1)A) Aim : To write a Verilog program for 2-BIT COMPARATOR using GATE LEVEL MODELLING

Code:

```
module    2bit_comp(E, L, G, A, B);
input     [1:0]A ,  [1:0]B;
wire    [11:0]w;
output    E, L, G;
not     N1(w[0], A[1]);
not     N2(w[1], A[0]);
not     N3(w[2], B[1]);
not     N4(w[3], B[0]);
xnor    x1(w[4], A[0], B[0]);
xnor    x1(w[5], A[1], B[1]);
and     A1(w[6], w[0], B[1]);
and     A2(w[7], w[1], w[0], B[0]);
and     A3(w[8], A[1], w[2]);
and     A4(w[9], A[1], w[2]);
and     A5(w[10], A[0], w[2], B[0]);
and     A6(w[11], w[3], A[1], A[0]);
and     A7(E, w[4], w[5]);
or      O1(L, w[6], w[7], w[8]);
or      O2(G, w[9], w[10], w[11]);
endmodule
```

2)A) Aim : To write the Verilog code for an electronic combination lock with two number buttons (0 and 1) and an unlock output of the combination “11011”.

Code :

```
module    lock(clk, x, z);
input    clk, x;
output   reg  z;
reg   [2:0]state;
initial  state = 3’b000;
always@(posedge  clk)
begin
   case(state)
      3’b000 : begin
                  If(x == 0)
                   begin
                        state <= 3’b000;
                        z <= 1’b0;
                  end
                  else
                  begin
                        state <= 3’b001;
                        z <= 1’b0;
                  end
            end
   3’b001 : begin
                  If(x == 0)
                   begin
                        state <= 3’b000;
                        z <= 1’b0;
```

```
                end
                else
                begin
                        state <= 3'b010;
                        z <= 1'b0;
                end
        end
3'b010 : begin
                If(x == 0)
                 begin
                        state <= 3'b011;
                        z <= 1'b0;
                end
                else
                begin
                        state <= 3'b000;
                        z <= 1'b0;
                end
        end
3'b011 : begin
                If(x == 0)
                 begin
                        state <= 3'b000;
                        z <= 1'b0;
                end
                else
                begin
                        state <= 3'b100;
```

```
                    z <= 1'b0;
                end
            end
    3'b100 : begin
                If(x == 0)
                 begin
                    state <= 3'b000;
                    z <= 1'b0;
                end
                else
                begin
                    state <= 3'b000;
                    z <= 1'b1;
                end
            end
    endcase
end
endmodule
```

3)A) Aim : To write Verilog code for Digital Clock using one Always block.

Code :

```
 module    digital_clock (clk, rst, sec, min, hrs);
input    clk, rst;
output    reg [5:0]sec, [5:0]min, [4:0]hrs;
always@(posedge  clk  or  posedge  rst)
begin
        if(rst == 1'b1)
                begin
                        sec <= 0;
                        min <= 0;
                        hrs <= 0;
                end
        else
                if(clk == 1)
                        begin
                                sec = sec+1;
                                if(sec == 59)
                                        begin
                                                sec = 0;
                                                min = min+1;
                                                if(min == 59)
                                                        begin
                                                                min = 0;
                                                                hrs = hrs+1;
                                                                if(hrs == 23)
                                                                        begin
                                                                                hrs = 0;
```

```
                                                    end
                                                end
                                            end
                                        end
    end
    endmodule
```

4)A)a) Aim: To write a Verilog code for representing 16*1 mux using 2*1 mux

Code:

```
module   2to1Mux(y, s, i1,i0);

input   s, i1, i0;

output   y;

assign   y = (s? (i1: i0));

endmodule

module   16to1Mux(y, i, s);

input   [15:0]i, [3:0]s;

output  y;

wire  [14:1]w;

2to1Mux    M1(w[1], s[0], i1, i0);

2to1Mux    M2(w[2], s[0], i3, i2);

2to1Mux    M3(w[3], s[0], i5, i4);

2to1Mux    M4(w[4], s[0], i7, i6);

2to1Mux    M5(w[5], s[0], i9, i8);

2to1Mux    M6(w[6], s[0], i10, i11);

2to1Mux    M7(w[7], s[0], i13, i12);

2to1Mux    M8(w[8], s[0], i15, i14);

2to1Mux    M9(w[9], s[1], w[2], w[1]);

2to1Mux    M10(w[10], s[1], w[4], w[3]);

2to1Mux    M11w[11], s[1], w[6], w[5]);

2to1Mux    M12(w[12], s[1], w[8], w[7);

2to1Mux    M13(w[13], s[2], w[10], w[9]);

2to1Mux    M14(w[14], s[2], w[12], w[11]);
```

```
2to1Mux    M15(y, s[3], w[14], w[13]);
endmodule
```

