数字电路与数字系统

实验报告

实　　验: 选择器

姓　名: 周心同

学　　号: 201220069

1.实验目的

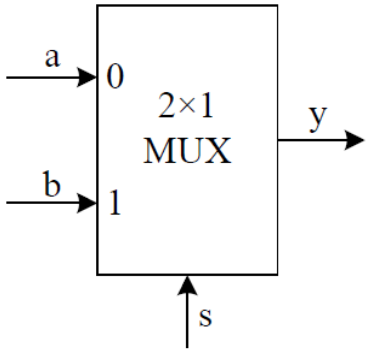
掌握几种常用的多路选择器的设计方法

自行设计一个多路选择器，熟悉电路设计的基本流程和 Quartus 的使用。

Verilog 语言中的 always 语句块、if-else 语句和 case 语句的使用

2.实验原理

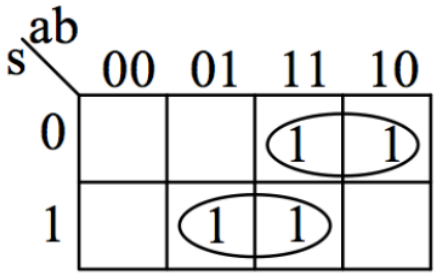
2.1 2 选 1 多路选择器



电路图

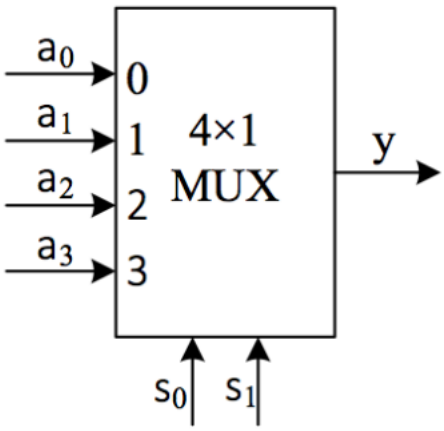
s	a	b
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

真值表



卡诺图

2.2 4 选 1 多路选择器



s_0	s_1	y
0	0	a_0
0	1	a_1
1	0	a_2
1	1	a_3

电路图　　卡诺图

3.实验环境

- 软件环境

Quartus 17.1 Lite

- 硬件环境（预计）

开发板：DE10 Standard

FPGA：Intel Cyclone V SE 5CSXFC6D6F31C6N

4.实验步骤和结果

4.1 代码设计

4.1.1 2 选 1 多路选择器

采用两种实现方式：assign、always。

assign 代码

```
module m_mux21(a,b,s,y);
    input a,b,s;
    output y;

    assign y=(~s&a)|(s&b);

endmodule
```

always 代码

```
module mux21b(a,b,s,y);
    input a,b,s;
    output reg y;

    always @ (*)
    begin
       if(s==0) y=a;
       else y=b;
    end
endmodule
```

4.1.2 4 选 1 多路选择器

assign 代码

```
module mux41(
   input [1:0] X0,
   input [1:0] X1,
   input [1:0] X2,
   input [1:0] X3,
   input [1:0] Y,
   output wire [1:0] F
   );

//add your code here

assign F[0]=(~Y[0]&~Y[1]&X0[0])|(Y[0]&~Y[1]&X1[0])|(~Y[0]&Y[1]&X2[0])|(Y[0]&Y[1]&X3[0]);
assign F[1]=(~Y[0]&~Y[1]&X0[1])|(Y[0]&~Y[1]&X1[1])|(~Y[0]&Y[1]&X2[1])|(Y[0]&Y[1]&X3[1]);

endmodule
```

