CSED273 디지털시스템설계 Final Project

묵찌빠 게임 구현

14조 (20210643 김현준, 20210528 장정인, 20210273 하태혁)

1. 주제

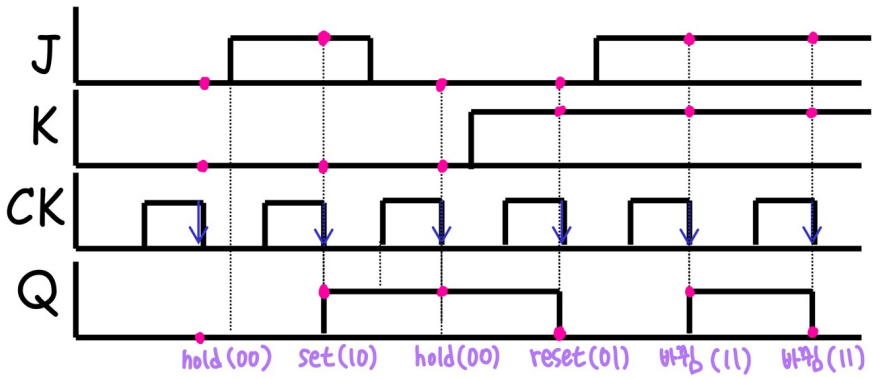
두 명의 플레이어 간의 삼세판 묵찌빠 게임을 구현한다.

사용자들의 입력(가위, 바위, 보)에 따라 Initial, start, attack, defence, win, lose, fin1, fin2의 8개의 state를 순환하는 FSM을 기반으로 하였고, bread board에 꽂힌 LED를 통해 어떤 player가 묵찌빠를 이겼는지, 삼세판의 최종 승자는 누구인지 확인할 수 있다. 수업에서 배웠던 FF, register, FSM 등의 logic circuit을 직접 구현하며 공부하는 것을 프로젝트의 목적으로 한다.

2. 구현과 관련된 수업 내용 및 배경 지식

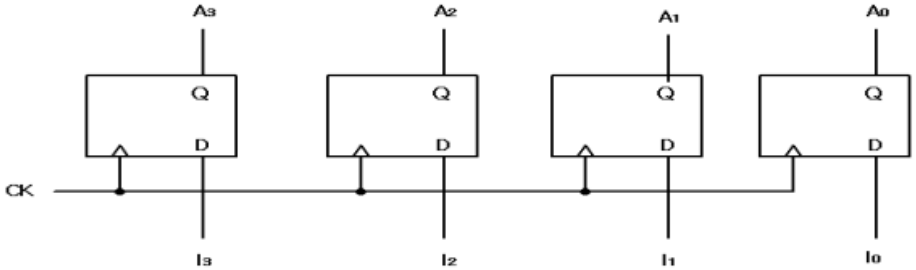
1. Negative edge-triggered Flip-flops

   Negative edge-triggered flip-flops는 CK=1일 때의 input pulse가 생길 때의 문제점을 해결하기 위해 CK의 signal이 1->0이 되는 순간에만 동작하는 JK flip-flop이다. JK flip-flop은 JK latch에 clock이 존재하는 형태를 말한다.



