

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:

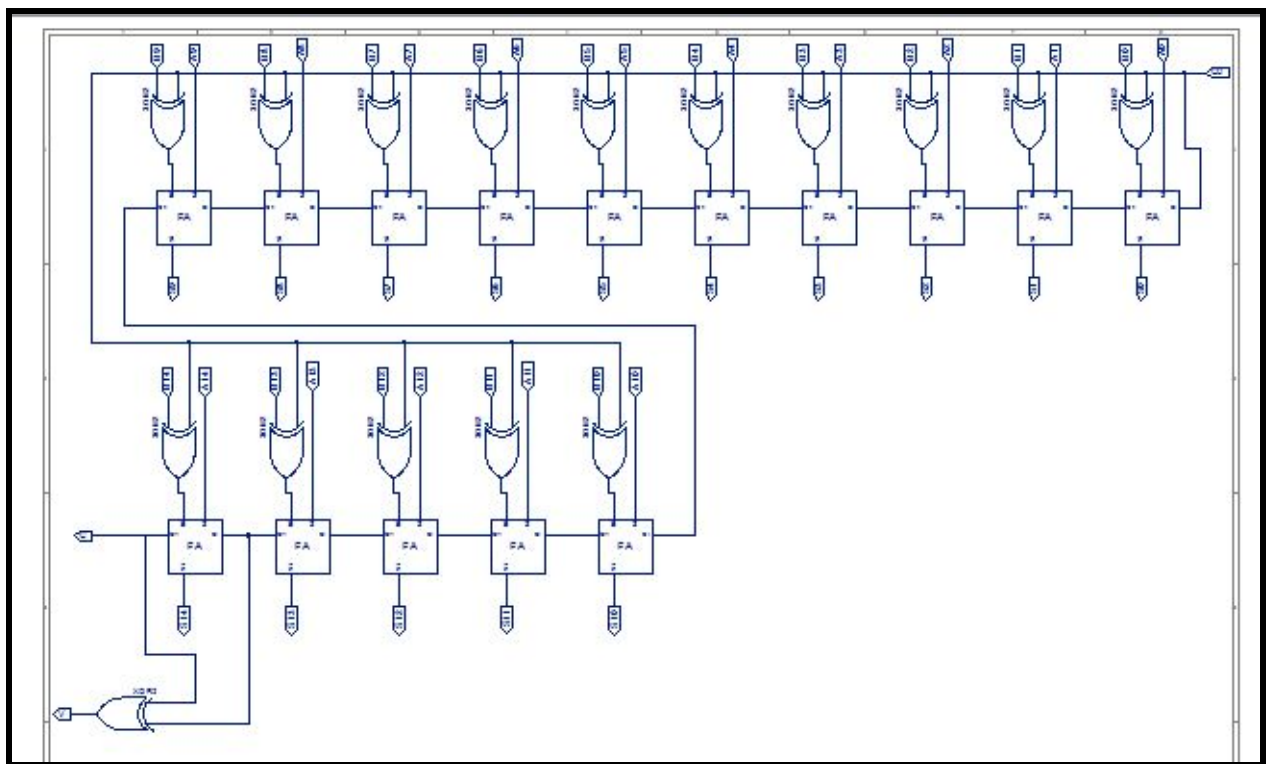
1. 15-bit ripple-carry adder-subtractor using full adders
2. 15-bit hybrid adder-subtractor using five 3-bit carry lookahead adders (CLAs).

