TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH

KHOA ĐIỆN – ĐIỆN TỬ

BỘ MÔN KỸ THUẬT MÁY TÍNH – VIỄN THÔNG

# BÁO CÁO TIỂU LUẬN

# MÔN HỌC: THỰC TẬP THIẾT KẾ HỆ THỐNG SỐ VÀ VI MẠCH TÍCH HỢP

**GVHD:** LÊ MINH THÀNH

**SVTH:** VÕ PHƯỚC TIẾN

MSSV: 20161381

*TP.HCM, tháng 5 năm 2023*

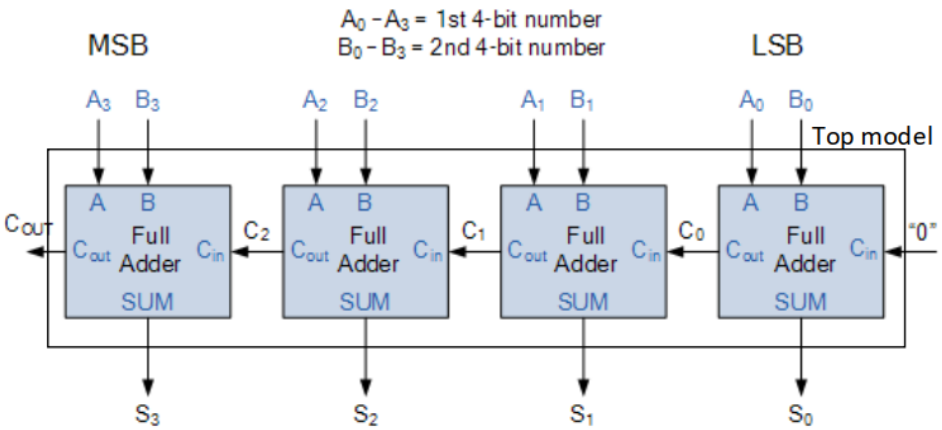
**MỤC LỤC**

1.Thiết kế mạch cộng ..... 3
1.1 Mạch cộng sử dụng mô hình cấu trúc ..... 3
1.2 Mạch cộng sử dụng toán hạng verilog ..... 7
2. Thiết kế mạch trừ ..... 11
2.1 Mạch trừ sử dụng mô hình cấu trúc ..... 11
2.2 Mạch trừ sử dụng toán hạng (Sub Operator) ..... 15
2.3 Mạch trừ sử dụng số bù 2 ..... 18
3 Mạch giải mã 2 sang 4 – Ngõ ra tích cực mức cao ..... 20
4 Mạch giải mã 2 sang 4 – Ngõ ra tích cực mức thấp ..... 24
5 Mạch giải mã 2 sang 4(Enable) – Ngõ ra tích cực mức cao ..... 28
6 Mạch giải mã 2 sang 4(Enable) – Ngõ ra tích cực mức thấp ..... 32
7 Mạch giải mã 3 sang 8 ..... 37
8 Mạch giải mã 3 sang 8 (ghép từ 2 mạch giải mã 2 sang 4) ..... 42
9 Mạch giải mã 3 sang 8 (Enable) – Ngõ ra tích cực mức thấp ..... 46
10 Mạch giải mã 3 sang 8 có chân Select ..... 51
11 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. ..... 56
12 Thiết kế mạch tạo xung 1hz. ..... 59
13 Thiết kế mạch tạo 4 xung ngõ ra với tần số lần lượt là 0.1Hz, 1Hz, 10Hz, 100Hz. ..... 61
14 Thiết kế mạch đếm đồng bộ, sử dụng phương pháp cài đặt các Flip – Flop. Xung đếm 1Hz được lấy từ mạch chia xung. ..... 65
15 Thiết kế mạch đếm lên 4 bit như bài 14, sử dụng phương pháp thiết kế đồng bộ. ..... 68
16 Thiết kế mạch đếm lên 8 bit, lựa chọn tần số đếm, lựa chọn đếm lên hoặc đếm xuống. ..... 72
17 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.. 78

## 1.Thiết kế mạch cộng

### 1.1 Mạch cộng 4 bit sử dụng mô hình cấu trúc

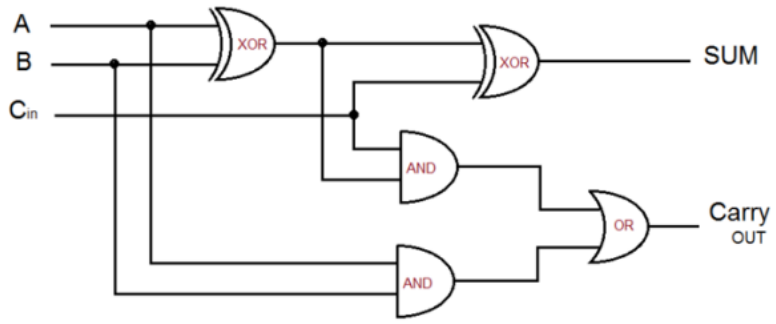
Sơ đồ khối




Trong sơ đồ trên là mạch cộng 4 bit có 4 ngõ vào $A_0$ đến $A_3$ , 4 ngõ vào $B_0$-$B_3$, 1 ngõ vào giá trị cờ nhớ $C_{in}$, 1 ngõ ra giá trị cờ nhớ $C_{out}$, 4 ngõ ra giá trị tổng Sum. Mô đun này được gọi là mô đun chính (Top module)

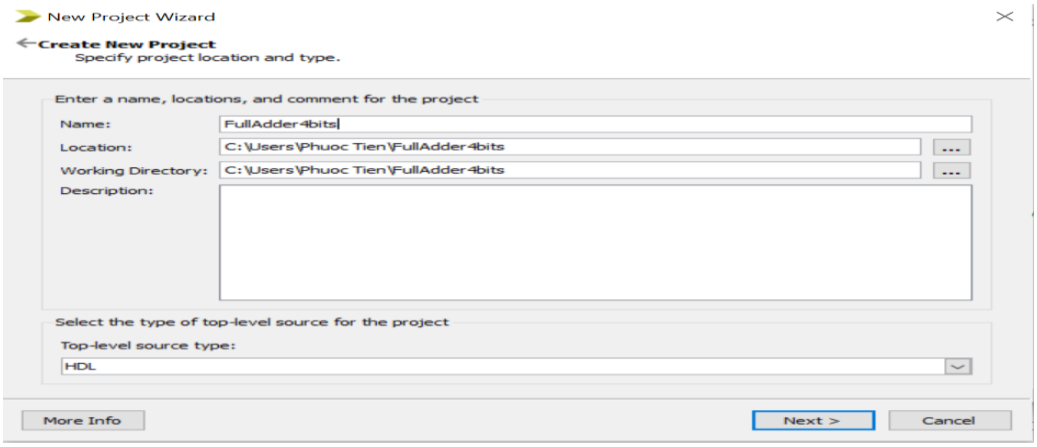
Bên trong các mô đun chính là các mô đun mạch cộng toàn phần 1 bit được thể hiện qua mạch logic bên dưới

*Mạch cộng toàn phần 1 bit




Tiến hành tạo Project mới cho phần thiết kế

Chọn File – New Project, đặt tên mô đun top là FullAdder4bits



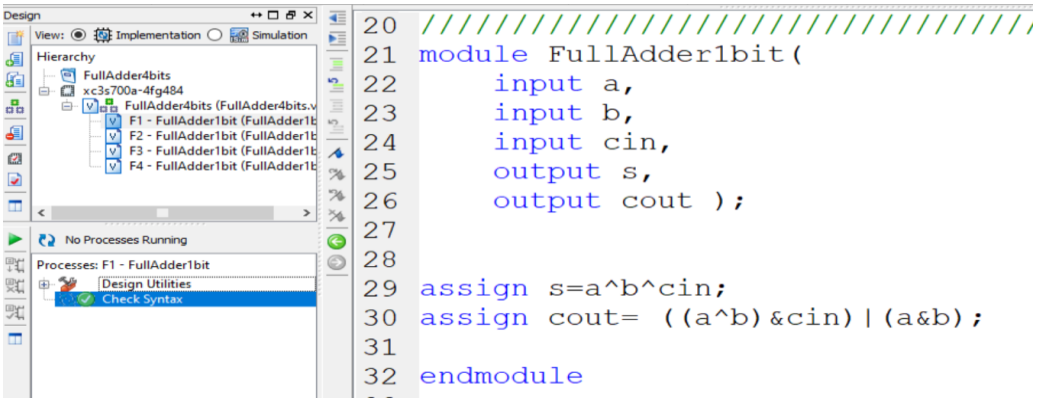

Trước khi tiến hành cài đặt cho mạch cộng 4bit ta sẽ tiến hành cài đặt cho mạch cộng toàn phần 1 bit

Tạo 1 project mới có tên FullAdder1bit

Mô tả mạch cộng toàn phần 1 bit như sau:

```
module FullAdder1bit(
    input a,
    input b,
    input cin,
    output s,
    output cout );


assign s=a^b^cin;
assign cout= ((a^b)&cin)|(a&b);

endmodule
```

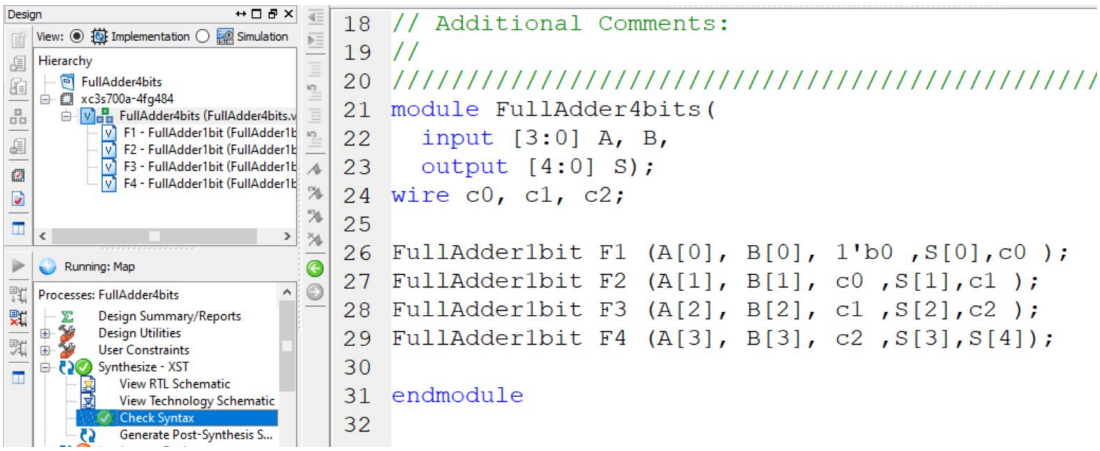
Tiến hành kiểm tra lỗi



Sau khi thiết lập cho mô đun phụ ta sẽ quay lại cài đặt cho mô đun chính ( FullAdder4bits) như sau

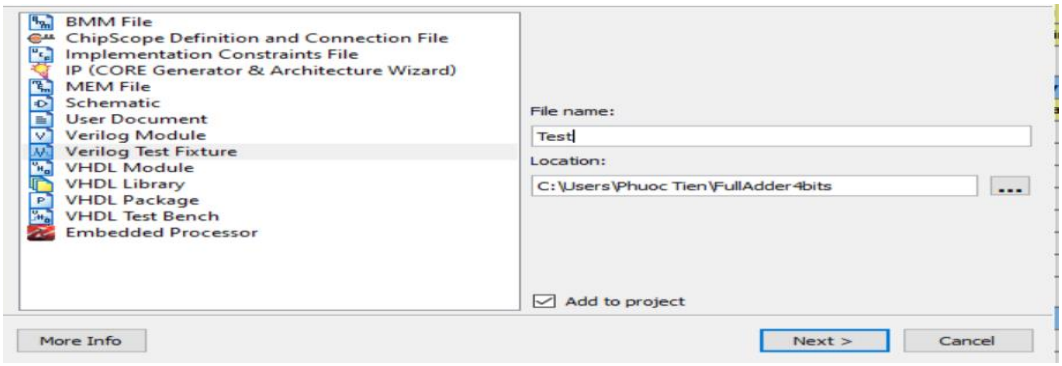
```
module FullAdder4bits(
  input [3:0] A, B,
  output [4:0] S);
wire c0, c1, c2;

FullAdder1bit F1 (A[0], B[0], 1'b0 ,S[0],c0 );
FullAdder1bit F2 (A[1], B[1], c0 ,S[1],c1 );
FullAdder1bit F3 (A[2], B[2], c1 ,S[2],c2 );
FullAdder1bit F4 (A[3], B[3], c2 ,S[3],S[4]);

endmodule
```

Tiến hành kiểm tra lỗi:



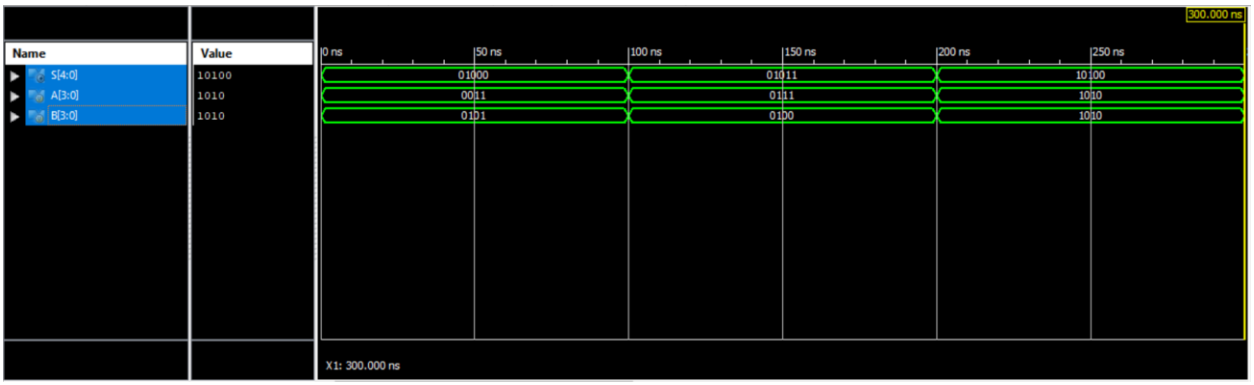

Sau khi hoàn tất cài đặt cho các mô đun ta sẽ tiến hành mô phỏng bằng cách tạo 1 Testbench

Chọn Project → chọn New Source → chọn Verilog Test Fixture



```
module Testbench;

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

   // Outputs
   wire [4:0] S;

   // Instantiate the Unit Under Test (UUT)
   FullAdder4bits uut (
      .A(A),
      .B(B),
      .S(S)
   );
   initial begin
      // Initialize Inputs
      A = 3;
      B = 5;
      // Wait 100 ns for global reset to finish
      #100;


      A = 7;
      B = 4;
      // Wait 100 ns for global reset to finish
      #100;

      A = 10;
      B = 10;
      // Wait 100 ns for global reset to finish
      #100;

      // Add stimulus here
$finish;
   end


endmodule
```

Kết quả mô phỏng được sau khi kiểm tra lỗi Testbench



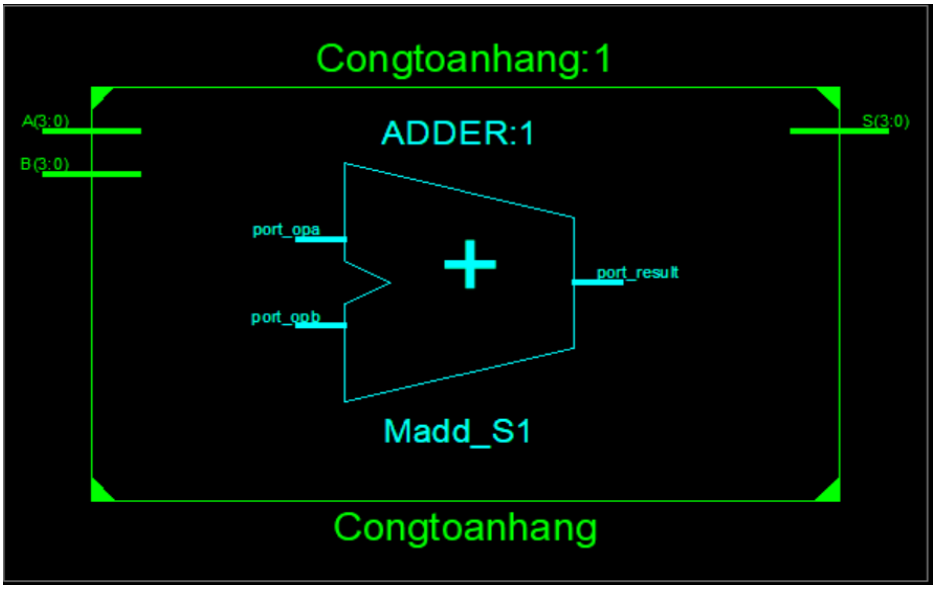

Giải thích: Tại TH1: A = 3(0011), B = 5(0101) => có S = 8 tương ứng 01000

Tại TH2: A = 4(0111), B = 7(0100) => có S = 11 tương ứng 01011

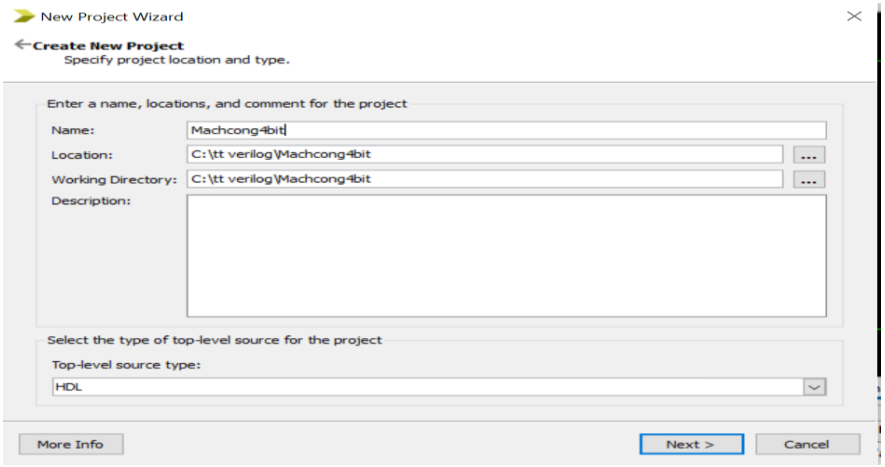
Tại TH3: A = 10(1010), B = 10(1010) => có S = 20 tương ứng 10100

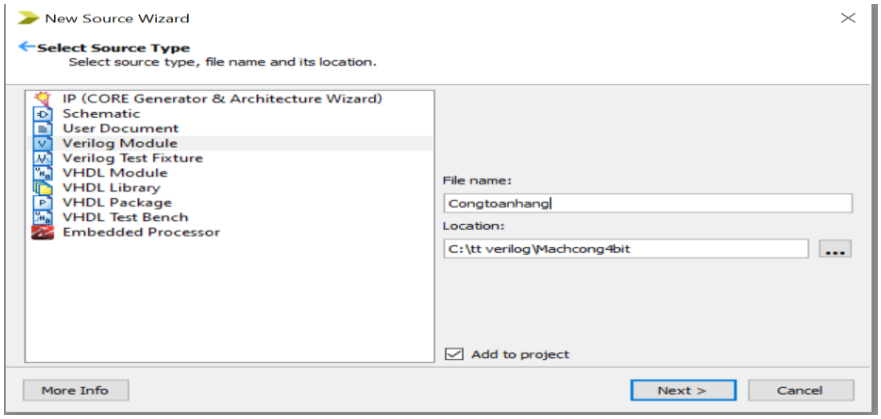
## 1.2 Mạch cộng sử dụng toán hạng verilog