always 代码

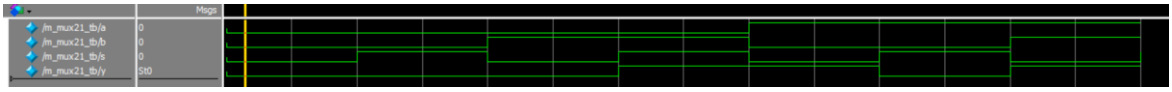
```
module mux41_2(
   input [1:0] X0,
   input [1:0] X1,
   input [1:0] X2,
   input [1:0] X3,
   input [1:0] Y,
   output reg [1:0] F
   );

 //add your code here

 always @ (*)
   begin
   if(Y==2'b00)
   F=X0;
   else if(Y==2'b01)
   F=X1;
   else if(Y==2'b10)
   F=X2;
   else
   F=X3;

   end
```

```
`timescale 1ns/10ps  //时间单位/精度

module mux41_2(
   input [1:0] X0,
   input [1:0] X1,
   input [1:0] X2,
   input [1:0] X3,
   input [1:0] Y,
   output reg [1:0] F
   );

//add your code here

always @ (*)

   case(Y)
   0:F=X0;
   1:F=X1;
   2:F=X2;
   3:F=X3;
   default:F=1'b0;
   endcase



/*   begin
   if(Y==2'b00)
   F=X0;
   else if(Y==2'b01)
   F=X1;
   else if(Y==2'b10)
   F=X2;
   else
   F=X3;

   end */


endmodule
```

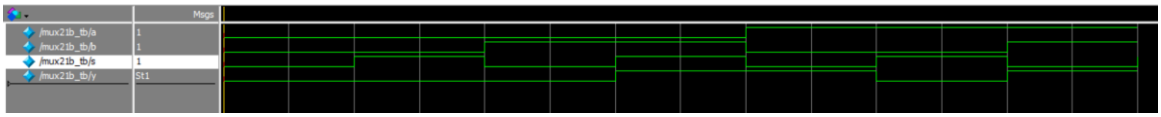
4.2 仿真测试

4.2.1 2 选 1 多路选择器仿真

```
// testbench
module m_mux21_tb;

reg a,b,s;
wire y;

m_mux21 m_mux21(
    .a(a),
    .b(b),
    .s(s),
    .y(y)
);

initial begin

      a<=0;b<=0;s<=0;
#10   a<=0;b<=0;s<=1;
#10   a<=0;b<=1;s<=0;
#10   a<=0;b<=1;s<=1;
#10   a<=1;b<=0;s<=0;
#10   a<=1;b<=0;s<=1;
#10   a<=1;b<=1;s<=0;
#10   a<=1;b<=1;s<=1;

end

endmodule
```



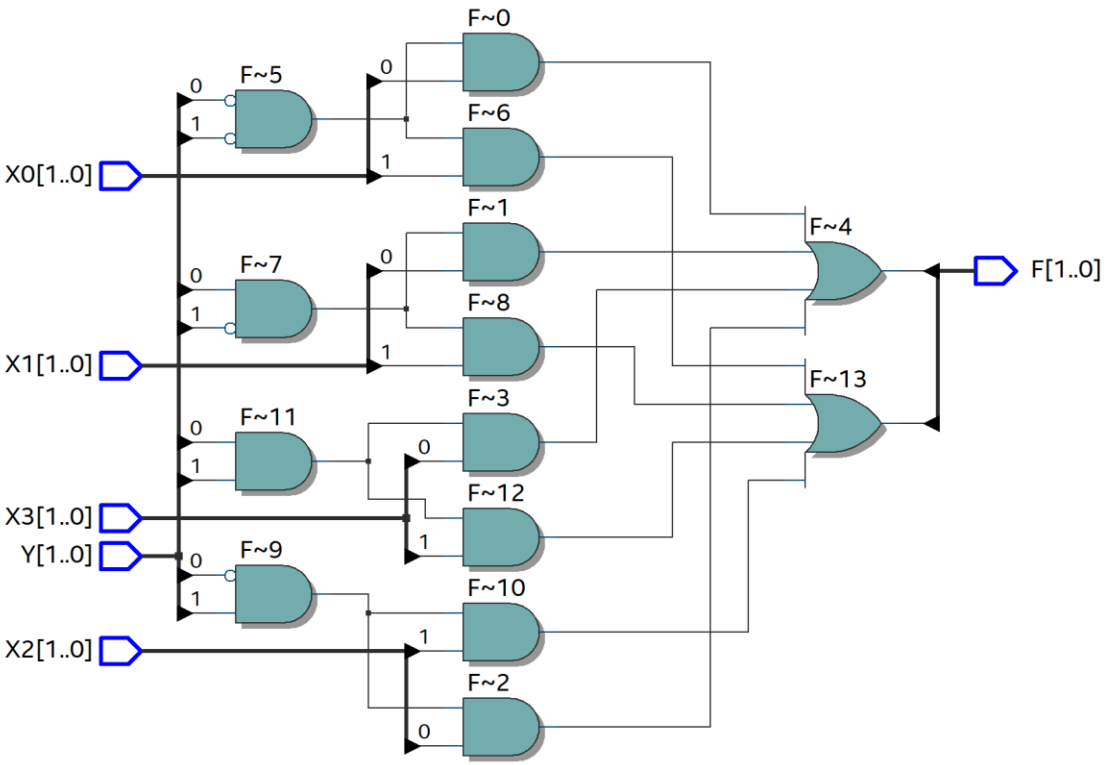
```
// testbench
module mux21b_tb;

reg a,b,s;
wire y;

mux21b mux21b(
    .a(a),
    .b(b),
    .s(s),
    .y(y)
);

initial begin

      a<=0;b<=0;s<=0;
#10   a<=0;b<=0;s<=1;
#10   a<=0;b<=1;s<=0;
#10   a<=0;b<=1;s<=1;
#10   a<=1;b<=0;s<=0;
#10   a<=1;b<=0;s<=1;
#10   a<=1;b<=1;s<=0;
#10   a<=1;b<=1;s<=1;


end

endmodule
```

