Digital Design Lab 2

Combinational Circuits

Name: Karim Mahmoud Kamal

AUC ID: V23010174

Section: 16

Eng. Mohamed El Shafey

2x1 Multiplexer

Structural Model:

```
module Mux2t01_gatelevel (I0, I1, select, Y);
    input I0;
    input I1;
    input select;
    output Y;
    wire w1;
    wire w2;
    and g1(w1, I0, select);
    and g2(w2, I1, ~select);
    or g3(Y, w1, w2);
endmodule
```

Behavioral Model:

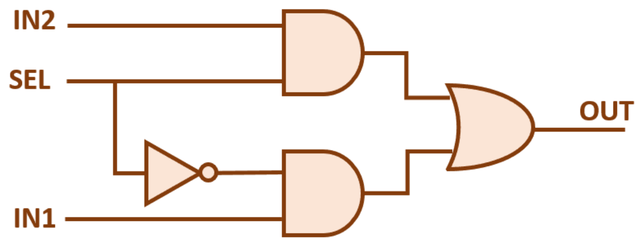
```
module mux2to1_behavioral (I0, I1, select, Y);
    input I0;
    input I1;
    input select;
    output reg Y;
    always @ (*)  begin
```

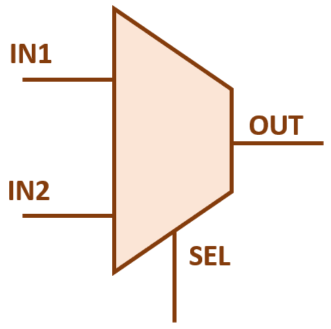
```
     if (select==0)
        Y=I0;
     else if (select==1)
        Y=I1;
    end
endmodule
```

<u>Dataflow Model:</u>

```
module mux2to1_data_flow (I0, I1, select, Y);
    input I0;
    input I1;
    input select;
    output Y;

    assign Y= (~select & I0) | (select & I1);
endmodule
```

Netlist and symbol of the Mux:





4x1Multiplexer

Structural Modeling:

```
module mux4to1_structural (I, S, Y);
    input [3:0] I;
    input [1:0] S;
    output Y;
    wire w1;
    wire w2;
    wire w3;
    wire w4;

    and g1(w1, ~S[1], ~S[0], I[0]);
    and g2(w2, ~S[1], S[0], I[1]);
    and g3(w3, S[1], ~S[0], I[2]);
    and g4(w4, S[1], S[0], I[3]);

    or g5(Y, w1, w2, w3, w4);


endmodule
```

__behavioral Modeling:__

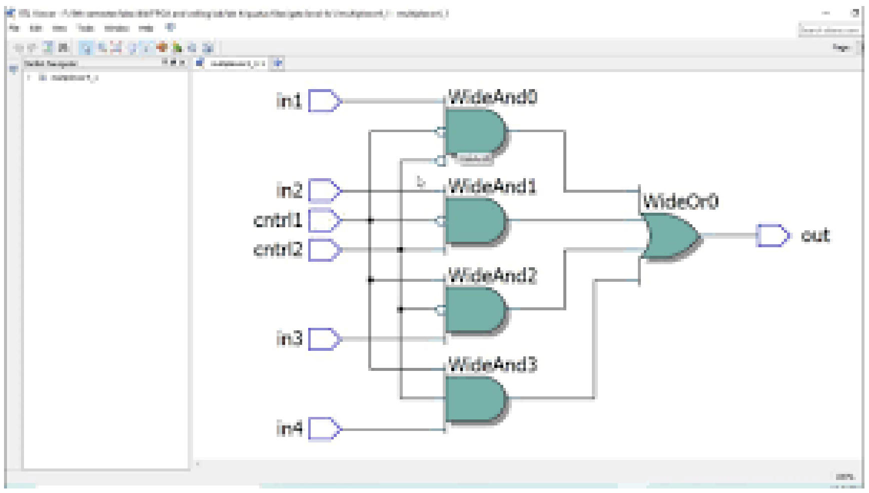
```
module mux4to1_behavioral (I, S, Y);

     input [3:0] I;

     input [1:0] S;

     output reg Y;


     always @ (I or S)

       begin

         case (S)

           2`b00 : Y=I[0];

           2`b01 : Y=I[1];

           2`b10 : Y=I[2];

           2`b11 : Y=I[3];


         endcase

       end
```

```
    endmodule
```

