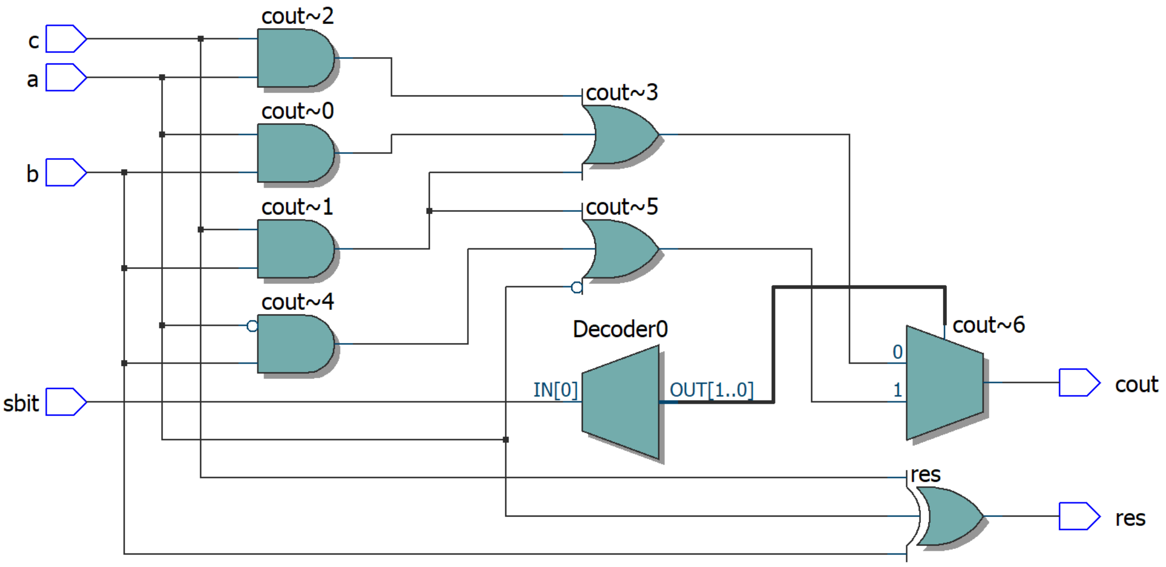
1) 1 bit adder and subtractor

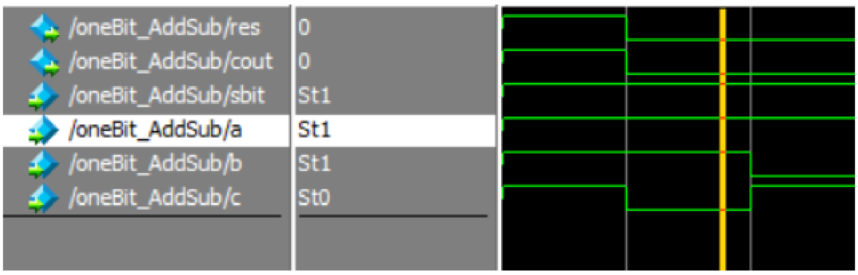
Code:

```
module oneBit_AddSub(res,cout,sbit,a,b,c);

output reg res,cout;
input sbit,a,b,c;

always@(a,b,c,sbit)begin
case(sbit)
        1'b0:begin
                res = a^b^c;
                cout = (a&b)|(c&b)|(a&c);
        end

        1'b1:begin
                res = a^b^c;
                cout = (~a & b) | (b & c) | (~& a);
        end

endcase
end

endmodule
```

Output;

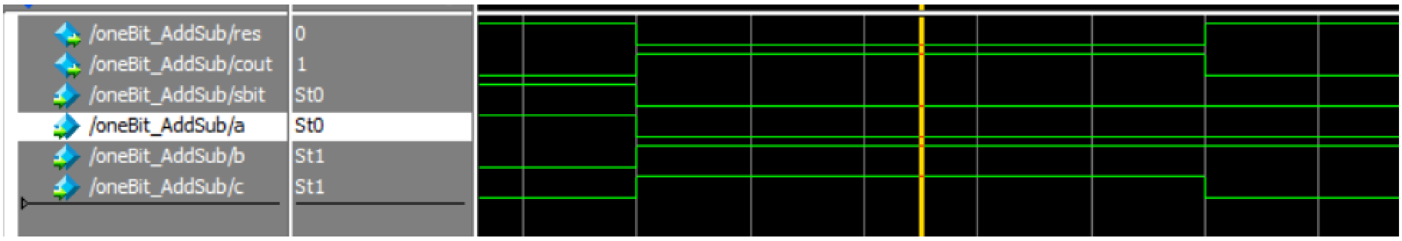


Subtraction:



```
/oneBit_AddSub/res    0
/oneBit_AddSub/cout   0
/oneBit_AddSub/sbit   St1
/oneBit_AddSub/a      St1
/oneBit_AddSub/b      St1
/oneBit_AddSub/c      St0
```

