텀프로젝트 결과보고서

Tic Tac Toe 구현하기

**TTT**

김도훈 2018115809

1. 프로젝트에 사용된 모듈
   1. Keypad scan

* 키패드 스캔을 통해서 원하는 자리에 돌을 착수할 수 있게 한다.
* Dot matrix만으로는 부족한 시야를 보드판을 움직여 줌으로서 해결해 주게 한다.
  1. Dot matrix
* 게임 보드판의 현황을 보여준다. 키패드 스캔을 통해 입력받을 수 있다.
*  좌측 상단은 X돌을 표시한 것이고, 우측 하단은 O돌을 표시했다.…
* \*과 #를 눌러서 보드를 한 칸씩 좌우로 밀 수 있다.
  1. 7 segment
* 7 segment로 현재 정보를 표시한다



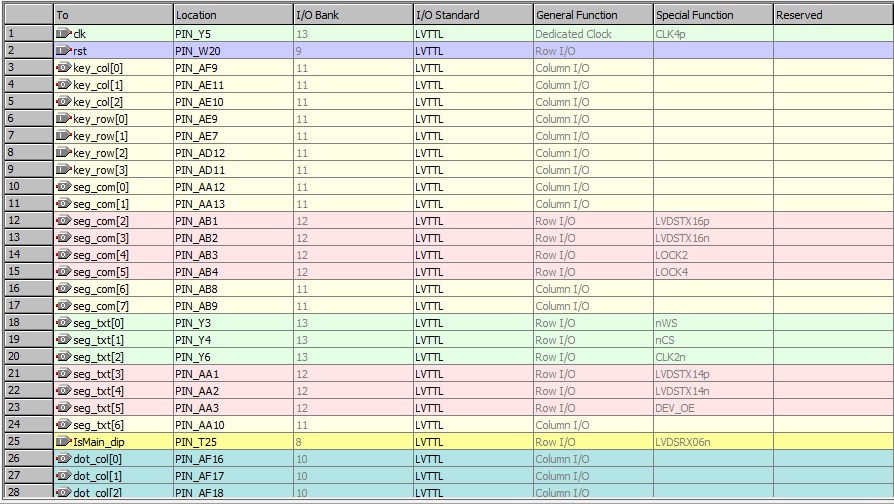
* 예를 들어 왼쪽 그림과 같이 게임 시작 전인 메인상태에서 PrESS UP를 표시한다.  
  또한, 승패를 결정하거나, 현재 누구 차례인지 표시하게 된다.
  1. User LED Display

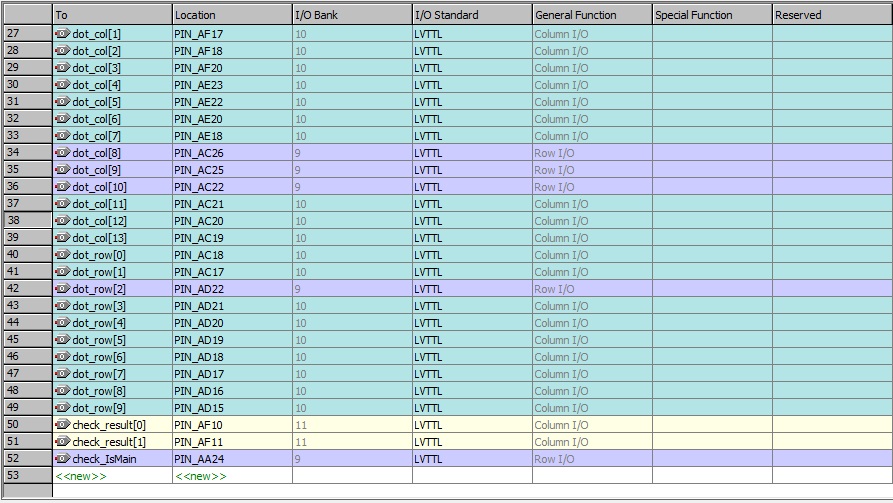
- LED Display로 부가적인 정보를 표시한다.

- D18: 게임이 끝나고 결과를 나타낼때를 표시한다.

