Bài Báo Cáo

Môn Thí Ngiệm Kĩ Thuật Số

Trường Đại học Bách Khoa TP HCM

TP.HCM --- 2013

Nhóm 3

Khoa: **Điện-Điện Tử**

Thành viên nhóm:

Họ Và Tên: MSSV

1. Tôn Thất Nguyên Phong……41102555
2. Lê Quang Sơn……41102924

Bài thí nghiệm 1

Switches, Lights, Multiplexers

1.Thínghiệm1.1:

Thực hiện mạch thí nghiệm có ngõ vào là 10 công tắc $SW9—0$, và ngõ ra là 10 đèn LED màu đỏ $LEDR9—0$ dùng để đọc trạng thái của các ngõ vào.

- Cácbướccầnthựchiện:

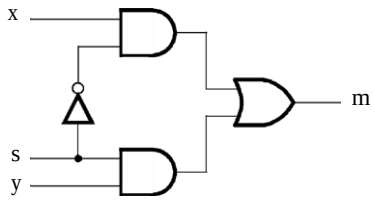
1. Tạo project mới.
2. Viết chương trình Verilog cho bài TN
3. Gán chân & biên dịch project.
4. Nạp project vào kit TN. Thử mạch.

-
- Chương trình của nhóm:

```
module  TN1_1 (SW, LEDR);
    input  [9:0] SW;
    output  [9:0] LEDR;
    assign LEDR = SW;
endmodule
```

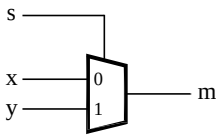
2.Thínghiệm1.2:

- Cho mạch multiplexer 2 sang 1 như hình 2 với ngõ vào chọn kênh s. Nếu s = 0 ngõ ra m sẽ bằng ngõ vào x, và nếu s = 1 thì ngõ ra m = y.



a) Sơ đồ mạch

s	m
0	x
1	y



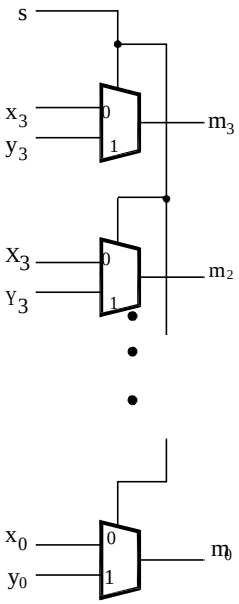
b) Bảng sự thật c) Ký hiệu

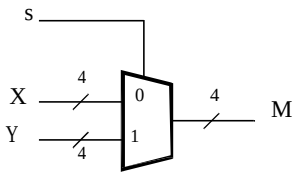
Hình 2. Mạch multiplexer 2

sang 1. Mạch có thể mô tả dùng mã Verilog như sau:

assign m = (s&x) (s& y);

- Dùng 4 bộ multiplexer 2 sang 1 như hình 2 để thực hiện mạch multiplexer 2 sang 1 - 4 bit như hình 3a. Mạch có 2 ngõ vào nhị phân 4 bit X và Y, và ngõ ra 4 bit M. Nếu s = 0 thì M = X, còn s = 1 thì M = Y.





- Cácbướccầnthựchiện:

1. Tạo project mới.
2. Viết chương trình Verilog với:
    i. s = SW_9 và nối với $LEDR_9$
    ii. X = SW_{3-0} và nối với $LEDR_{3-0}$
    iii. Y = SW_{7-4} và nối với $LEDR_{7-4}$
    iv. M = $LEDG_{3-0}$

2. Gán chân
3. Biên dịch project.
4. Nạp project vào kit TN.
5. Thử mạch bằng cách thay đổi các công tắc SW rồi theo dõi các đèn LED xanh, đỏ.

- Chương trình của nhóm:

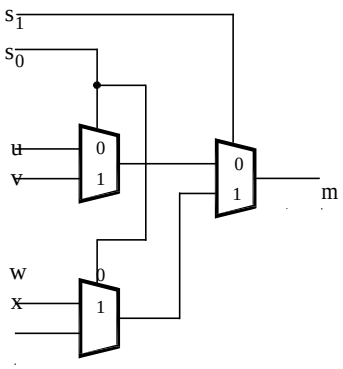
```
module TN1_2(SW,LEDR,LEDG);
input [9:0]SW;
output [9:0]LEDR;
output [3:0]LEDG;
assign LEDR=SW;
Mux21(SW[9],SW[0],SW[4],LEDG[0]);
Mux21(SW[9],SW[1],SW[5],LEDG[1]);
Mux21(SW[9],SW[2],SW[6],LEDG[2]);
Mux21(SW[9],SW[3],SW[7],LEDG[3]);
endmodule
module Mux21(S,X,Y,M);
input S,X,Y;
output M;
assign M=(~S&X)|(S&Y);
endmodule
```

3.Thínghiệm1.3:

- Dùng 3 bộ multiplexer 2 sang 1 như hình 2 để thực hiện mạch multiplexer 4 sang 1 như hình 4a.

