BEHAVIORAL LEVEL MODELLING

1.

```
module full_adder(output reg s,c_out,input a,b,c_in);
 always @ (a or b or c_in) begin
     {c_out,s} = a + b + c_in;
    end
endmodule
module a1(output [3:0] Sum,output Cout,input [3:0] A,B,input Cin);
 wire c1,c2,c3;
 full_adder FA1(Sum[0],c1,A[0],B[0],Cin),
    FA2(Sum[1],c2,A[1],B[1],c1),
    FA3(Sum[2],c3,A[2],B[2],c2),
    FA4(Sum[3],Cout,A[3],B[3],c3);
endmodule
module a1_tb;
 reg [3:0] a,b;
 reg c_in;
 wire c_out;
 wire [3:0] sum;
 integer i;
 a1 Instance (sum,c_out,a,b,c_in);
 initial begin
  a <= 0;
  b <= 0;
  c_in <= 0;
  $monitor ("a=0x%0h b=0x%0h c_in=0x%0h c_out=0x%0h sum=0x%0h", a, b, c_in, c_out, sum);
   for (i = 0; i < 4; i = i+1) begin
   #10 a <= $random;
   b <= $random;
   c_in <= $random;
   end
 end
endmodule
```

2.

```
module full_adder(output reg s,c_out,input a,b,c_in);
 always @ (a or b or c_in) begin
     {c_out,s} = a + b + c_in;
    end
endmodule
module a2(output [7:0] Sum,output Cout,input [7:0] A,B,input Cin);
 wire c1,c2,c3,c4,c5,c6,c7;
 full_adder FA1(Sum[0],c1,A[0],B[0],Cin),
    FA2(Sum[1],c2,A[1],B[1],c1),
    FA3(Sum[2],c3,A[2],B[2],c2),
  FA4(Sum[3],c4,A[3],B[3],c3),
    FA5(Sum[4],c5,A[4],B[4],c4),
  FA6(Sum[5],c6,A[5],B[5],c5),
    FA7(Sum[6],c7,A[6],B[6],c6),
    FA8(Sum[7],Cout,A[7],B[7],c7);
endmodule
module a2_tb;
 reg [7:0] a,b;
 reg c_in;
 wire c_out;
 wire [7:0] sum;
```

```
 integer i;
 a2 Instance (sum,c_out,a,b,c_in);
 initial begin
  a <= 0;
  b <= 0;
  c_in <= 0;
  $monitor ("a=0x%0h b=0x%0h c_in=0x%0h c_out=0x%0h sum=0x%0h", a, b, c_in, c_out, sum);
   for (i = 0; i < 8; i = i+1) begin
   #10 a <= $random;
   b <= $random;
   c_in <= $random;
   end
 end
endmodule
```