Sơ đồ khối




Tiến hành tạo Project mới cho phần thiết kế

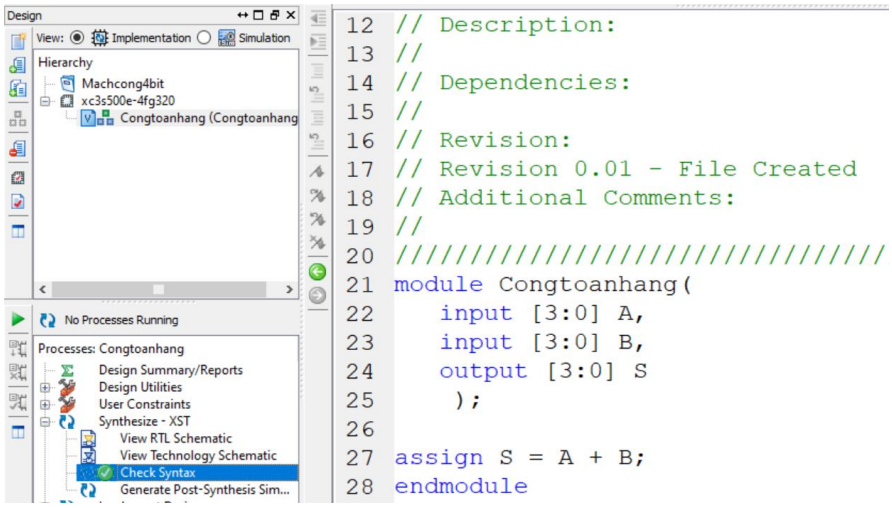




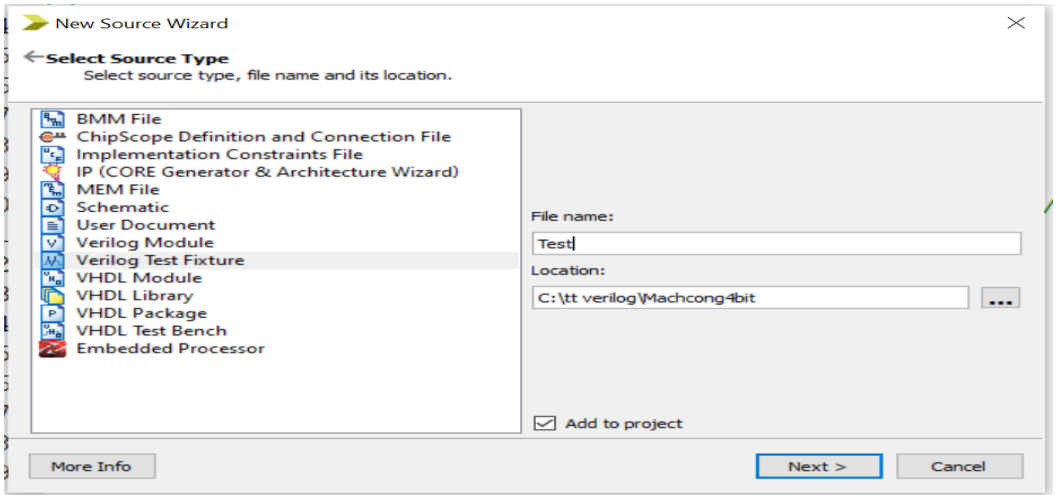

Thực hiện cài đặt cho mô đun như sau

```
module Congtoanhang(
    input [3:0] A,
    input [3:0] B,
    output [3:0] S
     );

assign S = A + B;
endmodule
```

Kiểm tra lỗi của mô đun vừa được cài đặt



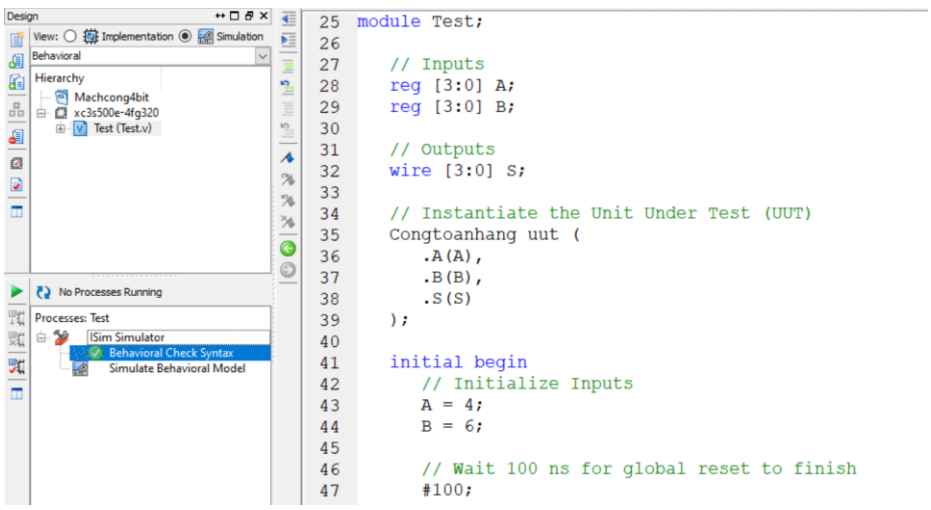

Tạo testbench để thực hiện mô phỏng



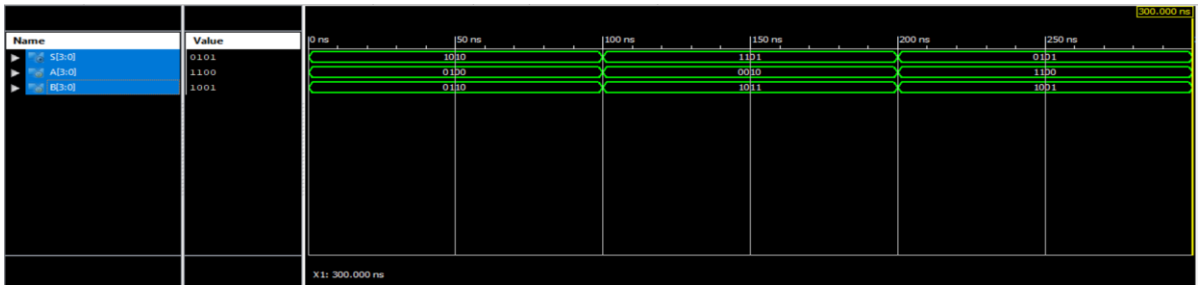

```
module Test;

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

   // Outputs
   wire [3:0] S;

   // Instantiate the Unit Under Test (UUT)
   Congtoanhang uut (
      .A(A),
      .B(B),
      .S(S)
   );

   initial begin
      // Initialize Inputs
      A = 4;
      B = 6;

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

```
        A = 2;
        B = 11;

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

        A = 12;
        B = 9;

        // Wait 100 ns for global reset to finish
        #100;
        // Add stimulus here
  $finish;
    end

endmodule
```

Thực hiện kiểm tra lỗi của testbench vừa tạo




Tiến hành mô phỏng mạch




Giải thích

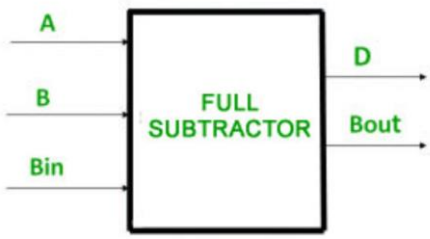
- Trường hợp 1: A=4 (0100) và B=6 (0110) nên sẽ có tổng là 10 (1010)

- Trường hợp 2: A=2 (0010) và B=11 (1011) nên sẽ có tổng là 13 (1101)
- Trường hợp 3: A=12 (1100) và B=9 (1001) nên sẽ có tổng là 21(10101) . Nhưng do ngõ ra chỉ có 4 bit → bị tràn 1 bit → kết quả sẽ chỉ hiển thị 4 bit là 0101

## 2. Thiết kế mạch trừ

### 2.1 Mạch trừ toàn phần sử dụng mô hình cấu trúc

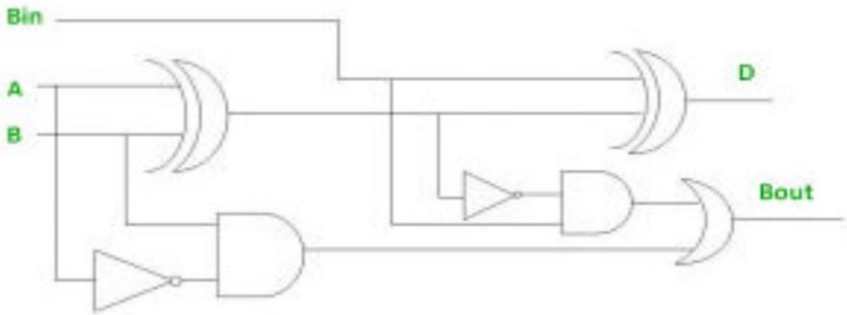
Sơ đồ khối




Bảng trạng thái

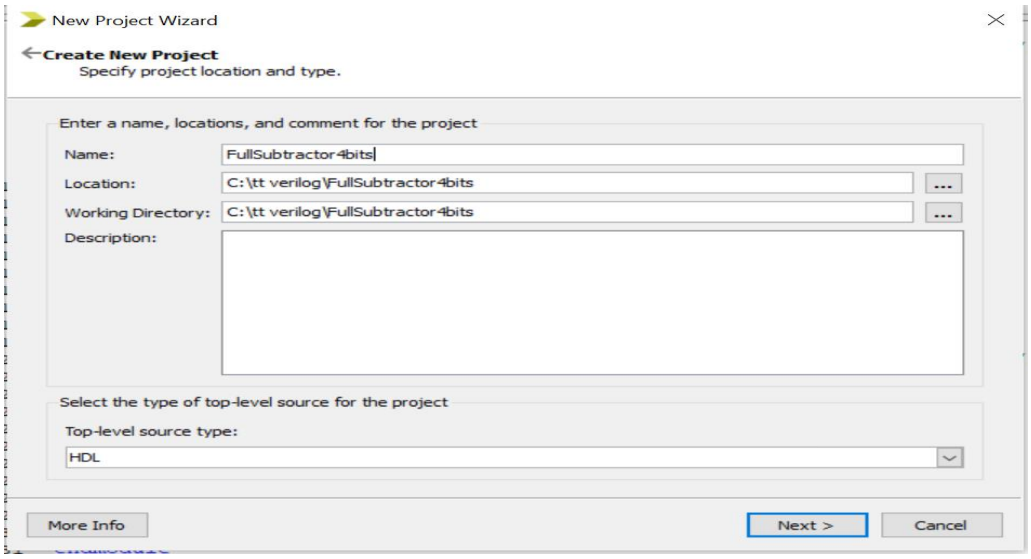
| INPUT | | | OUTPUT | |
|---|---|---|---|---|
| A | B | Bin | D | Bout |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

Mạch logic

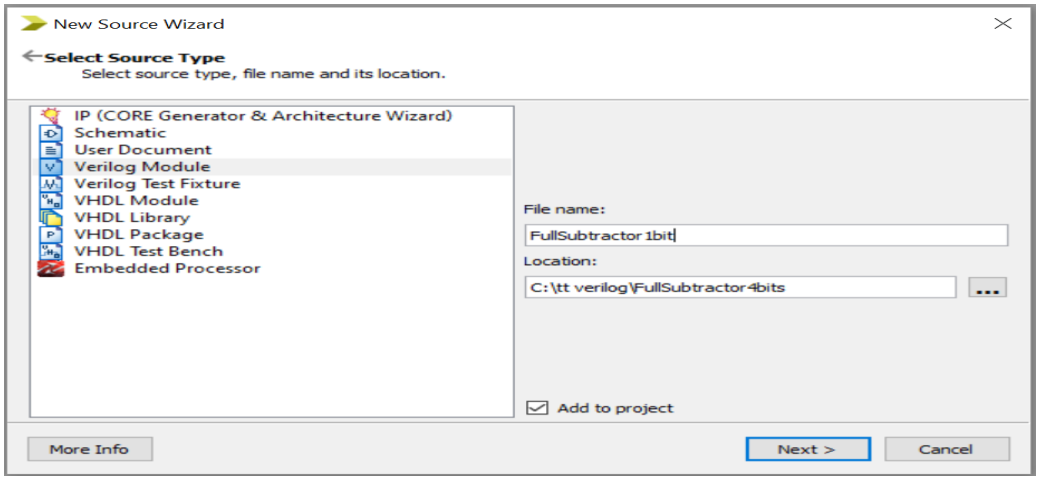



Tạo Project mới cho thiết kế

Chọn File – New Project, đặt tên mô đun top là FullSubtractor4bits



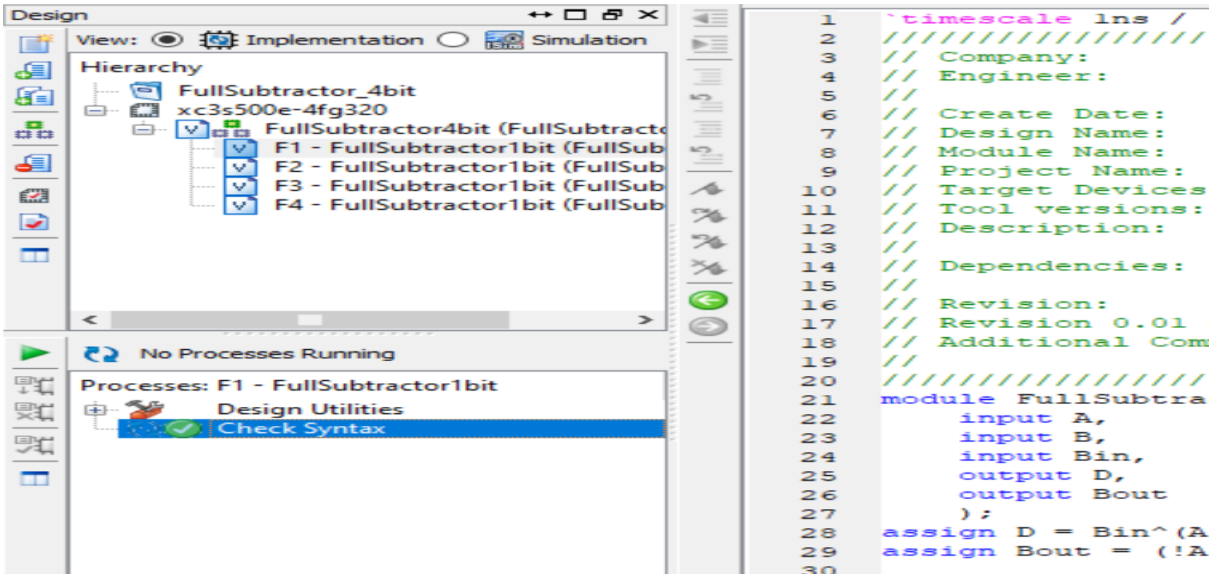

Thiết kế mô đun mạch trừ 1 bit




Mô tả mạch trừ toàn phần 1bit sử dụng mô hình cấu trúc như sau:

```
module FullSubtractor1bit(
    input A,
    input B,
    input Bin,
    output D,
    output Bout
    );
assign D = Bin^(A^B);
assign Bout = (!A&B)+((!A)&Bin)+(B&Bin);

endmodule
```

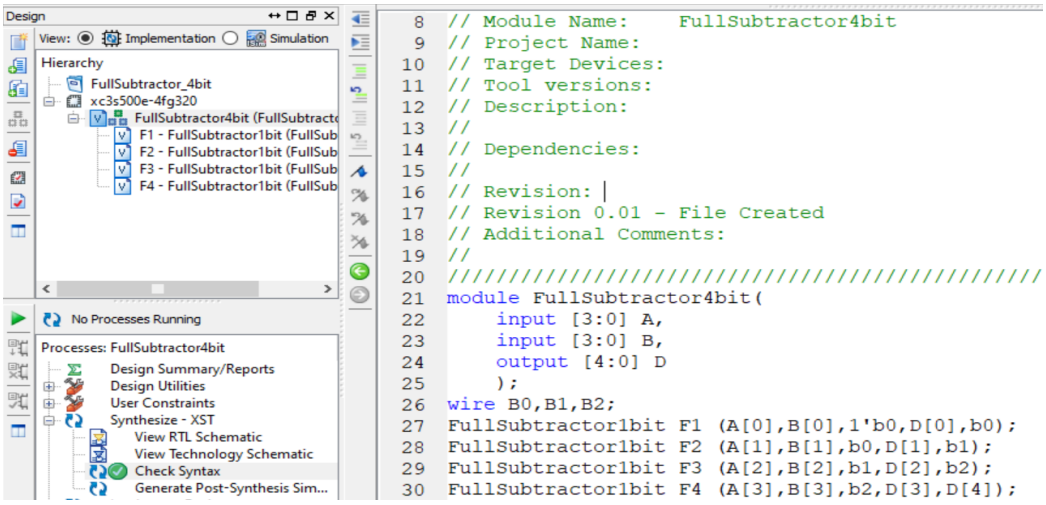
Tiến hành kiểm tra lỗi



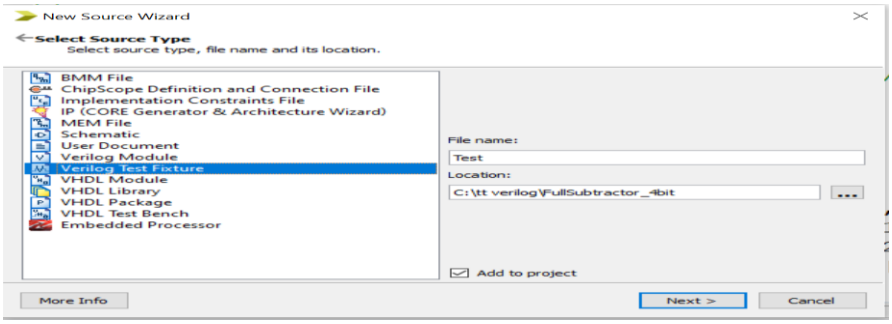

Quay lại cài đặt cho mô đun chính

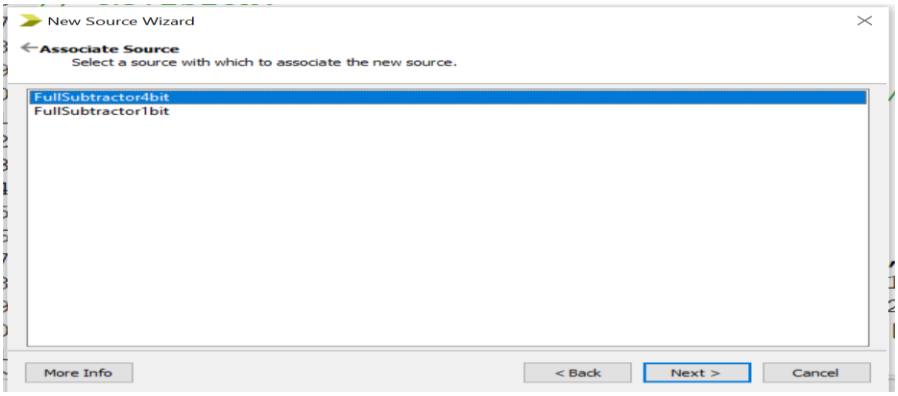
```
module FullSubtractor4bit(
    input [3:0] A,
    input [3:0] B,
    output [4:0] D
    );
wire B0,B1,B2;
FullSubtractor1bit F1 (A[0],B[0],1'b0,D[0],b0);
FullSubtractor1bit F2 (A[1],B[1],b0,D[1],b1);
FullSubtractor1bit F3 (A[2],B[2],b1,D[2],b2);
FullSubtractor1bit F4 (A[3],B[3],b2,D[3],D[4]);

endmodule
```

Tiến hành kiểm tra lỗi cho mô đun chính



Sau đó tạo testbench để mô phỏng mạch trừ toàn phần

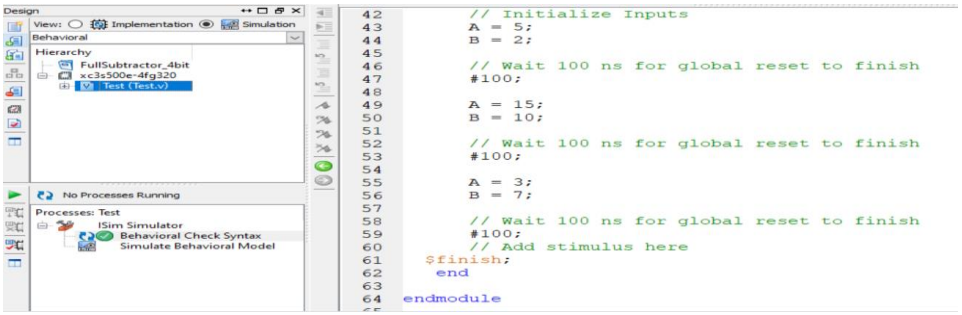


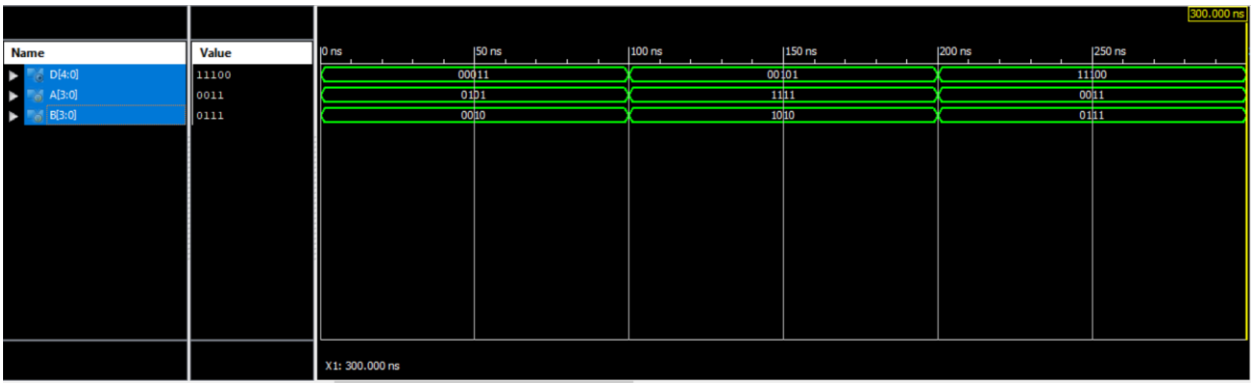

Tạo các tín hiệu để kiểm tra mạch trừ, giả sử trọng trường hợp này chúng ta tạo