  Mạch có 4 ngõ vào u, v, w và x; 1 ngõ ra m; 2 ngõ vào chọn kênh s_1 s_0

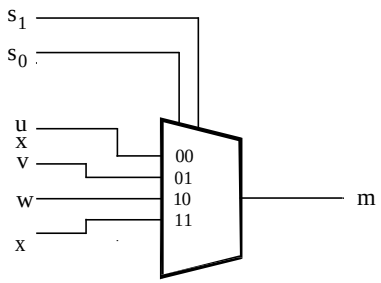
- Tương tự dùng 2 mạch multiplexer $4 \rightarrow 1$ như hình 4a để thực hiện mạch multiplexer $4 \rightarrow 1$ - 2 bit như hình 5



a) sơ đồ mạch

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

b) bảng sự thật



c) ký hiệu

Hình 4. Mạch multiplexer 4 sang 1

- Cácbướccầnthựchiện:

1. Tạo project mới.
2. Viết chương trình Verilog với:

   s_1 s_0 = SW9-8 và nối với LEDR9-8

   U-X = SW7-0 và nối với LEDR7-0

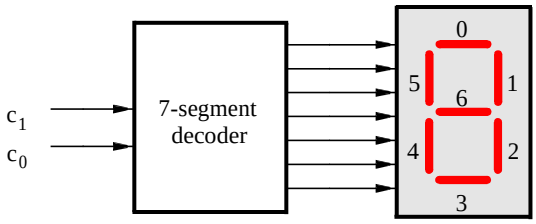
   M = LEDG1-0
3. Gán chân
4. Biên dịch project.
5. Nạp project vào kit TN.
6. Thử mạch bằng cách thay đổi các công tắc SW rồi theo dõi các đèn LED xanh, đỏ.

- Chương trình của nhóm:

```
module TN1_3(SW,LEDR,LEDG);
input [9:0]SW;
output [9:0]LEDR;
output [1:0]LEDG;
assign LEDR=SW;
Mux41(SW[8],SW[9],SW[0],SW[2],SW[4],SW[6],LEDG[0]);
Mux41(SW[8],SW[9],SW[1],SW[3],SW[5],SW[7],LEDG[1]);
endmodule
module Mux41(S0,S1,U,V,W,X,M);
input S0,S1,U,V,W,X;
output M;
wire t1,t0;
Mux21(S0,U,V,t0);
Mux21(S0,W,X,t1);
Mux21(S1,t0,t1,M);
endmodule
module Mux21(S,X,Y,M);
input S,X,Y;
output M;
assign M=(~S&X)|(S&Y);
endmodule
```

4.Thínghiệm1.4:

Thực hiện bộ giải mã có 2 ngõ vào c_1 c_0 và 7 ngõ ra từ 0 đến 6 dùng để hiển thị các ký tự trên bộ hiển thị 7 đoạn như hình 6.
Bảng 1 liệt kê các ký tự cần hiển thị (gồm H,E,L và ký tự O) tương ứng với các ngõ vào c_1 c_0. Các ngõ ra tích cực mức logic 0.



Hình 6. Bộ giải mã 7 đoạn

c1 c0	Ký tự
0 0	H
0 1	E
1 0	L
1 1	O

Bảng 1. Bảng mã chữ

- Cácbướccầnthựchiện:

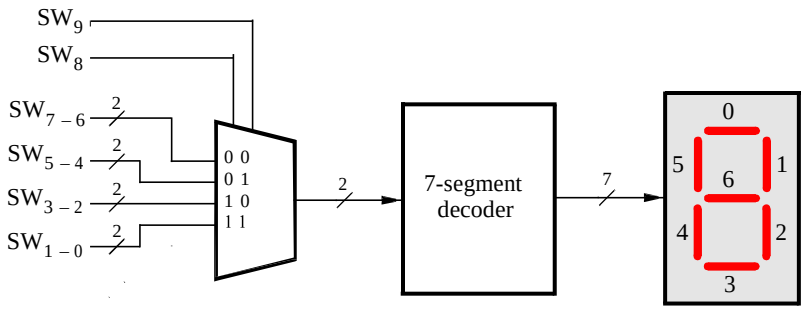
1. Tạo project mới.
2. Viết chương trình Verilog với:
   - Các ngõ vào c_1 c_0 nối với các công tắc SW1-0
   - Các ngõ ra 0 – 6 nối với HEX0$_0$, HEX0$_1$…..HEX0$_6$
3. Gán chân
4. Biên dịch project.
5. Nạp project vào kit TN.
6. Thử mạch bằng cách thay đổi các công tắc SW$_{1-0}$ rồi quan sát bộ hiển thị 7 đoạn.

- Chương trình của nhóm:

```
module TN1_4(SW,HEX0);
input [1:0]SW;
output [6:0]HEX0;
reg [6:0] HEX0;
always @ (SW[1:0])
begin
case (SW[1:0])
2'b00: HEX0= 7'b0001001;
2'b01: HEX0= 7'b0000110;
2'b10: HEX0= 7'b1000111;
2'b11: HEX0= 7'b1000000;
default: HEX0= 7'b1111111;
endcase
end
endmodule
```

5.Thínghiêm1.5:

Thực hiện mạch điện hiển thị "chữ xoay" như hình 7 hoạt động theo bảng 2.
Các công tắc $SW7-0$ dùng để tạo ký tự và $SW\,9-8$ dùng chọn ký tự hiển thị.



Hình 7. Mạch có thể chọn & hiển thị 1 trong 4 ký tự.

SW9	SW8	Hiển thị
0	0	H
0	1	E
1	0	L
1	1	O

Bảng 2. Hiển thị chữ xoay HELLO.

- Cácbướccầnthựchiện:

1. Tạo project mới.
2. Viết chương trình Verilog với:
3. Gán chân
4. Biên dịch project.
5. Nạp project vào kit TN.
6. Thử mạch bằng cách thay đổi các công tắc $SW1-0$ rồi quan sát bộ hiển thị 7 đoạn.

