Lab5 report

김지현 (20220302)

1. 개요

- 산술 논리 장치인 ALU (arithmetic logic unit) 에서 arithmetic unit 과 logic unit 을 각각 구현한 뒤 MUX 를 이용해 장치를 구현했다. 그리고 SR latch 를 기반으로 Negative reset Master-Slave JK Flip Flop 을 구현했다.

2. 이론적 배경

1) ALU (Arithmetic Logic Unit)

   - ALU 는 산술 논리 연산 장치의 약자로, 컴퓨터 내부의 하드웨어 구성 요소 중 하나이다. 두개의 input 변수를 받아와 산술 논리 연산을 진행한다. 덧셈, 뺄셈, 곱셈, 나눗셈 등의 산술 연산을 하는 arithmetic unit 과 AND, OR, NOT 등을 연산하는 Logic unit 으로 나눠져있다. ALU 는 프로세서 내부의 register, 데이터를 저장하는데 사용되는 부분, 와 상호작용하여 데이터를 처리한다. 세부 방법으로는 register 에 저장되어있는 데이터를 가져와 연산을 수행한 다음 다시 결과를 register 에 저장하도록 한다.

2) Synchronous circuit / Asynchronous circuit

   - 동기회로와 비동기회로의 차이점은 clock 의 유무이다. 동기회로(Synchronous circuit)는 clock 신호에 의해 제어되는 회로로, 연산이나 데이터의 처리 등의 작업을 시간 신호에 맞춰서 수행하게 된다. 일반적으로 clock signal 은 고정된 주기로 발생하며, 이 신호의 변화에 따라 회로의 상태가 변하게 된다. 반면에 비동기회로(Asynchronous circuit)는 입력 신호의 변화에 따라 직접 제어되는 회로이다.

3) JK latch

   - JK 래치는 디지털 논리 회로 중 하나로, SR latch 의 문제점을 보완한 회로이다. SR 래치는 입력 값 S 와 R이 모두 1 일 때, 출력이 불확실했는데 이러한 문제점을 보완해 JK 래치가 사용되었다. J 와 K 가 모두 1 일 땐 이전 상태가 유지 되고, J 입력 신호가 1 이 되면 set 1 이 되고 K 입력 신호가 1 이 되면 reset to 0 가 된다. 마지막으로 SR 래치에서 문제점을 보였던 J 와 K 가 모두 1 이면 이전 상태와 반대로 동작하도록 한다.

4) JK Flip Flop

- JK FF 는 널리 사용되는 순차 논리 회로중 하나이다. 앞에서 설명한 JK latch 와 유사하게 동작하지만 래치와는 달리 clock 에 의해 제어된다.