```
module Test;

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

    // Outputs
    wire [4:0] D;

    // Instantiate the Unit Under Test (UUT)
    FullSubtractor4bit uut (
        .A(A),
        .B(B),
        .D(D)
    );

    initial begin
        // Initialize Inputs
        A = 5;
        B = 2;

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


        A = 15;
        B = 10;

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

        A = 3;
        B = 7;

        // Wait 100 ns for global reset to finish
        #100;
        // Add stimulus here
   $finish;
    end

endmodule
```

Kiểm tra lỗi cho testbench




Kết quả mô phỏng mạch trừ




Giải thích

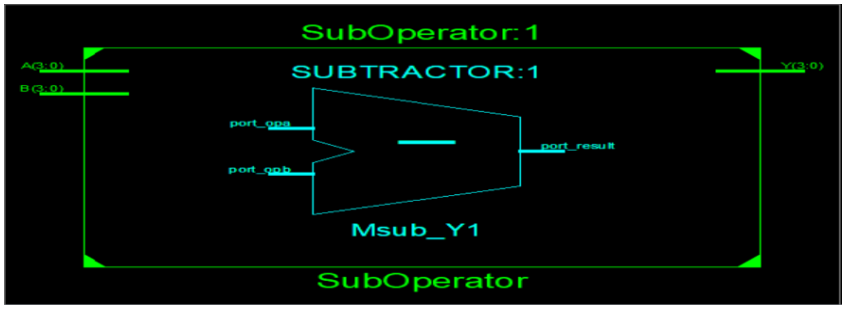
TH1: A = 5(0101), B = 2(0010) sẽ có D = 3 tương ứng 00011

TH2: A = 15(1111), B = 10(1010) sẽ có D = 5 tương ứng 00101

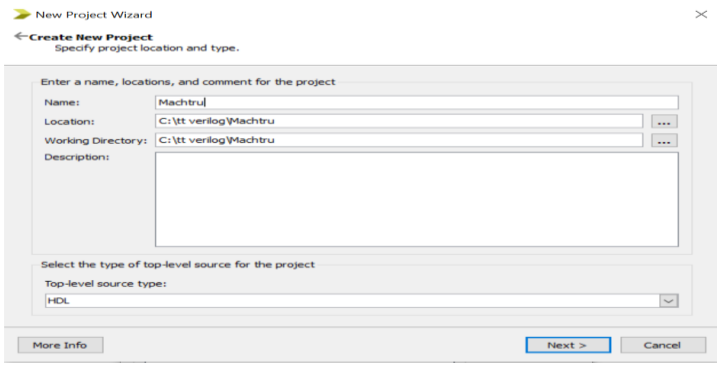
TH3: A = 3(0011), B = 7(0111) sẽ có D = 11100. Sau đó bỏ đi bit tràn, lấy bù 2 của 1100 rồi them dấu – phía trước sẽ cho ra kết quả cuối cùng là - (0100) = - 4.

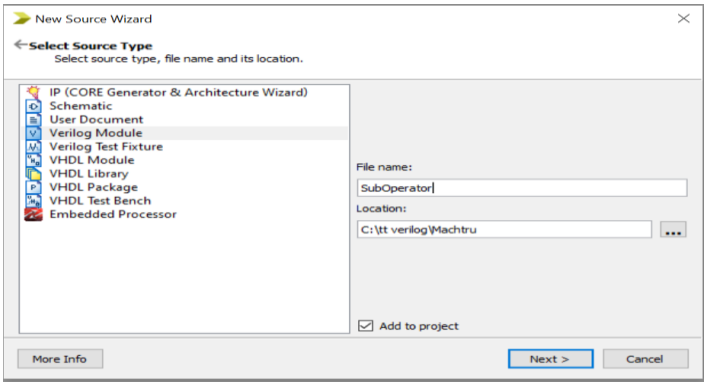
## 2.2 Mạch trừ sử dụng toán hạng ( Sub Operator)

Sơ đồ khối




Tạo projcet mới cho thiết kế

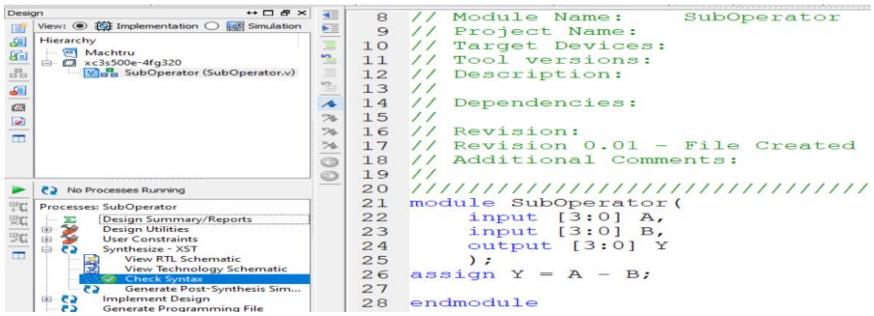


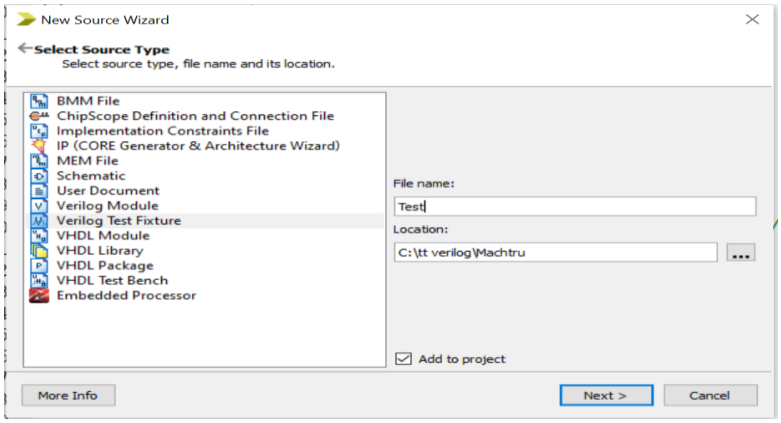

Mô tả mạch trừ (Sub operator) như sau

```
''''''''''''''''''''''''''''''''''''
module SubOperator(
    input [3:0] A,
    input [3:0] B,
    output [3:0] Y
    );
assign Y = A - B;

endmodule
```

Kiểm tra lỗi thiết kế



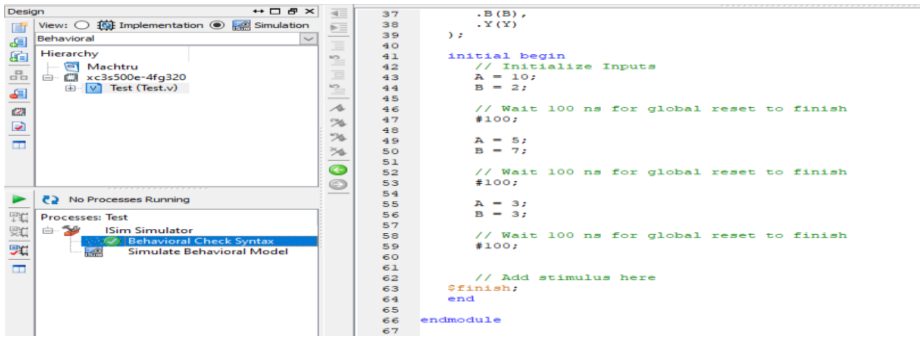

Sau khi hoàn tất kiểm tra lỗi ta tiến hành tạo file testbench để chạy mô phỏng



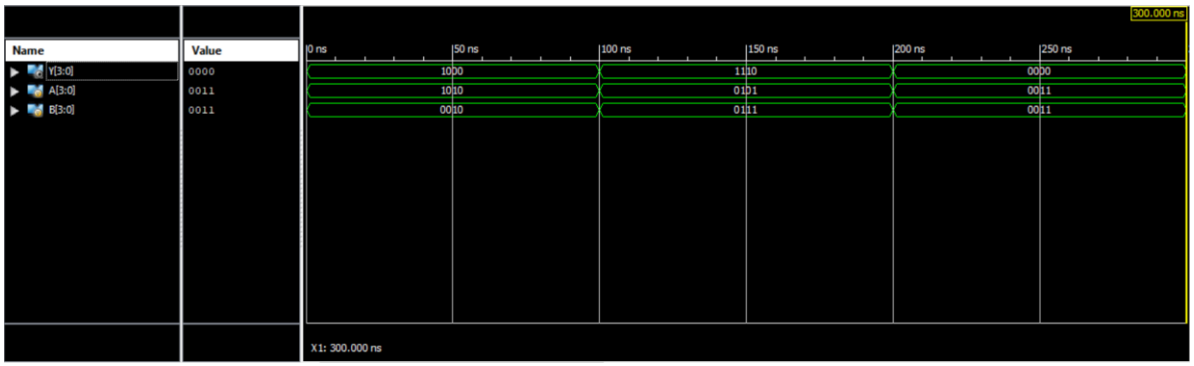

Cài đặt các thông số đầu vào cho testbench

```
module Test;

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

   // Outputs
   wire [3:0] Y;

   // Instantiate the Unit Under Test (UUT)
   SubOperator uut (
      .A(A),
      .B(B),
      .Y(Y)
   );

   initial begin
      // Initialize Inputs
      A = 10;
      B = 2;

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

      A = 5;
      B = 7;

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


      A = 3;
      B = 3;

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


      // Add stimulus here
   $finish;
   end

endmodule
```

Kiểm tra lỗi cho testbench



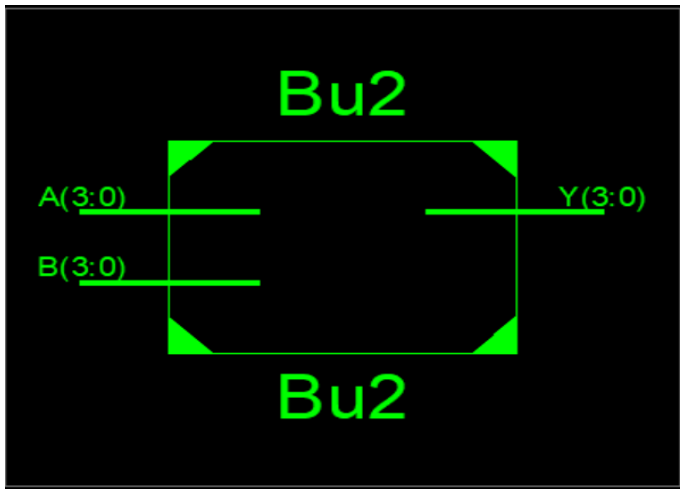

Bên dưới là kết quả sau khi chạy mô phỏng

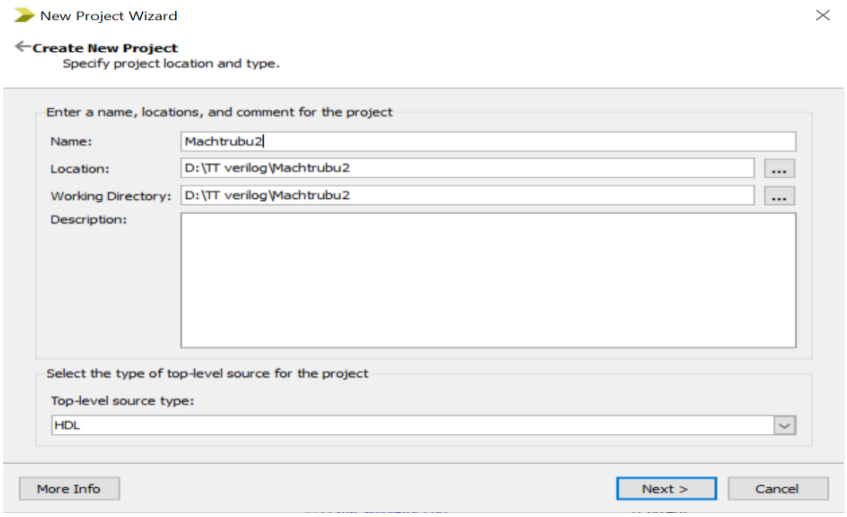



Giải thích: Ở trường hợp A = 5(0101), B = 7(0111) do A<B nên khi trừ sẽ có dư thêm 1 bít ( bit tràn ) do đầu ra chỉ có 4bit nên kết quả là 1110. Sau khi lấy bù 2 của 1110 là – (0010) = -2.

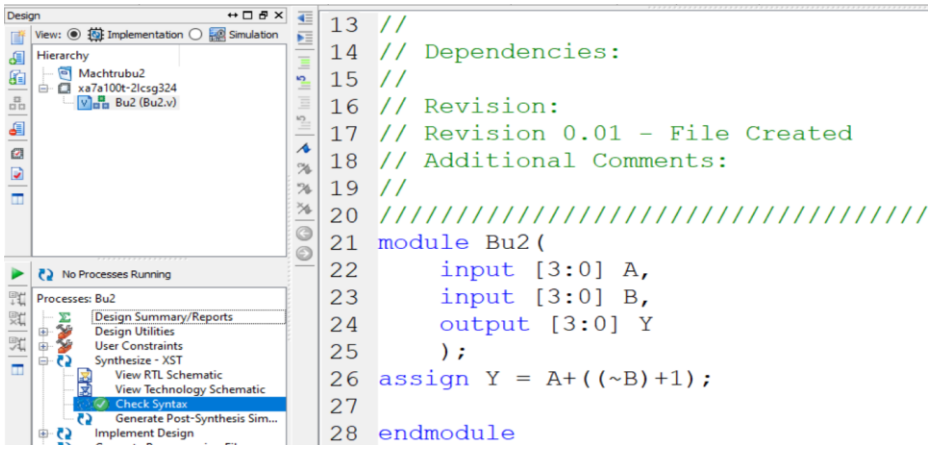
## 2.3 Mạch trừ sử dụng số bù 2

Sơ đồ khối






Thực hiện cấu hình cho mạch trừ sử dụng bù 2

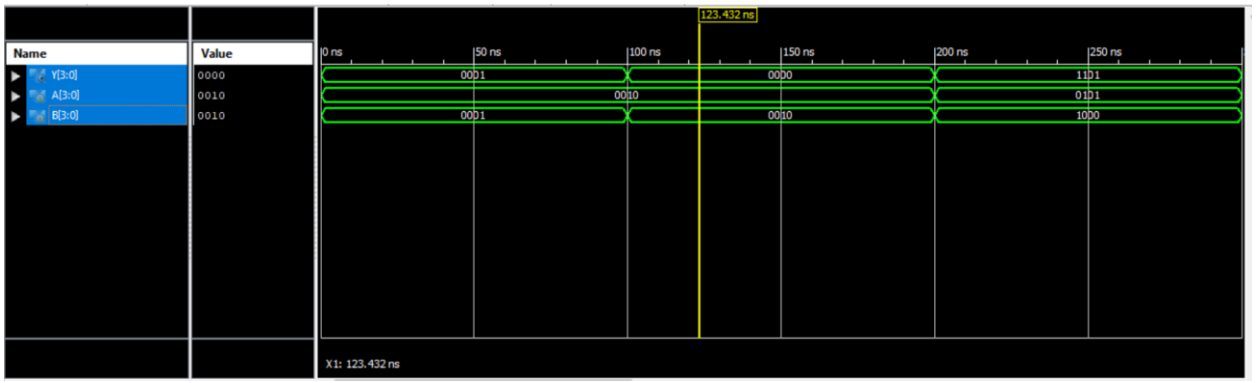



Thực hiện viết testbench

```
module Test;

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

    // Outputs
    wire [3:0] Y;

    // Instantiate the Unit Under Test (UUT)
    Bu2 uut (
        .A(A),
        .B(B),
        .Y(Y)
    );

    initial begin
        // Initialize Inputs
        A = 2;
        B = 1;
        // Wait 100 ns for global reset to finish
        #100;
```

```
        A = 2;
        B = 2;
        // Wait 100 ns for global reset to finish
        #100;

        A = 5;
        B = 8;
        // Wait 100 ns for global reset to finish
        #100;
        // Add stimulus here
      $finish;
    end

endmodule
```

Kết quả mô phỏng của mạch trừ sử dụng số bù 2




Giải thích:

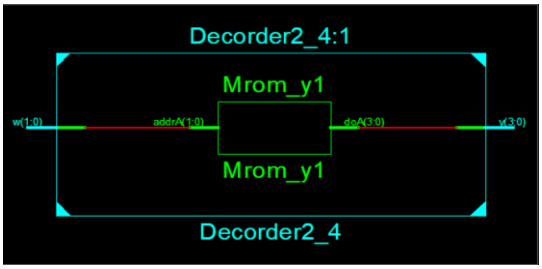
TH1: A = 0010, B = 0001 có bù 2 là 1111 => Y = A + bù 2 của B = 0001

TH2: A = 0010, B = 0010 có bù 2 là 1110 => Y = A + bù 2 của B = 0000

TH3: A = 0101, B = 1000 có bù 2 là 1000=> Y = A + bù 2 của B = 1101

## 3 Mạch giải mã 2 sang 4 – Ngõ ra tích cực mức cao

### 3.1 Sơ đồ khối




Hình 2.1.1 RTL Schematic

Sơ đồ khối Hình 2.1.1 là mạch giải mã từ 2 đường sang 4 đường có ngõ ra tích cực ở mức cao, có 1 ngõ vào là w(2bit) và 1 ngõ ra Y(4bit).

**3.2 Bảng trạng thái và Mạch Logic**

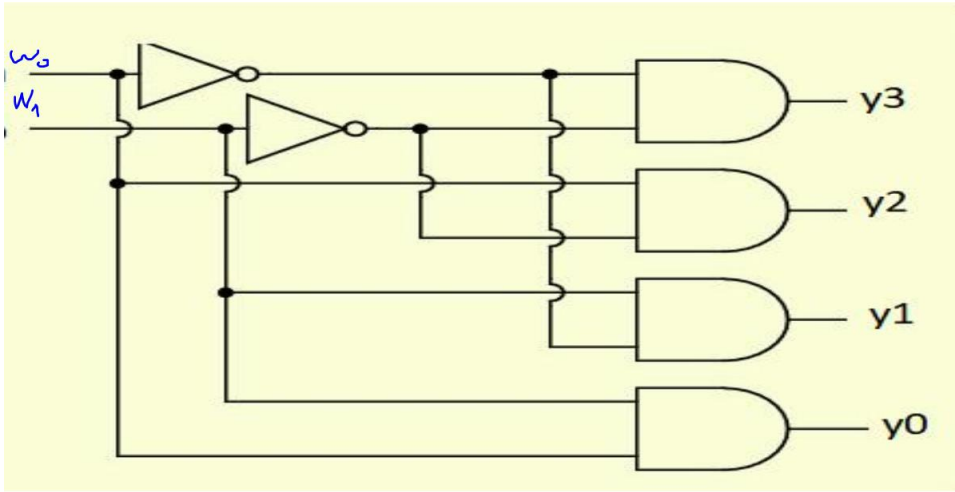
a) Bảng trạng thái

| $w_1$ | $w_0$ | $y_0$ | $y_1$ | $y_2$ | $y_3$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

Hình 2.1.2a

Ở bảng trạng thái Hình 2.1.2a mô tả các giá trị ngõ vào và ngõ ra. Khi ngõ vào có các giá trị từ 0→3 thì ngõ ra sẽ có các bít 1 dịch từ $Y_0$ đến $Y_3$ tại các vị trí còn lại đều có giá trị là 0.

b) Mạch logic




Hình 2.1.2b

**3.3 Mô tả mạch decoder 2_4 ngõ ra tích cực mức cao**

module Decoder2_4(

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

Giải thích Code

Khai báo ngõ vào w là wire độ dài 2 bit, ngõ ra y là 1 thanh ghi dịch có độ dài 4 bit là reg. Thực hiện khai báo lệnh always@() cho ngõ vào w tức là khi w thay đổi thì chương trình sẽ bắt đầu các lệnh tiếp theo.

Tiến hành kiểm tra lỗi

Hình 2.1.3

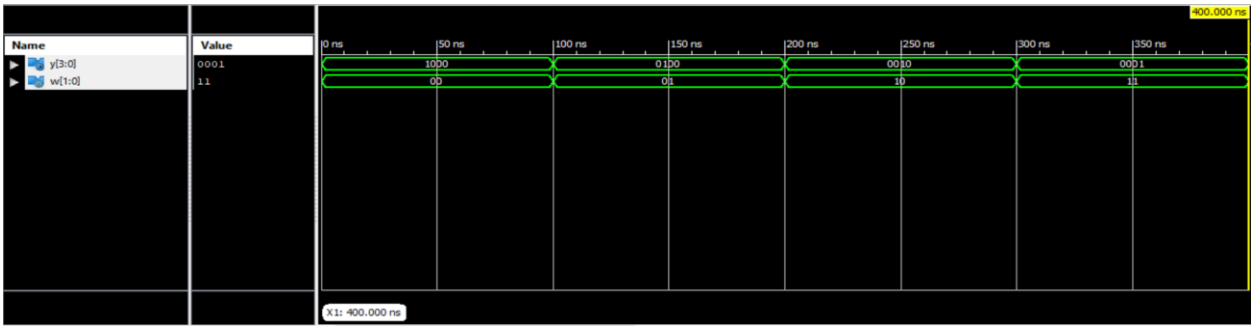
**3.4 Testbench và kết quả mô phỏng**

a) Testbench

