# 논리 회로 설계 및 실험


## 2인 대전 아케이드 게임기

## 최종 보고서

4조
201814320 윤현지
201814331 이해민

1. 설계 시나리오

1) 게임 시작 전

(1) 보드에 전원이 연결되면 LCD에 "On switch 8 to start game"을 출력한다.

(2) Dip Switch 8번을 올리면 게임을 시작, ROUND 1이 시작된다.

2) ROUND 1 - 가위바위보

(1) Player 1, 2가 번갈아 가며 보드와 각각 3번의 가위바위보 대결을 한다.

(2) Player는 가위(1), 바위(2), 보(3) 중 하나를 선택하여 Dip Switch를 올린다.

(3) Dip Switch를 올리면 FULL COLOR LED의 색으로 승(GREEN) / 무승부(BLUE) / 패(RED)를 확인할 수 있으며, Player가 승리할 시 해당 Player의 LED가 하나 켜진다.

(4) 올렸던 Dip Switch 1, 2, 3을 내리면 Player 차례가 넘어가며 다음 판을 진행한다.

(5) 6판을 모두 끝나면 ROUND 1 종료, ROUND 2로 넘어간다.

3) ROUND 2 – 흑과 백

(1) Player 1, 2가 번갈아가며 매 라운드에 포인트를 Keypad 입력으로 제시한다.

(2) 사용한 포인트가 한 자릿수일 경우 검정색(보라색), 두 자릿수일 경우 흰색이 Full Color LED에 나온다.

(3) 8 array LED 중 1~4는 Player 1, 5~8은 Player 2의 표시등으로 사용하며, 사용 포인트 20마다 오른쪽부터 1개씩 꺼진다.

(4) 더 높은 포인트를 입력한 Player가 승점 1점을 획득, 8 array Segment에 표시된다.

(5) 한 Player가 승점 5점을 획득하면 그 즉시 ROUND 2 종료, ROUND 3으로 넘어간다.

4) ROUND 3 - 숫자야구

(1) Player 1, 2가 번갈아 가며 Keypad로 숫자를 입력하며, 랜덤으로 정해진 보드의 숫자를 맞힌다.

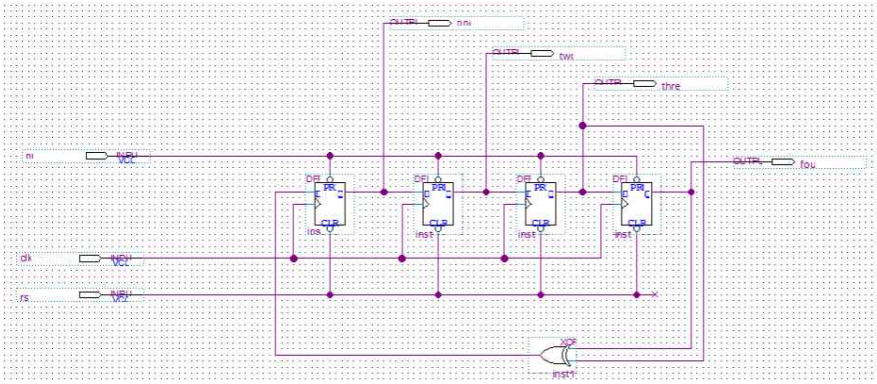
(2) 4자리 숫자가 입력되면 보드는 Strike, Ball을 판단하여 Strike는 8 array LED 1~4번을 왼쪽부터 개수만큼 켜고, Ball은 5~8번을 오른쪽부터 개수만큼 켠다.

(3) 8 array LED 1~4번이 모두 켜질 시, 즉 4 Strike가 될 시 ROUND 3을 종료한다.

