Lab 5. ALU 와 JK Flip-Flop

Digital System Design (CSED273)

20220100 박기현

1. 개요

이번 실험은 컴퓨터의 기초가 되는 산술 논리 장치(Arithmetic Logic Unit, ALU)와 정보 저장이 가능한 JK Flip-Flop 을 구현하는 것을 목표로 한다.

첫 번째 실험에서는 Arithmetic Unit 과 Logic Unit 모듈을 각각 구현하고, 이를 사용하여 ALU 를 완성한다.
두 번째 실험에서는 SR Latch 모듈을 구현하여 먼저 두 개의 SR Latch 를 단순히 이어 만든 Flip-Flop 을 구현하고, 그 다음 특정 구성을 변경하여 Negative reset master-slave JK Flip-Flop 을 구현한다.

2. 이론적 배경

1) 산술 논리 장치(Arithmetic Logic Unit, ALU)

ALU 는 입력에 대하여 여러 산술(Arithmetic) 연산과 논리(Logic) 연산을 수행하는 장치이다. 산술 장치에서는 사칙 연산 등을 수행하고, 논리 장치에서는 Bitwise 논리 연산 등을 수행한다.

본 실험에서는 산술 장치에서 transfer, increment, add, add increment, 1s complement subtract, 2s complement subtract, decrement, decrement transfer 를 수행하고, 논리 장치에서 and, or, xor, not 을 수행한다.

2) 비동기 회로(asynchronous circuit) / 동기 회로(synchronous circuit)

모든 조합 회로와 클럭(Clock) 신호를 따르지 않는 순차 회로를 비동기 회로(asynchronous circuit)라고 한다. 반면 동기화되어, 즉 다른 회로와 같은 순간에 맞추어 작동하기 위해 클럭 신호를 따르는 순차 회로를 동기 회로(synchronous circuit)라고 한다.

3) SR Latch

SR Latch 에서 S 는 Set 을, R 을 Reset 을 의미한다. Latch 의 출력값인 Q 와 Q'는 서로 complement 관계이므로, Q = 1 이면 Q' = 0, Q = 0 이면 Q' = 1 이 된다.

NOR 게이트로 구성된 SR Latch 는 S = 1, R = 0 또는 S = 0, R = 1 일 때 출력값이 변화하며, S = 0, R = 0 일 때 이전 상태를 유지한다. S = 1, R = 1 이 입력되면 출력값이 모두 0 이 되므로 Q 와 Q'가 complement 관계라는 전제에 모순이 발생한다. 따라서 forbidden, 즉 오류이다.

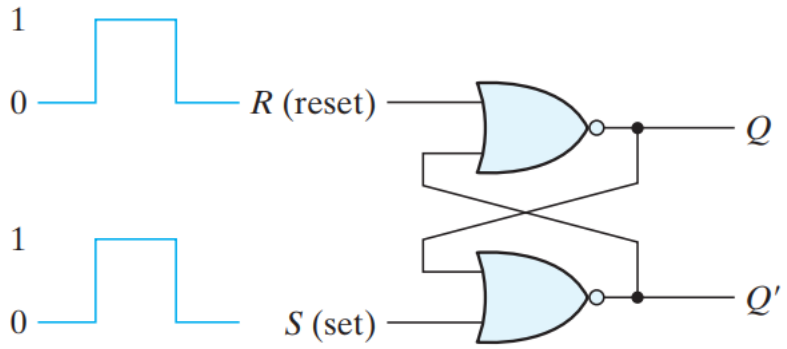


그림 1 SR Latch with NOR gates

Function table 로 나타내면 다음과 같다.

S	R	Q	Q'	
1	0	1	0	
0	0	1	0	After S = 1, R = 0
0	1	0	1	
0	0	0	1	After S = 0, R = 1
1	1	0	0	forbidden

NAND 게이트로 구성된 SR Latch 는 S = 0, R = 1 또는 S = 1, R = 0 일 때 출력값이 변화하며, S = 1, R = 1 일 때 이전 상태를 유지한다. S = 0, R = 0 이 입력되면 출력값이 모두 1 이 되므로 Q 와 Q'가 complement 관계라는 전제에 모순이 발생한다. 따라서 forbidden, 즉 오류이다.

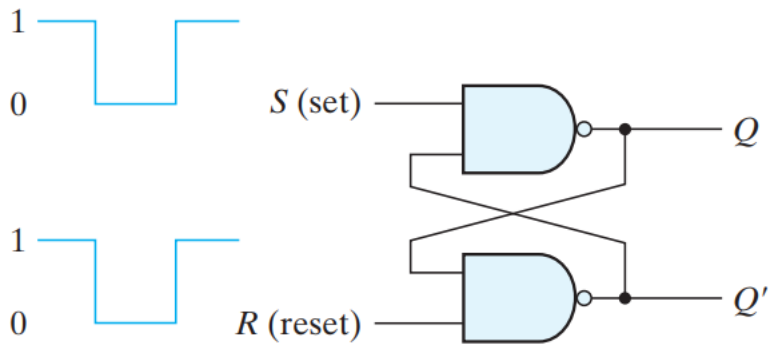


그림 2 SR Latch with NAND gates

Function table 로 나타내면 다음과 같다.

S	R	Q	Q′	
1	0	0	1	
1	1	0	1	After S = 1, R = 0
0	1	1	0	
1	1	1	0	After S = 0, R = 1
0	0	1	1	forbidden

4) JK Latch / JK Flip-Flop

JK Latch 는 SR Latch 에 추가적인 회로를 더해 S 와 R 이 동시에 1 인 상황에서도 정상적으로 작동하도록 수정한 Latch 이다.

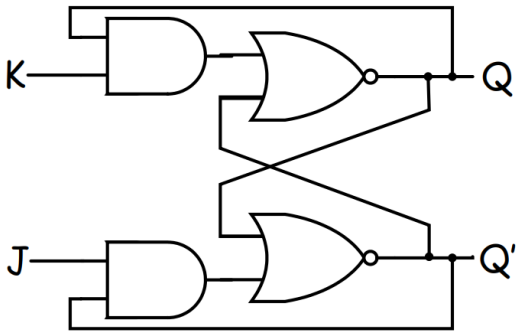


그림 3 JK Latch

Latch 는 입력이 바뀔 때 출력도 즉시 바뀌는 비동기 회로라면, Flip-Flop 은 입력이 바뀌더라도 출력이 클럭에 맞추어 반영되는 동기 회로이다. 즉 JK Flip-Flop 은 클럭 신호를 추가로 받아 이에 맞추어 작동한다.

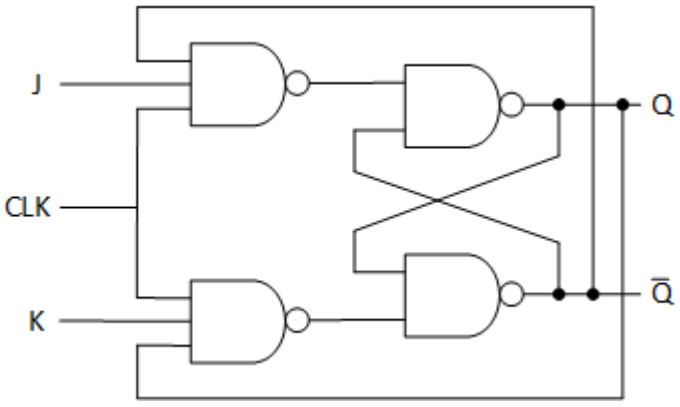


그림 4 JK Flip-Flop

5) Master-Slave JK Flip-Flop

Master-Slave JK Flip-Flop 은 두 개의 SR Latch 를 연결하여 만든 Flip-Flop 이다. 클럭이 1 인 동안 Master Latch 를 활성화하여 입력을 임시로 저장한 뒤, 클럭이 0 이 되는 순간 Slave Latch 를 활성화한다. 따라서 Master Latch 가 활성화되어 있는 동안 glitch 로 잠깐 입력값이 생기면 이 값이 Master Latch 에 저장되어 있다가 다음 클럭이 0 이 되는 순간에 Slave Latch 로 전파되는 문제가 존재한다. 이는 클럭이 1 인 동안 계속 입력을 받기 때문에 생기는 문제로 클럭이 0 에서 1 로, 혹은 1 에서 0 으로 바뀌는 순간에만 입력을 받는 Edge-trigger 회로를 사용하여 해결할 수 있다.

본 실험에서 구현하는 Negative reset Master-Slave JK Flip-Flop 에서 reset_n 이 0 이 되면 출력값은 0 으로 초기화된다.