Addition:



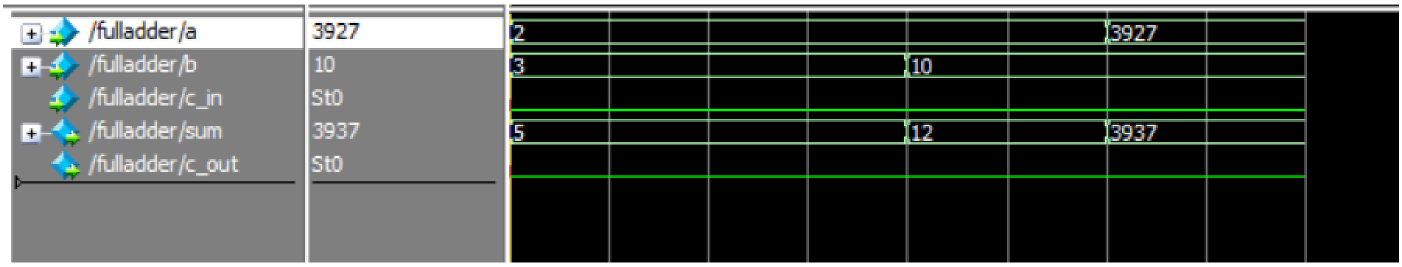
```
/oneBit_AddSub/res    0
/oneBit_AddSub/cout   1
/oneBit_AddSub/sbit   St0
/oneBit_AddSub/a      St0
/oneBit_AddSub/b      St1
/oneBit_AddSub/c      St1
```

2) 16 bit full adder.

```
module fulladder #(parameter N = 16)(
  input wire [N-1:0] a,
  input wire [N-1:0] b,
  input wire c_in,
  output wire [N-1:0] sum,
  output wire c_out
);

              wire [N:0]temp;
              wire [N-1:0]c_next;

              assign temp[0] = c_in;

  genvar i;
  generate
    for (i = 0; i < 16; i = i+1) begin : bit_adder
      full_adder fa (
        .a(a[i]),
        .b(b[i]),
        .c_in(temp[i]),
        .sum(sum[i]),
        .c_out(c_next[i])
      );
      assign temp[i+1] = c_next[i];
    end
  endgenerate
  assign c_out = c_next[N-1];
endmodule

module full_adder (
  input wire a,
  input wire b,
  input wire c_in,
  output wire sum,
  output wire c_out
);
  assign sum = a ^ b ^ c_in;
  assign c_out = (a & b) | (c_in & (a ^ b));
endmodule
```

