실험 4. 이진수 연산

20210774 김주은

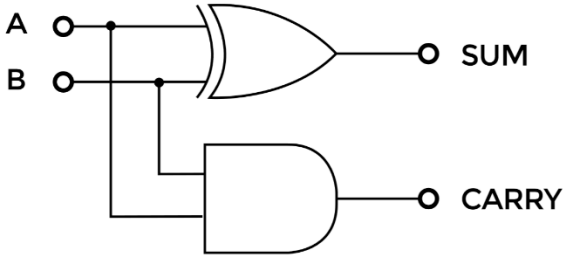
1. 개요

이번 실습의 목표는 이진수 덧셈을 이해한 후 이진수 덧셈을 실제적으로 구현하는 회로들을 이해하는 것이다. 그래서 이번 실습에서는 반가산기와 전가산기, 5-bit 리플 가산기/감산기, 5X3 이진 곱셈기를 구현한다. 이를 통해서 이진수의 덧셈, 뺄셈, 곱셈을 직접 구현해보는 것이다. 반가산기를 통해서 전가산기를 구현하고 반가산기와 전가산기 사이의 연관관계를 이해한다. 그리고 전가산기를 이용하여 5-bit 리플 가산기/감산기를 만들어 이 사이의 관계도 이해한다. 5X3 이진 곱셈기도 리플 가산기를 활용하여 이 사이의 연관관계도 이해한다.

2. 이론적 배경

1) 반가산기

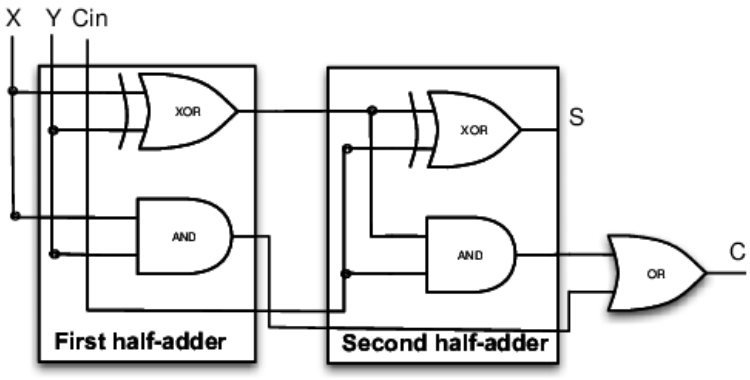
반가산기는 1-bit 이진수 두 개를 입력 받고 이 두 이진수의 합과 carry out 두 개의 출력을 받는다.



A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

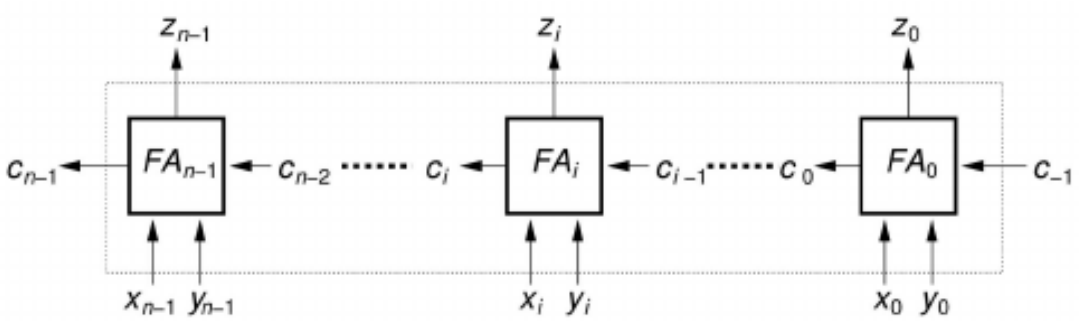
2) 전가산기

전가산기는 반가산기와 비슷하지만 1-bit 이진수 두 개에 더해 이전 가산기의 carry in까지 입력받는다. 입력 3개에 대한 출력값으로 합과 carry out을 출력한다. 이 전가산기는 half adder 2개를 사용하여 구현할 수 있다.



