EEL2020 DIGITAL DESIGN
LAB 3

Name: Dhruv
Roll No.: B20EE016

Work 1: Modify the Four Bit Adder created in Lab 1 to implement an Adder-Subtractor Unit.

Half Adder(with delay) Verilog Code:

```
`timescale 1ns / 1ps
module HalfAdder(a,b,s,c);
   input a,b;
   output s,c;
   assign #10 s=a^b;
   assign #5 c=a&b;
endmodule
```

Full Adder(with delay) Verilog Code:

```
`timescale 1ns / 1ps

module Full_Adder(a,b,c,sum,carry);
   input a,b,c;
   output sum,carry;
   wire sum1,carry1,carry2;
   HalfAdder H1(b,c,sum1,carry1);
   HalfAdder H2(a,sum1,sum,carry2);
   assign #5 carry=carry1|carry2;
endmodule
```

Multiplexer(with delay) Verilog Code:

```
`timescale 1ns / 1ps
module multiplexer(input i0,i1,s,output reg bitout);
always@(i0,i1,s)
begin
if(s == 0)
bitout = i0;
```

else
bitout = i1;
end
endmodule

Adder Subtractor Verilog Code:

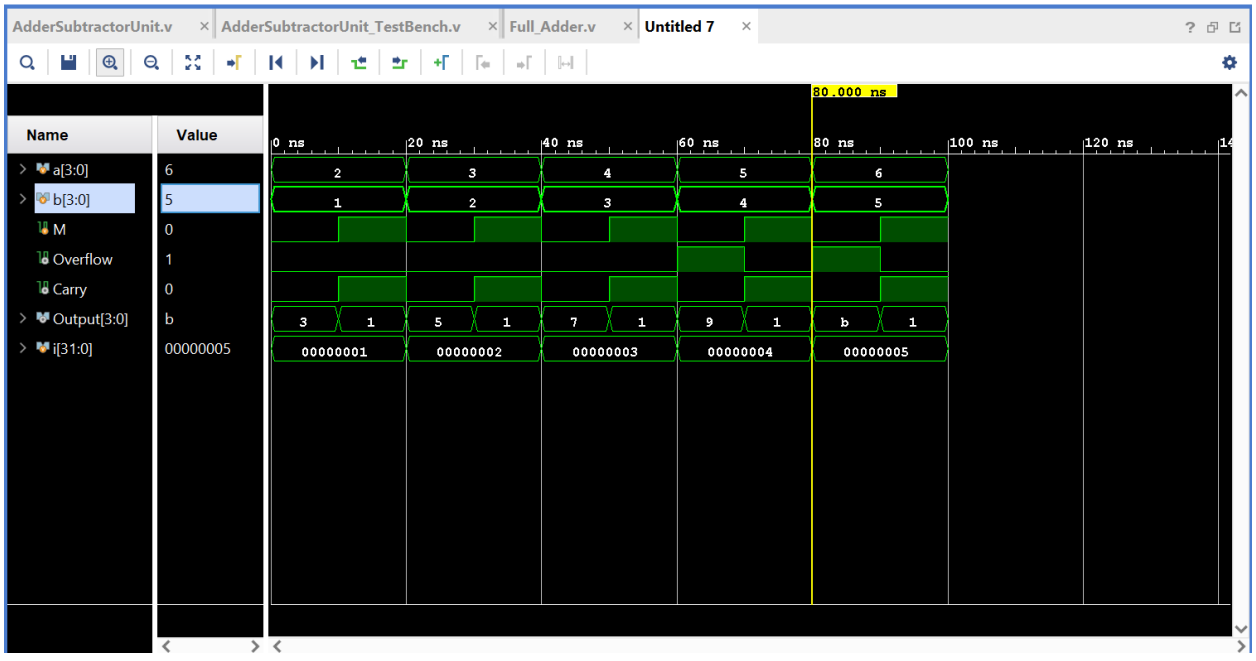
```
module AdderSubtractorUnit(a,b,M,Output,Overflow,Carry);
    input [3:0] a,b;
    input M;
    output [3:0] Output;
    output Overflow,Carry;
    wire c1,c2,c3,c4,b0,b1,b2,b3;
    assign b0=b[0]^M;
    assign b1=b[1]^M;
    assign b2=b[2]^M;
    assign b3=b[3]^M;

    Full_Adder F1(a[0],b0,M,Output[0],c1);
    Full_Adder F2(a[1],b1,c1,Output[1],c2);
    Full_Adder F3(a[2],b2,c2,Output[2],c3);
    Full_Adder F4(a[3],b3,c3,Output[3],c4);
    assign Carry=c4;
    assign Overflow =c3^c4;
endmodule
```

