CS303 – Logic & Digital System
Design CS303 – Lab#2 Report

Name-Surname/SID: Jeren Annagurbanova/28517

1. Project Description

Designing a circuit that can both add and subtract two signed 15-bit integers and realize whether there is an overflow or not.

Design:

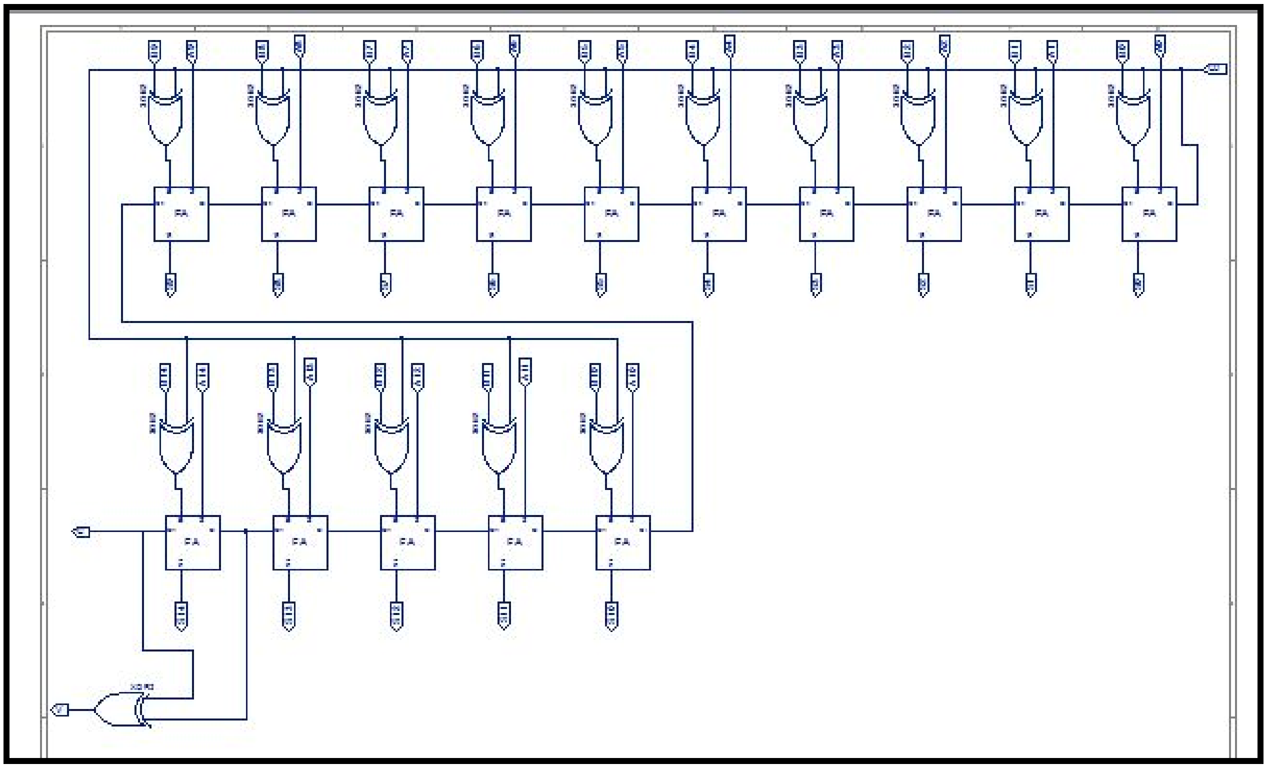
1. 15-bit ripple-carry adder-subtractor using full adders
2. 15-bit hybrid adder-subtractor using five 3-bit carry lookahead adders (CLAs).