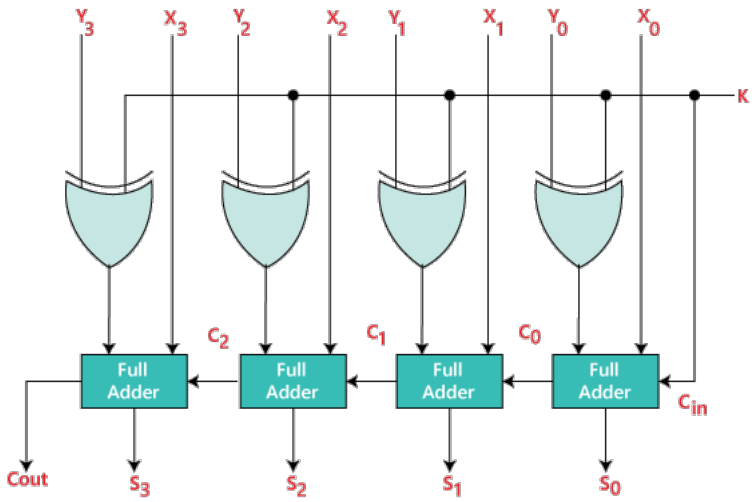
3) N-bit 리플 가산기/감산기

N-bit 리플 가산기는 N-bit 이진수 두 개를 더하여 N-bit 덧셈 결과와 carry out을 출력한다. 리플 가산기의 경우 N개의 full adder을 차례대로 연결하고 carry in과 carry out을 계속하여 연결한다. 그래서 carry의 경우 delay가 발생한다.



감산기의 경우는 보수를 활용하여 가산기의 입력값으로 넣으면 감산기로 구현이 된다.

N-bit 리플 가산기와 감산기를 한꺼번에 구현도 가능하다. 여기서 XORgate를 쓴다면 XOR gate는 not gate로도 이용되기 때문에 XOR gate를 써서 감산기와 가산기를 하나의 회로로 한꺼번에 구현할 수 있다.



회로로 구현하면 이렇게 구현된다.

4) MXN 이진 곱셈기

MXN에서 M과 N은 bit를 뜻한다. 즉, M-bit Multiplicand와 N-bit Multiplier의 각 자릿수 부분곱을 통해 얻은 이진수 N개를 합하여 계산한다. 그러므로 합을 할 때 가산기를 활용한다. 이진수이므로 곱은 and gate로 구현이 되기 때문에 and gate와 가산기로 이진 곱셈기가 구현가능하다.

3. 실험 준비

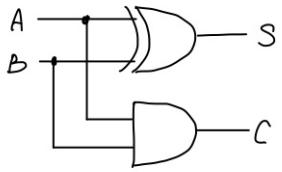
Lab 4-1)

<Half adder 진리표>

A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$S = A'B + AB' = A \oplus B$

$C = AB$



<full adder 진리표>

A	B	C_{in}	C_{out}	S
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

S

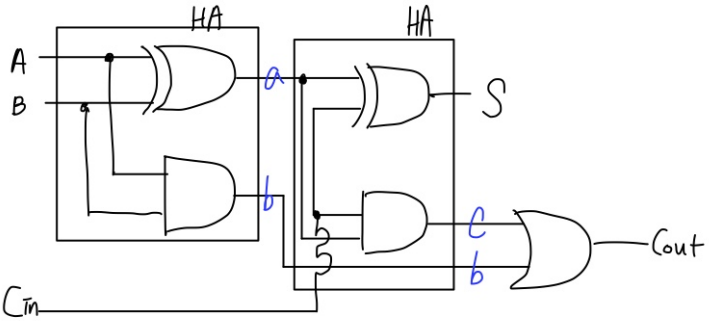
A \ BC	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$S = A \oplus B \oplus C_{in}$

