프로젝트 최종보고서

디지털 시스템 설계

조: zerone

조원: 김유진, 양서연, 오세림

ㄱ. 주제 및 목적

이번 프로젝트는 수업시간에 배운 Finite State Machine의 작동원리를 이해하고 state diagram을 직접 그려보는 활동을 통해 sequential circuit에 대한 이해도를 높이는 것에 목적이 있는 프로젝트이다. Sequential circuit에 대한 지식을 적용하여 사용자가 컴퓨터와 베스킨라빈스 31 게임을 진행할 수 있는 프로그램을 구현하는 것을 본 프로젝트의 주제로 한다. 또한, LED와 7-Segment 등을 이용함으로써 코드 구현과 더불어 실제 부품의 이해도를 높이고 적용해본다.

ㄴ. 배경 지식

i) Combinational Logic Circuit

Combinational logic circuit은 입력에 따라 출력이 즉시 계산되는 논리 회로이다. 논리 게이트들만의 연결로 이루어지며 기억소자가 없다는 특징을 지닌다. 논리 게이트들의 종류로는 AND(논리곱), OR(논리합), NOT(논리부정), NAND, NOR, XOR, XNOR 게이트가 있다. Combinational logic circuit을 기술하는 방법에는 크게 세가지가 있는데 첫째는 입출력 동작에 대한 논리적 관계를 표로 정리하여 나타낸 진리표가 있다. 둘째는 bool식이 있는데 bool algebra에 의해 표현된 식으로 bool 변수에 의해 계산이 가능한 논리식으로 표현하는 방식이다. 논리식을 간소화하는 방법으로는 Karnaugh Map과 Quine-McClusky 방법이 있다. 마지막으로는 게이트를 사용해 입출력의 논리적 관계를 논리 게이트 및 플립플롭으로 구성해 그림으로 표현한 회로도로 표현하는 방식이다.

- Multiplexer (MUX)

멀티플렉싱은 다수의 정보 장치의 데이터를 소수의 채널이나 선을 통하여 선택적으로 전송하는 것을 의미하는데, 멀티플렉서는 선택 신호에 따라 여러 입력 신호 중 하나를 골라 출력하는 회로이다. 주로 2^n개의 입력 신호를 n개의 선택 신호로 선택하고 그 중 하나를 출력한다.

- Decoder

디 디코더는 n개의 이진 입력을 받아 2^n개의 출력을 가지는 회로이다. 실제 디코더 소자에는 n개의 입력뿐만 아니라 EN, 혹은 Enable입력이 추가로 존재하는데, 이는 디코더를 켜거나 끄기 위해서 사용한다. 하나의 출력만 1이 되도록 출력한다는 특징을 지닌다.

ii) Sequential Logic Circuit

Sequential logic circuit은 입력 및 현재 상태에 따라 출력 및 다음 상태가 결정되는 논리 회로로, 즉 현재의 입력과 과거의 출력 상태 모두에 의거해 출력이 결정된다. 특징으로는 피드백 경로가 있고 기억성이 있어 일련의 연산 사이에 정보를 저장할 수 있는 회로가 구성된다는 점이 있다.

Sequential logic circuit은 combinational logic circuit과 플립플롭 등의 조합으로 다양한 형태로 만들 수 있다. 유한한 기억장치를 가지고 있는 기계라 Finite State Machine(FSM)이라고도 불린다.

- Register
Register는 클락을 공유하는 다수의 플립플롭들이 묶인 sequential logic circuit으로 상태 정보를 저장하는 기능을 제공한다. n개의 플립플롭을 갖는 레지스터인 경우에 2^n개의 서로 다른 상태들이 존재하고, n비트 정보의 저장이 가능하다.

- Shift Register
단방향 또는 양방향으로 매 클락마다 한 칸씩 이진 정보를 이동시킬 수 있는 register이다. 설계에 따라 데이터 비트를 좌측 또는 우측으로 이동시키는데, 한 비트씩 데이터가 shift되거나 새로운 데이터가 입력되어 기존 데이터가 한 자리씩 이동하게 된다.

- Counter
Counter는 다수의 플립플롭들이 미리 정해진 순서대로 상태가 변하는 sequential logic circuit로 특정 순서 또는 카운트를 나타내는 데 사용된다.

- Latch와 Flip-flop
전원이 공급되고 있는 한, 상태의 변화를 위한 신호가 발생할 때까지 현재의 상태를 그대로 유지하는 논리 회로이다. 래치는 입력의 변화가 바로 반영되어 출력도 바뀌는 비동기 회로이고 플립플롭은 입력이 바뀌더라도 출력이 클락에 맞춰 반영되는 동기 회로이다.