4)A)b)Aim: To write a Verilog code for representing 16*1 mux using 4*1 mux

Code:

```
module   16to1Mux(y, I, s);
input   [15:0]i, [3:0]s;
output   y;
wire   [4:1]w;
4to1Mux   M1(w[0], [1:0]s, [3:0]i);
4to1Mux   M1(w[1], [1:0]s, [7:4]i);
4to1Mux   M1(w[2], [1:0]s, [1:8]i);
4to1Mux   M1(w[3], [1:0]s, [15:12]i);
4to1Mux   M1(y, [1:0]s, [3:0]i);
endmodule
module    4to1Mux(y, s, i);
input  [1:0]s, [3:0]i;
output    y;
assign   y=s[1] ? (s[0] ? (i[3] : i[2])) : (s[0] ? (i[1] : i[0]));
endmodule
```

5)A) Aim: To design Asynchronous 4-bit up/down counter with JK Flip-Flops using Verilog HDL

Code:

```
module    j_kff(clk, j, k, reset, q, qbar);
input   j, k, clk, reset;
output    reg q, qbar;
always @(posedge clk)
begin
        if(reset==1)
              begin
                    q<=0;
                     qbar<=1;
              end
         else
             begin
                 if(j==0 & k==0)
                    begin
                            q<=q
                             qbar<=qbar;
                    end
          elseif(j==0 & k==1)
                    begin
                            q<=0;
                               qbar<=1;
                    end
           elseif(j==1 & k==0)
```

```
                begin
                        q<=1;
                          qbar<=0;
                end
        else
                begin
                          q<=~q;
                          qbar<=~qbar;
                end
end
endmodule
module      2to1Mux(y, s, i1, i0)
input     s, i1, i0 ;
assign   y=s? (i1 : i0);
endmodule

module   counter(clk, j, k, s, rst,  q3, q2, q1, q0);
input    clk, j, k, s, rst;
output    q3, q2, q1, q0;
wire   q3bar, q2bar, q1bar, q0bar,w1, w2, w3;
2to1Mux   M1(w1,s,q0,q0bar)
2to1Mux   M2(w2,s,q1,q1bar)
2to1Mux   M3(w3,s,q2,q2bar)
j_kff   F1(clk, j, k, rst, q0, q0bar)
j_kff   F2(w1, j, k, rst, q1, q1bar)
```

```
j_kff    F3(w2, j, k, rst, q2, q2bar)

j_kff    F4(w3, j, k, rst, q3, q3bar)

endmodule
```

6)A) Aim : To design the given system based on the given FSM using Verilog HDL.

Code :

```
module     route (clk, x, z);
input    clk, x;
output   reg  z;
reg [2:0] state;
initial  state = 3'b000;
always @ ( posedge  clk )
begin
          case( state )
                    3'b000 : begin
                              If ( x == 0)
                                        state <= 3'b000;
                              else
                                        state <= 3'b001;
                              end
                    3'b001 : begin
                              If ( x == 0)
                                        state <= 3'b010;
                              else
                                        state <= 3'b100;
                              end
                    3'b010 : begin
                              If ( x == 0)
                                        state <= 3'b011;
                              else
                                        state <= 3'b100;
                              end
```

```
3’b011 : begin
        If ( x == 0)
                state <= 3’b011;
        else
                state <= 3’b000;
        end
3’b100 : begin
        If ( x == 0)
                state <= 3’b111;
        else
                state <= 3’b101;
        end
3’b101 : begin
        If ( x == 0)
                state <= 3’b011;
        else
                state <= 3’b110;
        end
3’b110 : begin
        If ( x == 0)
                state <= 3’b111;
        else
                state <= 3’b110;
        end
3’b111 : begin
        If ( x == 0)
                state <= 3’b001;
        else
```

```
                    state <= 3'b101;
                end
        endcase
end
endmodule
```