Both adder-subtractors must detect overflow

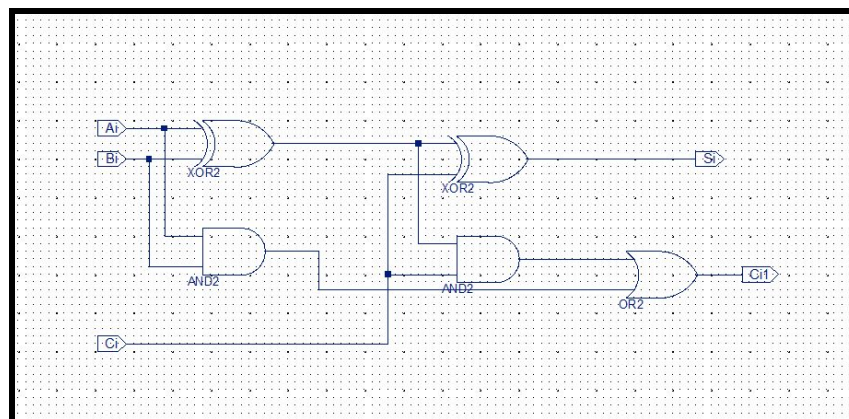
## 2. First Design

### 2.1. Schematic

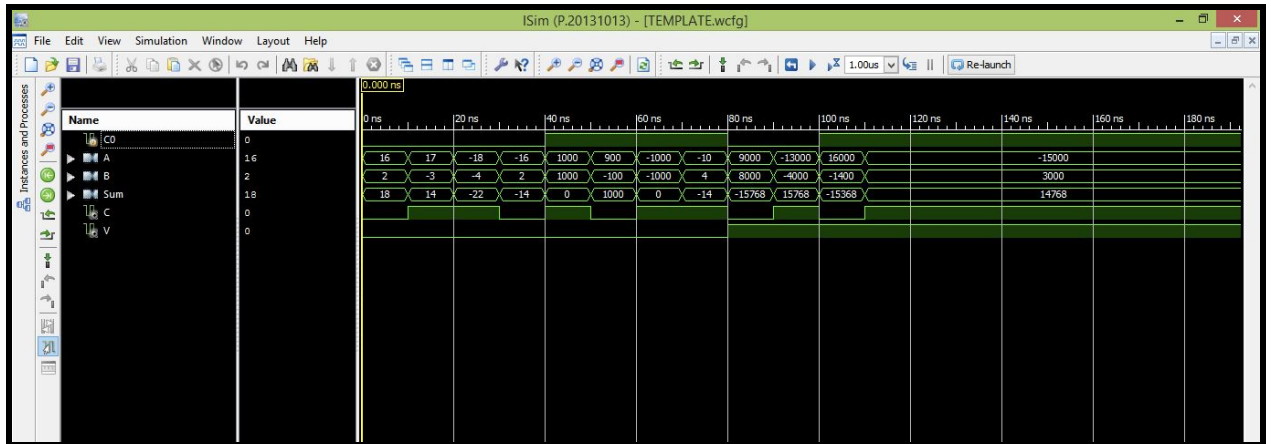
- Top module: 15 bit carry ripple adder-subtractor



- 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:

```

113 // Initialize Inputs
114
115
116 initial begin
117     // Addition
118     //No overflow 16+ 2 = 18;
119     CO = 0;
120     {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (16);
121     {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (2);
122     #10
123     //No overflow 17+ (-3) = 14;
124     CO = 0;
125     {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (17);
126     {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-3);
127     #10
128     //NO OVERFLOW -18 + (-4) = -22
129     CO = 0;
130     {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-18);
131     {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-4);
132     #10
133     //NO OVERFLOW -16 + (2) = -14
134     CO = 0;
135     {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-16);
136     {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (2);
137     #10

```

- Subtraction with no overflow:

```

137 #10
138 //Subtraction
139 //No overflow 1000 - 1000 = 0;
140 CO = 1;
141 {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (1000);
142 {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (1000);
143 #10
144 //No overflow 900 - (-100) = 1000;
145 CO = 1;
146 {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (900);
147 {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-100);
148 #10
149 //NO OVERFLOW -1000 - (-1000) = 0
150 CO = 1;
151 {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-1000);
152 {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-1000);
153 #10
154 //NO OVERFLOW -10 - 4 = -14
155 CO = 1;
156 {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-10);
157 {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (4);
158
159 #10