2. ROUND 1 – 가위바위보(project2.bdf)

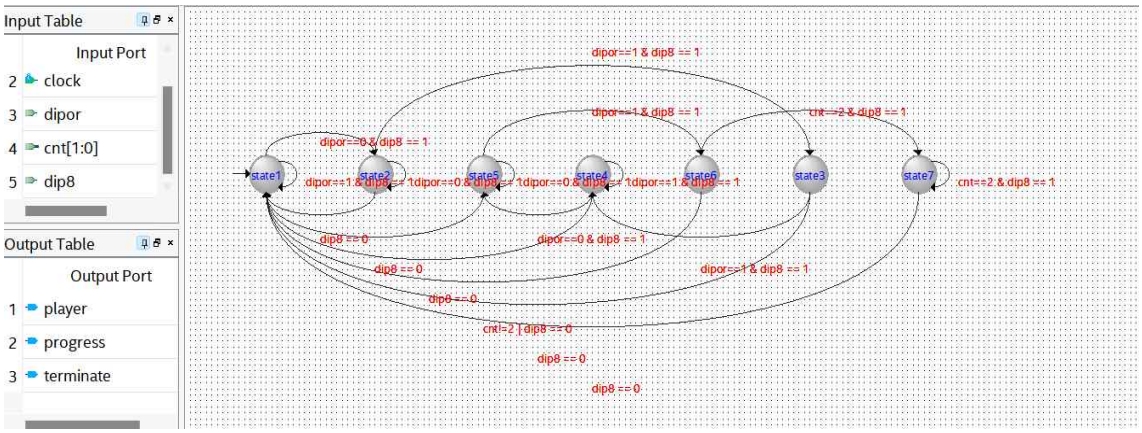
1) 회로 설계

(1) 난수발생기



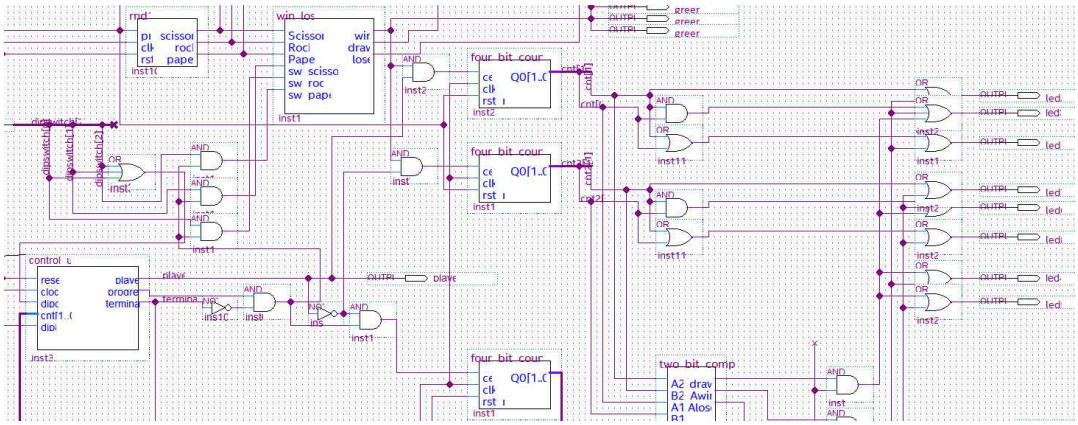
플레이어가 보드와 가위바위보 대결을 하는데, 이 때 보드가 무작위로 가위바위보를 결정할 수 있도록 하는 난수발생기이다. Shift Register를 사용하여 Input pr이 0에서 1로 바뀌는 순간 4개의 Output에서 0 또는 1이 출력된다. pr은 게임이 시작할 때 Dip Switch 8번을 올리도록 하여 0에서 1로 바꾸어 주게 되는데 이때 0001에서 1111까지 총 15가지 경우의 수가 나온다. 이 15가지 경우의 수를 5개씩 묶어 보드의 가위바위보를 결정하였다.

(2) control unit



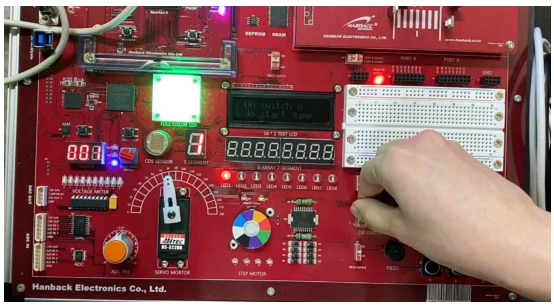

플레이어는 Dip Switch 1, 2, 3번 중 하나를 올려 가위바위보를 결정한다. 그리고 스위치를 내리면 다음 플레이어 차례로 넘어간다. 이를 플레이어당 세 번씩 할 수 있도록 만든 state machine이다. 그리고 Dip Switch 8번을 내리면 게임을 처음부터 할 수 있도록 설계하였다. player가 0이고 progress가 1이면 밖의 4진 Counter의 ce에 1을 넣어줘서 가위바위보 몇 번을 진행하였는지 셀 수 있게 한다. 그리고 state machine에서 그 count 값을 입력받아 2가 되는 순간 마지막 게임을 하고 output terminate에 1을 출력하여 가위바위보 게임을 종료하도록 한다.

(3) 그 외

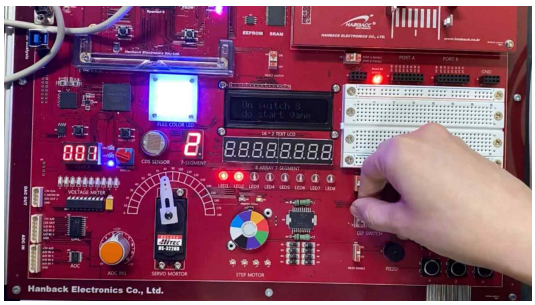


win_lose 모듈을 만들어서 보드의 가위바위보와 내가 Dip Switch로 입력한 가위바위보를 입력 받아 win, draw, lose를 결정하게 한다. 그리고 win count를 플레이어마다 4진 Counter에 연결해 세고 8 array LED에 연결하여 개수에 따라 켜지게 한다.

2) 동작 설명



↳ Player 1이 가위바위보를 이겼을 시



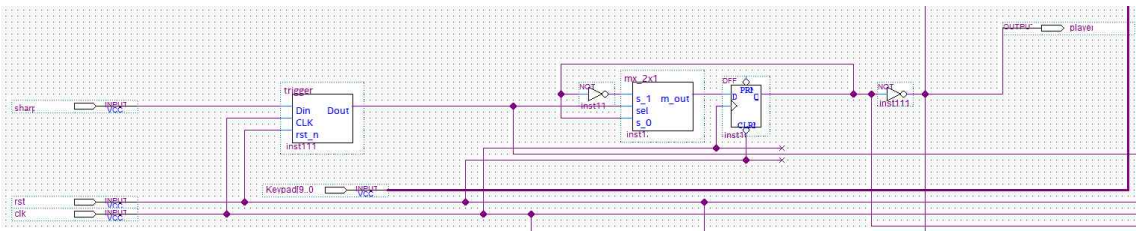
↳ Player 2가 가위바위보를 비겼을 시

7-Segment에는 지금 플레이하고 있는 플레이어가 출력되고, Dip Switch 1, 2, 3 중 하나만 올리면 승패 결과에 따라 Full color LED에 불이 켜진다. 이기면 초록색, 비기면 파란색, 지면 빨간색이 나온다. 그리고 이기게 되면 Player 1은 왼쪽, Player 2는 오른쪽부터 8 array LED가 하나씩 점등된다. 그리고 3번씩 게임을 마치면 이긴 플레이어 쪽의 LED가 모두 점등되면서 다음 라운드로 넘어간다. 비겼을 경우 가운데 4개의 LED가 점등된다.

