Lab5. ALU와 JK 플립플롭

20200437 김채현

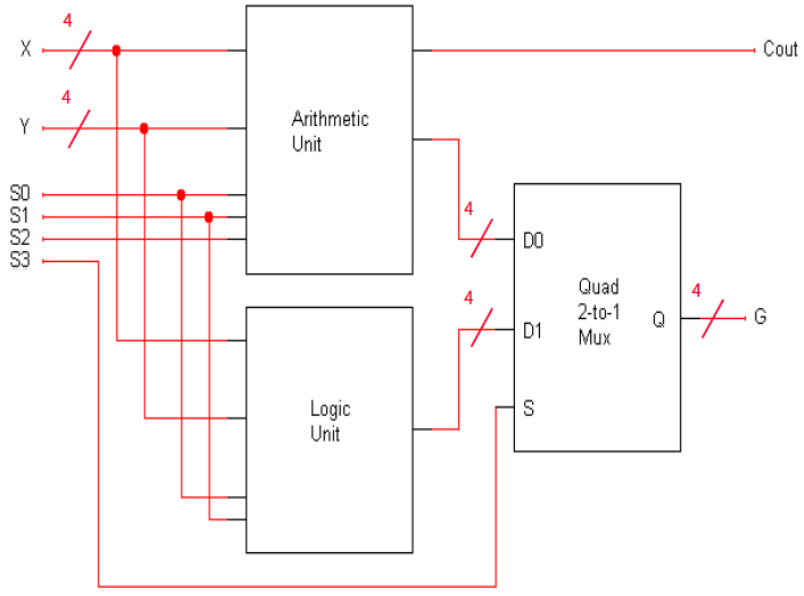
1. 개요

본 실험은 컴퓨터의 핵심 장치인 ALU를 구현하고, 정보 저장 장치인 JK 플립플롭을 구현하는 것을 목적으로 한다. 또한 test bench를 구현하여 그 작동을 확인한다.

2. 이론적 배경

1) ALU (Arithmetic Logic Unit)

ALU는 덧셈, 뺄셈 같은 두 숫자의 산술연산과 배타적 논리합, 논리곱, 논리합 같은 논리연산을 계산하는 디지털 회로이다. 그렇기에 ALU는 컴퓨터 중앙처리장치(CPU)의 기본 설계 블록이라 할 수 있다. ALU는 입력에 대해 산술 부분과(Arithmetic Unit) 및 논리 부분(Logic Unit)으로 나눌 수 있는데, 산술 장치는 사칙연산 등을, 그리고 논리 장치는 Bitwise 논리 연산 등을 맡는다. ALU는 연산할 Data Input과, 어떤 연산을 수행할지 결정하는 Selection Input을 통해 구현하고자 하는 연산을 수행하게 된다.

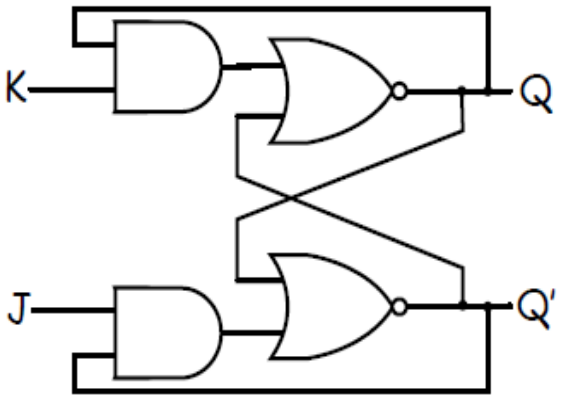


2) 비동기 회로 / 동기 회로

비동기 회로(asynchronous circuit)는 클럭 회로에 의해 운영되지 않고 명령어나 연산이 종료되었다는 신호를 기다리는 방식의 디지털 회로이다. 즉, 모든 조합 회로와 클록을 따르지 않는 순차 회로는 비동기 회로다. 동기 회로는 다른 회로와 같은 순간에 맞춰 작동하기 위해 클록 신호를 따른다.

3) JK 래치 / JK 플립플롭

디지털 순차회로를 구성하기 위한 기억 소자 중 하나로, SR Latch에 추가로 J, K, Input을 AND gate를 통하여 인가한 회로이다.



