SoC설계

Lab#8

Kogge Stone Adder

컴퓨터공학과

201402439

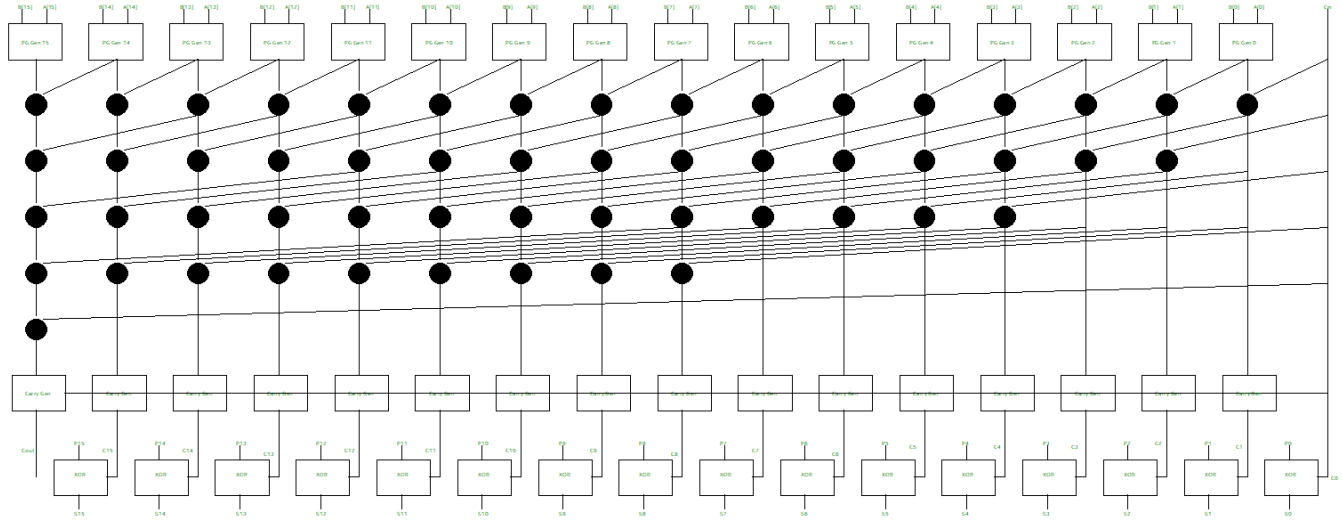
천원준

1. Purpose of the lab

이번 과제의 목표는 Kogge Stone Adder를 구현하고 제대로 동작하는지 확인하는 것 입니다.

2. Design Procedure

- Block diagram



3. Simulation

- PG Generator, Black Cell, Carry Generator

```
//Input module : Propagation, Generation

module PG_Gen(A, B, P, G);
   input A, B;
   output P, G;

   assign P = A ^ B; //propagation
   assign G = A & B; //generation

endmodule

module black_cell(Pin1, Pin2, Gin1, Gin2, P, G);
   input Pin1, Pin2, Gin1, Gin2;
   output P, G;

   assign P = Pin1 & Pin2;
   assign G = Gin2 | (Pin2 & Gin1);

endmodule

module Carry_Gen(P, G, Cin, Cout);
   input P, G, Cin;
   output Cout;

   assign Cout = G | (P & Cin);

endmodule
```

PG Generator는 두개의 입력을 받아서 입력에 대한 Propagation과 Generation을 구합니다. Propagation은 두 개의 입력 중 하나만 1일 경우에 발생합니다. 따라서, 두 입력을 XOR 연산해주면 Propagation을 구할 수 있습니다. Generation은 캐리가 발생했음을 알려주는 역할을 하는데, 두 입력이 전부 1일 경우에 캐리가 발생하므로 AND 연산을 해주어 Generation을 구할 수 있습니다.

Black Cell은 현재 자리와 이전 자리의 Propagation과 Generation들을 이용하여 Propagation과 Generation을 구하는 모듈입니다. Propagation은 Pin1과 Pin2 둘 다 1일 때 캐리의 전파가 일어나므로, AND 연산을 취해줍니다. Generation은 Gin2가 1이거나, Pin2와 Gin1이 둘 다 1일 경우에 캐리가 발생하므로, Gin2 | (Pin2 & Gin1) 연산을 취해줍니다.