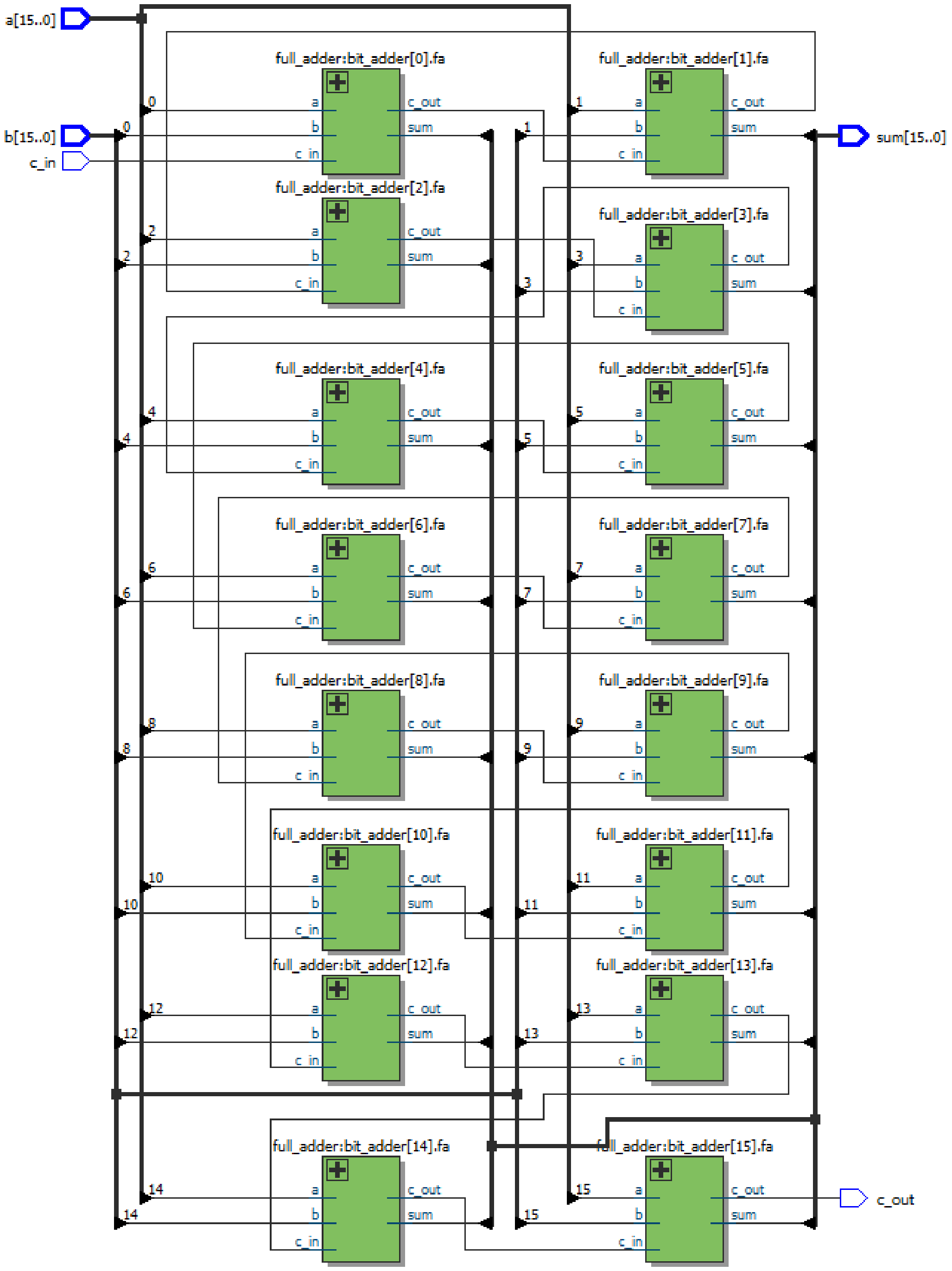
Output

Simulation Results:



```
/fulladder/a      3927    2          3927
/fulladder/b      10      3      10
/fulladder/c_in   St0
/fulladder/sum    3937    5      12  3937
/fulladder/c_out  St0
```

RTL View:

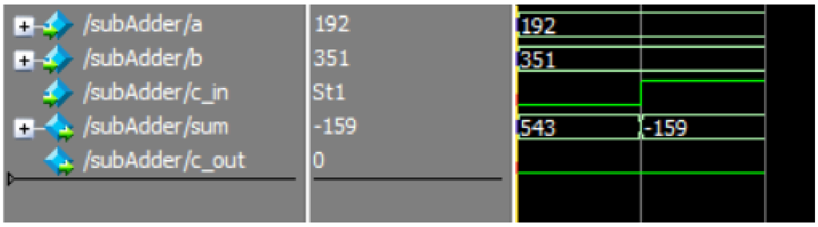

a[15..0]
b[15..0]
c_in
sum[15..0]
c_out
full_adder:bit_adder[0].fa
full_adder:bit_adder[1].fa
full_adder:bit_adder[2].fa
full_adder:bit_adder[3].fa
full_adder:bit_adder[4].fa
full_adder:bit_adder[5].fa
full_adder:bit_adder[6].fa
full_adder:bit_adder[7].fa
full_adder:bit_adder[8].fa
full_adder:bit_adder[9].fa
full_adder:bit_adder[10].fa
full_adder:bit_adder[11].fa
full_adder:bit_adder[12].fa
full_adder:bit_adder[13].fa
full_adder:bit_adder[14].fa
full_adder:bit_adder[15].fa
a
b
c_in
c_out
sum
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

3) 16 bit adder subtractor.

Code:

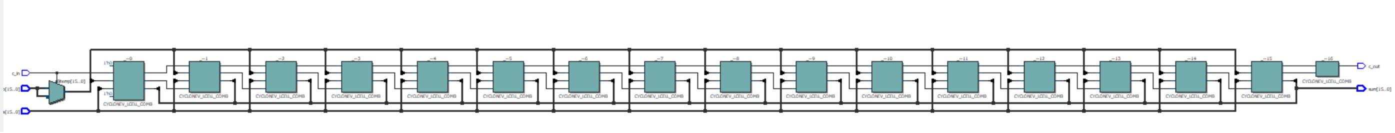
```
module subAdder #(parameter N = 16)(a,b,c_in,sum,c_out);

input [N-1:0]a,b;
input c_in;

output reg [N-1:0]sum;
output reg c_out;

reg [N-1:0] Btemp;

always@(*)begin
        if(c_in)begin
                Btemp = ~b;
        end
        else Btemp = b;
        {c_out,sum} = a + Btemp + c_in;
end

endmodule
```