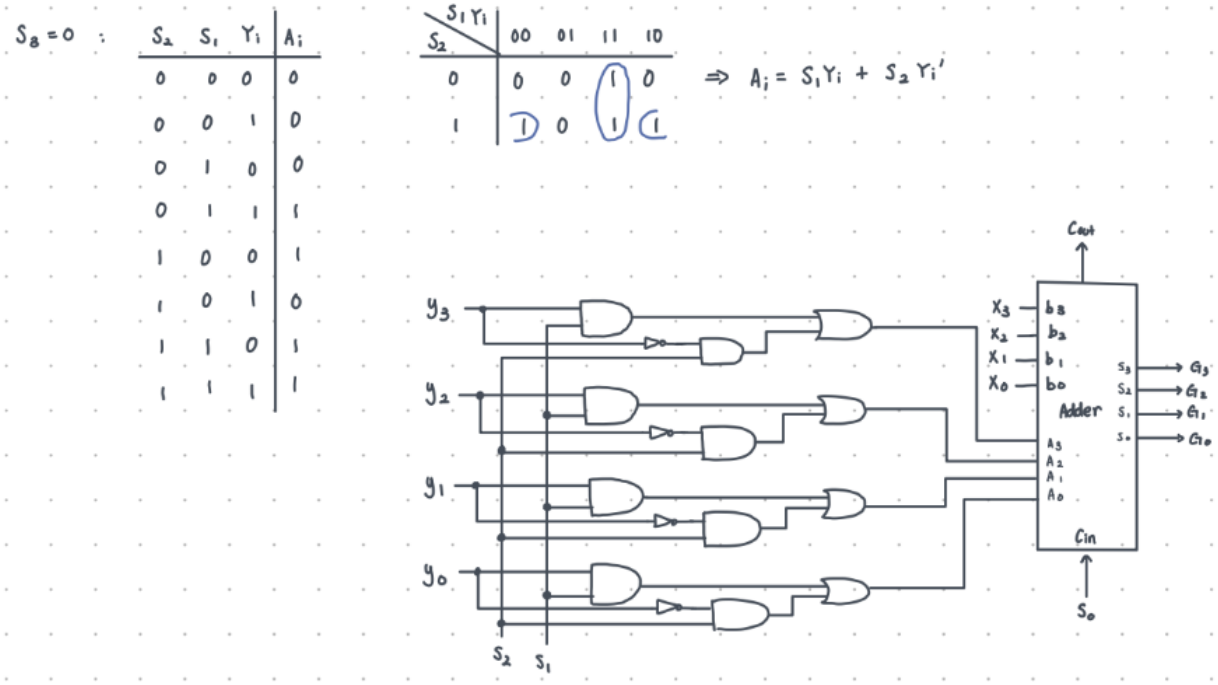
5) Master-Slave Flip Flop

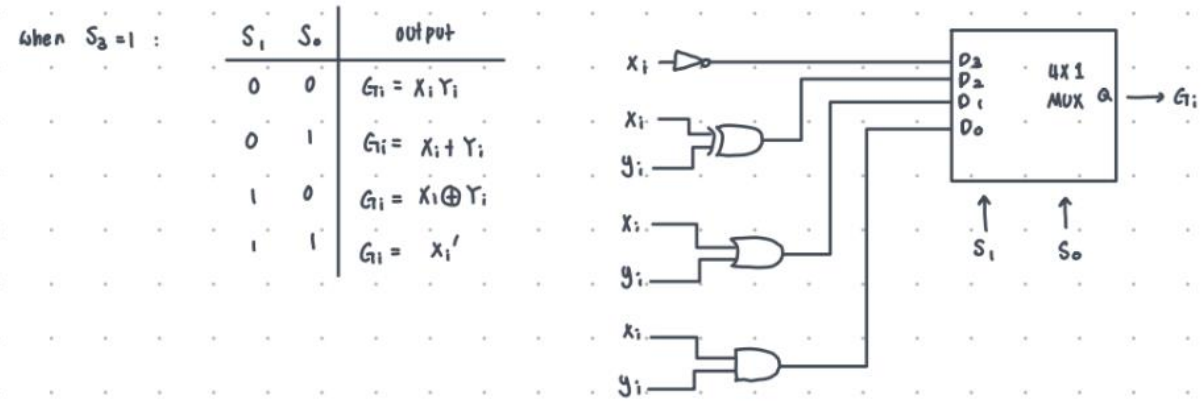
- Master-slave FF 는 두 개의 JK FF 로 구성된 순차 논리 회로이다. 첫번째로 사용되는 JK FF 는 master 이라고 불리며 clock 의 신호에 의해 동작한다. Master 의 출력 값이 두번째 JK FF 인 slave 플립플롭으로 전달된다. 먼저 clock 이 1 이 되면 master FF 를 동작 시켜 P 값을 저장한 뒤, clock 이 다시 0 이 되면 P 값을 이용한 추가적인 회로를 input 받아서 slave FF 를 동작 시킨다. 이후에 최종적인 출력 값 Q 가 나오게 된다.

3. 실험 준비

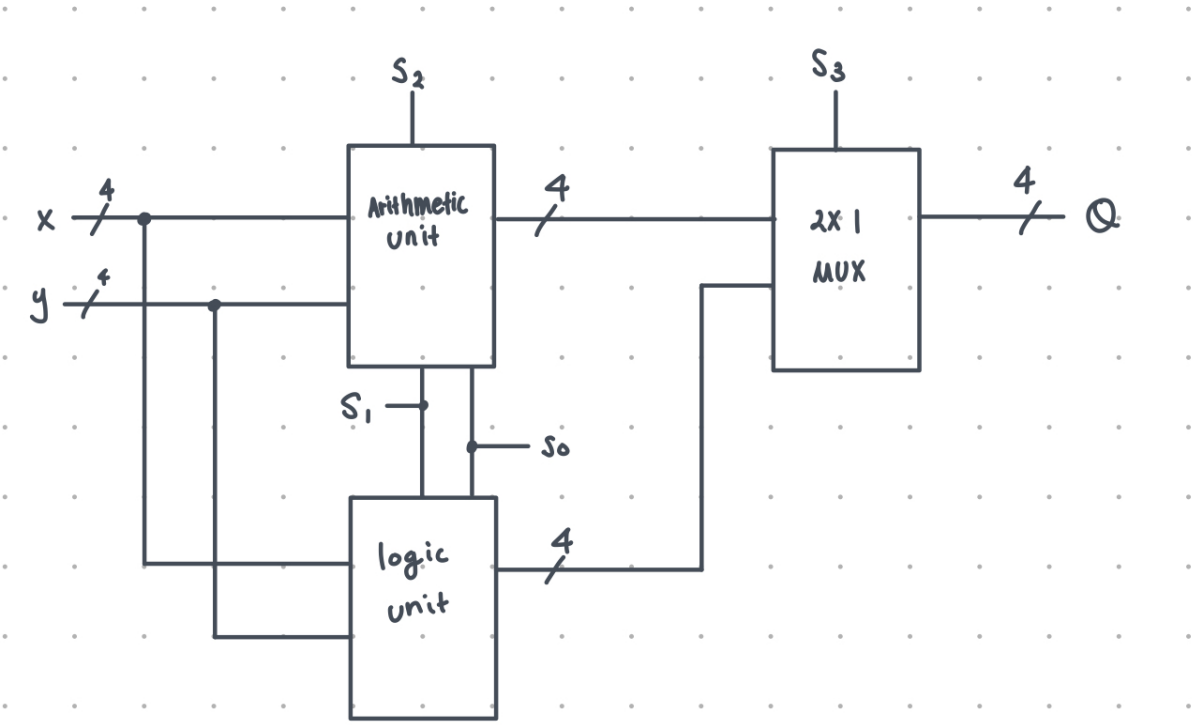
1) ALU (lab 5_1)

- S3 값에 따라 arithmetic unit 과 logic unit 으로 나눈뒤 K map 을 이용해 식을 단순화 시켰다. 이 식을 바탕으로 adder 를 이용한 회로도를 그렸다.