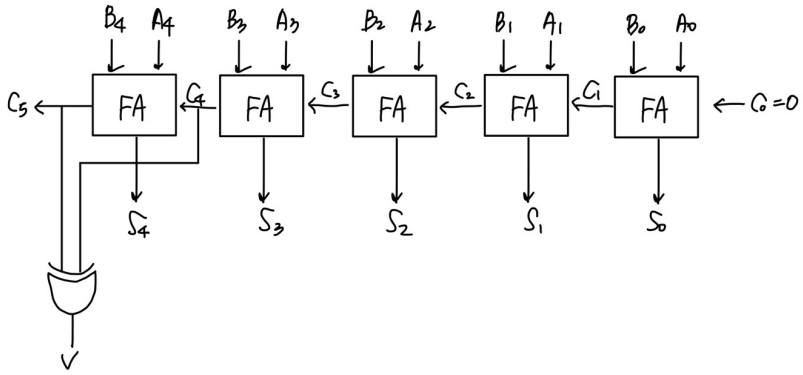
C_{out}

A \ BC	00	01	11	10
0	0	0	1	0
1	0	1	1	1

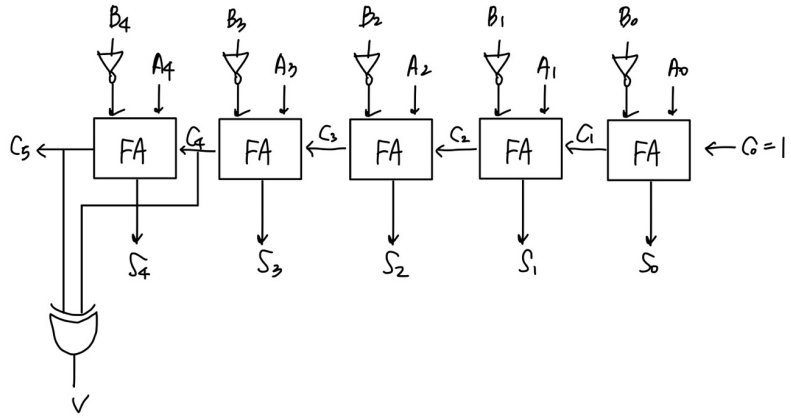
$C_{out} = AB + BC_{in} + AC_{in}$



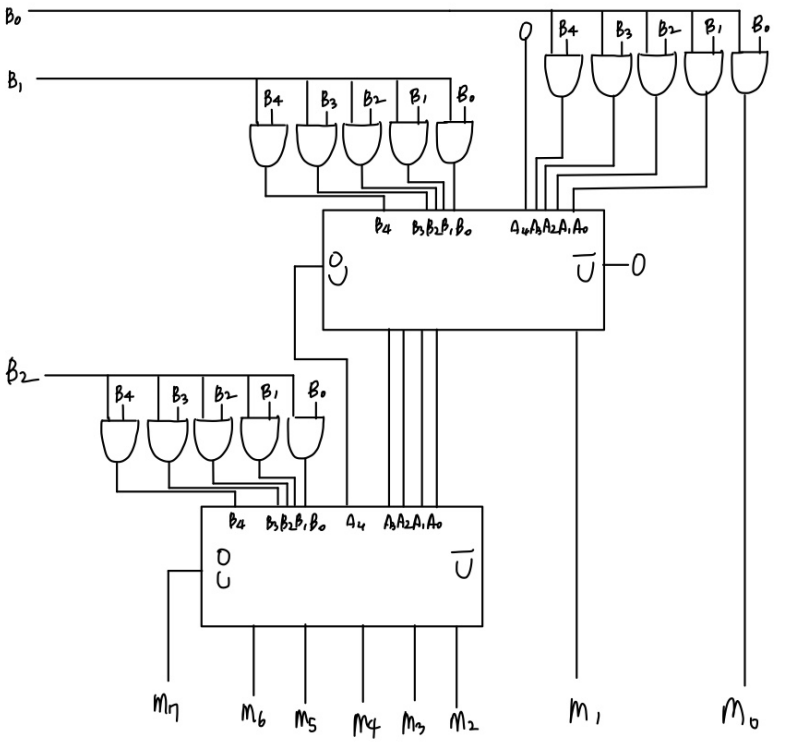
Lab 4-2)