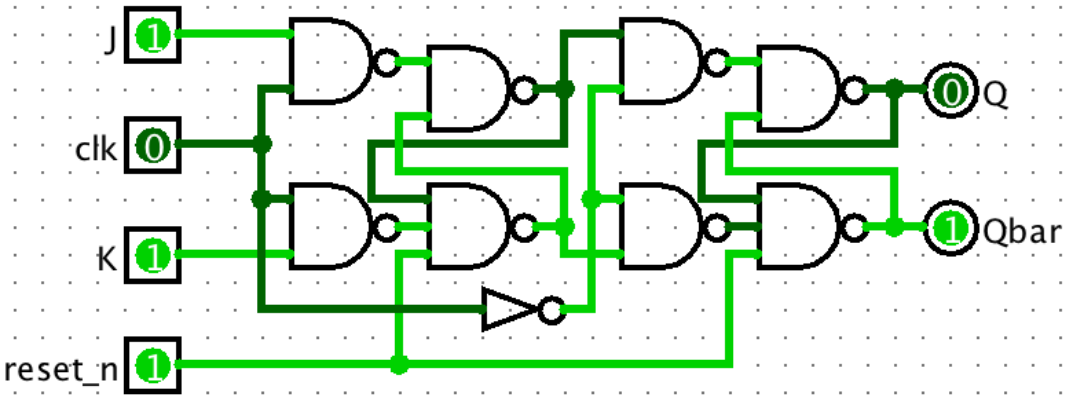


그림 5 Master-Slave SR Flip-Flop with two SR Latches

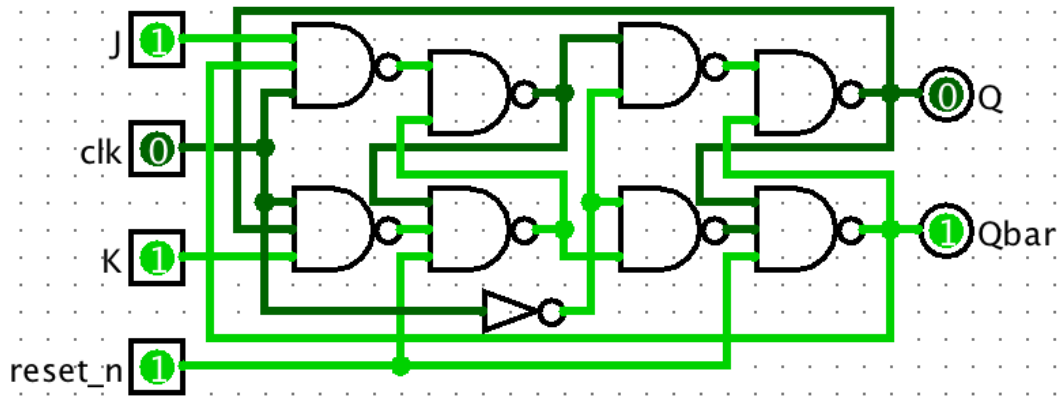


그림 6 Master-Slave JK Flop-Flop with two SR Latches

3. 실험 준비

1) ALU

ㄱ. 표 1 의 ALU 의 S_3 의 값에 따라 산술 장치와 논리 장치 두 부분으로 나누어 단순화하고 회로도를 그린다.

- 산술 장치를 구현할 때 adder 는 하나만 사용한다.

- 논리 장치를 작성할 때 4:1 MUX 를 사용한다.

	Select				동작	Adder 입력		
	S_3	S_2	S_1	S_0	$out = A + B + C_{in}$	A	B	C_{in}
산술 장치	0	0	0	0	x	0000	x	0
	0	0	0	1	$x+1$	0000	x	1
	0	0	1	0	$x+y$	y	x	0
	0	0	1	1	$x+y+1$	y	x	1
	0	1	0	0	$x+\overline{y}$	$\overline{y}$	x	0
	0	1	0	1	$x+\overline{y}+1$	$\overline{y}$	x	1
	0	1	1	0	$x-1$	1111	x	0
	0	1	1	1	x	1111	x	1
	Select				동작			
	S_3	S_2	S_1	S_0	out_i			
논리 장치	1	0	0	0	$x_i \ AND \ y_i$			
	1	0	0	1	$x_i \ OR \ y_i$			
	1	0	1	0	$x_i \ XOR \ y_i$			
	1	0	1	1	$\overline{x_i}$			

표 1 ALU

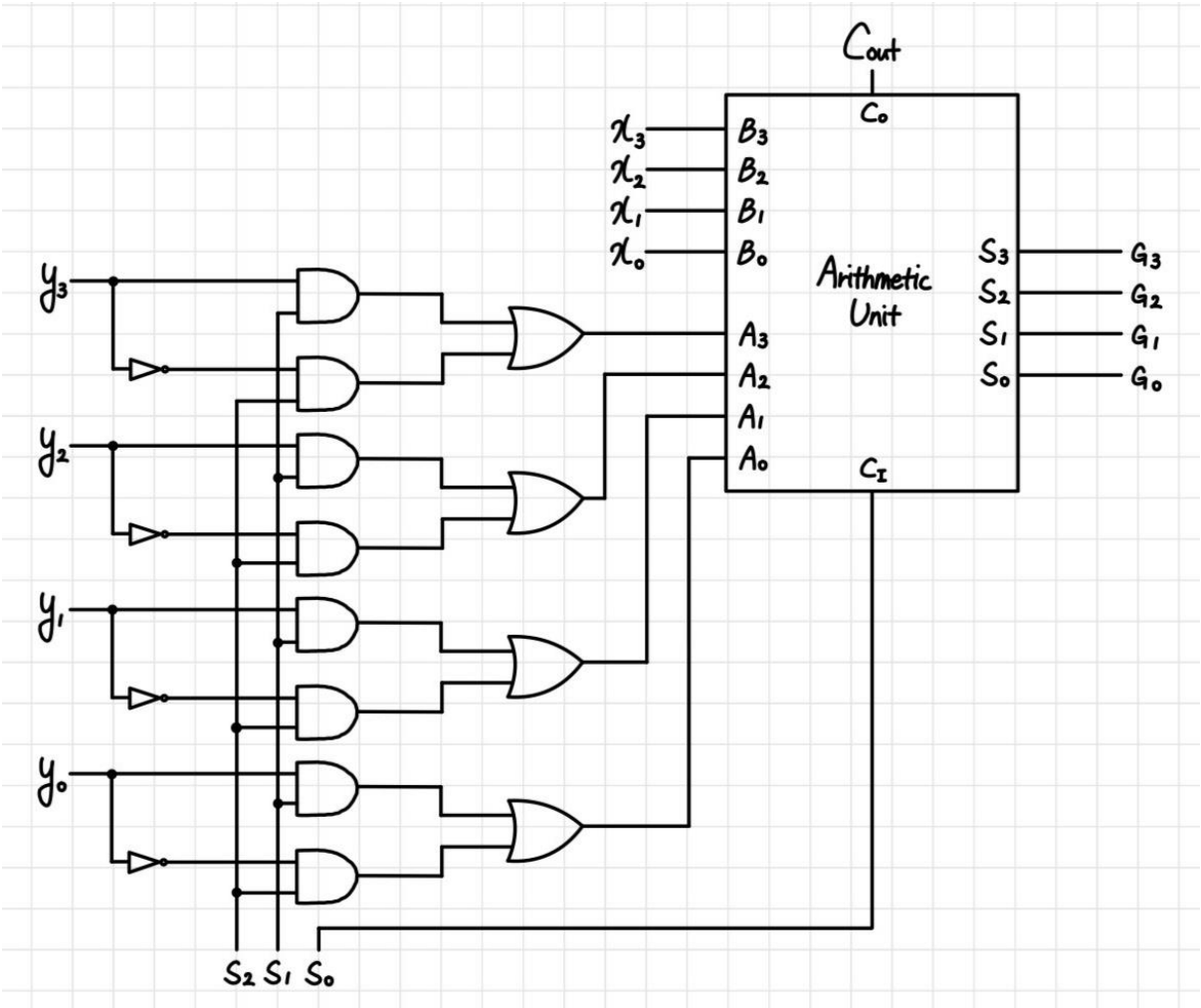
S_2	S_1	A
0	0	0000
0	1	y
1	0	y'
1	1	1111

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

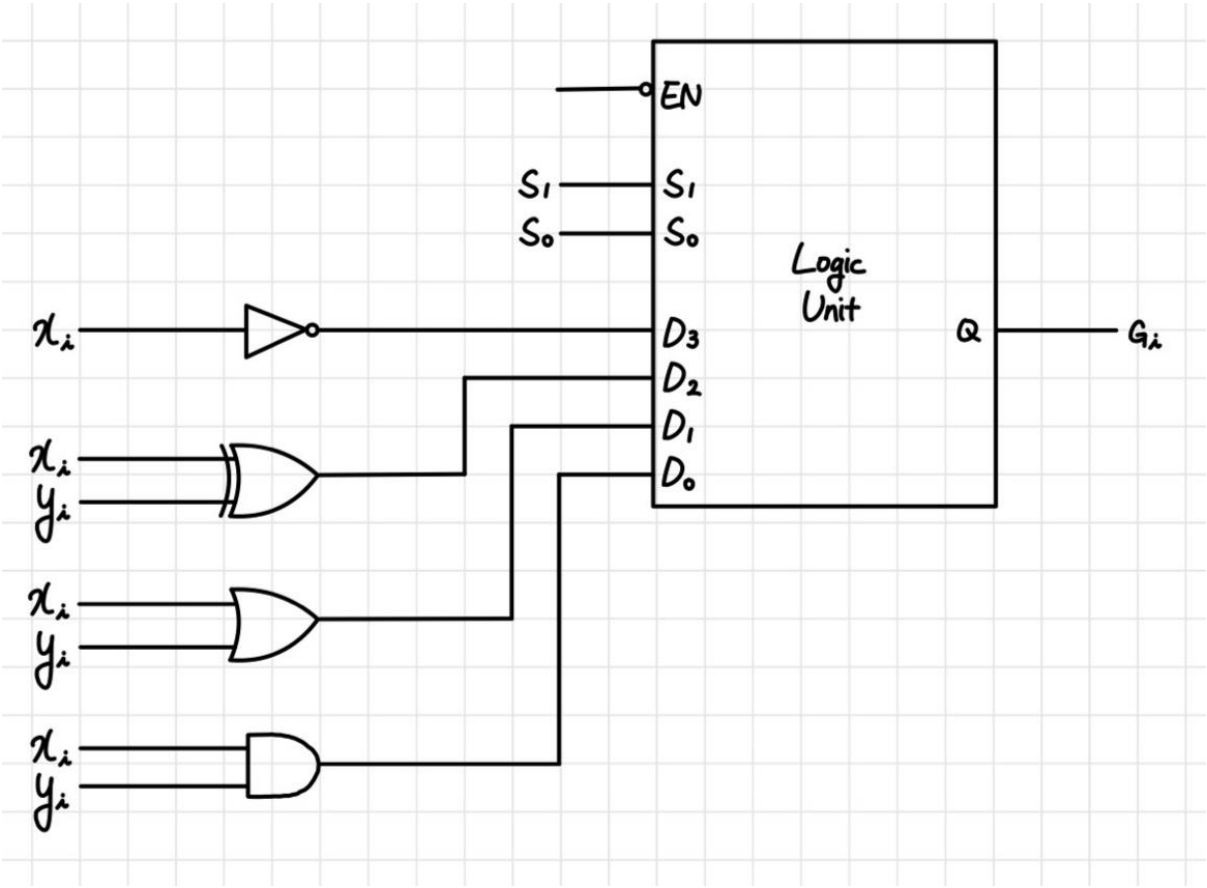
S_2 \ S_1y_i	00	01	11	10
0	0	0	1	0
1	1	0	1	1