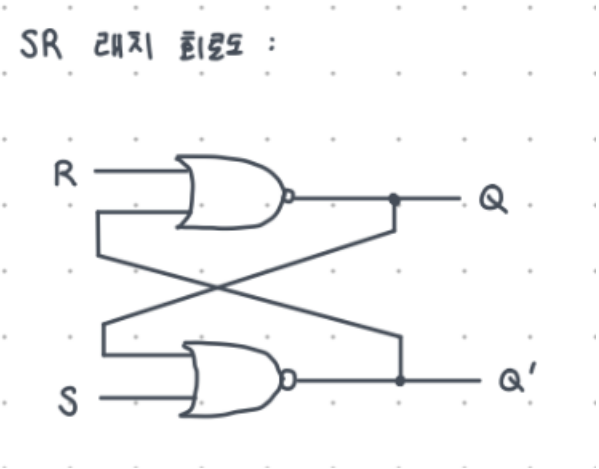
- 두 모듈을 2to1 MUX 로 묶어서 ALU 의 전체적인 회로도를 나타냈다.



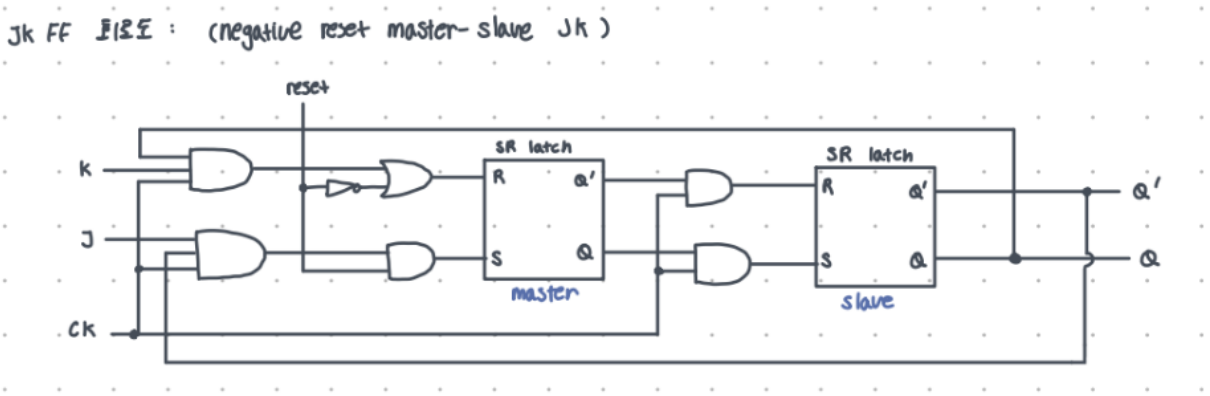
2)

Negative Reset Master-Slave JK FF

- SR latch 의 회로도를 그렸다.



- SR latch 를 사용해 negative reset master-slave JK FF 의 회로도를 그렸다.