3. ROUND 2 – 흑과 백(rnd2_player.bdf)

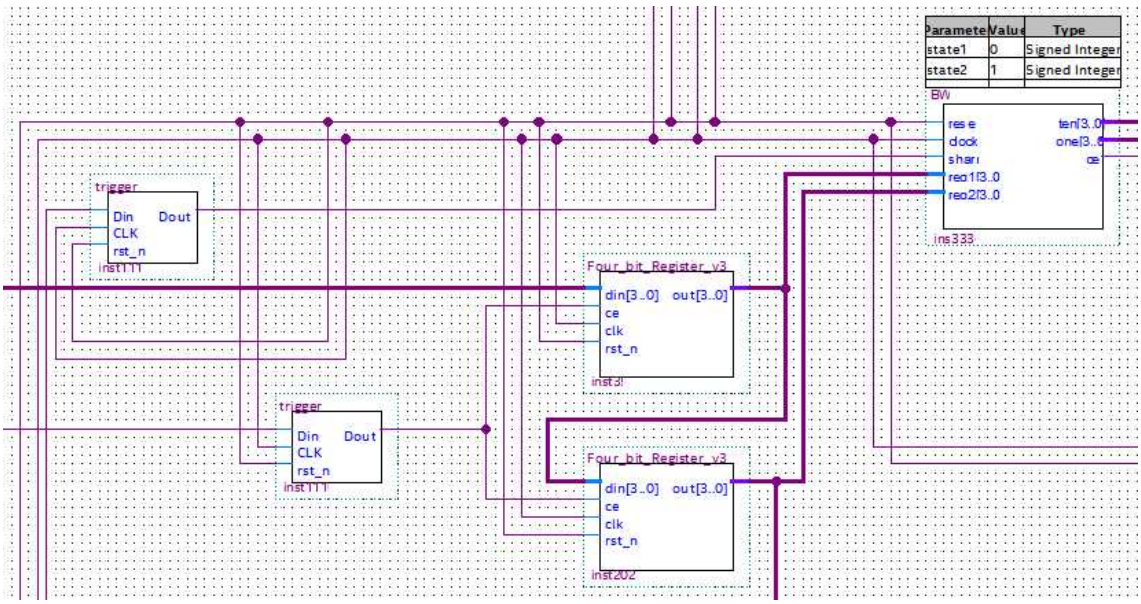
1) 회로 설계

(1) 플레이어 변경




포인트 입력 후 #을 누르면 Player가 1에서 2로, 2에서 1로 변경되어야 한다.
Player1을 신호 0으로, Player2를 신호 1로 보았다.

(2) 포인트 입력



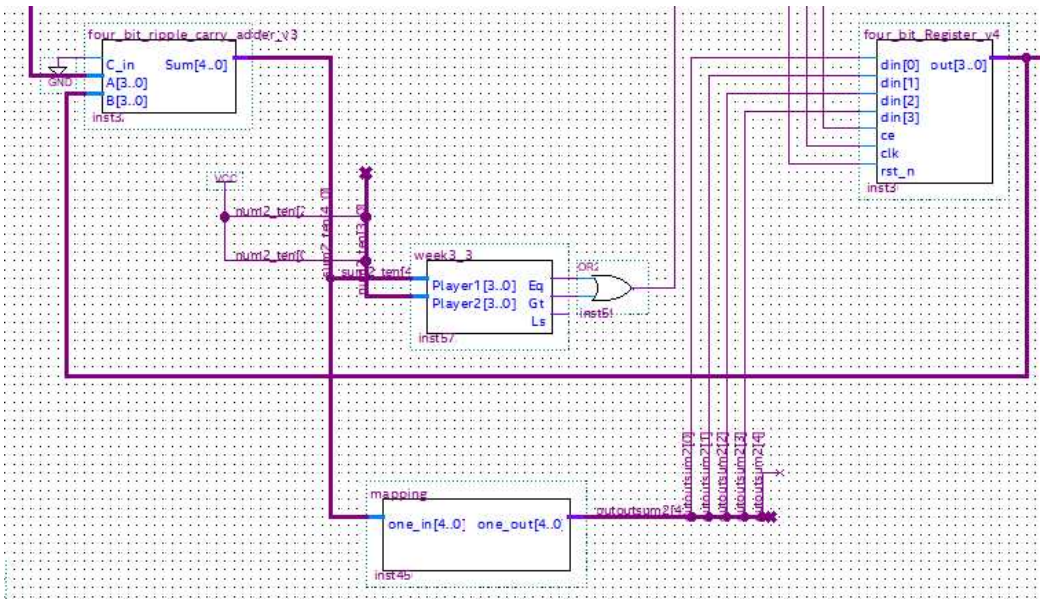

오른쪽에 있는 Trigger를 사용하여 Keypad를 한 번 누를 때마다 4bit Register의 ce가 1이 되어 값이 들어간다. Register의 위(reg1)에서부터 밑(reg2)으로 값이 차례차례 들어가 총 2자리의 값이 들어갈 수 있도록 하였다.

이때, Player가 포인트를 입력할 때마다 직전에 상대 Player가 입력했던 포인트 값이 Register에 남아서 reg1에 있던 값이 reg2로 밀린다. 따라서 Register의 값을 실시간으로 바로 쓰는 것이 아닌 #이 눌릴 때만 전송되도록 해야 한다.

이를 위해 왼쪽 Trigger를 Player 값과 연결하여 본인 차례에만 BW에서 Register 값을 내보내도록 하였다.

또한 #이 눌려짐과 동시에 BW의 Output인 ce가 1로 나가도록 하였다.