DataFlow Modeling:

```
module mux4to1_dataflow (I, S, Y);
    input [3:0] I;
    input [1:0] S;
    output Y;
    assign Y=(~S[1] & ~S[0] & I[0]) |
   (~S[1] & S[0] & I[1]) | (S[1] & ~S[0] &
    I[2]) | (S[1] & S[0] & I[3]);
endmodule
```

Netlist



Decoder

DataFlow Modeling:

```
module decoder_data_flow (A, B, C, D);
    input A;
    input B;
    input C;
    output [7:0]D;
    assign D[0]=~A & ~B & ~C;
    assign D[1]=~A & ~B &C;
    assign D[2]=~A & B & ~C;
    assign D[3]=~A & B & C;
    assign D[4]=A & ~B & ~C;
    assign D[5]=A & ~B & C;
    assign D[6]=A & B & ~C;
    assign D[7]=A & B & C;
endmodule
```

---

behavioral Modeling:

```
module decoder behavioral (I, D);
     input [2:0] I;
     output reg [7:0] D;
```

```
    always @ (I)
     begin
       case (I)
               3`b000: D=8`b10000000;
               3`b001: D=8`b01000000;
               3`b010: D=8`b00100000;
               3`b011: D=8`b00010000;
               3`b100: D=8`b00001000;
               3`b101: D=8`b00000100;
               3`b110: D=8`b00000010;
               3`b111: D=8`b00000001;
       endcase
     end
endmodule
```

Display Adder

Model for Adder Module:

```
module adderx (a, b, s, c);
    input a, b;
    output s, c;
    assign s = a ^ b;    // sum using xor
    assign c =  a & b;   // carry
endmodule
```

<u>**Model for Decoder Module:**</u>

```
module decoderx (s0, s1, a, b, c, d, e, f, g);
input s0, s1;
output a, b, c, d, e, f, g;

    assign a = ~s0;
    assign b = 1;
    assign c = ~s1;
    assign d = ~s0;
    assign e = ~s0;
    assign f = ~s1 & ~s0;
    assign g = s1 & ~s0;
endmodule
```

Model for Top Module:

```
module Display_adder (x, y, a, b, c, d, e, f, g);
   input x, y;
   output a, b, c, d, e, f, g;
   wire w0, w1;
   adderx U1 (x, y, w0, w1);
   decoderx U2 (w0, w1, a, b, c, d, e, f, g);
endmodule
```

Comparator

Behavioral Modeling:

```
module magcomp (A,B,Gt,Lt,Eq);
    input [7:0]A,B;
    output   Gt,Lt,Eq;
    reg   Gt, Lt, Eq;
   always @ (A or B)


     begin
      Gt <= ( A > B )? 1'b1 : 1'b0;
      Lt <= ( A < B )? 1'b1 : 1'b0;
      Eq <= ( A == B)? 1'b1 : 1'b0;
```

```
        end
endmodule
```

Lab 3 Assignment

Half Adder Module:

```
module half_adder ( a , b , sum , carry );
    input a,b;
    output sum, carry;

    xor x1 ( sum, a , b );
    and a1 ( carry , a , b );
endmodule
```

Full Adder Module:

```
module full_adder ( in1 , in2 , Cin , sum , carry );
    input in1, in2, Cin;
    output sum, carry;
    wire ws1, wc1, wc2;
    half_adder ha1 ( in1 , in2 , ws1 , wc1 );
    half_adder ha2 ( ws1 , Cin , sum , wc2 );
    or r1 ( carry , wc1 , wc2 );
endmodule
```

<u>**Four Bit Comparator Module:**</u>

```
module four_bit_comparator( x , y , v , n , z );
    input [3:0] x;
    input [3:0] y;
    output v,n,z;
    wire s0, s1, s2, s3;
    wire c1, c2, c3, c4;
    full_adder fa1 ( x[0] , ~y[0] , 1'b1 , s0 , c1 );
    full_adder fa2 ( x[1] , ~y[1] , c1 , s1 , c2 );
    full_adder fa3 ( x[2] , ~y[2] , c2 , s2 , c3 );
    full_adder fa4 ( x[3] , ~y[3] , c3 , s3 , c4 );
    nor nor1 ( z , s0 , s1 , s2 , s3 );
    xor xor1 ( v , c3 , c4 );
    assign n = s3;
endmodule
```