Simulation:



```
/subAdder/a      192    192
/subAdder/b      351    351
/subAdder/c_in   St1
/subAdder/sum    -159   543   -159
/subAdder/c_out  0
```

RTL View:



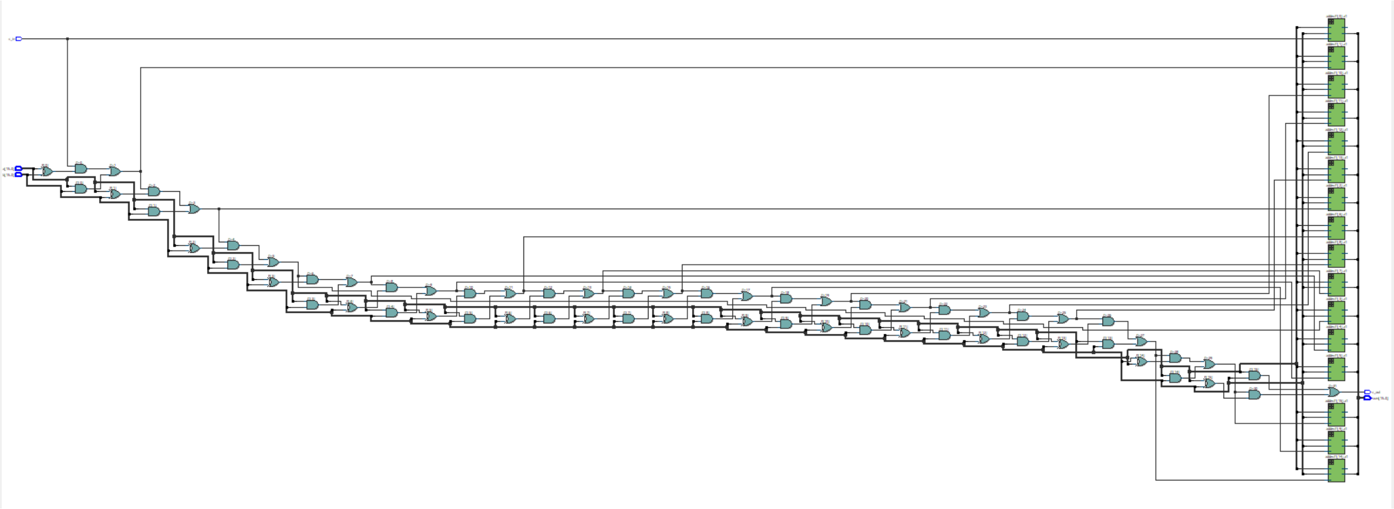
4) **Look ahead carry adder**

Code:

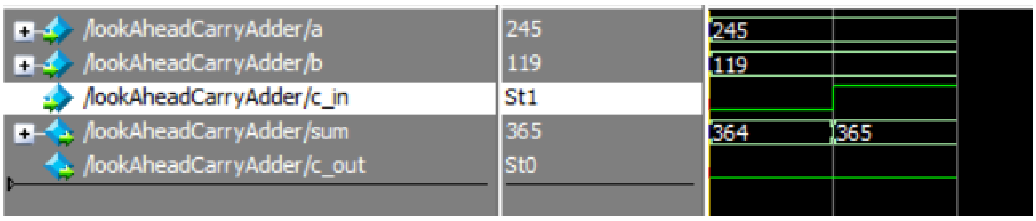
```
module lookAheadCarryAdder #(parameter N = 16)(input [N-1:0]a,b,
                                input c_in,
                                output [N-1:0] sum,
                                output c_out);

wire [N:0]C;
wire [N-1:0] G,P,SUM;

assign C[0] = c_in;


genvar i;
generate
for(i = 0; i < N; i = i+1)begin:l1
        adder a1(.a(a[i]),.b(b[i]),.c(C[i]),.sum(SUM[i]),.cout());
        end
endgenerate

genvar j;
generate
```

```
for(j = 0; j < N; j = j+1)begin:l2
        assign G[j] = a[j] & b[j];
        assign P[j] = a[j] ^ b[j];
        assign C[j+1] = G[j] | (P[j]&C[j]);
        end
endgenerate


assign sum = SUM;
assign c_out = C[N];

endmodule



module adder(input a,b,c,
                output reg sum,cout);

always@(*)begin
{cout,sum} = a+b+c;
end
endmodule
```

RTL View:



Simutlation:



```
/lookAheadCarryAdder/a        245    245
/lookAheadCarryAdder/b        119    119
/lookAheadCarryAdder/c_in     St1
/lookAheadCarryAdder/sum      365    364    365
/lookAheadCarryAdder/c_out    St0
```