(3) 사용한 포인트 누적 계산 – 일의 자리



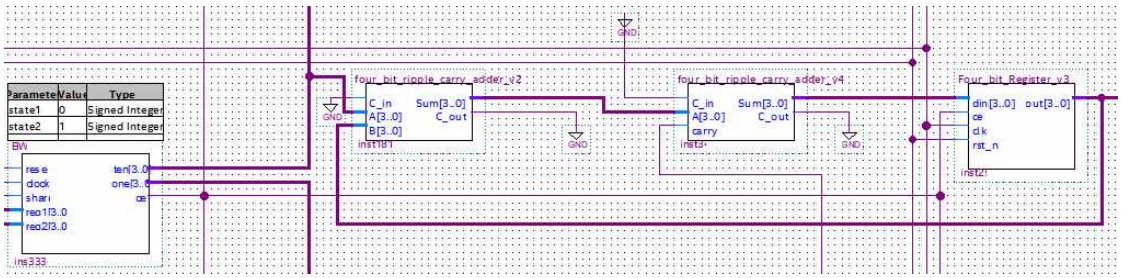

#이 눌러짐과 동시에 reg2의 값을 십의 자리, reg1의 값을 일의 자리로 한 clock 동안 내보내면, 위의 사진에서는 일의 자리만을 들고 와 Player의 누적 포인트의 일의 자리를 저장하고 있던 Register 값과 덧셈 연산을 하게 된다.

덧셈 연산 후에는 mapping에서 10 미만의 숫자는 그대로 값이 나가고, 10 이상의 숫자는 –10 한 값이 나가게 하였다.

또한 일의 자리 덧셈 결과 10 이상이 될 경우 십의 자리에 1을 더해줘야 하므로 비교기를 이용하여 carry 값이 발생 되게 하였다.

위의 과정이 모두 끝나면 새로운 값이 Register에 저장되는데, 이때 Register의 ce를 BW의 Output인 ce와 연결하여 #이 눌리는 동안만 Register가 활성화되도록 하였다.

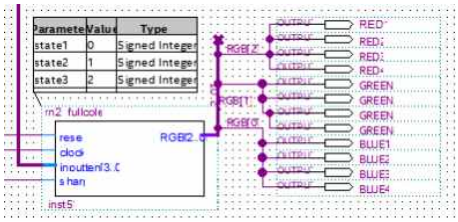
(4) 사용한 포인트 누적 계산 – 십의 자리




#이 눌러짐과 동시에 reg2의 값을 십의 자리, reg1의 값을 일의 자리로 한 clock 동안 내보내면, 위의 사진에서는 십의 자리만을 들고 와 Player의 누적 포인트의 십의 자리를 저장하고 있던 Register 값과 덧셈 연산을 하게 된다.

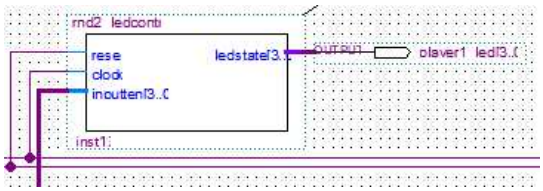
덧셈 연산 후에는 일의 자리에서 발생한 carry 값을 더하는 연산을 추가로 진행 후, 새로운 값이 Register에 저장되는데, 이때 Register의 ce를 BW의 Output인 ce와 연결하여 #이 눌리는 동안만 Register가 활성화되도록 하였다.

(5) 흑과 백 표시



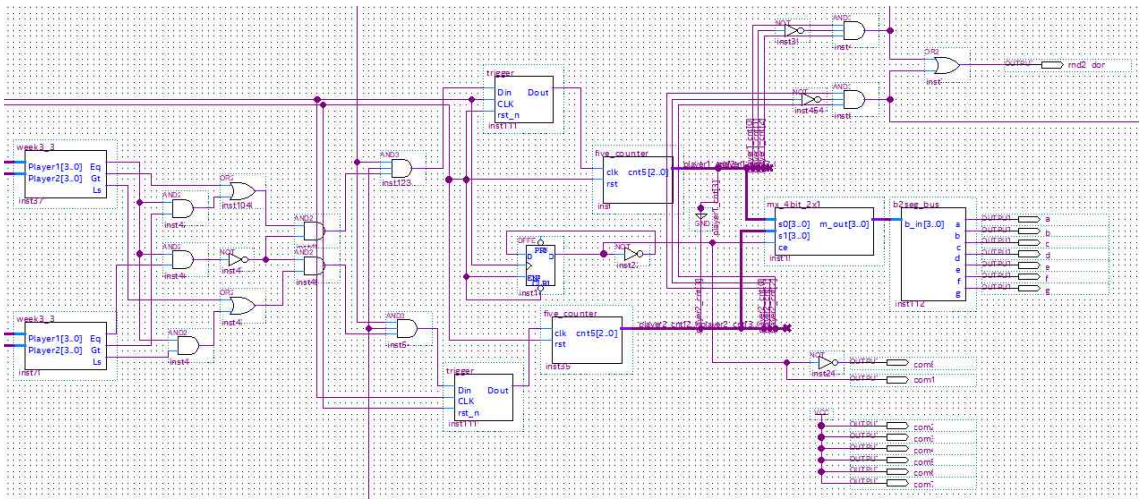

십의 자리를 뜻하는 reg2 값을 실시간으로 받다가 #이 눌리는 순간 들어온 숫자가 0인지 0이 아닌지 판단하여 0이면(한자릿수) 보라색(RED + BLUE), 0이 아니면(두자릿수) 흰색(RED + GREEN + BLUE)으로 Full Color LED 색이 나오도록 하였다.

(6) led 표시




누적 포인트의 십의 자리 숫자를 입력값으로 받아 사용 포인트 20마다 led 하나를 끄는 동작을 수행한다.