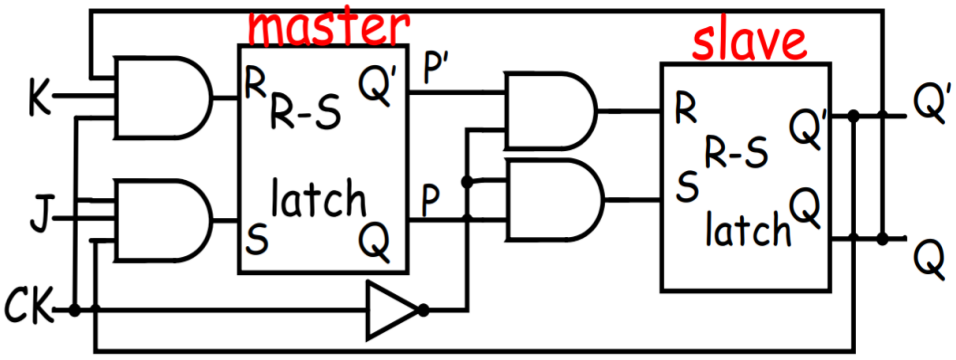
J	K	Q	Q^+	
0	0	0	0	hold
0	0	1	1	
0	1	0	0	reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	1	toggle
1	1	1	0	

JK Latch는 자신의 출력과 Control input J, K를 이용하여 Q, Q'를 출력한다. JK Latch에서 J, K가 모두 0인 경우는 값을 hold하며, K가 1인 경우는 reset, J가 1인 경우는 set, 그리고 J, K가 모두 1인 경우에는 현재 상태와 상관없이 값을 반전시킨다. JK Latch는 순차회로로써 처음 값을 예측하기 어려운데, 이를 해결하기 위해 Preset(Set Q to 1 already)과 Clear(Set Q to 0)를 통해 다양한 동작에 대응할 수 있도록 한다.

4) Master-slave JK 플립플롭

Flip-Flop은 Latch에 Clock을 추가하여 output이 특정 상황에서 바뀔 수 있도록 설계한 기억 소자이다. 그 중 Master-slave JK 플립플롭은 SR Latch 두 개를 연결하여 만든 플립플롭이다. Master latch가 활성화되어있는 동안 glitch로 잠깐 입력값이 생기면 이 값이 Master latch에 저장되어있다가 다음 clock이 0이 되는 순간에 Slave latch로 전파되는 문제가 있다. 이는 clock이 1인 동안 계속 입력을 받기 때문에 생기는 문제이므로, clock이 0에서 1로, 혹은 1에서 0으로 바뀌는 순간에만 입력을 받는 Edge-trigger 회로를 사용하여 해결할 수 있다.

Master/Slave FF의 회로도는 다음과 같다.



3. 실험 준비

1) ALU

ALU에서 산술 장치(Arithmetic Unit)와 논리 장치(Logit Unit)의 1bit에 대한 진리표는 다음과 같다.

S_2	S_1	Y_i	A_i
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

표 1. 산술 장치의 진리표

S_1	S_0	Output
0	0	$G_i = X_iY_i$
0	1	$G_i = X_i + Y_i$
1	0	$G_i = X_i \oplus Y_i$
1	1	$G_i = X_i'$

표 2. 논리 장치의 진리표

ALU 구현 방법은 AU와 LU를 따로 구현한 후, 합치는 방식을 사용한다. 진리표를 바탕으로 한 회로도는 다음과 같다.