3.

```
module a3(output reg [3:0] sum,output reg cout,input [3:0] a,b,input cin);
 reg p0,p1,p2,p3,g0,g1,g2,g3,c0,c1,c2,c3,c4;
always@(a or b)
 begin
 if((a[0]==1'b1 & b[0]==1'b0) | (a[0]==1'b0 & b[0]==1'b1))
 begin
 p0=1'b1;
 end
 else
 p0=1'b0;
 end
always@(a or b)
 begin
 if((a[1]==1'b1 & b[1]==1'b0) | (a[1]==1'b0 & b[1]==1'b1))
 begin
 p1=1'b1;
 end
 else
 p1=1'b0;
 end
always@(a or b)
   begin
 if((a[2]==1'b1 & b[2]==1'b0) | (a[2]==1'b0 & b[2]==1'b1))
 begin
 p2=1'b1;
 end
 else
 p2=1'b0;
 end
always@(a or b)
begin
 if((a[3]==1'b1 & b[3]==1'b0) | (a[3]==1'b0 & b[3]==1'b1))
 begin
 p3=1'b1;
 end
 else
 p3=1'b0;
 end
always@(a or b)
 begin
```

```
 if(a[0]==1'b1 & b[0]==1'b1)
 begin
 g0=1'b1;
 end
 else
 g0=1'b0;
 end
always@(a or b)
 begin
 if(a[1]==1'b1 & b[1]==1'b1)
 begin
 g1=1'b1;
 end
 else
 g1=1'b0;
 end
always@(a or b)
 begin
 if(a[2]==1'b1 & b[2]==1'b1)
 begin
 g2=1'b1;
 end
 else
 g2=1'b0;
 end
always@(a or b)
 begin
 if(a[3]==1'b1 & b[3]==1'b1)
 begin
 g3=1'b1;
 end
 else
 g3=1'b0;
 end
always@(cin)
 begin
 if(cin==1'b0)
 begin
 c0=1'b0;
 end
 else
 c0=1'b1;
 end
 reg f1,f2,f3,f4,f5,f6,f7,f8,f9,f10;
always@(p0 or cin)
 begin
 if(p0==1'b1 & cin==1'b1)
 begin
 f1=1'b1;
 end
 else
 f1=1'b0;
 end
always@(g0 or f1)
 begin
```

```
 if(g0==1'b0 & f1==1'b0)
 begin
 c1=1'b0;
 end
 else
 c1=1'b1;
 end
always@(p1 or g0)
 begin
 if(p1==1'b1 & g0==1'b1)
 begin
 f2=1'b1;
 end
 else
 f2=1'b0;
 end
always@(p1 or p0 or cin)
 begin
 if(p1==1'b1 & p0==1'b1 & cin==1'b1)
 begin
 f3=1'b1;
 end
 else
 f3=1'b0;
 end
always@(g1 or f2 or f3)
 begin
 if(g1==1'b0 & f2==1'b0 & f3==1'b0)
 begin
 c2=1'b0;
 end
 else
 c2=1'b1;
 end
always@(p2 or g1)
begin
if(p2==1'b1 & g1==1'b1)
begin
f4=1'b1;
end
else
f4=1'b0;
end
always@(p2 or p1 or g0)
 begin
 if(p2==1'b1 & p1==1'b1 & g0==1'b1)
 begin
 f5=1'b1;
 end
 else
 f5=1'b0;
 end
always@(p2 or p1 or p0 or cin)
 begin
 if(p2==1'b1 & p1==1'b1 & p0==1'b1 & cin==1'b1)
```

```
 begin
 f6=1'b1;
 end
 else
 f6=1'b0;
 end
always@(g2 or f4 or f5 or f6)
 begin
 if(g2==1'b0 & f4==1'b0 & f5==1'b0 & f6==1'b0)
 begin
 c3=1'b0;
 end
 else
 c3=1'b1;
 end
always@(p3 or g2)
 begin
 if(p3==1'b1 & g2==1'b1)
 begin
 f7=1'b1;
 end
 else
 f7=1'b0;
 end
always@(p3 or p2 or g1)
 begin
 if(p3==1'b1 & p2==1'b1 & g1==1'b1)
 begin
 f8=1'b1;
 end
 else
 f8=1'b0;
 end
always@(p3 or p2 or p1 or g0)
 begin
 if(p3==1'b1 & p2==1'b1 & p1==1'b1 & g0==1'b1)
 begin
 f9=1'b1;
 end
 else
 f9=1'b0;
 end
always@(p3 or p2 or p1 or p0 or cin)
 begin
 if(p3==1'b1 & p2==1'b1 & p1==1'b1 & p0==1'b1 & cin==1'b1)
 begin
 f10=1'b1;
 end
 else
 f10=1'b0;
 end
always@(g3 or f7 or f8 or f9 or f10)
 begin
 if(g3==1'b0 & f7==1'b0 & f8==1'b0 & f9==1'b0 & f10==1'b0)
 begin
```

```
 c4=1'b0;
 end
 else
 c4=1'b1;
 end
always@(p0 or c0)
begin
 if((p0==1'b1 & c0==1'b0) | (p0==1'b0 & c0==1'b1))
 begin
 sum[0]=1'b1;
 end
 else
 sum[0]=1'b0;
 end
always@(p1 or c1)
 begin
 if((p1==1'b1 & c1==1'b0) | (p1==1'b0 & c1==1'b1))
 begin
 sum[1]=1'b1;
 end
 else
 sum[1]=1'b0;
end
always@(p2 or c2)
 begin
 if((p2==1'b1 & c2==1'b0) | (p2==1'b0 & c2==1'b1))
 begin
 sum[2]=1'b1;
 end
 else
 sum[2]=1'b0;
 end
always@(p3 or c3)
 begin
 if((p3==1'b1 & c3==1'b0) | (p3==1'b0 & c3==1'b1))
 begin
 sum[3]=1'b1;
 end
 else
 sum[3]=1'b0;
 end
always@(c4)
begin
if(c4==1'b0)
begin
cout=1'b0;
end
else
cout=1'b1;
end
endmodule
module a3_tb;
 reg [3:0] a,b;
 reg c_in;
 wire c_out;
```

```
 wire [3:0] sum;
 integer i;
 a3 instance0(sum,c_out,a,b,c_in);
 initial begin
  a <= 0;
  b <= 0;
  c_in <= 0;
   for (i = 0; i < 4; i = i+1) begin
   #10 a <= $random;
   b <= $random;
   c_in <= $random;
   end
 end
endmodule
```