```

- Addition and subtraction with overflow:

```

159 #10
160 //WITH OVERFLOW
161
162 // Addition
163 // 9000+ 8000 = 17000; gives-> -15768 and shows overflow
164 CO = 0;
165 (A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0) = (9000);
166 (B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0) = (8000);
167 #10
168
169 // -(13000) + (-4000) = -17000; gives -> 15768 and shows overflow
170 CO = 0;
171 (A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0) = (-13000);
172 (B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0) = (-4000);
173 #10
174
175 //Subtraction
176 //16000 - (-1400) = 17400; gives -> -15368 and shows overflow
177 CO = 1;
178 (A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0) = (16000);
179 (B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0) = (-1400);
180 #10
181
182 // -15000 - 3000 = -18000; gives -> 14768 and shows overflow
183 CO = 1;
184 (A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0) = (-15000);
185 (B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0) = (3000);
186 end
187 endmodule

```

## 2.3. Implementation Results Present implementation results

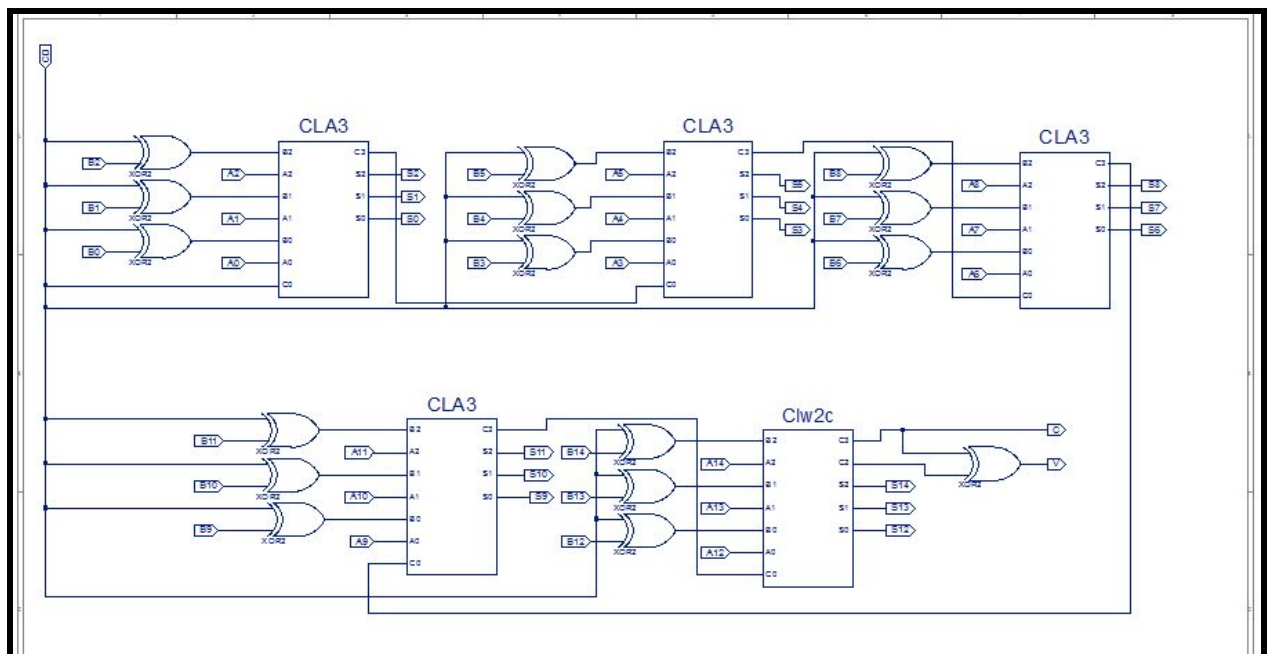
Area: # of LUTS 52 out of 1920

Time delay: 43.176ns

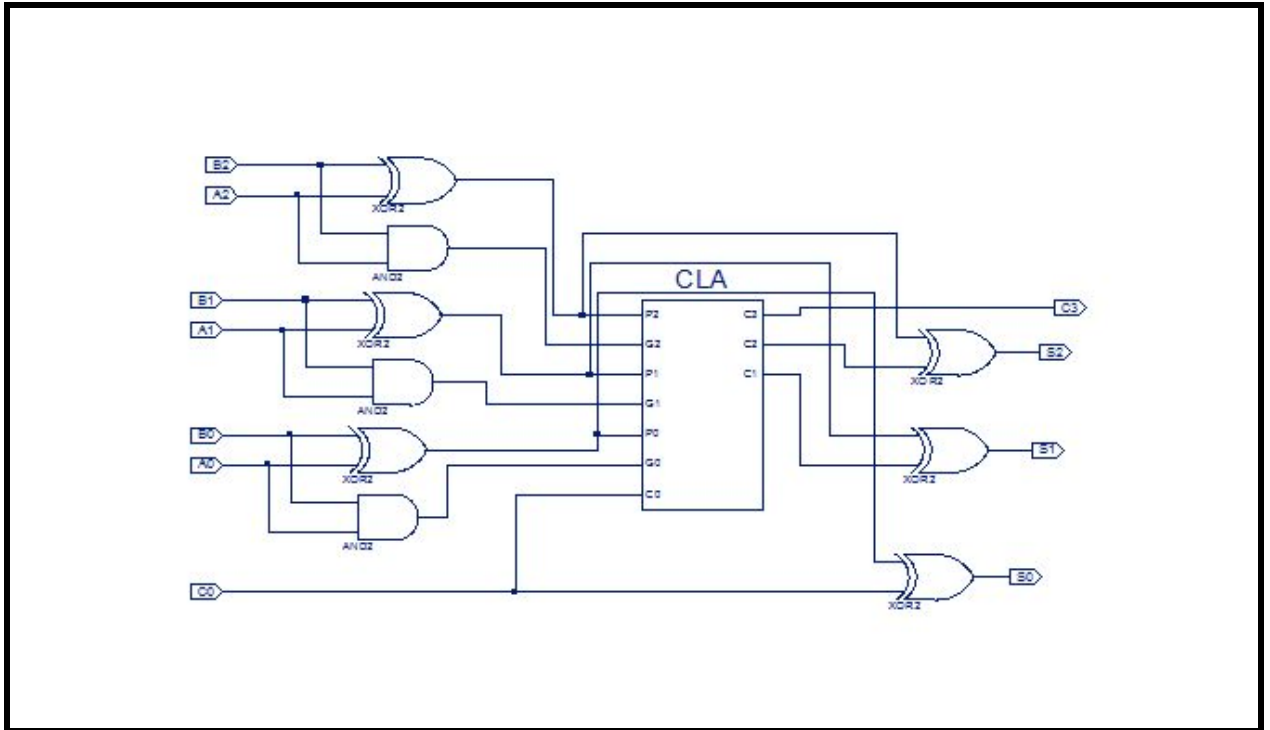
## 3. Second Design

### 3.1. Schematic

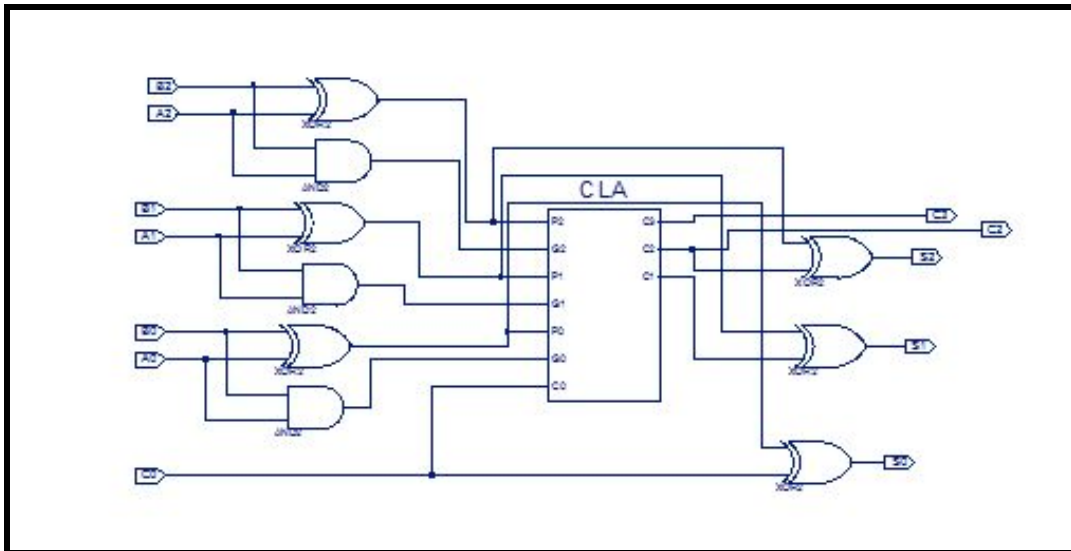
- Top module: 15-bit hybrid adder-subtractor using 5 3-bit(CLAs).