Carry Generator은 Black Cell 연산들을 거친 P와 G, 그리고 Cin을 통해 최종적인 캐리를 연산하는 모듈입니다. 최종적인 캐리는 최종 Generation이 1이거나, 최종 Propagation이 1이면서 Cin이 1인 경우에 발생하므로, G | (P & Cin) 연산을 해줍니다.

- myKSA

```
module myKSA(A, B, Cin, Sum, Cout);
   //input
   input [15:0] A, B;
   input Cin;

   //output
   output [15:0] Sum;
   output Cout;

   wire [15:0] P_st_1, G_st_1, P_A, G_A, P_B, G_B, P_C, G_C, P_D, G_D, P_E, G_E;
   wire [16:1] Carry;


   //PG Generator
   PG_Gen PG_Gen0(.A(A[0]), .B(B[0]), .P(P_st_1[0]), .G(G_st_1[0]));
   PG_Gen PG_Gen1(.A(A[1]), .B(B[1]), .P(P_st_1[1]), .G(G_st_1[1]));
   PG_Gen PG_Gen2(.A(A[2]), .B(B[2]), .P(P_st_1[2]), .G(G_st_1[2]));
   PG_Gen PG_Gen3(.A(A[3]), .B(B[3]), .P(P_st_1[3]), .G(G_st_1[3]));
   PG_Gen PG_Gen4(.A(A[4]), .B(B[4]), .P(P_st_1[4]), .G(G_st_1[4]));
   PG_Gen PG_Gen5(.A(A[5]), .B(B[5]), .P(P_st_1[5]), .G(G_st_1[5]));
   PG_Gen PG_Gen6(.A(A[6]), .B(B[6]), .P(P_st_1[6]), .G(G_st_1[6]));
   PG_Gen PG_Gen7(.A(A[7]), .B(B[7]), .P(P_st_1[7]), .G(G_st_1[7]));
   PG_Gen PG_Gen8(.A(A[8]), .B(B[8]), .P(P_st_1[8]), .G(G_st_1[8]));
   PG_Gen PG_Gen9(.A(A[9]), .B(B[9]), .P(P_st_1[9]), .G(G_st_1[9]));
   PG_Gen PG_Gen10(.A(A[10]), .B(B[10]), .P(P_st_1[10]), .G(G_st_1[10]));
   PG_Gen PG_Gen11(.A(A[11]), .B(B[11]), .P(P_st_1[11]), .G(G_st_1[11]));
   PG_Gen PG_Gen12(.A(A[12]), .B(B[12]), .P(P_st_1[12]), .G(G_st_1[12]));
   PG_Gen PG_Gen13(.A(A[13]), .B(B[13]), .P(P_st_1[13]), .G(G_st_1[13]));
   PG_Gen PG_Gen14(.A(A[14]), .B(B[14]), .P(P_st_1[14]), .G(G_st_1[14]));
   PG_Gen PG_Gen15(.A(A[15]), .B(B[15]), .P(P_st_1[15]), .G(G_st_1[15]));
```

```
   //Black Cell A
   black_cell BC_A0(.Pin2(P_st_1[0]), .Gin2(G_st_1[0]), .Pin1(1'b0), .Gin1(Cin), .P(P_A[0]), .G(G_A[0]));
   black_cell BC_A1(.Pin2(P_st_1[1]), .Gin2(G_st_1[1]), .Pin1(P_st_1[0]), .Gin1(G_st_1[0]), .P(P_A[1]), .G(G_A[1]));
   black_cell BC_A2(.Pin2(P_st_1[2]), .Gin2(G_st_1[2]), .Pin1(P_st_1[1]), .Gin1(G_st_1[1]), .P(P_A[2]), .G(G_A[2]));
   black_cell BC_A3(.Pin2(P_st_1[3]), .Gin2(G_st_1[3]), .Pin1(P_st_1[2]), .Gin1(G_st_1[2]), .P(P_A[3]), .G(G_A[3]));
   black_cell BC_A4(.Pin2(P_st_1[4]), .Gin2(G_st_1[4]), .Pin1(P_st_1[3]), .Gin1(G_st_1[3]), .P(P_A[4]), .G(G_A[4]));
   black_cell BC_A5(.Pin2(P_st_1[5]), .Gin2(G_st_1[5]), .Pin1(P_st_1[4]), .Gin1(G_st_1[4]), .P(P_A[5]), .G(G_A[5]));
   black_cell BC_A6(.Pin2(P_st_1[6]), .Gin2(G_st_1[6]), .Pin1(P_st_1[5]), .Gin1(G_st_1[5]), .P(P_A[6]), .G(G_A[6]));
   black_cell BC_A7(.Pin2(P_st_1[7]), .Gin2(G_st_1[7]), .Pin1(P_st_1[6]), .Gin1(G_st_1[6]), .P(P_A[7]), .G(G_A[7]));
   black_cell BC_A8(.Pin2(P_st_1[8]), .Gin2(G_st_1[8]), .Pin1(P_st_1[7]), .Gin1(G_st_1[7]), .P(P_A[8]), .G(G_A[8]));
   black_cell BC_A9(.Pin2(P_st_1[9]), .Gin2(G_st_1[9]), .Pin1(P_st_1[8]), .Gin1(G_st_1[8]), .P(P_A[9]), .G(G_A[9]));
   black_cell BC_A10(.Pin2(P_st_1[10]), .Gin2(G_st_1[10]), .Pin1(P_st_1[9]), .Gin1(G_st_1[9]), .P(P_A[10]), .G(G_A[10]));
   black_cell BC_A11(.Pin2(P_st_1[11]), .Gin2(G_st_1[11]), .Pin1(P_st_1[10]), .Gin1(G_st_1[10]), .P(P_A[11]), .G(G_A[11]));
   black_cell BC_A12(.Pin2(P_st_1[12]), .Gin2(G_st_1[12]), .Pin1(P_st_1[11]), .Gin1(G_st_1[11]), .P(P_A[12]), .G(G_A[12]));
   black_cell BC_A13(.Pin2(P_st_1[13]), .Gin2(G_st_1[13]), .Pin1(P_st_1[12]), .Gin1(G_st_1[12]), .P(P_A[13]), .G(G_A[13]));
