ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KĨ THUẬT MÁY TÍNH

BÀI BÁO CÁO ASSIGNMENT LAB01
THIẾT KẾ HỆ THỐNG SỐ VỚI HDL

Sinh viên: Trương Thiên Quý
MSSV: 23521321
Lớp: CE213.P21.2
Giảng viên hướng dẫn: Hồ Ngọc Diễm

BÀI THỰC HÀNH SỐ 2

I. Mục tiêu

Trong bài thực hành này, sinh viên sẽ thực hành lệnh **assign** (continuous assignment) và cách gọi module theo thứ bậc để thiết kế các mạch tổ hợp.

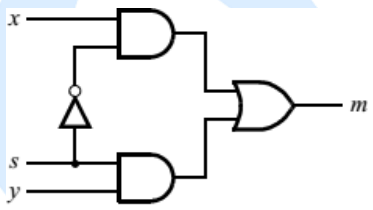
II. Chuẩn bị thực hành

- Sinh viên phải chuẩn bị code Verilog cho tất cả các câu trong phần nội dung thực hành và nộp cho GVHD vào đầu buổi học.
- Sinh viên nào không có bài chuẩn bị được xem là vắng buổi học hôm đó.
- Bài chuẩn bị được tính vào điểm bài báo cáo của Lab.

III. Nội dung thực hành

Câu 1.

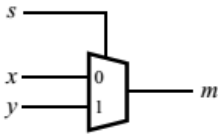
a) Thiết kế bộ Mux_2to1 1-bit:



sơ đồ mạch

s	m
0	x
1	y

Bảng sự thật

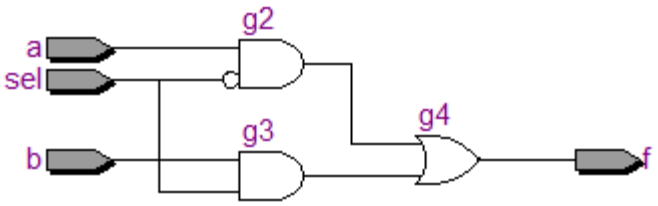


Ký hiệu

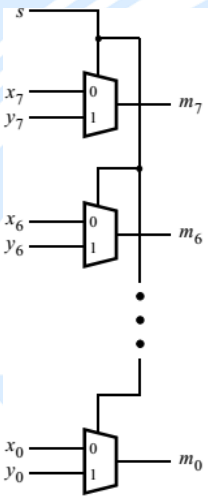
***Code Thực Thi:**

```
1  module Mux2to1_1bit(f, a, b, sel);
2  input a, b, sel;
3  output f;
4  not g1(nsel, sel);
5  and g2(f1, a, nsel);
6  and g3(f2, b, sel);
7  or g4(f, f1, f2);
8  endmodule
```

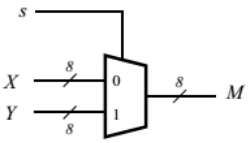
***Trình mô phỏng RTL Viewer:**



b) Sử dụng bộ Mux_2to1 1-bit để thiết kế bộ Mux_2to1_8-bit như sau:



Sơ đồ mạch



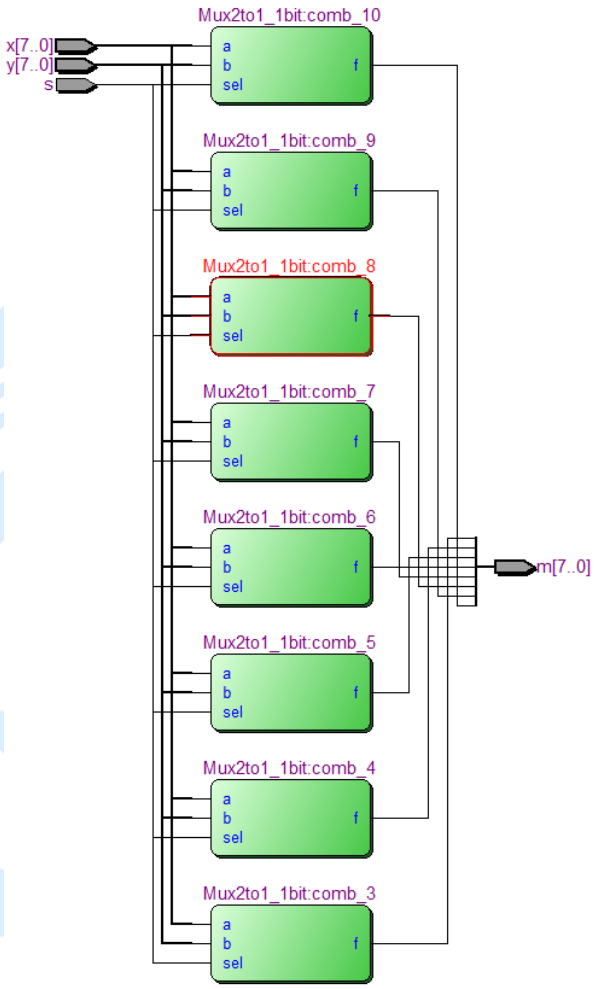
Ký hiệu

Yêu cầu: Nạp mạch và kiểm tra trên kit DE2

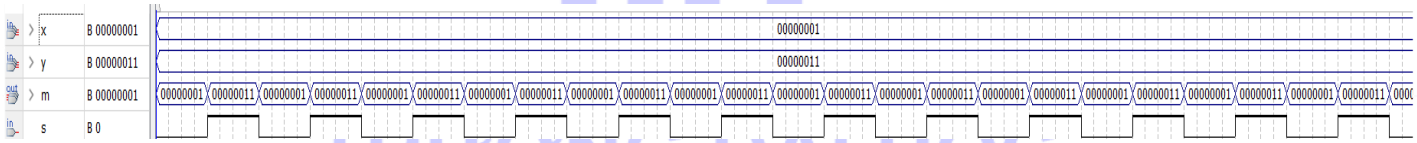
***Code Thực Thi:**

```
1   module Mux2to1_8bit(x, y, s, m);
2   input [7:0] x;
3   input [7:0] y;
4   input s;
5   output [7:0] m;
6   Mux2to1_1bit (m[7], x[7], y[7], s);
7   Mux2to1_1bit (m[6], x[6], y[6], s);
8   Mux2to1_1bit (m[5], x[5], y[5], s);
9   Mux2to1_1bit (m[4], x[4], y[4], s);
10  Mux2to1_1bit (m[3], x[3], y[3], s);
11  Mux2to1_1bit (m[2], x[2], y[2], s);
12  Mux2to1_1bit (m[1], x[1], y[1], s);
13  Mux2to1_1bit (m[0], x[0], y[0], s);
14  endmodule
```

RTL Viewer:



Wareform:

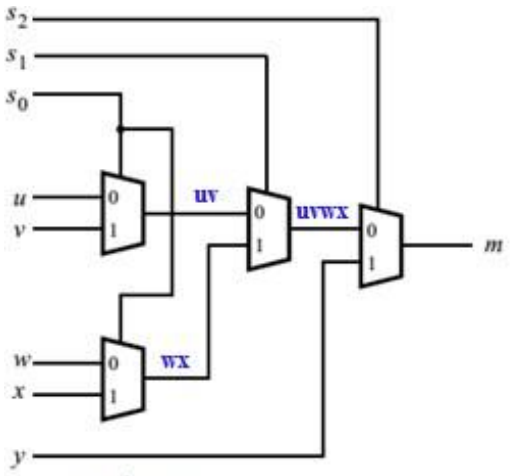


Chức năng của mạch:

- Input 2 đầu vào 8 bit (a = 00000001 và b = 00000011) và 1 đầu điều khiển (m).
- Khi đầu điều khiển m = 0, đầu vào 00000001 sẽ được chọn và truyền đến đầu ra.
- Khi đầu điều khiển m = 1, đầu vào 00000011 sẽ được chọn và truyền đến đầu ra.
- Như vậy, mạch Mux 2 to 1 8bit sẽ chọn 1 trong 2 đầu vào 8 bit dựa trên giá trị của đầu điều khiển m, Nếu m = 0 thì output sẽ là a[8], còn m = 1 thì output sẽ là b[8].

Câu 2.

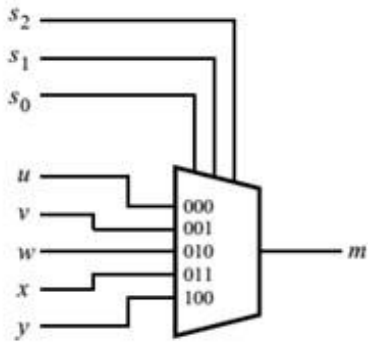
a) Thực hiện bộ Mux_5to1_1-bit từ các Mux_2to1_bit



Sơ đồ mạch

s_2	s_1	s_0	m
0	0	0	u
0	0	1	v
0	1	0	w
0	1	1	x
1	0	0	y
1	0	1	y
1	1	0	y
1	1	1	y

Bảng sự thật



Ký hiệu

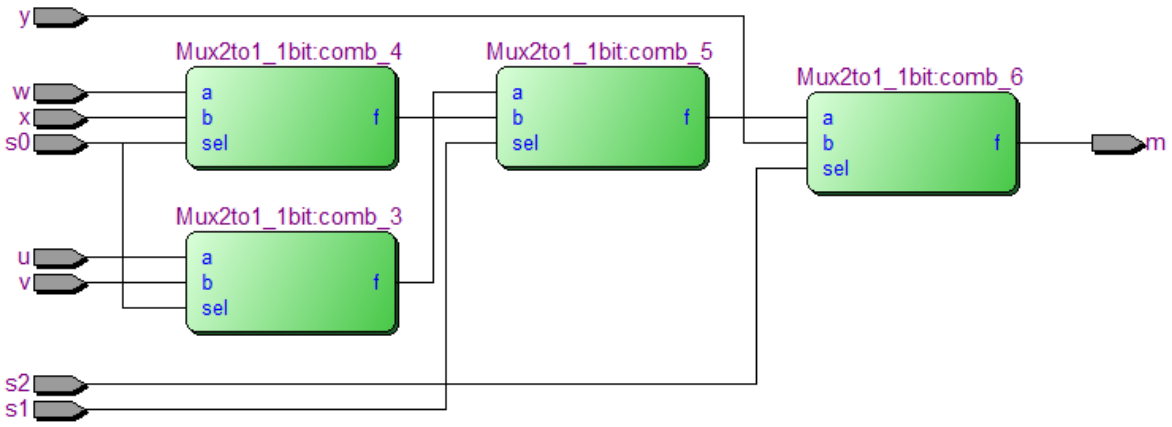
Yêu cầu:

Thực hiện mô phỏng với Vector Waveform cho mạch Mux_5to1 trên

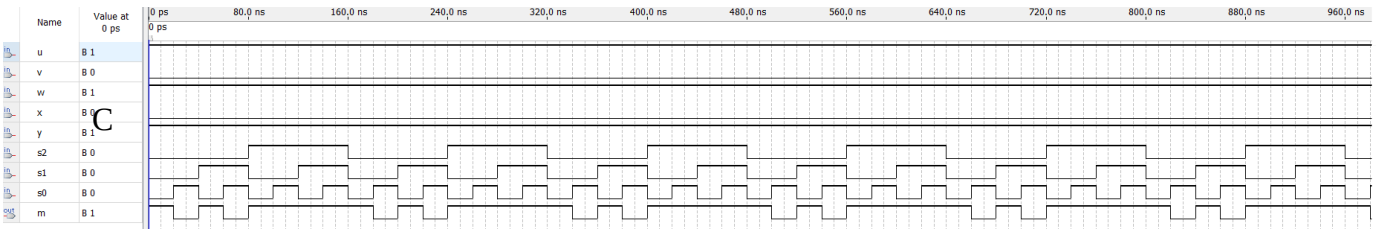
Code Thực Thi:

```
1  module Mux5to1_1bit(s2, s1, s0, u, v, w, x, y, m);
2  input s2, s1, s0, u, v, w, x, y;
3  output m;
4  wire uv, wx, uvwx;
5  Mux2to1_1bit(uv, u, v, s0);
6  Mux2to1_1bit(wx, w, x, s0);
7  Mux2to1_1bit(uvwx, uv, wx, s1);
8  Mux2to1_1bit(m, uvwx, y, s2);
9  endmodule
```

RTL Viewer:



Mô phỏng Wareform:

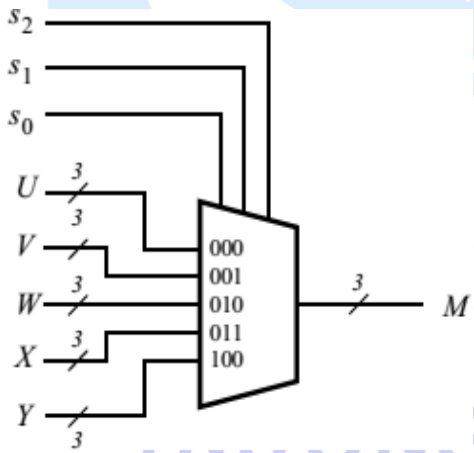


Chức năng của mạch:

- Input 5 đầu vào 1 bit (w, x, y, z, E) và 3 đầu điều khiển (A[0], A[1], A[2]).
- Dựa trên giá trị của 3 đầu điều khiển A[0], A[1], A[2], mạch Mux sẽ chọn và truyền 1 trong 5 đầu vào đến đầu ra.
- Ví dụ:
  - Khi A[0] = 0, A[1] = 0, A[2] = 0, đầu vào w sẽ được chọn và truyền đến đầu ra.
  - Khi A[0] = 1, A[1] = 0, A[2] = 0, đầu vào x sẽ được chọn và truyền đến đầu ra.
  - Và tương tự cho các trường hợp khác.

  Như vậy, mạch Mux 5 to 1_1 bit này thường được sử dụng trong các hệ thống số để chọn và truyền một trong nhiều nguồn dữ liệu 1 bit đến một đầu ra chung, dựa trên tín hiệu điều khiển.

b) Thiết kế bộ Mux_5to1_3-bit



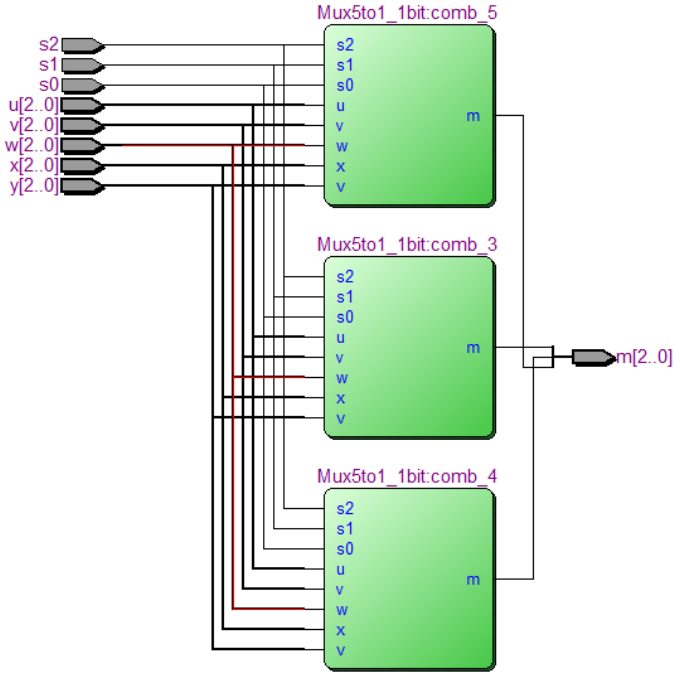
Yêu cầu:

Nạp mạch và kiểm tra trên kit DE2

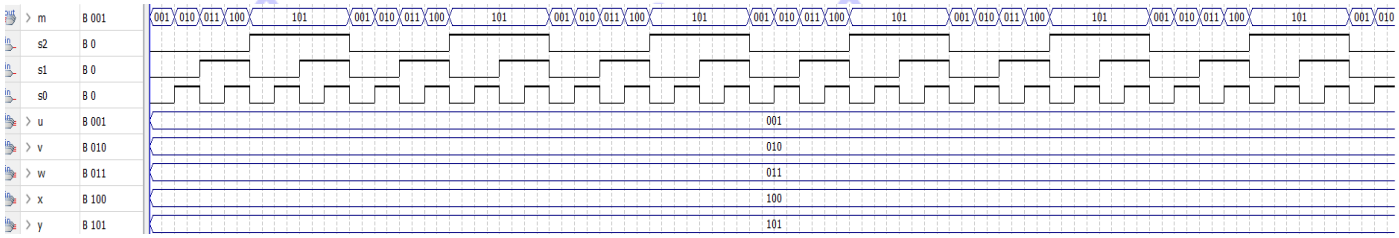
Code Thực Thi:

```
1   module Mux5to1_3bit(s2, s1, s0, u, v, w, x, y, m);
2   input s2, s1, s0;
3   output [2:0] m;
4   input [2:0] u;
5   input [2:0] v;
6   input [2:0] w;
7   input [2:0] x;
8   input [2:0] y;
9   Mux5to1_1bit(s2, s1, s0, u[2], v[2], w[2], x[2], y[2], m[2]);
10  Mux5to1_1bit(s2, s1, s0, u[1], v[1], w[1], x[1], y[1], m[1]);
11  Mux5to1_1bit(s2, s1, s0, u[0], v[0], w[0], x[0], y[0], m[0]);
12  endmodule
```

RTL Viewer:



Mô phỏng Wareform:



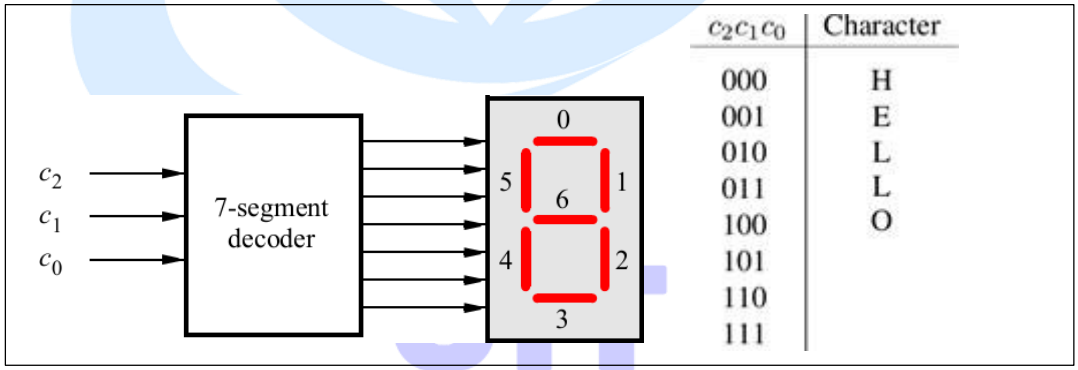
Chức năng của mạch:

- Input 5 đầu vào 3 bit (001, 010, 011, 100, 101) và 3 đầu điều khiển (A[0], A[1], A[2]ơ).
- Dựa trên giá trị của 3 đầu điều khiển [0], A[1], A[2], mạch Mux sẽ chọn và truyền 1 trong 5 đầu vào 3 bit đến đầu ra.
- Ví dụ:
  - Khi A[0] = 0, A[1] = 0, A[2] = 0, đầu vào 001 sẽ được chọn và truyền đến đầu ra.
  - Khi A[0] = 1, A[1] = 0, A[2] = 0, đầu vào 010 sẽ được chọn và truyền đến đầu ra.
  - Và tương tự cho các trường hợp khác.

➔ Mạch Mux5to1 3 bit này thường được sử dụng trong các hệ thống số để chọn và truyền một trong nhiều nguồn dữ liệu 3 bit đến một đầu ra chung, dựa trên tín hiệu điều khiển.

Câu 3.

a) Hiện thực bộ Decoder cho Led 7 đoạn (7-segment) có chức năng giải mã giá trị nhị phân 3 bit như trong bảng sự thật bên dưới:



$c_2c_1c_0$	Character
000	H
001	E
010	L
011	L
100	O
101	
110	
111	

Tín hiệu nhập là ba bit c2,c1,c0 nối vào các SW[17:15]. Tín hiệu xuất ra led 7 đoạn HEX0[6:0].

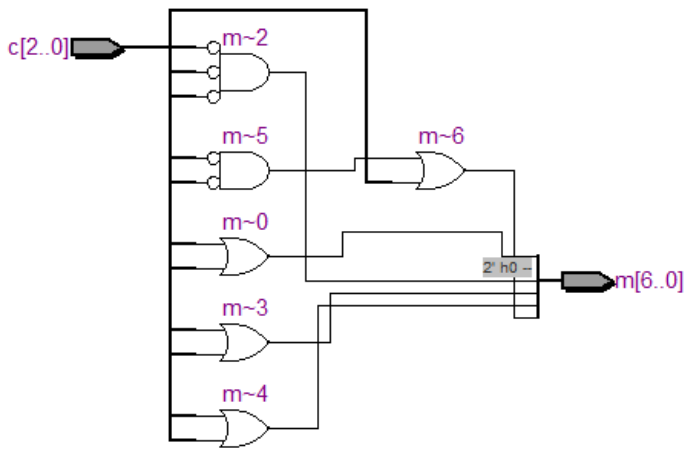
Bảng trạng thái và rút gọn:

c2	c1	c0	m6	m5	m4	m3	m2	m1	m0
0	0	0	0	0	0	1	0	0	1
0	0	1	0	0	0	0	1	1	0
0	1	0	1	0	0	0	1	1	1
0	1	1	1	0	0	0	1	1	1
1	0	0	1	0	0	0	0	0	0
1	0	1	X	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X	X
			c2 + c1	0	0	~c2~c1~c0	c1 + c0	c1+c0	~c2~c0 + c1

