thiết kế mạch đếm lên 8 bit, lựa chọn tần số đếm, lựa chọn đếm lên hoặc đếm xuống.



Hình 17.1. Sơ đồ khối bài 6

Module tạo xung

```
module Clock_div
#(parameter N= 30, M = 500000000)
( input wire clk,
  output wire [3:0]q
 );
reg [N-1:0] r_reg01H,r_reg1H,r_reg10H,r_reg100H;
wire [N-1:0]
r_next01H,r_next1H,r_next10H,r_next100H;
initial
begin
r_reg01H=0;
r_reg1H=0;
r_reg10H=0;
r_reg100H=0;
```

```
end
always @(posedge clk) begin
r_reg01H<=r_next01H;
r_reg1H<=r_next1H;
r_reg10H<=r_next10H;
r_reg100H<=r_next100H;
end
assign r_next01H = (r_reg01H==M)?0:r_reg01H + 1;
assign r_next1H = (r_reg1H==M/10)?0:r_reg1H + 1;
assign r_next10H = (r_reg10H==M/100)?0:r_reg10H + 1;
assign r_next100H = (r_reg100H==M/1000)?0:r_reg100H + 1;
assign q[0]=(r_reg01H<M/2)?0:1;
assign q[1]=(r_reg1H<M/20)?0:1;
assign q[2]=(r_reg10H<M/200)?0:1;
assign q[3]=(r_reg100H<M/2000)?0:1;
endmodule
```

Module Mux41

```
module Mux41
( input wire [3:0] clk,
  input wire [1:0] sw,
  output reg clk_o
 );
always @(clk,sw)
 case (sw)
  0: clk_o = clk[0];
```

```
  1: clk_o = clk[1];
  2: clk_o = clk[2];
  3: clk_o = clk[3];
 endcase
endmodule
```

Module đếm

```
module CounterUD
#(parameter N= 8)
( input wire clk,reset,ud,
  output wire [7:0]q
 );
 reg   [N-1:0] r_reg;
 wire [N-1:0] r_next;
 always @(posedge clk, posedge reset)
  if (reset)
   r_reg<=0;
  else
   r_reg<=r_next;
 assign r_next = (ud==1)?r_reg + 1:r_reg - 1;
 assign q=r_reg;
endmodule
```

Module Top

```
module LED_COUNTER(
input wire clk, reset,
input wire [1:0] SW,
```

```
input wire UD,
output wire [7:0] LED
 );
 wire [3:0] f;
 wire clk_o ;
  Clock_div clockdivider (clk, f) ;
  Mux41 mux4to1 (f,SW,clk_o);
  CounterUD counter (clk_o, reset,UD,LED);
endmodule
```

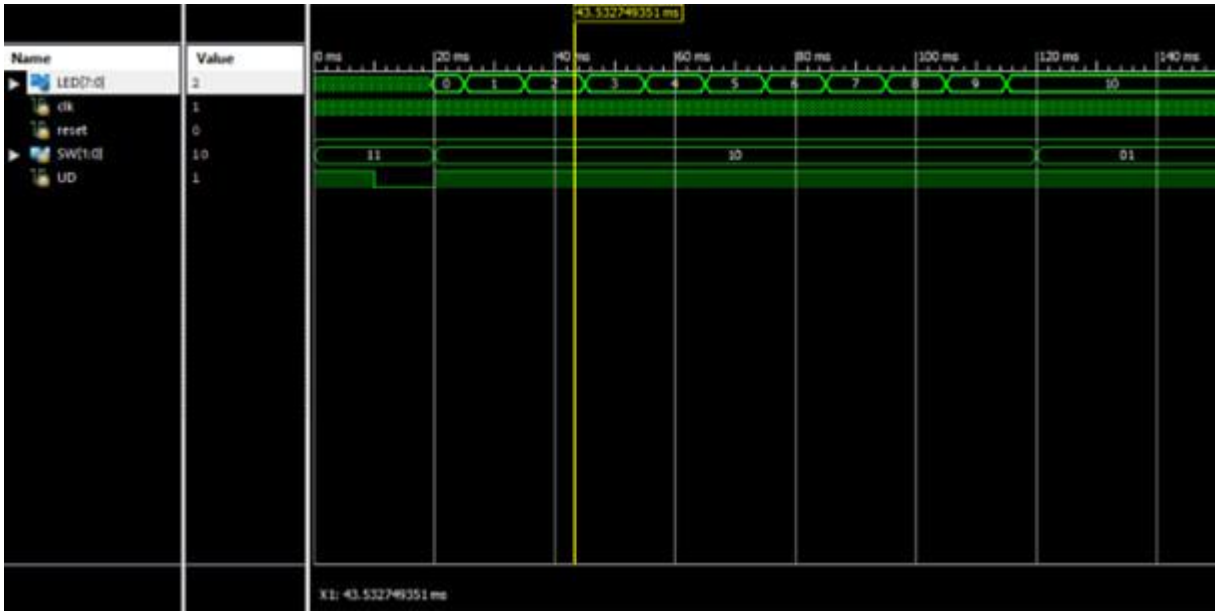
Test

```
module test;
        reg clk;
        reg reset;
        reg [1:0] SW;
        reg UD;
        wire [7:0] LED;
        LED_COUNTER uut (
                .clk(clk),
                .reset(reset),
                .SW(SW),
                .UD(UD),
                .LED(LED)
        );
initial begin
  clk = 0;
forever #1 clk = ~clk;
```

```
end


        initial begin
                reset = 1;
                #1;
                reset = 0;
                SW = 3;
                UD = 1;
                #10000000;
                SW = 3;
                UD = 0;
                #10000000;
                SW = 2;
                UD = 1;
                #100000000;
                SW = 1;
                UD = 1;
                #1000000000;
                SW = 0;
                UD = 1;
                #1000000000;
        end
endmodule
```