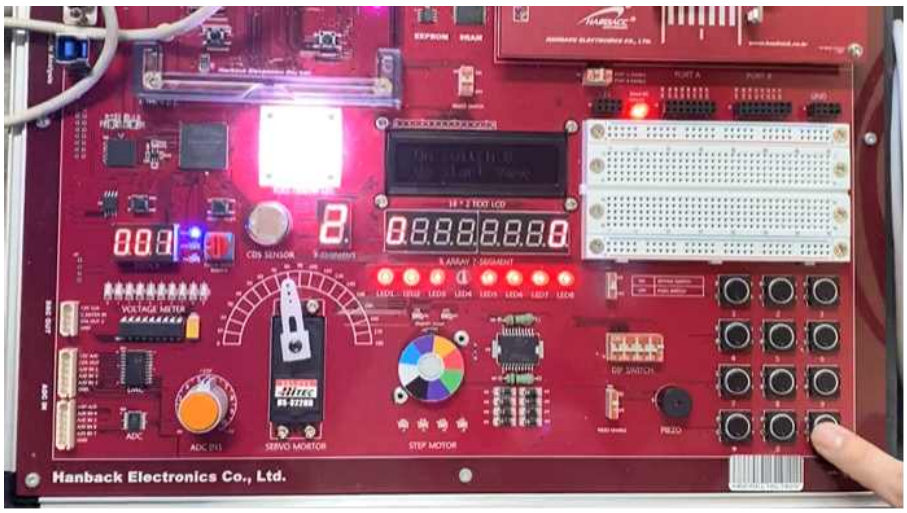
(7) 승점 표시 및 ROUND 2 종료




Player 1과 2가 실시간으로 입력한 포인트를 십의 자리끼리, 일의 자리끼리 비교하여 누가 이겼는지 판별하여 3bit Counter를 통해 해당 Player의 승점을 하나 올리고 8 array Segment에 표시한다.

Player 둘 중 한 명의 승점이 먼저 5점이 되는 순간, ROUND 2를 종료한다.

2) 동작 설명



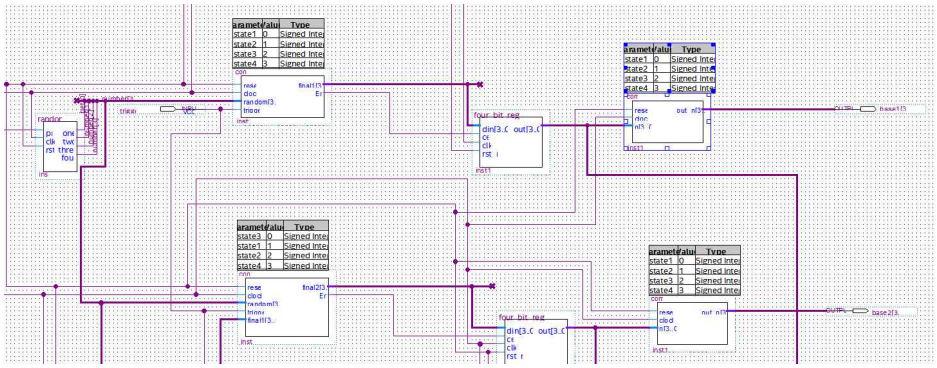

Player 1이 두 자릿수 20을 입력하여 Full Color LED가 흰색으로 켜지고, Player 1의 LED 중 가장 오른쪽 한 개가 꺼진다.
(사진 상에서 포인트를 입력하고 #을 누르는 즉시 Player가 교체되어 2라고 표시됨)


Player 2가 한 자릿수 3을 입력하여 Full Color LED가 보라색으로 켜지고, Player 2의 LED는 꺼지지 않는다.
또한 Player 1이 20을 입력, Player 2가 3을 입력하여 Player 1이 더 큰 포인트를 소비하였으므로 Player 1의 승점이 하나 올라간다.

4. ROUND 3 – 숫자야구(test5.bdf)

1) 회로 설계

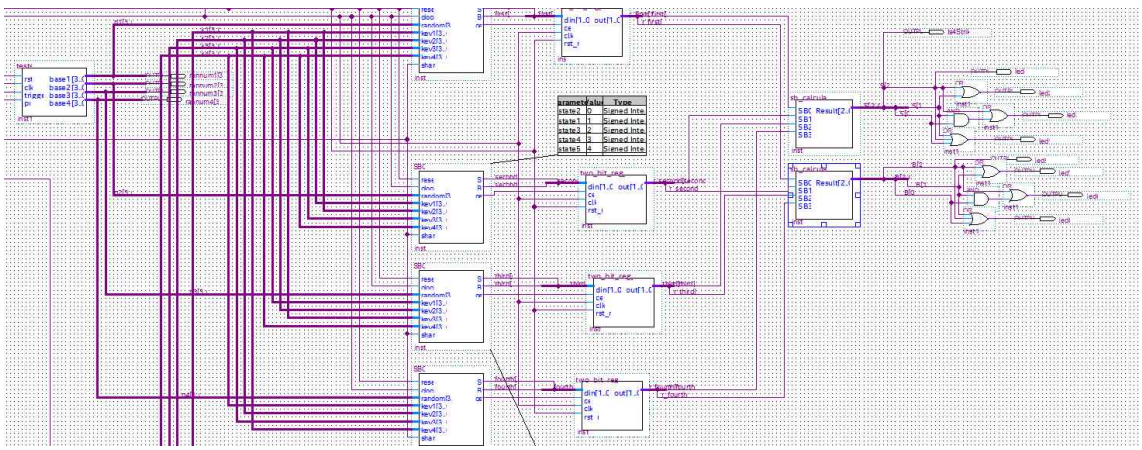
(1) 난수 발생 회로



난수발생기는 가위바위보 1. 1) (1)과 동일하다. 숫자야구의 난수는 0~9까지의 숫자 4개를 출력해야 하는데, 그 4개의 숫자는 겹치지 않아야 한다. 난수발생기에서 나온 0001 ~ 1111 까지의 숫자를 control unit을 통하여 조절한다. 첫 번째 Player가 Keypad에 네 자리 숫자를 입력하는 순간 Trigger가 활성화 되고 난수가 발생한다. 숫자는 0~9가 나와야 하는데 난수 발생은 1에서 15까지이므로 들어온 난수가 11 이상일 경우 다음 clock에서 난수를 결정하게 한다. 그리고 10이 들어 온 경우 후에 0으로 바꾸어 준다.