```
initial begin
        // Initialize Inputs
        w = 0;
        // Wait 100 ns for global reset to finish
        #100;
        w = 1;
```

```
        // Wait 100 ns for global reset to finish
        #100;
        w = 2;
        // Wait 100 ns for global reset to finish
        #100;
        w = 3;
        // Wait 100 ns for global reset to finish
        #100
        // Add stimulus here
    end
endmodule
```

b) Kết quả mô phỏng




Hình 2.1.4

**Nhận xét Hình 2.1.4**

- Khi t đi từ 0ns → 100ns ngõ vào có giá trị là 0 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_0$ là 1000.

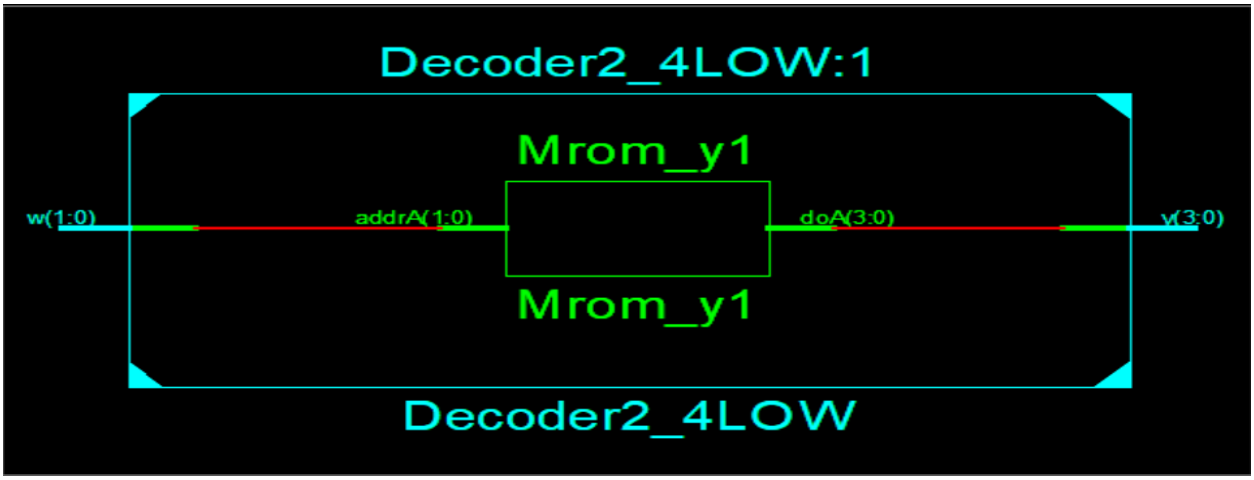
- Khi t đi từ 110ns → 200ns ngõ vào có giá trị là 1 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_1$ là 0100.

- Khi t đi từ 200ns → 300ns ngõ vào có giá trị là 0 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_2$ là 0010.

- Khi t đi từ 300ns → 400ns ngõ vào có giá trị là 0 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_3$ là 0001.

## 4 Mạch giải mã 2 sang 4 – Ngõ ra tích cực mức thấp

### 4.1 Sơ đồ khối




Hình 2.2.1 RTL Schematic

Trong sơ đồ khối Hình 2.2.1 là mạch giải mã từ 2 đường sang 4 đường có ngõ ra tích cực ở mức thấp, có 1 ngõ vào là w(2bit) và 1 ngõ ra Y(4bit).

### 4.2 Bảng trạng thái và mạch logic

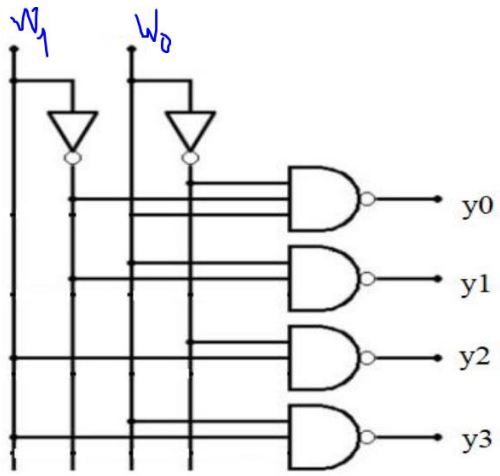
a) Bảng trạng thái

| $W_1$ | $W_0$ | $Y_0$ | $Y_1$ | $Y_2$ | $Y_3$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

Hình 2.2.2a

Ở bảng trạng thái Hình 2.2.2a mô tả các giá trị ngõ vào và ngõ ra. Khi ngõ vào có các giá trị từ 0→3 thì ngõ ra sẽ có các bit 0 dịch từ $Y_0$ đến $Y_3$ tại các vị trí còn lại đều có giá trị là 1.

b) Mạch logic




Hình 2.2.2b

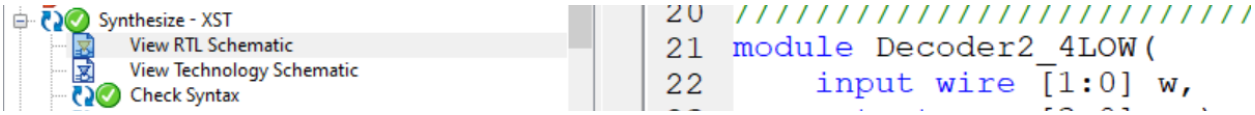
**4.3 Mô tả mạch decoder 2_4 Tích cực mức thấp**

```
module Decoder2_4LOW(
   input wire [1:0] w,
   output reg [3:0] y );
always @(w)
case(w)
0: y = 4'b0111;
1: y = 4'b1011;
2: y = 4'b1101;
3: y = 4'b1110;
endcase
endmodule
```

**Giải thích Code**

Khai báo ngõ vào w là wire độ dài 2 bit, ngõ ra y là 1 thanh ghi dịch có độ dài 4 bit là reg. Thực hiện khai báo lệnh always@() cho ngõ vào w tức là khi w thay đổi thì chương trình sẽ bắt đầu các lệnh tiếp theo.

**Tiến hành kiểm tra lỗi**




Hình 2.2.3

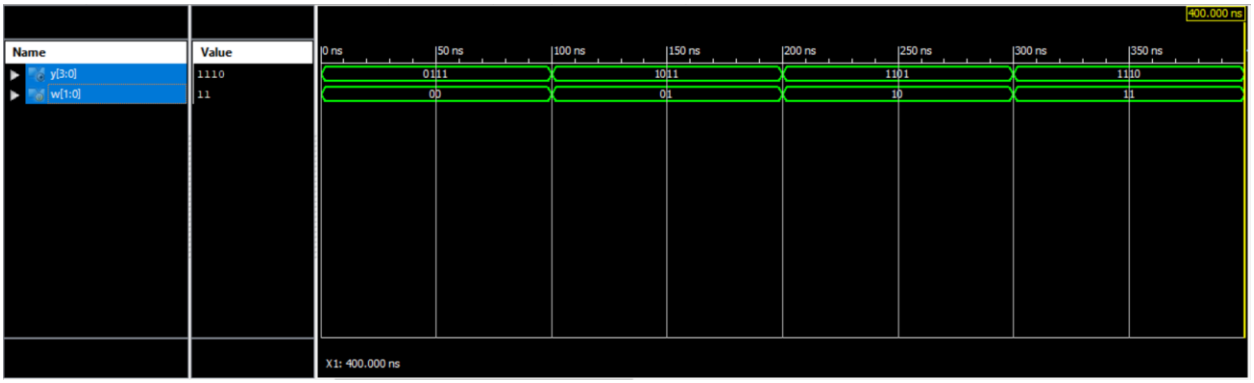
**4.4 Testbench và kết quả mô phỏng**

a) Testbench

```
module Test;
    // Inputs
    reg [1:0] w;
    // Outputs
    wire [3:0] y;
    // Instantiate the Unit Under Test (UUT)
    Decoder2_4LOW uut (
        .w(w),
        .y(y)
    );
    initial begin
        // Initialize Inputs
        w = 0;
        // Wait 100 ns for global reset to finish
        #100;
        w = 1;
        // Wait 100 ns for global reset to finish
        #100;
        w = 2;
        // Wait 100 ns for global reset to finish
        #100;
```

```
        w = 3;
        // Wait 100 ns for global reset to finish
        #100;
        // Add stimulus here
        $stop;
    end
endmodule
```

b) Kết quả mô phỏng




Hình 2.2.4

**Nhận xét Hình 2.2.4**

- Khi t đi từ 0ns → 100ns ngõ vào có giá trị là 0 thì ngõ ra có bit 0 được dịch vào tại vị trí $Y_0$ là 0111.

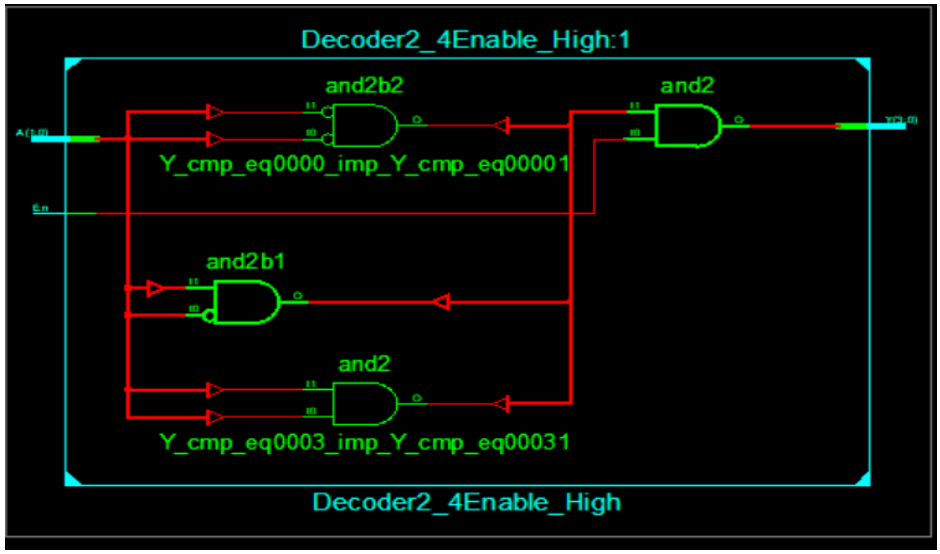
- Khi t đi từ 100ns → 200ns ngõ vào có giá trị là 1 thì ngõ ra có bit 0 được dịch vào tại vị trí $Y_1$ là 1011..

- Khi t đi từ 200ns → 300ns ngõ vào có giá trị là 2 thì ngõ ra có bit 0 được dịch vào tại vị trí $Y_2$ là 1101.

- Khi t đi từ 300ns → 400ns ngõ vào có giá trị là 3 thì ngõ ra có bit 0 được dịch vào tại vị trí $Y_3$ là 1110.

## 5 Mạch giải mã 2 sang 4(Enable) – Ngõ ra tích cực mức cao

### 5.1 Sơ đồ khối




Hình 2.3.1 RTL Schematic

Trong sơ đồ khối Hình 2.3.1 là mạch giải mã 2 đường sang 4 đường có 1 ngõ vào A(2bit) và 1 ngõ vào Enable tác động ở mức cao, 1 ngõ ra Y(8bit) xuất mức cao.

### 5.2 Bảng trạng thái và mạch logic

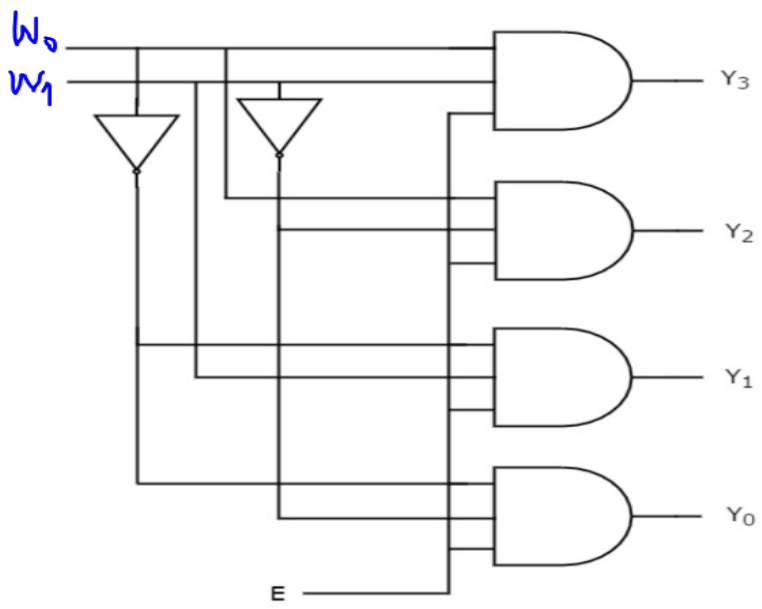
a) Bảng trạng thái

| Enable | Inputs | | Outputs | | | |
|---|---|---|---|---|---|---|
| E | $A_1$ | $A_0$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
| 0 | x | x | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |

Hình 2.3.2a

Ở bảng trạng thái Hình 2.3.2a mô tả các giá trị ngõ vào và ngõ ra. Khi ngõ vào có chân E=0 thì ngõ ra Y=0 bất chấp A. Khi E=1 thì ngõ vào A có các giá trị từ 0→3 tương ứng là ngõ ra có các bit 1 dịch từ $Y_0$ đến $Y_3$ và các vị trí còn lại của ngõ ra có giá trị là 0.

b) Mạch Logic




Hình 2.3.2b

**5.3 Mô tả mạch decoder 2_4 Enable – Ngõ ra tích cực mức cao**

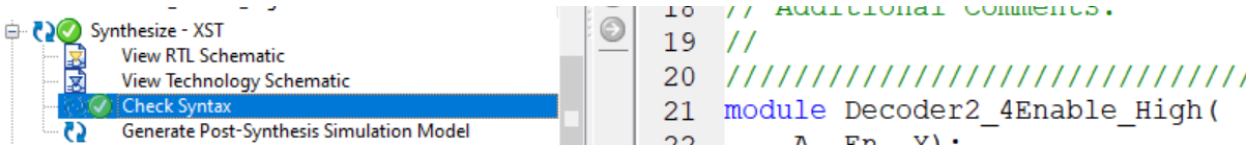
```
module Decoder2_4Enable_High(
   A, En, Y);
   input [1:0] A;
         input En;
         output [3:0] Y;
   reg [3:0] Y;
always @(A or En)
begin
if ( En == 1 )
 begin
 if( A==2'b00)
   Y=4'b0001;
 else if( A==2'b01)
   Y=4'b0010;
 else if( A==2'b10)
```

```
    Y=4'b0100;
 else if( A==2'b11)
    Y=4'b1000;
  end
else if ( En == 0)
  begin
  Y=4'b0000;
  end
endmodule
```

**Giải thích Code**

Khai báo input, output cho các đầu vào và ra. Thực hiện tạo khối always với điều kiện A hoặc En thay đổi sẽ thực hiện các lệnh bên trong và sử dụng lệnh If cho chương trình trên.

**Tiến hành kiểm tra lỗi**




Hình 2.3.3

**5.4 Testbench và mô phỏng**

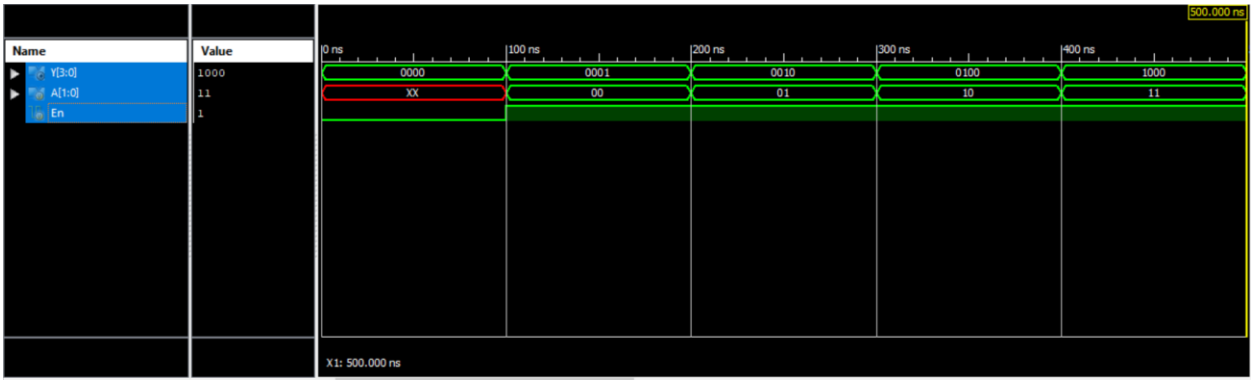
a) Testbench

```
module test;
        // Inputs
        reg [1:0] A;
        reg En;
        // Outputs
        wire [3:0] Y;
        // Instantiate the Unit Under Test (UUT)
```

```
Decoder2_4Enable_High uut (
        .A(A),
        .En(En),
        .Y(Y)
);
initial begin
        // Initialize Inputs
        En = 0;

        // Wait 100 ns for global reset to finish
        #100;
        A = 0;
        En = 1;
        // Wait 100 ns for global reset to finish
        #100;
        A = 1;
        En = 1;
        // Wait 100 ns for global reset to finish
        #100;
        A = 2;
        En = 1;
        // Wait 100 ns for global reset to finish
        #100;
        A = 3;
        En = 1;
        // Wait 100 ns for global reset to finish
```