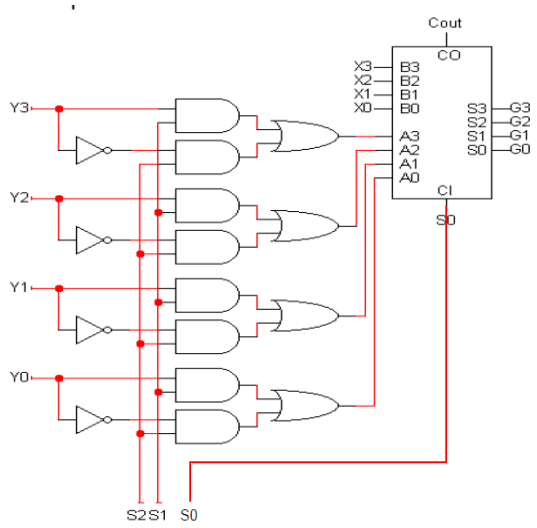


그림 1. 산술 장치의 회로도

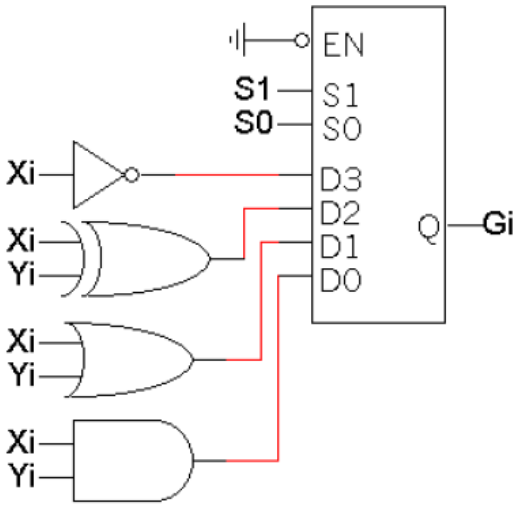


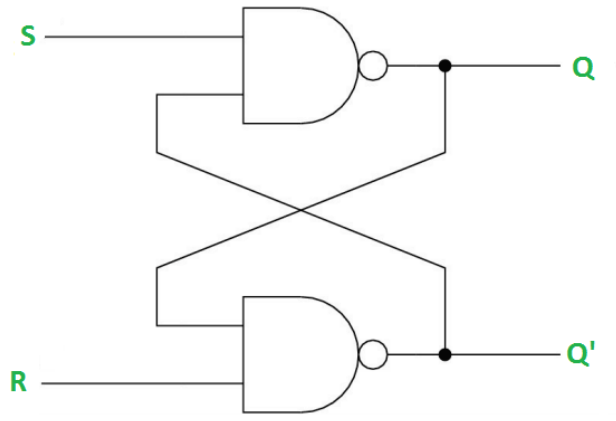
그림 2. 논리 장치의 회로도