Lab 4-3)

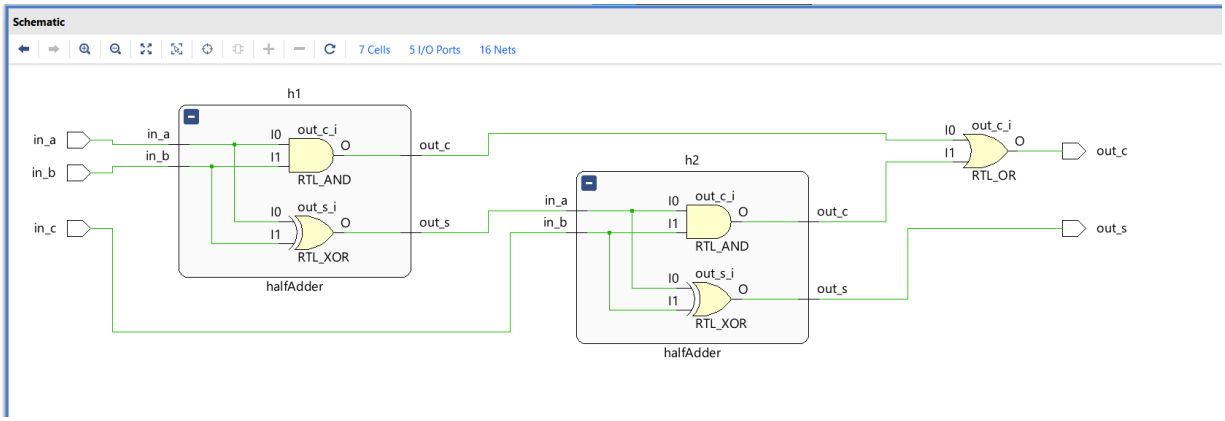

B4
B3
B2
B1
B0
A4
A3
A2
A1
A0
FA
FA
FA
FA
FA
C5
C4
C3
C2
C1
C0 = 1
S4
S3
S2
S1
S0
V

Lab 4-4)