Note-Delays were used only in Fast Adders

TestBench:

```
module AdderSubtractorUnit_TestBench();
    reg [3:0] a,b;
    reg M;
    wire Overflow,Carry;
    wire [3:0] Output;
    integer i;
    AdderSubtractorUnit A1(a,b,M,Output,Overflow,Carry);
    initial
    begin
        for(i = 1; i<9; i=i+1)
            begin
                a = i+1;
                b = i;
               assign M = 0;#10;
               assign M = 1; #10;
            end
    end
initial #100 $finish;
endmodule
```

Simulation:

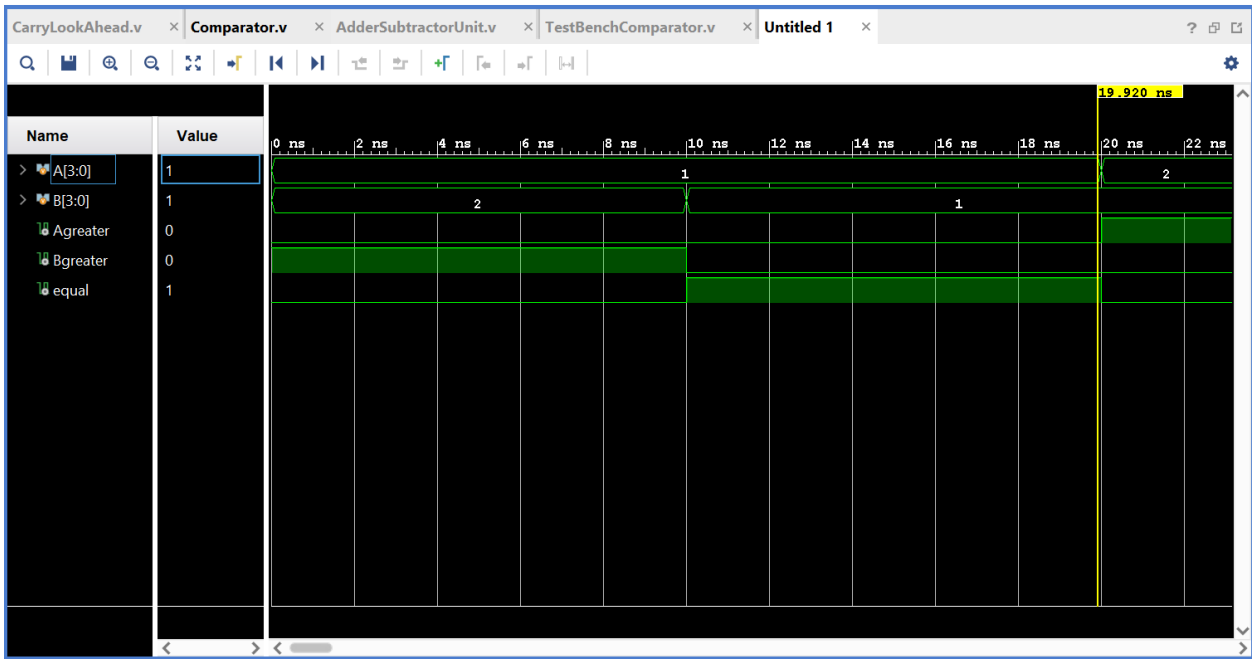


Comparator verilog code:

```
22
23  module Comparator(A,B,Agreater,Bgreater,equal);
24  input [3:0] A,B;
25  output Agreater, Bgreater, equal;
26  wire [3:0]Sum;
27  wire C,Overflow,temp;
28
29  AdderSubtractorUnit A1(A,B,1,Sum,Overflow,C);
30
31  assign equal = (A == B)? 1 : 0;
32  xnor (temp,Overflow,Sum[3]);
33  assign Agreater = (equal == 0 & temp == 1)?1:0;
34  xor(Bgreater,Overflow,Sum[3]);
35
36
37  endmodule
38
39
```

```
21
22
23  module TestBenchComparator( );
24  reg [3:0]A;
25  reg [3:0] B;
26  wire Agreater,Bgreater,equal;
27  Comparator C(A,B,Agreater,Bgreater,equal);
28
29  initial
30  begin
31  assign A = 1;
32  assign B = 2;
33  #10;
34  assign A = 1;
35  assign B = 1;
36  #10;
37  assign A = 2;
38  assign B = 1;
39  #10;
40  end
41  endmodule
42
43
```

