컴퓨터구조실험

과제 이름: 4-bit Ripple Carry Adder

담당교수: 이성원

학 과: 컴퓨터정보공학부

학 번: 2019202021

이 름: 정 성 엽

제출일: 2023/3/22

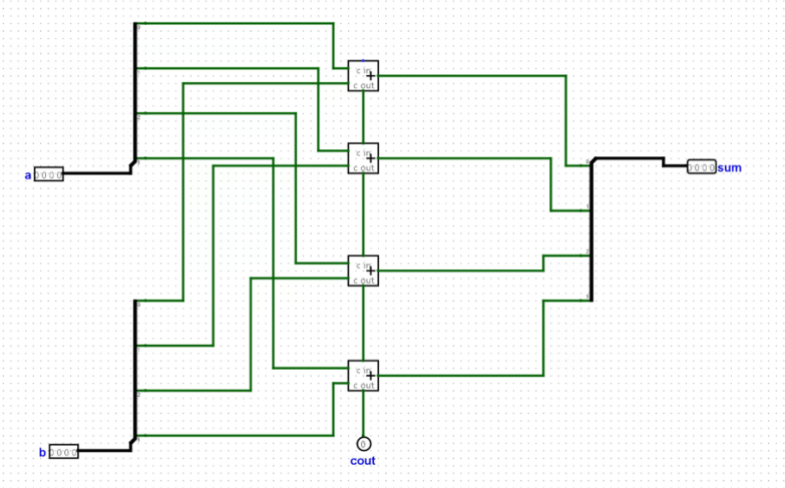
1. Introduction

A. 과제 소개

- Logisim-evolution와 Icarus Verilog, gtkwave를 통해서 4-bit Ripple Carry Adder를 설계하고 구현한다. Logisim-evolution을 이용하여 입력을 넣어보며 동작 결과를 확인하고 동작 결과의 근거를 파악해본다. 또한 Verilog 작성 후 gtkwave에서의 파형을 확인하여 Logisim-evolution 결과와 같게 나오는지 비교해본다.

2. 결과화면

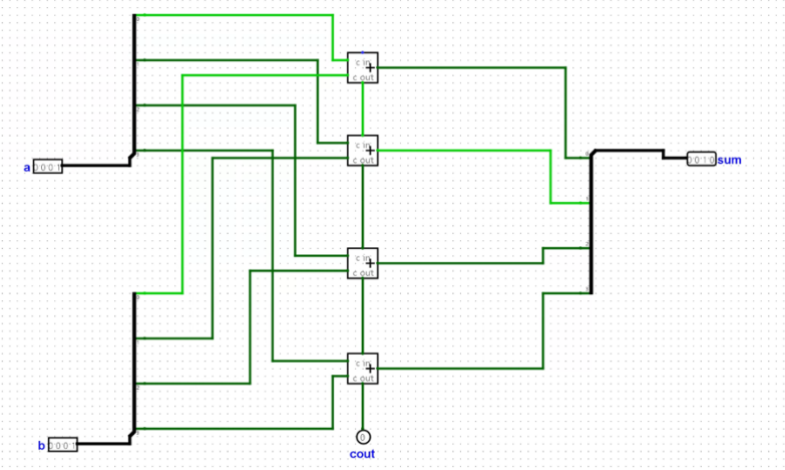
A. Logisim-evolution



<4-bit Ripple Carry Adder 회로>

1- bit Adder를 사용하여 4-bit Ripple Carry Adder를 구현하였다. 4bit의 Pin a, b는 Splitter를 통해 각 비트를 쪼갠다. 각 비트는 1-bit Adder로 들어가서 덧셈 연산이 실행되고 각 sum 값은 우측 splitter를 통해 4-bit sum에 취합된다. 또한 각 1-bit Adder의 c_out은 다음 비트의 1-bit Adder의 c_in으로 들어가거나 최종 c_out에 들어간다.