- Chương trình của nhóm:

```
module TN1_5(SW,HEX0,HEX1,HEX2,HEX3);

input [9:0]SW;

output [6:0]HEX0;

output [6:0]HEX1;

output [6:0]HEX2;

output [6:0]HEX3;
```

```
wire [1:0]A,B,C,D;
Mux412b(SW[9:8],SW[1:0],SW[7:6],SW[5:4],SW[3:2],A);
Mux412b(SW[9:8],SW[3:2],SW[1:0],SW[7:6],SW[5:4],B);
Mux412b(SW[9:8],SW[5:4],SW[3:2],SW[1:0],SW[7:6],C);
Mux412b(SW[9:8],SW[7:6],SW[5:4],SW[3:2],SW[1:0],D);
ganchu(A,HEX0);
ganchu(B,HEX1);
ganchu(C,HEX2);
ganchu(D,HEX3);
endmodule
module Mux412b(S,U,V,W,X,M);
input [1:0]S,U,V,W,X;
output [1:0]M;
Mux41(S[0],S[1],U[0],V[0],X[0],W[0],M[0]);
Mux41(S[0],S[1],U[1],V[1],X[1],W[1],M[1]);
endmodule
module Mux41(S0,S1,U,V,W,X,M);
input S0,S1,U,V,W,X;
output M;
wire t1,t0;
Mux21(S0,U,V,t0);
Mux21(S0,W,X,t1);
Mux21(S1,t0,t1,M);
endmodule
module Mux21(S,X,Y,M);
input S,X,Y;
output M;
assign M=(~S&X)|(S&Y);
endmodule
```

```
module ganchu(S,HE);
input [1:0]S;
output [6:0]HE;
reg [6:0] HE;
always @ (S[1:0])
begin
case (S[1:0])
2'b00: HE= 7'b0001001;
2'b01: HE= 7'b0000110;
2'b10: HE= 7'b1000111;
2'b11: HE= 7'b1000000;
default: HE= 7'b1111111;
endcase
end
endmodule
```

Bài thí nghiệm 2

Numbers & Displays

Đây là bài thí nghiệm thiết kế mạch tổ hợp để thực hiện bộ biến đổi số nhị phân sang số thập phân và mạch cộng hai số BCD.

1.Thí nghiệm2.1:

Dùng các đèn 7 đoạn HEX1 và HEX0 để hiển thị các số thập phân từ 0 đến 9. Giá trị hiển thị thay đổi được bằng các công tắc $SW7-4$ và $SW3-0$ tương ứng.

- Cácbướccầnthựchiện:
  1. Tạo project mới.
  2. Viết chương trình Verilog cho bài TN
  3. Gán chân & biên dịch project.
  4. Nạp project vào kit TN. Thử mạch bằng cách thay đổi các công tắc và quan sát các đèn hiển thị.
- Chương trình của nhóm:

```
module TN2_1(SW,LEDR,HEX0,HEX1);

input [7:0]SW;

output [6:0]HEX0,HEX1;

output [7:0]LEDR;

assign LEDR=SW;

ganso (SW[3:0],HEX0);

ganso (SW[7:4],HEX1);

endmodule

module ganso(S,M);

input [3:0]S;

output [6:0]M;

reg [6:0]M;

always@(S[3:0])

case(S[3:0])

4'b0000: M=7'b1000000;

4'b0001: M=7'b1001111;

4'b0010: M=7'b0100100;

4'b0011: M=7'b0110000;
```

```
4'b0100: M=7'b0011001;
4'b0101: M=7'b0010010;
4'b0110: M=7'b0000010;
4'b0111: M=7'b1111000;
4'b1000: M=7'b0000000;
4'b1001: M=7'b0010000;
default: M=7'b1111111;
endcase
endmodule
```

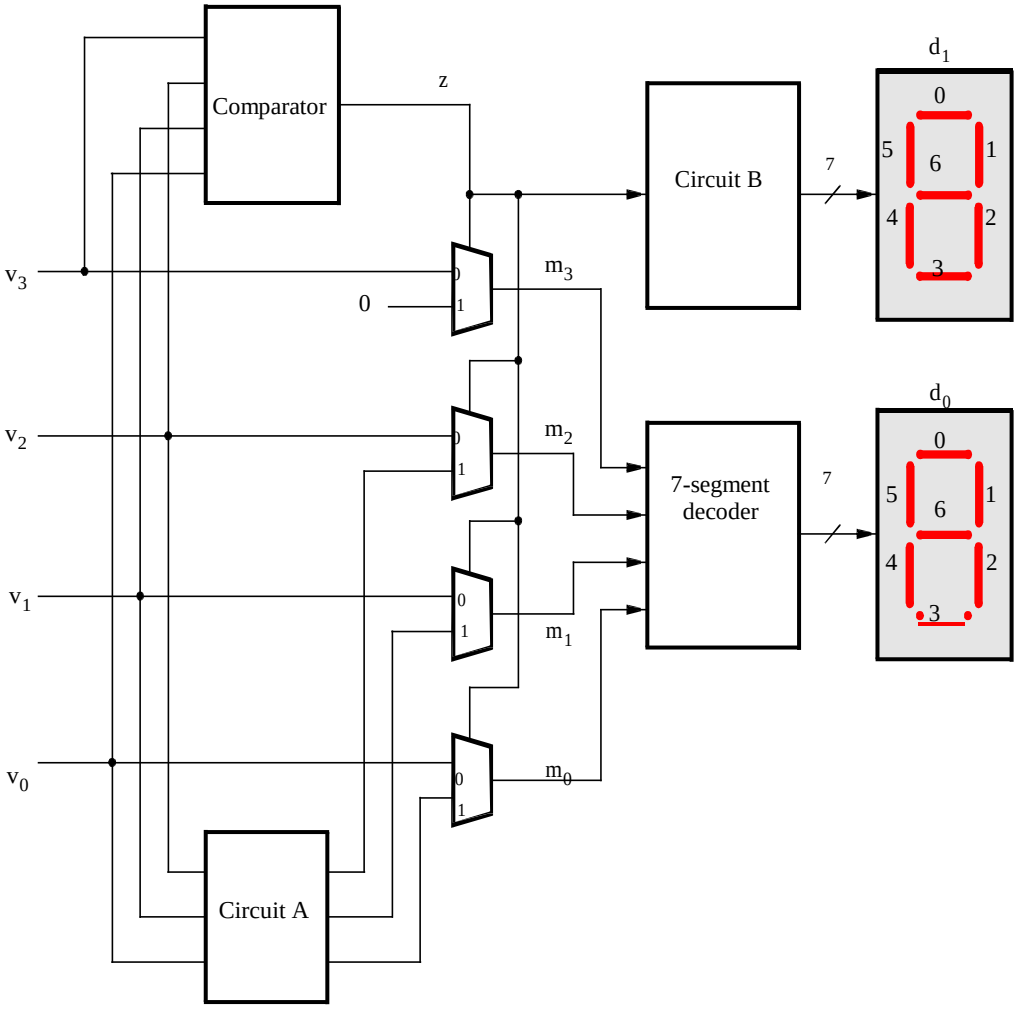
2.Thí nghiệm2.2:

Thực hiện 1 phần của mạch chuyển đổi số nhị phân 4 bit $V = v_3 v_2 v_1 v_0$ thành số thập phân $D = d_1 d_0$ như hình 1, bảng 1. Mạch bao gồm mạch so sánh (để kiểm tra V > 9), mạch multiplexer và mạch A (chưa cần thực hiện mạch B và bộ giải mã 7 đoạn). Mạch sẽ có ngõ vào V 4 bit, ngõ ra M 4 bit và ngõ ra z.

Binary value	Decimal digits	
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

Bảng 1. Bảng giá trị chuyển đổi nhị phân thập phân.

- Cácbướccầnthựchiện:
  1. Tạo project mới. Viết chương trình
  2. Biên dịch project và thực hiện mô phỏng
  3. Viết thêm đoạn chương trình cho mạch B và mạch giải mã 7 đoạn. Dùng các công tắc SW_{3-0} để nhập số nhị phân V và các đèn 7 đoạn HEX1, HEX0 để hiển thị số thập phân $d_1 d_0$
  4. Biên dịch lại rồi nạp project vào kit TN.
  5. Thử mạch: thay đổi giá trị V và quan sát các đèn hiển thị.



Hình 1. Mạch chuyển đổi nhị phân-thập phân.

- Chương trình của nhóm:

```
module TN2_2(SW,LEDR,HEX0,HEX1);
input [3:0]SW;
output [3:0]LEDR;
output [6:0]HEX0,HEX1;
assign LEDR=SW;
wire [3:0]M,N;
ss4bvoi9(SW,M,N);
ganso(M,HEX0);
ganso(N,HEX1);
endmodule
module ss4bvoi9(B,M,N);
```

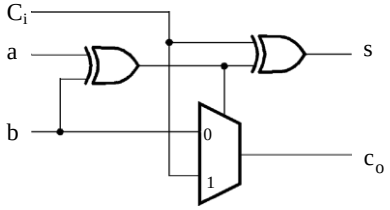
```
input [3:0]B;
output [3:0]M,N;
reg [3:0]M;
reg [3:0]N;
always@(B[3:0] or M or N)
if (B[3:0]<=4'b1001)
begin M=B;
N=4'b0000; end
else
begin M=B+4'b0110;
N=4'b0001; end
endmodule
module ganso(S,M);
input [3:0]S;
output [6:0]M;
reg [6:0]M;
always@(S[3:0])
case(S[3:0])
4'b0000: M=7'b1000000;
4'b0001: M=7'b1001111;
4'b0010: M=7'b0100100;
4'b0011: M=7'b0110000;
4'b0100: M=7'b0011001;
4'b0101: M=7'b0010010;
4'b0110: M=7'b0000010;
4'b0111: M=7'b1111000;
4'b1000: M=7'b0000000;
4'b1001: M=7'b0010000;
default: M=7'b1111111;
endcase
endmodule
```

3.Thínghiệm2.3:

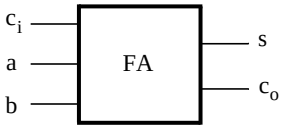
Cho mạch cộng toàn phần (FA) như hình 2a với các ngõ vào a, b, and c_i, các ngõ ra s và c_o.

$$c_o\, s = a + b + c_i.$$

Dùng 4 mạch cộng FA như trên để thực hiện mạch cộng 4 bit như hình 2d.



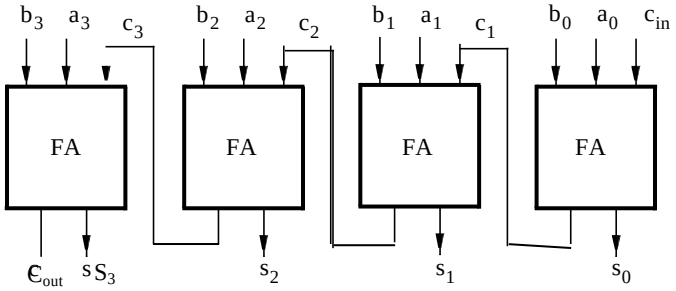
a) Mạch cộng FA



b) Ký hiệu

b	a	c_i	c_o	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

c) Bảng sự thật



d) Mạch cộng 4 bit

Hình 2. Mạch cộng.

