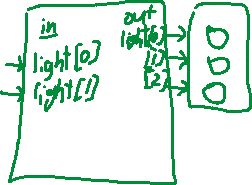
Q1



Module control (input [2:0] ival,

Output [1:0] light);

Assign green = ival[0] ? 2’b10 : 2’b11

Assign red = ival[2] ? 2’b00 : 2’b11

Assign yellow = ival[1] ? 2’b01 : 2’b11

endmodule

module convert\_light(input[1:0] lights\_in,

output[2:0] lights\_out);

assign lights\_out[2] = ~lights\_in[1]& ~lights\_in[0]; //red

assign lights\_out[1] = ~lights\_in[1]& lights\_in[0]; //yellow

assign lights\_out[0] = lights\_in[1] & ~lights\_in[0]; //green

endmodule

Q2a)



Module thermo #(parameter width = 8)(

input[width-1:0] Tset, Tact,

Output reg Hon,Con);

Output = Hi-Z

always@\*

begin

if (Tact-Tset >4)

output = Con;

else if (Tact-Tset<-4)

output = Hon;

end

endmodule

module thermos #(parameter SIZE=8)(

input[SIZE-1:0] Tact, Tset,

output Hon, Con);

assign Hon = Tact <= (Tste -4); //heater on

assign Con = Tact>= (Tset +4); //cooler on

b)

thermos U1(.Tset(TSET),

.Tact(TACT),

.Hon(Hon),

.Con(Con));

c)Change parameter width to 12

thermos #(.SIZE(12)) U1 (.Tact(WTact),.Tset(WTset), .Hon(Whon), .Con(WCon));

Q4a)

Module decoder3\_8(output reg[7:0] d\_out,

Input [2:0] ival);

always@\*

case(ival)

3’000 : d\_out = 8’b00000001;

3’001 : d\_out = 8’b00000010;

3’010 : d\_out = 8’b00000100;

3’011 : d\_out = 8’b00001000;

3’100 : d\_out = 8’b00010000;

3’101 : d\_out = 8’b00100000;

3’110 : d\_out = 8’b01000000;

3’111 : d\_out = 8’b10000000;

default: d\_out = 8’b00000000;

Endcase

end

Endmodule

b)

Add “ input sel; ”

Use if else: “ always@\*

begin

d\_out = 8’b00000000;

if(sel)begin

case(ival)

…

endcase

end

end

endmodule

output reg only when always \* is used