Simulation:



Work 2: 2. Write a Program to Implement the following Fast Adders [32 bit]:

a) Carry Look Ahead Adder Verilog Code:

```
2
3  module CarryLookAhead(A,B,Cin,OutputSum,out);
4  input [31:0] A,B;
5  input Cin;
6  output wire [31:0] OutputSum;
7  output out;
8  wire [8:0] Carry;
9
0  CarryLookAhead4Bit A1(A[3:0],B[3:0],Cin,OutputSum[3:0],Carry[0]);
1  CarryLookAhead4Bit A2(A[7:4],B[7:4],Cin,OutputSum[7:4],Carry[1]);
2  CarryLookAhead4Bit A3(A[11:8],B[11:8],Cin,OutputSum[11:8],Carry[2]);
3  CarryLookAhead4Bit A4(A[15:12],B[15:12],Cin,OutputSum[15:12],Carry[3]);
4  CarryLookAhead4Bit A5(A[19:16],B[19:16],Cin,OutputSum[19:16],Carry[4]);
5  CarryLookAhead4Bit A6(A[23:20],B[23:20],Cin,OutputSum[23:20],Carry[5]);
6  CarryLookAhead4Bit A7(A[27:24],B[27:24],Cin,OutputSum[27:24],Carry[6]);
7  CarryLookAhead4Bit A8(A[31:28],B[31:28],Cin,OutputSum[31:28],Carry[7]);
8
9
0
1  endmodule
```

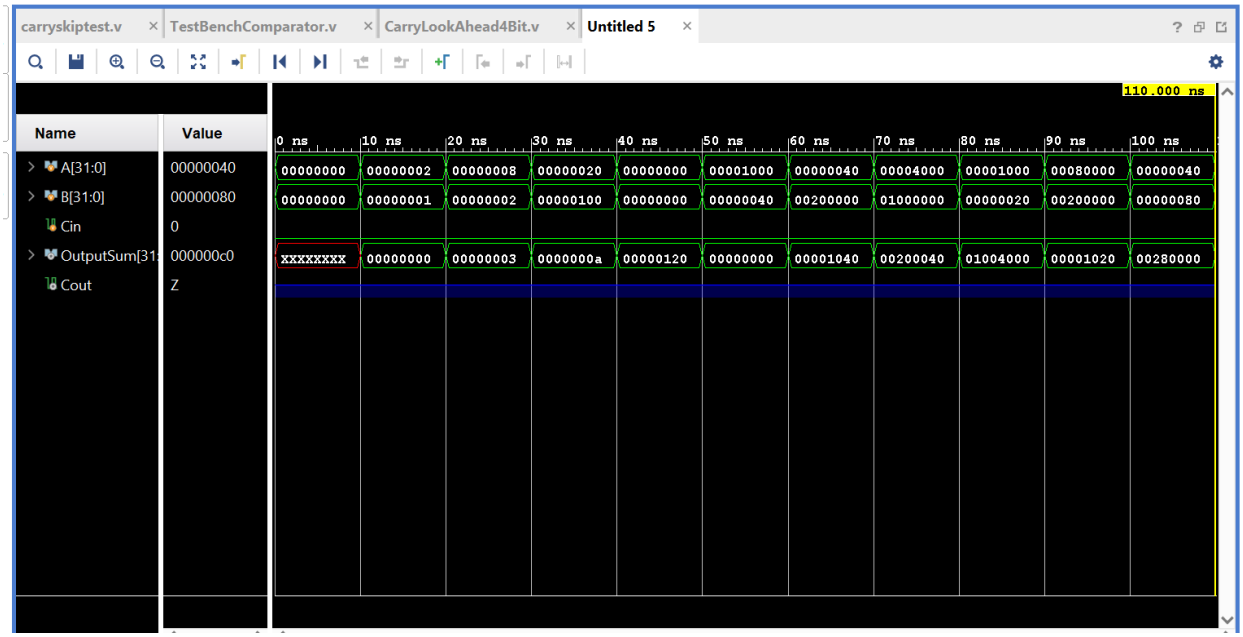
```
22
23  module CarryLookAhead4Bit(A,B,Cin,OutputSum,out);
24  input [3:0] A,B;
25  input Cin;
26  output [3:0] OutputSum;
27  output out;
28  wire [3:0] Carry;
29  wire [3:0] Generator, Propagator, Sum;
30  wire tempCarry;
31  genvar i;
32  assign #10 Propagator=A^B;
33  assign #5 Generator=A&B;
34
35  assign Carry[0] = Generator[0] | (Propagator[0] &Cin);
36  for(i = 1 ; i<4; i=i+1)
37  begin
38      assign Carry[i] = Generator[i] | (Propagator[i] & Carry[i-1]);
39  end
40  assign OutputSum=Propagator^Carry;
41  assign out=Carry[3];
42
43  endmodule
44
```