- Cácbướccầnthựchiện:
  1. Tạo project mới và viết chương trình Verilog cho mạch cộng:
     - Nối các ngõ vào A, B và c_{in} với các công tắc tương ứng $SW7{-}4$, $SW3{-}0$ và $SW8$ và với các đèn LED màu đỏ LEDR
     - Nối các ngõ ra c_{out} và S với các đèn LED màu xanh LEDG
  2. Gán chân, biên dịch và nạp project vào kit TN
  3. Thử mạch bằng cách thay đổi các giá trị khác nhau của A, B và c_{in}, quan sát các đèn hiển thị.

- Chương trình của nhóm:

```
module TN2_3(SW,LEDR,LEDG);
input [8:0]SW;
output [8:0]LEDR;
output [4:0]LEDG;
assign LEDR=SW;
wire [3:1]c;
wire [2:0]s;
FA(SW[4],SW[0],SW[8],c1,LEDG[0]);
FA(SW[5],SW[1],c1,c2,LEDG[1]);
FA(SW[6],SW[2],c2,c3,LEDG[2]);
FA(SW[7],SW[3],c3,LEDG[4],LEDG[3]);
endmodule
module FA(a,b,ci,co,s);
input a,b;
input ci;
output co;
output s;
assign s = a^b^ci;
Mux21(a^b,b,ci,co);
endmodule
module Mux21(S,X,Y,M);
input S,X,Y;
output M;
assign M=(~S&X)|(S&Y);
endmodule
```

4.Thínghiệm2.4:

Thực hiện mạch cộng 2 số BCD. Ngõ vào của mạch là 2 số A, B và ngõ vào cho số nhớ c_{in}. Ngõ ra là số BCD, tổng $S_1 S_0$ và số nhớ c_{out}.

- Cácbướccầnthựchiện:
  1. Tạo project mới cho mạch cộng số BCD. Phải thực hiện mạch cộng 2 số 4 bit A, B (thí nghiệm 2.3)
     và 1 mạch chuyển đổi 5 bit tổng $s_3 s_2 s_1 s_0 c_o$ thành 2 số BCD $S_1 S_0$ (thí nghiệm 2.2)
  2. Viết chương trình Verilog:
     - Nối các ngõ vào A, B và c_{in} với các công tắc tương ứng SW_{7-4}, SW_{3-0} và SW_8 và với các đèn LED màu đỏ $LEDR_{7-0}$
     - Nối các ngõ ra c_{out} và S với các đèn LED màu xanh $LEDG_{4-0}$
     - Dùng các đèn 7 đoạn HEX3, HEX2 để hiển thị giá trị của 2 số A và B và HEX1, HEX0 để hiển thị kết quả $S_1 S_0$.
  3. Gán chân, biên dịch và nạp project vào kit TN
  4. Thử mạch bằng cách thay đổi các giá trị khác nhau của A, B và c_{in}, quan sát các đèn hiển thị.

- Chương trình của nhóm:

```
module TN2_4(SW,LEDR,LEDG,HEX0,HEX1,HEX2,HEX3);

input [8:0]SW;

output [4:0]LEDG;

output [9:0]LEDR;

output [6:0]HEX0,HEX1,HEX2,HEX3;

wire [4:0]a;

assign LEDR=SW;

assign a[4:0] = SW[7:4] + SW[3:0] + SW[8];

assign LEDG = a;

reg [3:0]c;

reg [3:0]b;

always @ (a or c or b)

if (a <5'b01010)

begin c = a;

b = 4'b0000; end

else if (a<5'b10100)

begin c = a + 4'b0110;
```

```
b = 4'b0001; end
else if (a<5'b11110)
begin c = a + 4'b1100;
b = 4'b0010; end
else begin c=4'b0000;
b= 4'b0011;end
ganso(SW[7:4],HEX3);
ganso(SW[3:0],HEX2);
ganso(b,HEX1);
ganso(c,HEX0);
endmodule
module ganso(S,M);
input [3:0]S;
output [6:0]M;
reg [6:0]M;
always@(S[3:0])
case(S[3:0])
4'b0000: M=7'b1000000;
4'b0001: M=7'b1001111;
4'b0010: M=7'b0100100;
4'b0011: M=7'b0110000;
4'b0100: M=7'b0011001;
4'b0101: M=7'b0010010;
4'b0110: M=7'b0000010;
4'b0111: M=7'b1111000;
4'b1000: M=7'b0000000;
4'b1001: M=7'b0010000;
default: M=7'b1111111;
endcase
endmodule
```

5.Thínghiệm2.5:

Thiết kế mạch tổ hợp chuyển đổi 1 số nhị phân 6 bit thành số thập phân dưới dạng 2 số BCD. Dùng các công tắc
$SW5-0$ để nhập số nhị phân và các đèn 7 đoạn HEX1 và HEX0 để hiển thị số thập phân.

