#2 Project

2016116783 김성록

COMP311-1: Logic Circuit Design

Fall 2019, Prof. Taigon Song

Project 2. Due: Dec. 2, 8:59am [Total: 90 + 10 points]

[Your Student ID] Your Name

completion : 9/9, 100%

Checksheet item	No.	Done? [Y/N]
RCA adder design (no delay, 10 points)	1	
CLA adder design (no delay, 20 points)	2	
CSA adder design 1 (no delay, 15 points)	3	
CSA adder design 2 (no delay, 5 points)	4	
Four adder design with delay (10 points)	5	
Tradeoff between area and speed (Synthesis report, 10 points)	6	
Delay result/analysis (10 points)	7	
Any other discussion (10 points)	8	
Bonus: Submission date (+ ____ points)	9	

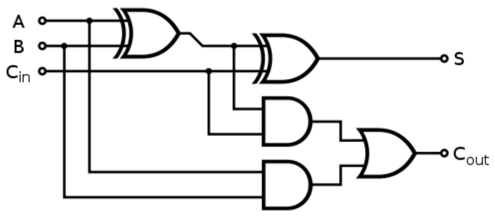
1bit, 4bit and 16bit RCA

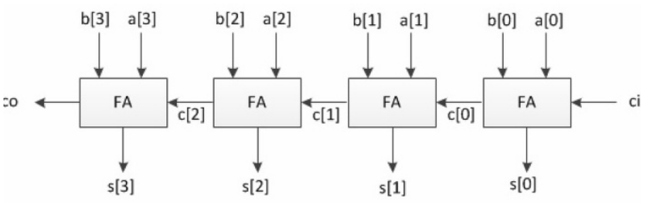
Basic form of Adder.

When 1 bit RCA calcuate SUM and C_out from A,B,C_in, 4bit RCA link to each 1bit RCA sequently.

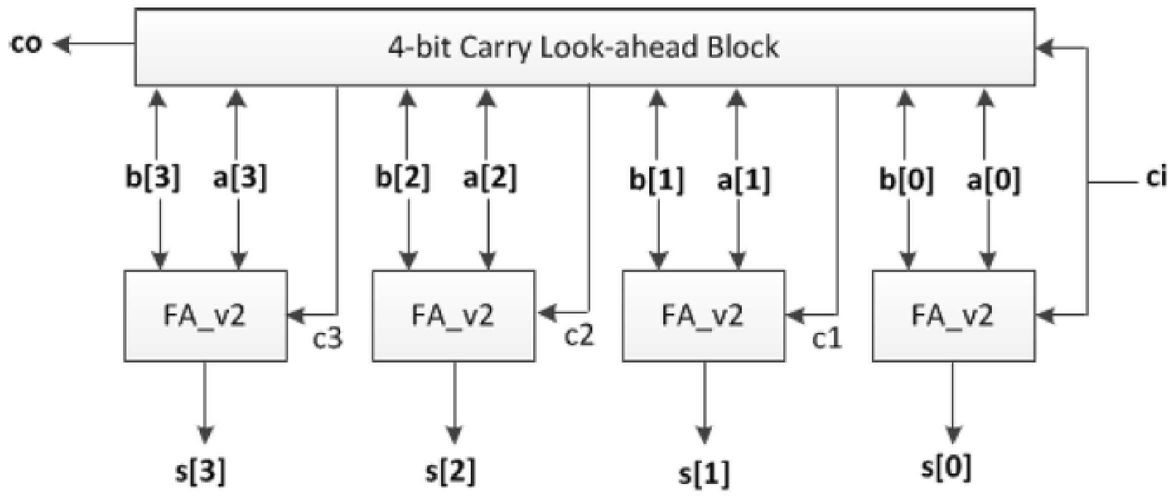
Then 16bit RCA link to 4 bit each 4bit RCA sequently.