Bonus

BCD Adder Module

```
module BCD_adder( x , y , s , carry );
    input [3:0] x;
    input [3:0] y;
    output [3:0] s;
    output carry;
    wire s0, s1, s2, s3;
    wire c1, c2, c3, c4;
    full_adder fa1 ( x[0] , y[0] , 1'b0 , s0 , c1 );
    full_adder fa2 ( x[1] , y[1] , c1 , s1 , c2 );
    full_adder fa3 ( x[2] , y[2] , c2 , s2 , c3 );
    full_adder fa4 ( x[3] , y[3] , c3 , s3 , c4 );
    assign carry = c4;
endmodule
```

8-Bit BCD Adder

```
module BCD_adder_final( x , y , s , carry, carry2 );
    input [3:0] x;
    input [3:0] y;
    output [3:0] s;
    output carry , carry2;

    wire [3:0] ss;

    wire c1, w1 ,w2;

    BCD_adder bcd1 ( x , y , ss , c1 );

    and a1 ( w1 , ss[3] , ss[2] );

    and a2 ( w2 , ss[3] , ss[1] );

    or o1  ( carry , w1 , w2 , c1 );

    BCD_adder bcd2 ( ss , {1'b0,carry,carry,1'b0} , s , carry2 );

endmodule


//-------------- behavioral -------------------------------------------

module behavioral_four_bit_comparator( x , y , v , n , z );

    input [3:0] x;

    input [3:0] y;

    output v,n,z;

    reg v,n,z;

    always @ (*) begin

            case({x,y})

                    8'b0000_0000: begin

                            v= 1'b0;

                            z= 1'b1;

                            n= 1'b0;
```

```
end
8'b1111_1111: begin
        v= 1'b0;
        z= 1'b1;
        n= 1'b0;
end
8'b1000_0000: begin
        v= 1'b0;
        z= 1'b0;
        n= 1'b1;
end
8'b0000_1000: begin
        v= 1'b0;
        z= 1'b1;
        n= 1'b0;
end
8'b0001_0000: begin
        v= 1'b1;
        z= 1'b0;
        n= 1'b1;
end
```

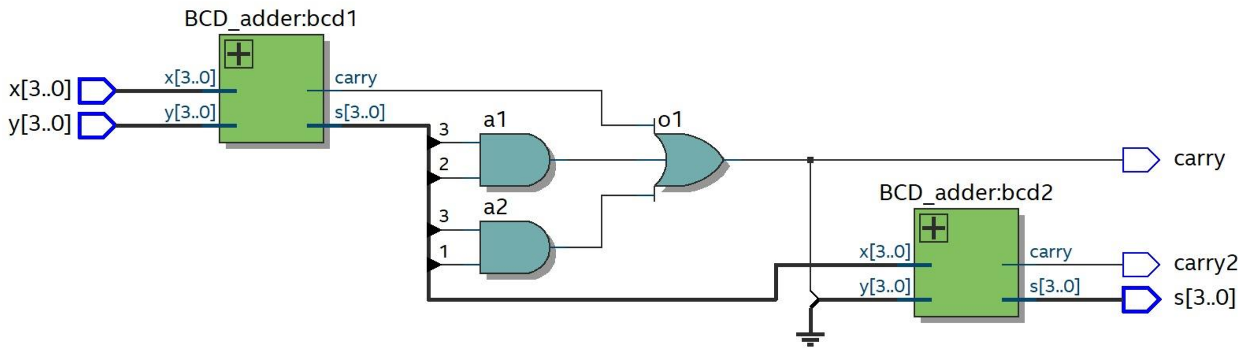
```
8'b0000_0001: begin
        v= 1'b0;
        z= 1'b0;
        n= 1'b0;
end
8'b1011_1001: begin
        v= 1'b0;
        z= 1'b0;
        n= 1'b1;
end
8'b1100_1001: begin
        v= 1'b0;
        z= 1'b0;
        n= 1'b0;
end
8'b1010_0010: begin
        v= 1'b0;
        z= 1'b0;
        n= 1'b0;
end
8'b0101_1100: begin
        v= 1'b;
```

```
                            z= 1'b;
                            n= 1'b;
                        end
                endcase
        end
endmodule
```