4.

```
module a4(input a,b,c,d,output reg w,x,y,z);
reg f1,f2,f3,f4,f5,f6,f7;
always@(c or b)
 begin
 if(c==1'b1 & b==1'b1)
 begin
 f1=1'b1;
 end
 else
 f1=1'b0;
 end
always@(d or b)
 begin
 if(d==1'b1 & b==1'b1)
 begin
 f2=1'b1;
 end
 else
 f2=1'b0;
 end
always@(f1 or f2 or a)
 begin
 if(f1==1'b0 & f2==1'b0 & a==1'b0)
 begin
 w=1'b0;
 end
 else
 w=1'b1;
 end
always@(c or b)
 begin
 if(c==1'b1 & b==1'b0)
 begin
 f3=1'b1;
 end
 else
 f3=1'b0;
 end
always@(d or b)
 begin
```

```
 if(d==1'b1 & b==1'b0)
 begin
 f4=1'b1;
 end
 else
 f4=1'b0;
 end
always@(c or b or d)
 begin
 if(c==1'b0 & d==1'b0 & b==1'b1)
 begin
 f5=1'b1;
 end
 else
 f5=1'b0;
 end
always@(f3 or f4 or f5)
 begin
 if(f3==1'b0 & f4==1'b0 & f5==1'b0)
 begin
 x=1'b0;
 end
 else
 x=1'b1;
 end
always@(c or d)
 begin
 if(c==1'b1 & d==1'b1)
 begin
 f6=1'b1;
 end
 else
 f6=1'b0;
 end
always@(c or d)
 begin
 if(c==1'b0 & d==1'b0)
 begin
 f7=1'b1;
 end
 else
 f7=1'b0;
 end
always@(f6 or f7)
 begin
 if(f6==1'b0 & f7==1'b0)
 begin
 y=1'b0;
 end
 else
 y=1'b1;
 end
always@(d)
 begin
 if(d==1'b0)
```

```
 begin
 z=1'b1;
 end
 else
 z=1'b0;
 end
endmodule
module a4_tb;
 reg a,b,c,d;
 wire w,x,y,z;
 integer i;
 a4 instance0(a,b,c,d,w,x,y,z);
 initial begin
  for(i=0;i<16;i=i+1)
   begin
   {a,b,c,d}=i;
   #10;
   end
 end
endmodule
```

5.

4 BIT BINARY ADDER

```
module full_adder(output reg s,c_out,input a,b,c_in);
  always @ (a or b or c_in) begin
     {c_out,s} = a + b + c_in;
    end
endmodule
module a5(output [4:0] Sum,input [3:0] A,B);
 wire c1,c2,c3;
 full_adder FA1(Sum[0],c1,A[0],B[0],1'b0),
    FA2(Sum[1],c2,A[1],B[1],c1),
    FA3(Sum[2],c3,A[2],B[2],c2),
    FA4(Sum[3],Sum[4],A[3],B[3],c3);
endmodule
module a5_tb;
 reg [3:0] a,b;
 wire [4:0] sum;
 integer i;
 a5 Instance (sum,a,b);
 initial begin
  a <= 0;
  b <= 0;
  $monitor ("a=0x%0h b=0x%0h sum=0x%0h", a, b,sum);
   for (i = 0; i < 4; i = i+1) begin
   #10 a <= $random;
   b <= $random;
   end
 end
endmodule
```

4 BIT BINARY SUBTRACTOR

```
module full_adder(output reg s,c_out,input a,b,c_in);
 always @ (a or b or c_in) begin
     {c_out,s} = a + b + c_in;
    end
endmodule
```

```
module a5(output [3:0] d,output c_out,input [3:0] A,B);
 wire c1,c2,c3;
 full_adder FA1(d[0],c1,A[0],~B[0],1'b1),
    FA2(d[1],c2,A[1],~B[1],c1),
    FA3(d[2],c3,A[2],~B[2],c2),
    FA4(d[3],c_out,A[3],~B[3],c3);
endmodule
module a5_tb;
 reg [3:0] a,b;
 wire c_out;
 wire [3:0] d;
 integer i;
 a5 Instance (d,c_out,a,b);
 initial begin
  a <= 0;
  b <= 0;
  $monitor ("a=0x%0h b=0x%0h c_out=0x%0h sum=0x%0h", a, b,c_out,d);
   for (i = 0; i < 10; i = i+1) begin
   #10 a <= $random;
   b <= $random;
   end
 end
endmodule
```