```
            #100;
            // Add stimulus here
$stop;
        end
endmodule
```

b) Mô phỏng




Hình 2.3.4

**Nhận xét Hình 2.3.4**

- Khi t đi từ 0ns → 100ns chân Enable tác động mức 0 làm cho ngõ ra Y=0 bất chấp các ngõ vào A.

-Khi t đi từ 100ns→500ns chân Enable bắt đầu tác động ở mức 1 làm cho dữ liệu ngõ ra dịch các bit 1 theo các giá trị ngõ vào.

+ Khi t đi từ 100ns → 200ns ngõ vào có giá trị là 0 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_0$ là 0001..

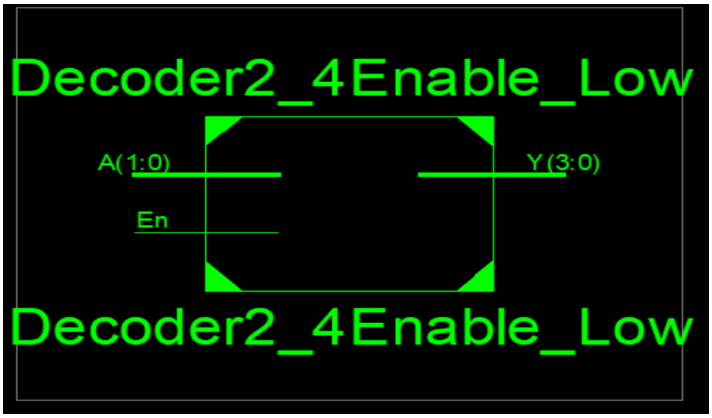
+ Khi t đi từ 200ns → 300ns ngõ vào có giá trị là 1 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_1$ là 0010.

+ Khi t đi từ 300ns → 400ns ngõ vào có giá trị là 2 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_2$ là 0100.

+ Khi t đi từ 400ns → 500ns ngõ vào có giá trị là 3 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_3$ là 1000

**6 Mạch giải mã 2 sang 4 (Enable) – Ngõ ra tích cực mức thấp**

**6.1 Sơ đồ khối**




Hình 2.4.1 RTL Schematic

Sơ đồ khối hình 2.4.1 mô tả các ngõ vào và ra của mạch giải mã 2 sang 4 có chân Enable tác động, ngõ ra tích cực mức cao.

**6.2 Bảng trạng thái và mạch logic**

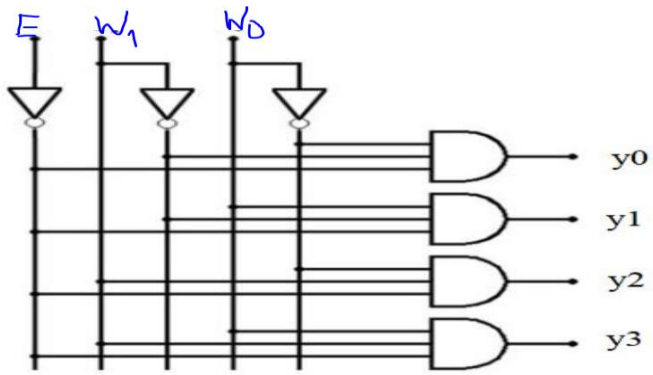
a) Bảng trạng thái

| En | Input | | Output | | | |
|---|---|---|---|---|---|---|
| | a | b | 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 2.4.2a

Nhìn vào bảng trạng thái Hình 2.4.2a ta thấy ngõ vào Enable tác động ở mức thấp. Khi En=1 thì ngõ ra sẽ là 1111 bất kể ngõ vào. Khi En = 0 thì ngõ ra sẽ có các bit 0 dịch từ $Y_0$ đến $Y_3$

b) Mạch logic




Hình 2.4.2b

**6.3 Mô tả mạch decoder 2_4 Enable – Tích cực mức thấp**

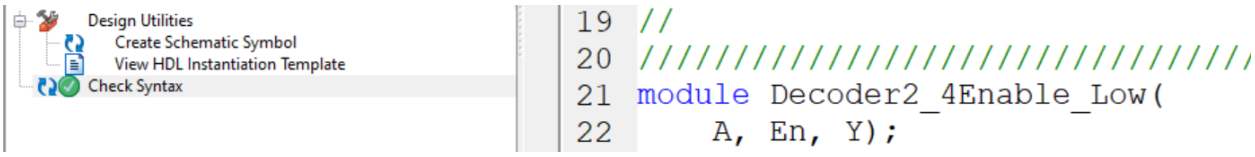
```
module Decoder2_4Enable_Low(
   A, En, Y);
   input [1:0] A;
         input En;
         output [3:0] Y;
   reg [3:0] Y;
always @(A or En)
begin
if ( En == 0 )
 begin
 if( A==2'b00)
   Y=4'b1110;
 else if( A==2'b01)
   Y=4'b1101;
 else if( A==2'b10)
   Y=4'b1011;
 else if( A==2'b11)
```

```
    Y=4'b0111;
  end
else if ( En == 1)
  begin
  Y=4'b1111;
  end
end
endmodule
```

**Giải thích Code**

Khai báo input, output cho các đầu vào và ra. Thực hiện tạo khối always với điều kiện A hoặc En thay đổi sẽ thực hiện các lệnh bên trong và sử dụng lệnh If cho chương trình trên.

**Tiến hành kiểm tra lỗi**




Hình 2.4.3

**6.4 Testbench và kết quả mô phỏng**

a) Testbench

```
module test2;
        // Inputs
        reg [1:0] A;
        reg En;
        // Outputs
        wire [3:0] Y;
        // Instantiate the Unit Under Test (UUT)
        Decoder2_4Enable_Low uut (
```

```
        .A(A),
        .En(En),
        .Y(Y)
);
initial begin
        // Initialize Inputs
        En = 0;
        // Wait 100 ns for global reset to finish
        #100;
        A = 0;
        En = 1;
        // Wait 100 ns for global reset to finish
        #100;
        A = 1;
        En = 1;
        // Wait 100 ns for global reset to finish
        #100;
        A = 2;
        En = 1;
        // Wait 100 ns for global reset to finish
        #100;
       A = 3;
        En = 1;
        // Wait 100 ns for global reset to finish
        #100;
        // Add stimulus here
```

```
    $stop;
        end
endmodule
```

b) Kết quả mô phỏng

Hình 2.4.4

**Nhận xét Hình 2.4.4**

- Khi t đi từ 0ns → 100ns chân Enable tác động mức 1 làm cho ngõ ra Y=1111 bất chấp các ngõ vào A.

-Khi t đi từ 100ns→500ns chân Enable bắt đầu tác động ở mức 0 làm cho dữ liệu ngõ ra dịch các bit 1 theo các giá trị ngõ vào.

+ Khi t đi từ 100ns → 200ns ngõ vào có giá trị là 0 thì ngõ ra có bit 0 được dịch vào tại vị trí $Y_0$ là 1110.

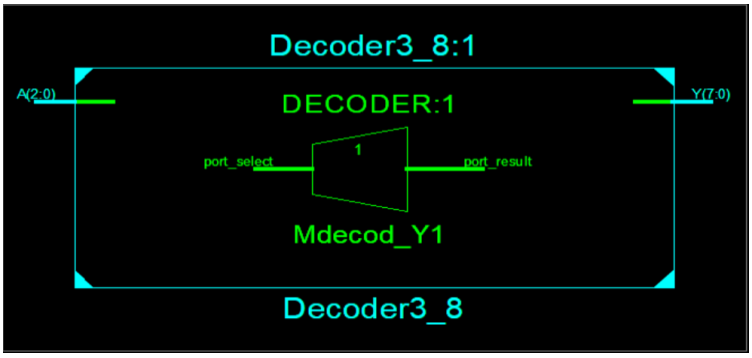
+ Khi t đi từ 200ns → 300ns ngõ vào có giá trị là 1 thì ngõ ra có bit 0 được dịch vào tại vị trí $Y_1$ là 1101.

+ Khi t đi từ 300ns → 400ns ngõ vào có giá trị là 2 thì ngõ ra có bit 0 được dịch vào tại vị trí $Y_2$ là 1011.

+ Khi t đi từ 400ns → 500ns ngõ vào có giá trị là 3 thì ngõ ra có bit 0 được dịch vào tại vị trí $Y_3$ là 0111.

## 7 Mạch giải mã 3 sang 8

7.1 Sơ đồ khối




Hình 2.5.1 RTL Schematic

Sơ đồ khối Hình 2.5.1 mô tả các ngõ vào và ra của mạch giải mã 3 sang 8 bao gồm 1 ngõ vào A(3bit) và 1 ngõ ra Y(8bit).

7.2 Bảng trạng thái

| $A_2$ | $A_1$ | $A_0$ | $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 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 2.5.2

Bảng trạng thái Hình 2.5.2 mô tả các mức giá trị của ngõ vào và ra của mạch giải mã 3 sang 8. Ngõ ra sẽ có các bit 1 được dịch từ $Y_0$ đến $Y_7$.

7.3 Mô tả mạch decoder 3_8 – ngõ ra tích cực mức cao

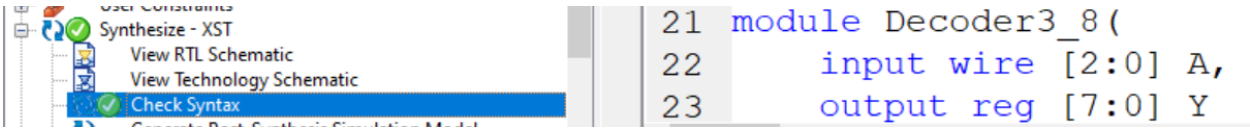
```
module Decoder3_8(
   input wire [2:0] A,
         output reg [7:0] Y
   );
always@(A)
 begin
  case(A)
         3'b000:  Y=8'b00000001;
          3'b001:  Y=8'b00000010;
```

```
        3'b010:  Y=8'b00000100;
        3'b011:  Y=8'b00001000;
        3'b100:  Y=8'b00010000;
        3'b101:  Y=8'b00100000;
        3'b110:  Y=8'b01000000;
        3'b111:  Y=8'b10000000;
  endcase
 end
endmodule
```

**Giải thích Code**

Khai báo ngõ vào có giá trị 3bit, ngõ ra 8bit với kiểu dữ liệu là reg. Sử dụng khối Always với điều kiện khi A thay đổi sẽ thực hiện tiếp tục các lệnh bên trong. Khi A = 0 đến A=7 tương ứng là các bít 1 được dịch từ $Y_0$ đến $Y_7$

**Tiến hành kiểm tra lỗi**




Hình 2.5.3

**7.4 Testbench và kết quả mô phỏng**

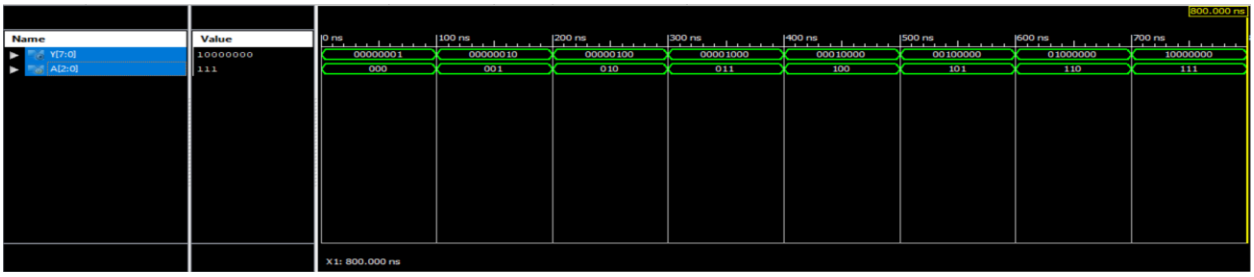
a) Testbench

```
module test;
        // Inputs
        reg [2:0] A;
        // Outputs
        wire [7:0] Y;
        // Instantiate the Unit Under Test (UUT)
        Decoder3_8 uut (
                .A(A),
```

```
        .Y(Y)
);
initial begin
        // Initialize Inputs
        A = 0;
        // Wait 100 ns for global reset to finish
        #100;
        A = 1;
        // Wait 100 ns for global reset to finish
        #100;
        A = 2;
        // Wait 100 ns for global reset to finish
        #100;
        A = 3;
        // Wait 100 ns for global reset to finish
        #100;
        A = 4;
        // Wait 100 ns for global reset to finish
        #100;
        A = 5;
        // Wait 100 ns for global reset to finish
        #100;
        A = 6;
        // Wait 100 ns for global reset to finish
        #100;
        A = 7;
```

```
            // Wait 100 ns for global reset to finish
            #100;
    $stop;
        end
endmodule
```

b) Kết quả mô phỏng




Hình 2.5.4

**Nhận xét kết quả mô phỏng Hình 2.5.4**

- Khi t đi từ 0ns → 100ns ngõ vào có giá trị là 0 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_0$ là 00000001.

- Khi t đi từ 100ns → 200ns ngõ vào có giá trị là 1 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_1$ là 00000010.

- Khi t đi từ 200ns → 300ns ngõ vào có giá trị là 2 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_2$ là 00000100.

- Khi t đi từ 300ns → 400ns ngõ vào có giá trị là 3 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_3$ là 00001000.

- Khi t đi từ 400ns → 500ns ngõ vào có giá trị là 4 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_0$ là 00010000.

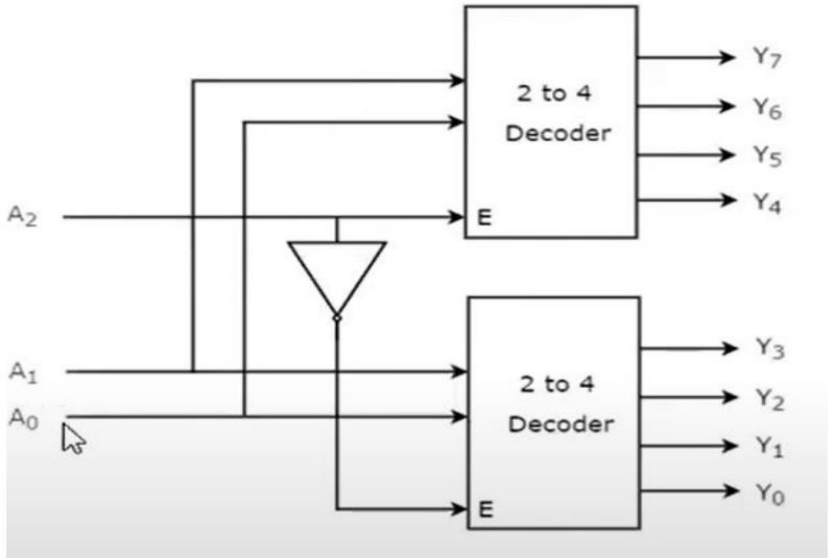
- Khi t đi từ 500ns → 600ns ngõ vào có giá trị là 5 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_1$ là 00100000.

- Khi t đi từ 600ns → 700ns ngõ vào có giá trị là 6 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_2$ là 01000000.

- Khi t đi từ 700ns → 800ns ngõ vào có giá trị là 7 thì ngõ ra có bit 0 được dịch vào tại vị trí $Y_3$ là 10000000.

**8 Mạch giải mã 3 sang 8 (sử dụng 2 mạch giải mã 2 sang 4)**

**8.1 Sơ đồ khối**




Hình 2.6.1

Sơ đồ khối Hình 2.6.1 mô tả các ngõ vào và ra của mạch giải mã 3 sang 8 sử dụng 2 bộ giải mã từ 2 sang 4

**8.2 Bảng trạng thái và Mạch logic**

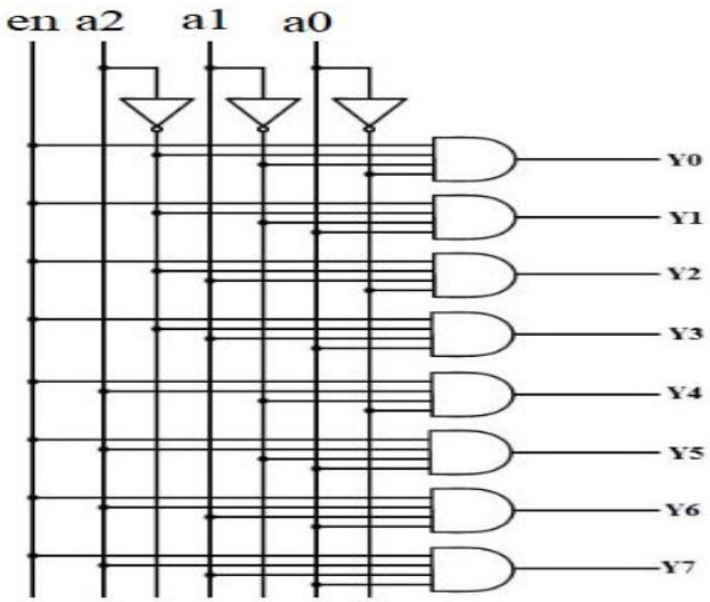
a) Bảng trạng thái

| Enable | INPUTS | | | Outputs | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| E | $A_2$ | $A_1$ | $A_0$ | $Y_7$ | $Y_6$ | $Y_5$ | $Y_4$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
| 0 | x | x | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Hình 2.6.2a

Bảng trạng thái Hình 2.6.2a mô tả các giá trị ngõ vào và ra của mạch giải mã 3 sang 8 từ bộ 2 bộ giải mã 2 sang 4 với chân En tác động mức cao.

b) Mạch logic




Hình 2.6.2b

**8.3 Mô tả mạch decoder 3 - 8 từ 2 bộ giải mã 2 sang 4 – ngõ ra tích cực mức cao**

a) Mô đun chính

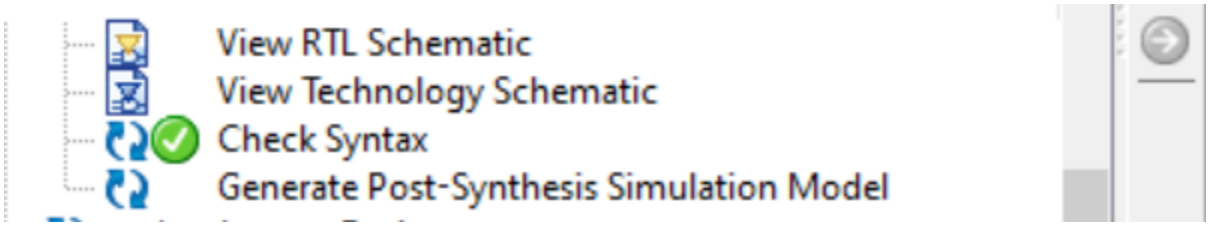
```
module decoder3to8(y,a
   );
            output [7:0]y;
            input [2:0]a;
            Decoder2x4 dl (y[7:4],a[1:0],a[2]);
            Decoder2x4 d2 (y[3:0],a[1:0],~a[2]);
Endmodule
```

b) Mô đun phụ ( decoder 2x4 )

```
module Decoder2x4(y,a,en);
```

```
output [3:0]y;
input [1:0]a;
input en;
reg [3:0]y;
        always@(a,en)
                if (en==0)
                        y=1'b0;
                else
                case (a)
                        2'b00: y=4'b0001;
                        2'b01: y=4'b0010;
                        2'b10: y=4'b0100;
                        2'b11: y=4'b1000;
                default:y=4'bxxxx;
                endcase
endmodule
```

**Tiến hành kiểm tra**




Hình 2.6.3

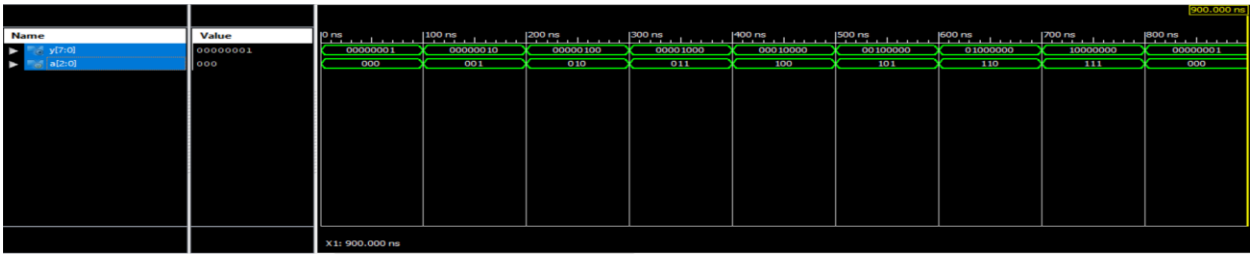
**8.4 Testbench và kết quả mô phỏng**

a) Testbench