예시1)

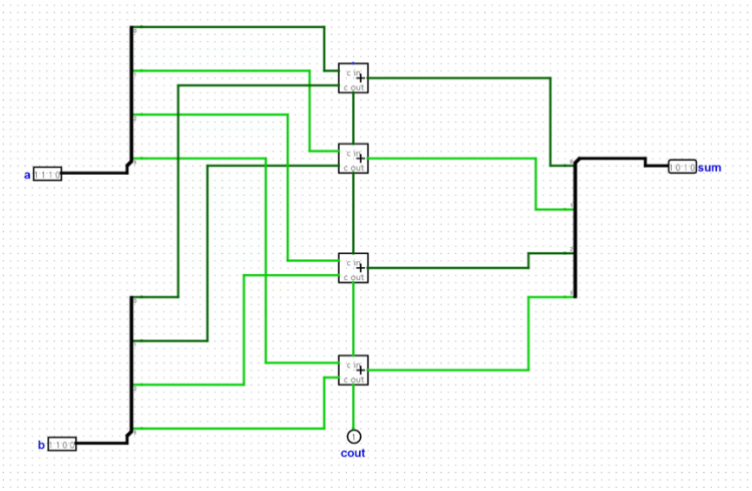


<4'b1과 4'b1의 덧셈 결과>

각 Pin에서 4'b1을 설정했을 때 Splitter를 통해 첫 번째 1-bit Adder에만 값이 들

어가고 이때 sum 값은 0이고 c_out은 1이므로 다음 1-bit Adder에 입력된다. 해당 Adder에서는 c_in 이외의 입력이 없으므로 sum 값은 1로 set된다. 이 set된 값은 우측 splitter를 통해 최종 sum값에 취합되어 4'b10이 되었다.

예시2)



<4'b1110과 4'b1100의 덧셈 결과>

위와 마찬가지로 각 Pin의 값은 Splitter를 통해 갈라지고 각 1-bit Adder에 들어가 덧셈 연산을 진행하고 c_out이 발생한 경우 다음 1-bit Adder에 c_in으로 값을 입력한다. 첫 번째 bit는 둘 다 0이라 해당 Adder에서의 sum과 c_out은 0이다. 두 번째 bit는 a만 1로 set되었고 해당 Adder에서 하나의 입력만 set 되었으므로 sum 값은 1이고 c_out은 0이다. 세 번째 bit는 둘 다 1이고 해당 Adder에서 두 개의 입력이 set 되었으므로 sum 값은 0이고 c_out은 1이다. 마지막 bit는 둘 다 1이고 해당 Adder에서 a, b, c_in 모두 1로 set 되었기 때문에 sum 값과 마지막 c_out 값은 1이다.

B. Icarus Verilog, gtkwave

- module FA

a	b	cin	cout	sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

sum 값은 ~a&~b&cin 또는 ~a&b&~cin 또는 a&~b&~cin 또는 a&b&cin일 때 1로 set 된다.

cout 값은 a&b 또는 a&cin 또는 b&cin 일 때 1로 set 된다.
이를 통해 Full Adder의 Verilog를 작성하면 아래와 같다.

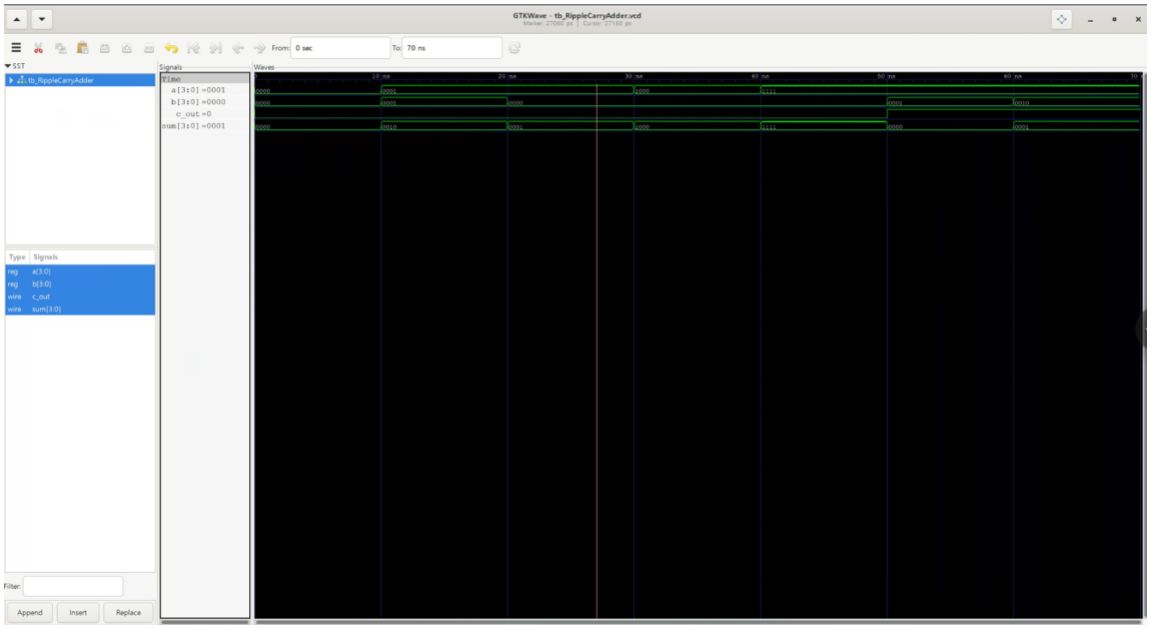
```
module FA (a, b, cin, cout, sum);
    input a, b, cin;
    output cout,sum;

    assign cout = (a&b)+(b&cin)+(a&cin);
    assign sum =  (~a&~b&cin)+(~a&b&~cin)+(a&b&cin)+(a&~b&~cin);

endmodule;
```