Chương trình của nhóm:

```
module TN2_5(SW,LEDR,HEX0,HEX1);
input [5:0]SW;
output [9:0]LEDR;
output [6:0]HEX0,HEX1;
wire [5:0]a;
assign LEDR=SW;
assign a[5:0] = SW[5:0];
reg [5:0]c;
reg [5:0]b;
always @ (a or c or b)
if (a <6'b001010)
begin c = a;
b = 6'b000000; end
else if (a<6'b010100)
begin c = a + 6'b00110;
b = 6'b000001; end
else if (a<6'b011110)
begin c = a + 6'b001100;
b = 6'b000010; end
else if (a<6'b101000)
begin c = a + 6'b010010;
b = 6'b000011; end
else if (a<6'b110010)
begin c = a + 6'b011000;
```

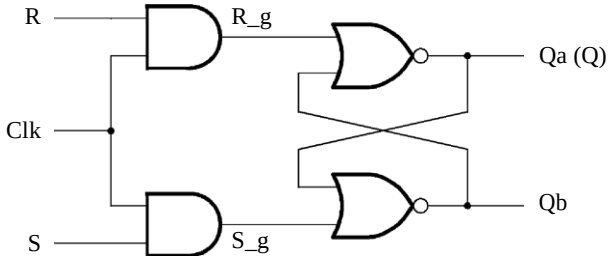
```
b = 6'b000100; end
else if (a<6'b111100)
begin c = a + 6'b011110;
b = 6'b000101; end
else begin c=a + 6'b100100;
b= 6'b000110;end
ganso(b,HEX1);
ganso(c,HEX0);
endmodule
module ganso(S,M);
input [3:0]S;
output [6:0]M;
reg [6:0]M;
always@(S[3:0])
case(S[3:0])
4'b0000: M=7'b1000000;
4'b0001: M=7'b1001111;
4'b0010: M=7'b0100100;
4'b0011: M=7'b0110000;
4'b0100: M=7'b0011001;
4'b0101: M=7'b0010010;
4'b0110: M=7'b0000010;
4'b0111: M=7'b1111000;
4'b1000: M=7'b0000000;
4'b1001: M=7'b0010000;
default: M=7'b1111111;
endcase
endmodule
```

Bài thí nghiệm 3

Latches, Flip-flops, Registers

1.Thínghiệm3.1:

Hình 1 mô tả mạch RS latch dùng cổng logic.
Có 2 cách dùng Verilog để mô tả mạch này: dùng cổng logic (hình 2a) và dùng công thức logic (hình 2b).



Hình 1. Mạch RS latch dùng cổng logic.

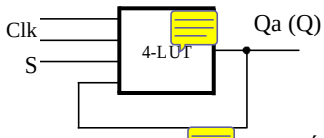
```
// A gated RS latch
module  part1 (Clk, R, S, Q);
    input  Clk, R, S;
    output  Q;
    wire R_g, S_g, Qa, Qb /* synthesis keep */ ;
    and  (R_g,  R,  Clk);
    and   (S_g,  S,  Clk);
    nor  (Qa,  R_g,  Qb);
    nor (Qb, S_g, Qa);
    assign  Q = Qa;
endmodule
```

Hình 2a. Dùng cổng logic để mô tả mạch RS latch.

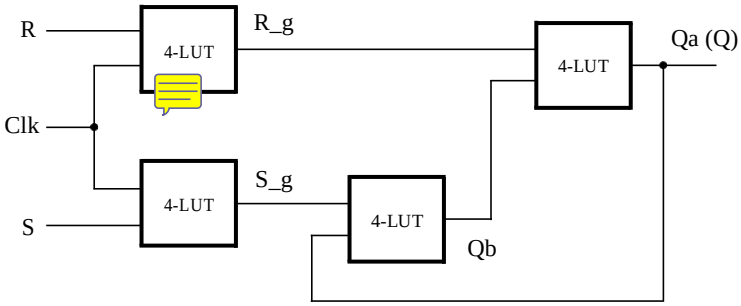
```
// A gated RS latch
module  part1 (Clk, R, S, Q);
    input  Clk, R, S;
    output  Q;
    wire R_g, S_g, Qa, Qb /* synthesis keep */ ;
    assign  R_g = R & Clk;
    assign  S_g = S & Clk;
    assign  Qa =    (R_g   Qb);
    assign  Qb =    (S_g   Qa);
    assign  Q = Qa;
endmodule
```

Hình 2b. Dùng công thức logic để mô tả mạch RS latch.

Có 2 cách thực hiện: dùng 1 LUT 4 ngõ vào (hình 3a) và dùng 3 LUT 2 ngõ vào (hình 3b).



(a) RS latch chỉ dùng 1 bảng tham chiếu 4 ngõ vào.



(b) RS latch dùng 4 bảng tham chiếu 2 ngõ vào.

Hình 3. Các cách thực hiện mạch RS latch

- Các bước cần thực hiện:

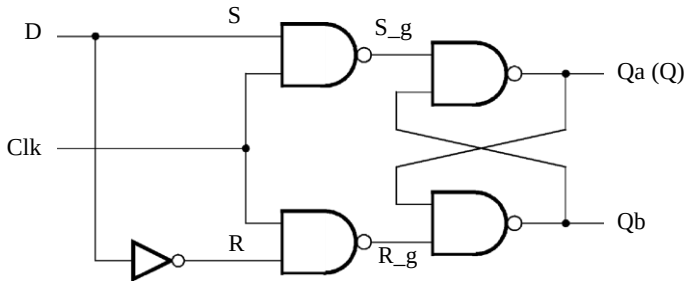
1. Tạo project RS latch
2. Viết chương trình Verilog theo hai cách 2a và 2b.
3. Biên dịch. Dùng tiện ích RTL Viewer để so sánh với sơ đồ mạch hình 1. Dùng tiện ích Technology Viewer để so sánh với sơ đồ mạch hình 3b.
4. Tạo Vector Waveform File (.vwf) cho các ngõ vào/ra. Tạo dạng sóng cho các ngõ vào R và S rồi dùng tiện ích Quartus II Simulator để quan sát các dạng sóng R_g, S_g, Qa và Qb

- Chương trình của nhóm:

```
module TN3_1(Clk,R,S,Q);

input Clk,R,S;

output Q;

wire R_g,S_g,Qa,Qb;

and (R_g,R,Clk);

and (S_g,S,Clk);

nor (Qa,R_g,Qb);

nor (Qb,S_g,Qa);

assign Q=Qa;

endmodule
```

2.Thínghiệm3.2:

Cho mạch D latch dùng cổng như hình 4.