```
module c;
```

```
        // Inputs
        reg [2:0] a;
        // Outputs
        wire [7:0] y;
        // Instantiate the Unit Under Test (UUT)
        decoder3to8 uut (
                .y(y),
                .a(a)
        );
        initial begin
                // Initialize Inputs
                        a = 0;
                #100;
                …
                        a = 8;
                #100;
                $stop;
                // Add stimulus here
        end
endmodule
```

b) Kết quả mô phỏng




Hình 2.6.4

**Nhận xét kết quả mô phỏng Hình 2.6.4**

- Khi t đi từ 0ns → 100ns ngõ vào có giá trị là 0 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_0$ là 00000001.

- Khi t đi từ 100ns → 200ns ngõ vào có giá trị là 1 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_1$ là 00000010.

- Khi t đi từ 200ns → 300ns ngõ vào có giá trị là 3 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_2$ là 00000100.

- Khi t đi từ 300ns → 400ns ngõ vào có giá trị là 4 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_3$ là 00001000.

- Khi t đi từ 400ns → 500ns ngõ vào có giá trị là 5 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_4$ là 00010000.

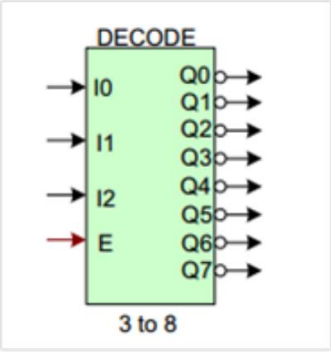
- Khi t đi từ 500ns → 600ns ngõ vào có giá trị là 6 thì ngõ ra có bit 1 được dịch vào tại vị trí $Y_5$ là 00100000.

- Khi t đi từ 600ns → 700ns ngõ vào có giá trị là 7 thì ngõ ra có bit 0 được dịch vào tại vị trí $Y_6$ là 01000000.

- Khi t đi từ 700ns → 800ns ngõ vào có giá trị là 7 thì ngõ ra có bit 0 được dịch vào tại vị trí $Y_7$là 10000000

**9 Mạch giải mã 3 sang 8 có chân Enable – Ngõ ra tích cực mức thấp**

**9.1 Sơ đồ khối**




Hình 2.7.1

Sơ đồ khối Hình 2.7.1 mô tả các ngõ vào I, En tác động mức cao, và có ngõ ra tích cực mức thấp.

**9.2 Bảng trạng thái và mạch logic**

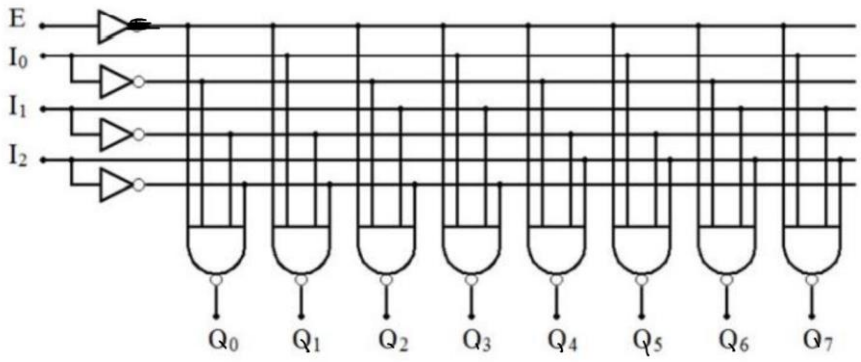
a) Bảng trạng thái

| Ngõ vào | | | | Ngõ ra | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| E | I2 | I1 | I0 | Q7 | Q6 | Q5 | Q4 | Q3 | Q2 | Q1 | Q0 |
| 0 | × | × | × | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Hình 2.7.2a

Bảng trạng thái hình 2.7.2a mô tả các giá trị ngõ vào và ra của mạch giải mã 3 sang 8 có chân Enable tác động mức cao thì ngõ ra sẽ các bit 0 được dịch từ $Q_0$ đến $Q_7$ tương ứng với các giá trị vào

b) Mạch logic




Hình 2.7.3b

**9.3 Mô tả mạch decoder 3_8 Enable – Ngõ ra tích cực mức thấp**

```
module Decoder3_8Enable( I, E, Q
    );
input [2:0] I;
input E;
output reg [7:0] Q;
always @(I or E)
begin
```

```
if(E==1)
  begin
  if ( I==3'b000)
    Q = 8'b11111110;
  else if(I==3'b001)
    Q = 8'b11111101;
  else if(I==3'b010)
    Q = 8'b11111011;
  else if(I==3'b011)
    Q = 8'b11110111;
  else if(I==3'b100)
    Q = 8'b11101111;
  else if(I==3'b101)
    Q = 8'b11011111;
  else if(I==3'b110)
    Q = 8'b10111111;
  else if(I==3'b111)
    Q = 8'b01111111;
   end
else if(E==0)
   begin
    Q = 8'b11111111;
            end
end
endmodule
```

**Giải thích Code**

Khai báo cho ngõ vào I(3bit), ngõ vào Enable, ngõ ra Q(8bit). Tạo lập khối Always với điều kiện I hoặc En thay đổi thì sẽ thực hiện các lệnh If bên trong

**Tiến hành kiểm tra lỗi**

Hình 2.7.3

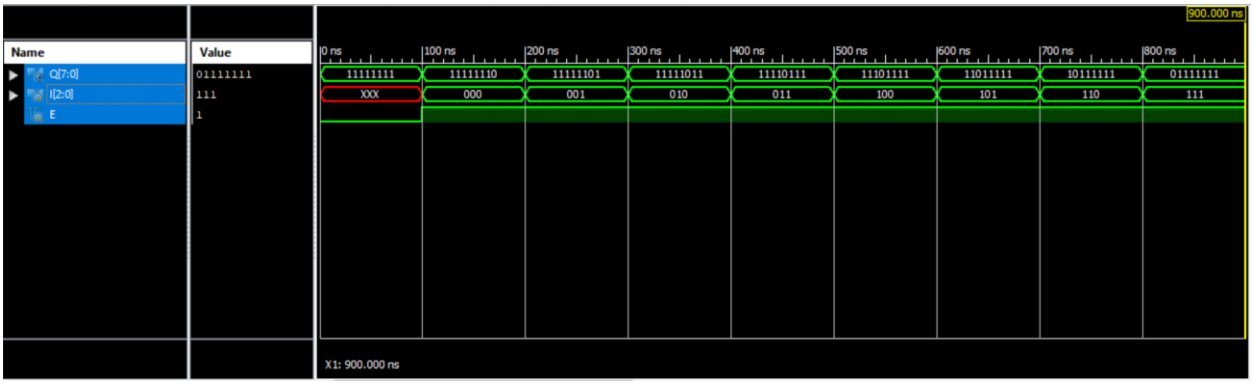
**9.4 Testbench và kết quả mô phỏng**

a) Testbench

```
module test;
    // Inputs
    reg [2:0] I;
    reg E;
    // Outputs
    wire [7:0] Q;
    // Instantiate the Unit Under Test (UUT)
    Decoder3_8Enable uut (
        .I(I),
        .E(E),
        .Q(Q)
    );
    initial begin
        // Initialize Inputs
        E = 0;
        // Wait 100 ns for global reset to finish
        #100;
        I = 0;
        E = 1;
```

```
        // Wait 100 ns for global reset to finish
        #100;
        I = 1;
        E = 1;
        // Wait 100 ns for global reset to finish
        #100;
        I = 7;
        E = 1;
        // Wait 100 ns for global reset to finish
        #100;
        // Add stimulus here
$stop;
    end
endmodule
```

b) Kết quả mô phỏng




Hình 2.7.4

**Nhận xét kết quả mô phỏng Hình 2.7.4**

Khi t đi từ 0ns → 100ns chân Enable tác động mức 0 làm cho ngõ ra Q=11111111 bất chấp các ngõ vào I

-Khi t đi từ 100ns→900ns chân Enable bắt đầu tác động ở mức 1 làm cho dữ liệu ngõ ra dịch các bit 0 theo các giá trị ngõ vào.

+ Khi t đi từ 100ns → 200ns ngõ vào có giá trị là 0 thì ngõ ra có bit 0 được dịch vào tại vị trí $Q_0$ là 11111110.

+ Khi t đi từ 200ns → 300ns ngõ vào có giá trị là 1 thì ngõ ra có bit 0 được dịch vào tại vị trí $Q_1$ là 11111101.

+ Khi t đi từ 300ns → 400ns ngõ vào có giá trị là 2 thì ngõ ra có bit 0 được dịch vào tại vị trí $Q_2$ là 11111011.

+ Khi t đi từ 400ns → 500ns ngõ vào có giá trị là 3 thì ngõ ra có bit 0 được dịch vào tại vị trí $Q_3$ là 11110111.

+ Khi t đi từ 500ns → 600ns ngõ vào có giá trị là 4 thì ngõ ra có bit 0 được dịch vào tại vị trí $Q_4$ là 11101111.

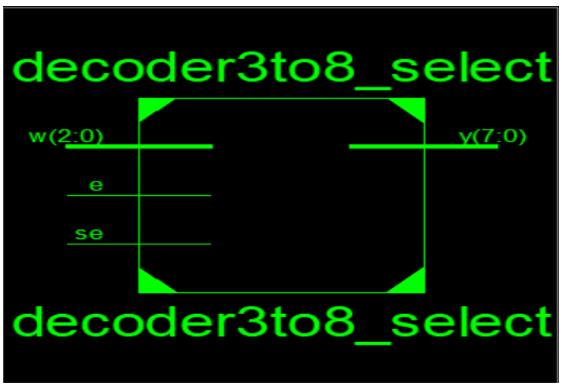
+ Khi t đi từ 600ns → 700ns ngõ vào có giá trị là 5 thì ngõ ra có bit 0 được dịch vào tại vị trí $Q_5$ là 11011111.

+ Khi t đi từ 700ns → 800ns ngõ vào có giá trị là 6 thì ngõ ra có bit 0 được dịch vào tại vị trí $Q_6$ là 10111111.

+ Khi t đi từ 800ns → 900ns ngõ vào có giá trị là 7 thì ngõ ra có bit 0 được dịch vào tại vị trí $Q_7$ là 01111111.

**10 Mạch giải mã 3 sang 8 có chân Select**

**10.1 Sơ đồ khối**




Hình 2.8.1 RTL Schematic

Sơ đồ khối Hình 2.8.1 mô tả các ngõ vào và ra của mạch giải mã 3 sang 8 có chân điều khiển Select

## 10.2 Bảng trạng thái

| Inputs | | | | | Outputs | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| en | se | w2 | w1 | w0 | y7 | y6 | y5 | y4 | y3 | y2 | y1 | y0 |
| 1 | x | x | x | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 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 | 1 | 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 2.8.2

Bảng trạng thái Hình 2.8.2 mô tả các giá trị ngõ vào, ngõ ra và các chân điều khiển Se, E của mạch giải mã 3 sang 8

## 10.3 Mô tả mạch decoder 3_8 Select – Ngõ ra tích cực mức cao

```
module decoder3to8_select(

input e,se,

input wire [2:0] w,

output reg [7:0] y

    );

always @(e or w or se)

begin

if (e==0)

if (se==1)

case(w)

0 : y=8'b10000000;

1 : y=8'b01000000;

2 : y=8'b00100000;

3 : y=8'b00010000;

4 : y=8'b00001000;

5 : y=8'b00000100;
```

6 : y=8'b00000010;

7 : y=8'b00000001;

endcase

else

case(w)

0 : y=8'b01111111;

1 : y=8'b10111111;

2 : y=8'b11011111;

3 : y=8'b11101111;

4 : y=8'b11110111;

5 : y=8'b11111011;

6 : y=8'b11111101;

7 : y=8'b11111110;

endcase

else

y =8'b00000000;

end

endmodule

**Giải thích Code**

Khai báo cho các ngõ vào w, e, se và ngõ ra y. Sử dụng lệnh always cho điều kiện w, se, e khi 1 trong 3 có sự thay đổi sẽ thực hiện các lệnh if, case bên trong

**Tiến hành kiểm tra**

Hình 2.8.3

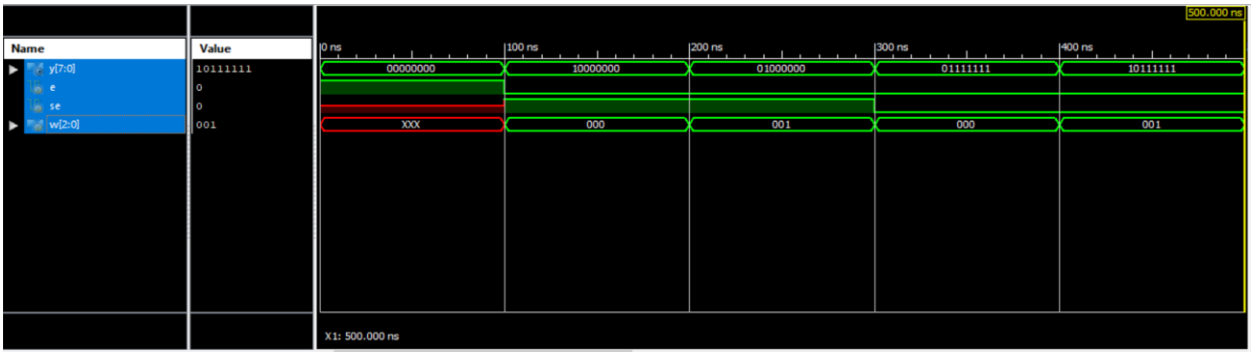
**10.4 Testbench và kết quả mô phỏng**

a) Testbench

```
module Test;
	// Inputs
	reg e;
	reg se;
	reg [2:0] w;
	// Outputs
	wire [7:0] y;
	// Instantiate the Unit Under Test (UUT)
	decoder3to8_select uut (
		.e(e),
		.se(se),
		.w(w),
		.y(y)
	);
	initial begin
		// Initialize Inputs
		e = 0;
		se = 0;
		w = 0;
		// Wait 100 ns for global reset to finish
		#100;
		e = 0;
		se = 1;
		w = 0;
		// Wait 100 ns for global reset to finish
		#100;
```

```
            e = 1;
            se = 0;
            w = 0;
            // Wait 100 ns for global reset to finish
            #100;
            // Add stimulus here
        $stop;
         end
endmodule
```

b) Kết quả mô phỏng




Hình 2.8.4

**Nhận xét kết quả mô phỏng Hình 2.8.4**

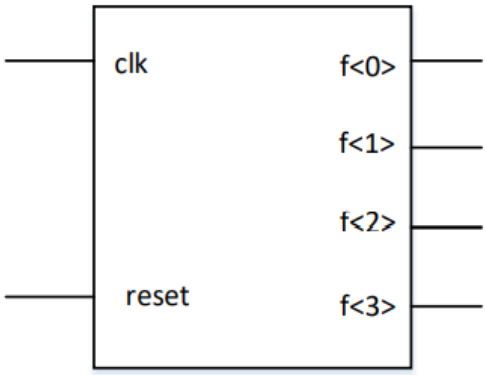
- Khoảng t từ 0ns → 100ns Enable ở mức 1 thì ngõ ra y sẽ có giá trị là 0 bất chấp các giá trị ngõ vào và Select

-Khoảng t từ 100ns → 300ns E = 0, Se = 1, Khi ngõ vào w=0,1,… thì ngõ ra sẽ có các bit 1 được dịch từ $Y_0$, $Y_1$,… ➔ Khi E=0,Se=1 Mạch giải mã từ 3 sang 8 sẽ có ngõ ra tích cực ở mức 1

-Khoảng t từ 300ns → 500ns E = 0, Se = 0, Khi ngõ vào w=0,1,… thì ngõ ra sẽ có các bit 0 được dịch từ $Y_0$, $Y_1$,… ➔ Khi E=0,Se=0 Mạch giải mã từ 3 sang 8 sẽ có ngõ ra tích cực ở mức 0.

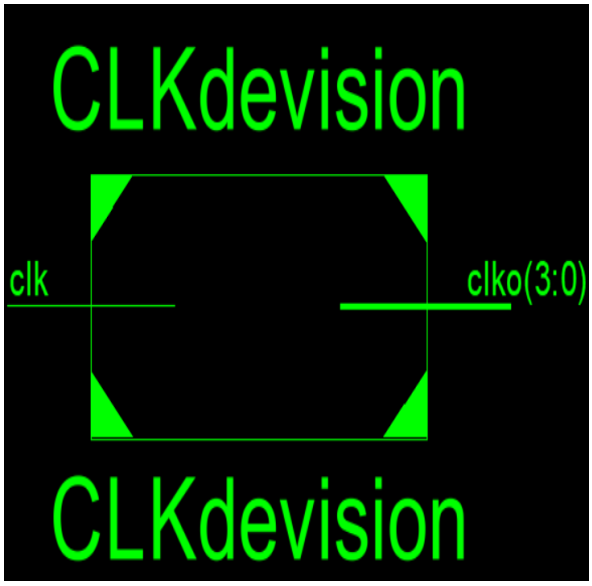
**11 Thiết kế mạch chia xung với ngõ vào 50Mhz, 4 xung ngõ ra với tần số f, 2f, 4f, 8f, trong đó lựa chọn f ~ 1hz.**

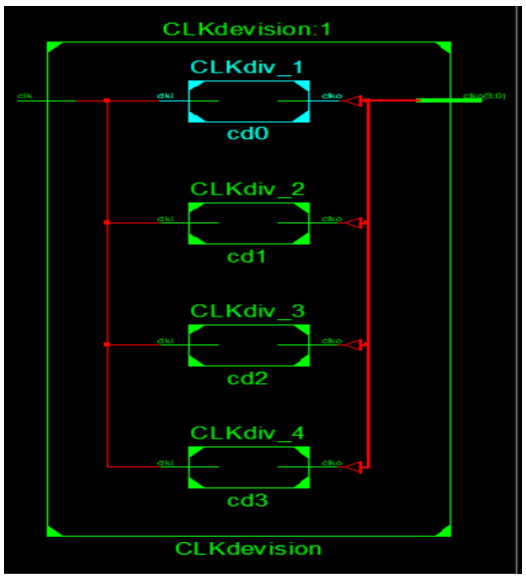
**11.1 Sơ đồ khối**




***Hình 3.3.1.1. Sơ đồ khối kế mạch chia xung với ngõ vào 50MHz và 4 ngõ ra tần số tương ứng 1hz, 2hz, 4hz, 8hz.***

**11.2 RTL schematic**




***Hình 3.3.1.2. Sơ đồ mạch RTL.***

**11.3 Mô tả mạch chia xung**

**11.3-1 Top module**

```
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
```

**11.3-2 Các chương trình con**

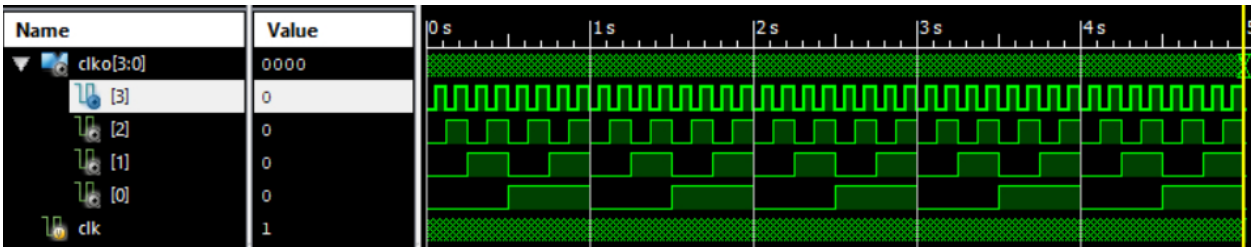
```
module CLKdiv
        #(parameter M=50000000)
        (input clki,
         output clko);
                wire [30:0] r_next;
                reg [30:0]  r_reg;
                initial r_reg=0;
                always @(posedge clki)
                        r_reg <= r_next;
                        assign r_next = (r_reg==M)?0:r_reg+1;
                        assign clko = (r_reg<=M/2)?0:1;
endmodule
```

**11.4 Tạo testbench**

```
module test;
        // Inputs
        reg clk;

        // Outputs
        wire [3:0] clko;

        // Instantiate the Unit Under Test (UUT)
        CLKdevision uut (
                .clk(clk),
```

```
        .clko(clko)
    );

    initial begin
        // Initialize Inputs
        clk = 0;
        // Wait 100 ns for global reset to finish
        #100;
        // Add stimulus here
    end
    always
    begin
    clk=~clk;
    #10;
    end

endmodule
```

**11.5 Kết quả mô phỏng**



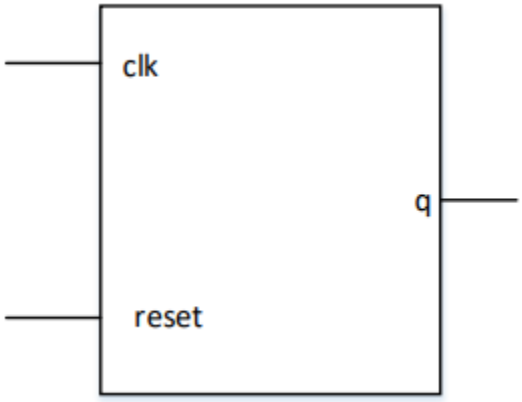

***Hình 3.3.1.4. Dạng sóng mô phỏng mạch chia xung với ngõ vào 50MHz và 4 ngõ ra tần số tương ứng 1hz, 2hz, 4hz, 8hz.***

- Nhận xét:

Với xung clk có tần số 50Mhz ta có được các xung ngõ ra clko[0], clko[1], clk[2], clk[3] tương ứng các tần số 1hz, 2hz, 4hz, 8hz.

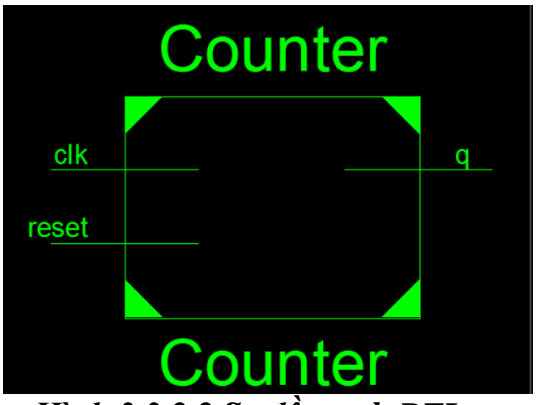
**12 Thiết kế mạch tạo xung 1hz.**

**12.1 Sơ đồ khối**




***Hình 3.3.2.1. Sơ đồ khối mạch tạo xung 1hz.***

**12.2 RTL schematic**




***Hình 3.3.2.2 Sơ đồ mạch RTL.***

**12.3 Mô tả mạch tạo xung 1Hz**