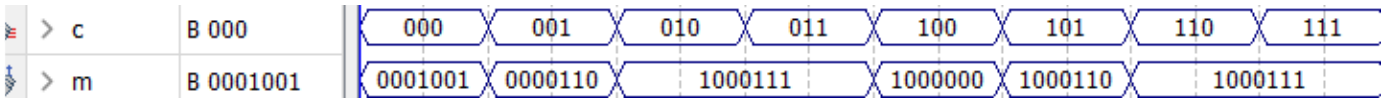
Code Thực Thi:

```
1   module BCDto7SegmentDecoder(c, m);
2   input [2:0] c;
3   output [6:0] m;
4   assign m[6] = (c[2] | c[1]);
5   assign m[5] = 0;
6   assign m[4] = 0;
7   assign m[3] = ~c[2] & ~c[1] & ~c[0];
8   assign m[2] = (c[1] | c[0]);
9   assign m[1] = (c[1] | c[0]);
10  assign m[0] = (~c[2] & ~c[0]) | c[1];
11  endmodule
```

RTL Viewer:



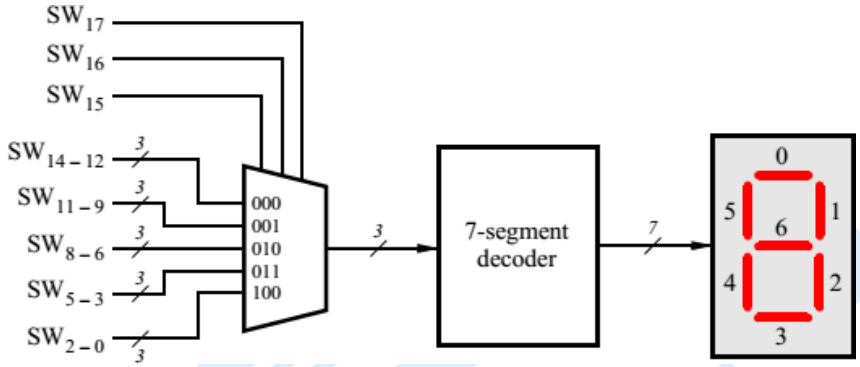
Mô phỏng Wareform:



Chức năng của Mạch:

- Input nhận đầu vào là một số nhị phân 3 bit (000 đến 111) để biểu diễn các chứ H, E, L, L, O trong mã BCD. Trong khi đó, các chữ được biểu diễn từ trạng thái 000 – 100, còn lại không biểu diễn (Random Value)
- Dựa trên giá trị của 3 bit đầu vào, mạch sẽ giải mã và tạo ra các tín hiệu 7 bit tương ứng để hiển thị chữ số tương ứng trên một màn hình 7-Segment.
- Ví dụ:
  - Khi đầu vào là 000, mạch sẽ tạo ra tín hiệu 7 bit 0001001 để hiển thị chữ số H trên 7-Segment.
  - Khi đầu vào là 100,mạch sẽ tạo ra tín hiệu 7 bit 1000000 để hiển thị chữ O trên 7-Segment.
  - Và tương tự cho các chữ số khác.

b) Kết hợp với mạch trong câu **a** và mạch trong câu 3 để thiết kế mạch sau:

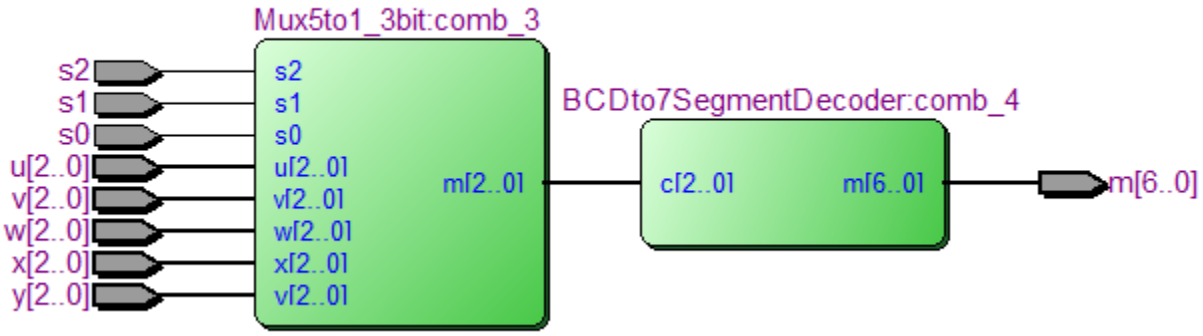


Chọn lựa các giá trị Switch SW[14:0] thích hợp để HEX0 hiển thị lần lượt các chữ H, E, L, L, O khi chuyển các switch SW[17:15].

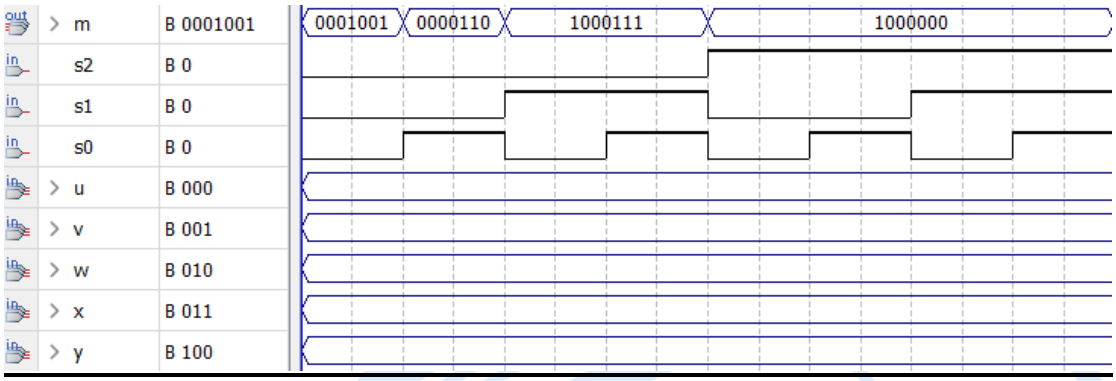
Code Thực Thi:

```
1   module Mix(s2, s1, s0, u, v, w, x, y, m);
2   input s2, s1, s0;
3   input [2:0] u;
4   input [2:0] v;
5   input [2:0] w;
6   input [2:0] x;
7   input [2:0] y;
8   wire [2:0] c;
9   output [6:0] m;
10  Mux5to1_3bit(s2, s1, s0, u, v, w, x, y, c);
11  BCDto7SegmentDecoder(c, m);
12  endmodule
```

RTL Viewer:



Mô phỏng Wareform:



Chức năng của Mạch:

- Mạch kết hợp mạch Mux5to1 3 bit từ câu a và bộ BCD7SegmentDecoder để tạo thành mạch hoàn chỉnh.
- Nhận các đầu vào là các tín hiệu s2, s1, s0, u[2:0], v[2:0], w[2:0], x[2:0], y[2:0].
- Sử dụng các tín hiệu s2, s1, s0 để lựa chọn (Mux) một trong 8 tín hiệu đầu vào u, v, w, x, y và gán cho tín hiệu c[2:0].
- Sử dụng tín hiệu c[2:0] làm đầu vào cho một mạch giải mã BCD sang 7-Segment (BCDto7SegmentDecoder) để tạo ra tín hiệu đầu ra m[6:0] theo yêu cầu đề bài.
- Tín hiệu đầu ra m[6:0] là tín hiệu hiển thị trên một màn hình 7-Segment.

c) Sử dụng thêm các HEX4, HEX3, HEX2, HEX1 và HEX0 để hiển thị mạch theo bảng sau:

SW_{17} SW_{16} SW_{15}	Character pattern				
	HEX4	HEX3	HEX2	HEX1	HEX0
000	H	E	L	L	O
001	E	L	L	O	H
010	L	L	O	H	E
011	L	O	H	E	L
100	O	H	E	L	L

Nạp mạch xuống KIT DE2 để kiểm tra kết quả.

Gợi ý: sử dụng 5 bộ (Mux + Decoder) trong câu b và kết nối các SW hợp lí.

Bảng trạng thái và rút gọn:

HEX4:

m6	m5	m4	m3	m2	m1	m0
0	0	0	1	0	0	1
0	0	0	0	1	1	0
1	0	0	0	1	1	1
1	0	0	0	1	1	1
1	0	0	0	0	0	0
X	X	X	X	X	X	X
X	X	X	X	X	X	X
X	X	X	X	X	X	X
c2 + c1	0	0	~c2~c1~c0	c1 + c0	c1+c0	~c2~c0 + c1

HEX3:

m6	m5	m4	m3	m2	m1	m0
0	0	0	0	1	1	0
1	0	0	0	1	1	1
1	0	0	0	1	1	1
1	0	0	0	0	0	0
0	0	0	1	0	0	1
X	X	X	X	X	X	X
X	X	X	X	X	X	X
X	X	X	X	X	X	X
c1+c0	0	0	c2	~c2~c0 + ~c2~c1	~c2~c0 + ~c2~c1	~c1c0 + c1~c0 + c2

HEX2:

m6	m5	m4	m3	m2	m1	m0
1	0	0	0	1	1	1
1	0	0	0	1	1	1
1	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	0	0	1	1	0
X	X	X	X	X	X	X
X	X	X	X	X	X	X
X	X	X	X	X	X	X
c2~c0 + ~c2~c1	0	0	c1c0	~c1	~c1	~c2~c1 + c0

HEX1:

c2	c1	c0	m6	m5	m4	m3	m2	m1	m0
0	0	0	1	0	0	0	1	1	1
0	0	1	1	0	0	0	0	0	0
0	1	0	0	0	0	1	0	0	1
0	1	1	0	0	0	0	1	1	0
1	0	0	1	0	0	0	1	1	1
1	0	1	X	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X	X
			~c1	0	0	c1~c0	~c1~c0 + c1c0	~c1~c0 + c1c0	~c0

HEX0:

c2	c1	c0	m6	m5	m4	m3	m2	m1	m0
0	0	0	1	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	1
0	1	1	0	0	0	0	1	1	0
0	1	0	1	0	0	0	1	1	1
1	0	0	1	0	0	0	1	1	1
1	0	1	X	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X	X
			~c1~c0 + c1c0	0	0	~c1c0	c2 + c1	c2 + c1	c2 + c0

Code Thực Thi:

HEX4 giống với câu a.