6.

```
module a6(a,b,carry_in,sum,carry);
    input [2:0] a,b;
    input carry_in;
    output [3:0] sum;
    output carry;
    reg [4:0] sum_temp;
    reg [3:0] sum;
    reg carry;
    always @(a,b,carry_in)
    begin
        sum_temp = a+b+carry_in;
        if(sum_temp > 9)    begin
            sum_temp = sum_temp+6;
            carry = 1;
            sum = sum_temp[3:0];    end
        else    begin
            carry = 0;
            sum = sum_temp[3:0];
        end
    end
endmodule
module a6_tb;
    reg [2:0] a;
    reg [2:0] b;
    reg carry_in;
    wire [3:0] sum;
    wire carry;
  a6 instance0(a,b,carry_in,sum,carry);
    initial begin
        a = 0;  b = 0;  carry_in = 0;   #100;
        a = 6;  b = 7;  carry_in = 0;   #100;
```

```
        a = 3;  b = 3;  carry_in = 1;   #100;
        a = 4;  b = 5;  carry_in = 0;   #100;
        a = 7;  b = 2;  carry_in = 0;   #100;
        a = 5;  b = 6;  carry_in = 1;   #100;
    end
endmodule
```

7.

```
module a7( a, b, c, d, s0, s1, out);
input wire a, b, c, d;
input wire s0, s1;
output reg out;
always @ (a or b or c or d or s0, s1)
begin
case (s0 | s1)
2'b00 : out <= a;
2'b01 : out <= b;
2'b10 : out <= c;
2'b11 : out <= d;
endcase
end
endmodule
module a7_tb;
wire  out;
reg  a,b,c,d,s0,s1;
a7 instance0(a,b,c,d,s0,s1,out);
 initial
 begin
 a=1'b0; b=1'b0; c=1'b0; d=1'b0;
 s0=1'b0; s1=1'b0;
 #500 $finish;
end
always #40 a=~a;
always #20 b=~b;
always #10 c=~c;
always #5 d=~d;
always #80 s0=~s0;
always #160 s1=~s1;
always@(a or b or c or d or s0 or s1)
$monitor("At time = %t, Output = %d", $time, out);
endmodule
```