네 자리 중 한 자리가 결정되면 다른 숫자들은 앞에 결정된 숫자들과 비교하여, 같을 경우 다음 clock에서 난수를 결정하도록 한다. 이를 state machine을 이용하여 설계하였다. 그렇게 결정된 4개의 4bit 숫자를 4개의 Register에 저장한다.

(2) Strike, Ball counting



Player는 네 자리수를 Keypad로 입력하고 #버튼을 누르면 위에서 발생한 난수와 비교한다. 자리와 숫자가 모두 같으면 Strike, 자리만 다르면 Ball로 네 자리를 모두 결정한다. 이는 state machine을 사용하였다. smf 파일로 만든 control unit에 한계가 있어서 Verilog를 추후에 수정하였다.

같은 모듈을 사용하여 난수의 한 자리씩, 네 번 비교한다. 자리와 숫자 같을 경우 S에 1을 주고, 다른 자리와 숫자가 같을 경우 B에 1을 준다. 그리고 ce에 1을 주어 2bit Register에 SB를 저장하도록 한다. 이렇게 네 자리를 모두 비교하고 네 개의 2bit Register에 있는 값들

을 더하여 총 Strike와 Ball을 계산한다. 그리고 Strike는 8 array LED의 왼쪽 네 개에, Ball은 오른쪽 네 개에 연결하여 볼 수 있게 하였다. 4 Strike가 되면 게임을 종료한다.

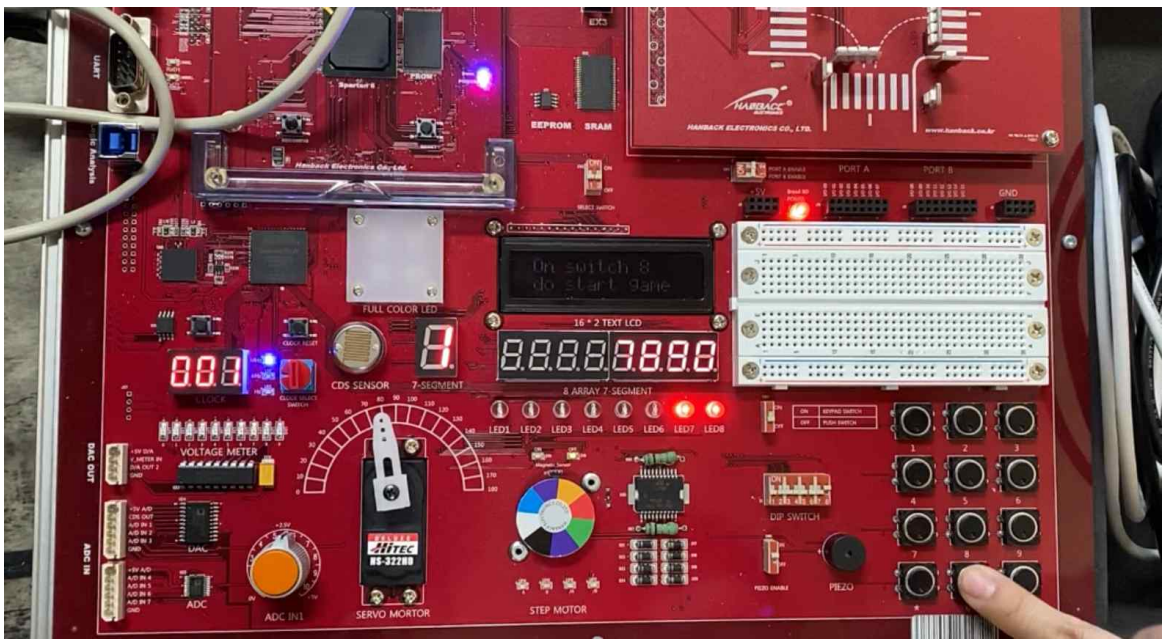
```
case (fstate)
    state2: begin
        if ((random[3:0] == key1[3:0]))
            reg_fstate <= state3;
        else if ((((random[3:0] == key2[3:0]) | (random[3:0] == key3[3:0])) | (random[3:0] == key4[3:0])))
            reg_fstate <= state4;
        // Inserting 'else' block to prevent latch inference
        else
            reg_fstate <= state5;

        S <= 1'b0;

        B <= 1'b0;

        ce <= 1'b0;
    end
    state1: begin
        if ((sharp == 1'b1))
            reg_fstate <= state2;
        else if ((sharp == 1'b0))
            reg_fstate <= state1;
        // Inserting 'else' block to prevent latch inference
        else
            reg_fstate <= state1;
```

2) 동작 설명

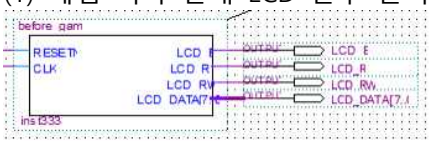



Keypad를 decimal-to-binary에 연결하고 Shift Register를 이용하여 네 개의 4bit Register에 있는 값을 8 array Segment에 출력한다. 그리고 플레이어의 차례를 알기 위해서 계속해서 7-Segment에 출력한다. 원하는 숫자 네 자리를 Keypad에 입력하고 #을 누르면 Strike와 Ball의 개수가 8 array LED에 나타난다.