HEX3:

```
module BCDto7SegmentDecoder1(c, m);
input [2:0] c;
output [6:0] m;
assign m[6] = c[1] | c[0];
assign m[5] = 0;
assign m[4] = 0;
assign m[3] = c[2];
assign m[2] = (~c[2] & ~c[0]) | (~c[2] & ~c[1]);
assign m[1] = (~c[2] & ~c[0]) | (~c[2] & ~c[1]);
assign m[0] = (~c[1] & c[0]) | (c[1] & ~c[0]) | c[2];
endmodule
```

HEX2:

```
module BCDto7SegmentDecoder2(c, m);
input [2:0] c;
output [6:0] m;
assign m[6] = (~c[2] & ~c[0]) | (~c[2] & ~c[1]);
assign m[5] = 0;
assign m[4] = 0;
assign m[3] = c[1] & c[0];
assign m[2] = ~c[1];
assign m[1] = ~c[1];
assign m[0] = (~c[2] & ~c[1]) | c[0];
endmodule
```

HEX1

```
module BCDto7SegmentDecoder3(c, m);
input [2:0] c;
output [6:0] m;
assign m[6] = ~c[1];
assign m[5] = 0;
assign m[4] = 0;
assign m[3] = c[1] & ~c[0];
assign m[2] = (~c[1] & ~c[0]) | (c[1] & c[0]);
assign m[1] = (~c[1] & ~c[0]) | (c[1] & c[0]);
assign m[0] = ~c[0];
endmodule
```

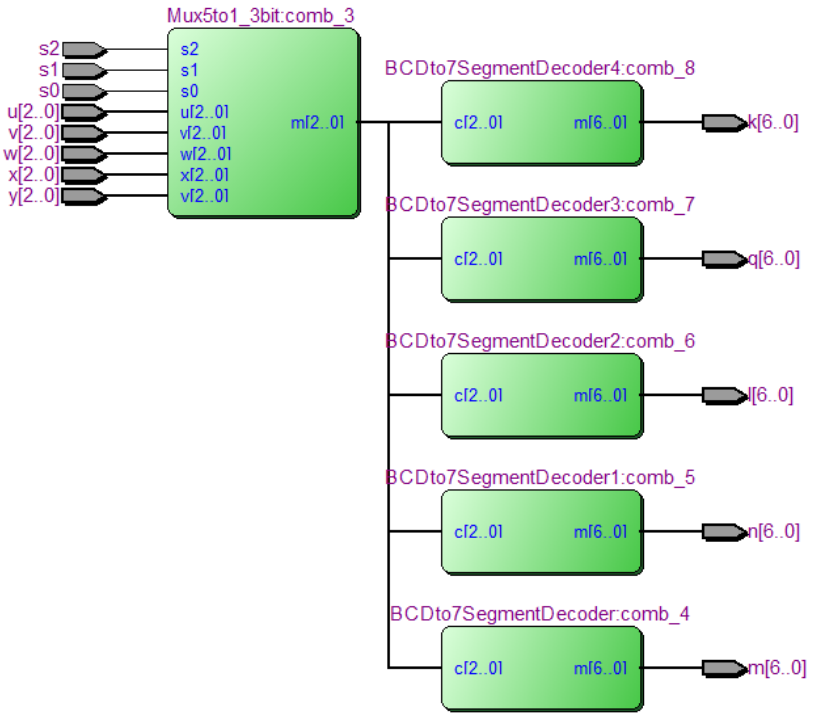
HEX0:

```
module BCDto7SegmentDecoder4(c, m);
input [2:0] c;
output [6:0] m;
assign m[6] = (~c[1] & ~c[0]) | (c[1] & c[0]);
assign m[5] = 0;
assign m[4] = 0;
assign m[3] = ~c[1] & c[0];
assign m[2] = c[2] | c[1];
assign m[1] = c[2] | c[1];
assign m[0] = c[2] | c[0];
endmodule
```

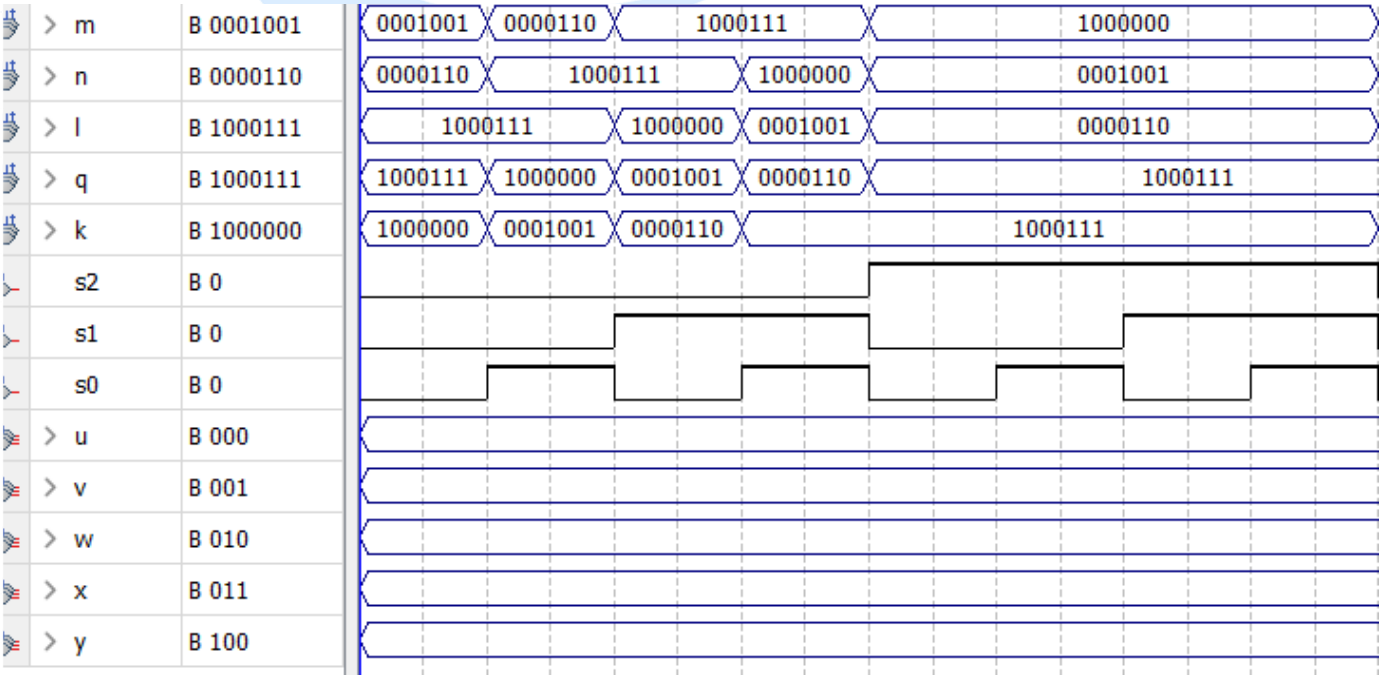
Hoàn chỉnh:

```
module Mix1(s2, s1, s0, u, v, w, x, y, m, n, l, q, k);
input s2, s1, s0;
input [2:0] u;
input [2:0] v;
input [2:0] w;
input [2:0] x;
input [2:0] y;
wire [2:0] c;
output [6:0] m;
output [6:0] n;
output [6:0] l;
output [6:0] q;
output [6:0] k;
Mux5to1_3bit(s2, s1, s0, u, v, w, x, y, c);
BCDto7SegmentDecoder(c, m);
BCDto7SegmentDecoder1(c, n);
BCDto7SegmentDecoder2(c, l);
BCDto7SegmentDecoder3(c, q);
BCDto7SegmentDecoder4(c, k);
endmodule
```

RTL Viewer:



Mô phỏng Wareform:



Chức năng của mạch:

- Nhận 5 đầu vào 3 bit (u, v, w, x, y) và 3 tín hiệu điều khiển s[2], s[1], s[0].
- Tạo ra 5 đầu ra 7 bit (k, l, m, n, q).
- Sử dụng các tín hiệu điều khiển s[2:0] để lựa chọn (Mux) một trong 5 tín hiệu đầu vào u, v, w, x, y và gán cho các đầu ra tương ứng.
- Cụ thể:
  - Khi s[2:0] = 000, mạch sẽ chọn u[2:0] và gán cho:
    - m[6:0] = 0001001 (ký tự 'H')
    - n[6:0] = 0000110 (ký tự 'E')
    - l[6:0] = 1000111 (ký tự 'L')
    - q[6:0] = 1000111 (ký tự 'L')
    - k[6:0] =1000000 (ký tự 'O')
  - Tương tự với các giá trị khác của s[2:0], mạch sẽ lựa chọn các đầu vào khác và tạo ra các chuỗi ký tự tương ứng.
- Khi s[2:0] = 101, mạch sẽ tạo ra chuỗi ký tự "HELLO" trên các đầu ra.

➔ Mạch này có chức năng lựa chọn và hiển thị một trong 5 chuỗi ký tự (O, L, H, E, L hoặc HELLO) dựa trên giá trị của tín hiệu điều khiển s[2:0].

d) Sử dụng hết 8 Led 7 đoạn trên KIT DE2 để thiết kế mạch hiển thị theo bảng sau

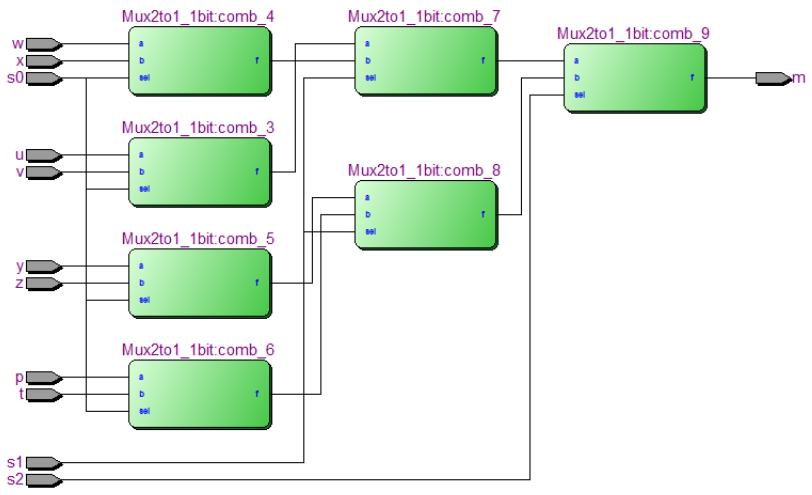
	Character pattern							
SW_{17} SW_{16} SW_{15}	H7	H6	H5	H4	H3	H2	H1	H0
000				H	E	L	L	O
001			H	E	L	L	O	
010		H	E	L	L	O		
011	H	E	L	L	O			
100	E	L	L	O				H
101	L	L	O				H	E
110	L	O				H	E	L
111	O				H	E	L	L

Mạch trên yêu cầu 1 bộ Mux8to1_1bit, nên ta sẽ thiết kế như sau:

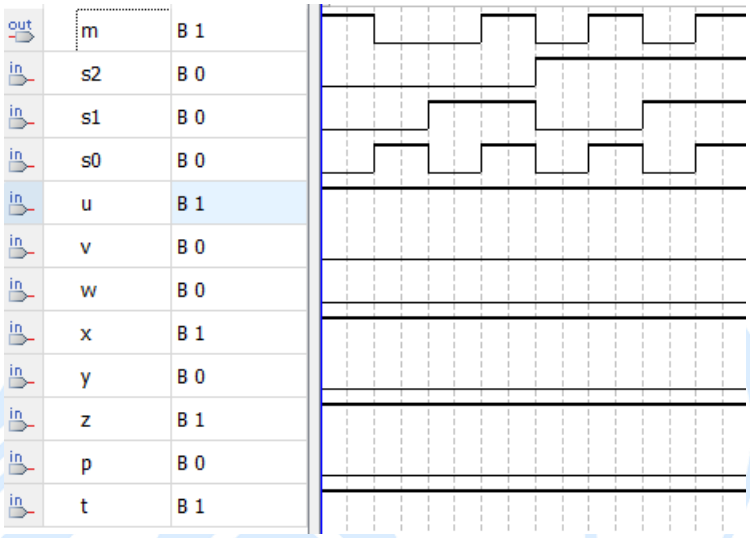
Code Thực Thi:

```
1   module Mux8to1_1bit(s2, s1, s0, u, v, w, x, y, z, p, t, m);
2   input s2, s1, s0, u, v, w, x, y, z, p, t;
3   output m;
4   wire uv, wx, yz, pt, uvwx, yzpt;
5   Mux2to1_1bit(uv, u, v, s0);
6   Mux2to1_1bit(wx, w, x, s0);
7   Mux2to1_1bit(yz, y, z, s0);
8   Mux2to1_1bit(pt, p, t, s0);
9   Mux2to1_1bit(uvwx, uv, wx, s1);
10  Mux2to1_1bit(yzpt, yz, pt, s1);
11  Mux2to1_1bit(m, uvwx, yzpt, s2);
12  endmodule
```