AU와 LU를 합치기 위해서는 Quad-2-to-1-MUX가 필요한데, Quad-2-to-1-MUX의 진리표는 다음과 같다.

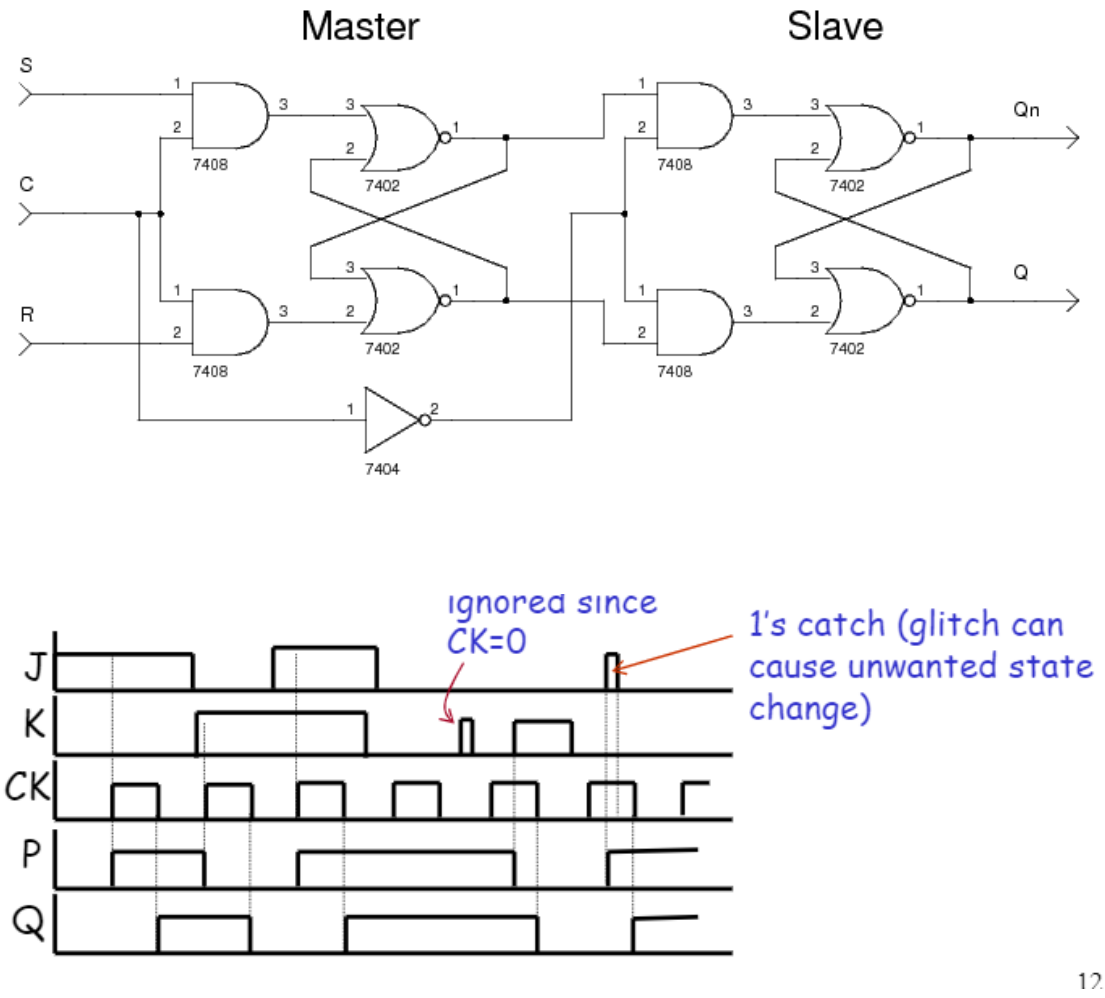
A	B	S	G
0	0	0	A
0	1	0	
1	0	0	
1	1	0	
0	0	1	B
0	1	1	
1	0	1	
1	1	1	