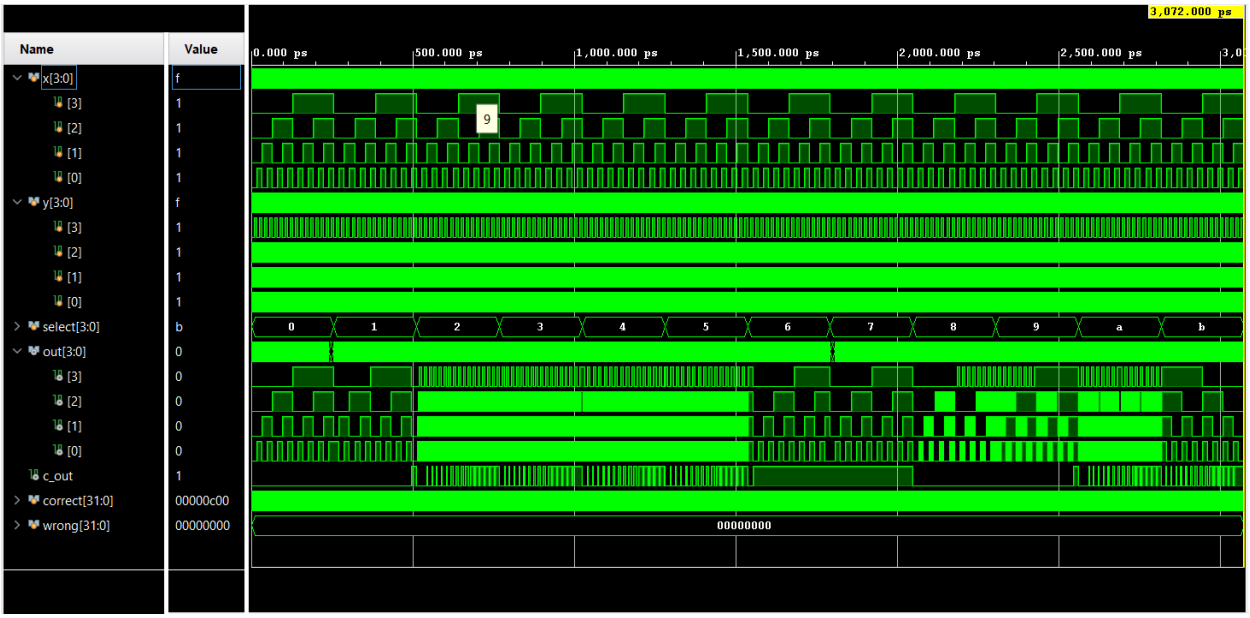


- 기존의 SR latch 를 사용하게 된다면 S = R = 1 일 때, glitch 가 발생하게 된다. 이런 glitch 를 방지하기 위해서 JK latch 를 이용해 toggle 하도록 만들었고 master-slave FF 는 이렇게 만들어진 J-K latch 를 이용한다. 그러나 CK 가 1 일 때, 또 다른 glitch 가 발생하게 되는데, 이번 오류는 master-slave JK FF 를 이용해서 해결할 수가 없다. 이 glitch 를 해결하기 위해서는 rising edge 또는 falling edge 에만 동작하도록 해야 할 것이다.