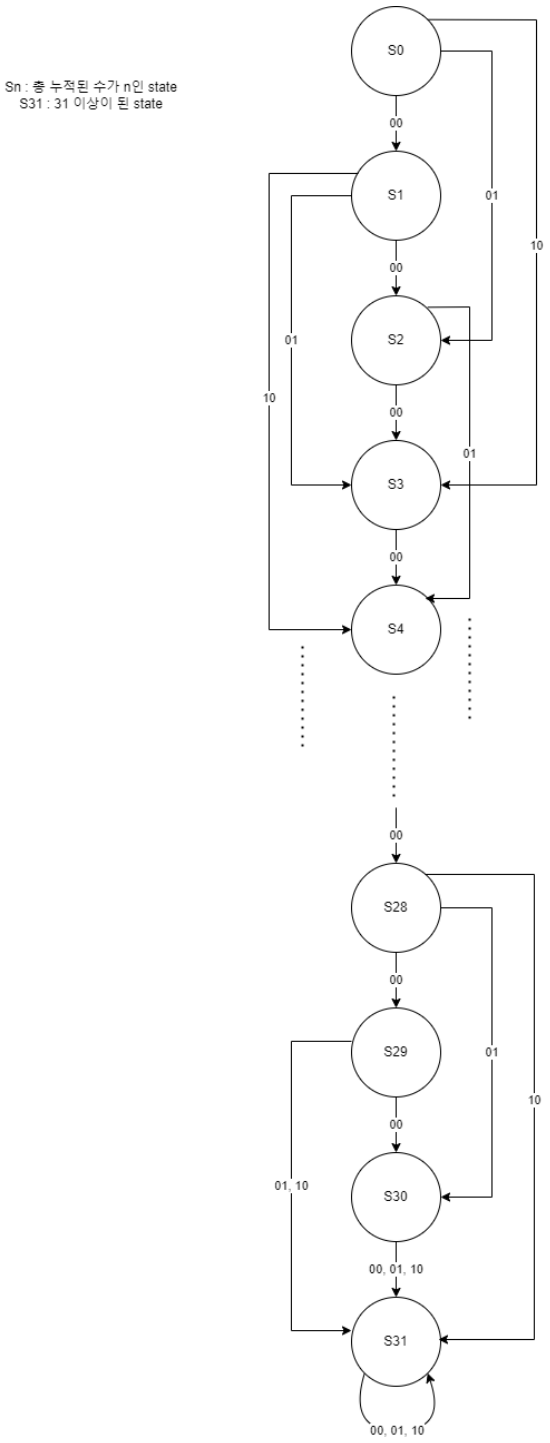
ㄷ. State transition diagram 및 State machine 설계

게임의 전체적인 진행에 따른 State transition table은 다음과 같다.

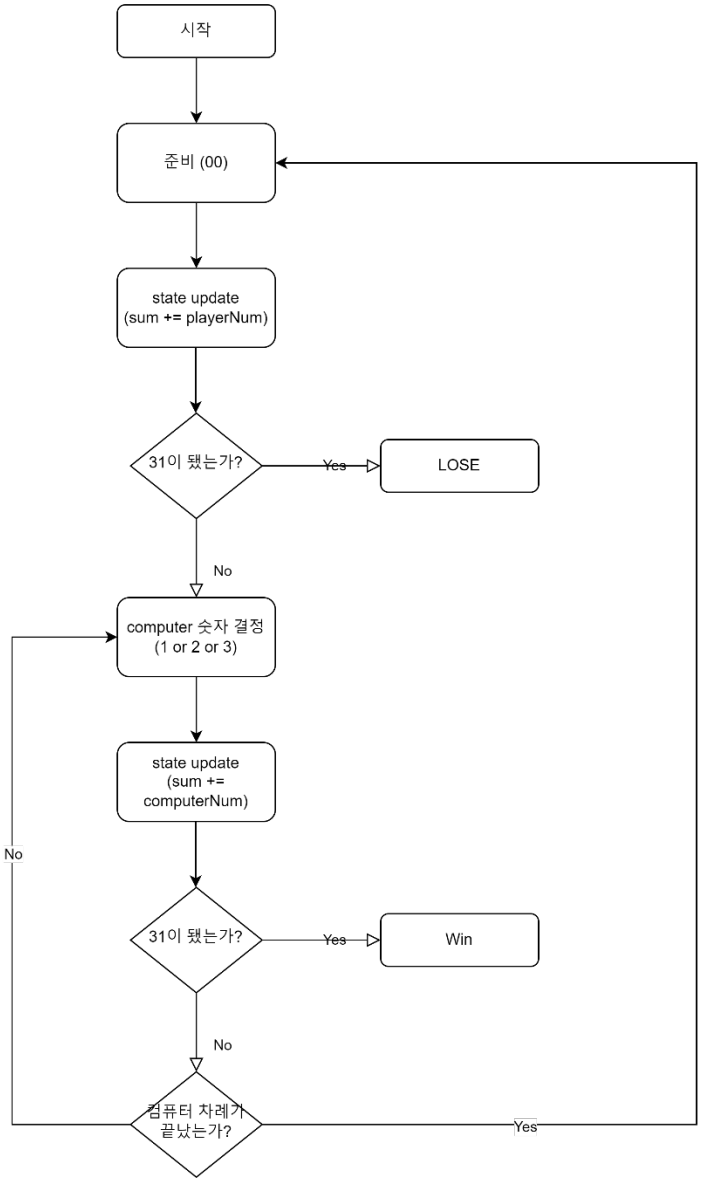
Present State	Next State			Output
	X = 0 0	0 1	1 0	
S_0	S_1	S_2	S_3	0
S_1	S_2	S_3	S_4	0
S_2	S_3	S_4	S_5	0
S_3	S_4	S_5	S_6	0
⋮				
S_{28}	S_{29}	S_{30}	S_{31}	0
S_{29}	S_{30}	S_{31}	S_{31}	0
S_{30}	S_{31}	S_{31}	S_{31}	0
S_{31}	S_{31}	S_{31}	S_{31}	1

이때 S_n은 현재 누적된 수가 n인 상태를 나타내며, S_{31}은 누적 수가 31 이상인 상태를 의미한다. 입력값 X는 컴퓨터 혹은 사용자의 입력을 나타낸다. 00, 01, 10 각각 입력값 1, 2, 3을 의미하고 입력값에 따라 Next State가 결정된다. 출력값 Output은 게임이 종료되었는지를 판별하는 비트로, S_{31}을 제외하면 나머지 상태에서는 0을 유지하고, Output이 1이 되면(누적 수 31 이상이 되면) 승자 판별 단계로 넘어간다. 또한, 해당 State transition table에서 입력값과 출력값이 모두 동일한 상태는 존재하지 않으므로 State Reduction은 이뤄지지 않는다.