Kết quả.

Hình 17.3. Thời điểm từ 0 ms đến hơn 35 ms.



Hình 17.4. Thời điểm từ 0 ms đến hơn 140 ms.

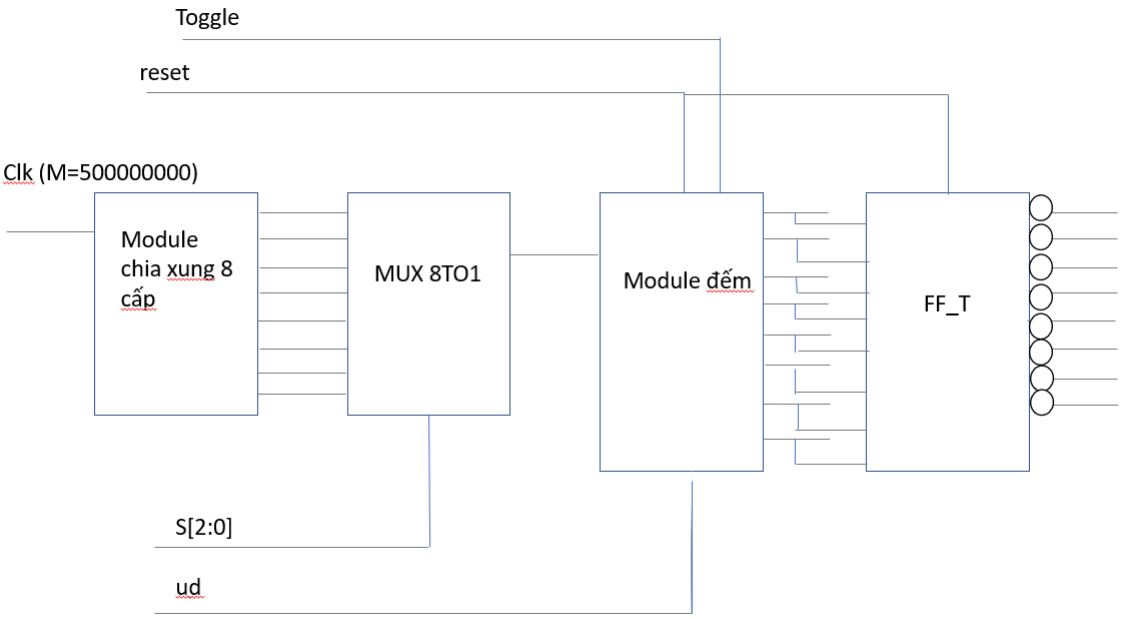
Hình 17.5. Thời điểm từ 100 ms đến hơn 220 ms.

Hình 17.6. Thời điểm 400 ms đến hơn 1400 ms.