$A_i = S_2y_i' + S_1y_i$



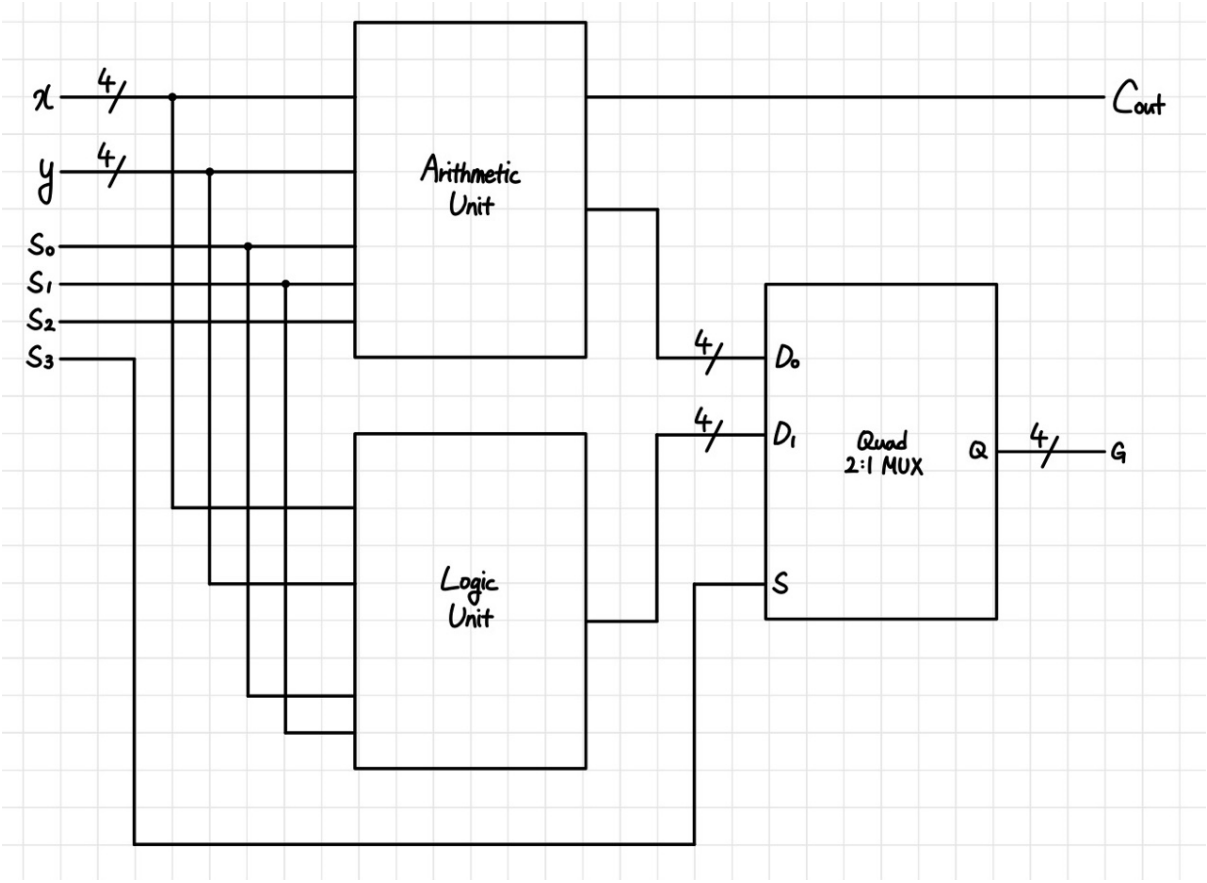
Arithmetic Unit 회로도

S_1	S_0	Output
0	0	$G_i = x_i y_i$
0	1	$G_i = x_i + y_i$
1	0	$G_i = x_i \oplus y_i$
1	1	$G_i = x_i'$



Logic Unit 회로도

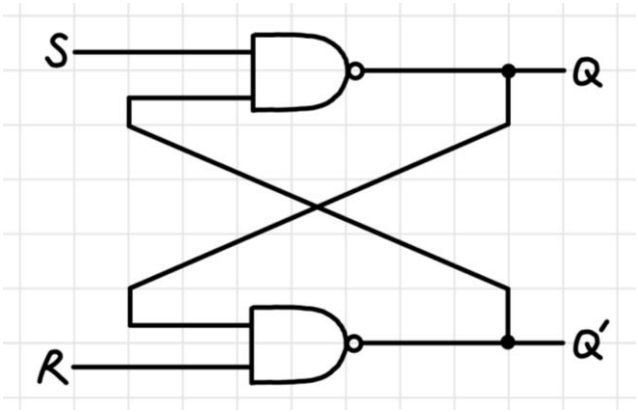
ㄴ. 두 모듈을 2:1 MUX 로 묶어 ALU 의 회로도를 그린다.



ALU 회로도

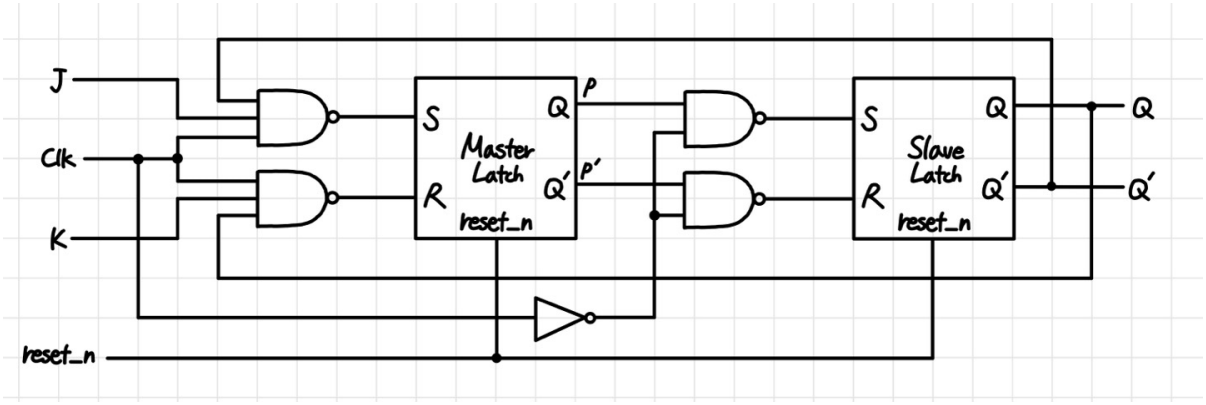
2) Master-Slave JK Flip-Flop

ㄱ. SR Latch 의 회로도를 그린다.



SR Latch 회로도

ㄴ. SR Latch 를 사용해 Negative reset Master-Slave JK Flip-Flop 의 회로도를 그린다.



Negative reset Master-Slave JK Flip-Flop 회로도

ㄷ. SR Latch 와 비교하여 Master-Slave JK Flip-Flop 이 해결 가능한 glitch 와 해결 불가능한 glitch 를 예상하고 분석한다.

SR Latch 는 NOR 게이트로 구성된 경우 두 입력값이 모두 1 이면, NAND 게이트로 구성된 경우 두 입력값이 모두 0 이면 무조건 forbidden 상태가 된다. 하지만 Master-Slave JK Flip-Flop 의 경우 클럭과 slave latch 의 출력값을 다시 입력부로 보내 추가적인 회로로 연산을 수행하여 master latch 의 입력값으로 보내므로 forbidden 상태로 되지 않고, 이어서 동작할 수 있게 할 것이라 생각한다. 하지만 Master-Slave JK Flip-Flop 의 경우 처음 입력 시에는 Q 와 Q'의 값이 불확실하므로 해결 못하는 부분도 존재할 것이라 생각한다. 또한, forbidden 상태가 되지 않도록 하는 것일뿐, 올바른 Q 의 값을 출력하는지도 알 수 없을 것이다.