Test Bench:

A	B	Ci	C i+1	Condition
0	0	0	0	No carry generate
0	0	1	0	
0	1	0	0	
0	1	1	1	No carry propagate
1	0	0	0	
1	0	1	1	
1	1	0	1	Carry generate
1	1	1	1	

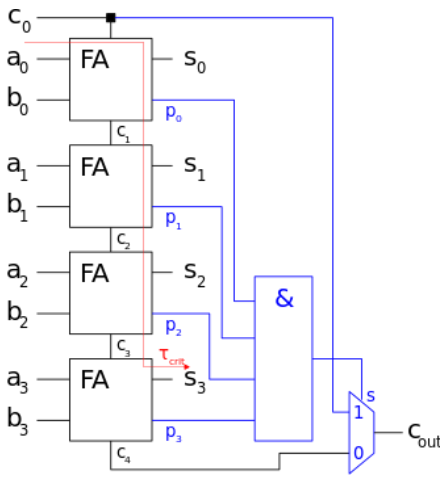
```
module TestCarryLookAhead();
reg [31:0] A,B;
reg Cin;
wire [31:0]OutputSum;
wire Cout;
CarryLookAhead CLA(A,B,Cin,OutputSum,Cout);
initial
    begin
        Cin=1'b0;
        A=32'b00000000000000000000000000000000; B=32'b00000000000000000000000000000000;
        #10 A=32'b00000000000000000000000000000010; B=32'b00000000000000000000000000000001;
        #10 A=32'b00000000000000000000000000001000; B=32'b00000000000000000000000000000010;
        #10 A=32'b00000000000000000000000000100000; B=32'b00000000000000000000000100000000;
        #10 A=32'b00000000000000000000000000000000; B=32'b00000000000000000000000000000000;
        #10 A=32'b00000000000000000001000000000000; B=32'b00000000000000000000000001000000;
        #10 A=32'b00000000000000000000000001000000; B=32'b00000000001000000000000000000000;
        #10 A=32'b00000000000000000100000000000000; B=32'b00000001000000000000000000000000;
        #10 A=32'b00000000000000000010000000000000; B=32'b00000000000000000000000000100000;
        #10 A=32'b00000000000010000000000000000000; B=32'b00000000001000000000000000000000;
        #10 A=32'b00000000000000000000000001000000; B=32'b00000000000000000000000010000000;


    end

initial #110 $finish;
endmodule
```

Simulation:



---

b) Carry Skip Adder Verilog Code:



`timescale 1ns / 1ps

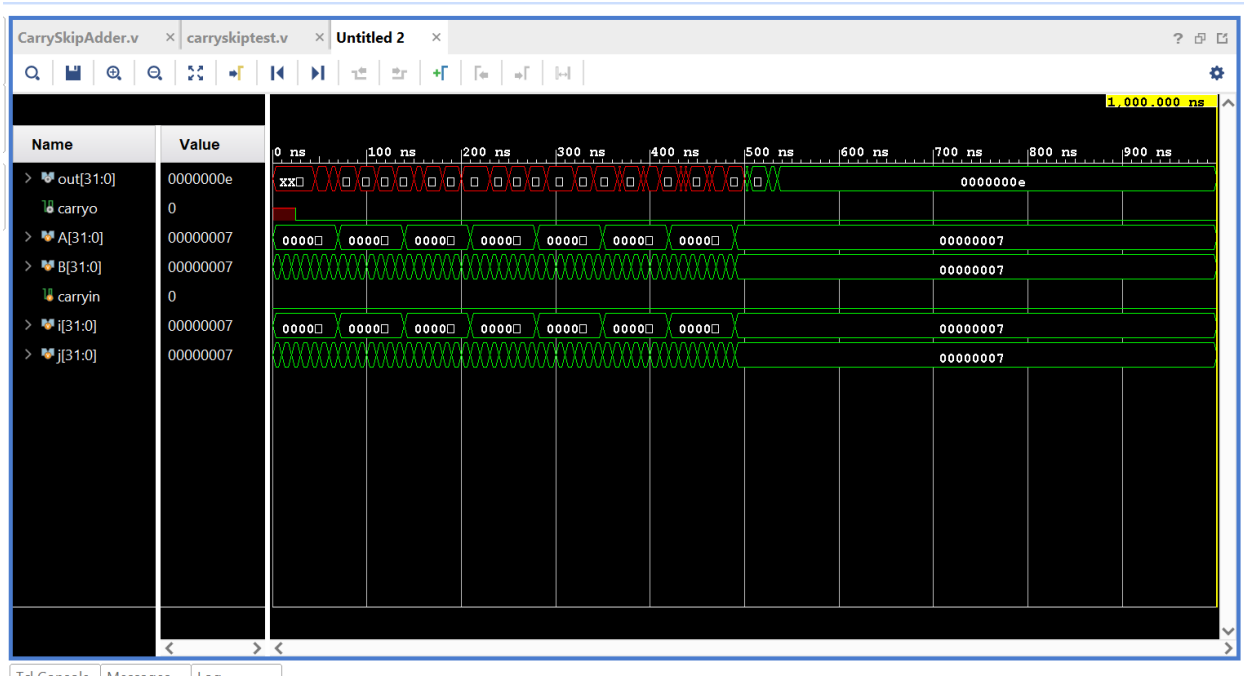
```
22
23  module CarrySkipAdder(out,carryo,A,B,carryin);
24  input [31:0]A,B;
25  output [31:0]out;
26  output carryo;
27  input carryin;
28  wire [31:0]g,p;
29  wire [32:0]c;
30  wire temp;
31  assign c[0] = carryin;
32  genvar i;
33  for (i=0; i<32; i=i+1)
34  begin
35  assign  p[i] = A[i] ^ B[i];
36  assign  c[i+1] = ( A[i] & B[i] ) | ( A[i] & c[i] ) | ( B[i] & c[i] );
37  assign  out[i] = A[i] ^ B[i] ^ c[i];
38  end
39
40  assign temp= p[0];
41  for(i = 1 ; i<32 ; i=i+1)
42  begin
43  assign temp= temp& p[i];
44  end
45
46  assign carryo = temp ? carryin : c[32];
47
48  endmodule
```

Test Bench:

Input			Output	
A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

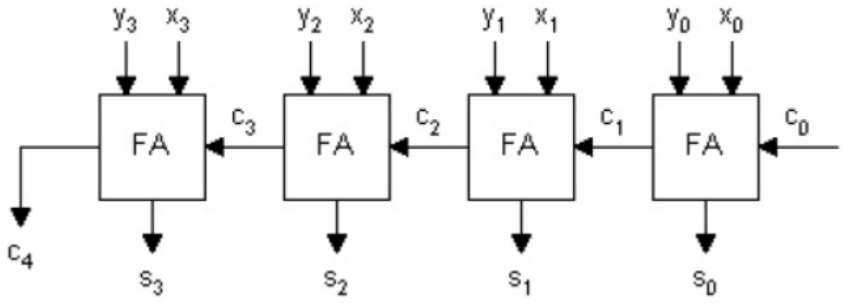
```
22
23  module carryskiptest();
24  wire [31:0]out;
25  wire carryo;
26  reg [31:0]A,B;
27  reg carryin;
28  integer i,j;
29  CarrySkipAdder C(out,carryo,A,B,carryin);
30  initial
31  begin
32  assign carryin = 0;
33  for(i = 0; i<7; i=i+1)
34  begin
35  for(j = 0; j<7 ; j=j+1)
36  begin
37  assign A = i;
38  assign B = j; #10;
39  end
40  end
41  end
42  endmodule
43
44
45
```

Simulation:



---

c) Carry Select Adder Verilog Code:




```
`timescale 1ns / 1ps


module CarrySelectAdder(input [31:0] A,
input[31:0]B, input cin,
output [31:0] S,
output cout);
wire [31:0] temp0,temp1,carry0,carry1;
genvar i;

Full_Adder F0(A[0],B[0],0,temp0[0],carry0[0]);
for(i = 1;i<32 ; i=i+1)
```

```
begin
Full_Adder F1(A[i],B[i],carry0[i-1],temp0[i],carry0[i]);
end

Full_Adder F2(A[0],B[0],1,temp1[0],carry1[0]);
for(i = 1;i<32 ; i=i+1)
begin
Full_Adder F0(A[i],B[i],carry1[i-1],temp1[i],carry1[i]);
end

multiplexer mux1(carry0[31],carry1[31],cin,cout);

for(i = 0; i<32 ; i=i+1)
begin
multiplexer mux2(temp0[i],temp1[i],cin,S[i]);
end
endmodule

module multiplexer(input i0,i1,s,output reg bitout);
always@(i0,i1,s)
begin
if(s == 0)
bitout = i0;
else
bitout = i1;
end
endmodule
```