사진에서 오른쪽 두 개의 LED가 켜졌다는 것은 2 Ball을 의미한다. 이렇게 플레이어를 바꾸어 가면서 네 자리 난수가 무엇인지 맞추게 되면 왼쪽 네 개의 LED에 4 Stike라는 의미로 불이 들어오게 된다.

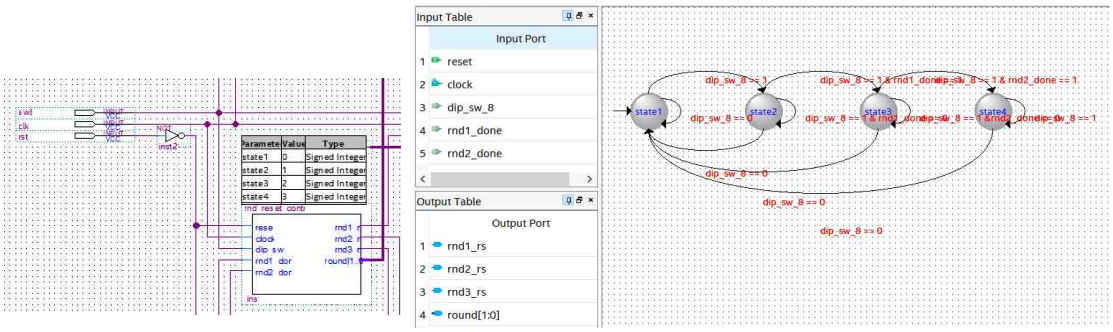
5. ROUND 통합(rnd123.bdf)
위에서 설계한 모든 ROUND를 통합하고, 각 ROUND의 reset를 제어한 회로이다.

1) 회로 설계
(1) 게임 시작 안내 LCD 문구 출력



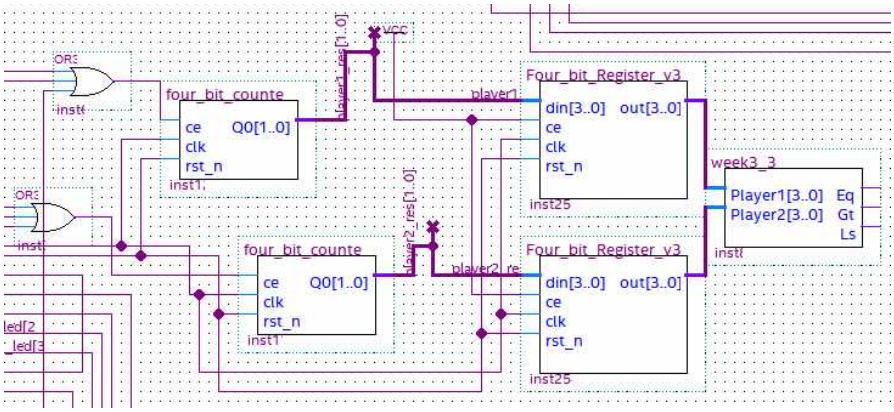

보드를 켜면 "On switch 8 to start game" 문구가 LCD에 뜨도록 하였다.

(2) 각 ROUND reset 제어 및 해당 ROUND 알림



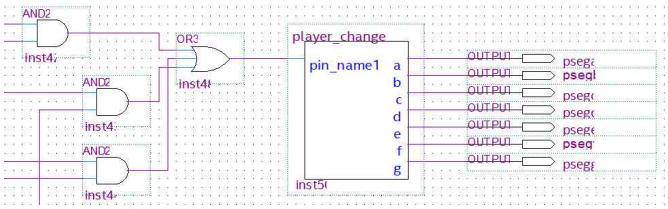

Switch 8번을 올릴 경우 ROUND 1을 시작한다고 간주, State2로 넘어가 rnd1_done(ROUND 1 종료)이 1이 될 때까지 rnd1_rst을 1로 주고, 나머지 ROUND의 rst은 0으로 준다. rnd1_done이 1이 되면 다음 State로 넘어가며, 모든 State에서 dip switch 8번이 0이 될 시 즉시 State1로 돌아간다.
ROUND 2와 3의 reset 제어도 마찬가지로 이루어진다.
또한 ROUND 1, 2, 3이 동작 중일 때 round 값을 각각 0, 1, 2로 주었다.

(3) 스코어 계산



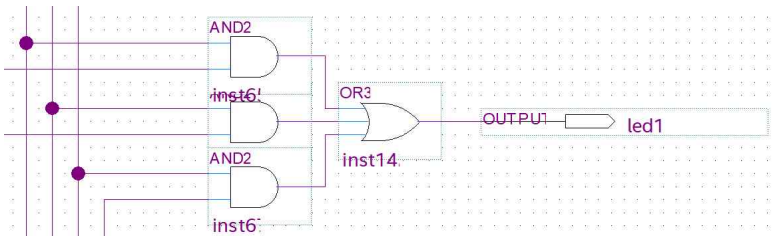

각 ROUND에서 해당 Player가 이길 때마다 2bit Counter를 통해 Register에 우승 횟수를 저장하고, 그 횟수를 비교하였다.

(4) Player 표시



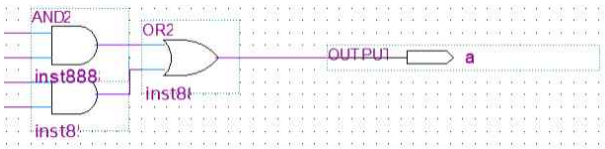

7-Segment에 현재 어느 Player의 차례인지 숫자를 띄운다.