Both adder-subtractors must detect overflow

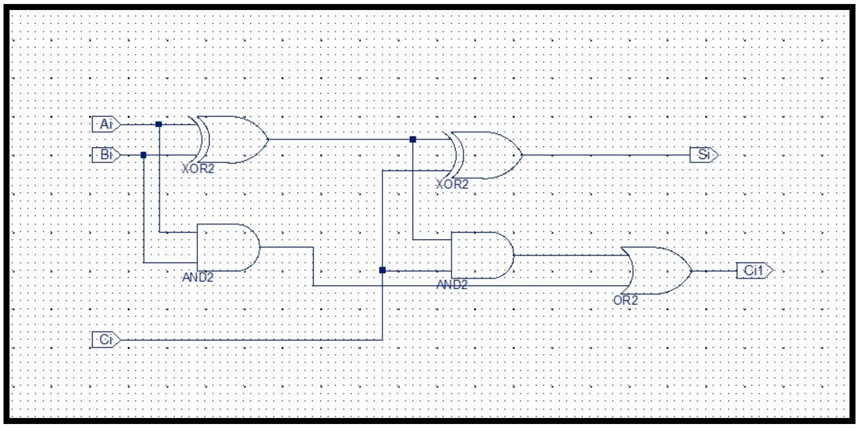
2. First Design

2.1. Schematic

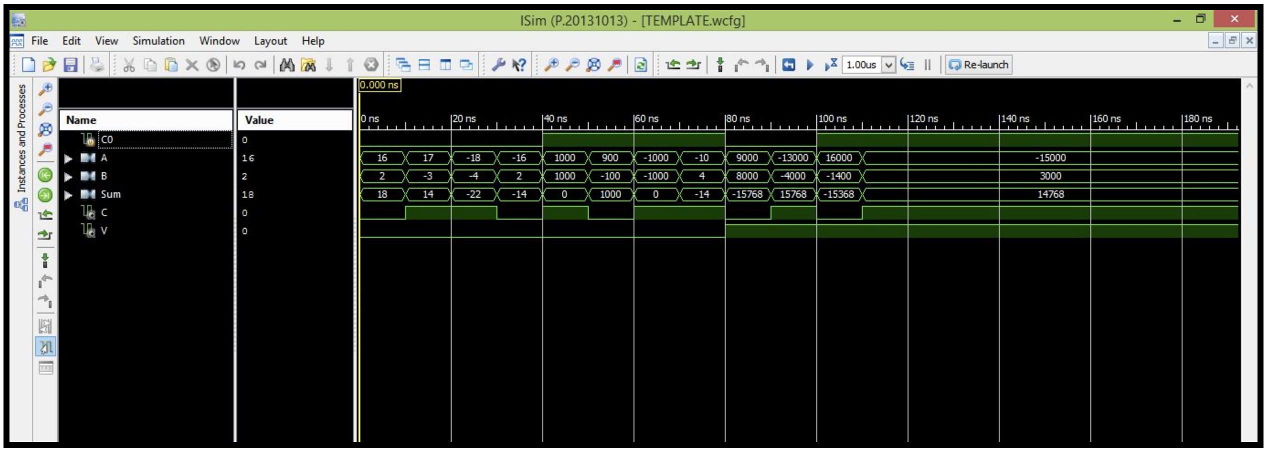
- Top module: 15 bit carry ripple adder-subtractor