RTL Viewer:



Mô phỏng Wareform:

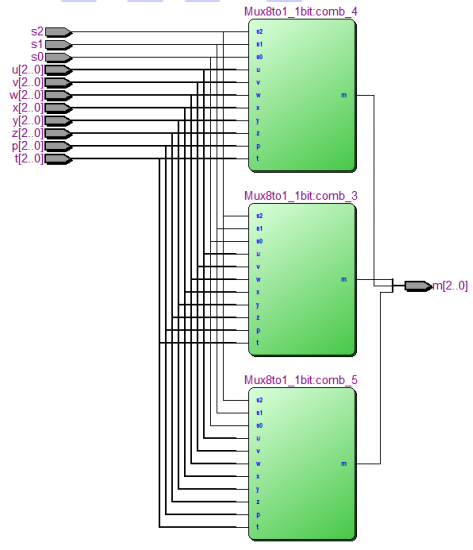


Từ đó, ta có mạch **Mux8to1_3bit:**

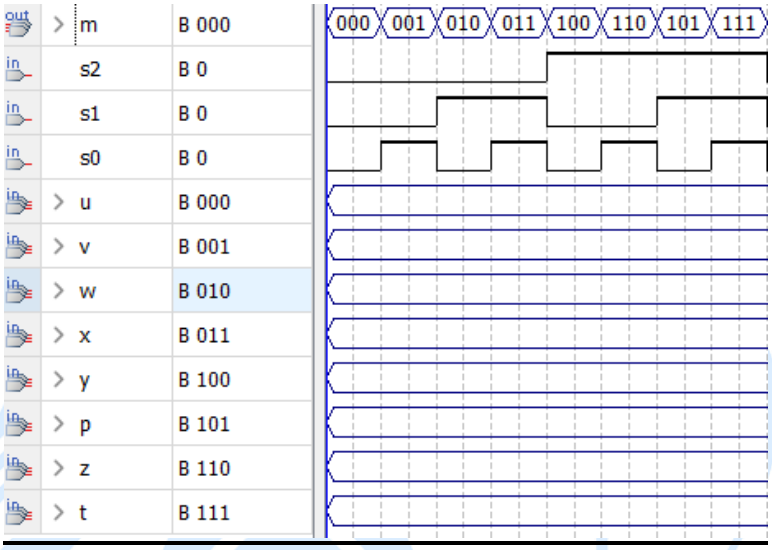
Code Thực Thi:

```
1   module Mux8to1_3bit(s2, s1, s0, u, v, w, x, y, z, p, t, m);
2   input s2, s1, s0;
3   output [2:0] m;
4   input [2:0] u;
5   input [2:0] v;
6   input [2:0] w;
7   input [2:0] x;
8   input [2:0] y;
9   input [2:0] z;
10  input [2:0] p;
11  input [2:0] t;
12  Mux8to1_1bit(s2, s1, s0, u[2], v[2], w[2], x[2], y[2], z[2], p[2], t[2], m[2]);
13  Mux8to1_1bit(s2, s1, s0, u[1], v[1], w[1], x[1], y[1], z[1], p[1], t[1], m[1]);
14  Mux8to1_1bit(s2, s1, s0, u[0], v[0], w[0], x[0], y[0], z[0], p[0], t[0], m[0]);
15  endmodule
```

RTL Viewer:



Mô phỏng Wareform:



Tiếp theo, ta tiến hành thiết kế Mạch theo yêu cầu:

Bảng Trạng Thái và Rút gọn:

H7:

c2	c1	c0	m6	m5	m4	m3	m2	m1	m0
0	0	0	X	X	X	X	X	X	X
0	0	1	X	X	X	X	X	X	X
0	1	0	X	X	X	X	X	X	X
0	1	1	0	0	0	1	0	0	1
1	0	0	0	0	0	0	1	1	0
1	0	1	1	0	0	0	1	1	1
1	1	0	1	0	0	0	1	1	1
1	1	1	1	0	0	0	0	0	0
			c2c0 + c1~c0	0	0	~c2	~c1 + ~c0	~c1 + ~c0	c1~c0 + ~c1c0 + ~c2

H6:

c2	c1	c0	m6	m5	m4	m3	m2	m1	m0
0	0	0	X	X	X	X	X	X	X
0	0	1	X	X	X	X	X	X	X
0	1	0	0	0	0	1	0	0	1
0	1	1	0	0	0	0	1	1	0
1	0	0	1	0	0	0	1	1	1
1	0	1	1	0	0	0	1	1	1
1	1	0	1	0	0	0	0	0	0
1	1	1	X	X	X	X	X	X	X
			c2	0	0	~c2~c0	~c1 + c0	~c1 + c0	~c2~c0 + ~c1

H5:

c2	c1	c0	m6	m5	m4	m3	m2	m1	m0
0	0	0	X	X	X	X	X	X	X
0	0	1	0	0	0	1	0	0	1
0	1	0	0	0	0	0	1	1	0
0	1	1	1	0	0	0	1	1	1
1	0	0	1	0	0	0	1	1	1
1	0	1	1	0	0	0	0	0	0
1	1	0	X	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X	X
			c1c0 + c2	0	0	~c2~c1	c1 + ~c0	c1 + ~c0	~c2c0 + ~c1~c0

H4:

c2	c1	c0	m6	m5	m4	m3	m2	m1	m0
0	0	0	0	0	0	1	0	0	1
0	0	1	0	0	0	0	1	1	0
0	1	0	1	0	0	0	1	1	1
0	1	1	1	0	0	0	1	1	1
1	0	0	1	0	0	0	0	0	0
1	0	1	X	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X	X
			c2 + c1	0	0	~c2~c1~c0	c1 + c0	c1 + c0	~c2~c0 + c1

H3:

c2	c1	c0	m6	m5	m4	m3	m2	m1	m0
0	0	0	0	0	0	0	1	1	0
0	0	1	1	0	0	0	1	1	1
0	1	0	1	0	0	0	1	1	1
0	1	1	1	0	0	0	0	0	0
1	0	0	X	X	X	X	X	X	X
1	0	1	X	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X	X
1	1	1	0	0	0	1	0	0	1
			~c2c1 + ~c2c0	0	0	c2	~c1 + ~c0	~c1 + ~c0	~c1c0 + c1~c0 + c2

H2:

c2	c1	c0	m6	m5	m4	m3	m2	m1	m0
0	0	0	1	0	0	0	1	1	1
0	0	1	1	0	0	0	1	1	1
0	1	0	1	0	0	0	0	0	0
0	1	1	X	X	X	X	X	X	X
1	0	0	X	X	X	X	X	X	X
1	0	1	X	X	X	X	X	X	X
1	1	0	0	0	0	1	0	0	1
1	1	1	0	0	0	0	1	1	0
			~c2	0	0	c2~c0	~c1 + c0	~c1 + c0	c2~c0 + ~c1

H1:

c2	c1	c0	m6	m5	m4	m3	m2	m1	m0
0	0	0	1	0	0	0	1	1	1
0	0	1	1	0	0	0	0	0	0
0	1	0	X	X	X	X	X	X	X
0	1	1	X	X	X	X	X	X	X
1	0	0	X	X	X	X	X	X	X
1	0	1	0	0	0	1	0	0	1
1	1	0	0	0	0	0	1	1	0
1	1	1	1	0	0	0	1	1	1
			c1c0 + ~c2	0	0	c2~c1	c1 + ~c0	c1 + ~c0	c2c0 + ~c2~c0

H0:

c2	c1	c0	m6	m5	m4	m3	m2	m1	m0
0	0	0	1	0	0	0	0	0	0
0	0	1	X	X	X	X	X	X	X
0	1	0	X	X	X	X	X	X	X
0	1	1	X	X	X	X	X	X	X
1	0	0	0	0	0	1	0	0	1
1	0	1	0	0	0	0	1	1	0
1	1	0	1	0	0	0	1	1	1
1	1	1	1	0	0	0	1	1	1
			~c2 + c1	0	0	c2~c1~c0	c1 + c0	c1 + c0	c2~c0 + c1

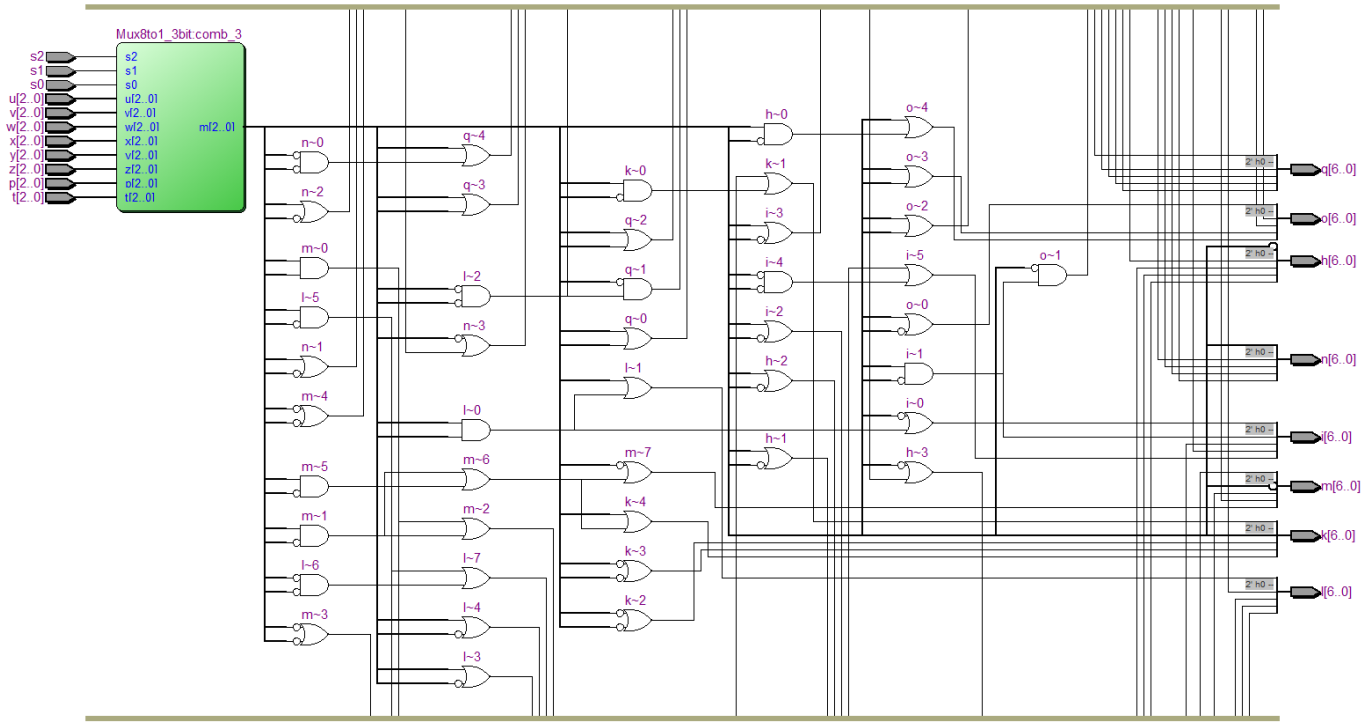
Code Thực Thi:

```
1  module Mix2(s2, s1, s0, u, v, w, x, y, z, p, t, m, n, l, q, k, h, i, o);
2  input s2, s1, s0;
3  input [2:0] u;
4  input [2:0] v;
5  input [2:0] w;
6  input [2:0] x;
7  input [2:0] y;
8  input [2:0] z;
9  input [2:0] p;
10 input [2:0] t;
11 wire [2:0] c;
12 output [6:0] m;
13 output [6:0] n;
14 output [6:0] l;
15 output [6:0] q;
16 output [6:0] k;
17 output [6:0] h;
18 output [6:0] i;
19 output [6:0] o;
20 Mux8to1_3bit(s2, s1, s0, u, v, w, x, y, z, p, t, c);
21 assign m[6] = (c[2] & c[0]) | (c[1] & ~c[0]);
22 assign m[5] = 0;
23 assign m[4] = 0;
24 assign m[3] = ~c[2];
25 assign m[2] = (~c[1] | ~c[0]);
26 assign m[1] = (~c[1] | ~c[0]);
27 assign m[0] = (c[1] & ~c[0]) | (~c[1] & c[0]) | ~c[2];
28 assign n[6] = c[2];
29 assign n[5] = 0;
30 assign n[4] = 0;
31 assign n[3] = ~c[2] & ~c[0];
32 assign n[2] = ~c[1] | c[0];
```