위의 Full Adder를 통해서 4-bit Ripple Carry Adder의 Verilog를 작성하면 아래와 같다.

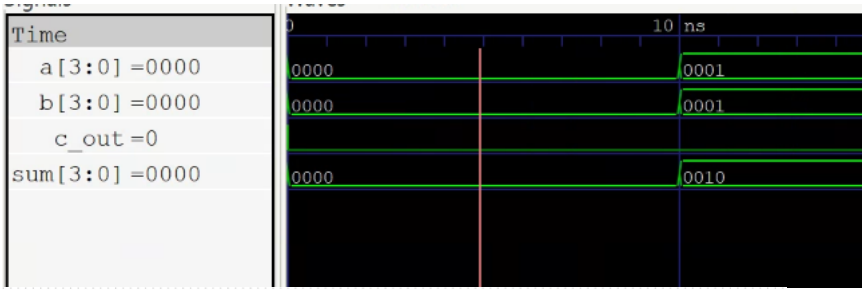
```
module RippleCarryAdder (a, b, c_out,sum);
    input[3:0] a, b;
    output c_out;
    output [3:0] sum;

    wire c0, c1, c2;

    FA fa0(.a(a[0]),.b(b[0]),.cin(1'b0),.cout(c0),.sum(sum[0]));
    FA fa1(.a(a[1]),.b(b[1]),.cin(c0),.cout(c1),.sum(sum[1]));
    FA fa2(.a(a[2]),.b(b[2]),.cin(c1),.cout(c2),.sum(sum[2]));
    FA fa3(.a(a[3]),.b(b[3]),.cin(c2),.cout(c_out),.sum(sum[3]));

endmodule;
```

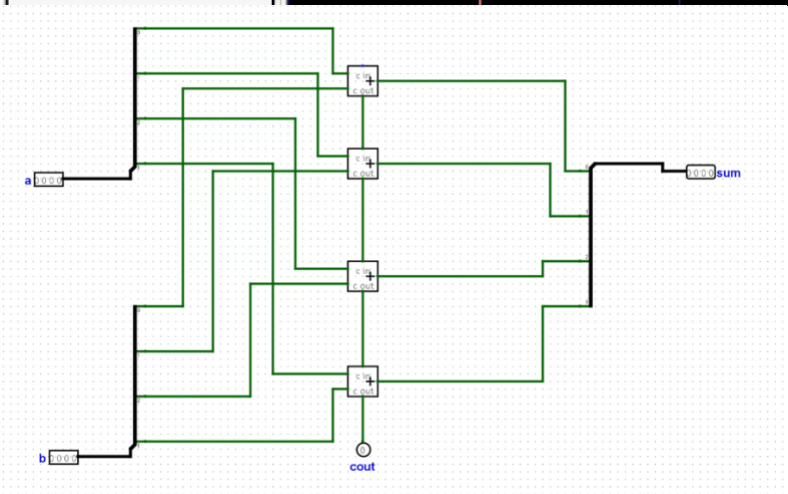
FA 모듈을 4번 사용하여 4-bit Ripple Carry Adder를 구현하였으며 각 FA에서의 cin 값을 위해 wire로 c0, c1, c2를 선언하였다. 이 때 첫 번째 FA 모듈에서는 cin 값을 1'b0으로 할당했는데 적지 않고 생략해도 된다.

