CE/CZ1005 Digital Logic
Tutorial 7

Q1. A traffic light for a toy car track has a 3-bit one-hot vector input, with MSB corresponding to red and the LSB to green. However, the controller delivered with the track has only a 2-bit output for the lights, encoded as: red = “00”, yellow = “01” and green = “10”. The controller outputs “11” when there are no lights active.

Design a Verilog interface (using assign statements) to go between the controller and the lights.

Q2. A digital thermostat has two 8-bit unsigned binary inputs representing the target temperature and the actual temperature in degrees centigrade (°C). The thermostat has two outputs: one to turn a heater on when the actual temperature is 4°C below the target, and one to turn a cooler on when the actual temperature is 4°C above the target

(a) Design a Verilog module, *thermo*, with two 8-bit inputs, Tset and Tact and two 1-bit outputs Hon and Con. Use a parameter statement to specify the 8-bit width.

(b) The module designed in part (a) is instantiated in another module. Write the Verilog statement to instantiate the thermo module with identifier U1 using the same signal names in the upper module.

(c) Part way through the design process, the design team finds out that the temperature sensor and the target set-point both have 12-bit outputs. Describe a simple change to the Verilog statement to instantiate the thermo module so that this will not affect the operation.

Q3. The following Verilog module has a number of errors that would result in it synthesizing incorrectly.

(a) Identify the errors and rewrite the module to correct them.

(b) How could you rewrite that module using a single conditional assignment?

```
module thingamajig (input [3:0] a, b, c,
                    input [1:0] sel,
                    output [5:0] result);

      always @ (a,b)
      begin
         case (sel)
            2’b00 : result = a;
            2’b01 : result = b;
            2’b10 : result = c;
         endcase
      end
endmodule;
```

Q4. (a) Write a Verilog module that implements a 3-to-8 decoder using a case statement in an always block.

(b) What modification would you make in order to add an enable (en) to the circuit?

Q1.

```
module convert_light (input [1:0] lights_in,
                      output [2:0] lights_out);

   assign lights_out[2] = ~light_in[1] & ~light_in[0];   //red
   assign lights_out[1] = ~light_in[1] & light_in[0];    //yel
   assign lights_out[0] = light_in[1] & ~light_in[0];    //grn

endmodule
```

Why don't we need to worry about the no light case (eg. 11)?

Q2. (a)

```
module thermo #(parameter SIZE = 8) (
                input [SIZE-1:0] Tact, Tset,
                output Hon, Con);

   assign Hon = (Tact + 4) < Tset;   //Heater On
   assign Con = Tact > (Tset + 4);   //Cooler On

endmodule
```

What problem could potentially occur if we used:

```
   assign Hon = Tact < (Tset - 4);
```

(b)

```
thermo U1 (.Tact(Tact), .Tset(Tset), .Hon(Hon),
           .Con(Con));
```

(c)

```
thermo #(.SIZE(12)) U1 (.Tact(Tact), .Tset(Tset),
         .Hon(Hon), .Con(Con));
```

Q3. (a) There are 5 mistakes/4 errors

```
module thingamajig (input [3:0] a, b, c,
                    input [1:0] sel,
                    output reg [3:0] result);

   always @ (a,b,c,sel)     //or better use “always @ *”
   begin
       case (sel)
           2’b00 : result = a;
           2’b01 : result = b;
           2’b10 : result = c;
           //default or 2’b11 case or default assignment
           // (above case). Else it will synthesize storage.
       endcase
   end

endmodule; //no semicolon (Remove)
```

(b)

```
module thingamajig (input [3:0] a, b, c,
                    input [1:0] sel,
                    output [3:0] result);

   assign result = (sel==2’b00) ? a :
                   (sel==2’b01) ? b : c;

endmodule
```

Here, c is also assigned to result when sel is `10` and `11`.
If you specifically wanted `0` to be assigned for the 11 case, then you would need to modify the assign statement to:

```
   assign result = (sel==2’b00) ? a :
                   (sel==2’b01) ? b :
                   (sel==2’b10) ? c : 4’b0000;
```

Q4. (a)

```
module dec3to8 (input [2:0] i, output reg [7:0] o);
   always @ * begin
       case (i)
          3'b000 : o = 8'b0000_0001;
          3'b001 : o = 8'b0000_0010;
          3'b010 : o = 8'b0000_0100;
          3'b011 : o = 8'b0000_1000;
          3'b100 : o = 8'b0001_0000;
          3'b101 : o = 8'b0010_0000;
          3'b110 : o = 8'b0100_0000;
          3'b111 : o = 8'b1000_0000;
          default: o = 8'b0000_0000;
       endcase
   end
endmodule
```

(b) Add the enable within the always block.

```
module dec3to8 (input en, input [2:0] i, output reg [7:0] o);
   always @ * begin
      o = 8'b0000_0000;      // default assignment to o
      if (en) begin
          case (i)
             3'b000 : o = 8'b0000_0001;
             3'b001 : o = 8'b0000_0010;
             3'b010 : o = 8'b0000_0100;
             3'b011 : o = 8'b0000_1000;
             3'b100 : o = 8'b0001_0000;
             3'b101 : o = 8'b0010_0000;
             3'b110 : o = 8'b0100_0000;
             3'b111 : o = 8'b1000_0000;
          endcase
      end
   end
endmodule
```

Note: The **begin** and **end** keywords with the **if** statement are not needed.

(b) Alternative solution (does not use if statement). But if version is better.

Add enable input to port declaration, and remove **reg** from the output declaration for o:

```
module dec3to8 (input en, input [2:0] i, output [7:0] o);
```

Declare 8-bit variable in the module body. Allocated from inside an always so must be **reg**:

```
reg [7:0] o_int;
```

Assign to o_int from inside the case statement (instead of assigning to o):

```
3'b011 : o_int = 8'b0000_1000;
```

Add a conditional assignment to assign to o, so must remove reg from the output declaration:

```
assign o = en ? o_int : 8'b0000_0000;
```