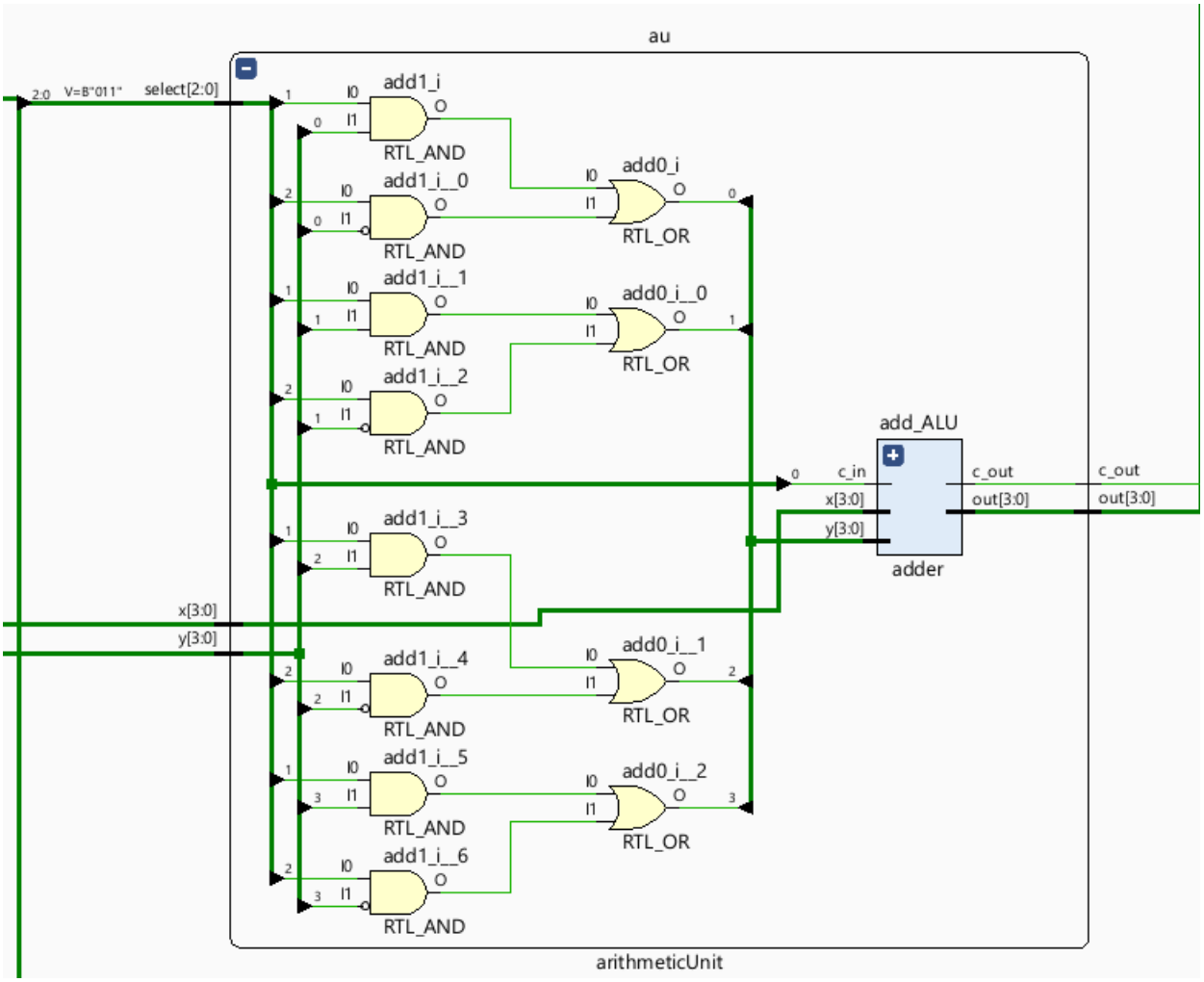
4. 결과

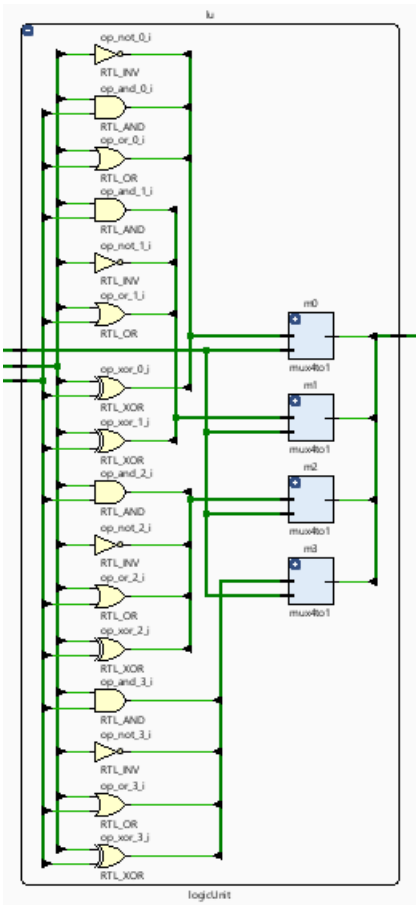
1) Arithmetic Logic Unit (ALU)

- 아래는 lab5_1 테스트 벤치를 사용해 시뮬레이션을 돌린 결과이다. 제일 아래의 두 항목을 보면 모두 correct 한 것을 알 수 있다.

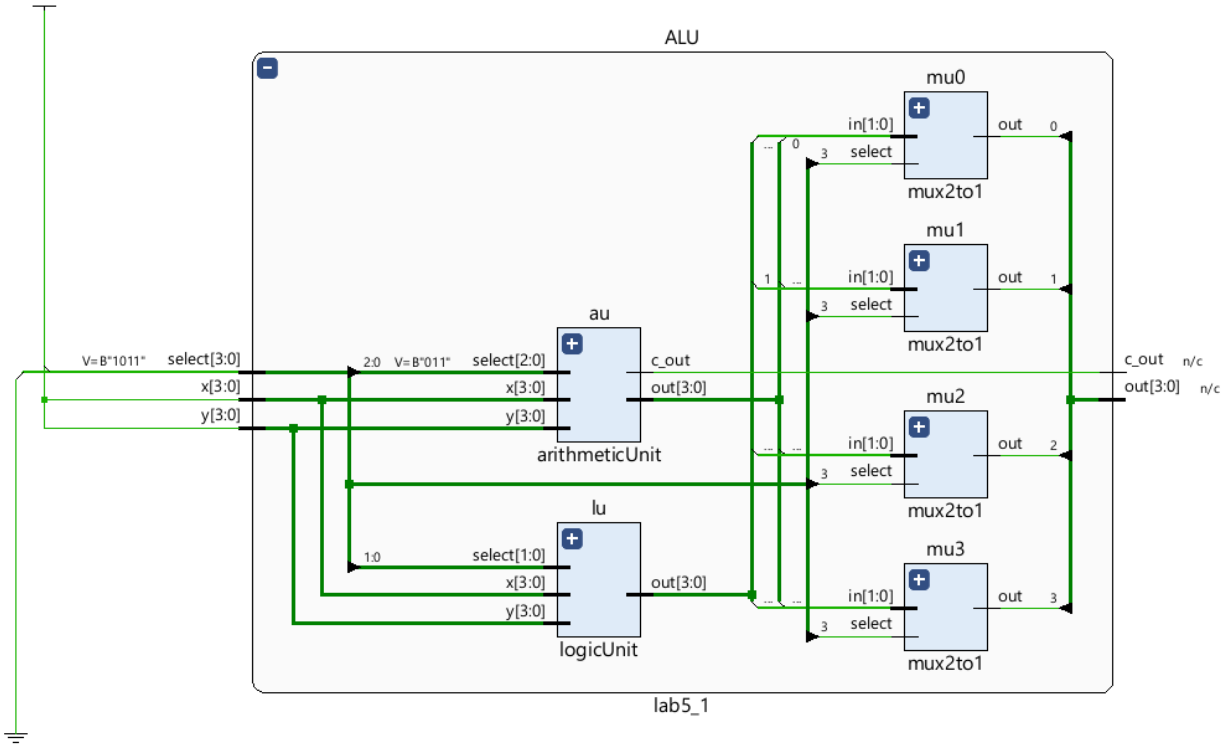


- 아래는 위에서 설명한 arithmetic unit 과 logic unit 을 schematic 으로 구현한 회로도 이다.