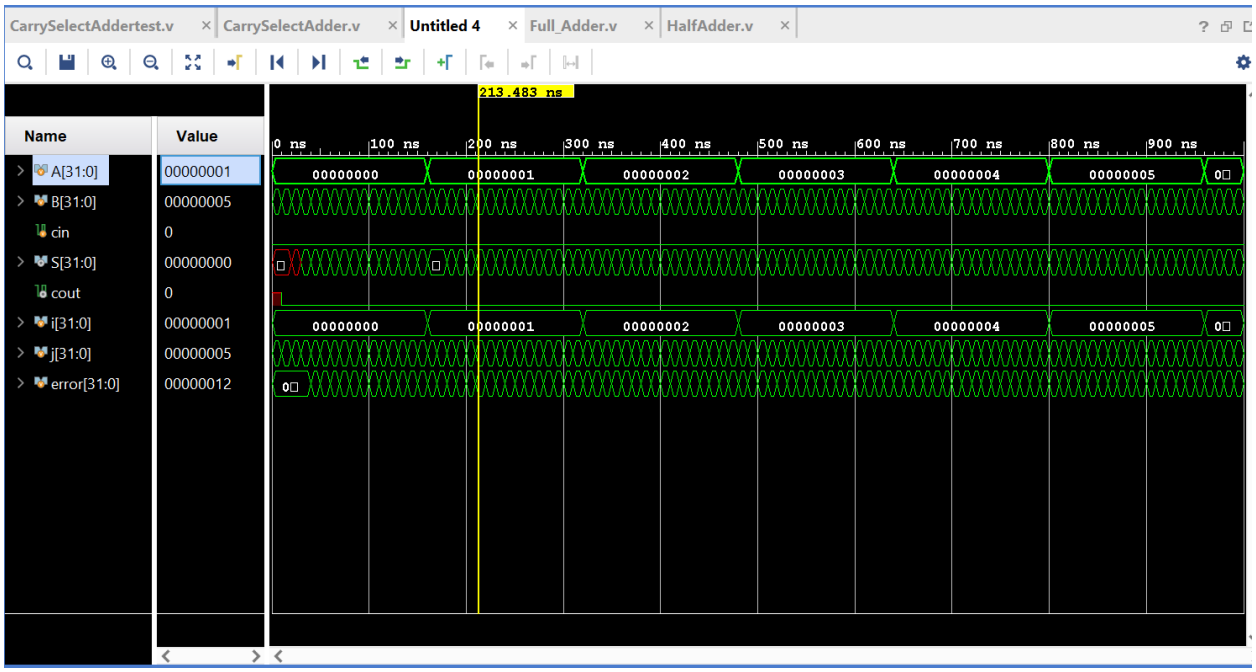
Test Bench:

Inputs			Outputs	
A	B	C	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

```
`timescale 1ns / 1ps


module CarrySelectAddertest( );
reg [31:0] A;
reg [31:0] B;
reg cin;

wire [31:0] S;
wire cout;
integer i,j,error;

CarrySelectAdder CSA (.A(A), .B(B), .cin(cin), .S(S), .cout(cout));

initial begin

A = 0;
B = 0;
error = 0;

cin = 0;
for(i=0;i<16;i=i+1) begin
for(j=0;j<16;j=j+1) begin
A = i;
B = j;
#10;
if({cout,S} != (i+j))
error <= error + 1;
end
end

cin = 1;
for(i=0;i<16;i=i+1) begin
for(j=0;j<16;j=j+1) begin
A = i;

B = j;
#10;
if({cout,S} != (i+j+1))
error <= error + 1;
end
end
```

end
endmodule

Simulation:



Conclusions:

In this fast adder we predict the carry before hand so that the other units in line do not have to wait for the generated carry to arrive. One of the major drawbacks is that it uses a lot of gates or Hardware for higher bits and thus the gate delay affects a lot.