이 코드를 Icarus Verilog로 컴파일 한 후에 gtkwave로 파형을 확인하여 Logisim-evolution의 값과 비교하였다.



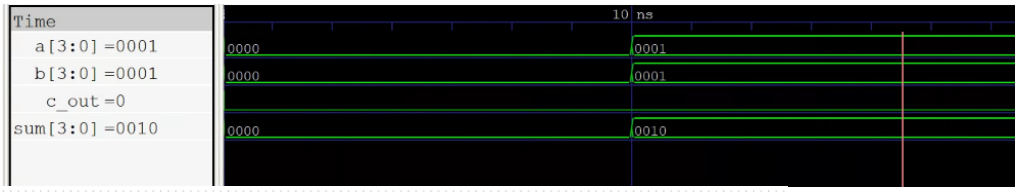
1) 모두 0일 때

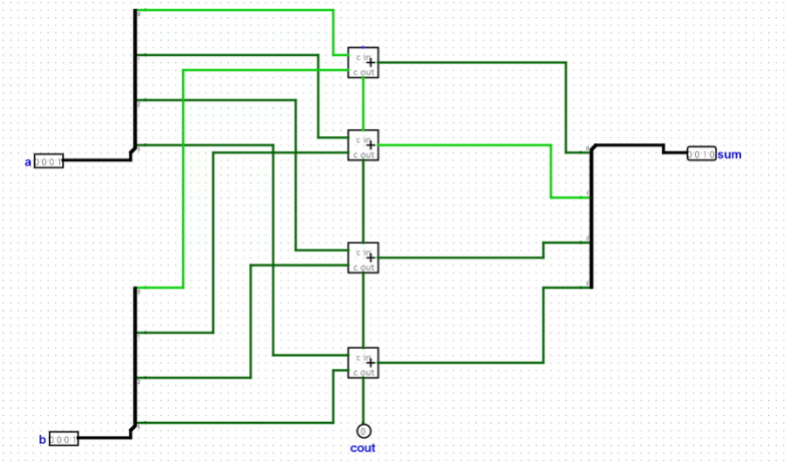




모두 0일 때 sum과 c_out 모두 0인 것을 확인하였다.

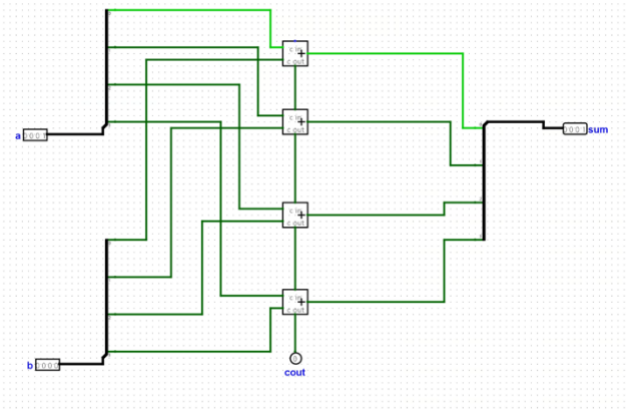
2) 4'b0001과 4'b0001의 합





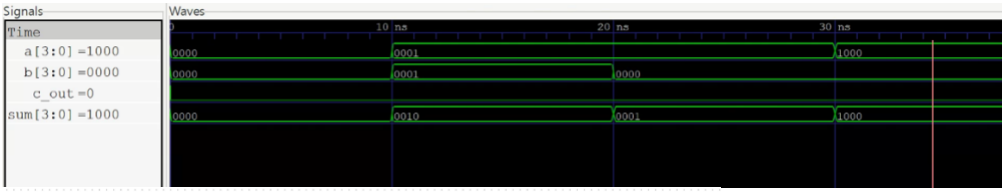
4'b0001과 4'b0001을 합하였을 때 sum이 4'b0010 c_out이 1'b0인 것을 확인하였다.