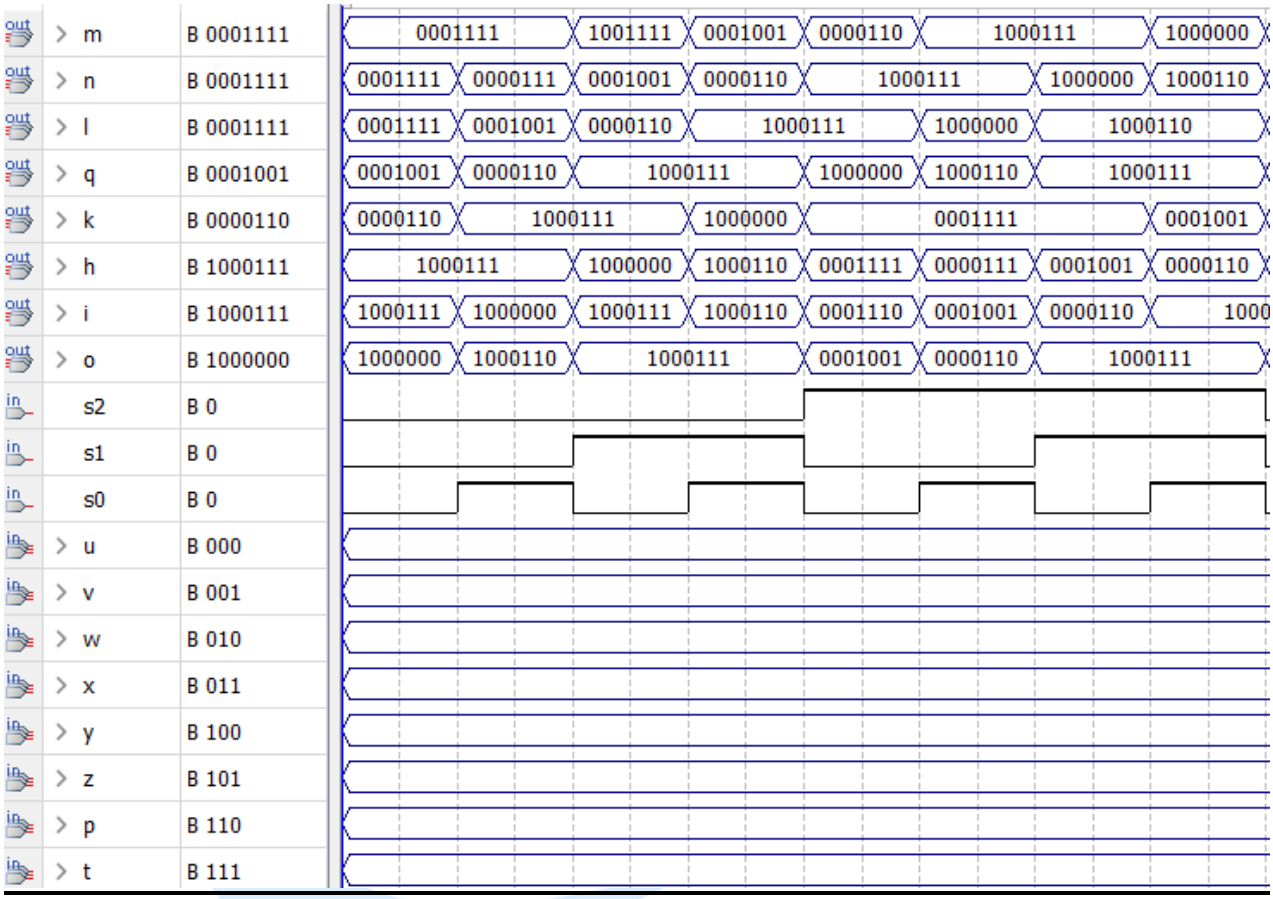
```
32  assign n[2] = ~c[1] | c[0];
33  assign n[1] = ~c[1] | c[0];
34  assign n[0] = (~c[2] & ~c[0]) | ~c[1];
35  assign l[6] = (c[1] & c[0]) | c[2];
36  assign l[5] = 0;
37  assign l[4] = 0;
38  assign l[3] = ~c[2] & ~c[1];
39  assign l[2] = c[1] | ~c[0];
40  assign l[1] = c[1] | ~c[0];
41  assign l[0] = (~c[2] & c[0]) | (~c[1] & ~c[0]);
42  assign q[6] = c[2] | c[1];
43  assign q[5] = 0;
44  assign q[4] = 0;
45  assign q[3] = ~c[2] & ~c[1] & ~c[0];
46  assign q[2] = c[1] | c[0];
47  assign q[1] = c[1] | c[0];
48  assign q[0] = (~c[2] & ~c[0]) | c[1];
49  assign k[6] = (~c[2] & c[1]) | (~c[2] & c[0]);
50  assign k[5] = 0;
51  assign k[4] = 0;
52  assign k[3] = c[2];
53  assign k[2] = ~c[1] | ~c[0];
54  assign k[1] = ~c[1] | ~c[0];
55  assign k[0] = (~c[1] & c[0]) | (c[1] & ~c[0]) | c[2];
56  assign h[6] = ~c[2];
57  assign h[5] = 0;
58  assign h[4] = 0;
59  assign h[3] = c[2] & ~c[0];
60  assign h[2] = ~c[1] | c[0];
61  assign h[1] = ~c[1] | c[0];
62  assign h[0] = (c[2] & ~c[0]) | ~c[1];
```

```
63  assign i[6] = (c[1] & c[0]) | ~c[2];
64  assign i[5] = 0;
65  assign i[4] = 0;
66  assign i[3] = c[2] & ~c[1];
67  assign i[2] = c[1] | ~c[0];
68  assign i[1] = c[1] | ~c[0];
69  assign i[0] = (c[2] & c[0]) | (~c[2] & ~c[0]);
70  assign o[6] = ~c[2] | c[1];
71  assign o[5] = 0;
72  assign o[4] = 0;
73  assign o[3] = c[2] & ~c[1] & ~c[0];
74  assign o[2] = c[1] | c[0];
75  assign o[1] = c[1] | c[0];
76  assign o[0] = (c[2] & ~c[0]) | c[1];
77  endmodule
```

RTL Viewer:



Mô phỏng Wareform:



Chức năng của mạch:

- Cũng giống như câu b nhưng ở đây ta sẽ xài bộ Mux8to1_3bit để chọn 1 trong 8 đầu ra theo yêu cầu đề. Mạch có chức năng chon 1 trong 8 đầu vào (u, v, w, x, y, z, p, t) và cho ra 8 output 6 bit tương ứng (h, I, k, l, m, n, o, q). Sau khi đến trạng thái cuối cùng mạch sẽ hiển thị chữ theo như yêu cầu đề bài.

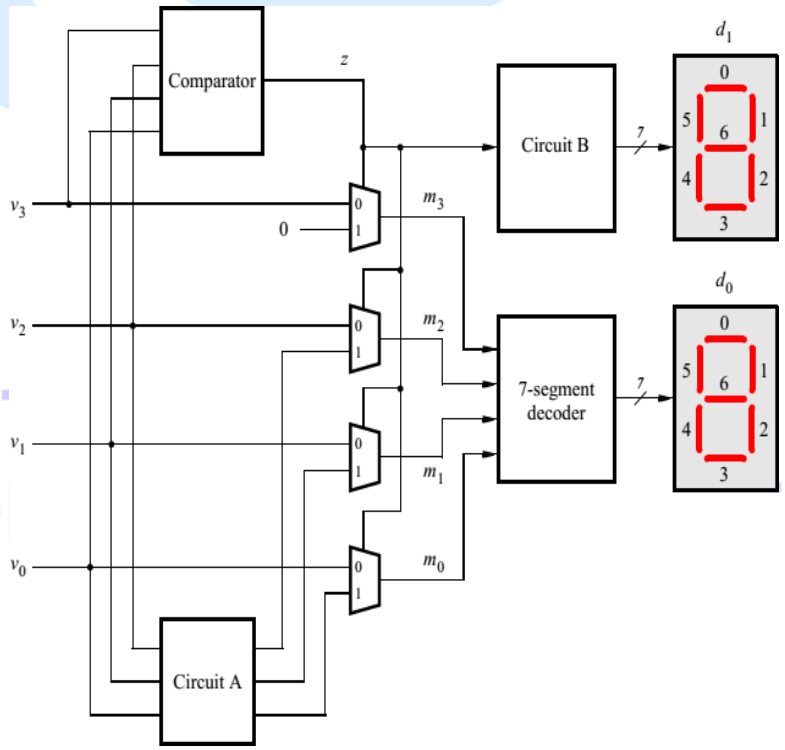
Câu 4.

Thiết kế mạch chuyển đổi số Binary sang số Decimal như bảng sau:

Binary value SW[3:0]	Decimal digits HEX1	HEX0
0000	0	0
0001	0	1
0010	0	2
...	...	...
1001	0	9
1010	1	0
1011	1	1
1100	1	2
1101	1	3
1110	1	4
1111	1	5

Gợi ý: Mạch chuyển đổi có thể được thiết kế theo sơ đồ bên dưới.