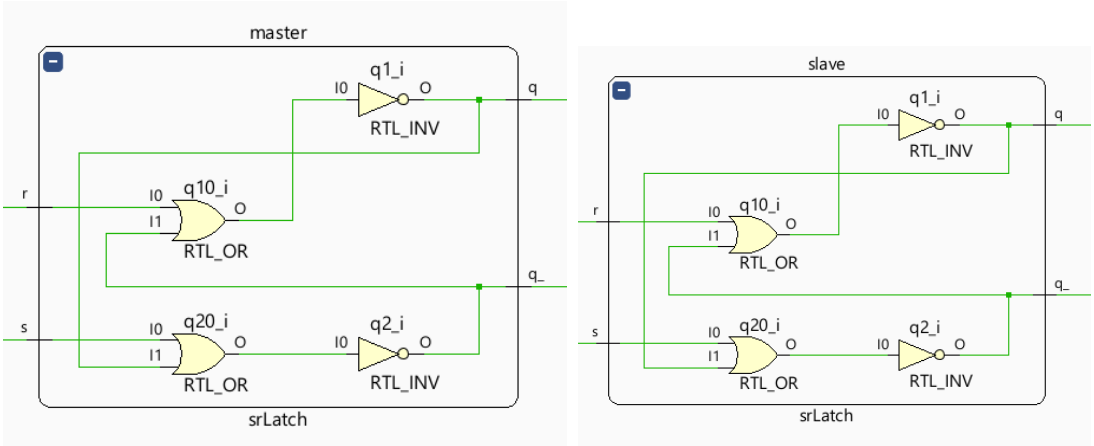


- 아래는 위에서 설명한 두 unit 을 2to1 MUX 를 이용해 구현한 회로도 이다.

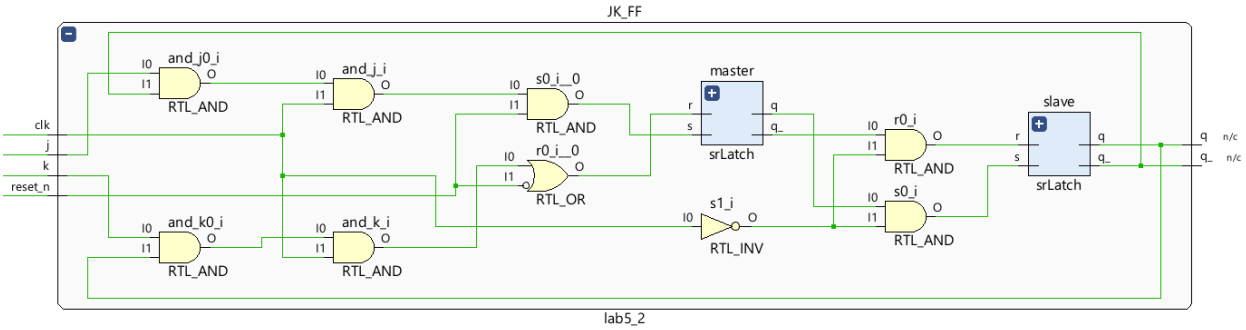


2) Negative reset Master Slave JK Flip Flop

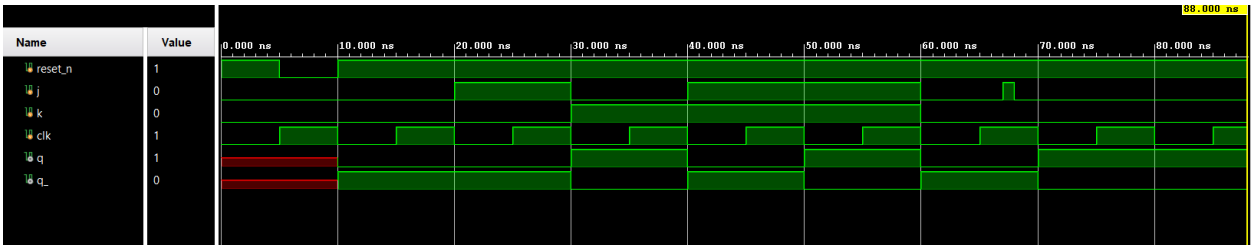
- 아래는 SR 래치로 구성된 master 과 slave 를 구현한 회로도 이다.



- 아래는 master slave JK FF 의 전체적인 회로도 이다.



- 아래는 lab5_2 테스트 벤치를 사용해 시뮬레이션을 실행한 결과이다.



- ALU 를 이용해 lab 5_1 실습을 진행하였다. ALU 를 이용한 동작을 x 와 y input 과 selection input 에 따라 result 를 나타내도록 하였다. 동작은 밑의 차트와 같고 결과는 밑의 사진과 같다.

x input	y input	selection input	result (in x and y)	result
1	3	0	x	1
		1	x+1	2
		2	x+y	4
		3	x+y+1	5
		4	x+y’	13
		5	x+y’+1	14
		6	x-1	0
		7	x	1
5	3	8	$x_i\ AND\ y_i$	1 $((1000)_2)$
		9	$x_i\ OR\ y_i$	7$((1110)_2)$
		10	$x_i\ XOR\ y_i$	6$((0110)_2)$