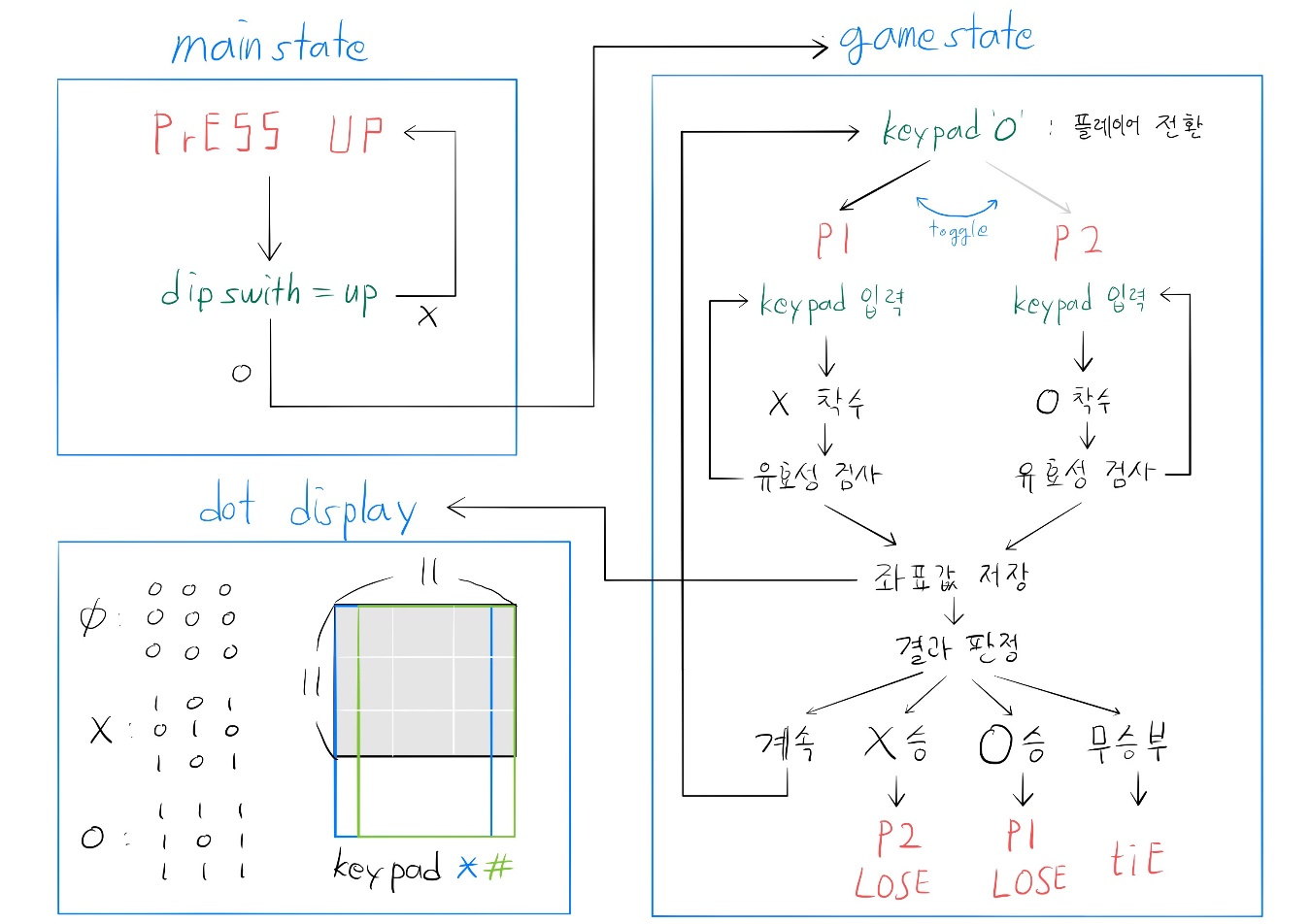
- D32: 게임이 Main 상태임을 표시하고 게임을 시작하면 불이 꺼진다.