Hình 4. Mạch D latch dùng cổng logic.

- Các bước cần thực hiện:

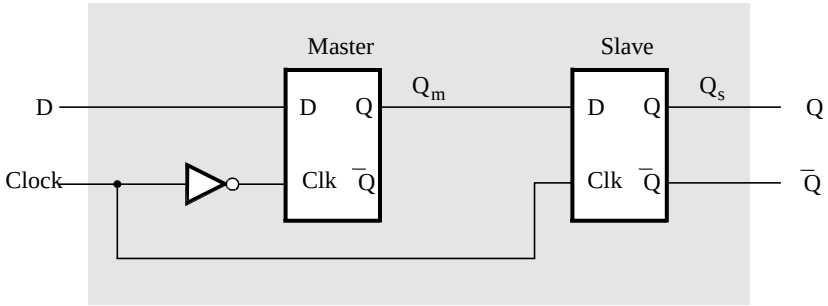
1. Tạo project mới với chương trình Verilog dạng 2b cho mạch D latch.
2. Biên dịch chương trình. Dùng tiện ích Technology Viewer để khảo sát mạch.
3. Mô phỏng để kiểm tra hoạt động của mạch.
4. Dùng công tắc SW_0 cho ngõ vào D, và SW_1 cho ngõ vào Clk. Nối ngõ ra Q đến $LEDR_0$.
5. Biên dịch chương trình lại và nạp project vào kit TN.
6. Thử mạch bằng cách thay đổi các ngõ vào D, Clk và quan sát ngõ ra Q.

- Chương trình của nhóm:

```
module TN3_2(SW,LEDR,LEDG);

input [1:0]SW;

output [1:0]LEDG,LEDR;

assign LEDR=SW;

wire S,R,R_g,S_g,Qa,Qb;

assign S=SW[0];

not (R,SW[0]);

nand (S_g,S,SW[1]);

nand (R_g,R,SW[1]);

nand (Qa,S_g,Qb);

nand (Qb,R_g,Qa);

assign LEDG[0]=Qa,LEDG[1]=Qb;

endmodule
```

3.Thínghiệm3.3:

Cho mạch master-slave D flip-flop hình 5.



Hình 5. Mạch master-slave D flip-flop.

- Các bước cần thực hiện:

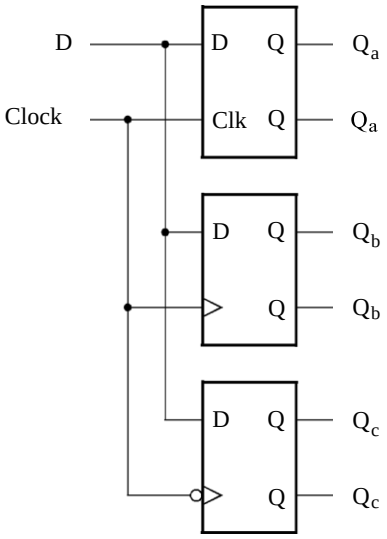
1. Tạo project mới dùng 2 D flip-flop của thí nghiệm 3.2.
2. Dùng công tắc SW_0 cho ngõ vào D, và SW_1 cho ngõ vào Clk. Nối ngõ ra Q đến $LEDR_0$.
3. Biên dịch chương trình.
4. Dùng tiện ích Technology Viewer để khảo sát mạch. Mô phỏng để kiểm tra hoạt động của mạch.
5. Thử mạch bằng cách thay đổi các ngõ vào D, Clk và quan sát ngõ ra Q.

- Chương trình của nhóm:

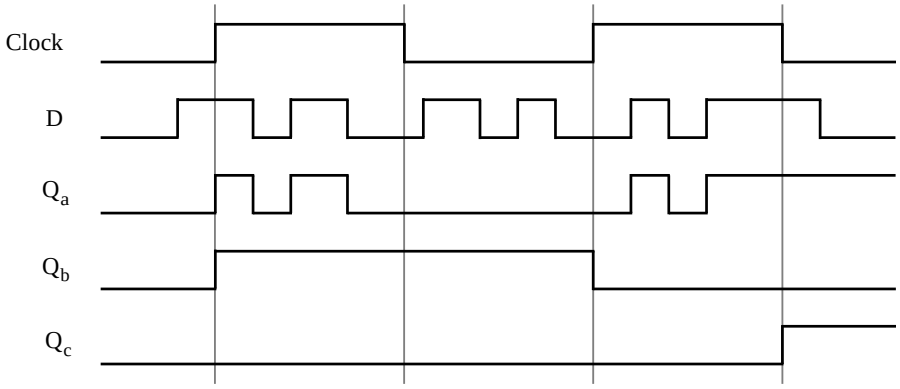
```
module TN3_3(SW,LEDG,LEDR);
input [1:0] SW;
output [1:0] LEDR,LEDG;
assign LEDR=SW;
wire t1,t0;
Dlatch1(SW[0],~SW[1],t1,t0);
Dlatch1(t1,SW[1],LEDG[0],LEDG[1]);
endmodule
module Dlatch1(D,Clk,Q,Q0);
input Clk,D;
output Q,Q0;
wire S,R,R_g,S_g,Qa,Qb;
assign S=D;
not (R,D);
nand (S_g,S,Clk);
nand (R_g,R,Clk);
nand (Qa,S_g,Qb);
nand (Qb,R_g,Qa);
assign Q=Qa,Q0=Qb;
endmodule
```

4.Thínghiệm3.4:

Cho mạch điện hình 6 với D latch, D flip- flop kích cạnh lên và D flip- flop kích cạnh xuống.