위 표를 바탕으로 작성한 State transition diagram은 아래와 같다.



ㄹ. 전체적인 동작 설명



전체적인 동작 시퀀스의 flowchart는 위와 같다.
기계를 켜서 시작하면, 게임 준비 상태, 즉 누적 값이 0인 상태에 들어간다. 먼저 플레이어부터 게임을 시작하며, 이후 차례가 바뀌어 컴퓨터가 1~3의 값 중 랜덤한 값을 입력한다. 이를 컴퓨터 플레이어 수만큼 반복한다. 각 값을 입력 받을 때 마다, 누적값이 31이 되었는가 확인한다. 누적값이 31이 된 순간 입력한 쪽이 사용자라면 Lose를, 컴퓨터라면 Win을 기계에 출력한다.

ㅁ. 입출력 방법 및 동작 설명

대략적인 기계 구상은 아래와 같다.

(1) 시작 전

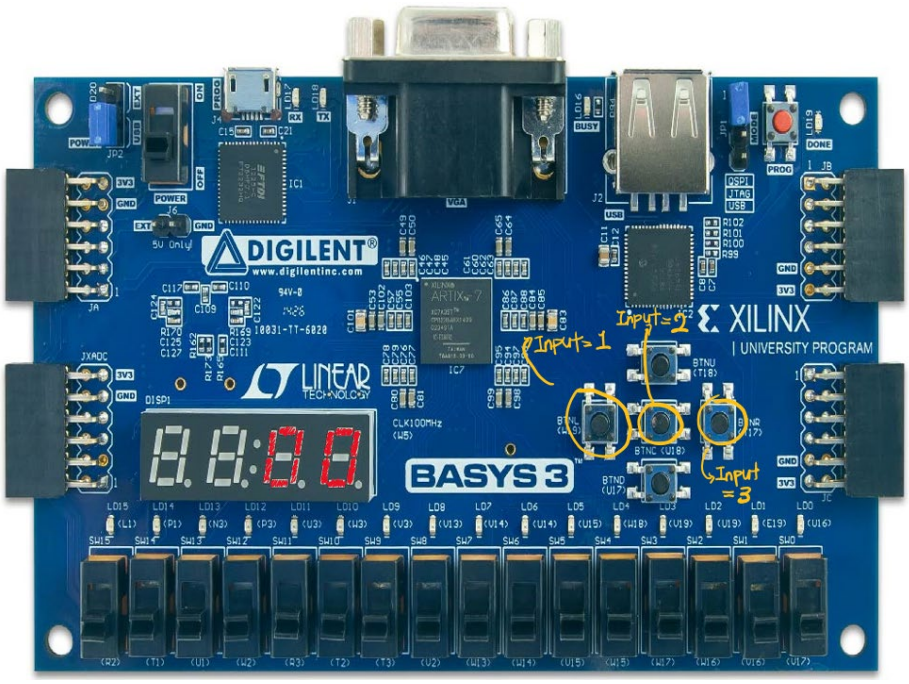


그림1. 게임 실행을 위한 FPGA 기계

시작 전에는 위의 그림과 같이 7 segment LED는 null 상태에 놓여있다.

(2) 사용자의 숫자 결정

사용자는 오른쪽에 위치한 버튼을 이용해 1 ~ 3의 숫자 중 하나를 선택한다. 사용자가 선택한 숫자는 왼쪽 7 segment LED에 업데이트 된다. 그림의 예시는 사용자가 맨 오른쪽 버튼을 눌러 숫자 3을 골라 왼쪽 LED에 3이라는 숫자가 업데이트 된 상황이다.

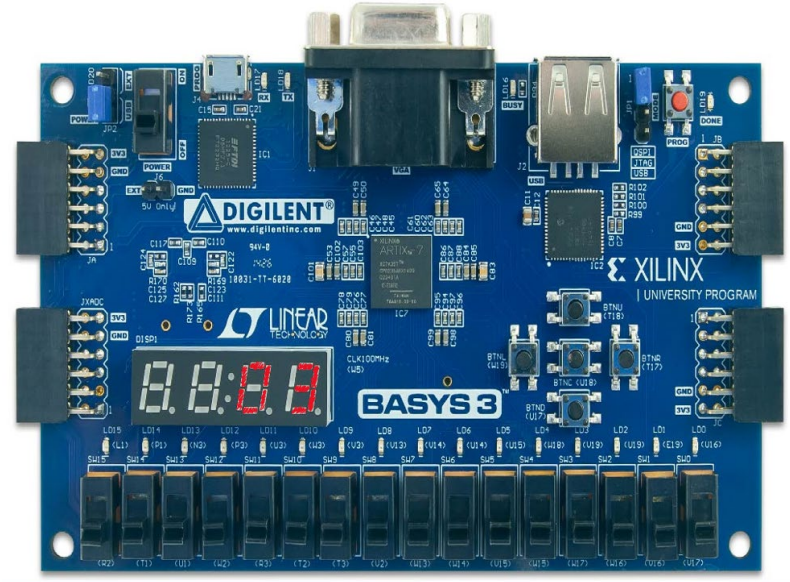


그림2. 사용자가 숫자 3을 입력해 왼쪽 LED에 숫자 3이 업데이트 된 상황

(3) 컴퓨터의 숫자 결정

다음으로는 위의 LED를 통해 컴퓨터의 차례를 표시하고, 컴퓨터가 입력한 값이 왼쪽 LED 화면에 업데이트 되어 누적된 값을 표시한다. 다음은 첫번째 컴퓨터가 숫자 2를 입력한 후, 두번째 컴퓨터가 숫자 1을 입력했을 때의 상황을 보여주고 있다.