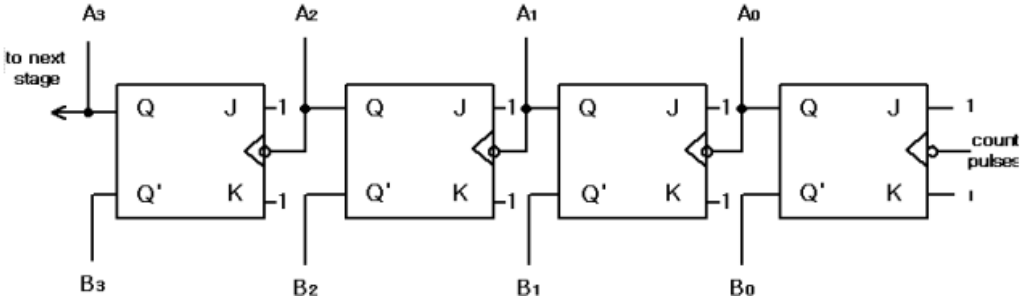
2. Register

   Binary data를 저장하기 위한 장치이고, flip flop 여러개를 이용하여 구현할 수 있다. 본 프로젝트에서 register는 입력된 R, P, C 정보를 잠시 저장하고, 이를 FSM의 입력으로 쓰기 위해서 사용된다. Register의 예시로, D Flip flop을 이용하여 4-bit parallel load register를 구현하면 아래와 같이 나타난다.



3. Counter

이전 state의 값이 다음 output을 계산하는 데에 사용이 되고, 마지막 state value는 맨 처음의 output을 계산하는 데 사용되는 형태로 구성된다. 즉, 계속 돌면서 1씩 증가되는 형태를 띄며, 이번 프로젝트에서는 3판 2선승제를 위해 승리 횟수를 카운트하기 위해 사용된다. 예시로, 4-bit binary ripple counter는 아래와 같이 구성된다.

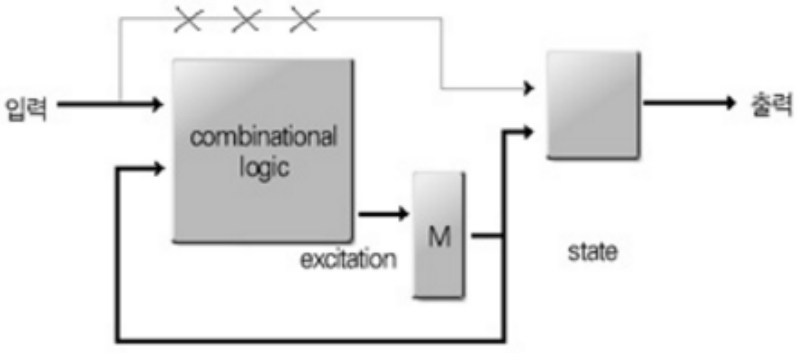


4. Finite State Machine

Finite State Machine(FSM)은 유한 개의 state를 가질 수 있는 오토마타이다. 동시에는 하나의 state만 가질 수 있으며, 어떠한 사건, 즉 입력 등에 따라 다른 state로 변화할 수 있다.

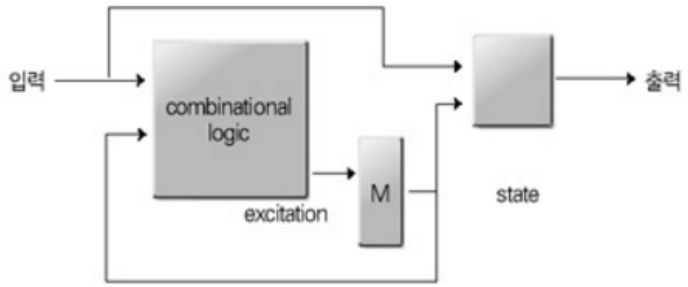
5. Moore Machine

Moore Machine은 출력이 현재 state에 의해서만 결정되는 finite state machine이다.