1. 모듈에 대한 핀 할당





1. 논리 흐름도 + 제품설명서

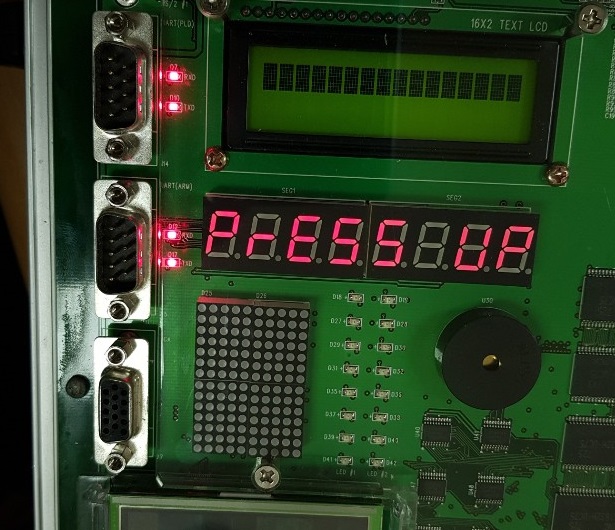


* 1. 메인 화면 상태 (mainstate)
* 프로그램 구동 시 가장 먼저 실행되고있는 상태로, 딥 스위치의 1행 1열 스위치를 위로 올리면 게임 플레이 상태로 진입한다. 8개의 7-Segment에 Press Up을 표시하여 딥스위치의 조작을 유도하였다.
  1. 게임 플레이 상태 (gamestate)
* 게임 플레이 상태로 진입하면 P1의 상태로 시작한다. Keypad의 1~9번 버튼을 눌러 플레이어가 X 또는 O를 놓을 위치를 지정하면, 그 위치가 유효한지 즉 이미 X 또는 O가 놓여져있는 위치인지 아닌지 판별한다. 만약 이미 채워진 위치라면 아무런 동작을 취하지 않고 유효한 입력을 기다린다. 비어있는 위치라면 그 위치에 무엇이 놓여졌는지를 나타내는 값을 저장한다.
* 다음으로 현재 게임 진행 상태에 대한 결과 판정을 하여 X로 한 줄이 완성되었으면 X승, O로 한 줄이 완성되었으면 O승, 한 줄이 완성되지 않은 채로 9칸이 모두 채워져 게임이 끝나면 무승부를 반환하고, 이것이 모두 아닌 경우 즉 게임이 끝나지 않은 경우 keypad의 0번 버튼을 눌러지기를 기다린다. keypad 0번 버튼을 누르면 플레이어가 P1에서 P2로 (혹은 P2 에서 P1로) 전환되어 다음 턴이 계속 진행 된다.
* 게임이 끝난 경우 7-Segment에 결과를 출력한다. X (P1)이 이겼을 경우 ‘P2 Lose’를, O (P2)가 이겼을 경우 ‘P1 Lose'를, 비겼을 경우 ‘tie'를 출력하고 reset을 기다린다.
  1. 게임 화면 출력 (dot\_display)
* 저장되어있는 게임의 현재 진행 상황을 dot matrix display에 출력한다.
* X와 O의 모양을 구분하지 위해서는 적어도 가로 세로 3칸씩이 필요하고, 칸과 칸 사이를 구분하기 위해서 한 칸씩이 필요하기 때문에 현재의 게임 보드판을 시각화하기 위해서는 11X11 크기의 display가 필요하다
* 하지만 dot matrix display의 경우 가로가 10칸이므로 벡터 부분 선택을 활용하여 keypad의 #과 \*을 눌러 오른쪽, 왼쪽으로 한 칸 씩 이미지를 움직일 수 있게 하였다.
  1. 리셋 (Reset)
* 게임이 끝나거나 처음부터 다시 시작할 수 있게 하기 위해 보드판을 초기화시킬수 있는 장치를 마련한다. PUSH #0버튼을 누르면 초기화된다.

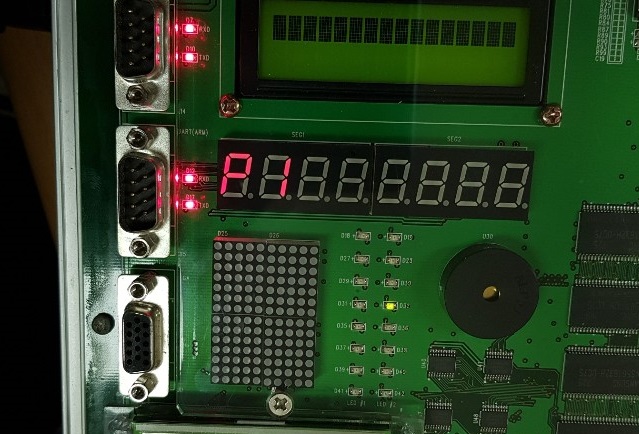
1. 만드는 과정에서 문제점 + 해결한 과정
   1. 컴파일 에러: Can’t elaborate user hierarchy “”

* 컴파일 할 때 이 오류가 생긴 적 있었다. 어느 줄에서 에러가 나는지 알 수 없어서 난감했었다.
* 주석을 이용하여 오류가 생기는 문장을 찾아보고 컴파일이 안 되는 부분을 다르게 코드를 작성하였다.
  1. 2비트 result 선언: reg [1:0] result;
* 컴파일이 잘 되어도 원하는 결과로 실행이 잘 되지 않았다. reg result; 이라고 작성하고 0~3의 수를 사용했기에 일어난 일이었다. 간단한 것이지만 발견하기 어려웠다.
  1. 컴파일 에러: Net 'IsMain' or a directly connected net is driven by more than one source, and not all drivers are three-state.
* 하나의 변수가 여러 개의 always문에서 변수값이 할당되게 하면 안된다. 하나의 변수는 하나의always문으로 작성하게 하고, 초기화 역시 같은 always문 안에서 처리하게 한다.

1. 현재 해결하지 못한 점 + 현황
   * 키패드를 눌러 dot matrix에 O, X를 두는 경우 가끔 한번 눌러도 여러개의 수가 동시에 출력되는 오류 -> 짧게 누르면 그럴 확률이 줄어듦
   * 간헐적으로 dot-matrix에 수를 두기위해 키패드를 눌렀을 때 화면을 움직이는 키패드가 입력될 경우가 있음
   * 돌을 착수했을 때 차례가 자동으로 바뀌지 않는다. 현재 수동으로 바뀌게 설계해두었다.
2. 실행 결과
   * Main 상태일 때