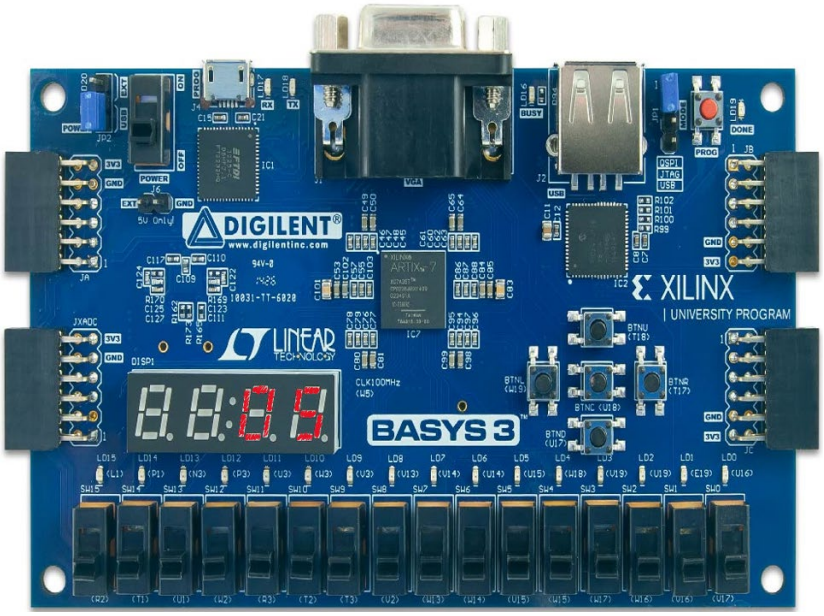


그림3. 컴퓨터가 숫자 2를 눌러, 사용자가 앞서 입력한 3에 2를 더해 현재 5가 누적되었음을 알려줌.

(4) 컴퓨터의 차례 종료 후 다시 사용자의 차례

컴퓨터 플레이어1과 플레이어2의 입력이 끝나고 나면 다시 사용자의 차례가 돌아오고, 기계는 준비 상태에 들어간다. 누적 값이 31이상이 될 때까지 사용자와 컴퓨터의 입력을 차례로 반복한다.

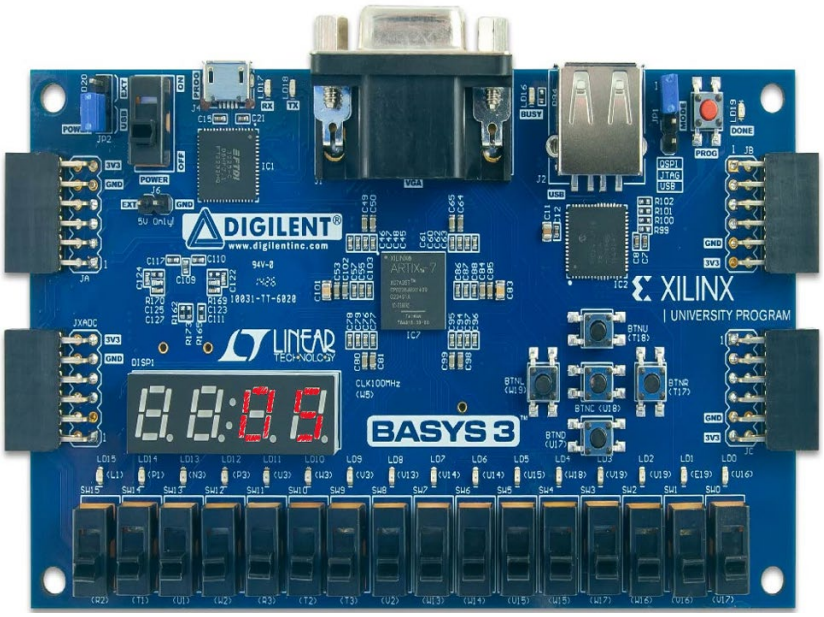


그림4. 컴퓨터의 입력 종료 후 다시 사용자의 입력을 받기 위해 대기 상태에 들어감.

(5) 누적 값이 31 이상이 되었을 때

사용자 혹은 컴퓨터가 입력한 후 누적 값이 31이상이 된다면 게임은 종료되고, 오른쪽 LED에 승패가 표시된다. 다음은 사용자가 이겼을 때와 컴퓨터가 이겼을 때 각각을 나타낸 그림이다.