Nhận xét:

- Sau khi có xung reset thì mạch bắt đầu đếm.
- Bắt đầu đếm với sw = 3 tương ứng mức tần số 100 Hz và ud = 1 mạch bắt đầu đếm lên với chu kỳ 1 ms, tại thời điểm 10.5 ms ud = 0 mạch bắt đầu đếm xuống.
- Tại thời điểm 20 ms với sw = 2 mạch đếm với tần số 10 Hz.
- Tại thời điểm 120 ms với sw = 1 mạch đếm với tần số 1 Hz.
- Tại thời điểm 1120 ms với sw = 0 mạch đếm với tần số 0,1 Hz.

18. Thiết kế mạch đếm lên 8 bit, lựa chọn 8 tần số đếm khác nhau, lựa chọn đếm lên hoặc đếm xuống, có tính hiệu cho phép dừng đếm (Pause), có tín hiệu đảo trạng thái ngõ ra.



Hình 18.1. Sơ đồ khối bài 7

Code Module tạo xung

```
module Clock_div_8f
(parameter N= 30, M = 500000000)
( input wire clk,
  output wire [7:0] clk_o
   );
        Wire [N-1:0]
r_next01Hz,r_next1Hz,r_next10Hz,r_next100Hz,r_next1000Hz,r_next10000Hz,r_next100000Hz,r_next1000000Hz;
  reg                                                    [N-1:0]
r_reg01Hz,r_reg1Hz,r_reg10Hz,r_reg100Hz,r_reg1000Hz,r_reg10000Hz,r_reg100000Hz,r_reg1000000Hz;
```

```
 initial r_reg01Hz=0;
 initial r_reg1Hz=0;
 initial r_reg10Hz=0;
 initial r_reg100Hz=0;
 initial r_reg1000Hz=0;
 initial r_reg10000Hz=0;
 initial r_reg100000Hz=0;
 initial r_reg1000000Hz=0;
always @(posedge clk)
 begin
 r_reg01Hz<=r_next01Hz;
 r_reg1Hz<=r_next1Hz;
 r_reg10Hz<=r_next10Hz;
 r_reg100Hz<=r_next100Hz;
 r_reg1000Hz<=r_next1000Hz;
 r_reg10000Hz<=r_next10000Hz;
 r_reg100000Hz<=r_next100000Hz;
 r_reg1000000Hz<=r_next1000000Hz;
 end

 assign r_next01Hz      = (r_reg01Hz==M)?0 : r_reg01Hz + 1;
 assign r_next1Hz       = (r_reg1Hz==M/10)?0 : r_reg1Hz + 1;
 assign r_next10Hz      = (r_reg10Hz==M/100)?0 : r_reg10Hz + 1;
 assign r_next100Hz     = (r_reg100Hz==M/1000)?0 : r_reg100Hz +1;
 assign r_next1000Hz    = (r_reg1000Hz==M/10000)?0 : r_reg1000Hz + 1;
 assign r_next10000Hz   = (r_reg10000Hz==M/100000)?0 : r_reg10000Hz + 1;
```

```
assign r_next100000Hz  = (r_reg100000Hz==M/1000000)?0 : r_reg100000Hz + 1;
assign r_next1000000Hz = (r_reg1000000Hz==M/10000000)?0 : r_reg1000000Hz +1;


assign clk_o[0]=(r_reg01Hz<M/2)?0:1;
assign clk_o[1]=(r_reg1Hz<M/20)?0:1;
assign clk_o[2]=(r_reg10Hz<M/200)?0:1;
assign clk_o[3]=(r_reg100Hz<M/2000)?0:1;
assign clk_o[4]=(r_reg1000Hz<M/20000)?0:1;
assign clk_o[5]=(r_reg10000Hz<M/200000)?0:1;
assign clk_o[6]=(r_reg100000Hz<M/2000000)?0:1;
assign clk_o[7]=(r_reg1000000Hz<M/20000000)?0:1;
endmodule
```

Code Module MUX81

```
module MUX81(
   input wire [7:0] clk,
          input wire [2:0] s,
          output reg clk_o
   );
           always @(clk,s)
     case(s)
       0: clk_o= clk[0];
              1: clk_o= clk[1];
              2: clk_o= clk[2];
              3: clk_o= clk[3];
              4: clk_o= clk[4];
```

```
            5: clk_o= clk[5];
            6: clk_o= clk[6];
            7: clk_o= clk[7];
     endcase
endmodule
```