   black_cell BC_A14(.Pin2(P_st_1[14]), .Gin2(G_st_1[14]), .Pin1(P_st_1[13]), .Gin1(G_st_1[13]), .P(P_A[14]), .G(G_A[14]));
   black_cell BC_A15(.Pin2(P_st_1[15]), .Gin2(G_st_1[15]), .Pin1(P_st_1[14]), .Gin1(G_st_1[14]), .P(P_A[15]), .G(G_A[15]));

   //Black Cell B
   black_cell BC_B1(.Pin2(P_A[1]), .Gin2(G_A[1]), .Pin1(1'b0), .Gin1(Cin), .P(P_B[1]), .G(G_B[1]));
   black_cell BC_B2(.Pin2(P_A[2]), .Gin2(G_A[2]), .Pin1(P_A[0]), .Gin1(G_A[0]), .P(P_B[2]), .G(G_B[2]));
   black_cell BC_B3(.Pin2(P_A[3]), .Gin2(G_A[3]), .Pin1(P_A[1]), .Gin1(G_A[1]), .P(P_B[3]), .G(G_B[3]));
   black_cell BC_B4(.Pin2(P_A[4]), .Gin2(G_A[4]), .Pin1(P_A[2]), .Gin1(G_A[2]), .P(P_B[4]), .G(G_B[4]));
   black_cell BC_B5(.Pin2(P_A[5]), .Gin2(G_A[5]), .Pin1(P_A[3]), .Gin1(G_A[3]), .P(P_B[5]), .G(G_B[5]));
   black_cell BC_B6(.Pin2(P_A[6]), .Gin2(G_A[6]), .Pin1(P_A[4]), .Gin1(G_A[4]), .P(P_B[6]), .G(G_B[6]));
   black_cell BC_B7(.Pin2(P_A[7]), .Gin2(G_A[7]), .Pin1(P_A[5]), .Gin1(G_A[5]), .P(P_B[7]), .G(G_B[7]));
   black_cell BC_B8(.Pin2(P_A[8]), .Gin2(G_A[8]), .Pin1(P_A[6]), .Gin1(G_A[6]), .P(P_B[8]), .G(G_B[8]));
   black_cell BC_B9(.Pin2(P_A[9]), .Gin2(G_A[9]), .Pin1(P_A[7]), .Gin1(G_A[7]), .P(P_B[9]), .G(G_B[9]));
   black_cell BC_B10(.Pin2(P_A[10]), .Gin2(G_A[10]), .Pin1(P_A[8]), .Gin1(G_A[8]), .P(P_B[10]), .G(G_B[10]));
   black_cell BC_B11(.Pin2(P_A[11]), .Gin2(G_A[11]), .Pin1(P_A[9]), .Gin1(G_A[9]), .P(P_B[11]), .G(G_B[11]));
   black_cell BC_B12(.Pin2(P_A[12]), .Gin2(G_A[12]), .Pin1(P_A[10]), .Gin1(G_A[10]), .P(P_B[12]), .G(G_B[12]));
   black_cell BC_B13(.Pin2(P_A[13]), .Gin2(G_A[13]), .Pin1(P_A[11]), .Gin1(G_A[11]), .P(P_B[13]), .G(G_B[13]));
   black_cell BC_B14(.Pin2(P_A[14]), .Gin2(G_A[14]), .Pin1(P_A[12]), .Gin1(G_A[12]), .P(P_B[14]), .G(G_B[14]));
   black_cell BC_B15(.Pin2(P_A[15]), .Gin2(G_A[15]), .Pin1(P_A[13]), .Gin1(G_A[13]), .P(P_B[15]), .G(G_B[15]));
```

```
    //Black Cell C
    black_cell BC_C3(.Pin2(P_B[3]), .Gin2(G_B[3]), .Pin1(1'b0), .Gin1(Cin), .P(P_C[3]), .G(G_C[3]));
    black_cell BC_C4(.Pin2(P_B[4]), .Gin2(G_B[4]), .Pin1(P_A[0]), .Gin1(G_A[0]), .P(P_C[4]), .G(G_C[4]));
    black_cell BC_C5(.Pin2(P_B[5]), .Gin2(G_B[5]), .Pin1(P_B[1]), .Gin1(G_B[1]), .P(P_C[5]), .G(G_C[5]));
    black_cell BC_C6(.Pin2(P_B[6]), .Gin2(G_B[6]), .Pin1(P_B[2]), .Gin1(G_B[2]), .P(P_C[6]), .G(G_C[6]));
    black_cell BC_C7(.Pin2(P_B[7]), .Gin2(G_B[7]), .Pin1(P_B[3]), .Gin1(G_B[3]), .P(P_C[7]), .G(G_C[7]));
    black_cell BC_C8(.Pin2(P_B[8]), .Gin2(G_B[8]), .Pin1(P_B[4]), .Gin1(G_B[4]), .P(P_C[8]), .G(G_C[8]));
    black_cell BC_C9(.Pin2(P_B[9]), .Gin2(G_B[9]), .Pin1(P_B[5]), .Gin1(G_B[5]), .P(P_C[9]), .G(G_C[9]));
    black_cell BC_C10(.Pin2(P_B[10]), .Gin2(G_B[10]), .Pin1(P_B[6]), .Gin1(G_B[6]), .P(P_C[10]), .G(G_C[10]));
    black_cell BC_C11(.Pin2(P_B[11]), .Gin2(G_B[11]), .Pin1(P_B[7]), .Gin1(G_B[7]), .P(P_C[11]), .G(G_C[11]));
    black_cell BC_C12(.Pin2(P_B[12]), .Gin2(G_B[12]), .Pin1(P_B[8]), .Gin1(G_B[8]), .P(P_C[12]), .G(G_C[12]));
    black_cell BC_C13(.Pin2(P_B[13]), .Gin2(G_B[13]), .Pin1(P_B[9]), .Gin1(G_B[9]), .P(P_C[13]), .G(G_C[13]));
    black_cell BC_C14(.Pin2(P_B[14]), .Gin2(G_B[14]), .Pin1(P_B[10]), .Gin1(G_B[10]), .P(P_C[14]), .G(G_C[14]));