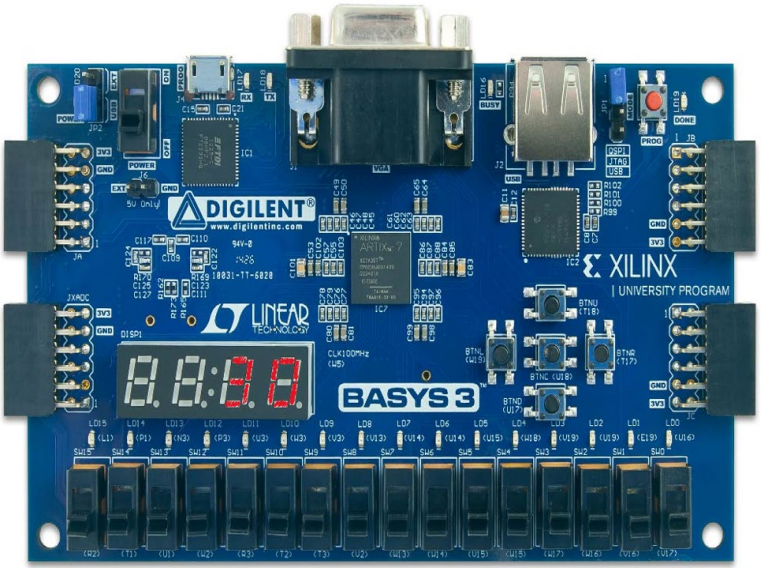


그림5. 누적 값이 30인 상황에서 컴퓨터 1이 1~3 사이의 숫자를 선택해야 하는 상황.

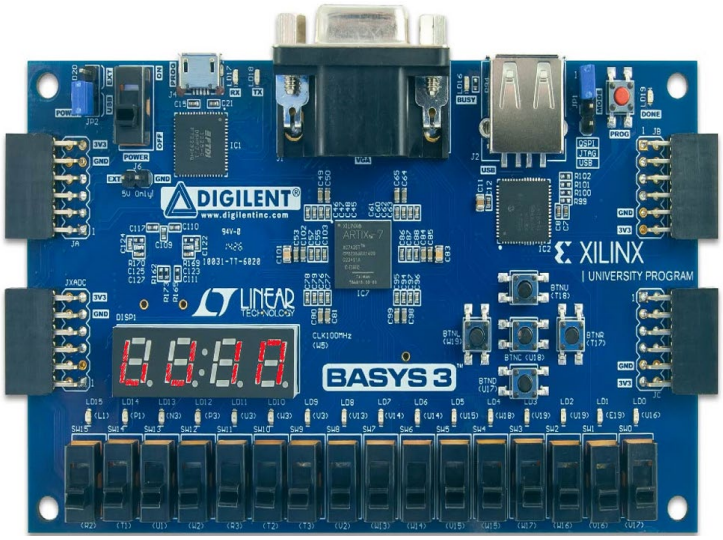


그림6. 컴퓨터 1이 숫자를 선택한 후, 사용자가 이긴 상태를 나타냄.

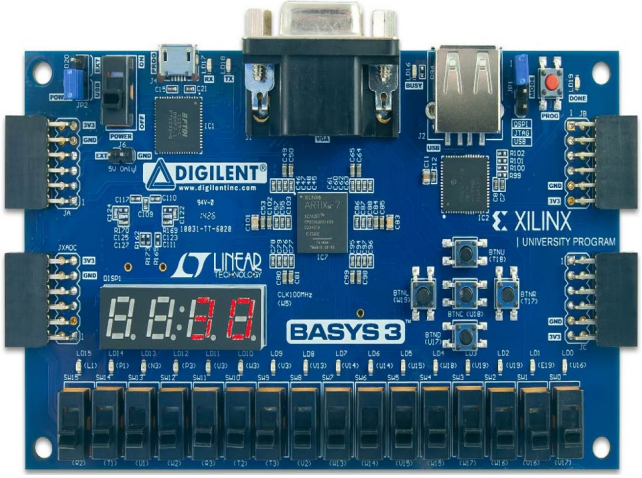


그림7. 누적 값이 30인 상황에서 사용자가 1~3 사이의 숫자를 선택해야 하는 상황

/사용자가 1을 선택했다 가정

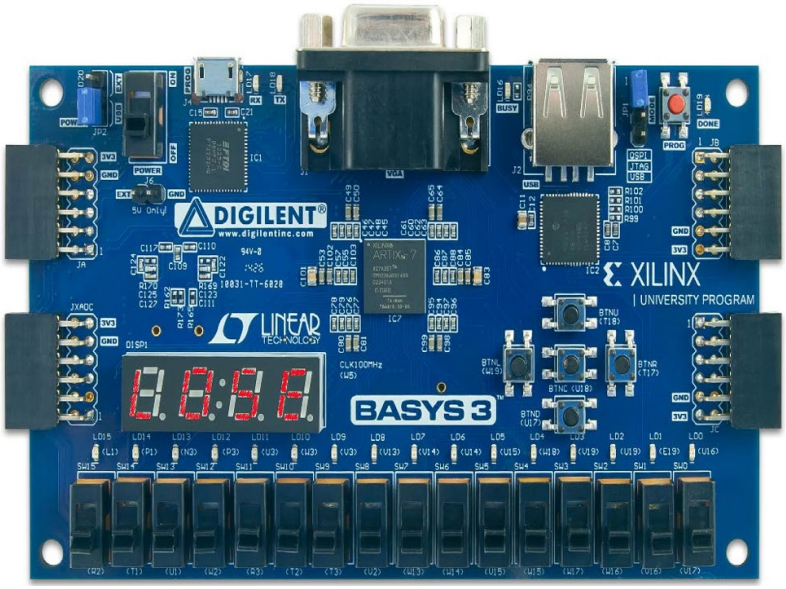


그림8. 사용자가 숫자를 선택한 후, 사용자가 진 상태를 나타냄.