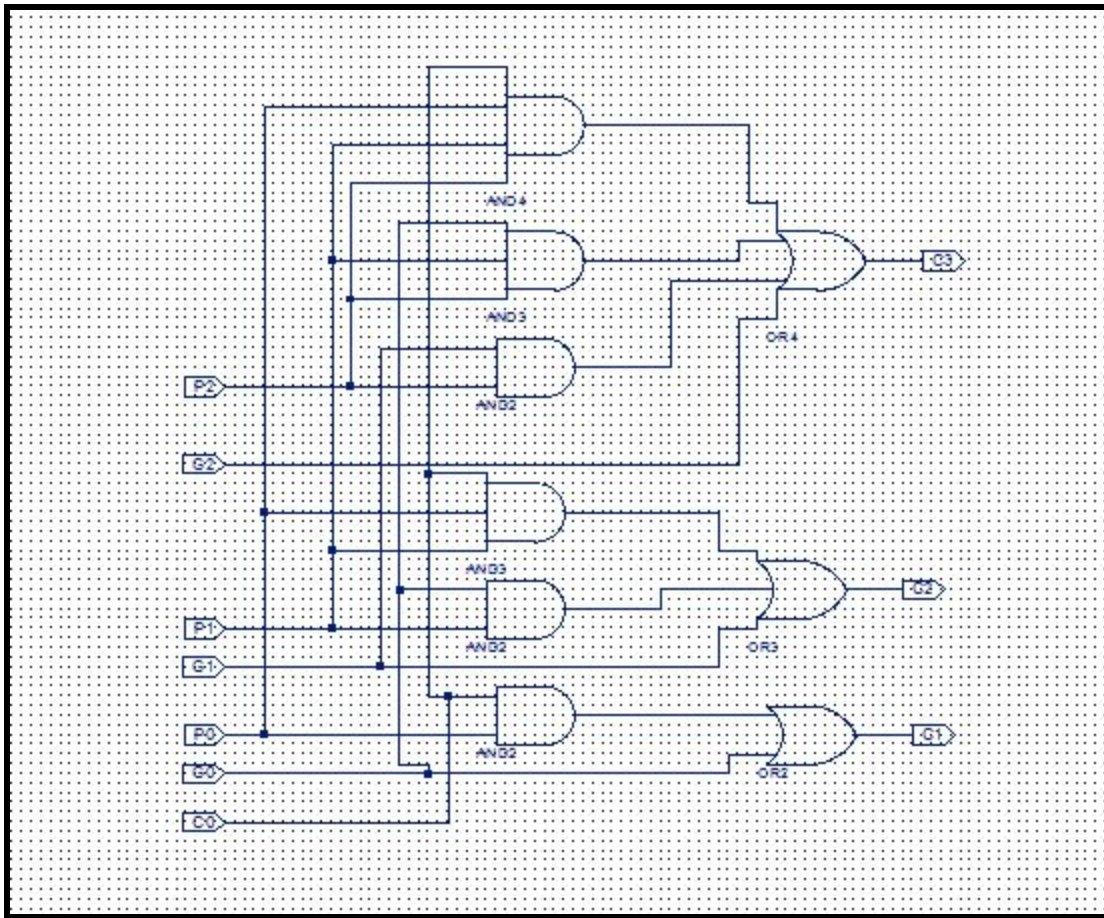
- Sub module: CLA3 (3 bit cla , with one carry out(c3))