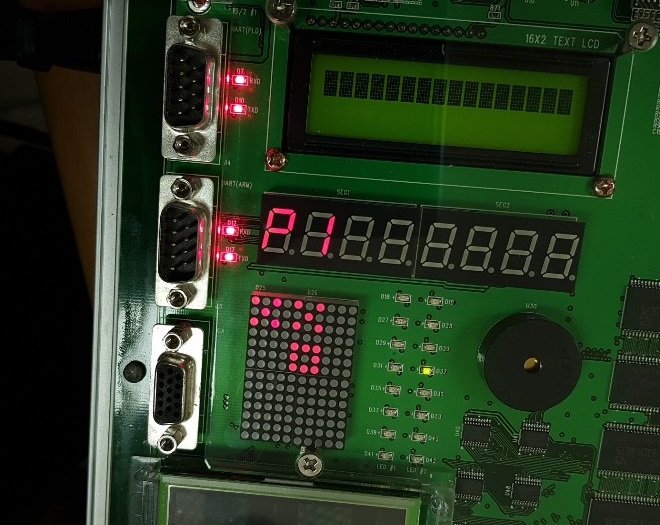


* + dip 스위치를 올리고 게임 상태로 들어갔을 때



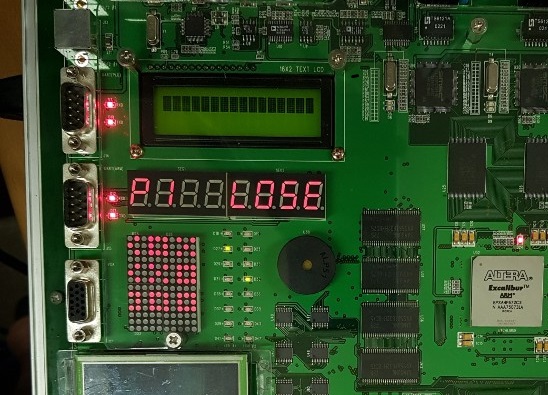
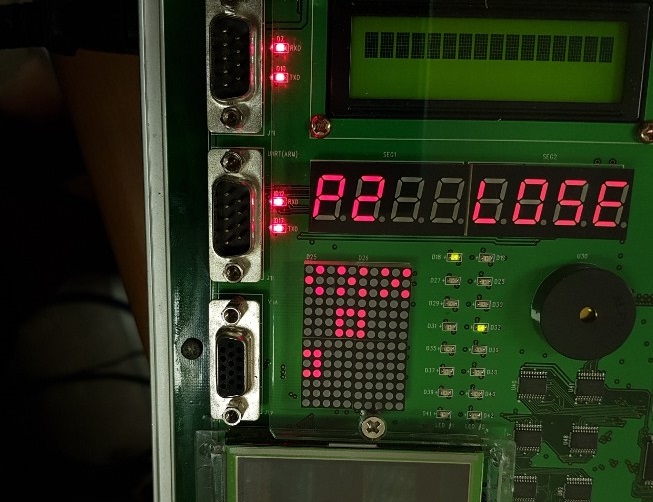
* + 게임 플레이중



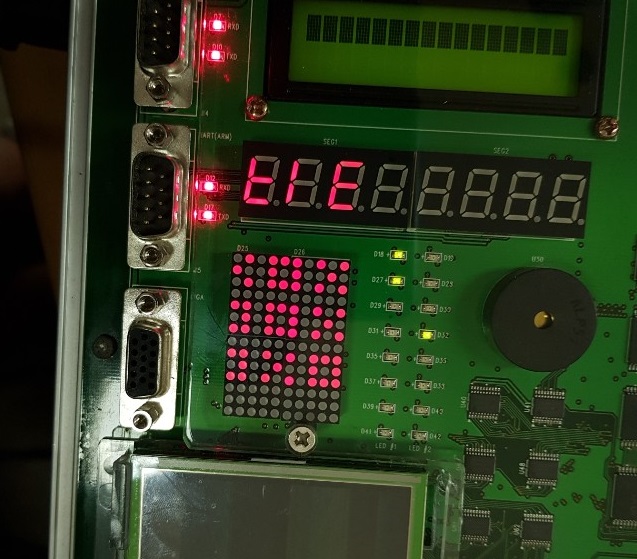
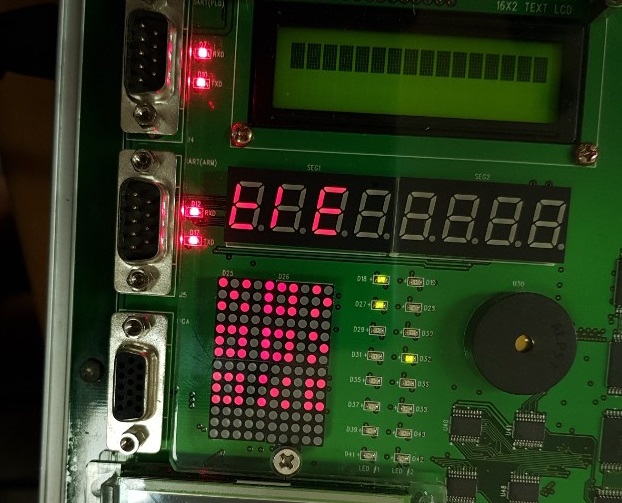