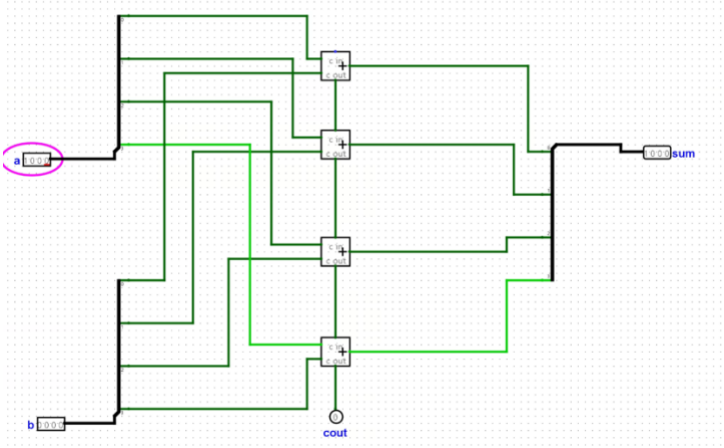
3) 4'b0001과 4'b0의 합



4'b0001과 4'b0을 합하였을 때 sum이 4'b0001 c_out이 1'b0인 것을 확인하였다.

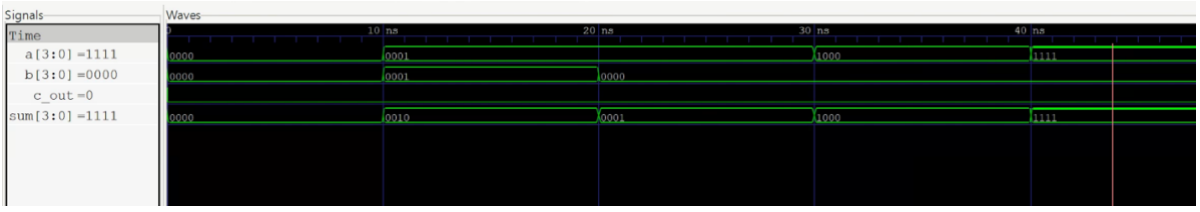
4) 4'b1000과 4'b0의 합

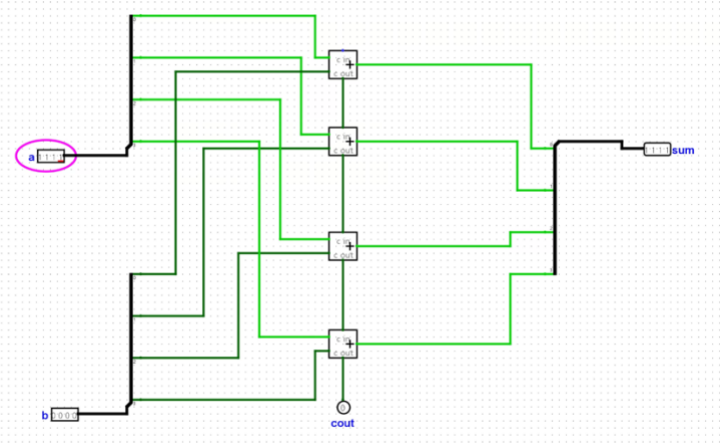




4'b1000과 4'b0000을 합하였을 때 sum이 4'b1000 c_out이 1'b0인 것을 확인하였다.

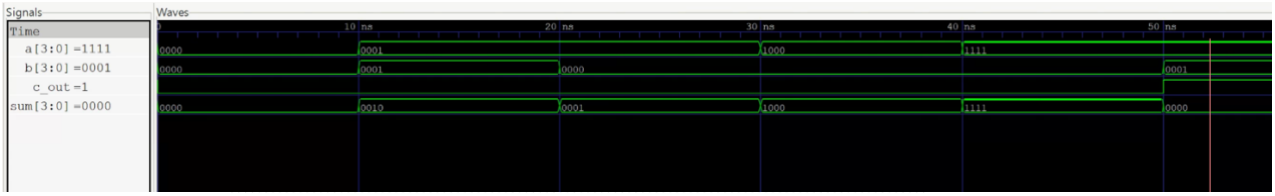
5) 4'b1111과 4'b0의 합

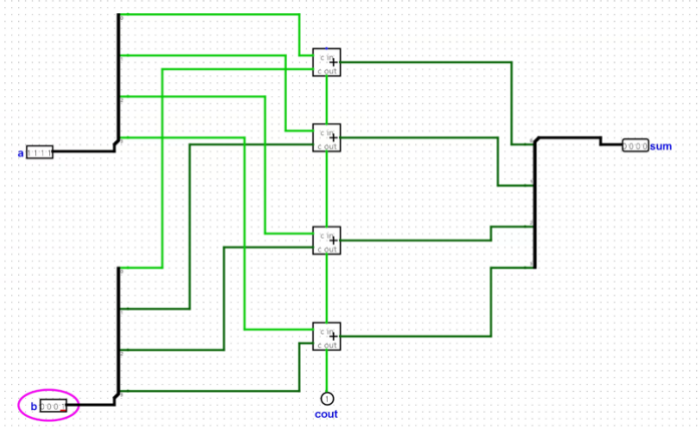




4'b1111과 4'b0000을 합하였을 때 sum이 4'b1111 c_out이 1'b0인 것을 확인하였다.