(6) WIN 혹은 LOSE가 출력되어 승패가 결정되고 난 이후
다시 초기 상태 00으로 돌아가 처음부터 다시 게임을 즐길 수 있다.

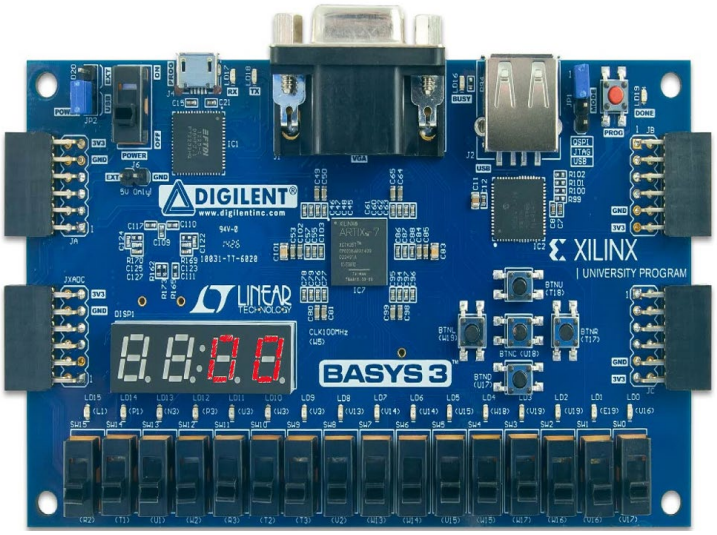
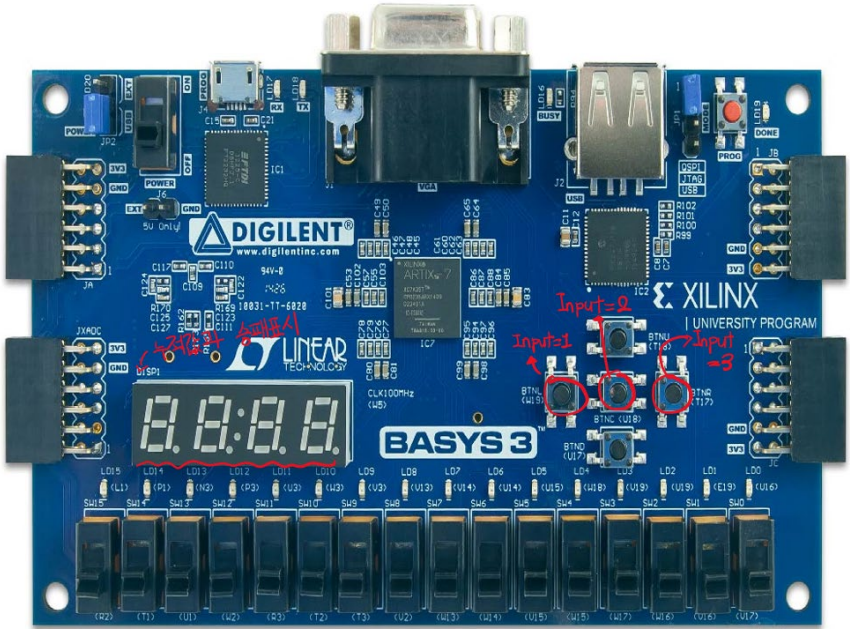


그림9. 다시 초기값 00으로 세팅되어 새롭게 게임을 즐길 수 있음.

ㅂ. 필요 부품 및 사용처

BASYS 3 FPGA Board와 Micro USB cable을 사용하여 본 프로젝트를 구현한다. 컴퓨터의 차례를 나타낼 LED는 FPGA 보드 하단의 슬라이딩 스위치 위 LED를 이용하여 표시한다. 처음에 컴퓨터 플레이어의 수를 입력 받고 사용자가 1개에서 3개까지의 범위 내 외칠 숫자 개수를 입력 받는 것은 FPGA 보드의 슬라이딩 스위치를 이용한다. FPGA 보드에 내장되어 있는 4자리 7-segment LED를 사용하여 현재 누적 값과 최종 승패 결과를 출력한다.