* + 결과

- P1이 이긴 경우 - P2가 이긴 경우



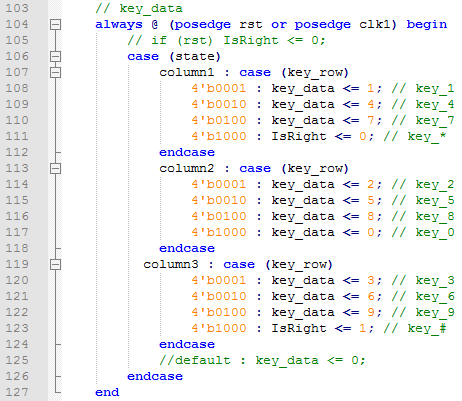
- 비긴(TIE) 경우(방향 키로 좌우를 움직임)



1. 소스코드, Github (<https://github.com/greathoney/Verilog_termproject>)

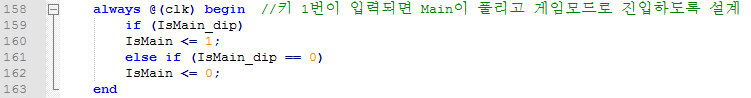
- 키패드 스캔

- \*를 누를 경우 왼쪽, #을 누를 경우 오른쪽으로 이동할 수 있도록 설정



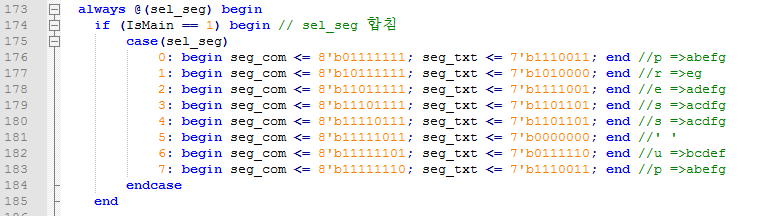
- mainstate 토글

- 딥 스위치를 on 시키면 Main 상태에서 게임 상태로 넘어감



- 세그먼트 데이터 할당

- 게임이 Main 상태인 경우 7-segment에 'PrESS UP' 출력



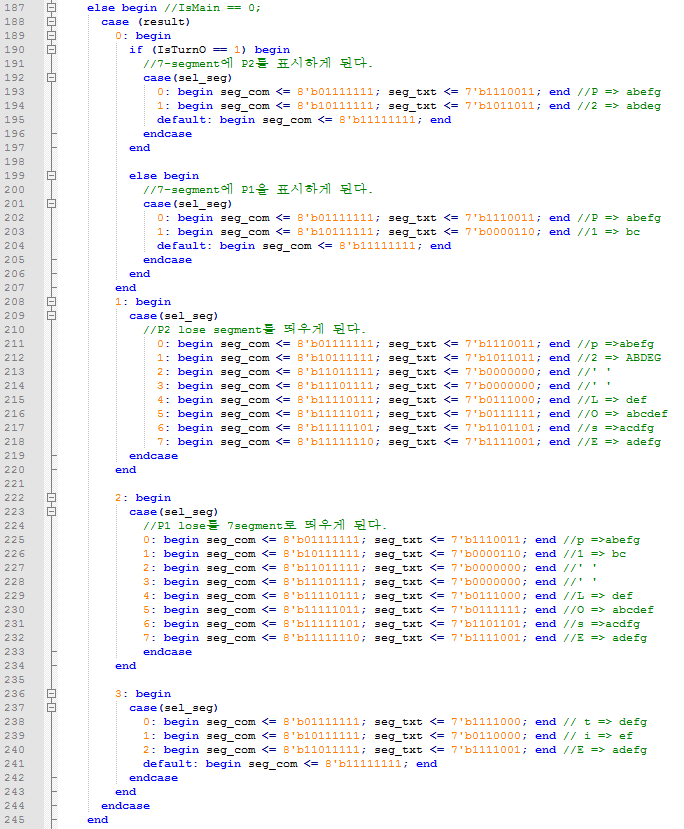
- 게임 상태로 넘어간 경우, result 값에 따라 상태가 달라짐

- result=0(게임중) -> 7-segment에 플레이어 'P1'과 'P2'를 IsTurn0 값에 따라 번갈아가며 출력

- result=1(P1 승) -> 7-segment에 'P2 LOSE' 출력

- result=2(P2 승) -> 7-segment에 'P1 LOSE' 출력

- result=3(동점) -> 7-segment에 'tIE' 출력



- 딥스위치가 on일 경우, key\_data 값에 따라 board에 값 할당

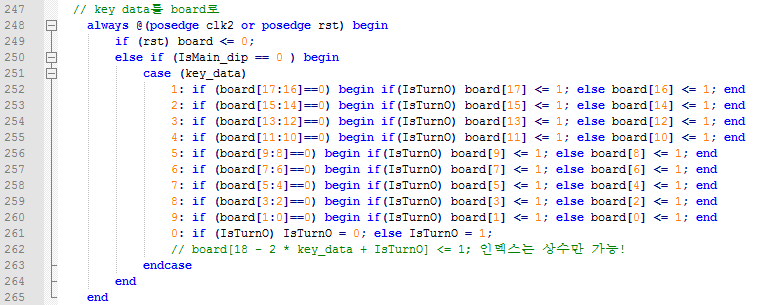
Board는 9자리의 위치가 각각 어떤 상태에 있는지를 서술하는 18비트 벡터값이다.