6. Mealy Machine

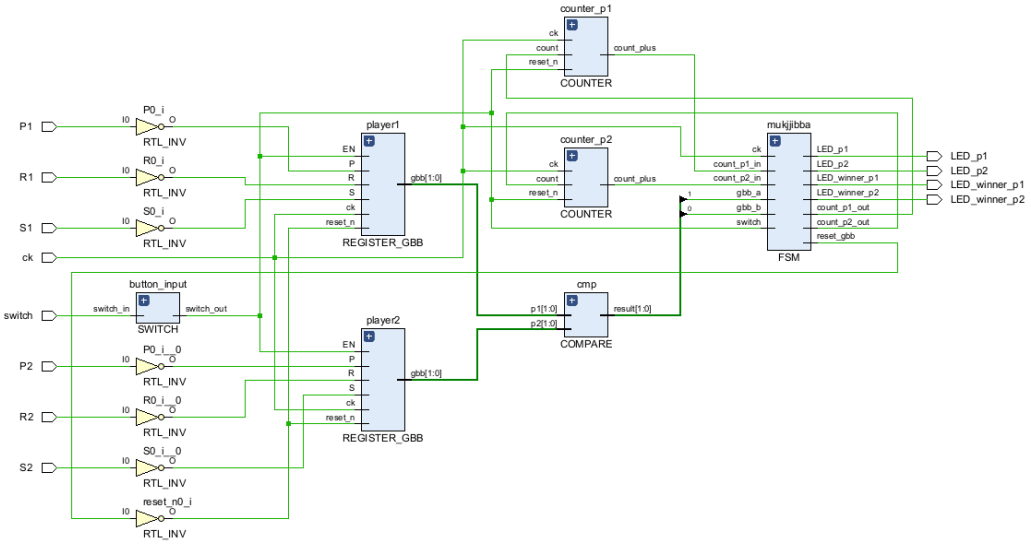
Mealy Machine은 출력이 현재 state와 입력에 따라 결정되는 finite state machine이다. Moore Machine보다 반응 속도가 빠르며, 입력 값이 출력 값에 직접적으로 영향을 미친다.



3. Modeling digital circuits

1) 전체 디자인

전체적인 회로 디자인은 아래와 같이 구성하였다.



2) Register_gbb 모듈 및 compare 모듈

플레이어 2명은 가위, 바위 또는 보를 3버튼을 통해 내게 되는데, 이 입력을 받아와서 레지스터에 저장하는 REGISTER_gbb 모듈을 디자인하였다. 이 레지스터는 2비트 정보를 갖고 있으며 아래와 같이 정보를 갖는다. 또한, 레지스터를 리셋해줄 수 있는 리셋 역할의 input도 받을 수 있다. 게임이 다시 시작될 때마다 이 리셋을 트리거하여 레지스터를 디폴트 값인 00으로 리셋하게 된다.

gbb_a	gbb_b	
0	0	default
0	1	rock
1	0	scissors
1	1	paper

다음으로 이 레지스터 두 개의 값을 비교하여 FSM의 input으로 가위바위보의 결과를 가져다주는 compare 모듈을 디자인하였다. 이 모듈은 REGISTER_gbb 두 개의 입력을 받아 결과를 출력하며, 아래와 같이 정보를 출력한다.

player1		player2		output	
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	0	1
0	1	1	0	1	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	1	1	1
1	0	1	0	0	1
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	1	1	0
1	1	1	0	1	1
1	1	1	1	0	1

default	0	0
draw	0	1
p1 wins	1	0
p2 wins	1	1

디폴트는 00, 두 플레이어가 비긴 경우 01, p1이 이긴 경우 10, p2가 이긴 경우 11을 출력하게 된다. 이 입력을 받아서 FSM이 작동된다. 이때 디폴트를 설정해 놓은 이유는, 가위바위보 및 묵찌빠의 경우 두 플레이어의 입력이 모두 들어와야 다음 state로 이동해야 하는데, 디폴트를 설정하지 않을 경우, 한 명의 입력만 들어와도 바로 다음 state로 넘어갈 수 있기 때문이다. 즉, 디폴트를 설정해 놓음으로써, 두 입력을 모두 받아야 state transition이 일어날 수 있도록 하였다.

3) SWITCH 모듈

다음으로, SWITCH 모듈을 사용하였다. 이 모듈은, 한 번 눌리면 자동으로 올라오는 형태의 스위치를, 한 번 누르면 잠겨 1의 출력이 유지되는 스위치로 만들기 위해서 논리적으로 구성한 모듈이다. 이 모듈은 flipflop의 clock신호를 스위치 신호에 not을 붙임으로써 구현하였다.