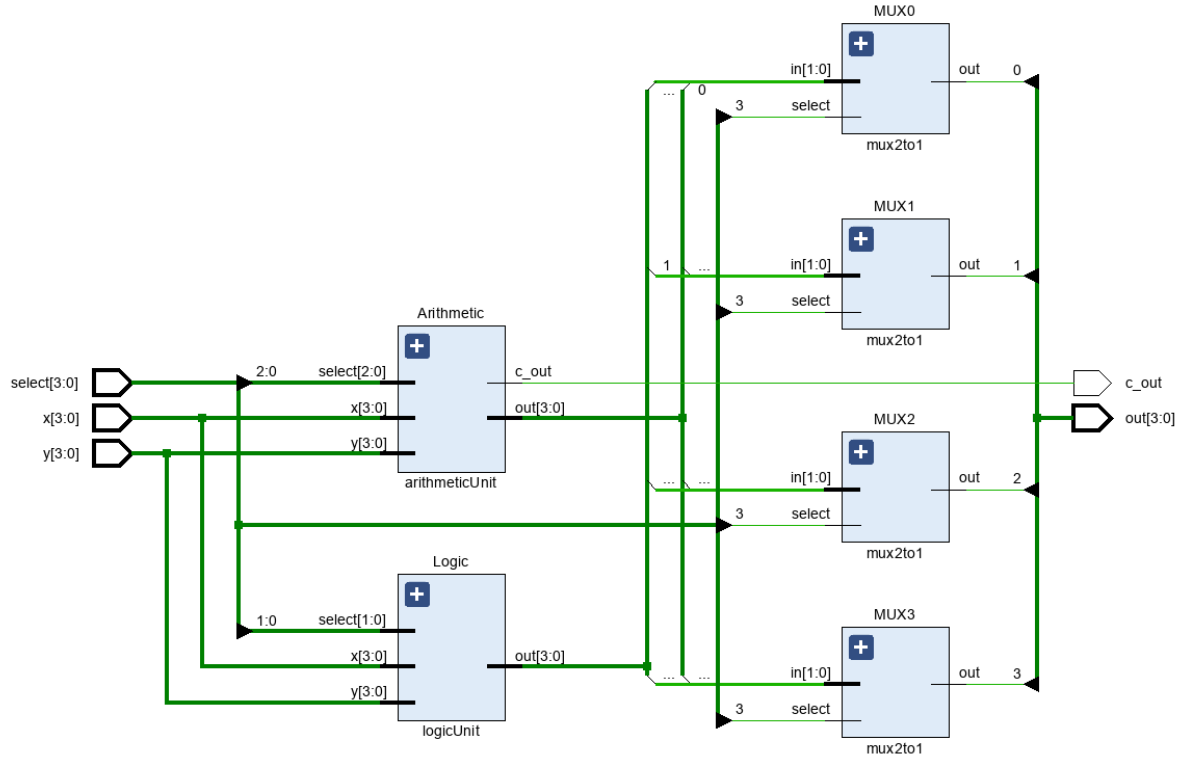
분석 결과는 4. 결과 에 작성한다.

4. 결과

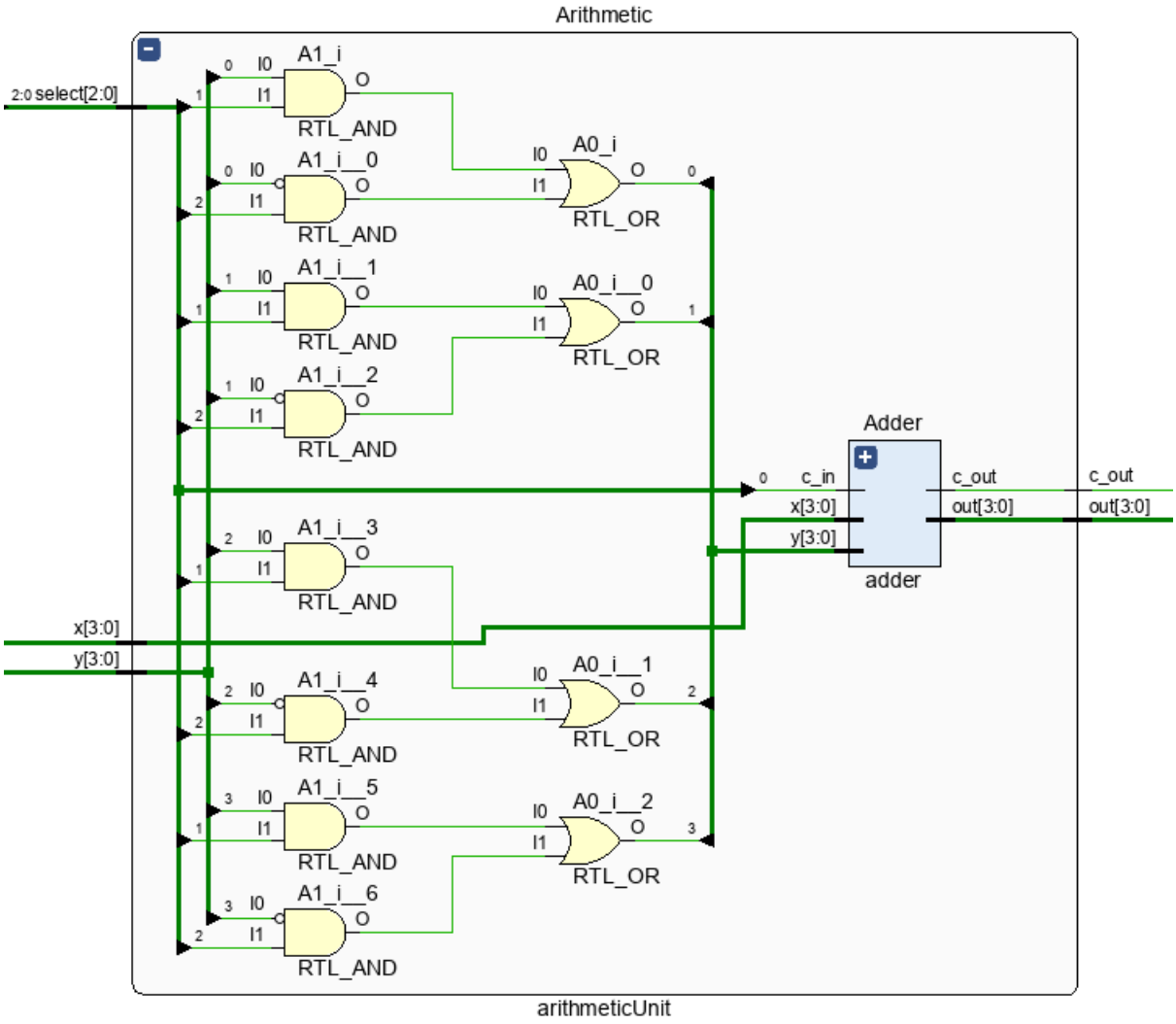
1) ALU

ㄱ. 산술 장치와 논리 장치 모듈을 구현하고 이를 사용하여 ALU 를 완성한다.

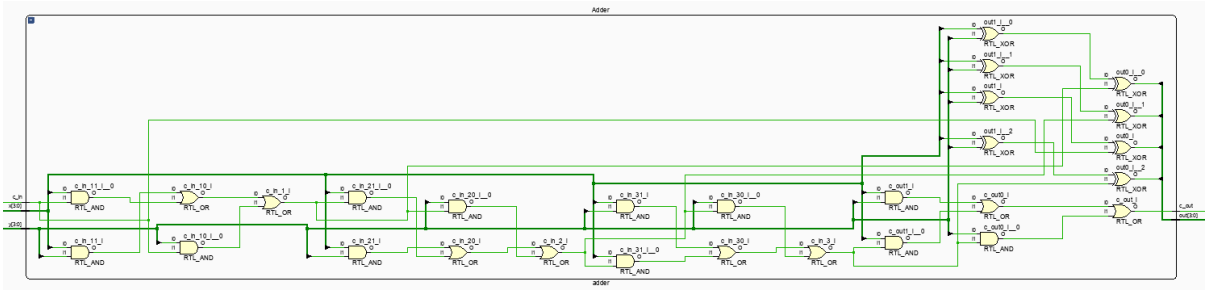
ㄴ. Schematic 기능으로 회로를 확인한다.


