Gabe DiMartino

CPEN 230

Homework 7

4.19 3to8 Binary Decoder

```
module 3to8-decoder(w1,w2,w3,f);
    input w1,w2,w3;
    output reg f;

    always @* begin
        case({w3,w2,w1})
        3'b001: f = 1;
        3'b010: f = 1;
        3'b011: f = 1;
        3'b101: f = 1;
        3'b110: f = 1;
        default: f = 0;
        endcase
    end

endmodule
```

4.21 3to8 Binary Encoder

```
module 8to3_encoder(
    input [7:0] sw,
    output [2:0] f
);

always @(*) begin
    case(sw)
        8'b00000001: f = 3'b000;
        8'b00000010: f = 3'b001;
        8'b00000100: f = 3'b010;
        8'b00001000: f = 3'b011;
        8'b00010000: f = 3'b100;
        8'b00100000: f = 3'b101;
        8'b01000000: f = 3'b110;
        8'b10000000: f = 3'b111;
        default: f = 3'b000; //The En bit is not activated, or no switches
are activated
    endcase
end

endmodule
```

4.22 Fix the modified 2to4 decoder.

```
module 2to4_decoder (W, En, Y);
    input [1:0] W;
    input En;
    output reg [3:0] Y; //Flip 0:3
    integer k;

  always @(W, En)
    begin // Add begin
        for (k = 0; k <= 3; k = k + 1) begin // Added Begin
      if (W == k)
        Y[k] = En;
      else
        Y[k] = 0; //Fixed to set all values not equal to K as 0. Also made
sure all info was inside the always block
    end // Add End
    end // Add End
endmodule
```

4.26 if2to4 and h3to8

```
module if2to4 (
    input [1:0] sw,
    output [3:0] f
);
    always @(*) begin
        if (sw == 2'b00)
            f = 4'b0001;
        else if (sw == 2'b01)
            f = 4'b0010;
        else if (sw == 2'b10)
            f = 4'b0100;
        else
            f = 4'b1000;
    end
endmodule

module h3to8(
    input [2:0] in,
    output [7:0] out
);

    wire [3:0] in1, in2;

    if2to4 u1(.sw(in[1:0]), .f(in1));
    if2to4 u2(.sw(in[2:1]), .f(in2));

    assign out = {in2, in1};

endmodule
```

4.27 h6to64

```
module h3to8(
    input [2:0] in,
    output [7:0] out
);

    wire [3:0] in1, in2;

    if2to4 u1(.sw(in[1:0]), .f(in1));
    if2to4 u2(.sw(in[2:1]), .f(in2));

    assign out = {in2, in1};

endmodule

module h6to64 (
    input [5:0] in,
    output [63:0] out
);

    wire [7:0] sub1, sub2, sub3, sub4, sub5, sub6, sub7, sub8, sub9;

    h3to8 u1 (.in(in[2:0]), .out(sub1));
    h3to8 u2 (.in(in[5:3]), .out(sub2));
    h3to8 u3 (.in(in[2:0]), .out(sub3));
    h3to8 u4 (.in(in[5:3]), .out(sub4));
    h3to8 u5 (.in(in[2:0]), .out(sub5));
    h3to8 u6 (.in(in[5:3]), .out(sub6));
    h3to8 u7 (.in(in[2:0]), .out(sub7));
    h3to8 u8 (.in(in[5:3]), .out(sub8));
    h3to8 u8 (.in(in[2:0]), .out(sub9));

    assign out = {sub9, sub8, sub7, sub6, sub5, sub4, sub3, sub2, sub1}

endmodule
```

4.28 dec2to4 and 4to1 mux

```
module dec2to4 (W, En, Y);
    input [1:0] W;
    input En;
    output reg [0:3] Y;
    always @(W, 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 4to1mux (S, W, F);
    input [1:0] S;
    input [3:0] W;
    output reg F;

    wire [0:3] F_intermediate;

    dec2to4 decoder1 (.W(S), .En(1'b1), .Y(F_intermediate));

    assign F_intermediate[0] = W[0] & F_intermediate[0];
    assign F_intermediate[1] = W[1] & F_intermediate[1];
    assign F_intermediate[2] = W[2] & F_intermediate[2];
    assign F_intermediate[3] = W[3] & F_intermediate[3];

    assign F = F_intermediate[0] | F_intermediate[1] | F_intermediate[2] |
F_intermediate[3];

endmodule
```