Code Module đếm

```
module Counter(
    input wire clk,reset,ud,toggle,
    output wire [7:0]q);
    wire [7:0] r_next;
    reg [7:0] r_reg;
    initial r_reg=0;
 always @(posedge clk, posedge reset)
  if (reset)
   r_reg<=0;
  else
   r_reg<=r_next;
  assign r_next = (ud==1)? r_reg + 1 : r_reg - 1;
  assign q=(toggle==0)?r_reg:~r_reg;
endmodule
```

Module FF_T

```
module ff_t(
   input T, rs, clk,
   output reg Q
```

```
   );
            always @ (posedge clk, posedge rs)
      if(rs==1)
       Q=0;
      else
        if(T==1)
          Q=~Q;
endmodule
```

Module TOP

```
module top(
   input wire clk, reset,
   input wire [2:0] S,
   input wire UD,Pause,Toggle,
   output wire [7:0] Q);


  wire [3:0] f;
  wire clk_o ;
  wire clk1,clk_gate;


   Clock_div_8f   m1 (clk1, f) ;
   MUX81          m2 (f,S,clk_o);
   Counter        m3 (clk_o,reset,UD,Toggle,Q);
   ff_t           m4 (1'b1,reset,Pause,clk_gate);
   and(clk1,clk,~clk_gate);
```