MUX0
in[1:0]
out
0
select
3
mux2to1
MUX1
in[1:0]
out
1
select
3
mux2to1
MUX2
in[1:0]
out
2
select
3
mux2to1
MUX3
in[1:0]
out
3
select
3
mux2to1
Arithmetic
2:0
select[2:0]
x[3:0]
y[3:0]
c_out
out[3:0]
arithmeticUnit
Logic
1:0
select[1:0]
x[3:0]
y[3:0]
out[3:0]
logicUnit
select[3:0]
x[3:0]
y[3:0]
c_out
out[3:0]

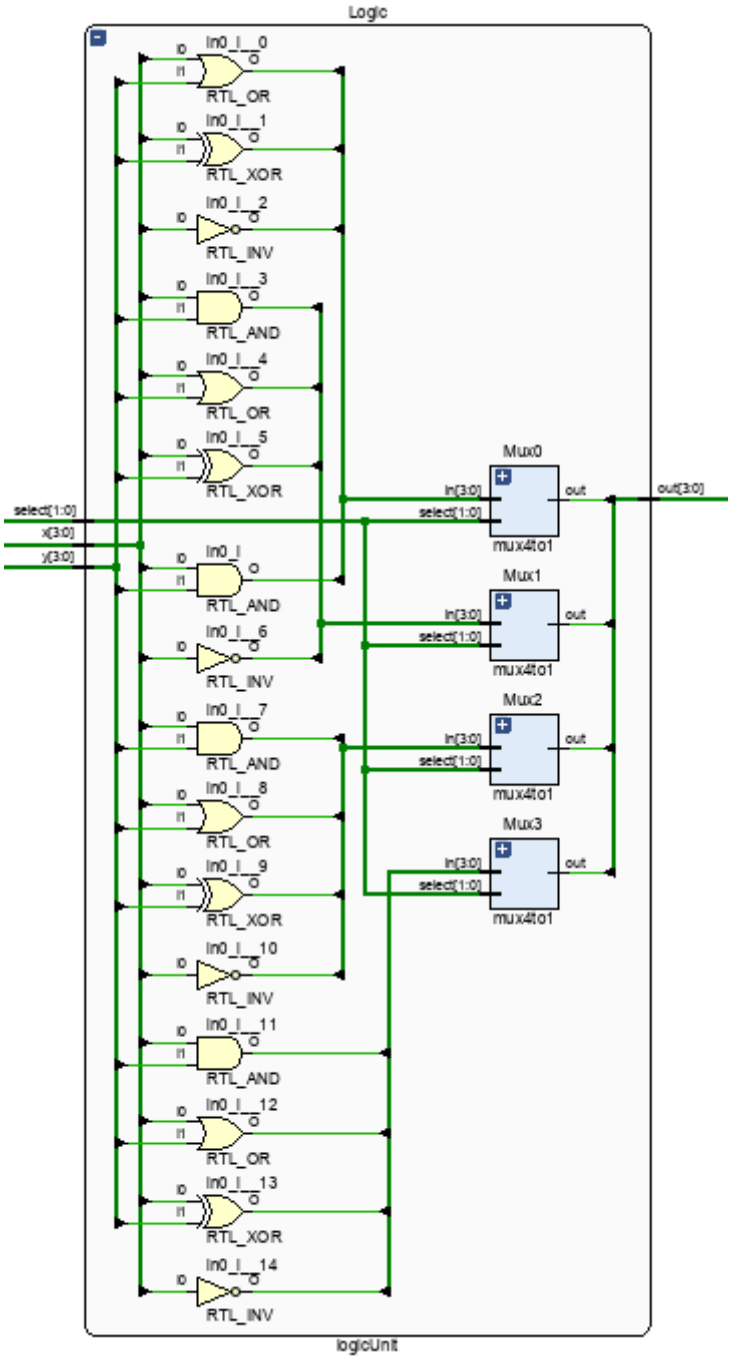
ALU



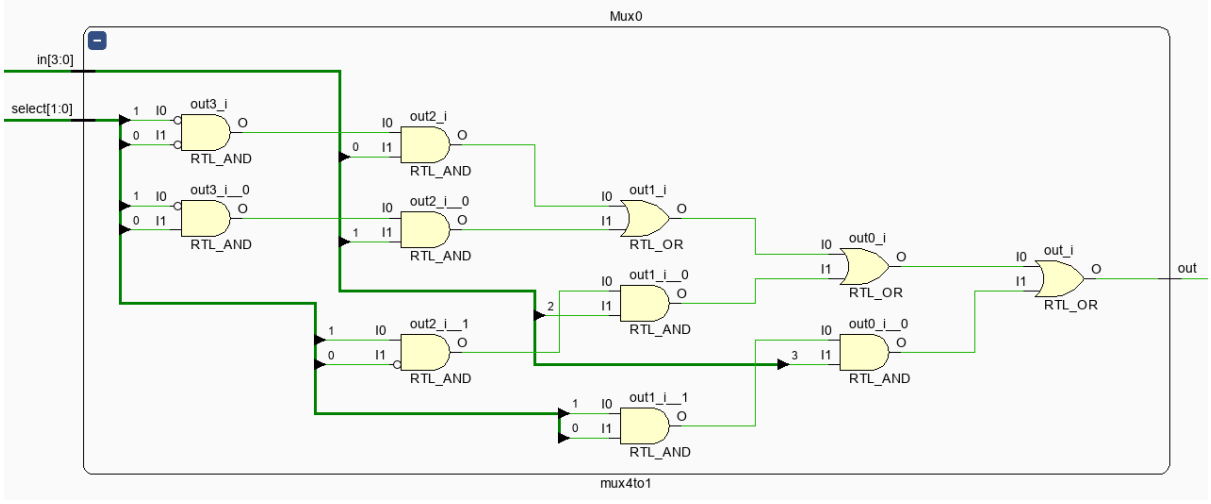
Arithmetic Unit



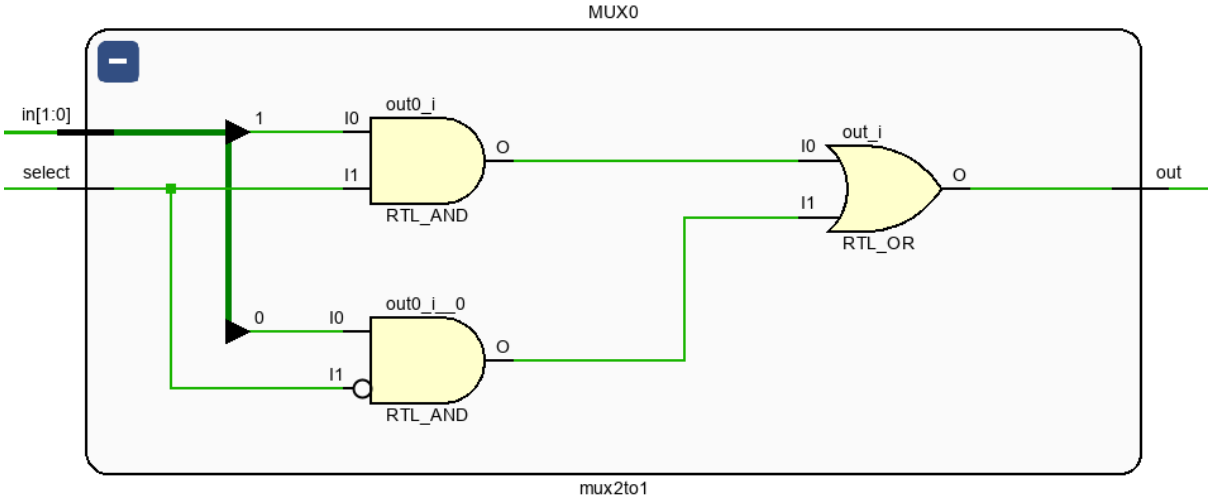
adder


Logic
In0_I__0
RTL_OR
In0_I__1
RTL_XOR
In0_I__2
RTL_INV
In0_I__3
RTL_AND
In0_I__4
RTL_OR
In0_I__5
RTL_XOR
In0_I
RTL_AND
In0_I__6
RTL_INV
In0_I__7
RTL_AND
In0_I__8
RTL_OR
In0_I__9
RTL_XOR
In0_I__10
RTL_INV
In0_I__11
RTL_AND
In0_I__12
RTL_OR
In0_I__13
RTL_XOR
In0_I__14
RTL_INV
select[1:0]
x[3:0]
y[3:0]
Mux0
Mux1
Mux2
Mux3
in[3:0]
select[1:0]
out
mux4to1
out[3:0]
logicUnit

Logic Unit

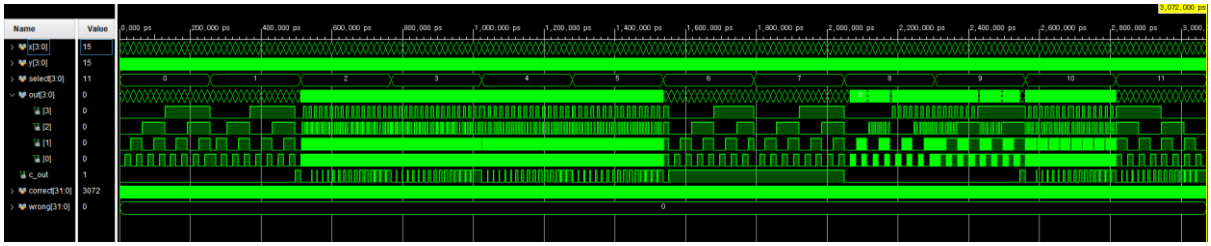


4:1 MUX



2:1 MUX

ㄷ. 테스트벤치를 완성하고 시뮬레이션을 실행하여 정상 작동을 확인한다.