		11	x_i'	$10((0101)_2)$

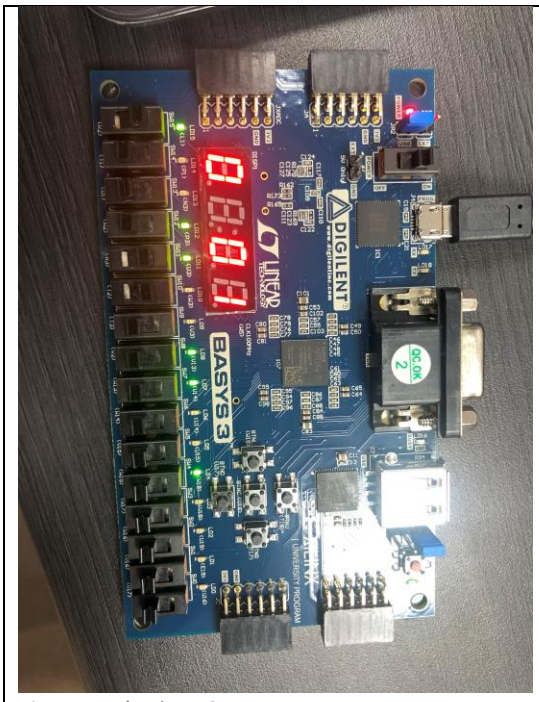


Figure 1 selection : 0

Figure 2 selection : 1

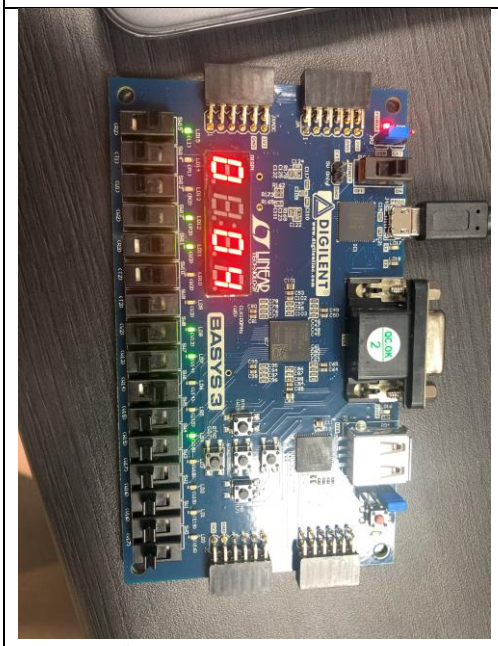


Figure 3 selection: 2

Figure 4 selection: 3

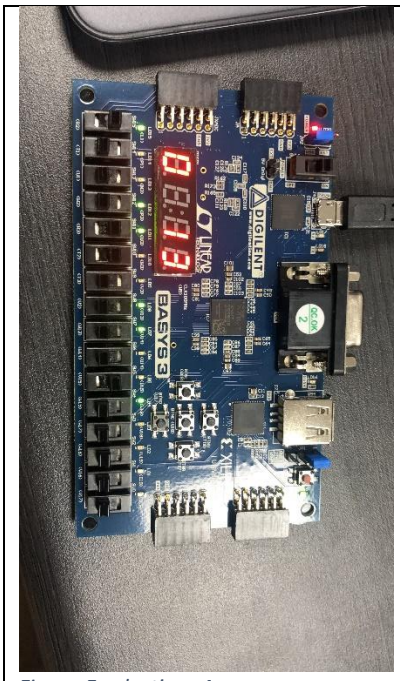


Figure 5 selection: 4

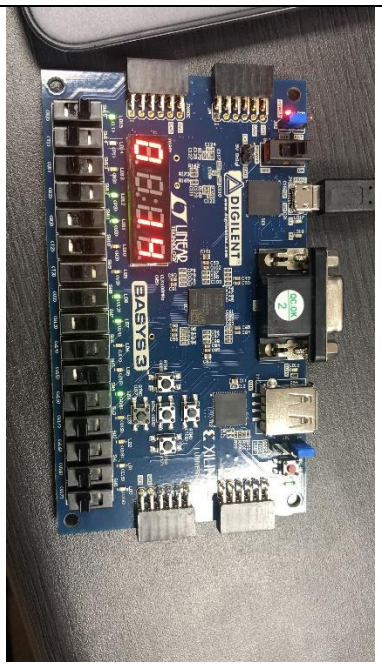


Figure 6 selection: 5

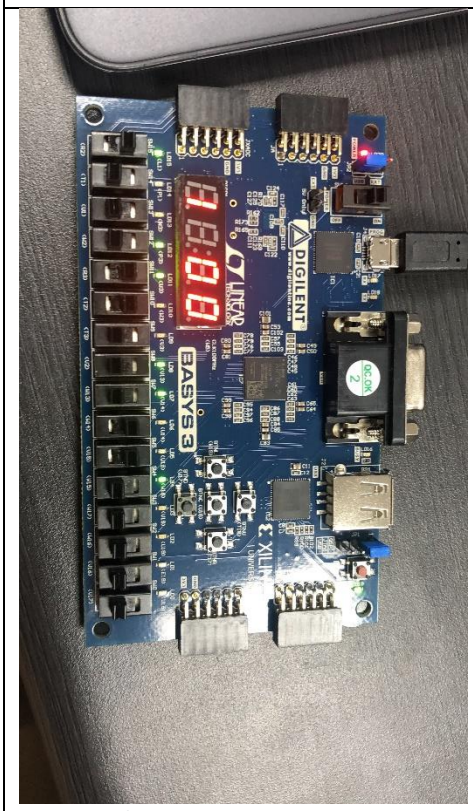


Figure 7 selection: 6

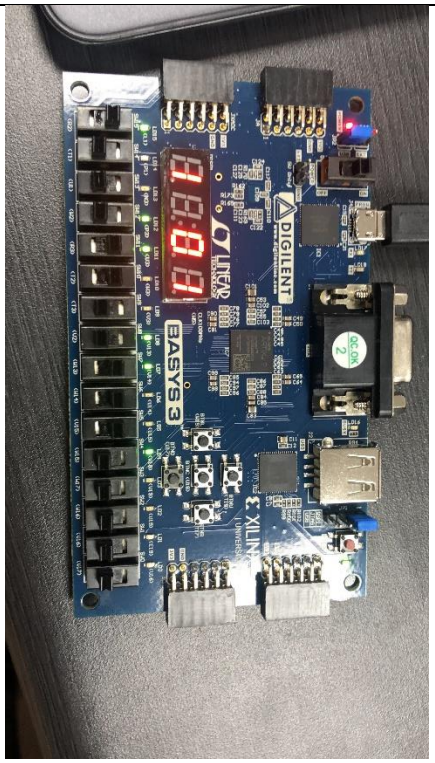


Figure 8 selection: 7