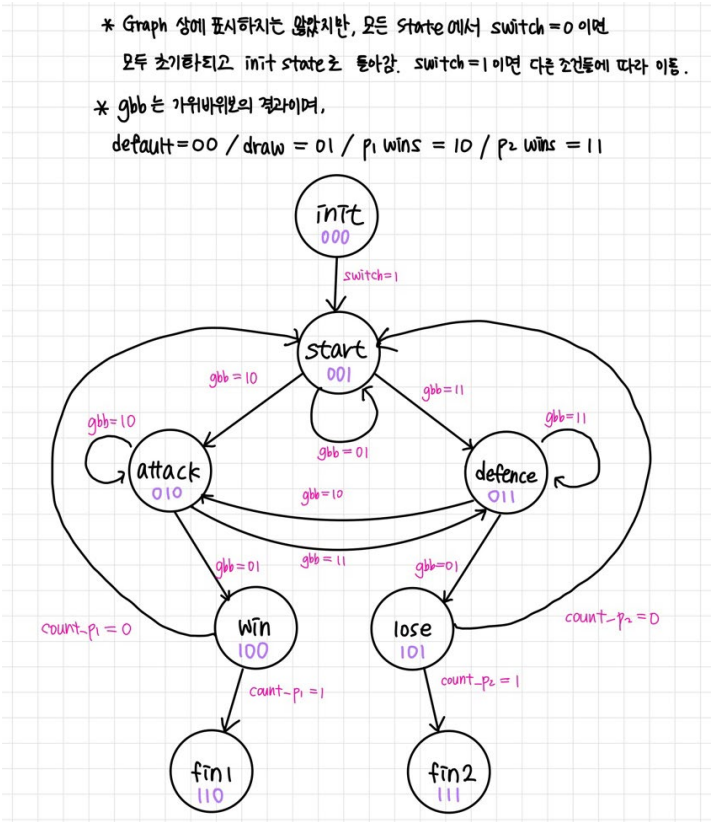
4) COUNTER 모듈

COUNTER 모듈은 player가 이긴 횟수를 기록하기 위한 모듈로 디자인하였다. 본 게임에서는 삼세판을 하게 되므로, 이 카운터는 처음에 0 값을 지니다가 1의 입력이 들어오면 T FF의 Toggle이 일어나 1로 변하고 다음으로 1이 또 들어오면 다시 0으로 이동한다. 여기로 들어오는 입력은 FSM의 output이다.

4. State Transition Diagram 및 State machine 설계

2) State Transition Diagram

묵찌빠 게임의 state transition diagram이다. 그림에서 설명한 대로 아래의 diagram에서의 공격, 방어, 승리, 패배는 player1의 기준으로 작성하였다. Init에서 게임 시작 switch를 누르면 게임을 시작할 수 있는 상태가 되며, 이후 6개의 스위치 (p1의 가위 바위 보, p2의 가위 바위 보)를 통해 들어오는 input을 통해 승패를 가리고, 그 승패 결과에 따라 state를 이동한다. 삼세판이므로 player가 이길 때마다 각자의 점수 count를 올려 주고, count=0인 상태에서 처음 승리하게 되면 count가 올라가며 다시 start state로 이동해 묵찌빠 게임을 다시 진행하며, count=1인 상태에서 한 번 더 이기게 되면 최종 승리(fin1, fin2) state로 이동하게 된다. Switch를 한 번 더 눌러서 0이 되면 어떤 state에 있었든 모든 정보가 초기화되고 init state로 돌아간다.