2) Master-slave JK 플립플롭

NOR gate를 이용하여 SR latch를 구현하고, 이를 이용해 Master/Slave FF를 구현하였다. NOR gate를 이용한 SR latch는 다음과 같다.



reset을 추가하여 SR latch를 이용한 Negative reset Master Slave JK 플립플롭의 회로도는 다음과 같다.

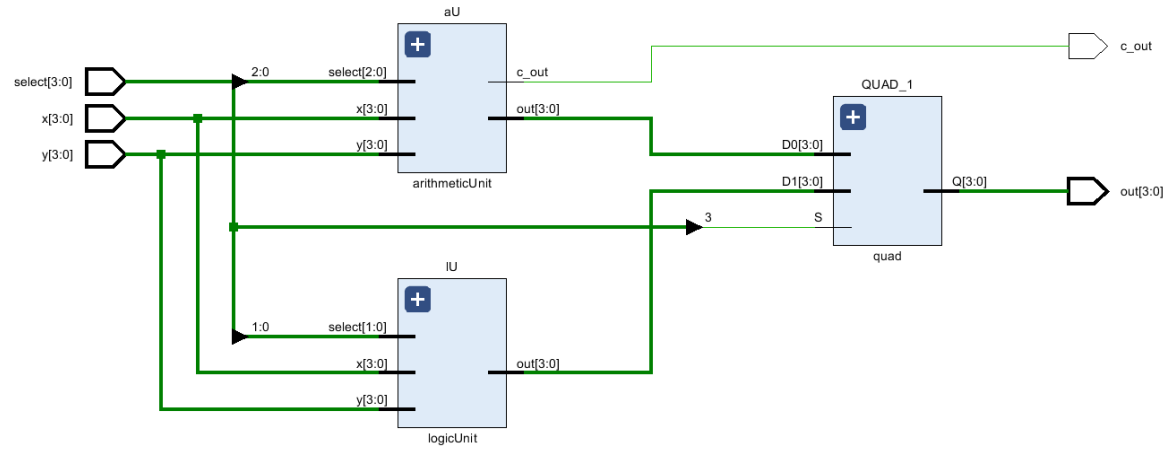


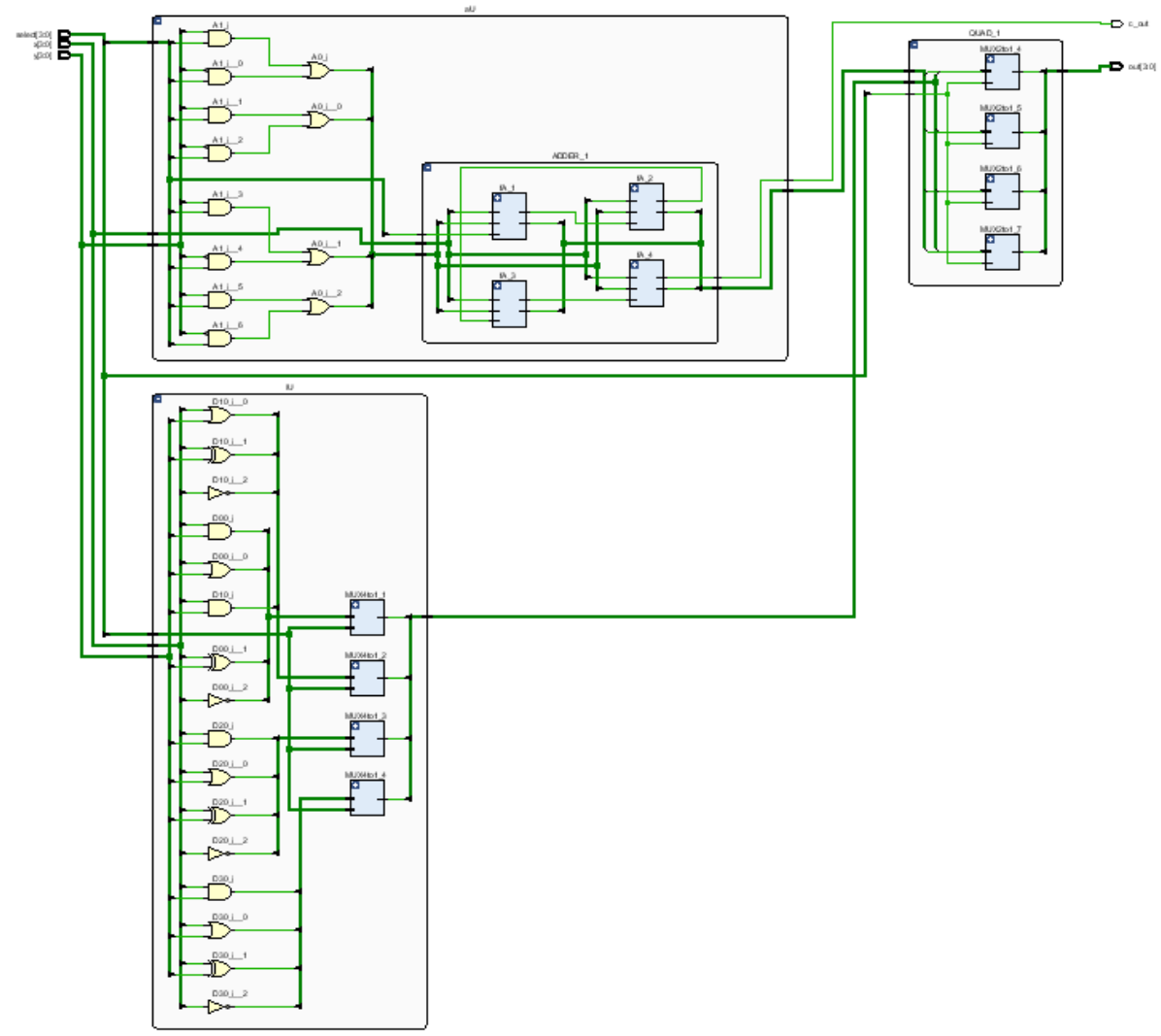
위의 강의자료를 참고하여 해결 가능한 glitch와 해결할 수 없는 glitch를 예상해볼 수 있다. clock이 0일 때는 해결 가능한 glitch이지만 clock이 1일 때는 해결할 수 없는 glitch가 발생한다.