- Sub module: Full adder FA

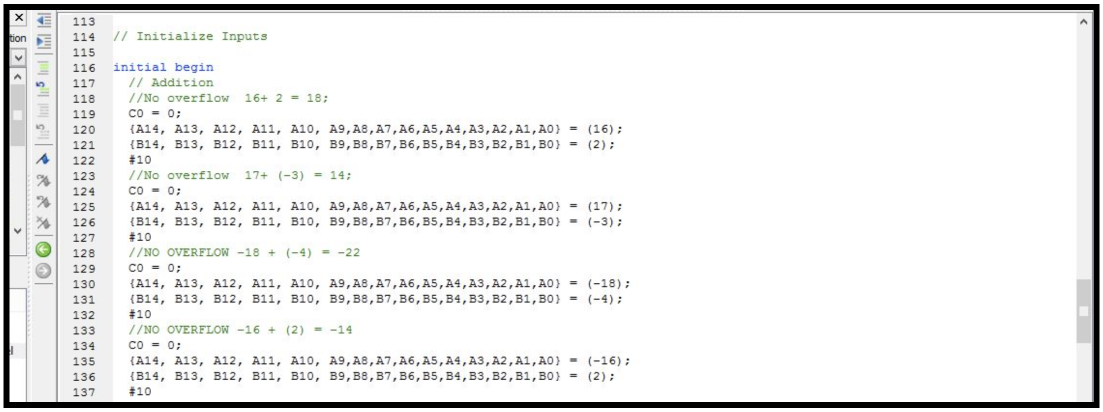


2.2. Simulation



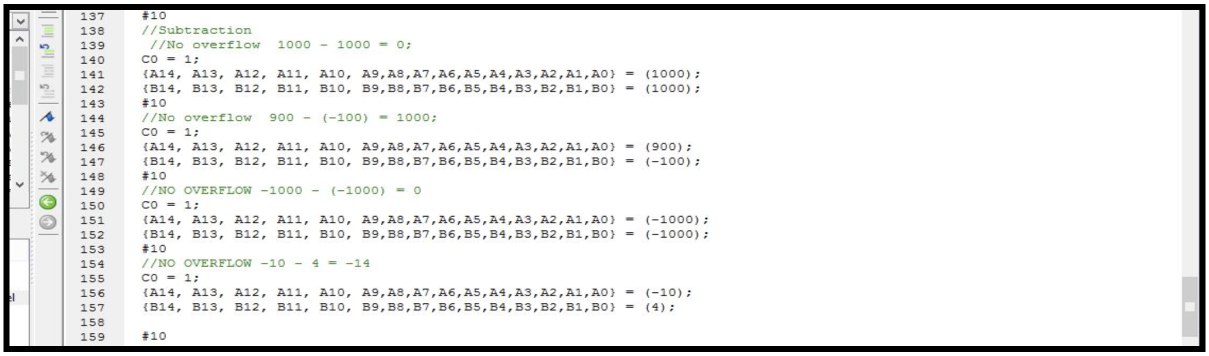
Tried different test cases with addition and subtraction, both with and without overflow:

- Addition with no overflow:



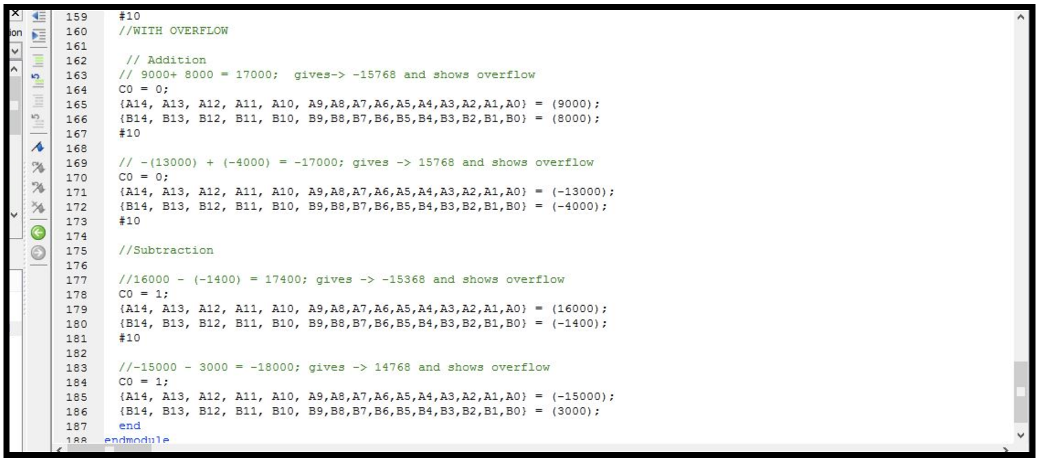
```
// Initialize Inputs

initial begin
  // Addition
  //No overflow  16+ 2 = 18;
  C0 = 0;
  {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (16);
  {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (2);
  #10
  //No overflow  17+ (-3) = 14;
  C0 = 0;
  {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (17);
  {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-3);
  #10
  //NO OVERFLOW -18 + (-4) = -22
  C0 = 0;
  {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-18);
  {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-4);
  #10
  //NO OVERFLOW -16 + (2) = -14
  C0 = 0;
  {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-16);
  {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (2);
  #10
```