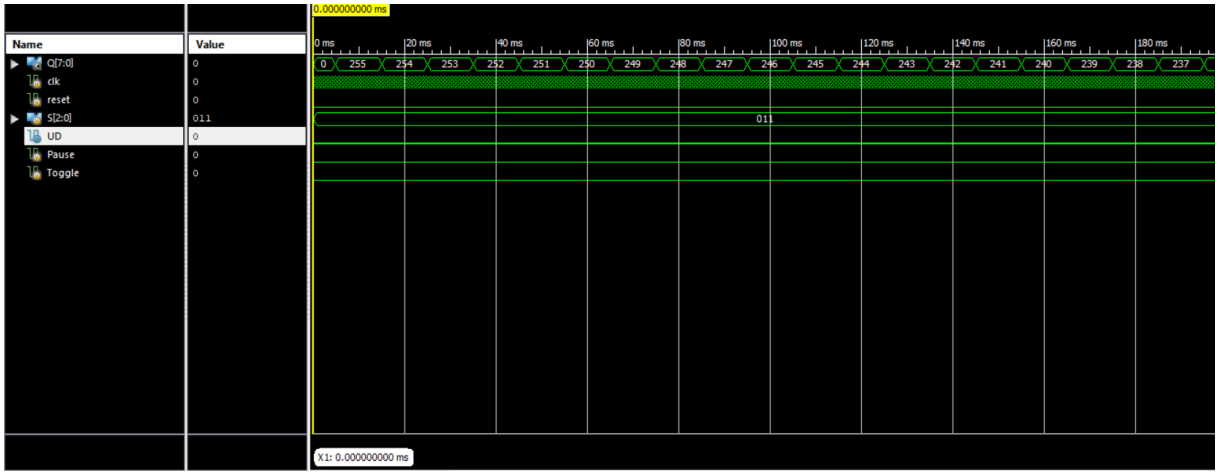
```
endmodule
```

Test

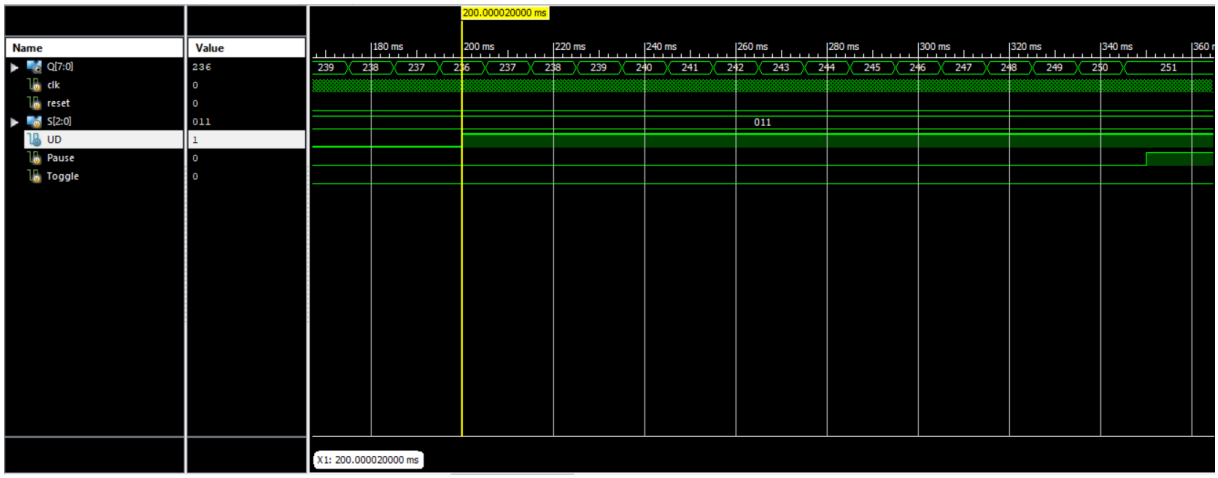
```
module test;
	reg clk;
	reg reset;
	reg [2:0] S;
	reg UD;
	reg Pause;
	reg Toggle;
	wire [7:0] Q;
	top uut (
		.clk(clk),
		.reset(reset),
		.S(S),
		.UD(UD),
		.Pause(Pause),
		.Toggle(Toggle),
		.Q(Q)
	);

	initial begin
		clk = 0;
		reset = 0;
		S = 3;
		UD = 0;
		Pause = 0;
```

```
                Toggle = 0;
                #10;
                reset=1;
                #10;
                reset=0;
                #200000000;
                UD=1;
                #150000000;
                Pause=1;
                #100000000;
                Pause=0;
                #100000000;
                Pause=1;
                #100000000;
                Pause=0;
                S=4;
                Toggle = 1;
        #100;
                S=5;
                #1000;
        end
always
    begin
        #10  clk=~clk;
    end
endmodule
```

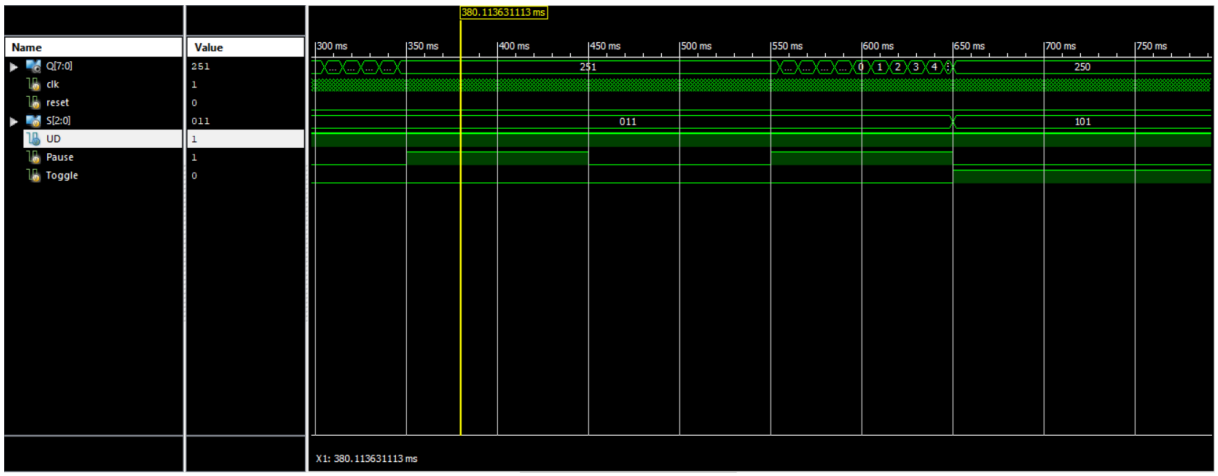
Kết quả



Hình 18.2. Thời điểm từ 0 ms đến hơn 180 ms



Hình 18.3. Thời điểm từ 170 ms đến hơn 360 ms



Hình 18.4. Thời điểm từ 300 ms đến hơn 750 ms

Nhận xét:

Chạy ban đầu với S = 3 tần số tương ứng 100 Hz, UD = 0 mạch đếm lên. Sau khi UD = 1 tại 200 ms mạch bắt đầu đếm xuống. Tại 650 ms với S = 4 và Toggle lên 1 mạch bắt đầu đếm với tần số 1 Khz và đảo trạng thái ngõ ra từ 5(00000101) sang 250(11111010).