(5) LED 연결



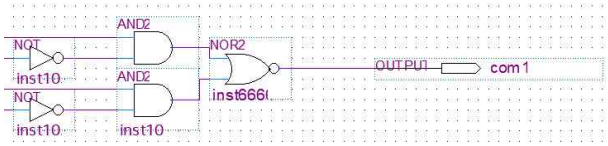

현재 ROUND의 LED 신호가 다른 ROUND의 LED 신호와 겹치지 않도록 하기 위해, 각 ROUND의 LED 신호와 reset 제어 state machine의 Output으로 나오던 round를 AND 연산하였다. 그 후에 그 신호들을 모두 모아 OR 연산하여 해당 LED에 연결하였다.

(6) 8 array Segment 연결



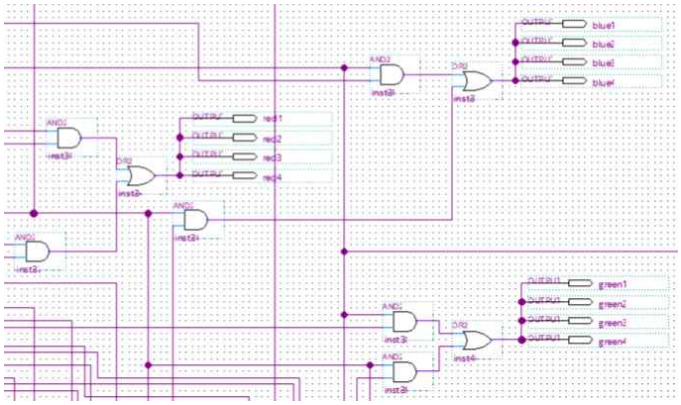

8 array Segment 연결 또한 LED 연결과 마찬가지로 round와 and 연산을 하였다.
8 array Segment는 ROUND 1을 제외한 2와 3에서만 쓰였으므로, 2개만 OR 연산을 하였다.

(7) com 연결



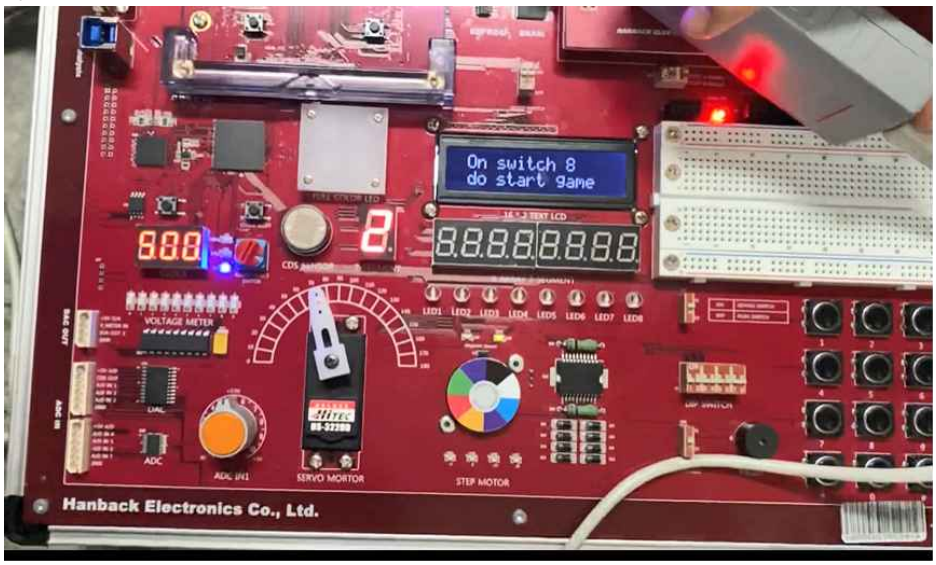

com 연결 또한 마찬가지로 round와 and 연산을 하였다.
com 또한 ROUND 1을 제외한 2와 3에서만 쓰였으므로, 2개만 OR 연산을 하였다.

(8) Full Color LED 연결



Full Color LED 연결 또한 마찬가지로 round와 and 연산을 하였다.
Full Color LED는 ROUND 3을 제외한 1과 2에서만 쓰였으므로, 2개만 OR 연산을 하였다.

2) 동작 설명




↳ 게임 시작 안내 문구 출력

6. 에러 사항

만든 프로젝트에는 따로 에러 사항은 없었으나, 초기 제안서에 비교했을 때 아쉬운 부분들이 존재한다.

1) 총 3라운드에 걸친 승자를 게임 ROUND 3까지 모두 끝낸 후 출력하려고 했으나, 프로젝트 발표 전에 추가하려고 하니 에러가 발생하며 정상적으로 작동하던 세 개의 게임들도 돌아가지 않는 상황이 발생하여 급하게 삭제하였다.

2) ROUND 2 흑과 백은 99 포인트를 가지고 하는 게임인데 99 포인트를 넘게 사용하였을 때의 예외 상황을 프로젝트에 설정해 두지 않아서 무제한 포인트가 되어버렸다.

7. 역할 분담표

| 조원 이름 | 역할 |
|---|---|
| 윤현지 | 게임 ROUND 1, 3 및 보드와 프로젝트 연결 담당 |
| 이해민 | 게임 ROUND 2, 게임 취합 및 보고서 정리 담당 |

크게는 이렇게 나누었으나, 이번 프로젝트는 전부 다 같이 고민하고 설계한 결과물이다.

윤현지 조원이 ROUND 2와 통합 회로에서의 보드와 프로젝트를 연결하는 요소들(LED, LCD, com, Segment)의 회로를 담당하고, 이해민 조원이 ROUND 1, 3에 쓰인 난수발생기를 제작, 난수 값 제어에 아이디어를 제공하는 등 자주 만나서 모든 부분에 대해 어떤 식으로 설계할지 회의하고, 설계 후 서로 도와가며 함께 문제 사항을 해결하였다.

※ 본 텀프로젝트의 project 파일명은 project1.qpf입니다. 비슷한 이름의 project.qpf 파일이 아님을 알려드립니다.