아무것도 놓여져 있지 않은 상태를 00, X가 놓여져 있는 상태를 01, O가 놓여져 있는 상태를 10으로 하여 한자리당 2비트씩 할당하여 사용하였다.

Board에 저장하기 이전에 거치는 if 문에서 ‘두자리 비트의 부분 선택이 0이면’ 이라는 부분은 그 자리에 O나 X가 하나라도 놓아져 있는가를 판단하는 것이므로 아무것도 놓아져 있지 않으면 값을 할당한다.

key\_data: 1~9 -> board에 자리가 비어있으면 값을 할당

key\_Data: 0 -> P1과 P2의 차례를 바꾸어줌

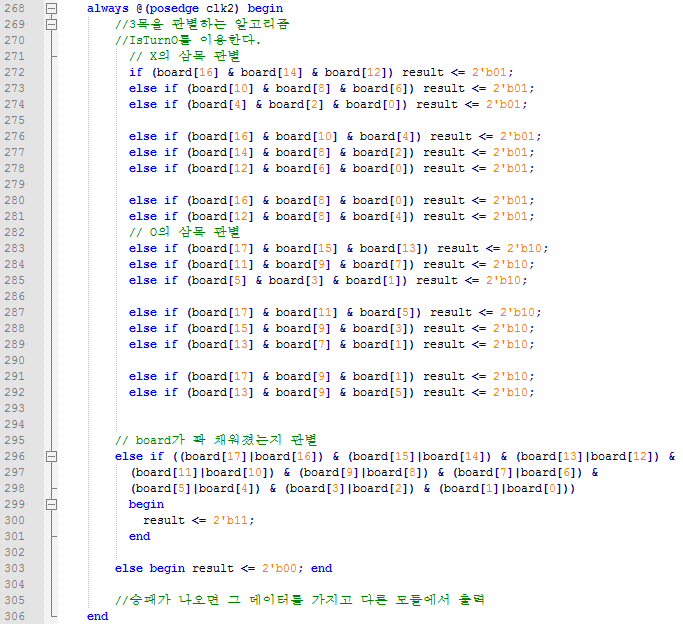


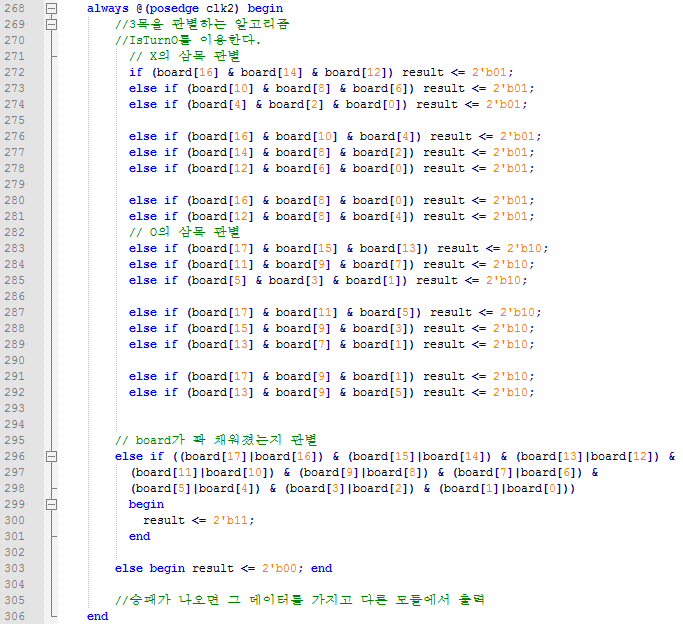
- 3목을 판별 <= 같은 모양으로 한줄이 채워질 경우 result에 값이 할당됨

- X 모양으로 찰 경우 -> result=1(X 승)