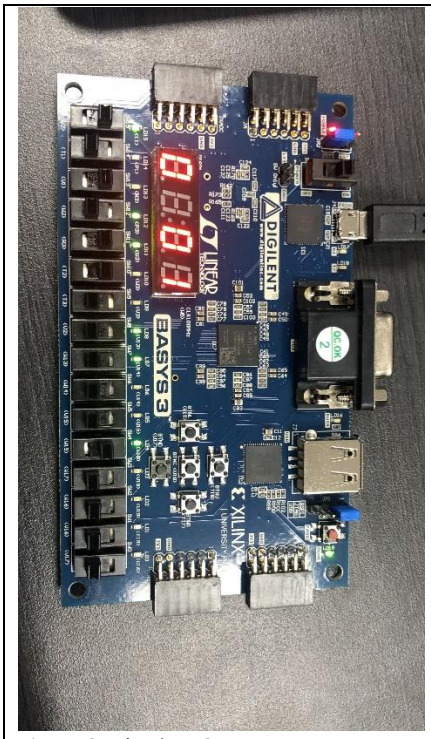


Figure 9 selection: 8

Figure 10 selection: 9

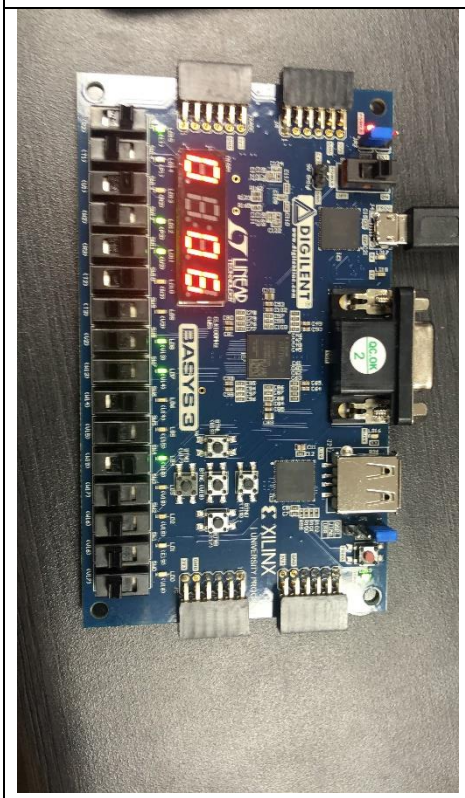


Figure 11 selection: 10

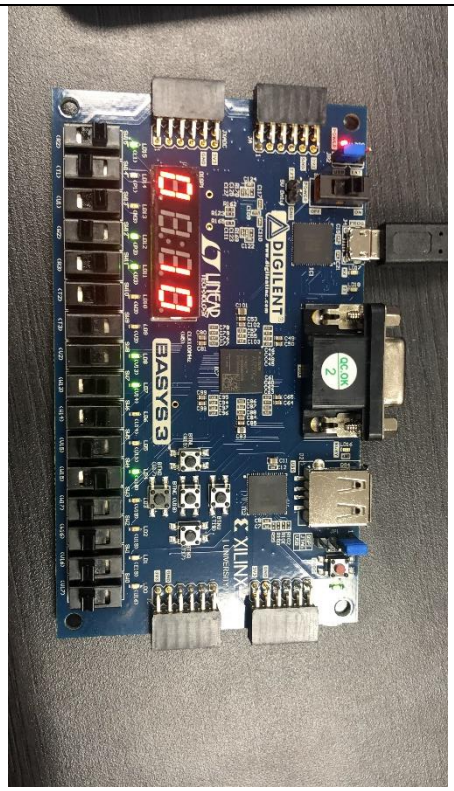


Figure 12 selection: 11

3) **논의**

- Lab5_1 를 작성할 때, 이때까지 배웠던 adder, MUX 등을 혼합해서 사용하려고 하니 어려움을 느꼈다. 이번 실험과 같은 큰 회로를 구성하기 위해서는 이런 단순한 코드들을 잘 이해하고 응용할 수 있어야 한다는 것을 배웠다. 그리고 test bench 를 작성할 때, x, y, c_out_expected, 그리고 out_expected 값을 지정하는데에도 Verilog 언어를 확실히 이해하고 작성해야 한다는 것을 배웠다.