- Subtraction with no overflow:



```
#10
//Subtraction
 //No overflow  1000 - 1000 = 0;
C0 = 1;
{A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (1000);
{B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (1000);
#10
//No overflow  900 - (-100) = 1000;
C0 = 1;
{A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (900);
{B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-100);
#10
//NO OVERFLOW -1000 - (-1000) = 0
C0 = 1;
{A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-1000);
{B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-1000);
#10
//NO OVERFLOW -10 - 4 = -14
C0 = 1;
{A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-10);
{B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (4);

#10
```

- Addition and subtraction with overflow:



```
#10
//WITH OVERFLOW

 // Addition
// 9000+ 8000 = 17000;  gives-> -15768 and shows overflow
C0 = 0;
{A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (9000);
{B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (8000);
#10

// -(13000) + (-4000) = -17000; gives -> 15768 and shows overflow
C0 = 0;
{A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-13000);
{B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-4000);
#10

//Subtraction

//16000 - (-1400) = 17400; gives -> -15368 and shows overflow
C0 = 1;
{A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (16000);
{B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-1400);
#10

//-15000 - 3000 = -18000; gives -> 14768 and shows overflow
C0 = 1;
{A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-15000);
{B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (3000);
end
endmodule
```

2.3. Implementation Results Present implementation results

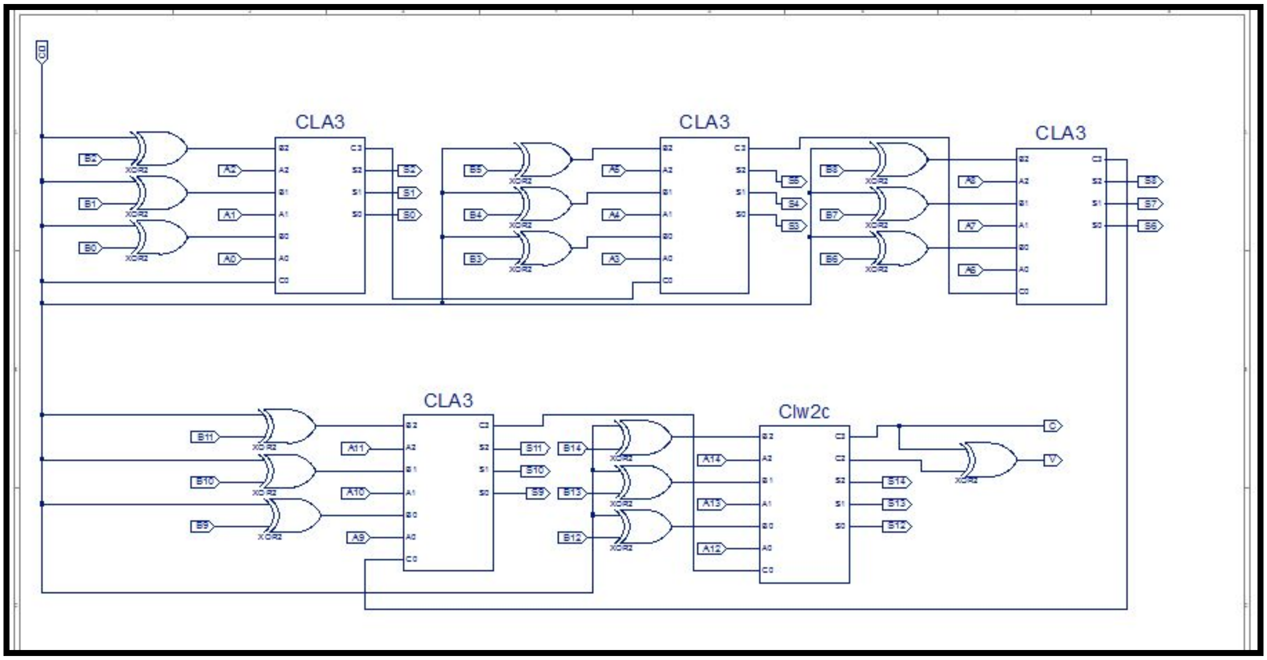
Area: # of LUTS 52 out of 1920

Time delay: 43.176ns

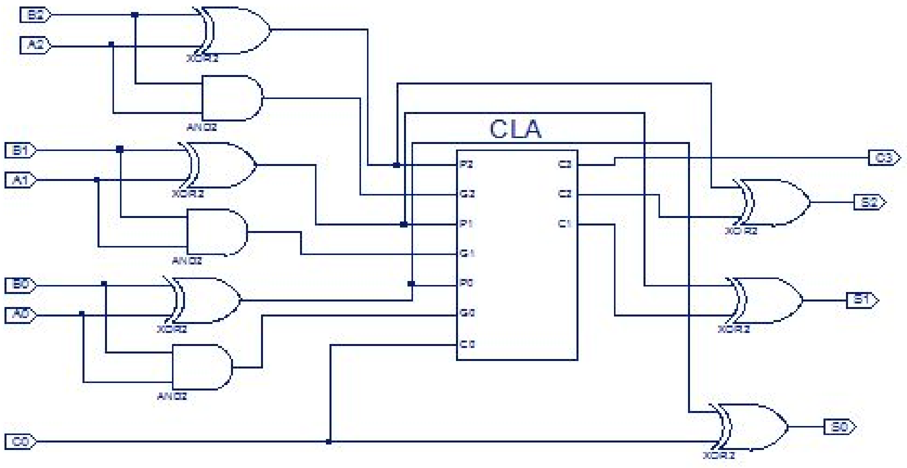
3. Second Design

3.1. Schematic

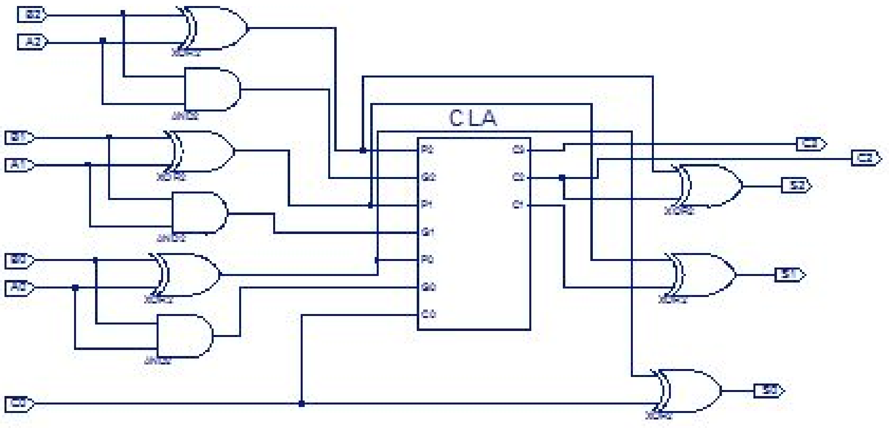
- Top module: 15-bit hybrid adder-subtractor using 5 3-bit(CLAs).