Mạch Comparator để kiểm tra số nhập với 9. Ngõ ra Comparator z=0 nếu số nhỏ hơn 9, và z=1 nếu số lớn hơn 9.



Đầu tiên, thiết kế bộ **Comparator:**

Code Thực Thi:

```
module Comparator(c, z);
input [3:0] c;
output z;
assign z = (c[3] & c[1]) | (c[3] & c[2]);
endmodule
```

Tiếp theo, thiết kế bộ **BCDto7SegmentDecoder** theo yêu cầu bài toán:

Bảng trạng thái:

c3	c2	c1	c0	m6	m5	m4	m3	m2	m1	m0
0	0	0	0	1	0	0	0	0	0	0
0	0	0	1	1	1	1	1	0	0	1
0	0	1	0	0	1	0	0	1	0	0
0	0	1	1	0	1	1	0	0	0	0
0	1	0	0	0	0	1	1	0	0	1
0	1	0	1	0	0	1	0	0	1	0
0	1	1	0	0	0	0	0	0	1	0
0	1	1	1	1	1	1	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0	0	0	0
1	0	1	0	X	X	X	X	X	X	X
1	0	1	1	X	X	X	X	X	X	X
1	1	0	0	X	X	X	X	X	X	X
1	1	0	1	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X
				~c3~c2~c1 + c2c1c0	~c3~c2c0 + ~c2c1 + c1c0	c2~c1 + c0	~c3~c2~c1c0 + c2~c1~c0 + c2c1c0	~c2c1~c0	c2~c1c0 + c2c1~c0	~c3~c2~c1c0 + c2~c1~c0

Code Thực Thi:

```
module BCDto7SegmentDecoder5(c,m);
input [3:0] c;
output [6:0] m;
assign m[6] = (~c[3] & ~c[2] & ~c[1]) | (c[2] & c[1] & c[0]);
assign m[5] = (~c[3] & ~c[2] & c[0]) | (~c[2] & c[1]) | (c[1] & c[0]);
assign m[4] = (c[2] & ~c[1]) | c[0];
assign m[3] = (~c[3] & ~c[2] & ~c[1] & c[0]) | (c[2] & ~c[1] & ~c[0]) | (c[2] & c[1] & c[0]);
assign m[2] = ~c[2] & c[1] & ~c[0];
assign m[1] = (c[2] & ~c[1] & c[0]) | (c[2] & c[1] & ~c[0]);
assign m[0] = (~c[3] & ~c[2] & ~c[1] & c[0]) | (c[2] & ~c[1] & ~c[0]);
endmodule
```

Sau đó, thiết kế hai bộ **Circuit A** và **Circuit B**:

Code Thực Thi:

```
module CircuitA(c,m);
input [2:0] c;
output [2:0] m;
assign m[2] = c[2] & c[1];
assign m[1] = c[2] & ~c[1];
assign m[0] = c[0];
endmodule
```

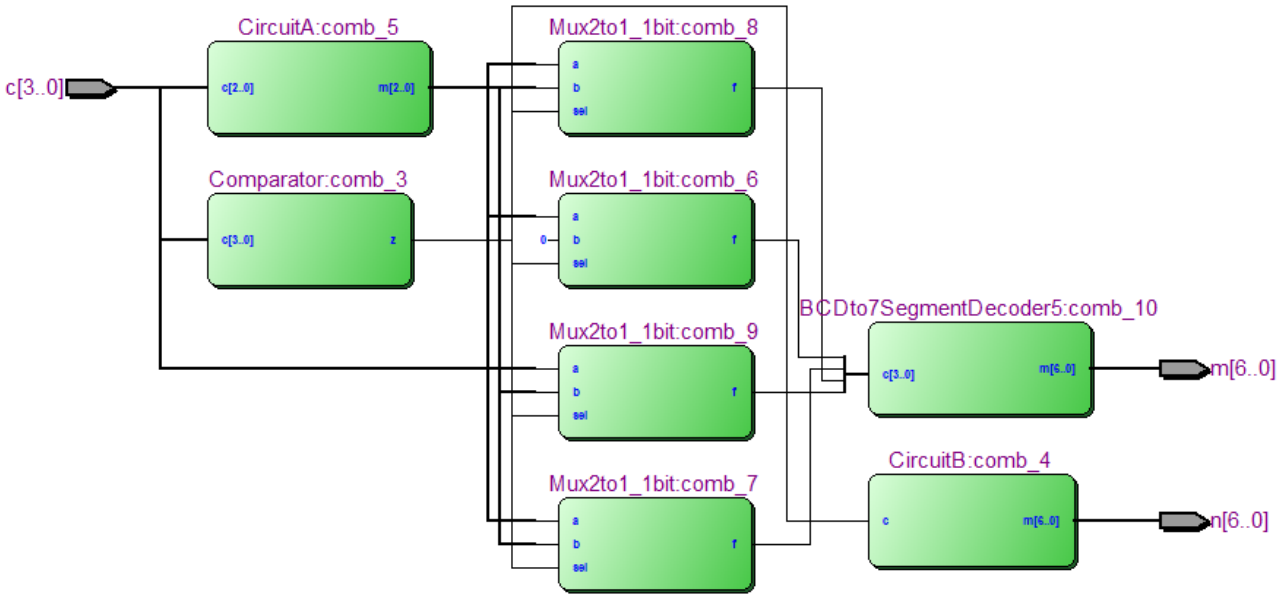
```
module CircuitB(c, m);
    input c;
    output reg [6:0] m;

    always @(*) begin
        m = (c == 0) ? 7'b1000000 : 7'b1111001;
    end
endmodule
```

Cuối cùng ta được 1 bộ mạch hoàn chỉnh theo yêu cầu đề bài:

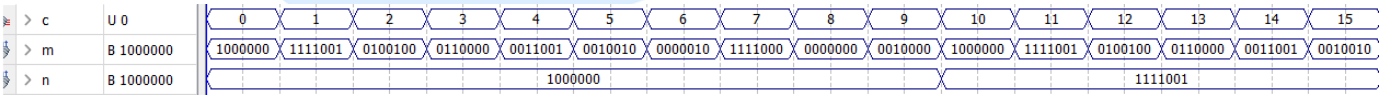
Code Thực Thi:

```
module Mix3(c,m,n);
input [3:0] c;
output [6:0] m;
output [6:0] n;
wire z;
Comparator(c,z);
CircuitB(z,n);
wire [2:0] q;
CircuitA(c[2:0], q);
wire [3:0] l;
Mux2to1_1bit(l[3], c[3], 0, z);
Mux2to1_1bit(l[2], c[2], q[2], z);
Mux2to1_1bit(l[1], c[1], q[1], z);
Mux2to1_1bit(l[0], c[0], q[0], z);
BCDto7SegmentDecoder5(l,m);
endmodule
```

RTL Viewer:



Mô phỏng Wareform:



Chức năng của Mạch:

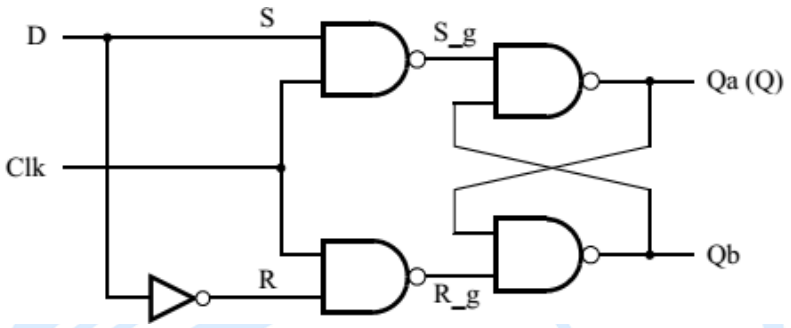
- Mạch kết hợp Comparator, Circuit A và Circuit B, 4 bộ Mux2to1_1bit và 1 bộ BCD7SegmentDecoder để hoàn thành yêu cầu bài.
- Nhận 1 input c[4..0], sau đó cho ra output m[6..0] để hiển thị giá trị đơn vị và n[6..0] để hiển thị hàng chục, cụ thể như sau:
  - Khi đầu vào thực hiện giá trị từ 0 đến 9, output m sẽ cho ra giá trị hiển thị tương ứng và khi đó n sẽ luôn có giá trị trên BCD là 0 (01->09).
  - Khi đầu vào thực hiện giá trị từ 10 đến 15, output m sẽ cho ra giá trị hiển thị là 0 đến 5 và khi đó n sẽ mang giá trị 1 ứng với hang chục (10->15)

➔ Như vậy, mạch thực hiện chức năng hiển thị số tương ứng từ 1 đến 15, kết hợp với n để hiển thị giá trị hang chục thêm trực quan và sinh động.

➔

Câu 5.

a) Thực hiện mạch D-latch có sơ đồ như sau:



Yêu cầu: Kiểm tra bằng mô phỏng sau đó nạp mạch xuống KIT DE2 và kiểm tra kết quả.

Gợi ý gán chân:

-D = SW[0]

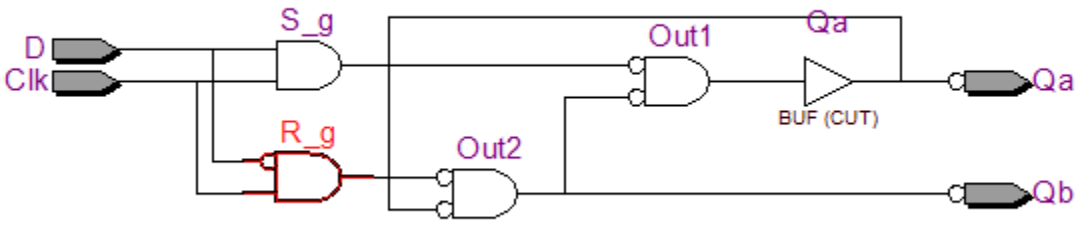
-Clk = SW[1]

-Q = LEDR[0]

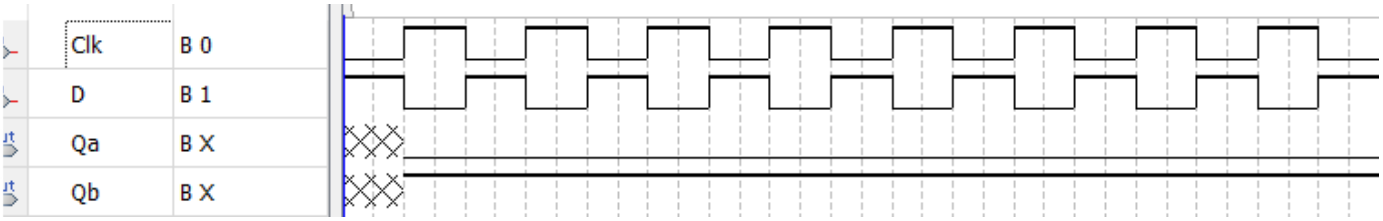
Code Thực Thi:

```
1  module D_Latch(D, Clk, Qa, Qb);
2  input D, Clk;
3  output Qa, Qb;
4  not g1(nD, D);
5  nand S_g(f1, D, Clk);
6  nand R_g(f2, nD, Clk);
7  nand Out1(Qa, f1, Qb);
8  nand Out2(Qb, f2, Qa);
9  endmodule
```