    black_cell BC_C15(.Pin2(P_B[15]), .Gin2(G_B[15]), .Pin1(P_B[11]), .Gin1(G_B[11]), .P(P_C[15]), .G(G_C[15]));

    //Black Cell D
    black_cell BC_D7(.Pin2(P_C[7]), .Gin2(G_C[7]), .Pin1(1'b0), .Gin1(Cin), .P(P_D[7]), .G(G_D[7]));
    black_cell BC_D8(.Pin2(P_C[8]), .Gin2(G_C[8]), .Pin1(P_A[0]), .Gin1(G_A[0]), .P(P_D[8]), .G(G_D[8]));
    black_cell BC_D9(.Pin2(P_C[9]), .Gin2(G_C[9]), .Pin1(P_B[1]), .Gin1(G_B[1]), .P(P_D[9]), .G(G_D[9]));
    black_cell BC_D10(.Pin2(P_C[10]), .Gin2(G_C[10]), .Pin1(P_B[2]), .Gin1(G_B[2]), .P(P_D[10]), .G(G_D[10]));
    black_cell BC_D11(.Pin2(P_C[11]), .Gin2(G_C[11]), .Pin1(P_C[3]), .Gin1(G_C[3]), .P(P_D[11]), .G(G_D[11]));
    black_cell BC_D12(.Pin2(P_C[12]), .Gin2(G_C[12]), .Pin1(P_C[4]), .Gin1(G_C[4]), .P(P_D[12]), .G(G_D[12]));
    black_cell BC_D13(.Pin2(P_C[13]), .Gin2(G_C[13]), .Pin1(P_C[5]), .Gin1(G_C[5]), .P(P_D[13]), .G(G_D[13]));
    black_cell BC_D14(.Pin2(P_C[14]), .Gin2(G_C[14]), .Pin1(P_C[6]), .Gin1(G_C[6]), .P(P_D[14]), .G(G_D[14]));
    black_cell BC_D15(.Pin2(P_C[15]), .Gin2(G_C[15]), .Pin1(P_C[7]), .Gin1(G_C[7]), .P(P_D[15]), .G(G_D[15]));

    //Black Cell E
    black_cell BC_E15(.Pin2(P_D[15]), .Gin2(G_D[15]), .Pin1(1'b0), .Gin1(Cin), .P(P_E[15]), .G(G_E[15]));

    //Carry Genration
    Carry_Gen carry_1(.P(P_A[0]), .G(G_A[0]), .Cin(Cin), .Cout(Carry[1]));
    Carry_Gen carry_2(.P(P_B[1]), .G(G_B[1]), .Cin(Cin), .Cout(Carry[2]));
    Carry_Gen carry_3(.P(P_B[2]), .G(G_B[2]), .Cin(Cin), .Cout(Carry[3]));
    Carry_Gen carry_4(.P(P_C[3]), .G(G_C[3]), .Cin(Cin), .Cout(Carry[4]));
    Carry_Gen carry_5(.P(P_C[4]), .G(G_C[4]), .Cin(Cin), .Cout(Carry[5]));
    Carry_Gen carry_6(.P(P_C[5]), .G(G_C[5]), .Cin(Cin), .Cout(Carry[6]));
    Carry_Gen carry_7(.P(P_C[6]), .G(G_C[6]), .Cin(Cin), .Cout(Carry[7]));
    Carry_Gen carry_8(.P(P_D[7]), .G(G_D[7]), .Cin(Cin), .Cout(Carry[8]));
    Carry_Gen carry_9(.P(P_D[8]), .G(G_D[8]), .Cin(Cin), .Cout(Carry[9]));
    Carry_Gen carry_10(.P(P_D[9]), .G(G_D[9]), .Cin(Cin), .Cout(Carry[10]));
    Carry_Gen carry_11(.P(P_D[10]), .G(G_D[10]), .Cin(Cin), .Cout(Carry[11]));
    Carry_Gen carry_12(.P(P_D[11]), .G(G_D[11]), .Cin(Cin), .Cout(Carry[12]));
    Carry_Gen carry_13(.P(P_D[12]), .G(G_D[12]), .Cin(Cin), .Cout(Carry[13]));
    Carry_Gen carry_14(.P(P_D[13]), .G(G_D[13]), .Cin(Cin), .Cout(Carry[14]));
    Carry_Gen carry_15(.P(P_D[14]), .G(G_D[14]), .Cin(Cin), .Cout(Carry[15]));
    Carry_Gen carry_16(.P(P_D[15]), .G(G_D[15]), .Cin(Cin), .Cout(Carry[16]));
```

```
    //Sum
    assign Sum[0] = Cin ^ P_st_1[0];
    assign Sum[1] = Carry[1] ^ P_st_1[1];
    assign Sum[2] = Carry[2] ^ P_st_1[2];
    assign Sum[3] = Carry[3] ^ P_st_1[3];
    assign Sum[4] = Carry[4] ^ P_st_1[4];
    assign Sum[5] = Carry[5] ^ P_st_1[5];
    assign Sum[6] = Carry[6] ^ P_st_1[6];
    assign Sum[7] = Carry[7] ^ P_st_1[7];
    assign Sum[8] = Carry[8] ^ P_st_1[8];
    assign Sum[9] = Carry[9] ^ P_st_1[9];
    assign Sum[10] = Carry[10] ^ P_st_1[10];
    assign Sum[11] = Carry[11] ^ P_st_1[11];
    assign Sum[12] = Carry[12] ^ P_st_1[12];
    assign Sum[13] = Carry[13] ^ P_st_1[13];
    assign Sum[14] = Carry[14] ^ P_st_1[14];
    assign Sum[15] = Carry[15] ^ P_st_1[15];

    assign Cout = Carry[16];
endmodule
```