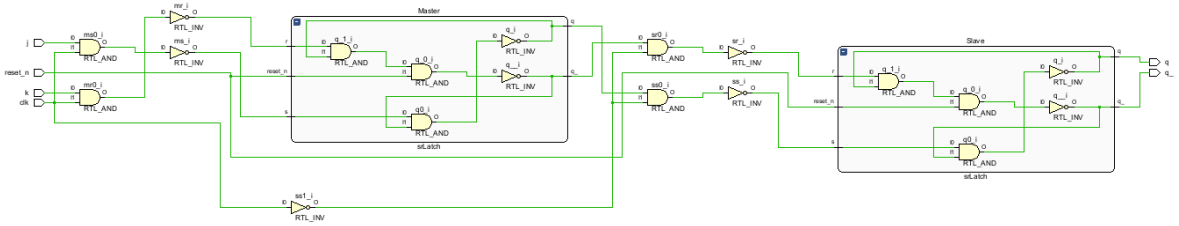


시뮬레이션 결과

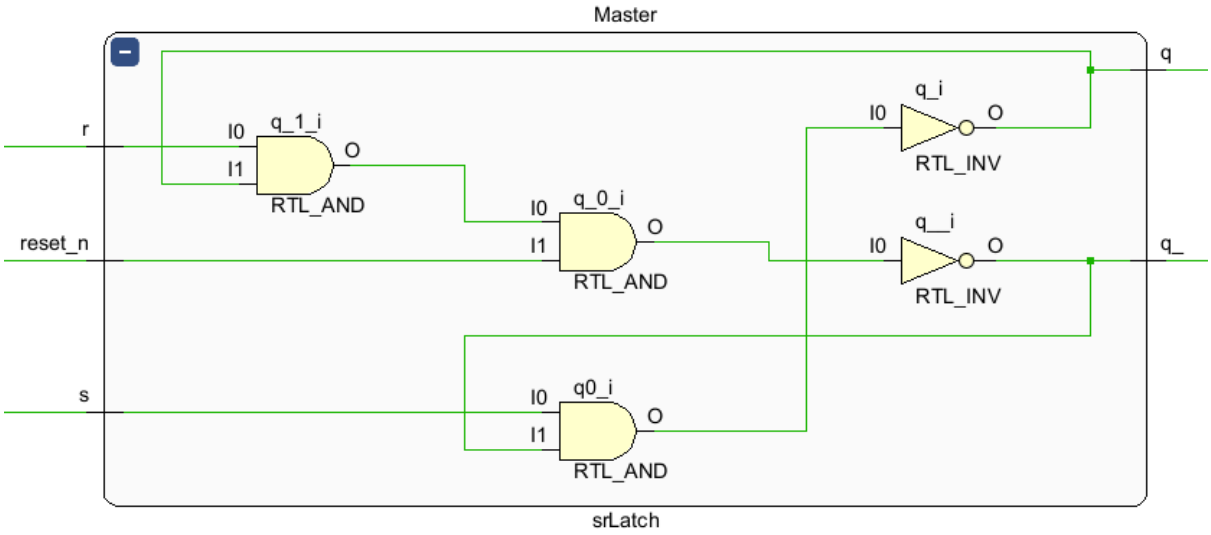
2) Master-Slave JK Flip-Flop

ㄱ. SR Latch 모듈을 구현하고 이를 사용하여 그림 5 와 그림 6 의 Flip-Flop 을 완성한다.

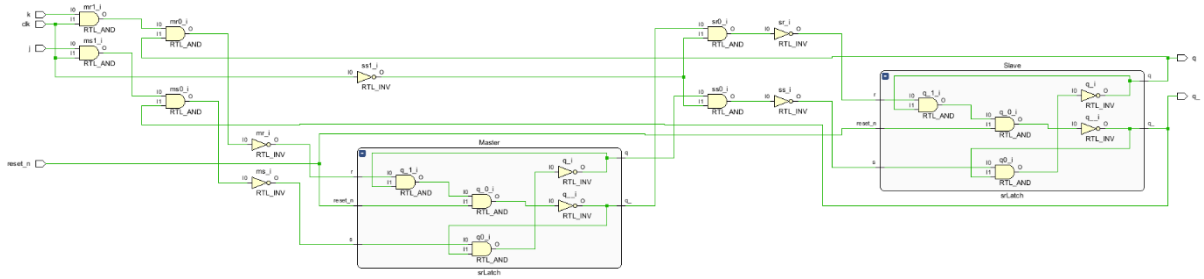
ㄴ. Schematic 기능으로 회로를 확인한다.



Negative reset Master-Slave SR Flip-Flop with SR Latch

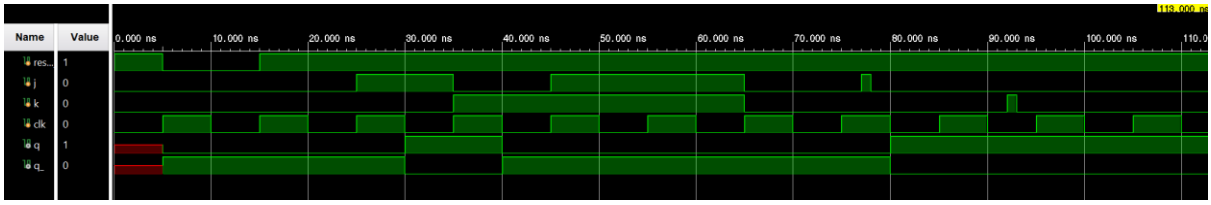


SR Latch

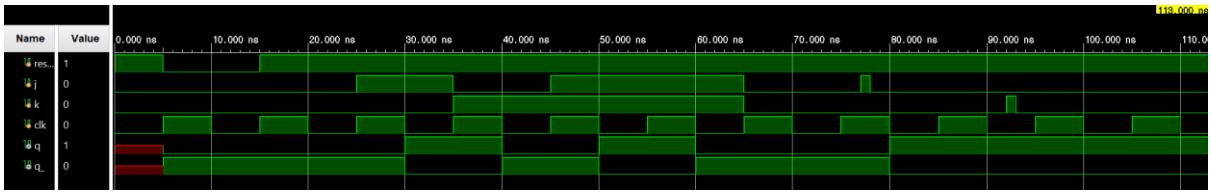


Negative reset Master-Slave JK Flip-Flop with SR Latch

ㄷ. 테스트벤치를 완성하고 시뮬레이션을 실행하여 주어진 조건에서 정상 작동을 확인한다.



module 1 시뮬레이션 결과

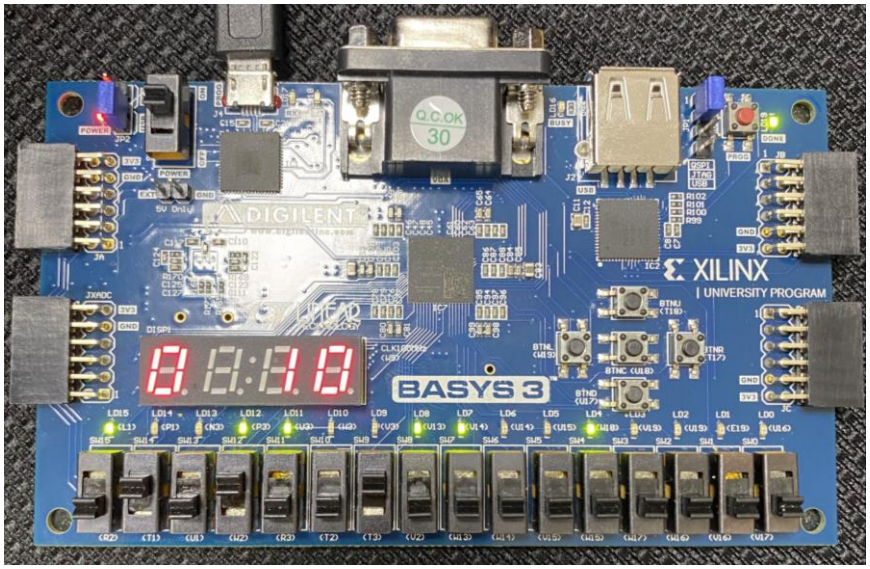


module 2 시뮬레이션 결과

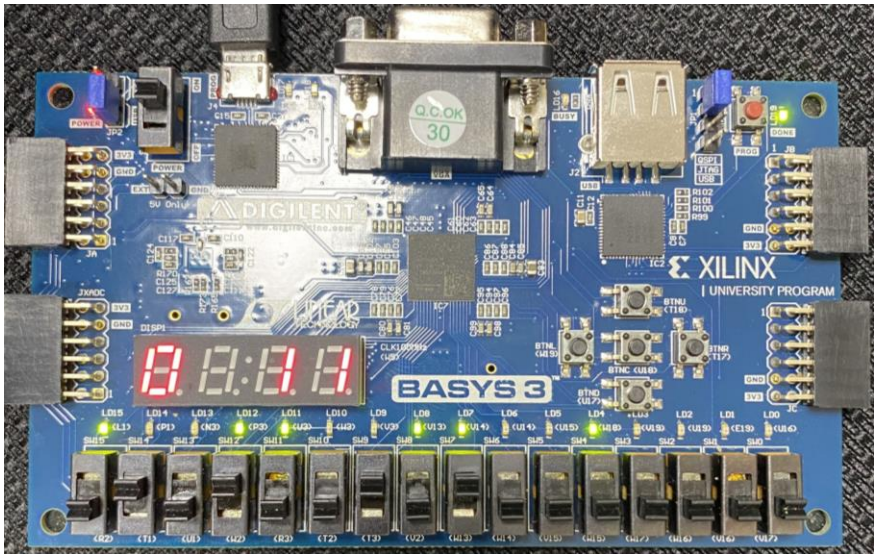
ㄹ. 테스트벤치를 수정해 SR Latch 와 첫 번째 모듈 대비 두 번째 모듈이 해결하는 glitch 를 분석한다.