```
module Counter
#(parameter N= 26, M = 50000000)
( input wire clk, reset,
 output wire q
 );
// signal declaration
        reg [N-1:0] r_reg;
        wire [N-1:0] r_next;
```

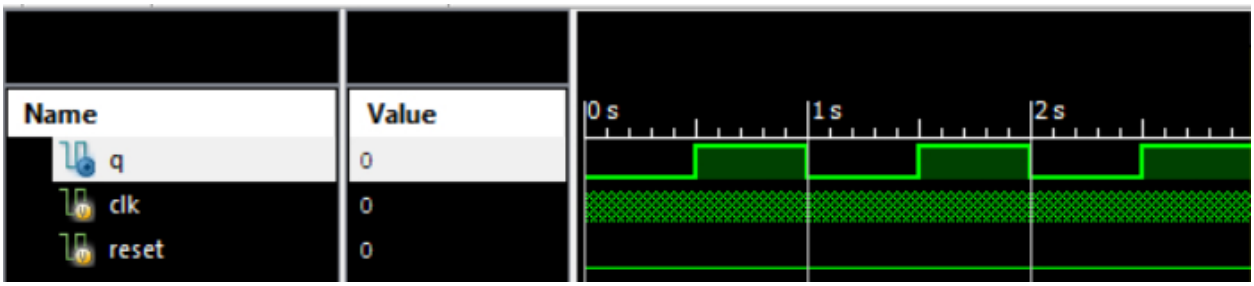
```
// body, register
        always @(posedge clk, posedge reset)
            if (reset)
                r_reg <= 0;
            else
                r_reg<=r_next;
        // next state logic
        assign r_next = (r_reg==M)?0:r_reg + 1;
        // output logic
        assign q=(r_reg<M/2)?0:1;
endmodule
```

**12.4 Tạo Testbech**

```
module hz;
    // Inputs
    reg clk;
    reg reset;
    // Outputs
    wire q;
    // Instantiate the Unit Under Test (UUT)
    Counter uut (
        .clk(clk),
        .reset(reset),
        .q(q)
    );

    initial begin
        // Initialize Inputs
        clk = 0;
        reset = 1;
        #20;
        reset=0;

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

```
            #100;

            // Add stimulus here
        end
        always
        begin
        #10;
        clk=~clk;
        end
endmodule
```

**12.5 Kết quả mô phỏng**



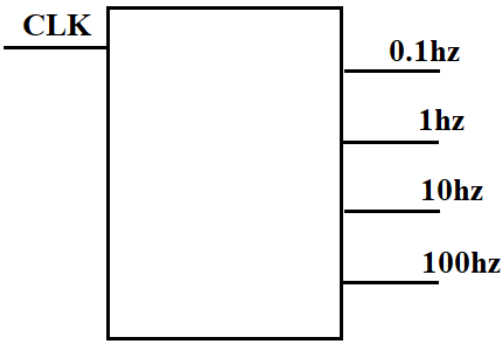

***Hình 3.3.2.5. Dạng sóng mô phỏng của mạch tạo xung 1hz.***

- Nhận xét:

Với xung clk ban đầu có tần số 50Mhz ta được xung 1hz tương ứng với ngõ ra q.

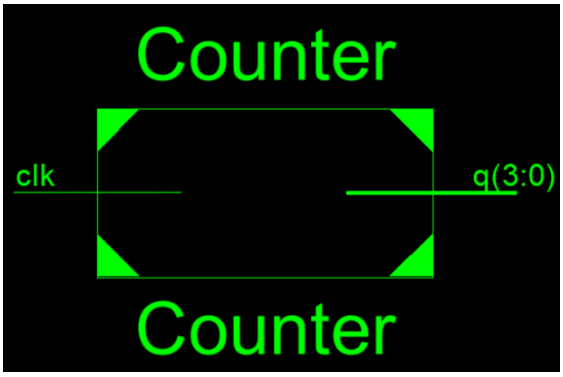
**13 Thiết kế mạch tạo 4 xung ngõ ra với tần số lần lượt là 0.1Hz, 1Hz, 10Hz, 100Hz.**

**13.1 Vẽ sơ đồ khối**




***Hình 3.3.3.1. Sơ đồ khối kế mạch tạo xung 4 ngõ ra với tần số lần lượt là 0.1hz, 1hz, 10hz, 100hz.***

**13.2 RTL Schematic**




***Hình 3.3.3.2 Sơ đồ khối RTL***

**13.3 Mô tả mạch tạo 4 xung ngõ ra**

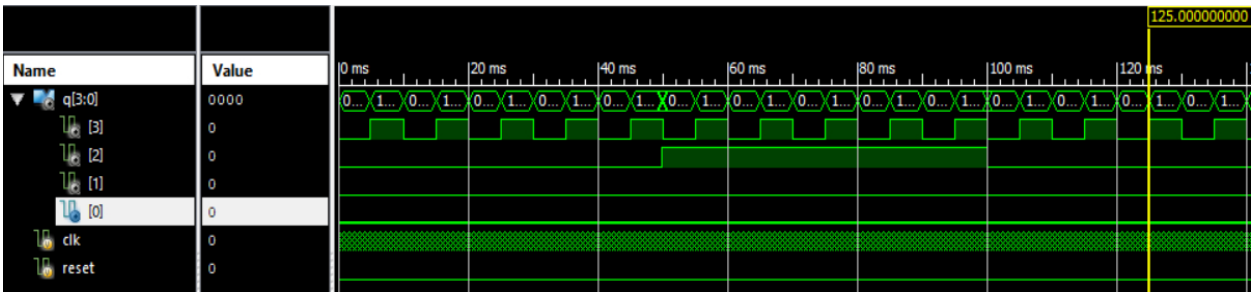
```
module Counter
 #(parameter N= 30, M = 500000000) // 500,000,000 for 0.1Hz
( input wire clk,
 output wire [3:0]q
 );
// signal declaration
        reg [N-1:0] r_reg01H,r_reg1H,r_reg10H,r_reg100H;
        wire [N-1:0]r_next01H,r_next1H,r_next10H,r_next100H;
// body, register initial r_reg01H = 0;
        initial r_reg01H = 0;
        initial r_reg1H = 0;
        initial r_reg10H = 0;
        initial r_reg100H = 0;
        always @(posedge clk) begin
                r_reg01H<=r_next01H;
                r_reg1H<=r_next1H;
                r_reg10H<=r_next10H;
                r_reg100H<=r_next100H;
        end
// next state logic
        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;
```

```
 // output logic
        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
```

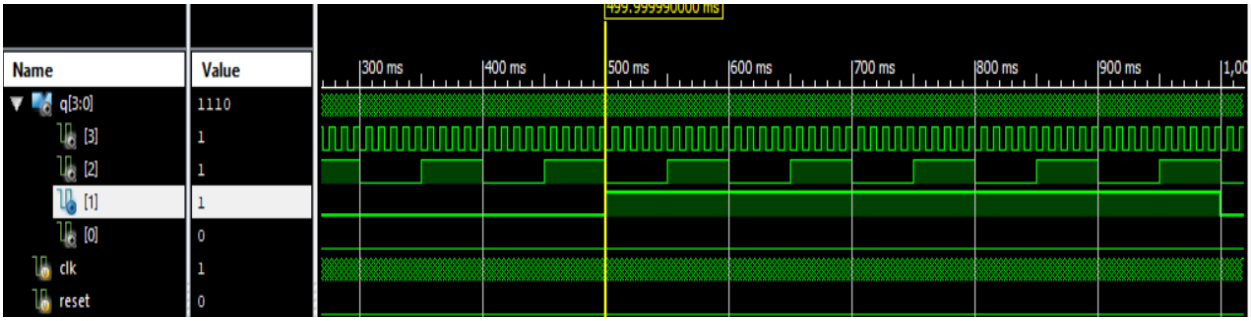
**13.4 Tạo Testbench**

```
module test;
// Inputs
reg clk;
reg reset;
// Outputs
wire [3:0] q;
        // Instantiate the Unit Under Test (UUT)
        Counter uut (
                .clk(clk),
.q(q)
        );
        initial begin
                clk = 0;
                reset = 0;
                #10;
                reset = 1;
                #10;
                reset = 0;
        end
        always
                begin
                        #10;
                        clk=~clk;
                end
endmodule
```

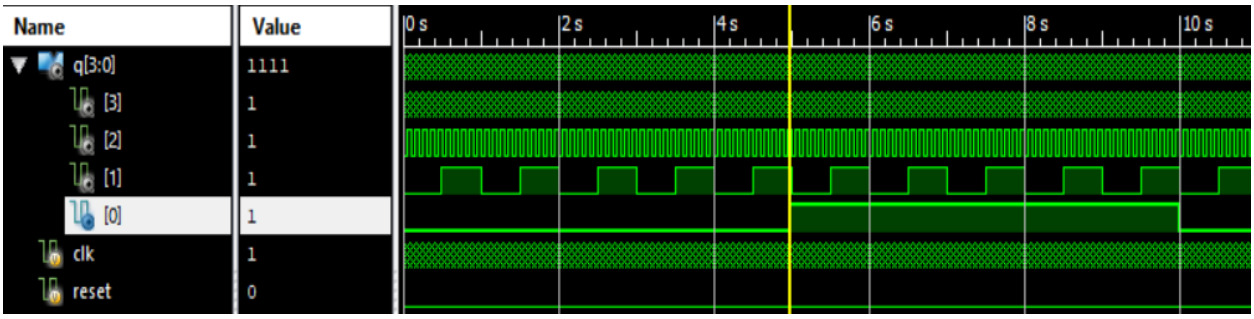
**13.5 Kết quả mô phỏng**




*Hình 3.3.3.5-1. Dạng sóng mô phỏng của mạch tạo 4 xung ngõ ra*




*Hình 3.3.3.5-2. Dạng sóng mô phỏng của mạch tạo 4 xung ngõ ra*



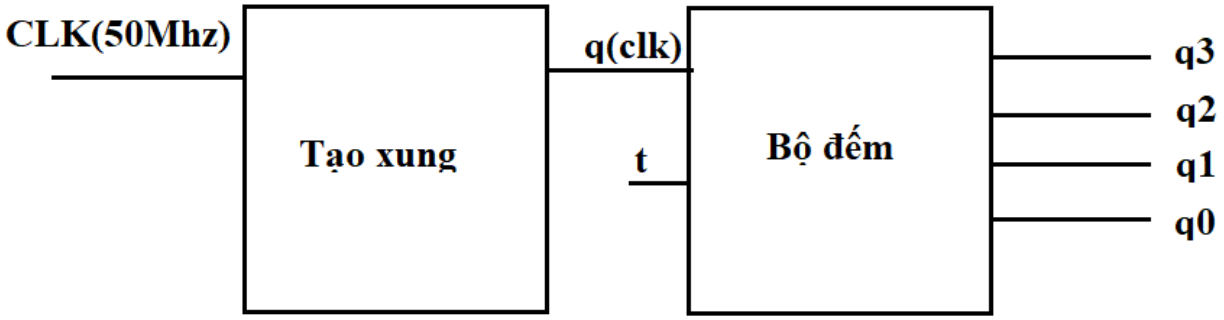

*Hình 3.3.3.5-3. Dạng sóng mô phỏng của mạch tạo 4 xung ngõ ra*

➢ Nhận xét

- Hình 3.3.3.5-1, trong khoảng thời gian từ 0-100ms, ngõ ra q[3] dao động với tần số 100hz.
- Hình 3.3.3.5-1, trong khoảng thời gian từ 0-100ms, ngõ ra q[2] dao động với tần số 10hz.
- Hình 3.3.3.5-2, trong khoảng thời gian từ 0-1000ms, ngõ ra q[1] dao động với tần số 1hz.
- Hình 3.3.3.5-3, trong khoảng thời gian từ 0-10s, ngõ ra q[0] dao động với tần số 0.1hz.

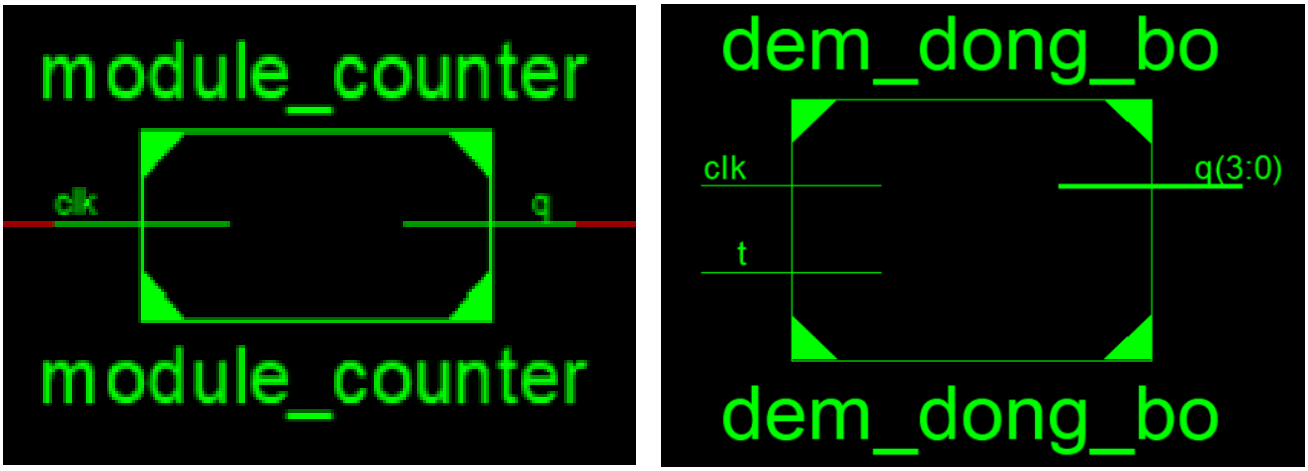
**14. Thiết kế mạch đếm đồng bộ, sử dụng phương pháp cài đặt các Flip – Flop. Xung đếm 1Hz được lấy từ mạch chia xung.**

**14.1 Sơ đồ khối**




*Hình 3.3.4.1. Sơ đồ khối thiết kế mạch đếm đồng bộ, sử dụng phương pháp cài đặt các Flip – Flop với xung đếm 1Hz lấy từ mạch chia xung.*

**14.2 RTL schematic**




*Hình 3.3.4.2 Sơ đồ khối RTL*

**14.3 Mô tả mạch**

**14.3-1 Module mạch tạo xung 1Hz**

```
module module_counter
#(parameter N= 26, M = 50000000)
( input wire clk,
 output wire q );
```

```
// signal declaration
reg [N-1:0] r_reg;
wire [N-1:0] r_next;
// body, register
    initial
       r_reg = 0;
       always @(posedge clk)
       r_reg<=r_next;
// next state logic
       assign r_next = (r_reg==M)?0:r_reg + 1;
// output logic
       assign q=(r_reg<M/2)?0:1;
endmodule
```

**14.3-2 Module Flip-Flop T**

```
module ffft(
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
```

**14.3-3 Module mạch đếm đồng bộ**

```
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);
     ffft a ( t, clk1, t0 );
     ffft b ( t0,clk1, t1 );
     ffft c ( t0&&t1, clk1, t2);
     ffft d ( t0&&t1&&t2, clk1, t3);
  always @(posedge clk)
     begin
        q[0]=t0;
        q[1]=t1;
        q[2]=t2;
        q[3]=t3;
      end
endmodule
```

**14.4 Tạo Testbench**

```
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
begin
#1 ;
Clk=~clk ;
end
```

```
always
begin
#1 ;
 t=~t ;
end
endmodule
```

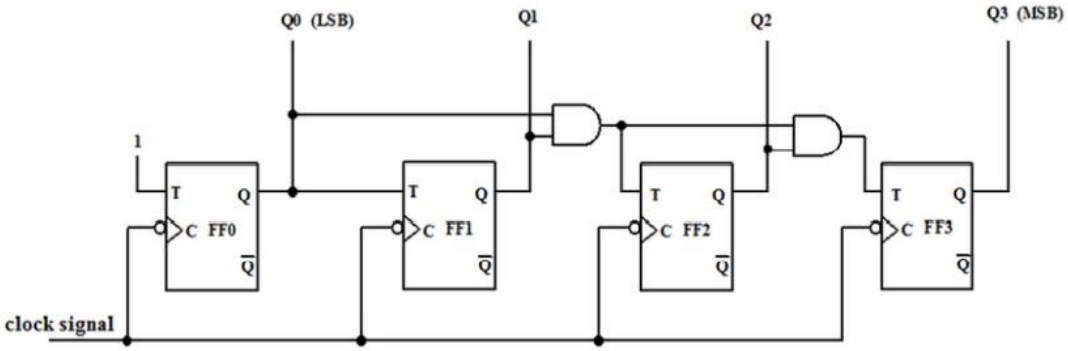
### 14.5 Kết quả mô phỏng

*Hình 3.3.4.5. Dạng sóng mô phỏng của mạch đếm*

- Nhận xét : Mạch đếm đồng bộ đang đếm lên từ 0 đến 15 và quay lại

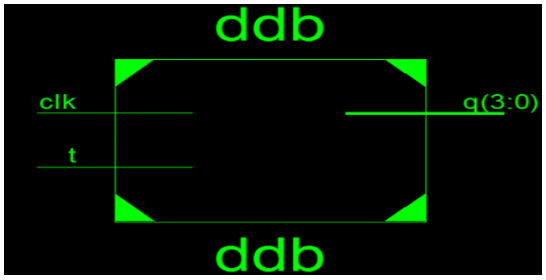
## 15 Thiết kế mạch đếm lên 4 bit như bài 3.4, sử dụng phương pháp thiết kế đồng bộ.

### 15.1 Sơ đồ khối




*Hình 3.3.5.1. Sơ đồ khối thiết kế mạch đếm đồng bộ.*

**15.2 RTL Schematic**




***Hình 3.3.5.2. Sơ đồ khối RTL.***

**15.3 Mô tả mạch**

**15.3-1 Module TFF**

```
module fft(
        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
```

**15.3-2 Module mạch đếm đồng bộ**

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

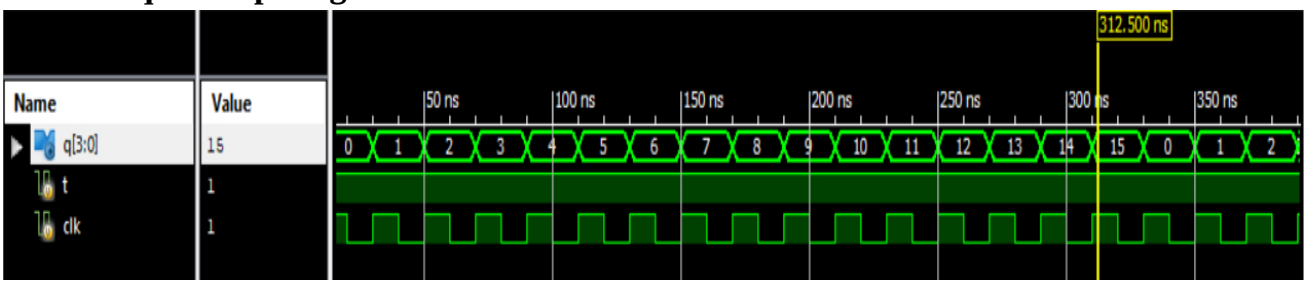
```
        input clk
    );
        wire [3:0] t0,t1,t2,t3;

            fft a1 ( t, clk, t0 );
            fft a2 ( t0,clk, t1 );
            fft a3 ( t0&&t1, clk, t2);
            fft a4 ( t0&&t1&&t2, clk, t3);
        always @(posedge clk)
            begin
                q[0]=t0;
                q[1]=t1;
                q[2]=t2;
                q[3]=t3;
            end
 endmodule
```

**15.4 Tạo Testbench**

```
 module tt;
        // Inputs
        reg t;
        reg clk;
        // Outputs
        wire [3:0] q;
        // Instantiate the Unit Under Test (UUT)
        ddb uut (
            .t(t),
```

```
        .q(q),
        .clk(clk)
    );
        initial begin
        // Initialize Inputs
        t = 1;
        clk = 0;
        // Wait 100 ns for global reset to finish
        end
        always
        begin
        #10;
        clk=~clk;
        end
endmodule
```

**15.5 Kết quả mô phỏng**



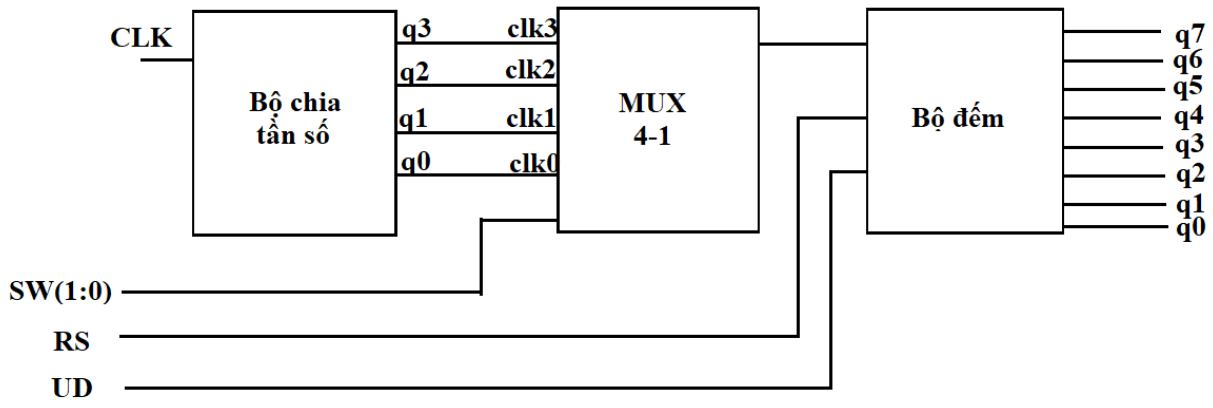

***Hình 3.3.5.5. Dạng sóng mô phỏng của mạch đếm đồng bộ.***

- Nhận xét:

Mạch đếm đồng bộ cho ra kết quả đúng từ 0-15 và quay lại.