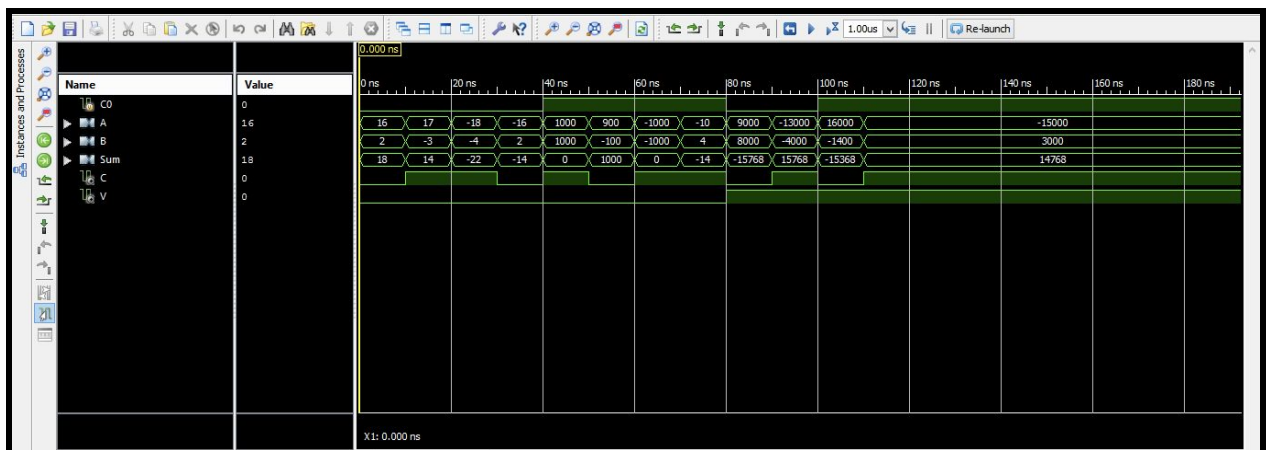
- Sub module: CLw2c (3 bit cla's, with 2 carry's out(c3,c2), needed 2 carry's to show overflow )



- Sub module: CLA( CL generator)



### 3.2. Simulation



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



- Addition with no overflow:

```

113 Initial Begin
114
115 //NO OVERFLOW
116
117 // Addition
118 // 16+ 2 = 18;
119 CO = 0;
120 {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (16);
121 {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (2);
122 #10
123 // 17+ (-3) = 14;
124 CO = 0;
125 {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (17);
126 {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-3);
127 #10
128 // -18 + (-4) = -22
129 CO = 0;
130 {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-18);
131 {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-4);
132 #10
133 // -16 + (2) = -14
134 CO = 0;
135 {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-16);
136 {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (2);
137 #10

```

- Subtraction with no overflow:

```

138 //Subtraction
139 //1000 - 1000 = 0;
140 CO = 1;
141 {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (1000);
142 {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (1000);
143 #10
144 //900 - (-100) = 1000;
145 CO = 1;
146 {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (900);
147 {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-100);
148 #10
149 //-1000 - (-1000) = 0
150 CO = 1;
151 {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-1000);
152 {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-1000);
153 #10
154 //-10 - 4 = -14
155 CO = 1;
156 {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-10);
157 {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (4);
158 #10

```

- Addition and subtraction with Overflow:

```

159
160 //WITH OVERFLOW
161 // Addition
162 // 9000+ 8000 = 17000; gives-> -15768 and shows overflow
163 CO = 0;
164 {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (9000);
165 {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (8000);
166 #10
167
168 // -(13000) + (-4000) = -17000; gives -> 15768 and shows overflow
169 CO = 0;
170 {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-13000);
171 {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-4000);
172 #10
173
174 //Subtraction
175 //16000 - (-1400) = 17400; gives -> -15368 and shows overflow
176 CO = 1;
177 {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (16000);
178 {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (-1400);
179 #10
180
181 // -15000 - 3000 = -18000; gives -> 14768 and shows overflow
182 CO = 1;
183 {A14, A13, A12, A11, A10, A9,A8,A7,A6,A5,A4,A3,A2,A1,A0} = (-15000);
184 {B14, B13, B12, B11, B10, B9,B8,B7,B6,B5,B4,B3,B2,B1,B0} = (3000);
185 #10
186
187 end
188 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:  $52 * 43.176\text{ns} = 2245.152$

For CL:  $76 * 21.122\text{ns} = 1605.272$

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