SR Latch 는 NOR 게이트로 구성된 경우 두 입력값이 모두 1 이면, NAND 게이트로 구성된 경우 두 입력값이 모두 0 이면 무조건 forbidden 상태가 된다. 또한, 첫 번째 모듈의 경우에는 같은 상황에서 Q 의 값이 유지되는 경우도 있고, 반전되는 경우도 있는 것을 보아 출력이 불확실함을 확인할 수 있다. 반면 두 번째 모듈, 즉 Master-Slave JK Flip-Flop 의 경우 같은 상황에서 회로를 추가하여 slave latch 의 출력값을 입력값 j, k 와 함께 연산을 진행하고, 그 출력값을 master latch 의 입력값으로 보내 같은 상황에서도 forbidden 상태가 되는 것이 아니라 계속 동작할 수 있게 한다. 또한, 첫 번째 모듈과 다르게 같은 상황에서 Q 의 값을 toggling 함으로써 첫 번째 모듈에서 해결하지 못한 부분을 어느 정도 해결했다고 볼 수 있다. 하지만 toggle 되는 현상이 Q 의 옳은 출력값을 의미하는 것은 아니므로 완전히 해결했다고 볼 수 없다. 또한, 마찬가지로 처음 입력 시에 Q 의 값을 알 수 없으므로 정상적으로 출력되지 않음을 위 시뮬레이션 결과에서 확인할 수 있다.