4. 결과

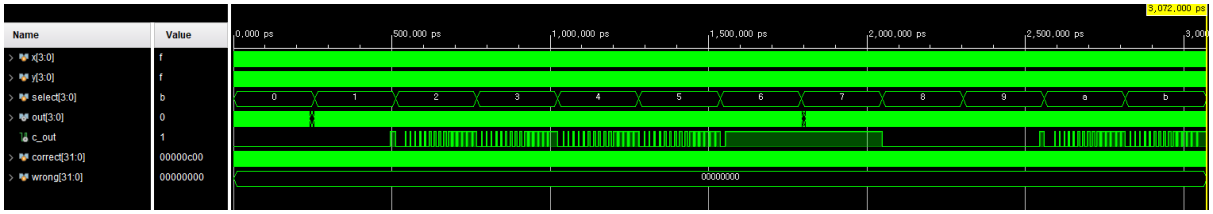
1) ALU

Schematic은 다음과 같다.


aU
select[3:0]
x[3:0]
y[3:0]
2:0
select[2:0]
x[3:0]
y[3:0]
c_out
out[3:0]
arithmeticUnit
c_out
QUAD_1
D0[3:0]
D1[3:0]
S
Q[3:0]
out[3:0]
quad
3
lU
1:0
select[1:0]
x[3:0]
y[3:0]
out[3:0]
logicUnit

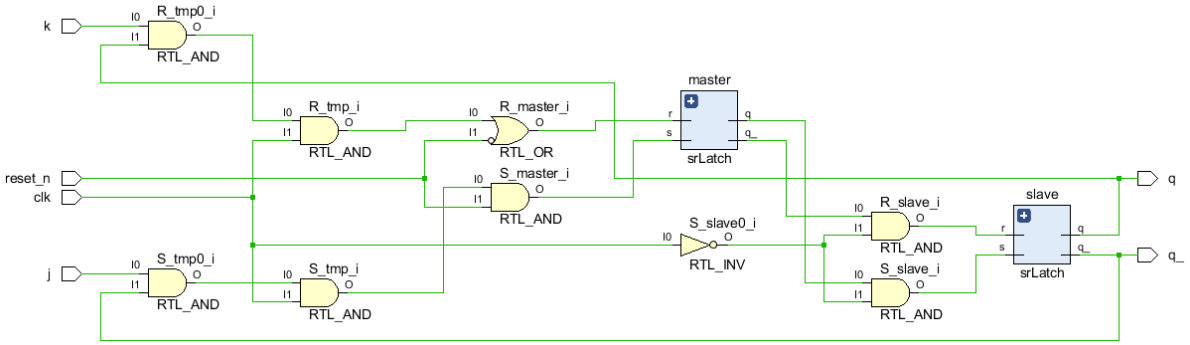


test-bench 결과는 다음과 같다.

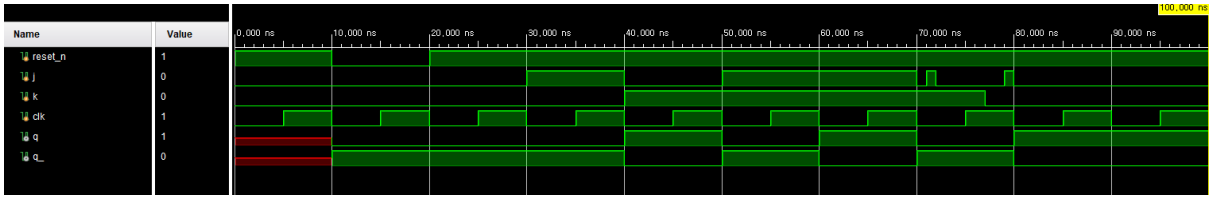


2) Master-slave JK 플립플롭

schematic은 다음과 같다.



test bench 결과는 다음과 같다. 직접 glitch를 구현하여 Master-slave JK 플립플록에서 해결한 glitch와 해결하지 못한 glitch를 보였다. clock이 1일 때는 해결할 수 없는 glitch가 2개 발생함을 볼 수 있다.



5. 논의

현재 컴퓨터구조 수업을 들으면서 CPU를 구현하는 과정에서 ALU가 실제로 중요한 역할로 작동함을 느끼고 있었는데, 디지털시스템설계 수업에서 설계한 ALU는 다른 느낌이었다. 또한 Test bench를 직접 구현하는 과정이 어려웠던 것 같다.