3) State Transition Table

시작 button을 누를 경우 게임이 시작된다. 사용자1과 사용자2의 스위치에서 받은 입력을 정의하기 위해 register를 사용했다. 바위, 가위, 보의 레지스터 출력 값은 아래 표와 같다.

p1	p2	gbb	
바위	바위	01	default: 00
바위	가위	10	draw: 01
바위	보	11	p1 wins: 10
가위	바위	11	p2 wins: 11
가위	가위	01	
가위	보	10	
보	바위	10	
보	가위	11	
보	보	01	

Player1과 player2가 낸 두 input값을 비교하여 Player1을 기준으로 p1이 승리하면 10, p1이 패배해서 방어 턴을 가질 경우 11, 무승부가 01의 output을 가지도록 하였다. 이는 전체 FSM의 input으로 들어가는데, 2bit이므로 output[1] = gbb_a, output[0] = gbb_b로 이름 지었다. 아래는 state를 도는 FSM의 transition table이다.

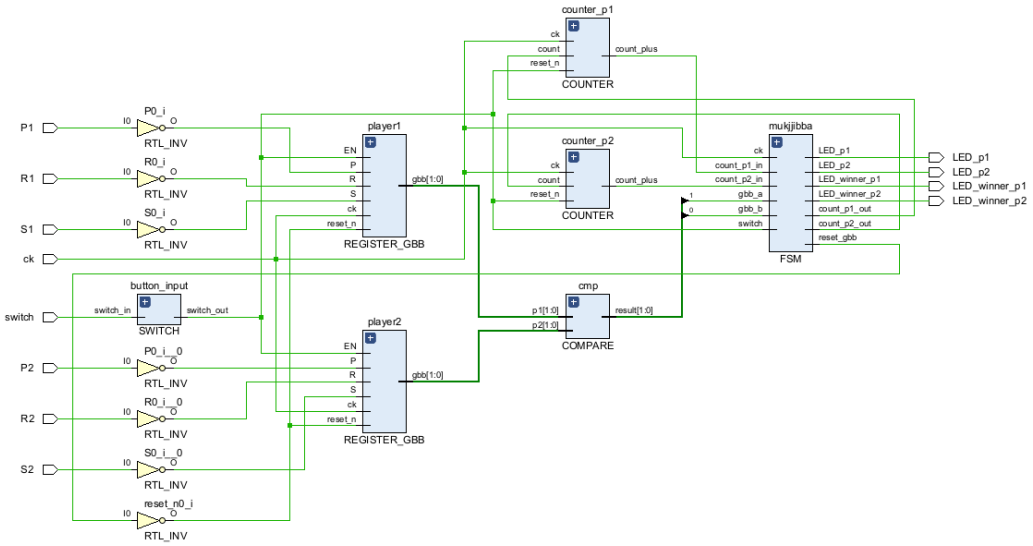
Present state				Input					Next state				Output					Implementation with T-FF		
state	A	B	C	switch	count_p1	count_p2	gbb_a	gbb_b	state	A+	B+	C+	reset_gbb	led winner_p1	led winner_p2	count_p1	count_p2	TA	TB	TC
init	0	0	0	0	x	x	x	x	init	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	1	x	x	x	x	start	0	0	1	1	0	0	0	0	0	0	1
start	0	0	1	0	x	x	x	x	init	0	0	0	0	0	0	0	0	0	0	1
	0	0	1	1	x	x	0	0	start	0	0	1	0	0	0	0	0	0	0	0
	0	0	1	1	x	x	0	1	start	0	0	1	1	0	0	0	0	0	0	0
	0	0	1	1	x	x	1	0	attack	0	1	0	1	0	0	0	0	0	1	1
	0	0	1	1	x	x	1	1	defence	0	1	1	1	0	0	0	0	0	1	0
attack	0	1	0	0	x	x	x	x	init	0	0	0	0	0	0	0	0	0	1	0
	0	1	0	1	x	x	0	0	attack	0	1	0	0	0	0	0	0	0	0	0
	0	1	0	1	x	x	0	1	win	1	0	0	1	0	0	0	0	1	1	0
	0	1	0	1	x	x	1	0	attack	0	1	0	1	0	0	0	0	0	0	0
	0	1	0	1	x	x	1	1	defence	0	1	1	1	0	0	0	0	0	0	1
defence	0	1	1	0	x	x	x	x	init	0	0	0	0	0	0	0	0	0	1	1
	0	1	1	1	x	x	0	0	defence	0	1	1	0	0	0	0	0	0	0	0
	0	1	1	1	x	x	0	1	lose	1	0	1	1	0	0	0	0	1	1	0
	0	1	1	1	x	x	1	0	attack	0	1	0	1	0	0	0	0	0	0	1
	0	1	1	1	x	x	1	1	defence	0	1	1	1	0	0	0	0	0	0	0
win	1	0	0	0	x	x	x	x	init	0	0	0	0	0	0	1	0	1	0	0
	1	0	0	1	0	0	x	x	start	0	0	1	0	0	0	1	0	1	0	1
	1	0	0	1	0	1	x	x	start	0	0	1	0	0	0	1	0	1	0	1
	1	0	0	1	1	0	x	x	fin1	1	1	0	0	1	0	1	0	0	1	0
	1	0	0	1	1	1	x	x	fin1	1	1	0	0	1	0	1	0	0	1	0
lose	1	0	1	0	x	x	x	x	init	0	0	0	0	0	0	0	1	1	0	1
	1	0	1	1	0	0	x	x	start	0	0	1	0	0	0	0	1	1	0	0
	1	0	1	1	0	1	x	x	fin2	1	1	1	0	0	1	0	1	0	1	0
	1	0	1	1	1	0	x	x	start	0	0	1	0	0	0	0	1	1	0	0
	1	0	1	1	1	1	x	x	fin2	1	1	1	0	0	1	0	1	0	1	0
fin1	1	1	0	0	x	x	x	x	init	0	0	0	0	0	0	0	0	1	1	0
	1	1	0	1	x	x	x	x	fin1	1	1	0	0	1	0	0	0	0	0	0
fin2	1	1	1	0	x	x	x	x	init	0	0	0	0	0	0	0	0	1	1	1
	1	1	1	1	x	x	x	x	fin2	1	1	1	0	0	1	0	0	0	0	0