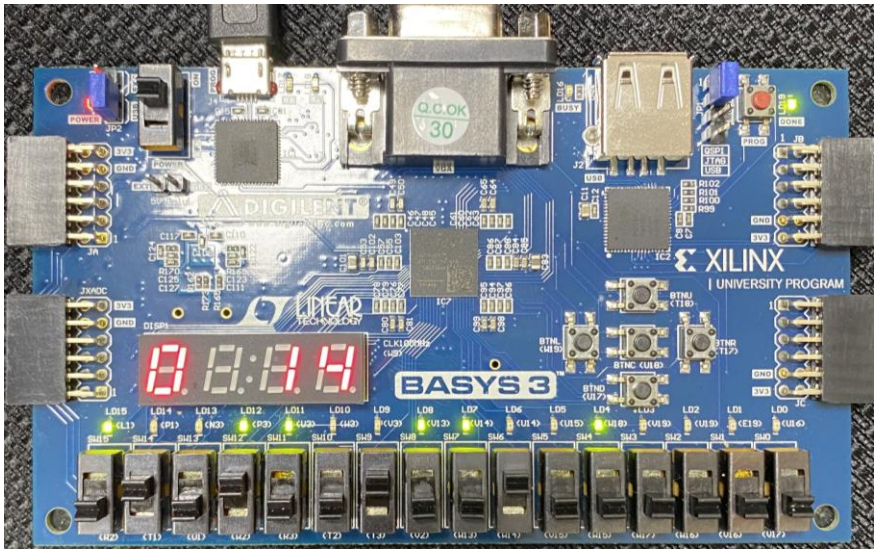
Select = 0 / x = 10 / y = 4



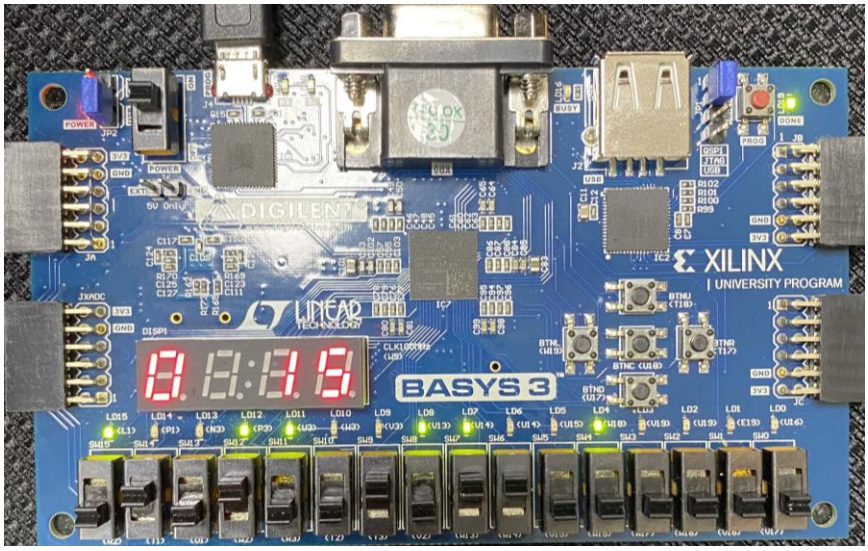
Select = 1 / x = 10 / y = 4



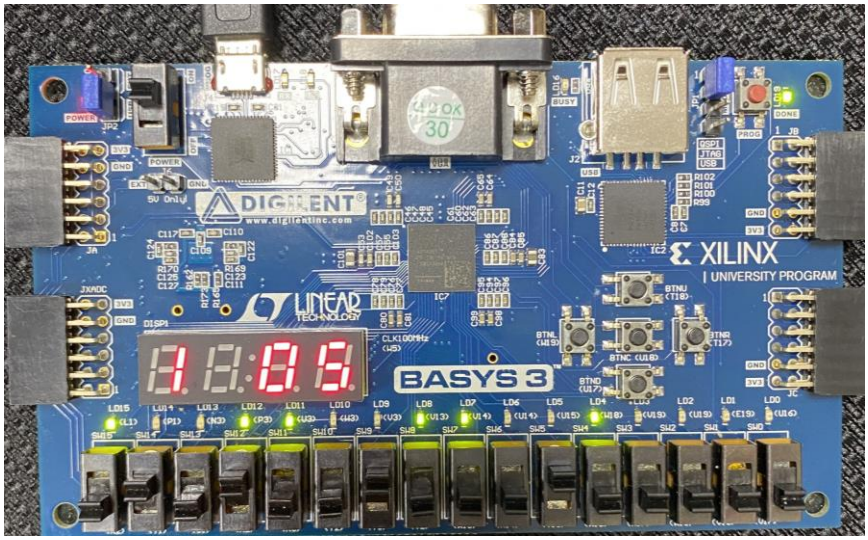
Select = 2 / x = 10 / y = 4



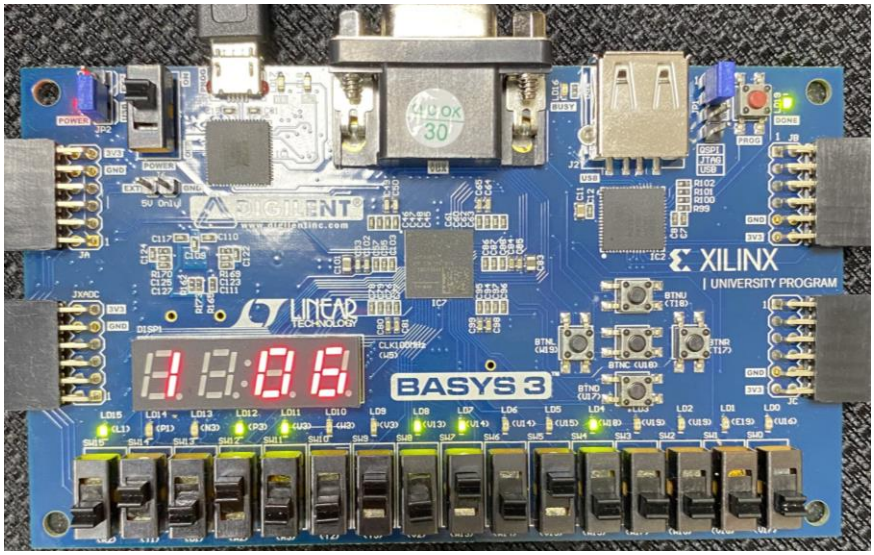
Select = 3 / x = 10 / y = 4



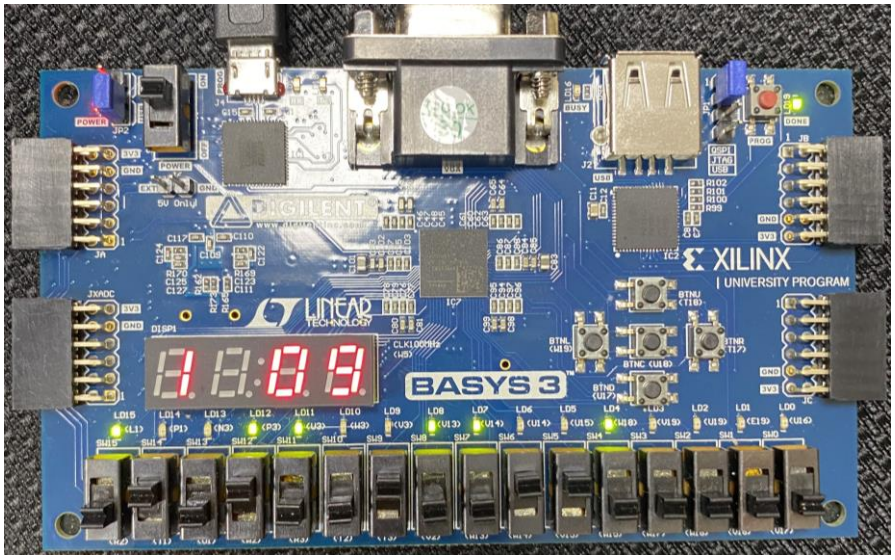
Select = 4 / x = 10 / y = 4



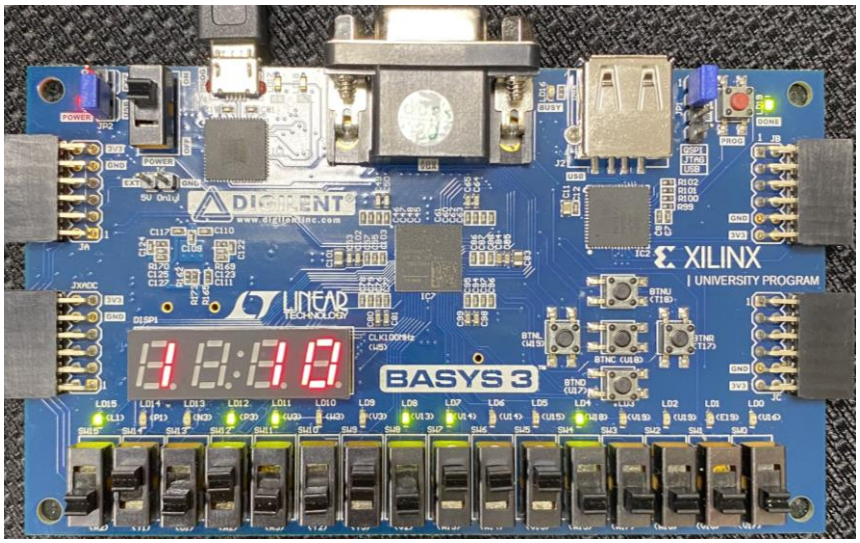
Select = 5 / x = 10 / y = 4



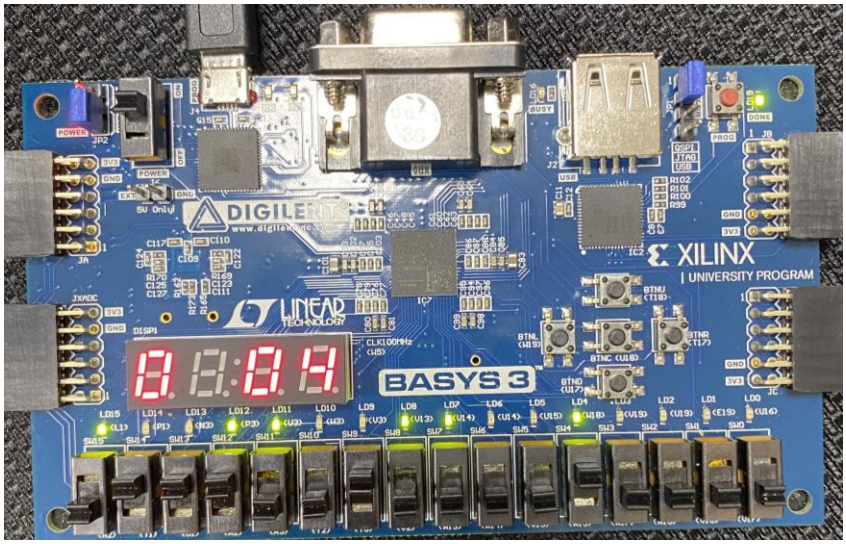
Select = 6 / x = 10 / y = 4



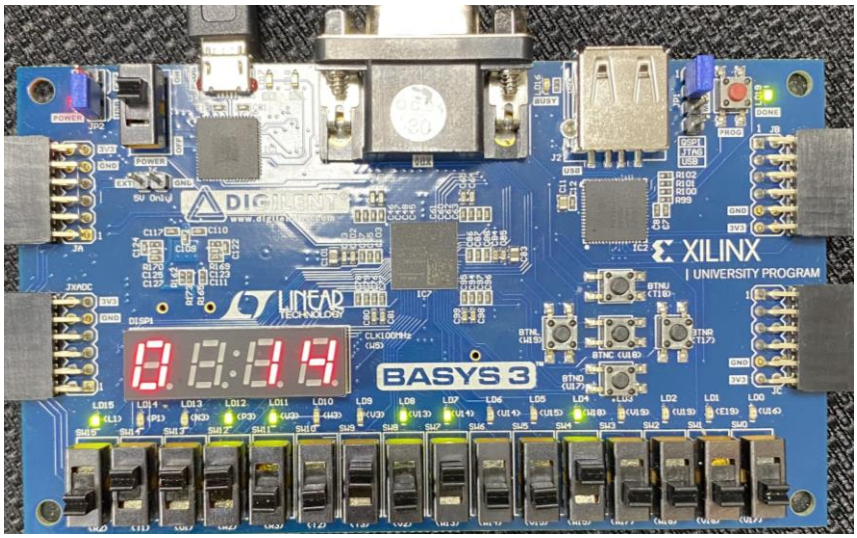
Select = 7 / x = 10 / y = 4



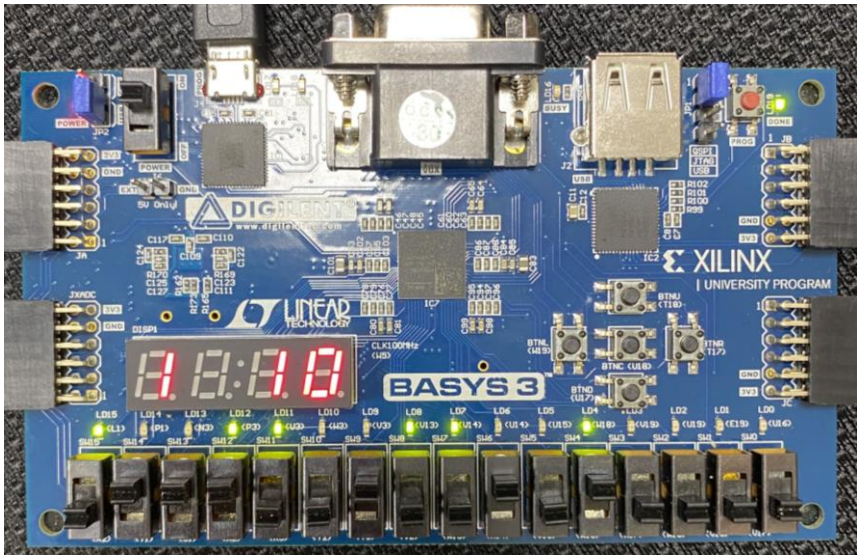
Select = 8 / x = 14 / y = 4



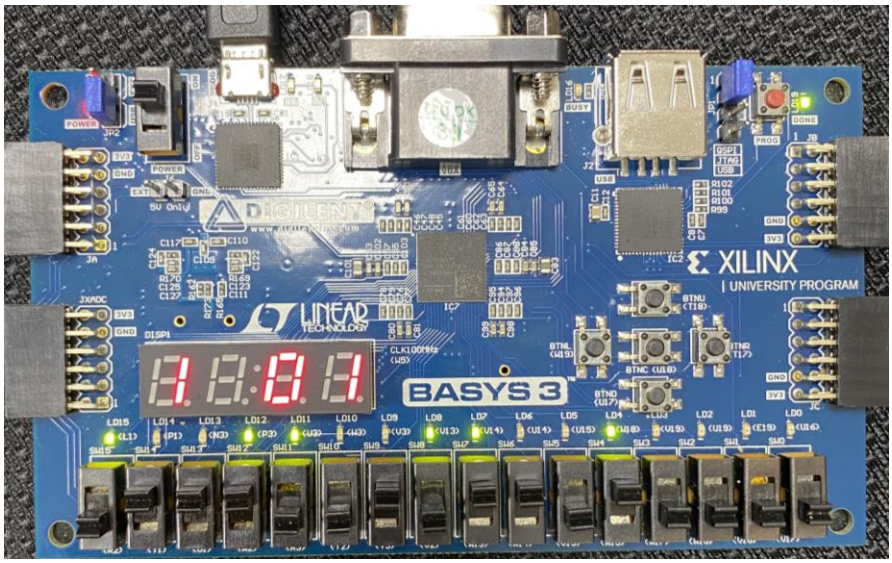
Select = 9 / x = 14 / y = 4



Select = 10 / x = 14 / y = 4



Select = 11/ x = 14 / y = 4



5. 논의

1) 느낀 점

이번 실험을 통해 Arithmetic Unit 과 Logic Unit 이 결합된 ALU 회로에 대해 완전히 이해할 수 있었다. ALU 라는 하나의 회로를 이용하여 오직 Selection 의 값으로 원하고자 하는 연산을 수행할 수 있다는 점이 인상 깊었고, 이번 FPGA 실습으로 직접 구현한 Verilog 코드를 적용해볼 수 있어서 학습하는 데 큰 도움이 되었던 실험이라고 생각한다.

2) 어려웠던 점 및 해결 방법

두 번째 실험을 하면서 어려움을 많이 겪었는데, 우선 negative reset 의 의미에 대해서 완전히 이해를 하지 못해 구현하는 것이 힘들었다. 하지만 조교님의 추가 설명 덕분에 잘 이해할 수 있었고, 원하는 회로를 구현할 수 있었다.

또한, Master-Slave SR Flip-Flop 과 Master-Slave JK Flip-Flop 의 차이를 완전히 이해하지 못해 시뮬레이션 결과를 해석하는 것에 어려움을 겪었다. 이 부분에 대해서는 수업 내용의 이해도가 아직 부족하다고 판단되어 SR Latch 와 JK Latch 의 차이점을 먼저 학습하고, Flip-Flop 의 역할을 이해한 뒤, 시뮬레이션 결과를 분석하니 각 상황별의 특징을 하나씩 이해할 수 있었다고 생각한다.

6. 출처 및 참고문헌

그림 1, 2 Digital Design, 6/E, Global Edition

그림 3 강의 Chapter 6 PPT

그림 4 https://blog.frankdecaire.com/2016/12/03/jk-flip-flop-simulator/