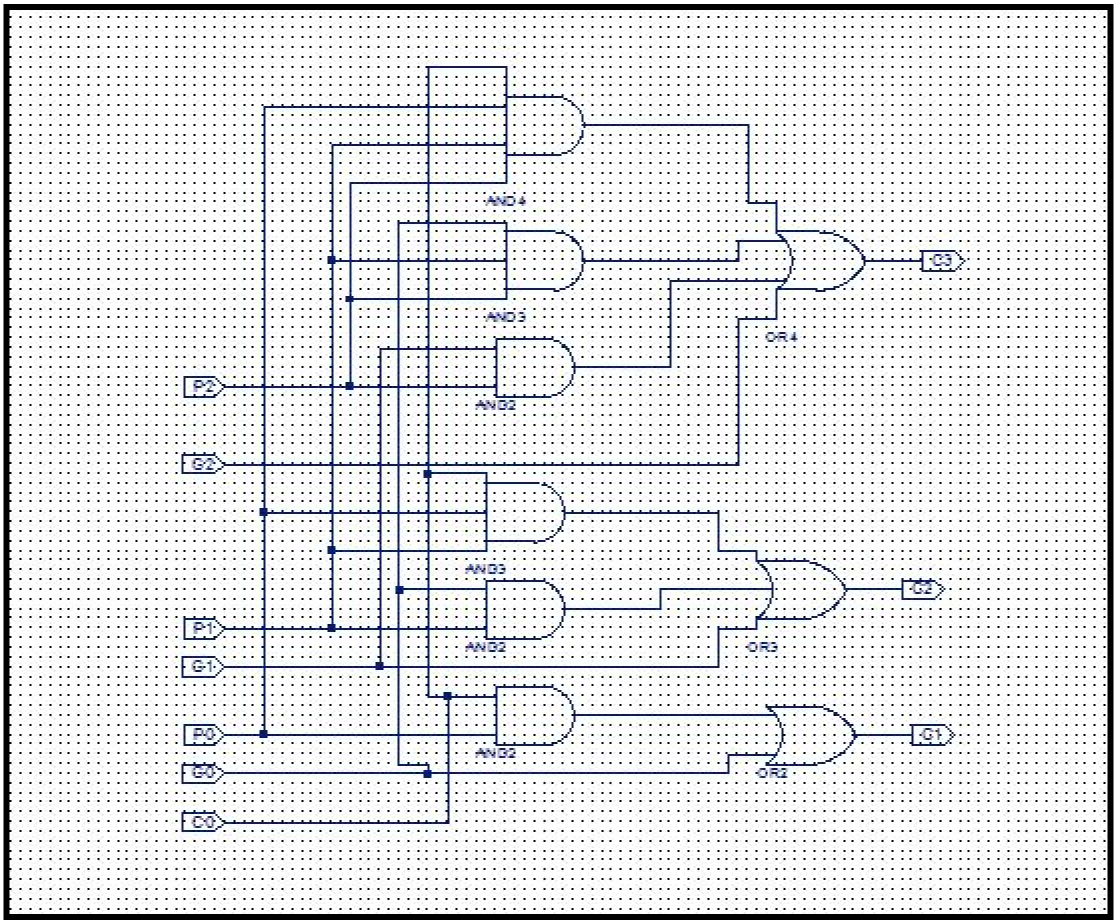
- Sub module: CLA3 (3 bit cla , with one carry out(c3))



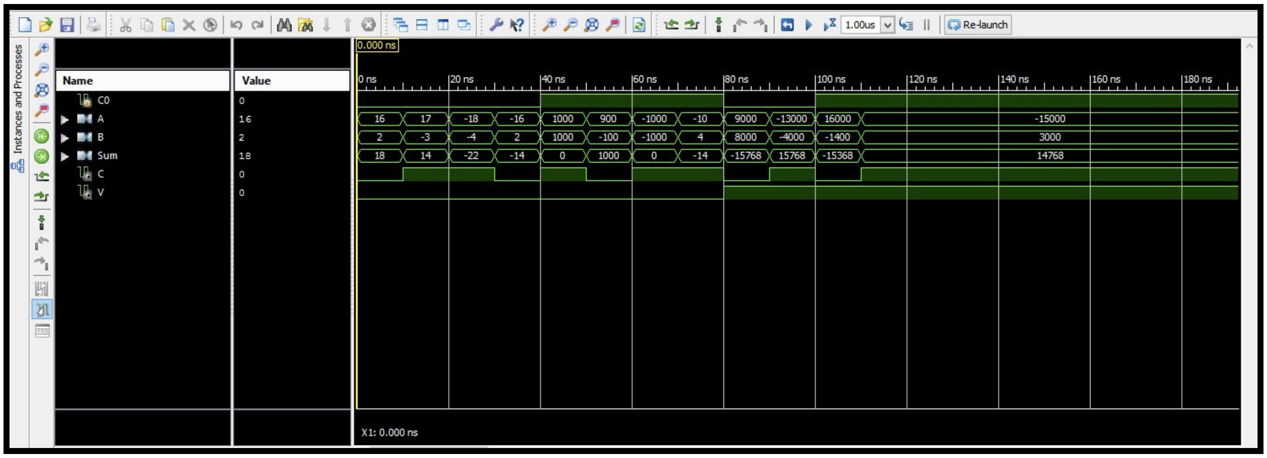
- Sub module: CLw2c (3 bit cla's, with 2 carry's out(c3,c2), needed 2 carry's to show overflow)



