GATELEVEL MODELLING

1.

```
module full_adder(output s,c_out,input a,b,c_in);
 wire f1,f2,f3;
      xor(f3,a,b);
 xor(s,f3,c_in);
 and(f1,a,b);
 and(f2,c_in,f3);
 or(c_out,f1,f2);
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 s,c_out,input a,b,c_in);
     wire f1,f2,f3;
      xor(f3,a,b);
 xor(s,f3,c_in);
 and(f1,a,b);
 and(f2,c_in,f3);
 or(c_out,f1,f2);
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(a,b,cin,sum,cout);
 input [3:0] a,b;
 input cin;
 output [3:0] sum;
 output cout;
 wire p0,p1,p2,p3,g0,g1,g2,g3,c0,c1,c2,c3,c4;
 xor(p0,a[0],b[0]);
 xor(p1,a[1],b[1]);
 xor(p2,a[2],b[2]);
 xor(p3,a[3],b[3]);
 and(g0,a[0],b[0]);
 and(g1,a[1],b[1]);
 and(g2,a[2],b[2]);
 and(g3,a[3],b[3]);
 or(c0,cin,cin);
 wire f1,f2,f3,f4,f5,f6,f7,f8,f9,f10;
 and(f1,p0,cin);
 or(c1,g0,f1);
 and(f2,p1,g0);
 and(f3,p1,p0,cin);
 or(c2,g1,f2,f3);
 and(f4,p2,g1);
 and(f5,p2,p1,g0);
 and(f6,p2,p1,p0,cin);
 or(c3,g2,f4,f5,f6);
 and(f7,p3,g2);
 and(f8,p3,p2,g1);
 and(f9,p3,p2,p1,go);
 and(f10,p3,p2,p1,p0,cin);
 or(c4,g3,f7,f8,f9,f10);
 xor(sum[0],p0,c0);
 xor(sum[1],p1,c1);
 xor(sum[2],p2,c2);
 xor(sum[3],p3,c3);
 or(cout,c4,c4);
endmodule
module a3_tb;
 reg [3:0] a,b;
 reg c_in;
 wire c_out;
 wire [3:0] sum;
 integer i;
 a3 instance0(a,b,c_in,sum,c_out);
 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 w,x,y,z);
 wire f1,f2,f3,f4,f5,f6,f7;
 and(f1,b,c);
 and(f2,b,d);
 or(w,a,f1,f2);
 and(f3,~b,c);
 and(f4,~b,d);
 and(f5,b,~c,~d);
 or(x,f3,f4,f5);
 and(f6,c,d);
 and(f7,~c,~d);
 or(y,f6,f7);
 not(z,d);
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 s,c_out,input a,b,c_in);
 wire f1,f2,f3;
      xor(f3,a,b);
 xor(s,f3,c_in);
 and(f1,a,b);
 and(f2,c_in,f3);
 or(c_out,f1,f2);
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 s,c_out,input a,b,c_in);
 wire f1,f2,f3;
      xor(f3,a,b);
 xor(s,f3,c_in);
 and(f1,a,b);
 and(f2,c_in,f3);
 or(c_out,f1,f2);
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;
    wire w1,w2;
    wire [3:0] w;
    assign {w1,w}=a+b+carry_in;
    wire f1,f2;
    and(f1,w[3],w[2]);
    and(f2,w[1],w[3]);
    or(w2,f1,f2);
    or(carry,w1,w2);
    wire [3:0] c;
    assign c={1'b0,carry,carry,1'b0};
    assign sum=c+w;
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 = 7;  carry_in = 1;   #100;
       a = 5;  b = 6;  carry_in = 1;   #100;
    end
endmodule
7.
module a7(out, a, b, c, d, s0, s1);
output out;
input a, b, c, d, s0, s1;
wire sobar, s1bar,f1,f2,f3,f4;
not (s0bar, s0), (s1bar, s1);
and (f1, a, s0bar, s1bar), (f2, b, s0bar, s1),(f3, c, s0, s1bar), (f4, d, s0, s1);
or(out,f1,f2,f3,f4);
endmodule
module a7_tb;
wire  out;
reg  a,b,c,d,s0, s1;
a7 instance0(out,a,b,c,d,s0,s1);
 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 out;
  wire a,b,c,d,n1,n2,a1,a2,a3,a4;
  not n(n1,sel[1]);
  not nn(n2,sel[0]);
  and (a1,in[0],n1,n2);
  and (a2,in[1],n2,sel[1]);
  and (a3,in[2],sel[0],n1);
  and (a4,in[3],sel[0],sel[1]);
  or or1(out,a1,a2,a3,a4);
 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 out;
  wire a,b,c,d,n1,n2,a1,a2,a3,a4;
  not n(n1,sel[1]);
  not nn(n2,sel[0]);
  and (a1,in[0],n1,n2);
  and (a2,in[1],n2,sel[1]);
  and (a3,in[2],sel[0],n1);
  and (a4,in[3],sel[0],sel[1]);
  or or1(out,a1,a2,a3,a4);
 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(a,d);
    input [2:0] a;
    output [7:0] d;
    and(d[0],(~a[2]),(~a[1]),(~a[0]));
    and(d[1],(~a[2]),(~a[1]),(a[0]));
  and(d[2],(~a[2]),(a[1]),(~a[0]));
  and(d[3],(~a[2]),(a[1]),(a[0]));
  and(d[4],(a[2]),(~a[1]),(~a[0]));
  and(d[5],(a[2]),(~a[1]),(a[0]));
  and(d[6],(a[2]),(a[1]),(~a[0]));
  and(d[7],(a[2]),(a[1]),(a[0]));
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(a,b,less,equal,greater);
    input [0:3] a;
    input [0:3] b;
    output less;
    output equal;
    output greater;
   wire f0,f1,f2,f3;
       xnor(f0,a[0],b[0]);
   xnor(f1,a[1],b[1]);
   xnor(f2,a[2],b[2]);
   xnor(f3,a[3],b[3]);
   and(equal,f0,f1,f2,f3);
   wire g0,g1,g2,g3;
   and(g0,a[0],~b[0]);
   and(g1,a[1],~b[1]);
   and(g2,a[2],~b[2]);
   and(g3,a[3],~b[3]);
   wire l0,l1,l2,l3;
   and(l0,~a[0],b[0]);
   and(l1,~a[1],b[1]);
   and(l2,~a[2],b[2]);
   and(l3,~a[3],b[3]);
   wire s1,s2,s3,s4,s5,s6;
   and(s1,f0,g1);
   and(s2,f0,f1,g2);
   and(s3,f0,f1,f2,g3);
   or(greater,g0,s1,s2,s3);
   and(s4,f0,l1);
   and(s5,f0,f1,l2);
   and(s6,f0,f1,f2,l3);
   or(less,l0,s4,s5,s6);
endmodule
module a11_tb;
    reg [0:3] Data_in_A;
    reg [0:3] 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 = 4'b0101;
        Data_in_B =4'b1010;
        #100;
        Data_in_A = 4'b1110;
        Data_in_B = 4'b1001;
        #100;
        Data_in_A = 4'b0110;
        Data_in_B = 4'b0110;
        #100;
    end
endmodule
```