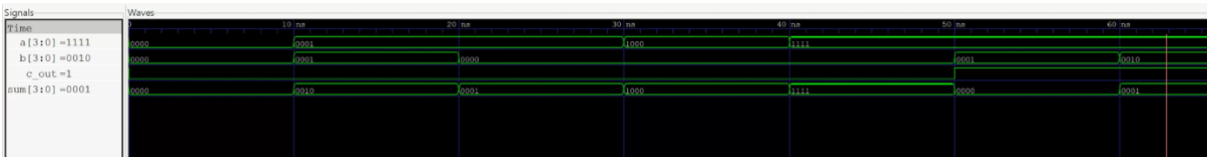
6) 4'b1111과 4'b0001의 합

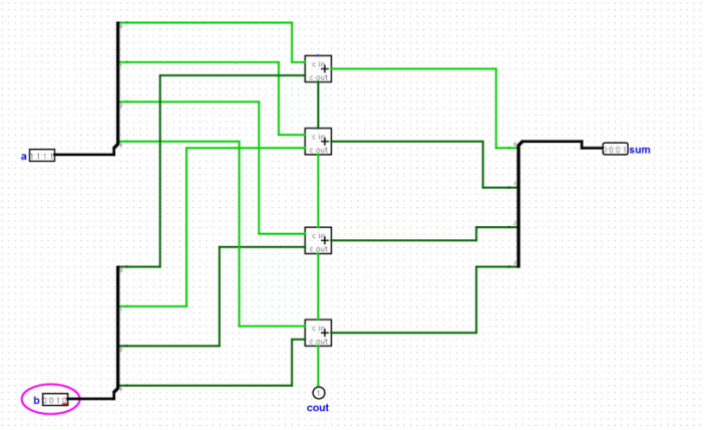




4'b1111과 4'b0001을 합하였을 때 sum이 4'b0 c_out이 1'b1인 것을 확인하였다.

7) 4'b1111과 4'b0010의 합





4'b1111과 4'b0010을 합하였을 때 sum이 4'b0001 c_out이 1'b1인 것을 확인하였다.

이를 통해 Logisim-evolution의 회로와 Verilog를 통해 구현한 회로가 일치함을 확인하였다.

3. 고찰

Logisim-evolution을 통해서 4-bit Ripple Carry Adder를 구현하였다. 이전 컴퓨터공학기초실험2에서 회로도를 그리기 위해 draw.io를 사용했는데 해당 회로는 작동을 확인하지 못하지만 Logisim-evolution은 회로를 구현 후 작동 가능하다는 점이 매력적이었다. 다만 각 부품에 대한 기능을 제대로 알고 사용해야함은 공부를 필요로 했다. 이번 4-bit Ripple Carry Adder에서 4bit 값을 각 Full Adder로 전달하기 위해 각 1bit 씩 pin을 잡아야 하나 했는데 Splitter를 통해서 4bit pin 또한 각 Full Adder로 전달할 수 있음을 확인했다. Full Adder에 값을 연결하고 c_out은 다음 Full Adder로 전달하였고 sum 값은 후에 Splitter로 다시 취합하여 최종 sum 값으로 전달하였다.

Verilog로 Full Adder를 구현할 때는 a, b, cin 값에 따른 cout과 sum 값의 변화를 확인하여 논리식으로 만들어 이를 Verilog로 구현하였다. Full Adder 구현 후 Ripple Carry Adder에서는 Full Adder 4개 사용하여 연결하였다. 후에 이의 test bench를 gtkwave로 파형을 확인한 결과 Ripple Carry Adder가 정상적으로 작동함을 확인하고 먼저 만든 Logisim-evolution과도 결과가 일치함을 확인했다.

4. Reference

1) 컴퓨터구조실험 2주차 실습 자료/광운대학교 컴퓨터정보공학부/ 이성원 교수님/2023