(a) Sơ đồ mạch



(b) Giản dồ thời gian

Hình 6. Sơ đồ mạch và dạng sóng của thí nghiệm 3.4.

- Các bước cần thực hiện:

1. Tạo project mới.
2. Viết chương trình dựa trên đoạn chương trình gợi ý như hình 7.
3. Biên dịch chương trình.
4. Dùng tiện ích Technology Viewer để khảo sát mạch.
5. Mô phỏng để kiểm tra hoạt động của mạch. So sánh hoạt động của các phần tử trong mạch.

- Chương trình của nhóm:

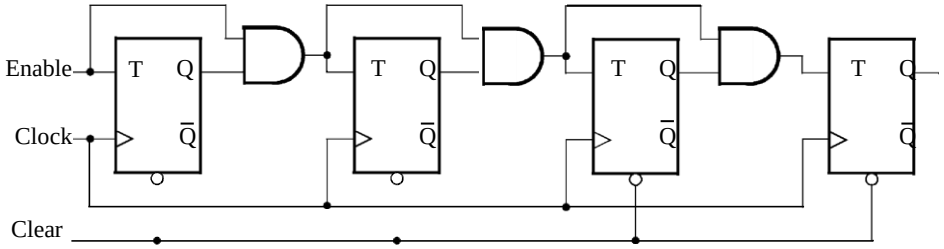
```
module TN3_4(SW,LEDG,LEDR);
input [1:0] SW;
output [2:0] LEDG,LEDR;
assign LEDR=SW;
Dlatch1(SW[0],SW[1],LEDG[0]);
Dflipflop(SW[0],SW[1],LEDG[1]);
Dflipflop(~SW[0],SW[1],LEDG[2]);
endmodule
module Dflipflop(Clk,D,Q);
input Clk,D;
output Q;
wire t1;
Dlatch1(~Clk,D,t1);
Dlatch1(Clk,t1,Q);
endmodule
module Dlatch1(Clk,D,Q);
input Clk,D;
output reg Q;
always @(Clk,D)
if (Clk) Q=D;
endmodule
```

Bài thí nghiệm 4

Counters

1. Thínghiệm4.1:

Cho mạch đếm đồng bộ 4 bit dùng 4 T flip-flops như hình 1.



Hình 1. Bộ đếm 4 bit.

- Các bước cần thực hiện:

1. Tạo project mới thực hiện bộ đếm 16 bit dùng 4 mạch đếm như hình 1. Biên dịch chương trình. Ghi nhận số phần tử logic (LEs) đã được dùng? Tần số hoạt động tối đa (Fmax) của mạch đếm là bao nhiêu?
2. Mô phỏng hoạt động của mạch.
3. Gán thêm nút nhấn KEY_0 làm ngõ vào Clock, các công tắc SW_1, SW_0 làm ngõ vào Enable, Reset và các đèn 7 đoạn $HEX_{3\text{-}0}$ để hiển thị giá trị thập lục phân của ngõ ra mạch đếm.
4. Biên dịch lại và nạp project vào kit TN.
5. Thử hoạt động của mạch bằng cách thay đổi các công tắc và quan sát các đèn 7 đoạn.
6. Thực hiện mạch đếm 4 bit rồi dùng tiện ích RTL Viewer quan sát mạch và so sánh với mạch điện hình 1.

- Chương trình của nhóm:

```
moduleTN4_1(SW,LEDR,LEDG);
input [1:0]SW;
output [3:0]LEDR,LEDG;
assign LEDR=SW;
wire Q0,Q1,Q2,Q3,T1,T2,T3;
Tflipflop(SW[0],SW[1],Q3);
and (T1,Q3,SW[1]);
Tflipflop(SW[0],T1,Q2);
and (T2,Q2,T1);
Tflipflop(SW[0],T2,Q1);
and (T3,Q1,T2);
Tflipflop(SW[0],T3,Q0);
```

```
assign LEDG[3]=Q0,LEDG[2]=Q1,LEDG[1]=Q2,LEDG[0]=Q3;
endmodule
module Tflipflop(Clk,T,Q);
input T,Clk;
output Q;
wire D;
xor (D,T,Q);
Dflipflop(Clk,D,Q);
endmodule
module Dflipflop(Clk,D,Q);
input D,Clk;
output Q;
wire t1;
Dlatch1(~Clk,D,t1);
Dlatch1(Clk,t1,Q);
endmodule
module Dlatch1(Clk,D,Q);
input Clk,D;
output Q;
wire S,R,R_g,S_g,Qa,Qb;
assign S=D;
not (R,D);
nand (S_g,S,Clk);
nand (R_g,R,Clk);
nand (Qa,S_g,Qb);
nand (Qb,R_g,Qa);
assign Q=Qa;
endmodule
```

2.Thínghiệm4.2:

Thực hiện lại thí nghiệm 4.1 dùng mã Verilog sau:

```
Q <=  Q + 1;
```

Biên dịch chương trình.
So sánh số phần tử logic (LEs) đã được dùng, tần số hoạt động tối đa (Fmax) của mạch đếm. Dùng RTL Viewer để khảo sát và nhận xét những khác biệt so với thí nghiệm 4.1.