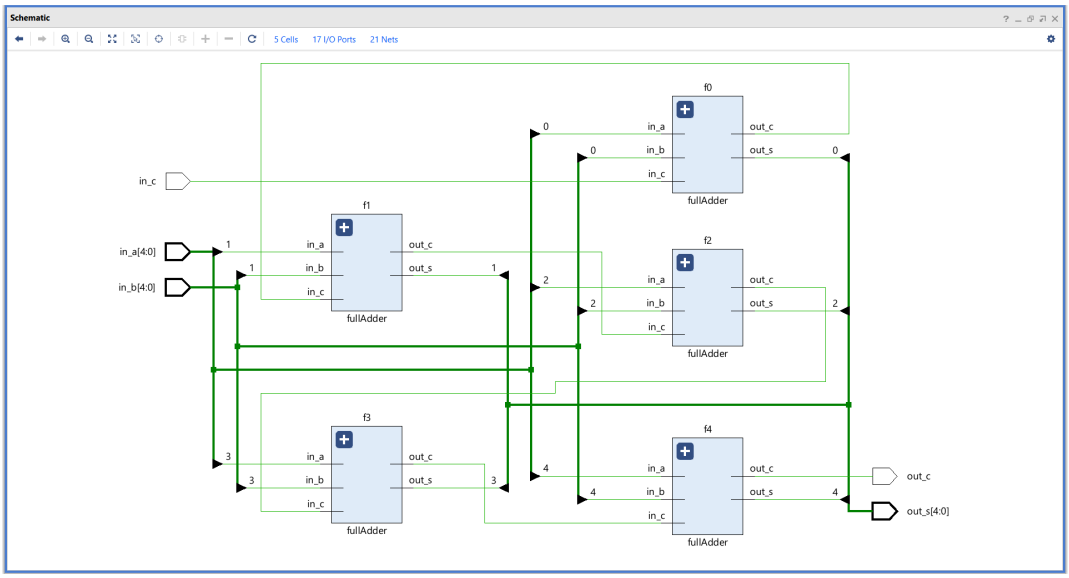

B0
B1
B2
0
B4
B3
B2
B1
B0
A4A3A2A1A0
U
U
0
M7
M6
M5
M4
M3
M2
M1
M0

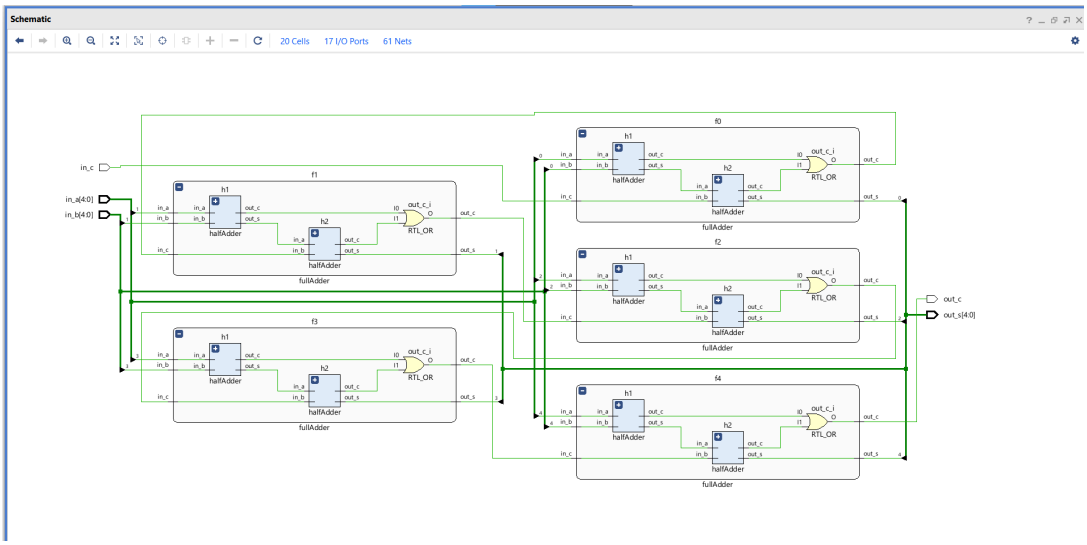
4. 결과

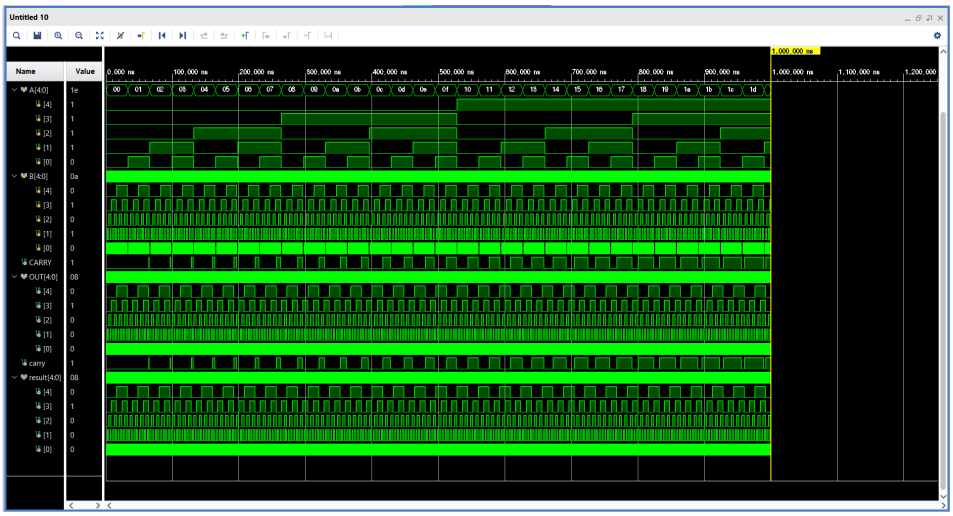
Lab 4-1)