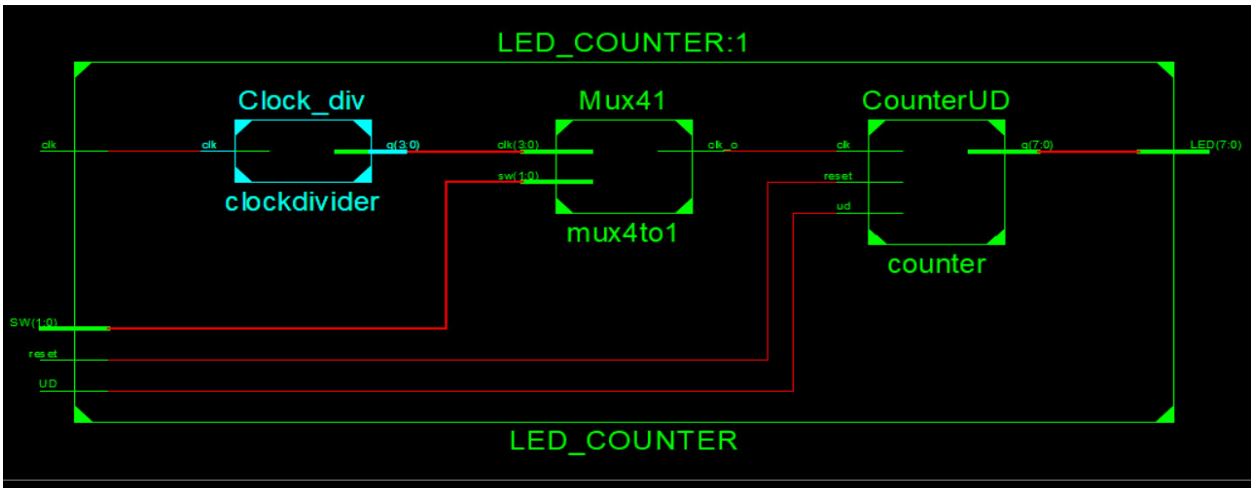
**16 Thiết kế mạch đếm lên 8 bit, lựa chọn tần số đếm, lựa chọn đếm lên hoặc đếm xuống.**

**16.1 Vẽ sơ đồ khối**




*Hình 3.3.6-1. Sơ đồ khối Thiết kế mạch đếm lên 8 bit, lựa chọn tần số đếm, lựa chọn đếm lên hoặc đếm xuống.*

**16.2 RTL Schematic**




*Hình 3.3.6.2. Sơ đồ khối RTL Thiết kế mạch đếm lên 8 bit, lựa chọn tần số đếm, lựa chọn đếm lên hoặc đếm xuống.*

**16.3 Mô tả mạch**

**16.3-1 Module mạch chia tần số**

```
module Clock_div
#(parameter N= 30, M = 500000000) // 500,000,000 for 0.1Hz
( input wire clk,
```

```
 output wire [3:0]q
 );
// signal declaration
        reg [N-1:0] r_reg01H,r_reg1H,r_reg10H,r_reg100H;
        wire [N-1:0] r_next01H,r_next1H,r_next10H,r_next100H;
// body, register
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
// next state logic
            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;
// output logic
            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
```

**16.3-2 Module MUX 4-1**

```
module Mux41
( input wire [3:0] clk,
 input wire [1:0] sw,
 output reg clk_o
 );
// signal declaration
// 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
```

**16.3 - 3 Module mạch đếm**

```
module CounterUD
#(parameter N= 8) // 500,000,000 for 0.1Hz
( input wire clk,reset,ud,
 output wire [7:0]q
 );
```

```
// signal declaration
        reg [N-1:0] r_reg;
        wire [N-1:0] r_next;
// body, register
        always @(posedge clk, posedge reset)
          if (reset)
                r_reg<=0;
          else
                r_reg<=r_next;
// next state logic
        assign r_next = (ud==1)?r_reg + 1:r_reg - 1;
// output logic
        assign q=r_reg;
endmodule
```

**16.3 - 4 Module chính**

```
module LED_COUNTER(
input wire clk, reset,
input wire [1:0] SW,
input wire UD,
output wire [7:0] LED
 );
// wire declaration
        wire [3:0] f;
        wire clk_o ;
// module instance
        Clock_div clockdivider (clk, f) ;
```

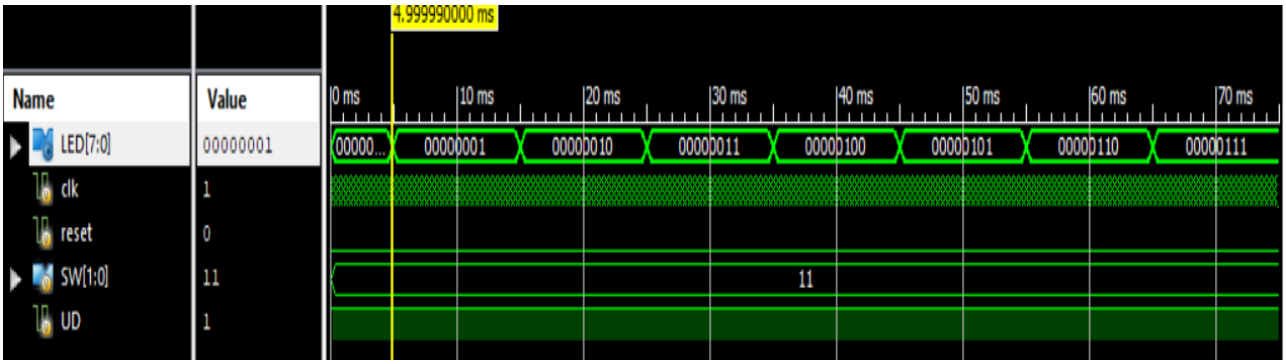
```
        Mux41 mux4to1 (f,SW,clk_o);
        CounterUD counter (clk_o, reset,UD,LED);
endmodule
```

**16.4 Tạo Testbench**

```
module tt;
        // Inputs
        reg clk;
        reg reset;
        reg [1:0] SW;
        reg UD;
        // Outputs
        wire [7:0] LED;
        // Instantiate the Unit Under Test (UUT)
        LED_COUNTER uut (
            .clk(clk),
            .reset(reset),
            .SW(SW),
            .UD(UD),
            .LED(LED)
        );
initial begin
  clk = 0;
forever #10 clk = !clk;
end
        initial begin
            // Initialize Inputs
```

```
            reset = 1;
            SW = 3;
            UD = 0;
            // Wait 100 ns for global reset to finish
            #100;
    reset = 0;
            SW = 3;
            UD= 1;
            // Wait 100 ns for global reset to finish
            #100;
            // Add stimulus here
        end
endmodule
```

**16.5 Kết quả mô phỏng**



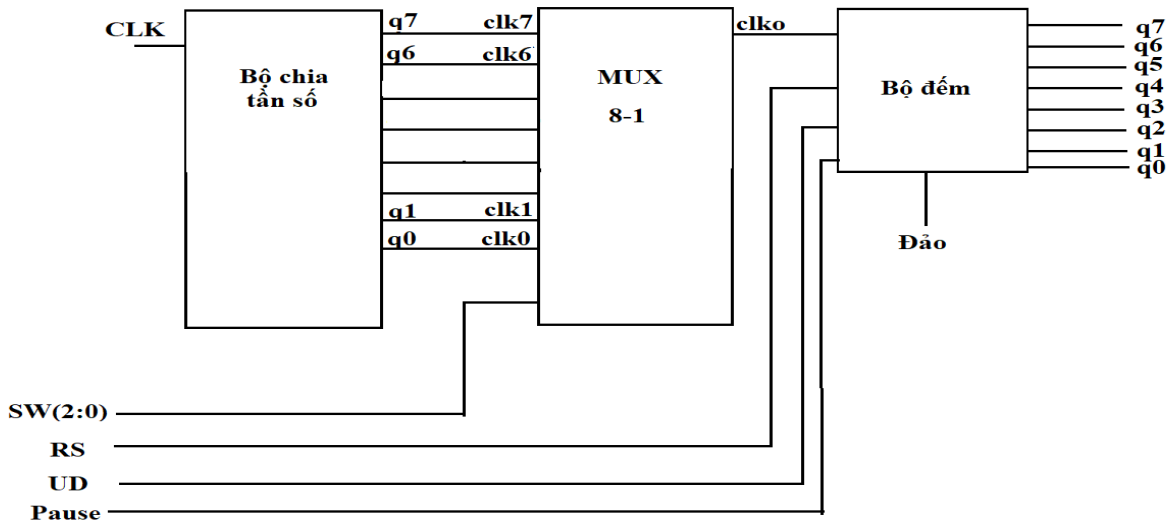

***Hình 3.3.6.5. Dạng sóng mô phỏng của mạch đếm.***

- Nhận xét

  Với SW=3 tức chọn xung 100hz để mạch đếm đếm lên với UD=1.

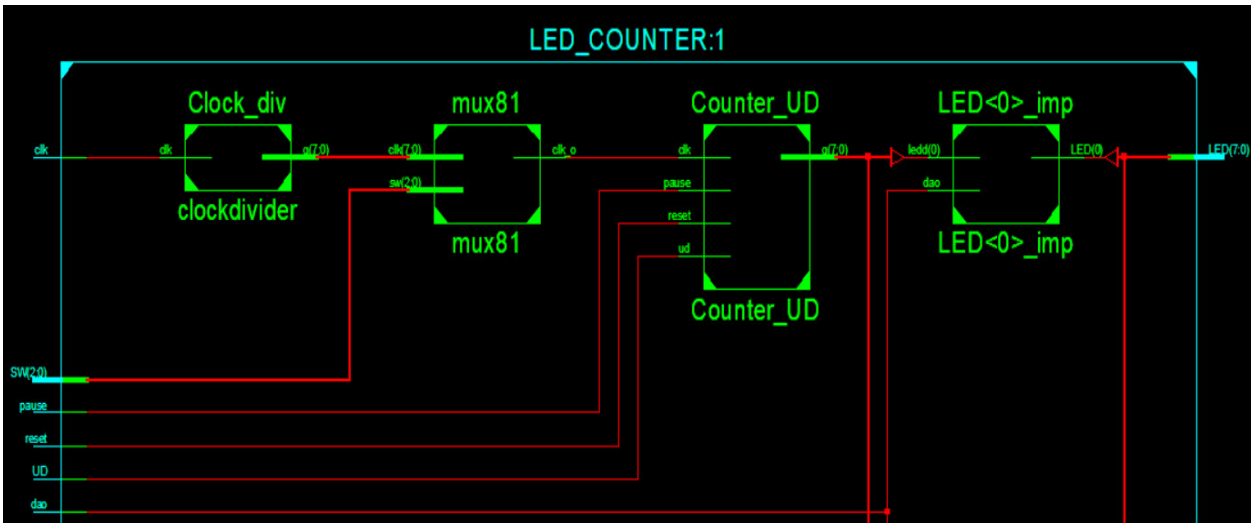
**17 Thiết kế mạch đếm lên 8 bit, lựa chọn 8 tần số đếm khác nhau, lựa chọn đếm lên hoặc đếm xuống, có tính hiệu cho phép dừng đếm (Pause), có tín hiệu đảo trạng thái ngõ ra.**

**17.1 Sơ đồ khối**




*Hình 3.3.7-1. Sơ đồ khối 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.*

**17.2 RTL Schematic**




*Hình 3.3.7.2. Sơ đồ khối RTL.*

**17.3 Mô tả mạch**

**17.3-1 Module mạch chia tần số**

```
module Clock_div
#(parameter N= 30, M = 500000000) // 500,000,000 for 0.1Hz
( input wire clk,
 output wire [7:0]q
 );
// signal declaration
reg [N-1:0]
r_reg01H,r_reg1H,r_reg10H,r_reg100H,r_reg1000H,r_reg10000H,r_reg100000H,r_reg1000000H;
wire [N-1:0]
r_next01H,r_next1H,r_next10H,r_next100H,r_next1000H,r_next10000H,r_next100000H,r_next1000000H;
// body, register
initial
begin
r_reg01H=0;
r_reg1H=0;
r_reg10H=0;
r_reg100H=0;
r_reg1000H=0;
r_reg10000H=0;
r_reg100000H=0;
r_reg1000000H=0;
end
always @(posedge clk) begin
```

```
r_reg01H<=r_next01H;
r_reg1H<=r_next1H;
r_reg10H<=r_next10H;
r_reg100H<=r_next100H;
r_reg1000H<=r_next1000H;
r_reg10000H<=r_next10000H;
r_reg100000H<=r_next100000H;
r_reg1000000H<=r_next1000000H;
end
// next state logic
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 r_next1000H = (r_reg1000H==M/10000)?0:r_reg100H + 1;
assign r_next10000H = (r_reg10000H==M/100000)?0:r_reg100H + 1;
assign r_next100000H = (r_reg1000000H==M/1000000)?0:r_reg100H + 1;
assign r_next1000000H = (r_reg1000000H==M/10000000)?0:r_reg100H + 1;
 // output logic
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;
assign q[4]=(r_reg1000H<M/20000)?0:1;
assign q[5]=(r_reg10000H<M/200000)?0:1;
assign q[6]=(r_reg100000H<M/2000000)?0:1;
```

```
assign q[7]=(r_reg1000000H<M/20000000)?0:1;
endmodule
```

**17.3-2 Module MUX 8-1**

```
module mux81
( input wire [7:0] clk,
 input wire [2:0] sw,
 output reg clk_o
 );
// signal declaration
// 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];
                        4: clk_o = clk[4];
                        5: clk_o = clk[5];
                        6: clk_o = clk[6];
                        7: clk_o = clk[7];
                endcase
endmodule
```

**17.3-3 Module mạch đếm**

```
module Counter_UD
#(parameter N= 8) // 500,000,000 for 0.1Hz
( input wire clk,reset,ud,pause,
```

```
 output wire [7:0]q
 );
// signal declaration
reg [N-1:0] r_reg;
wire [N-1:0] r_next;
// body, register
            always @(posedge clk, posedge reset)
                if (reset)
                        r_reg<=0;
                else
                        r_reg<=r_next;
// next state logic
assign r_next = (ud==1 && pause ==0)?r_reg + 1:(ud==0 && pause ==0)?r_reg -
1:(ud==0 && pause==1)?r_reg:r_reg;
// output logic
assign q=r_reg;
endmodule
```

**17.4 Tạo Testbench**

```
module testbench;
// Inputs
reg clk;
reg reset;
reg [2:0] S;
reg UD;
reg Pause;
reg Toggle;
```

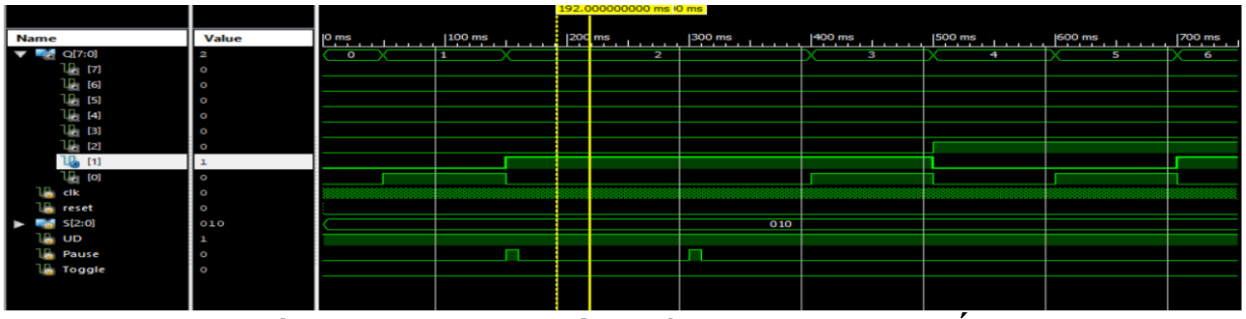
```
// Outputs
wire [7:0] Q;
// Instantiate the Unit Under Test (UUT)
Counter8bitUDPauseToggle uut (
.clk(clk),
.reset(reset),
.S(S),
.UD(UD),
.Pause(Pause),
.Toggle(Toggle),
.Q(Q)
);
```

27

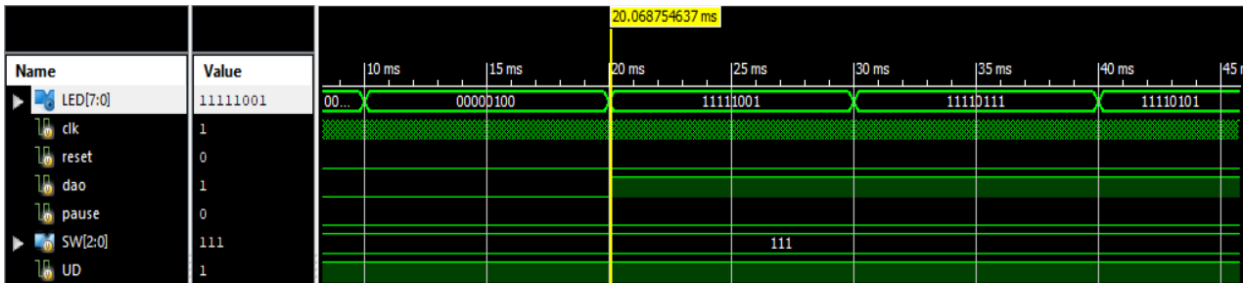
```
initial begin
// Initialize Inputs
clk = 0;
reset = 0;
S = 2;
UD = 0;
Pause = 0;
Toggle = 0;
#10;
reset=1;
#10;
reset=0;
#200000000;
```

```
UD=1;
#150000000;
Pause=1;
#10000000;
Pause=0;
#140000000;
Pause=1;
#10000000;
Pause=0;
// Add stimulus here
end
always
 begin
#10;
clk=~clk;
end
endmodule
```

**17.5 Kết quả mô phỏng**




***Hình 3.3.7.5-1 Dạng sóng mô phỏng của mạch đếm.***




***Hình 3.3.7.5-2 Dạng sóng mô phỏng của mạch đếm.***

- Nhận xét

*Hình 3.3.7.5-1*

-SW=010 => chọn tần số xung 10hz.

-UD=1 nên mạch đếm lên.

-Tại thời điểm 150ms và 300ms Pause lên mức 1 ➔ mạch đếm trong khoảng thời gian tạm dừng.

*Hình 3.3.7.5-2*

-Tại thời điểm 200ms tín hiệu đảo lên mức 1 ➔ mạch đếm bị đảo từ 0 thành 1.