Be notice that word **link** means one of the RCA module's C_out is going to be c_in for the next one.





```
`timescale 1ns/1ns
module rca1b(
  input a,b,
  input c_in,
  output out,
  output c_out);

  wire s1, c1, s2;

  // Calculate sum of a,b and c_in for 2^0
  xor #6 xor1(s1,a,b);
  xor #6 xor2(out,s1,c_in);

  // Calculate c_out for 2^1
  and #7 and1(c1,s1,c_in);
  and #7 and2(s2,a,b);
  or #7 or1(c_out,c1,s2);

endmodule

module rca4b(
  input [3:0] a,b,
  input c_in,
  output [3:0] out,
  output c_out);

  wire c1,c2,c3;

  // attach 4 rca1b sequently
  rca1b u1(a[0],b[0],c_in,out[0],c1);
  rca1b u2(a[1],b[1],c1,out[1],c2);
  rca1b u3(a[2],b[2],c2,out[2],c3);
  rca1b u4(a[3],b[3],c3,out[3],c_out);

endmodule

module rca16b(
  input [15:0] a1,b1,
  input cin,
  output [15:0] sum,
  output cout);

  wire c1,c2,c3;

  // attach 4 rca4b sequently
  rca4b k1(a1[3:0],b1[3:0],cin,sum[3:0],c1);
  rca4b k2(a1[7:4],b1[7:4],c1,sum[7:4],c2);
  rca4b k3(a1[11:8],b1[11:8],c2,sum[11:8],c3);
  rca4b k4(a1[15:12],b1[15:12],c3,sum[15:12],cout);

endmodule
```