- Sub module: CLA(CL generator)



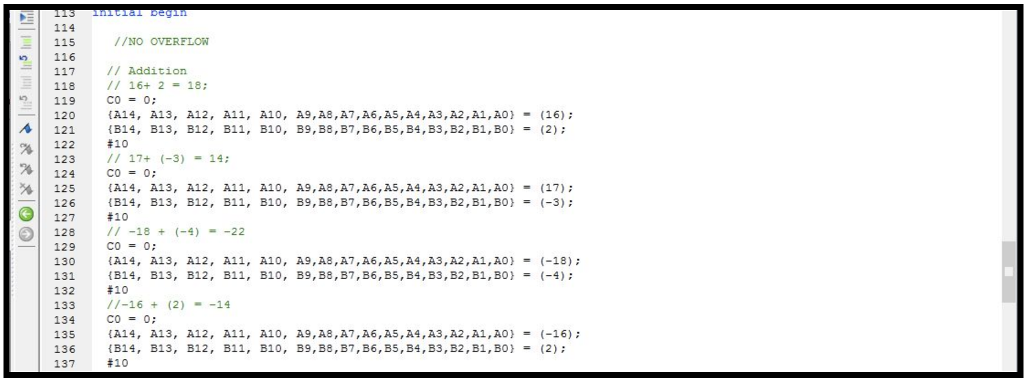
3.2. Simulation



Name	Value
C0	0
A	16
B	2
Sum	18
C	0
V	0

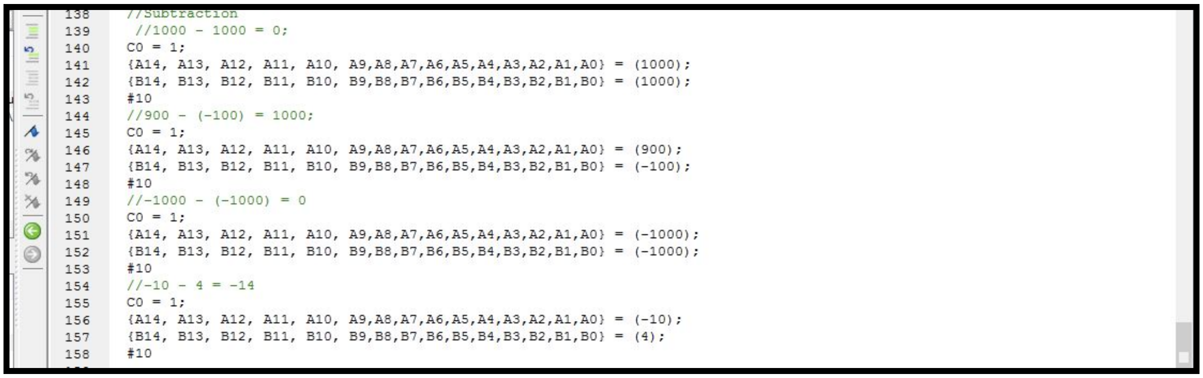
Tried different test cases with addition and subtraction, both with and without overflow:

- Addition with no overflow:



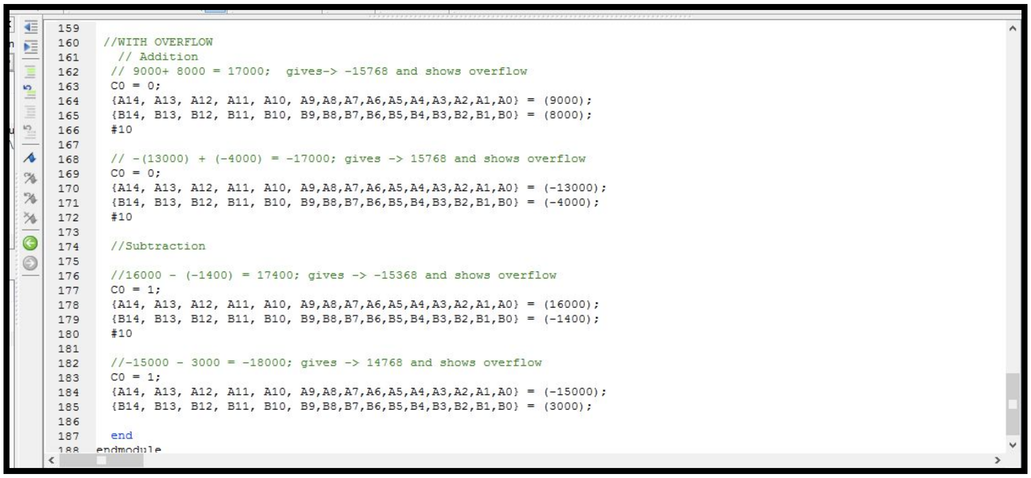
```
initial begin

   //NO OVERFLOW

  // Addition
  // 16+ 2 = 18;
  C0 = 0;
  {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (16);
  {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (2);
  #10
  // 17+ (-3) = 14;
  C0 = 0;
  {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (17);
  {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-3);
  #10
  // -18 + (-4) = -22
  C0 = 0;
  {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-18);
  {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-4);
  #10
  //-16 + (2) = -14
  C0 = 0;
  {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-16);
  {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (2);
  #10
```

- Subtraction with no overflow:



```
  //Subtraction
   //1000 - 1000 = 0;
  C0 = 1;
  {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (1000);
  {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (1000);
  #10
  //900 - (-100) = 1000;
  C0 = 1;
  {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (900);
  {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-100);
  #10
  //-1000 - (-1000) = 0
  C0 = 1;
  {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-1000);
  {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-1000);
  #10
  //-10 - 4 = -14
  C0 = 1;
  {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-10);
  {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (4);
  #10
```

- Addition and subtraction with Overflow:



```

 //WITH OVERFLOW
   // Addition
  // 9000+ 8000 = 17000;  gives-> -15768 and shows overflow
  C0 = 0;
  {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (9000);
  {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (8000);
  #10

  // -(13000) + (-4000) = -17000; gives -> 15768 and shows overflow
  C0 = 0;
  {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-13000);
  {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-4000);
  #10

  //Subtraction

  //16000 - (-1400) = 17400; gives -> -15368 and shows overflow
  C0 = 1;
  {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (16000);
  {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-1400);
  #10

  //-15000 - 3000 = -18000; gives -> 14768 and shows overflow
  C0 = 1;
  {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-15000);
  {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (3000);

  end
endmodule
```

-

3.3. Implementation Results Present implementation results (area and speed).
Area: # of LUTs 76 out of 1920
Time delay: 21.122 ns

4. Discussion

1) Ripple carry adder/subtractor is better in terms of area
2) Carry lookahead is better in terms of time

3) A new metric to measure the time-area tradeoff in two designs by multiplying the number of LUTs and time.
   For RC: 52 * 43.176ns = 2245.152
   For CL: 76 * 21.122ns =1605.272

   CLA is much better than Ripple carry according to the new metric.

4) Good design is the one with less area, shorter time (which leads to higher speed), and new metric should also be as small as possible.

Fall 2020