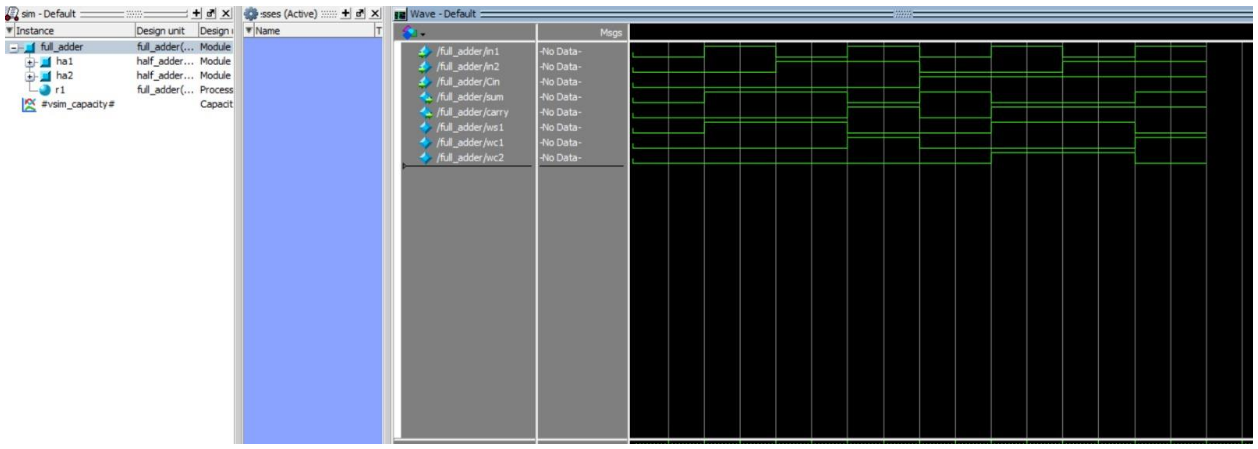
Behavioral of BCD Adder

```
module modulebcdadd (cin, X, Y, S, cout);
input cin;
input [3:0] X, Y;
output reg [3:0] S;
output reg cout;
 reg [4:0] Z;
always @(X, Y, cin)
    begin
      Z = X + Y + cin;
       if (Z < 10)
          {cout, S} = Z;
       else
         {cout, S} = Z + 6;
    end

endmodule
```

8-Bit BCD Adder Netlist



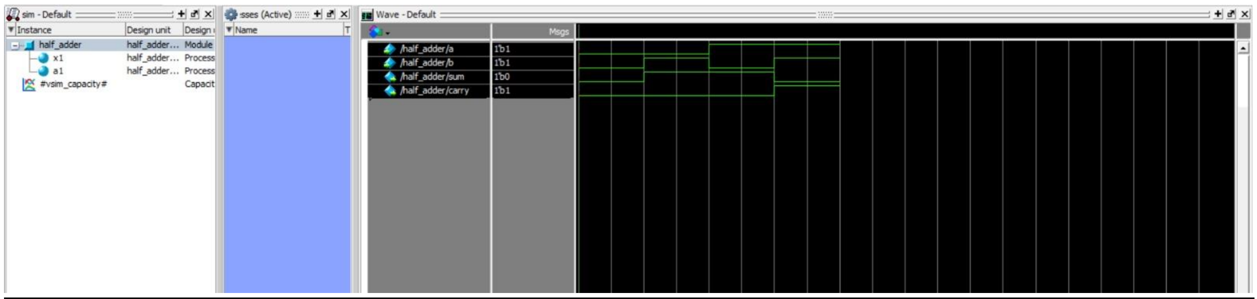
Test Bench of Full Adder



Instance	Design unit	Design
full_adder	full_adder(...	Module
ha1	half_adder...	Module
ha2	half_adder...	Module
r1	full_adder(...	Process
#vsim_capacity#		Capacit

```
/full_adder/in1    -No Data-
/full_adder/in2    -No Data-
/full_adder/Cin    -No Data-
/full_adder/sum    -No Data-
/full_adder/carry  -No Data-
/full_adder/ws1    -No Data-
/full_adder/wc1    -No Data-
/full_adder/wc2    -No Data-
```