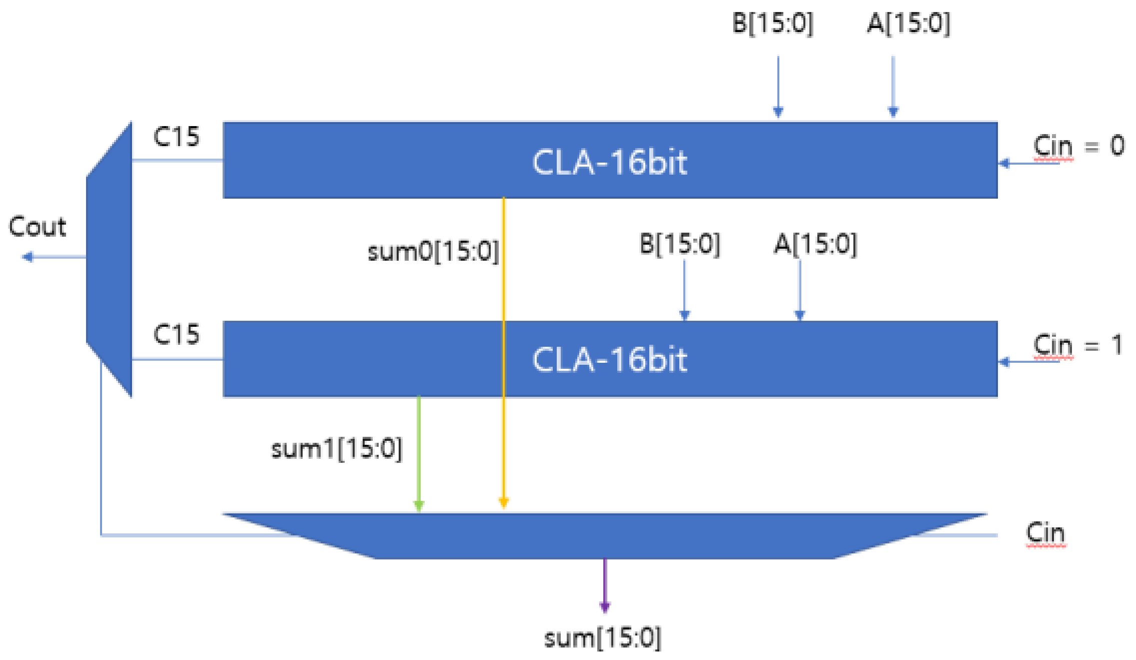


4bit CLA

```
`timescale 1ns/1ns
module cla4b(
  input [3:0] a,b,
  input c_in,
  output [3:0] out,
  output c_out);

  wire [3:0] g, p;

  //Generation Signal
  and #7 and1(g[0],a[0],b[0]);
  and #7 and2(g[1],a[1],b[1]);
  and #7 and3(g[2],a[2],b[2]);
  and #7 and4(g[3],a[3],b[3]);

  //Propagation Signal
  xor #7 or1(p[0],a[0],b[0]);
  xor #7 or2(p[1],a[1],b[1]);
  xor #7 or3(p[2],a[2],b[2]);
  xor #7 or4(p[3],a[3],b[3]);


  //Prepare for attaining c_out
  wire c_proc0, c_proc1, c_proc2,
     c_proc3, c_proc4, c_proc5,
     c_proc6, c_proc7, c_proc8, c_proc9;
  wire c0,c1,c2,c3;

  //get c_out
  and #14(c0,c_in,1);

  and #7 and5(c_proc0,p[0],c_in);
  or #7 or5(c1,g[0],c_proc0);

  and #7 and6(c_proc1,p[1],p[0],c_in);
  and #7 and7(c_proc2,p[1],g[0]);
  or #7 or6(c2,g[1],c_proc1,c_proc2);

  and #7 and8(c_proc3,p[2],p[1],p[0],c_in);
  and #7 and9(c_proc4,p[2],p[1],g[0]);
  and #7 and10(c_proc5,p[2],g[1]);
  or #7 or7(c3,g[2],c_proc3,c_proc4,c_proc5);

  and #7 and11(c_proc6,p[3],p[2],p[1],p[0],c_in);
  and #7 and12(c_proc7,p[3],p[2],p[1],g[0]);
  and #7 and13(c_proc8,p[3],p[2],g[1]);
  and #7 and14(c_proc9,p[3],g[2]);
  or #7 or8(c_out,g[3],c_proc6,c_proc7,c_proc8,c_proc9);

  //Calculate output
  xor #6 xor1(out[0], p[0], c0);
  xor #6 xor2(out[1], p[1], c1);
  xor #6 xor3(out[2], p[2], c2);
  xor #6 xor4(out[3], p[3], c3);

endmodule
```

16bit RCA

```
module cla16b(
  input [15:0] a1,b1,
  input cin,
  output [15:0] sum,
  output cout);
  wire c1,c2,c3;

  // attach 4 cla4b like rca16b
  cla4b u1(a1[3:0],b1[3:0],cin,sum[3:0],c1);
  cla4b u2(a1[7:4],b1[7:4],c1,sum[7:4],c2);
  cla4b u3(a1[11:8],b1[11:8],c2,sum[11:8],c3);
  cla4b u4(a1[15:12],b1[15:12],c3,sum[15:12],cout);

endmodule
```



2bit and 16bit MUX 2to1

```
`timescale 1ns/1ns
module mux1b_2to1(
  input a,b,
  input sel,
  output out);

  // if sel==0 : out = a, else out = b;
  assign #3 out = sel? b : a;

endmodule

module mux16b_2to1(
  input [15:0] a,b,
  input sel,
  output [15:0] out);

  // if sel==0 : out = a, else out = b
  assign #3 out = sel? b : a;

endmodule
```

16bit CSA-1

```
module csa16b1(
  input [15:0] a1,b1,
  input cin,
  output [15:0] sum,
  output cout);

  wire cout0,cout1;
  wire [15:0] sum0, sum1;

  // calculate sum for each c_in
  cla16b u0(a1,b1,cin,sum0,cout0);
  cla16b u1(a1,b1,cin,sum1,cout1);

  // select sum and c_out by c_in using mux
  mux16b_2to1 mux0(sum0,sum1,cin,sum);
  mux1b_2to1 mux1(cout0,cout1,cin,cout);

endmodule
```

16bit CSA-2

```
module csa16b2(
  input [15:0] a1,b1,
  input cin,
  output [15:0] sum,
  output cout);

  wire cout0,cout1;
  wire [15:0] sum0, sum1;

  // calculate sum for each c_in
  rca16b u0(a1,b1,cin,sum0,cout0);
  rca16b u1(a1,b1,cin,sum1,cout1);

  // select sum and c_out by c_in using mux
  mux16b_2to1 mux0(sum0,sum1,cin,sum);
  mux1b_2to1 mux1(cout0,cout1,cin,cout);

endmodule
```