처음엔 top module 명을 KSA로 했다가, 모듈명이 중복된다는 오류가 떠서 myKSA로 고쳤습니다. 16비트 숫자 A, B와 1비트 캐리 Cin을 입력으로 받고, 16비트 숫자 Sum과 1비트 캐리 Cout을 출력으로 갖습니다.

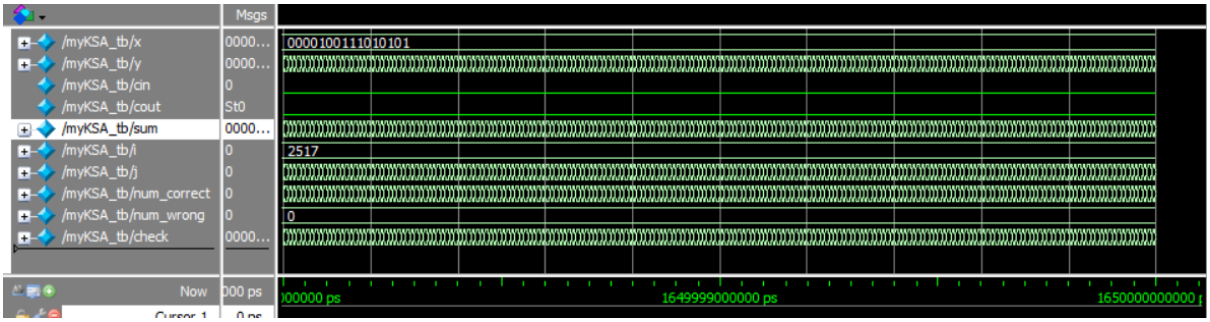
그리고 각 PG Generator, Black Cell, Carry Generator 들을 연결해줄 wire를 선언한 후, 이들을 이어줍니다.