5. 전체적인 동작 설명

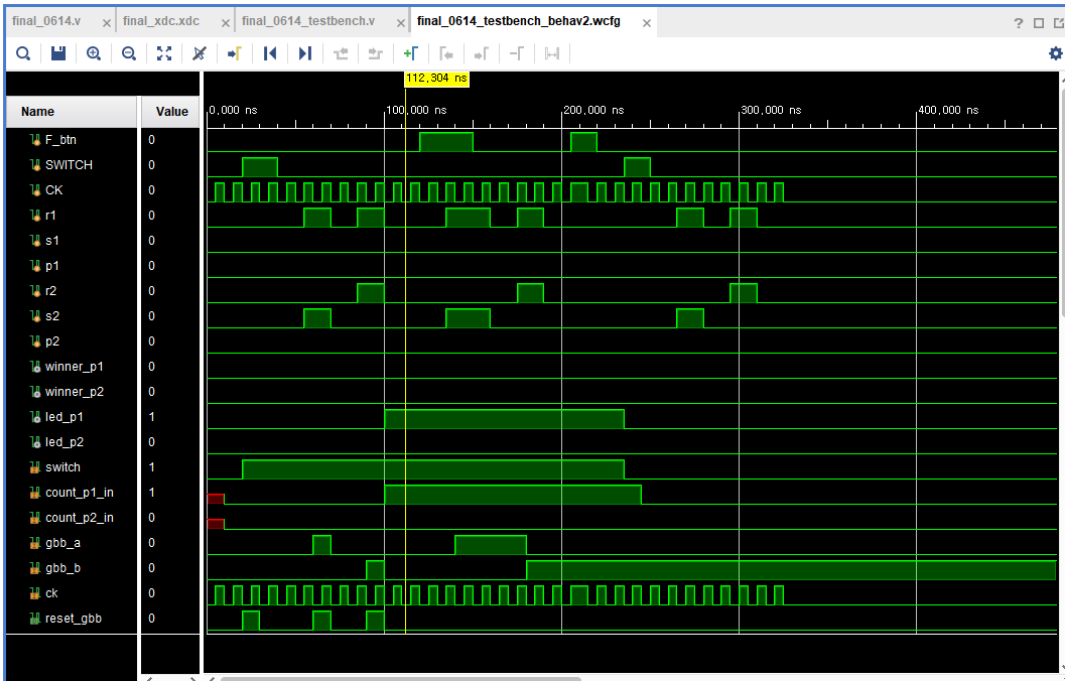
2명의 사용자가 삼세판 묵찌빠 게임을 진행한다. Switch를 누르면 게임이 시작되며, 게임이 진행되는 동안 사용자는 가위, 바위, 보 중 하나를 낼 수 있다. 셋 중 하나를 내면, 그 결과를 바위:01, 가위:10, 보:11로 바꾸어주는 register가 각각의 player에게 작동하고, 그 register의 output에 따라 compare 모듈이 작동한다. 이는 default:00, draw:01, p1 wins:10, p2 wins:11의 결과를 출력한다.