RTL Viewer:



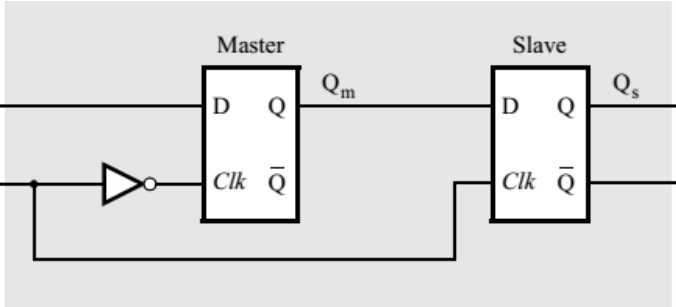
Mô phỏng Wareform:



Chức năng của Mạch:

- Đây là một mạch D Latch cơ bản kích theo cạnh lên của CLK. Cụ thể:
  - Khi mạch tích cực lên, output Q sẽ mang giá trị của D
  - Khi mạch tích cực xuống, output Q sẽ mang giá trị trước đó của Q

b) Sử dụng D-latch trong câu a để thiết kế D-Flipflop kích cạnh lên theo dạng Master-Slave có sơ đồ như sau:

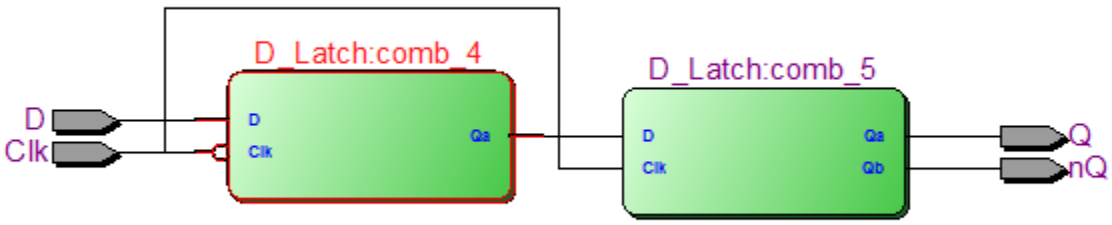


Yêu cầu: Kiểm tra bằng mô phỏng sau đó nạp mạch và kiểm tra trên KIT DE2

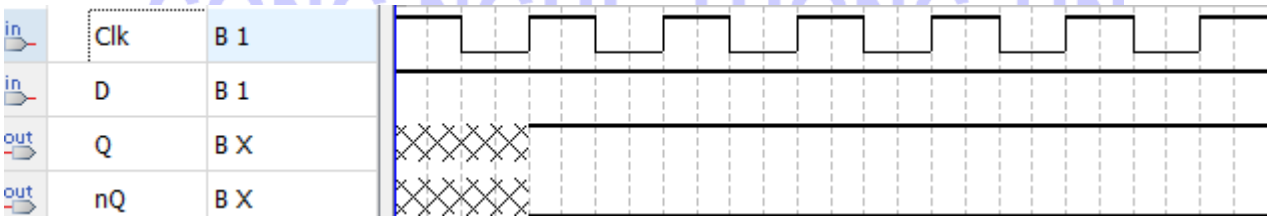
Code Thực Thi:

```
1  module Master_Slave(D, Clk, Q, nQ);
2  input D, Clk;
3  output Q, nQ;
4  not (nClk, Clk);
5  wire Qm, Qn;
6  D_Latch(D, nClk, Qm, Qn);
7  D_Latch(Qm, Clk, Q, nQ);
8  endmodule
```

RTL Viewer:



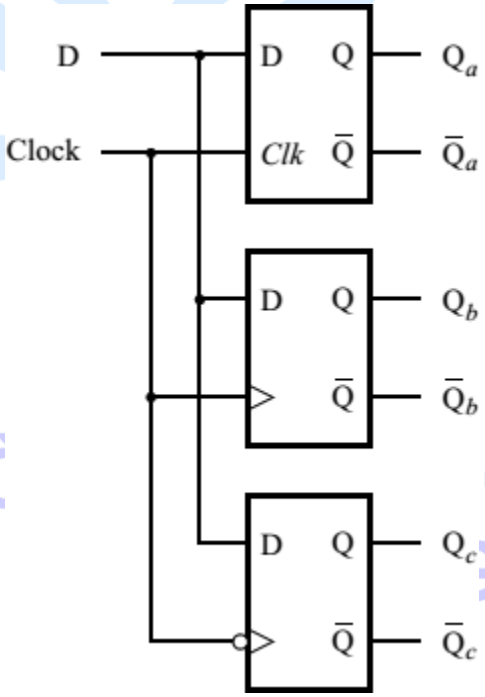
Mô phỏng Wareform:



Chức năng của Mạch:

- Clock chung cung cấp xung clock cho cả Master và Slave.
- Dữ liệu được đưa vào đầu D của Master.
- Đầu ra Q của Master (Qm) được kết nối với đầu D của Slave.
- Đầu ra Q của Slave (Qs) là đầu ra cuối cùng của hệ thống, cụ thể như sau:
    - Giả sử ở thời điểm ban đầu, đầu vào D của Master được đặt ở mức logic 1. Khi xung clock đầu tiên xuất hiện, D Flip-Flop của Master sẽ lưu trữ giá trị 1 vào đầu ra Qm.
    - Tại thời điểm này, Qm = 1 và được truyền vào đầu D của Slave. Khi xung clock tiếp theo xuất hiện, D Flip-Flop của Slave sẽ lưu trữ giá trị 1 vào đầu ra Qs.
    - Như vậy, sau 2 chu kỳ clock, giá trị 1 ban đầu ở đầu vào D của Master đã được truyền sang đầu ra Qs của Slave.
    - Nếu ở chu kỳ tiếp theo, đầu vào D của Master thay đổi sang giá trị 0, thì qua 2 chu kỳ clock tiếp theo, giá trị 0 này sẽ được truyền từ Master sang Slave và xuất hiện ở đầu ra Qs.
- Như vậy, mạch này thực hiện chức năng chuyển dữ liệu từ Master sang Slave theo từng chu kỳ clock, tạo ra một hệ thống Master-Slave đơn giản.

c) Thực hiện mạch sau:

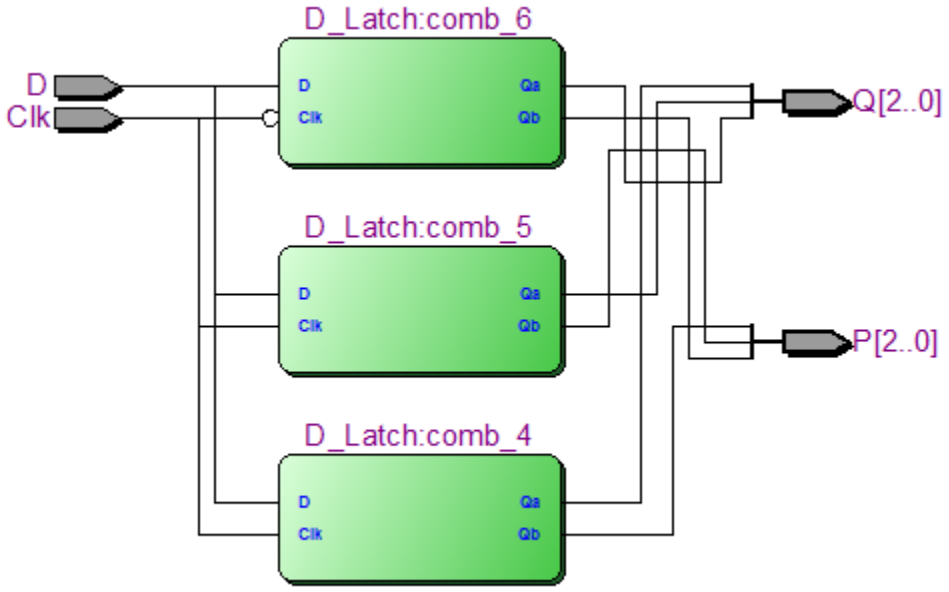


Yêu cầu: Thực hiện mô phỏng Vector Waveform để so sánh giá trị 3 ngõ ra Qa, Qb, Qc

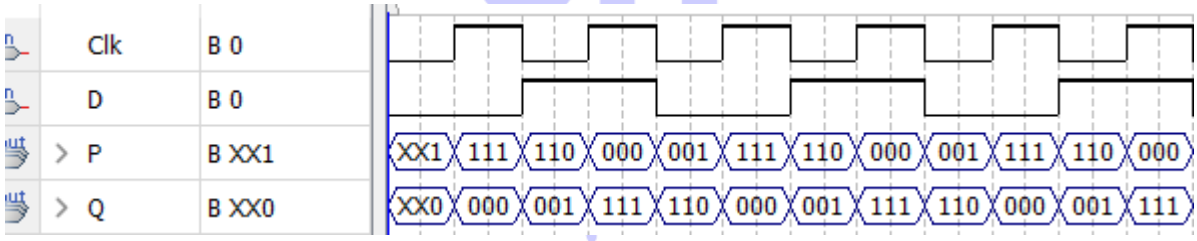
Code Thực Thi:

```
1  module DFF3(D, Clk, Q, P);
2  input D, Clk;
3  output [2:0] Q;
4  output [2:0] P;
5  not (nClk, Clk);
6  D_Latch(D, Clk, Q[2], P[2]);
7  D_Latch(D, Clk, Q[1], P[1]);
8  D_Latch(D, nClk, Q[0], P[0]);
9  endmodule
```

RTL Viewer:



Mô phỏng Wareform:



Chức năng:

- Clock chung cung cấp xung clock cho cả Master và Slave.
- Dữ liệu được đưa vào đầu D của Master.
- Đầu ra Q của Master (Qa) được kết nối với đầu D của Slave.
- Đầu ra Q của Slave (Qb, Qc) là đầu ra cuối cùng của hệ thống.

Ví dụ cụ thể:

- Giả sử ở thời điểm ban đầu, đầu vào D của Master là 1.
- Khi xung clock đầu tiên xuất hiện, D Flip-Flop của Master sẽ lưu trữ giá trị 1 vào Qa.
- Tại thời điểm này, Qa = 1 và được truyền vào đầu D của Slave.
- Khi xung clock tiếp theo xuất hiện, D Flip-Flop của Slave sẽ lưu trữ giá trị 1 vào Qb, Qc.
- Như vậy, sau 2 chu kỳ clock, giá trị 1 ban đầu ở đầu vào D của Master đã được truyền sang đầu ra Qb, Qc của Slave.

  ➔ Mạch này hoạt động như một hệ thống truyền dữ liệu tuần tự từ Master sang Slave, với độ trễ 2 chu kỳ clock.

Lưu ý: SV chỉ sử dụng lệnh Continuous assignment để thiết kế cho các bài tập trên.

TÀI LIỆU THAM KHẢO

Digital_Logic lab của **Altera**.