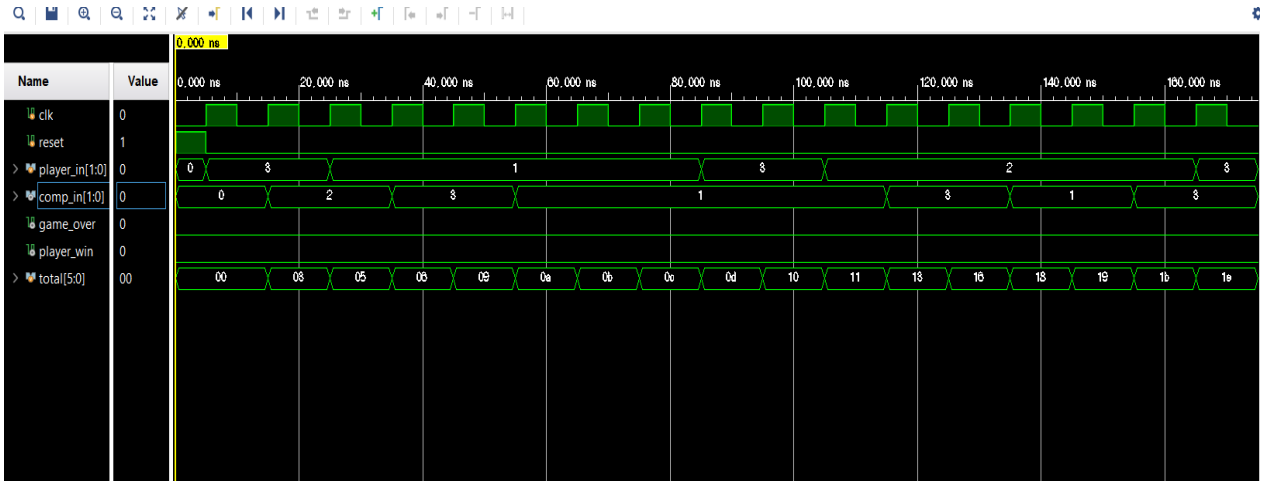


ㅅ. 프로젝트 결과

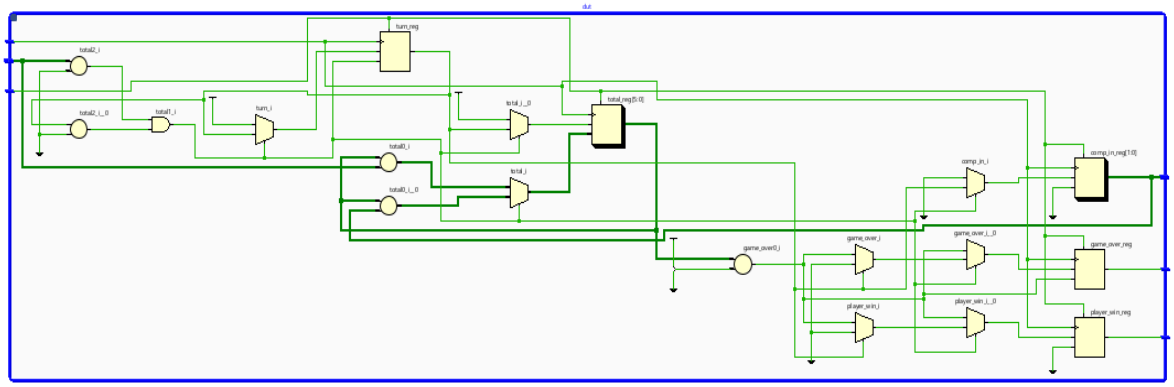
프로젝트 시뮬레이션을 돌린 결과는 아래 콘솔 창을 통해 확인해볼 수 있다. 게임은 사용자가 먼저 숫자를 입력하는 것으로 시작되는데, 본 프로그램에서는 사용자의 숫자를 랜덤으로 생성하는 것으로 사용자가 숫자를 입력하는 것을 대신하였다. 사용자의 입력 후에는 컴퓨터가 숫자를 입력하고 현재까지의 합이 누적되어 콘솔 창에 표시되는 것을 확인할 수 있다. 현재까지의 합이 31 이상이 되면 플레이어와 컴퓨터 중 누가 승리하였는지를 표시하면서 게임은 종료된다. 본 시뮬레이션은 플레이어와 컴퓨터 1:1로 게임을 하는 상황을 보여주고 있다.

```
# run 1000ns
플레이어: 3, 현재까지의 합:  3
컴퓨터: 2, 현재까지의 합:  5
플레이어: 1, 현재까지의 합:  6
컴퓨터: 3, 현재까지의 합:  9
플레이어: 1, 현재까지의 합: 10
컴퓨터: 1, 현재까지의 합: 11
플레이어: 1, 현재까지의 합: 12
컴퓨터: 1, 현재까지의 합: 13
플레이어: 3, 현재까지의 합: 16
컴퓨터: 1, 현재까지의 합: 17
플레이어: 2, 현재까지의 합: 19
컴퓨터: 3, 현재까지의 합: 22
플레이어: 2, 현재까지의 합: 24
컴퓨터: 1, 현재까지의 합: 25
플레이어: 2, 현재까지의 합: 27
컴퓨터: 3, 현재까지의 합: 30
플레이어: 3, 현재까지의 합: 33
게임 종료 - 컴퓨터 승리
```

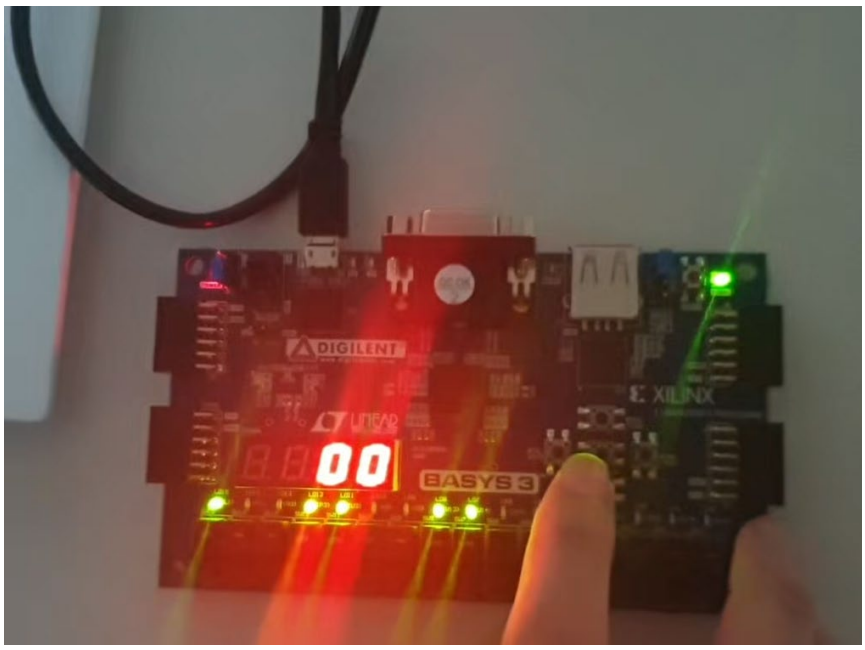
시뮬레이션 결과를 보면 아래의 그림과 같음을 알 수 있다.



