

Week 8

1)

```
// swapmux
```

```
module l8a(x,y,xout,yout);
```

```
    input x,y;
```

```
    output xout, yout;
```

```
    wire m;
```

```
    mux2to1 mux1(y,{~x,x},m);
```

```
    mux2to1 mux2(m,{~y,y},yout);
```

```
    mux2to1 mux3(yout,{~m,m},xout);
```

```
endmodule
```

```
module mux2to1(s,w,f);
```

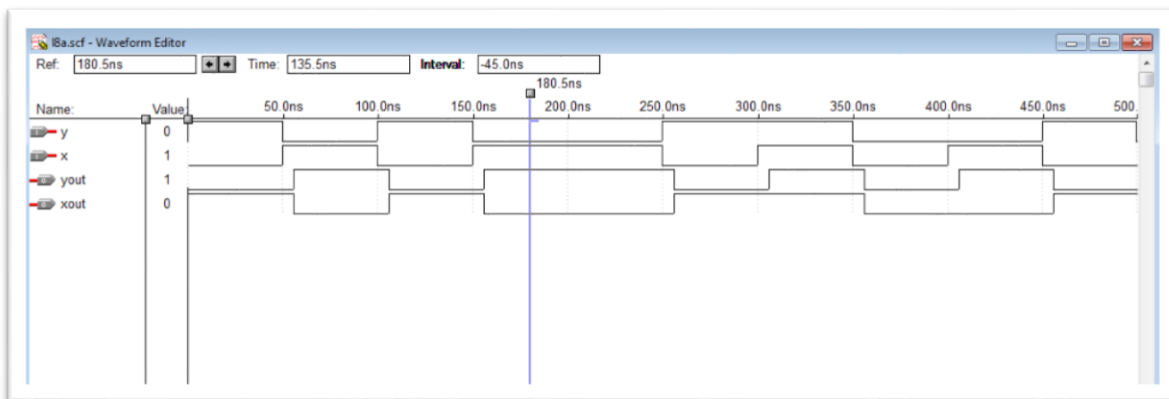
```
    input s;
```

```
    input[0:1]w;
```

```
    output f;
```

```
    assign f=s?w[1]:w[0];
```

```
endmodule
```



2)

// processor

module l8b(Data, Reset, w, Clock, F, Rx, Ry, R0, R1, R2, R3, Count, I, BusWires);

input [3:0] Data;

input Reset, w, Clock;

input [1:0] F, Rx, Ry;

output [1:0] Count, I;

output [3:0] BusWires, R0, R1, R2, R3;

reg [3:0] BusWires;

reg [3:0] Sum;

reg [0:3] Rin, Rout;

reg Extern, Ain, Gin, Gout, AddSub;

wire [0:3] Xreg, Y;

wire [3:0] A, G;

wire [1:6] Func, FuncReg, Sel;

upcount1 counter (Reset, Clock, Count);

assign Func = {F, Rx, Ry};

wire FRin = w & ~Count[1] & ~Count[0];

regn3 functionreg (Func, FRin, Clock, FuncReg);

assign I = FuncReg[1:2];

dec2to4 decX (FuncReg[3:4], 1'b1, Xreg);

dec2to4 decY (FuncReg[5:6], 1'b1, Y);

always @(Count or I or Xreg or Y)

begin

Extern = 1'b0; Ain = 1'b0; Gin = 1'b0;

Gout = 1'b0; AddSub = 1'b0; Rin = 4'b0; Rout = 4'b0;

case (Count)

2'b00: ;

2'b01:

case (I)

```

2'b00:
    begin
        Extern = 1'b1; Rin = Xreg;
    end
2'b01:
    begin
        Rout = Y; Rin = Xreg;
    end
default:
    begin
        Rout = Xreg; Ain = 1'b1;
    end
endcase

2'b10:
case(I)
    2'b10: begin
        Rout = Y; Gin = 1'b1;
    end
    2'b11: begin
        Rout = Y; AddSub = 1'b1; Gin = 1'b1;
    end
    default: ;
endcase

2'b11:
case (I)
    2'b10, 2'b11:
        begin
            Gout = 1'b1; Rin = Xreg;
        end
    default: ;
endcase
endcase
end

```

```
regn2 reg_0 (BusWires, Rin[0], Clock, R0);  
regn2 reg_1 (BusWires, Rin[1], Clock, R1);  
regn2 reg_2 (BusWires, Rin[2], Clock, R2);  
regn2 reg_3 (BusWires, Rin[3], Clock, R3);  
regn2 reg_A (BusWires, Ain, Clock, A);
```

```
always @(AddSub or A or BusWires)
```

```
begin
```

```
    if (!AddSub)
```

```
        Sum = A + BusWires;
```

```
    else
```

```
        Sum = A - BusWires;
```

```
end
```

```
regn2 reg_G (Sum, Gin, Clock, G);
```

```
assign Sel = {Rout, Gout, Extern};
```

```
always @(Sel or R0 or R1 or R2 or R3 or G or Data)
```

```
begin
```

```
    if (Sel == 6'b100000)
```

```
        BusWires = R0;
```

```
    else if (Sel == 6'b010000)
```

```
        BusWires = R1;
```

```
    else if (Sel == 6'b001000)
```

```
        BusWires = R2;
```

```
    else if (Sel == 6'b000100)
```

```
        BusWires = R3;
```

```
    else if (Sel == 6'b000010)
```

```
        BusWires = G;
```

```
    else BusWires = Data;
```

```
end
```

```
endmodule
```

```
module upcount1(Clear, Clock, Q);
```

```
    input Clear, Clock;
```

```
    output [1:0] Q;
```

```
    reg [1:0] Q;
```

```
    always @(posedge Clock)
```

```
    if (Clear)
```

```
        Q <= 0;
```

```
    else
```

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

```
endmodule
```

```
module regn2(R, L, Clock, Q);
```

```
    parameter n = 4;
```

```
    input [n-1:0] R;
```

```
    input L, Clock;
```

```
    output[n-1:0] Q;
```

```
    reg [n-1:0] Q;
```

```
    always @(posedge Clock)
```

```
    if (L)
```

```
        Q <= R;
```

```
endmodule
```

```
module dec2to4 (W, En, Y);
```

```
    input [1:0] W;
```

```
    input En;
```

```
    output [0:3] Y;
```

```
    reg [0:3] Y;
```

```
    always @(W or En)
```

```
    case ({En,W})
```

```
    3'b100: Y = 4'b1000;  
    3'b101: Y = 4'b0100;  
    3'b110: Y = 4'b0010;  
    3'b111: Y = 4'b0001;  
    default: Y = 4'b0000;  
  
endcase  
endmodule
```

```
module regn3(R, L, Clock, Q);  
    parameter n = 6;  
    input [n-1:0] R;  
    input L, Clock;  
    output[n-1:0] Q;  
    reg [n-1:0] Q;  
  
    always @(posedge Clock)  
        if (L)  
            Q <= R;  
endmodule
```