최종적으로 구한 캐리와 맨 처음 구한 각 비트의 Propagation을 XOR 연산하여 Sum을 구합니다.

- myKSA testbench

```
`timescale 1ns/100ps

module myKSA_tb;

//input(reg)
reg [15:0] x, y;
reg cin;

//output(wire)
wire cout;
wire [15:0] sum;

integer i, j;
integer num_correct, num_wrong;
reg [16:0] check;

myKSA KSA0(.A(x), .B(y), .Cin(cin), .Cout(cout), .Sum(sum));

initial begin
    num_correct = 0; num_wrong = 0;
    for(i=0; i<65536; i=i+1)begin
        x=i;
        for(j=0; j<65536; j=j+1)begin
            y=j;
            cin = 1'b0;
            check = x + y + cin;

            #10
            if({cout, sum} == check)
                num_correct = num_correct+1;
            else
                num_wrong = num_wrong+1;
        end
    end

    $display("num_correct = %d, num_wrong = %d", num_correct, num_wrong); //print
end
endmodule
```

Ripple Carry Adder 실습에서 사용한 testbench와 거의 유사합니다. 16비트 덧셈기를 테스트하기 위해 입력을 0부터 65535(2^16 – 1)까지 주었습니다.



시뮬레이션에 생각보다 많은 시간이 소요되어, 끝까지 결과를 확인할 수 없었으나, i 반복자가 2517회 반복될 동안 연산이 틀린 경우가 한 번도 없었습니다.

4. Evaluation

설계한 덧셈기의 결과값과, {Cout, Sum} 값이 같으므로, 제대로 설계되었다고 볼 수 있습니다.

5. Discussion

이번 실습의 key는 Black Cell들을 잘 연결해주는 것입니다.

Full Adder의 개수가 n개라고 할 때, RCA의 시간복잡도는 O(n) 이고, KSA의 시간복잡도는 O(1)입니다. 연산 성능을 올리기 위해서 하드웨어의 복잡도가 매우 올라갔다는 점이 신기했습니다.

설계를 더욱 향상시킬 방법은 없는 것 같습니다.