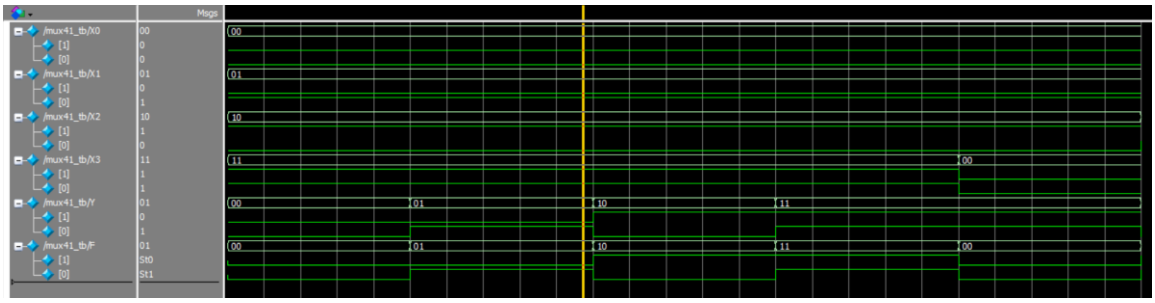


4.2.2 4 选 1 多路选择器仿真

assign

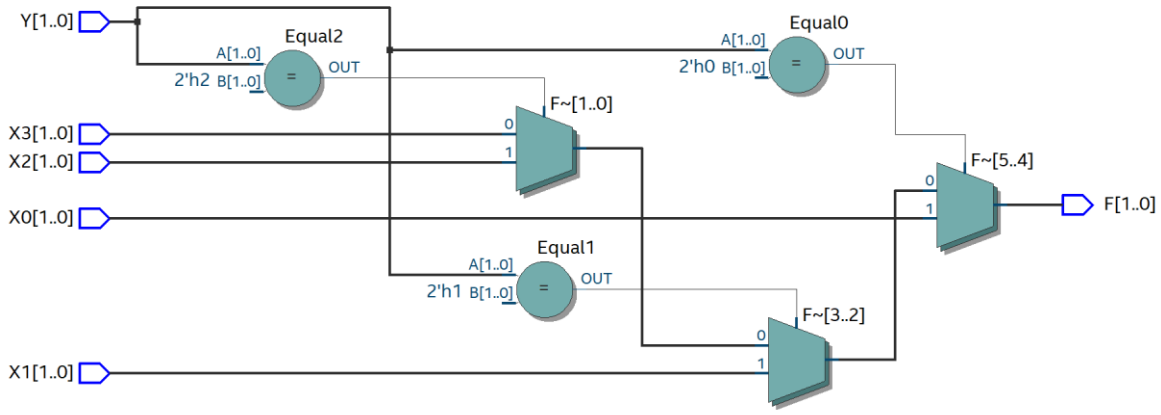
```
// testbench
module mux41_tb;

reg [1:0] X0;
reg [1:0] X1;
reg [1:0] X2;
reg [1:0] X3;
reg [1:0] Y;
wire [1:0] F;

mux41 mux41(
    .X0(X0),
    .X1(X1),
    .X2(X2),
    .X3(X3),
    .Y(Y),
    .F(F)
);

initial begin

    X0=0; X1=1; X2=2; X3=3; Y=0;
#10 X0=0; X1=1; X2=2; X3=3; Y=1;
#10 X0=0; X1=1; X2=2; X3=3; Y=2;
#10 X0=0; X1=1; X2=2; X3=3; Y=3;
#10 X0=0; X1=1; X2=2; X3=0; Y=3;
#10 X0=0; X1=1; X2=3; X3=0; Y=2;

end

endmodule
```