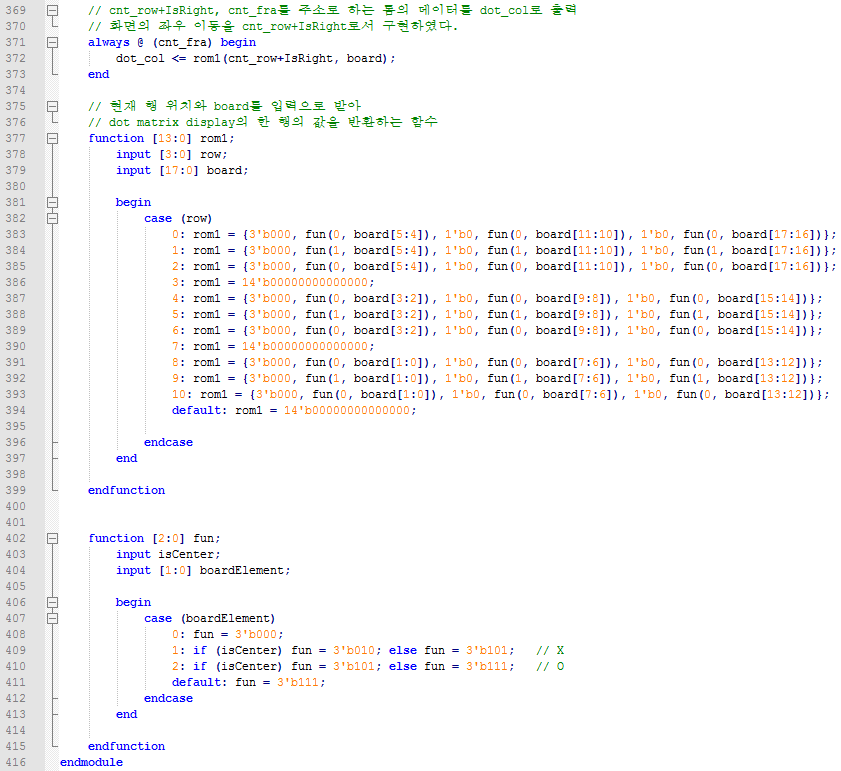
- O 모양으로 찰 경우 -> result=2(O 승)

- 먼저 3목을 완성하는 경우 없이 모든 board가 다 차면 result=3(동점)

- 3목이 완성되지 않고 board에 값이 다 차지 않은 경우 : result=0(게임 중) 