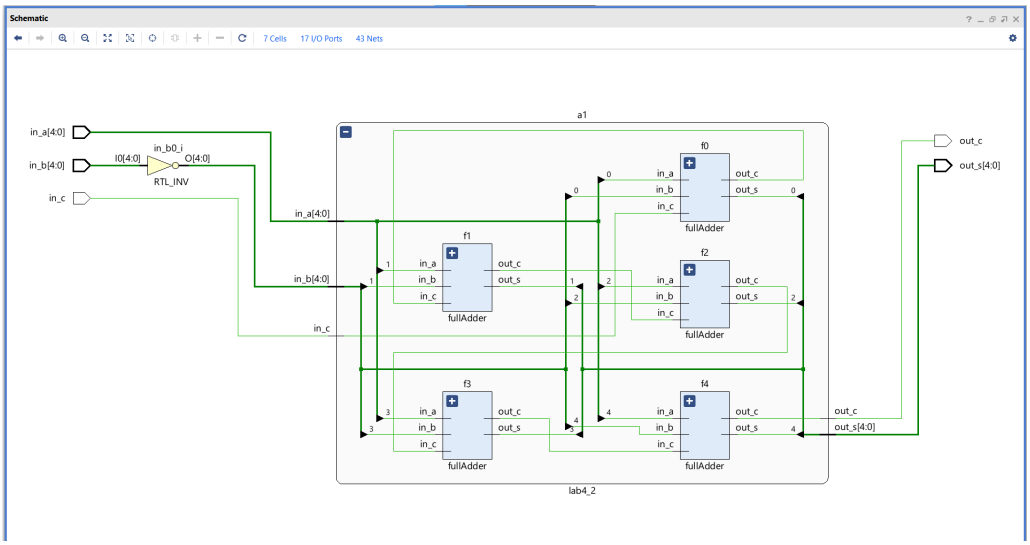
Lab 4-2)