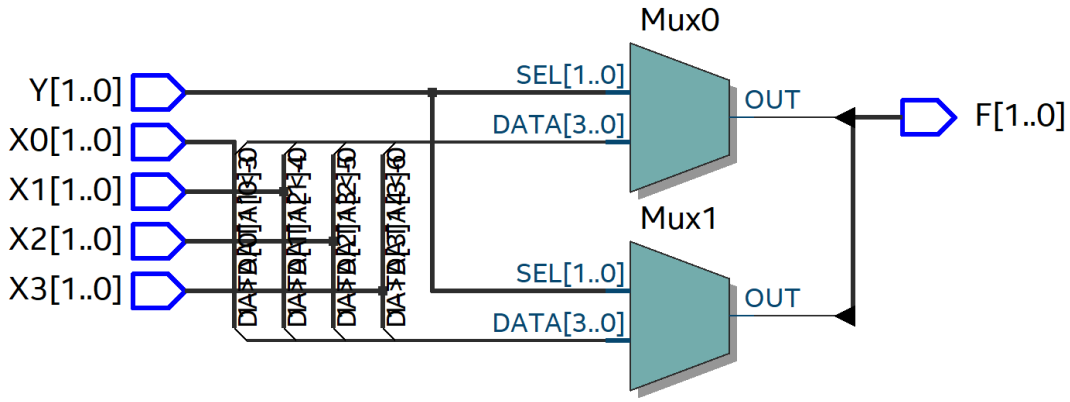


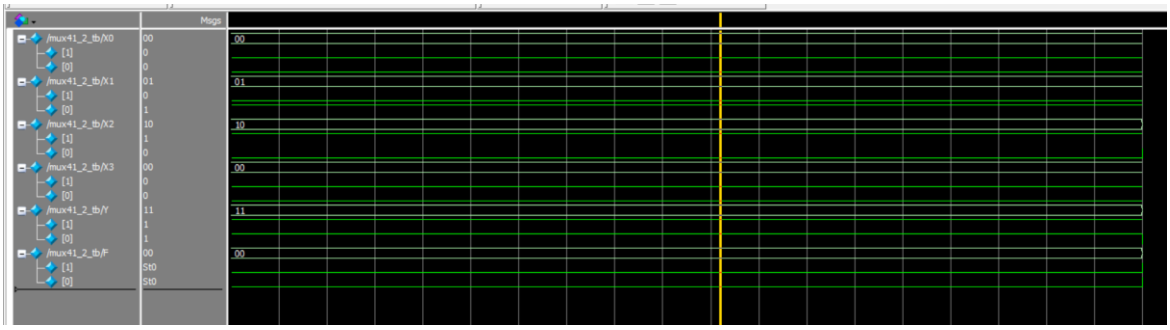


always

```
// testbench
module mux41_2_tb;

reg [1:0] X0;
reg [1:0] X1;
reg [1:0] X2;
reg [1:0] X3;
reg [1:0] Y;
reg [1:0] F;

mux41_2 mux41_2(
    .X0(X0),
    .X1(X1),
    .X2(X2),
    .X3(X3),
    .Y(Y),
    .F(F)
);

initial begin

     X0=0; X1=1; X2=2; X3=3; Y=0;
#10  X0=0; X1=1; X2=2; X3=3; Y=1;
#10  X0=0; X1=1; X2=2; X3=3; Y=2;
#10  X0=0; X1=1; X2=2; X3=3; Y=3;
#10  X0=0; X1=1; X2=2; X3=0; Y=3;
#10  X0=0; X1=1; X2=3; X3=0; Y=2;

end

endmodule
```







5. 实验中遇到的问题及解决办法

问题：在实验 1 中，由于生成了一个 SW 和 LEDR 为接口的 module，仿真时接口不好写。

解决方法：对每个选择器都单独进行了仿真测试

6.实验启示和建议

仿真测试时输出端接 wire