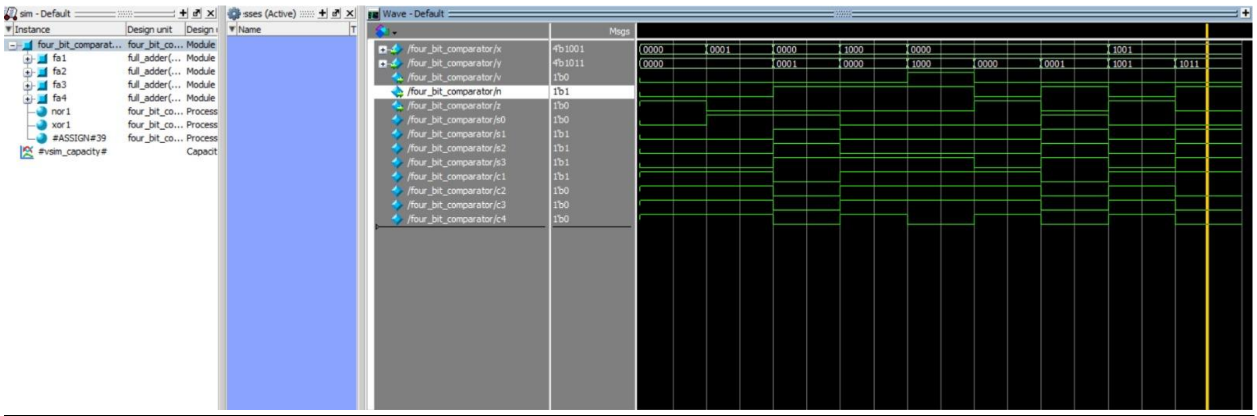
Test Bench of Half Adder



```
sim - Default
Instance            Design unit     Design
half_adder          half_adder...   Module
  x1                half_adder...   Process
  a1                half_adder...   Process
#vsim_capacity#                     Capacit
```

```
Wave - Default                     Msgs
/half_adder/a                      1'b1
/half_adder/b                      1'b1
/half_adder/sum                    1'b0
/half_adder/carry                  1'b1
```

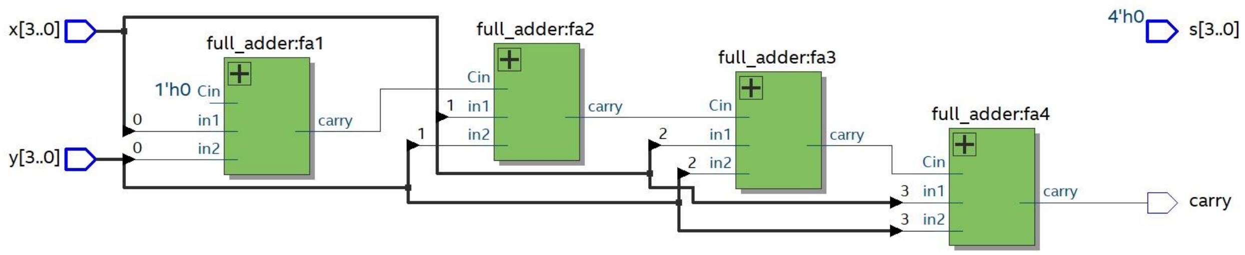
Test Bench of 4-bit Comparator



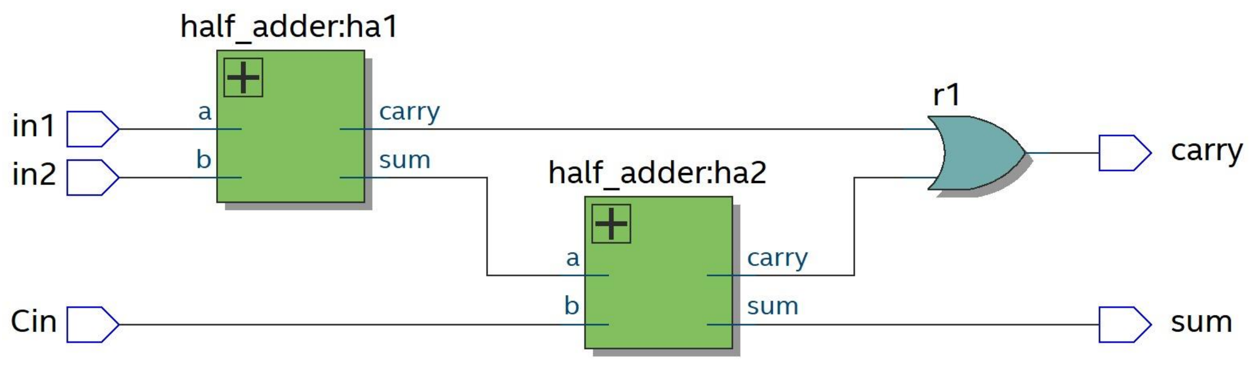
```
sim - Default
Instance            Design unit     Design
four_bit_comparat... four_bit_co... Module
  fa1               full_adder(...  Module
  fa2               full_adder(...  Module
  fa3               full_adder(...  Module
  fa4               full_adder(...  Module
  nor1              four_bit_co...  Process
  xor1              four_bit_co...  Process
  #ASSIGN#39        four_bit_co...  Process
#vsim_capacity#                     Capacit
```

```
Wave - Default                     Msgs
/four_bit_comparator/x             4'b1001   0000  0001  0000  1000  0000  1001
/four_bit_comparator/y             4'b1011   0000  0001  1000  0000  0001  1001  1011
/four_bit_comparator/v             1'b0
/four_bit_comparator/n             1'b1
/four_bit_comparator/z             1'b0
/four_bit_comparator/s0            1'b0
/four_bit_comparator/s1            1'b1
/four_bit_comparator/s2            1'b1
/four_bit_comparator/s3            1'b1
/four_bit_comparator/c1            1'b1
/four_bit_comparator/c2            1'b0
/four_bit_comparator/c3            1'b0
/four_bit_comparator/c4            1'b0
```