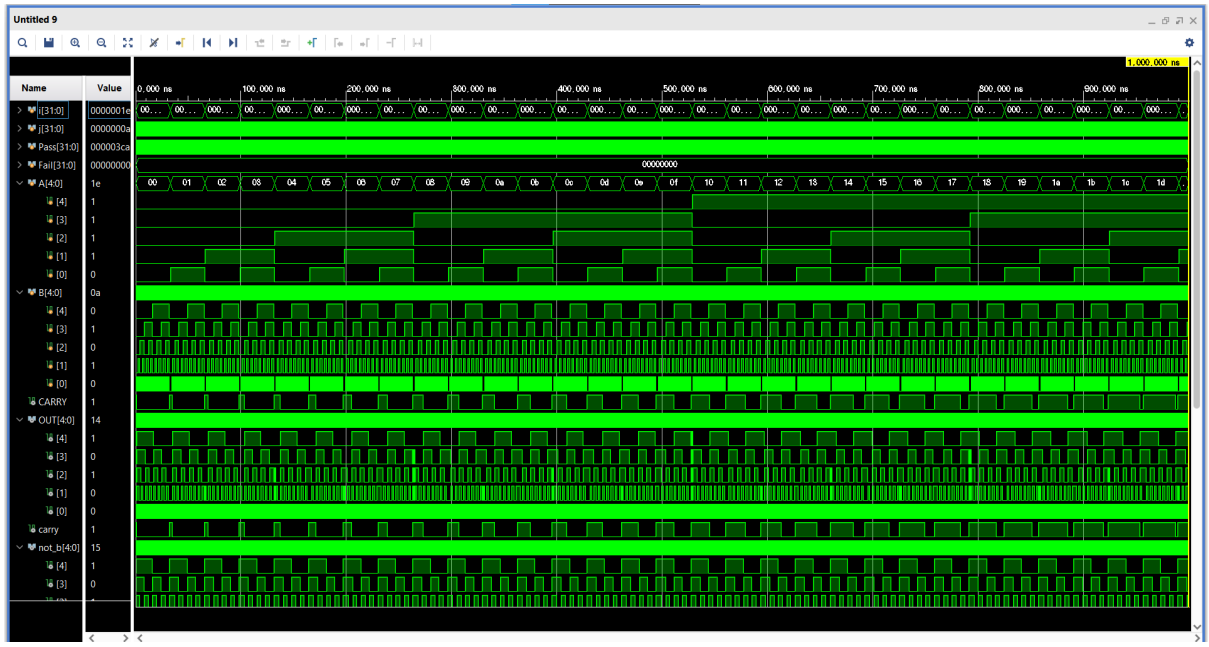




Lab 4-3)

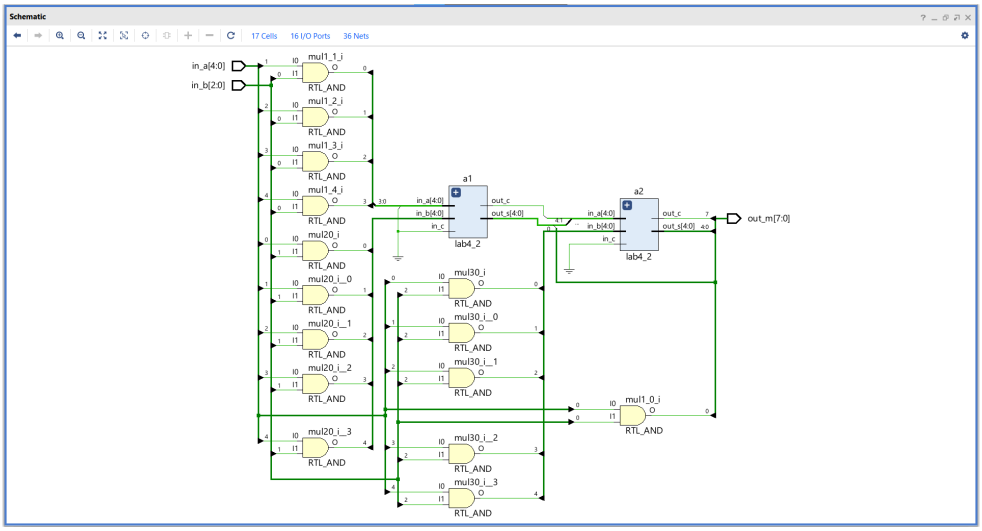


Lab4_2는 내부는 동일하므로 캡쳐하지 않았다.


Untitled 9
Name
Value
i[31:0]
j[31:0]
Pass[31:0]
Fail[31:0]
A[4:0]
B[4:0]
CARRY
OUT[4:0]
carry
not_b[4:0]
1,000.000 ns


Untitled 9
Name
Value
OUT[4:0]
carry
not_b[4:0]
comp...5:0]
result[4:0]
1,000.000 ns

Lab 4-4)



Lab4_2는 내부는 동일하므로 캡쳐하지 않았다.

5. 논의

코딩 과정에 있어서 여러 오류가 발생했는데 가장 큰 실수는 fulladder 등 미리 만든 모듈에 있어서 모듈의 이름을 짓지 않고 썼기 때문에 발생한 오류였다. 그래서 다시 이름을 지어 모듈을 만드니 해결되었다.

그리고 감산기를 구현하는 경우에도 2의 보수를 구현하는 과정에서 많은 고민을 했는데 먼저 complement를 구하고 1을 더해야 하는 부분이 문제였다. 하지만 테스트벤치에서 자동적으로 입력을 할 때 1을 더해주는 것을 확인하고 assign과 ~을 활용하여 보수를 구해주는 과정만 완성시

켜 구현에 완성하였다.