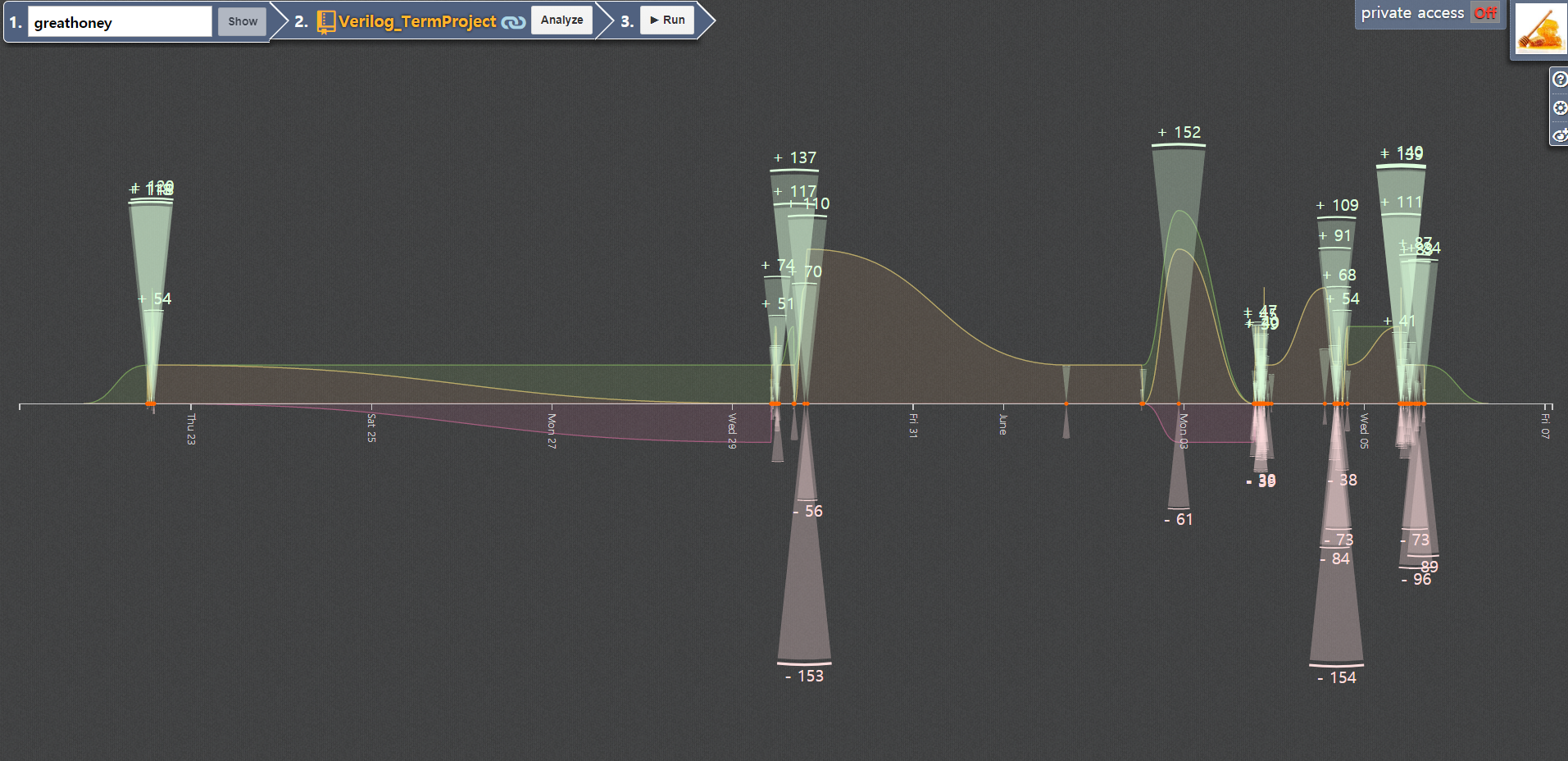


- 벡터 Board로부터 dot matrix display에 화면에 출력하기 위한 구문과 rom 함수



1. 모임 일지

Github를 이용하여 원활한 코드 공유와 더불어 효율적인 작업이 가능했다. 또한 모두가 같이 코드 문제를 보고 해결할 수 있었다. Commit된 날짜가 조에서 만난 시간이었고 만난 시간에 조원들과 많은 코딩을 했음을 확인할 수 있다.



6/3 월 ~ 6/6 목

5/29 수

5/22 수

5/22 수요일 (9:00 ~ 18:00)

Github를 통해 프로젝트 진행 상황을 공유하며 함께 작업할 수 있는 환경을 구축하였다. Tic-tac-toe 프로젝트 이해를 돕기 위해서 dot\_display와 keypad을 미리 주어진 예시를 통하여 엔트리 장비에 구현해보았다. dot\_display의 구현 중 컴파일 에러가 발생하는 부분에 대한 원인을 확인하고 해결하였다. Keypad와 dot\_display를 어떻게 연결시킬지 고민하였다.

5/29 수요일 (9:00 ~ 18:00)

지난 모임에 만들어둔 dot\_display.v와 keypad\_scan.v를 기반으로, 전체 모듈들을 파생시키는 모듈 TTT.v와 메인 화면 상태를 기술한 gameState.v, 게임 화면 상태를 기술한 mainState.v를 구현하였다. 또한 dot\_display에 특정한 형태를 출력하기 위한 rom function을 어떻게 구현할지 고민하였고, 함수와 벡터 결합 연산자를 이용하여 해결하면 된다는 결론을 내렸다.

6/3 월요일

여러가지 컴파일 에러들을 해결하기 위한 노력을 하였다. 여러 개의 모듈을 단위로 verilog파일을 분할하여 만들었었는데, 모듈들의 입출력 연결에서 팬아웃의 문제가 발생하여 이를 해결하려 하였으나 실패하였고, 여러 모듈들을 하나의 모듈로 통합하여 작성하도록 수정하게 되었다.

6/4 화요일 ~ 6/5 수요일

하나의 레지스터의 값을 복수의 always 구문에서 값을 변경해서는 안된다는 사실을 알게되었고, 이를 해결하기 위해 always 구문들을 크게 재구성하고 통합하였다.

6/6 목요일

Result 벡터를 스칼라 레지스터로 선언한 것에 의한 오류로 장장 3시간동안 헤매었다. 블록단위 로 하나씩 주석처리를 해가면서 오류가 어디에 있는지 알아보려 하였으나 실패하였고, result를 구하는 부분의 if else 문에 오류가 있나 싶어 switch문으로 리팩터링해보았으나 결과는 그대로 였다. 마음을 가라앉히고 천천히 처음부터 다시 보는 중에 선언을 잘못한것을 발견하였고, 그 하나를 고침으로 모두 잘 동작하게 되었다.

1. 프로젝트를 마치며

* 윤대헌

컴파일 할 때 생각치도 못한 오류가 나서 힘들었지만, 내가 설계하고자 하는 바가 실제로 구동했을 때 성취감을 얻었고 재미있었다. 순차적으로 실행되는 프로그래밍이 아닌 병렬적으로 실행되어서 그런 부분에 대해 여러가지를 생각하게 되었고 하드웨어 기술 프로그래밍인 만큼 고려해야 할 부분도 많아서 흥미로웠다.

* 김도훈

하드웨어를 제어하기 위한 프로그래밍을 해보았는데, 작성한 결과를 컴퓨터 화면이 아니라 실제 기기에서의 작동을 보는 것이 색다른 쾌감이 있었다. 또한 팀원들과 협업하고 코드를 디버깅하는 과정에서 오류를 찾아가며 해결 방법을 모색하면서 HDL과 FPGA에 대해 더 이해할 수 있었다.

* 김명지

처음에는 어떻게 해야 할지 막막했지만 주제가 정해지고 여러 작업을 거쳐 원하는 결과가 실행되었을 때 신기하고 뿌듯했다. 프로그래밍 능력이 미숙해 어려움도 있었지만 조원들에게 많은 도움을 받았고, 프로젝트를 하면서 많은 것을 얻고 알아갈 수 있었다.