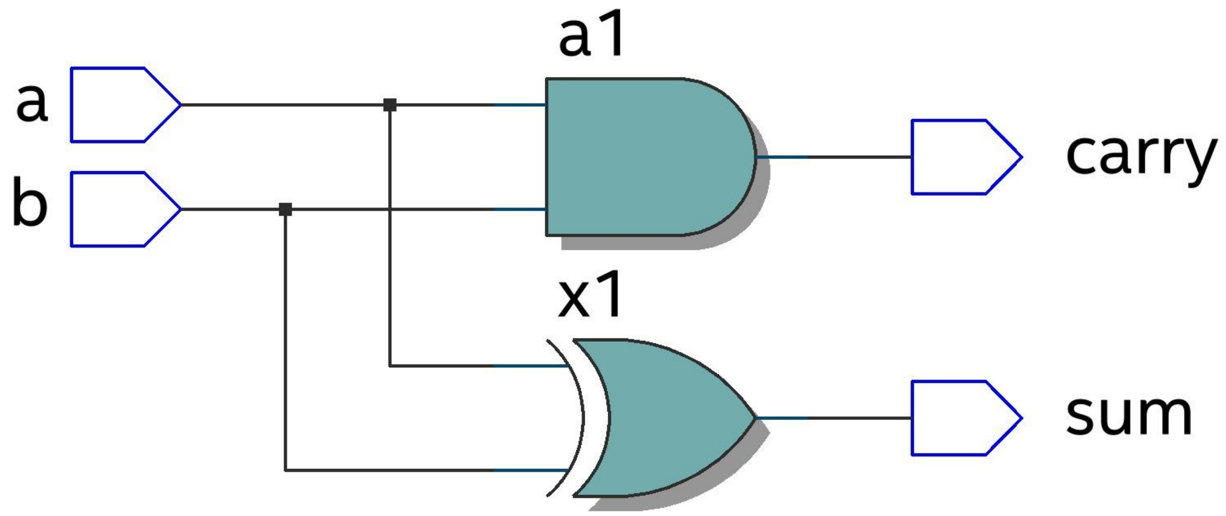
Netlist of 4-Bit BCD Adder



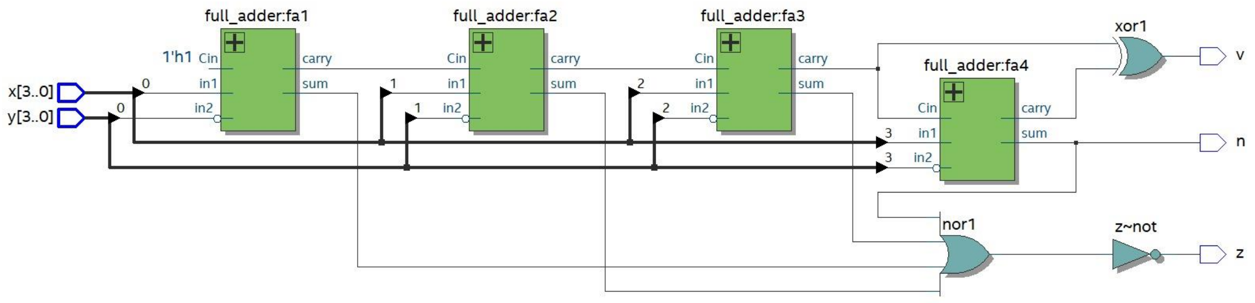
Netlist of Full Adder



Netlist of Half Adder



Netlist of Comparator


full_adder:fa1
full_adder:fa2
full_adder:fa3
full_adder:fa4
1'h1
Cin
in1
in2
carry
sum
x[3..0]
y[3..0]
0
1
2
3
xor1
nor1
z~not
v
n
z