승자가 나올 때까지 가위바위보를 반복하며 승자가 정해지면, 그 사용자에게 공격 기회가 주어진다. 공격에서 성공할 경우(가위바위보 무승부) 승리하게 되고, 가위바위보를 이기게 되면 계속 공격 기회를 유지하고, 공격에서 실패하면(가위바위보 패배) 공격자가 바뀐다.

승패가 정해지면 이긴 사용자의 방향에 작은 LED가 켜진다. 총 삼세판을 진행하고, 게임이 끝났다면 이긴 사용자의 방향에 큰 LED가 켜진다.



아래 사진은 정상 작동을 확인하기 위해 디자인한 testbench로 확인한 simulation 결과이다. P1이 이기는 경우, p2가 이기는 경우 등 모든 경우를 testbench를 통해서 확인해보았고, 제대로 작동하는 것을 알 수 있었다. 또한, switch의 온오프에 대해서도 제대로 초기화가 되는지 확인해보았고, 모두 정상작동하는 것을 확인하였다.

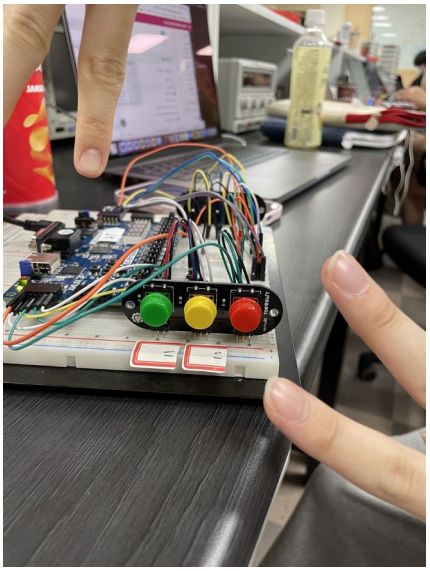


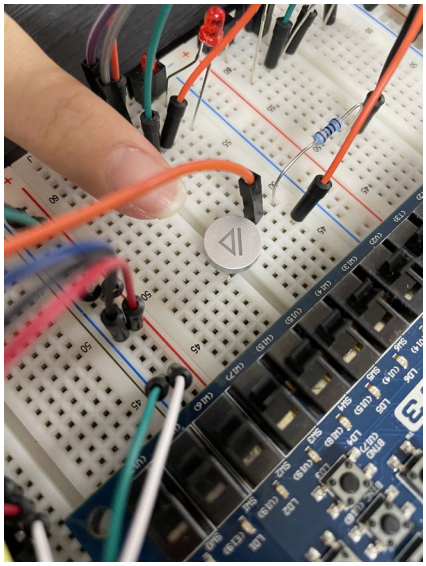
6. 입출력 방법 및 동작 설명

- 입출력 장치

: 사용자 p1 스위치 3개, 사용자 p2 스위치 3개, Start/Reset 스위치 1개, LED 4개(묵찌빠 p1, p2 삼세판 최종 p1, p2)

아래 왼쪽 사진이 가위(노랑), 바위(빨강) 보(초록) 입력 스위치이고, 오른쪽 사진이 시작 스위치이다.