8.

```
module mux4to1_gate(out,in,sel);
  input [0:3] in;
  input [0:1] sel;
  output reg out;
   always @ (in or sel)
begin
case (sel)
2'b00 : out <= in[0];
2'b01 : out <= in[1];
2'b10 : out <= in[2];
2'b11 : out <= in[3];
endcase
end
 endmodule
```

```
module a8(out,in,sel);
 input [0:15] in;
 input [0:3] sel;
 output out;
 wire [0:3] ma;
 mux4to1_gate mux1(ma[0],in[0:3],sel[2:3]);
 mux4to1_gate mux2(ma[1],in[4:7],sel[2:3]);
 mux4to1_gate mux3(ma[2],in[8:11],sel[2:3]);
 mux4to1_gate mux4(ma[3],in[12:15],sel[2:3]);
 mux4to1_gate mux5(out,ma,sel[0:1]);
endmodule
module a8_tb;
reg [0:15] in;
reg [0:3] sel;
wire out;
a8 instance0(out,in,sel);
initial
begin
$monitor("in=%b | sel=%b | out=%b",in,sel,out);
end
initial
begin

in=16'b1000000000000000; sel=4'b0000;

#30 in=16'b0100000000000000; sel=4'b0001;

#30 in=16'b0010000000000000; sel=4'b0010;

#30 in=16'b0001000000000000; sel=4'b0011;

#30 in=16'b0000100000000000; sel=4'b0100;

#30 in=16'b0000010000000000; sel=4'b0101;

#30 in=16'b0000001000000000; sel=4'b0110;

#30 in=16'b0000000100000000; sel=4'b0111;

#30 in=16'b0000000010000000; sel=4'b1000;

#30 in=16'b0000000001000000; sel=4'b1001;

#30 in=16'b0000000000100000; sel=4'b1010;

#30 in=16'b0000000000010000; sel=4'b1011;

#30 in=16'b0000000000001000; sel=4'b1100;

#30 in=16'b0000000000000100; sel=4'b1101;

#30 in=16'b0000000000000010; sel=4'b1110;

#30 in=16'b0000000000000001; sel=4'b1111;
```

```
end
endmodule
9.
module mux4to1_gate(out,in,sel);
  input [0:3] in;
  input [0:1] sel;
  output reg out;
   always @ (in or sel)
begin
case (sel)
2'b00 : out <= in[0];
2'b01 : out <= in[1];
2'b10 : out <= in[2];
2'b11 : out <= in[3];
endcase
end
 endmodule
module a9(out,in,sel);
 input [0:15] in;
 input [0:3] sel;
 output out;
 wire [0:3] ma;
 mux4to1_gate mux1(ma[0],in[0:3],sel[2:3]);
 mux4to1_gate mux2(ma[1],in[4:7],sel[2:3]);
 mux4to1_gate mux3(ma[2],in[8:11],sel[2:3]);
 mux4to1_gate mux4(ma[3],in[12:15],sel[2:3]);
 mux4to1_gate mux5(out,ma,sel[0:1]);
endmodule
module a9_tb;
reg [0:15] in;
reg [0:3] sel;
wire out;
a9 instance0(out,in,sel);
initial
begin
$monitor("in=%b | sel=%b | out=%b",in,sel,out);
end
initial
begin

in=16'b1000000000000000; sel=4'b0000;

#30 in=16'b0100000000000000; sel=4'b0001;

#30 in=16'b0010000000000000; sel=4'b0010;

#30 in=16'b0001000000000000; sel=4'b0011;

#30 in=16'b0000100000000000; sel=4'b0100;

#30 in=16'b0000010000000000; sel=4'b0101;

#30 in=16'b0000001000000000; sel=4'b0110;

#30 in=16'b0000000100000000; sel=4'b0111;
```

```
#30 in=16'b0000000010000000; sel=4'b1000;

#30 in=16'b0000000001000000; sel=4'b1001;

#30 in=16'b0000000000100000; sel=4'b1010;

#30 in=16'b0000000000010000; sel=4'b1011;

#30 in=16'b0000000000001000; sel=4'b1100;

#30 in=16'b0000000000000100; sel=4'b1101;

#30 in=16'b0000000000000010; sel=4'b1110;

#30 in=16'b0000000000000001; sel=4'b1111;

end
endmodule
```

10.

```
module a10(Data_in,Data_out);
   input [2:0] Data_in;
   output [7:0] Data_out;
   reg [7:0] Data_out;
   always @(Data_in)
   case (Data_in)
      3'b000 : Data_out = 8'b00000001;
      3'b001 : Data_out = 8'b00000010;
      3'b010 : Data_out = 8'b00000100;
      3'b011 : Data_out = 8'b00001000;
      3'b100 : Data_out = 8'b00010000;
      3'b101 : Data_out = 8'b00100000;
      3'b110 : Data_out = 8'b01000000;
      3'b111 : Data_out = 8'b10000000;
      default : Data_out = 8'b00000000;
   endcase
endmodule
module a10_tb;
   reg [2:0] Data_in;
   wire [7:0] Data_out;
  a10 instance0(Data_in,Data_out);
   initial begin
      Data_in = 3'b000;    #100;
      Data_in = 3'b001;    #100;
      Data_in = 3'b010;    #100;
      Data_in = 3'b011;    #100;
      Data_in = 3'b100;    #100;
      Data_in = 3'b101;    #100;
      Data_in = 3'b110;    #100;
      Data_in = 3'b111;    #100;
   end
endmodule
```

11.

```
module a11(Data_in_A,Data_in_B,less,equal,greater);
   input [3:0] Data_in_A;
```

```
    input [3:0] Data_in_B;
    output less;
     output equal;
     output greater;
     reg less;
     reg equal;
     reg greater;
    always @(Data_in_A or Data_in_B)
     begin
       if(Data_in_A > Data_in_B)   begin
          less = 0;
          equal = 0;
          greater = 1;    end
       else if(Data_in_A == Data_in_B) begin
          less = 0;
          equal = 1;
          greater = 0;    end
       else    begin
          less = 1;
          equal = 0;
          greater =0;
       end
    end
endmodule
module a11_tb;
    reg [3:0] Data_in_A;
    reg [3:0] Data_in_B;
    wire less;
    wire equal;
    wire greater;
  a11 instance0(Data_in_A,Data_in_B,less,equal,greater);
    initial begin
       Data_in_A = 10;
       Data_in_B = 12;
       #100;
       Data_in_A = 15;
       Data_in_B = 11;
       #100;
       Data_in_A = 10;
       Data_in_B = 10;
       #100;
    end
endmodule
```