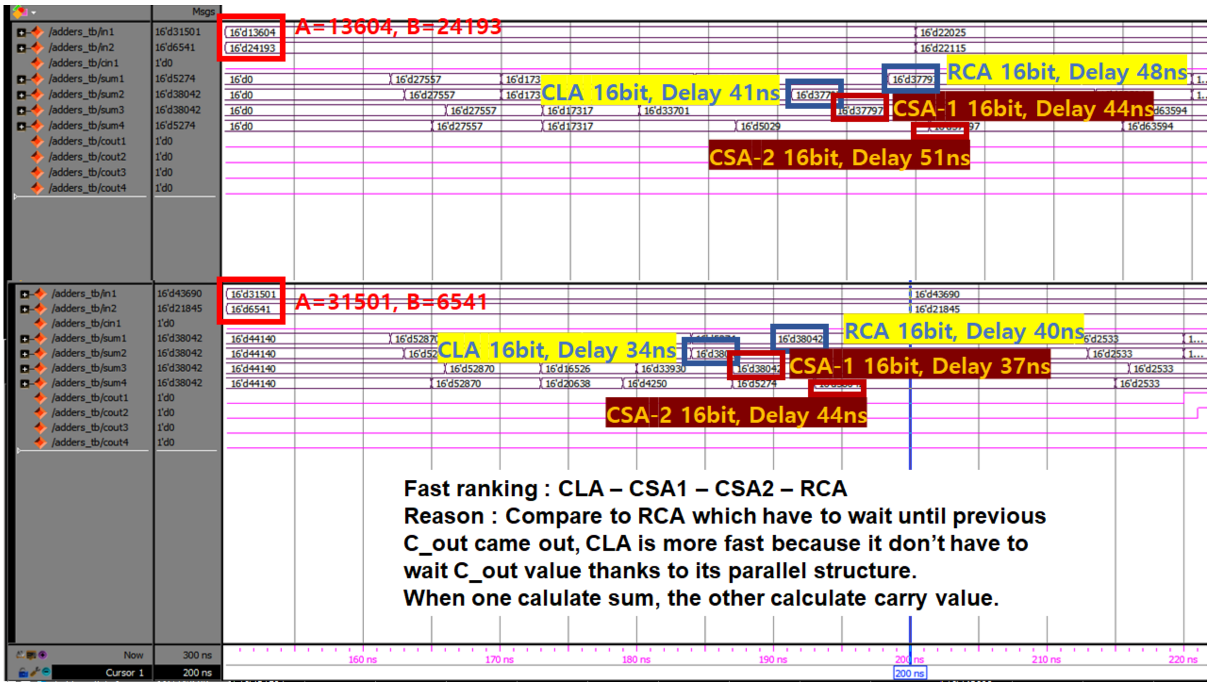
Tradeoff between area and speed

The number of gates

- RCA 16b : 80
- CLA 16b : 108
- CSA-1 16b : 216 + 2 Mux
- CSA-2 16b : 160 + 2 Mux

CLA 16 bit occupy the area as 108 gate but more fast than RCA. So Speed and area are inversely proportional, but not always because CSA is inefficient for speed.

Delay Result / Analysis



Fast ranking : CLA – CSA1 – CSA2 – RCA
Reason : Compare to RCA which have to wait until previous C_out came out, CLA is more fast because it don't have to wait C_out value thanks to its parallel structure.
When one calulate sum, the other calculate carry value.

Discussion

CLA is Fastest above all adders, but it can be more faster through designing 16bit CLA by primitive gates. The CLA I create is made by 4 bit CLA, linked with each other sequently. Otherwise, CSA-1 is slower than CLA because CSA-1 perform CLA process twice but faster than RCA since it use CLA. Likewise CSA-2 is slower than RCA.

The things I can't understand is why delay is different changed by input. Thre reason I think is that the bigger input, the slower output came. Just Think that when 0 inserted as input, its calculation speed will be faster on and gates since it didn't perform any other logical calculation. As you see on the screenshot, because B of the bottom is much smaller than the top, The case of bottom is much faster and less delay value.