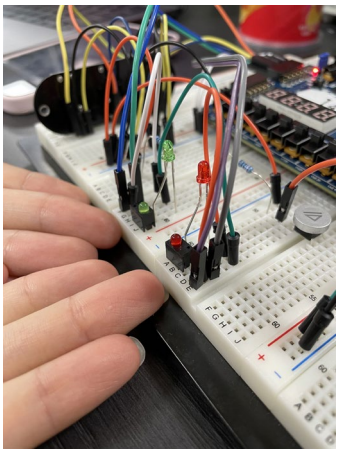
입력 장치 중에 3버튼 모듈의 경우 버튼이 눌리지 않은 상태가 1을 출력하고, 눌린 상태가 0을 출력한다. 따라서 이 입력을 받아와서 not 게이트를 붙여주어 반전을 시켜서 FSM에서 사용하였다.

- 출력 장치

: Win check LED 2개, 최종 win check LED 2개

사용자 스위치 3개는 각각 가위, 바위, 보를 나타내고, 묵찌빠 게임 1승당 작은 LED의

불이 승자의 방향에 켜지고, 삼세판이 끝나면 큰 LED의 불이 승자의 방향에 켜진다.



7. 토론 및 제언

Project를 진행하면서 counter, register, comparer 등의 모듈을 만들어 연결하는 과정이 처음에는 막막했지만, 마지막에 전체 모듈을 만들어 wire로 연결해주고, 상태 표를 만들어서 조건에 따라 state를 전이해주는 과정부터 다시 체크하며 만들었더니 계산을 실수했던 부분 등을 고쳐나가며 무사히 완성할 수 있었다. 또한 한 학기간 공부했던 lab 과제와 크게 다르지 않다는 것을 느꼈다. Bitstream을 생성하는 과정에서도 오류가 많이 생겼지만, port 하나하나를 체크해보며 오류를 찾아가며 회로를 구성하였는데, 완성 후에 뿌듯함을 느꼈다. 또한 수업에서 배웠던 register, counter, FSM, FF 등의 모듈을 직접 구성하고 연결하고, 프로그램 전체를 구상하는 과정에서 디지털시스템설계 과목을 가장 잘 이해할 수 있는 시간이었던 것 같다.

추가로, 팀원 한 명이 본격적으로 본 프로젝트를 시작하기 전에, 사고로 크게 다쳐서 불행하게도 나머지 팀원 두 명이 프로젝트를 수행하였다.

추가로, 실험적으로 버튼을 동시에 뗄 수 없는 문제가 존재한다. 즉, clock이 ns 단위이기 때문에 동시에 뗀다고 해도 어느 한 player의 버튼이 늦게 떼어지고, 그러면서 draw를 처리한 다음 남아 있는 register 입력이 생기게 된다. 이것은 win, lose 상황에서는 문제가 되지 않지만 draw 상태에서는 문제가 발생할 수가 있었다. 예를 들어, player1과 player2가 동시에 묵을 냈을 때, player1이 조금 더 늦게 뗀다고 하면, palyer1의 가위바위보 register에 묵 값이 남아있게 된다. 따라서 다음에 둘이 찌를 냈을 때에 player1이 이기는 것으로 인식이 된다. 이 부분은 draw state를 따로 만들거나, always begin에서 딜레이를 주게 되면 해결할 수 있지만, draw state는 output이 start state와 같아서 없앴고, always 관련 문법은 사용 불가하다. 따라서 현재로서는 이러한 문제를 해결할 수가 없었다.