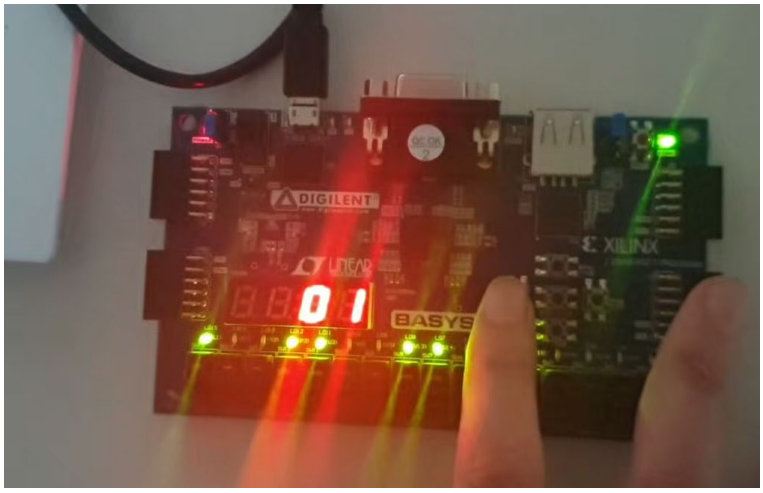
Schematic은 아래의 사진과 같음을 알 수 있다.



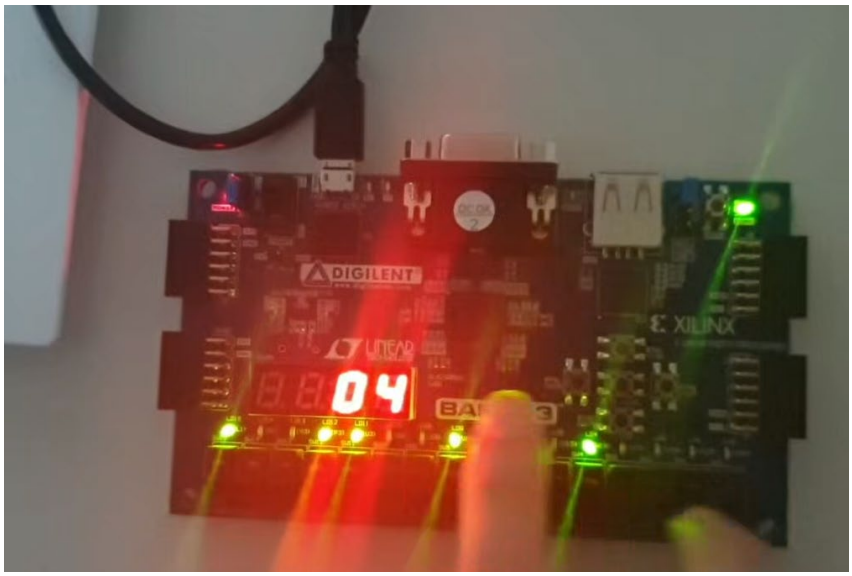
다음은 FPGA를 이용하여 컴퓨터와 플레이어가 베스킨라빈스 게임을 진행하는 사진이다.



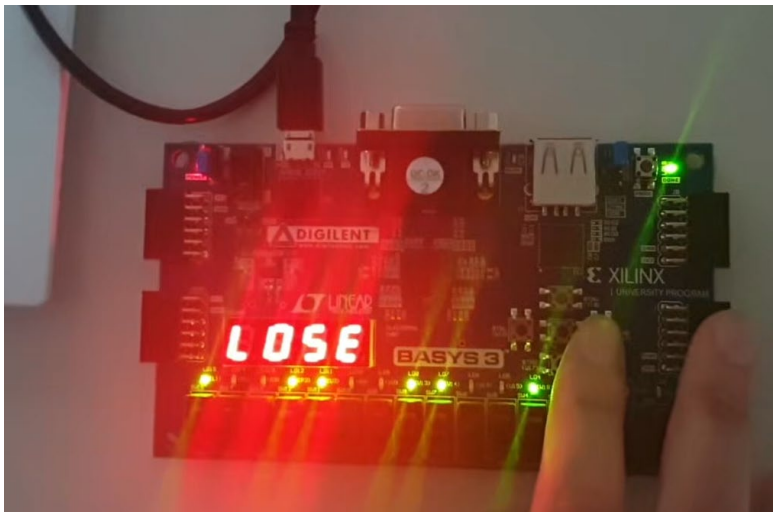
초기 상태에는 00이라는 숫자가 뜨고 사용자는 왼쪽, 가운데, 오른쪽 버튼을 통해 원하는 값을 입력할 수 있다. 이 때 왼쪽버튼부터 1, 2, 3에 해당하는 값을 갖는다.



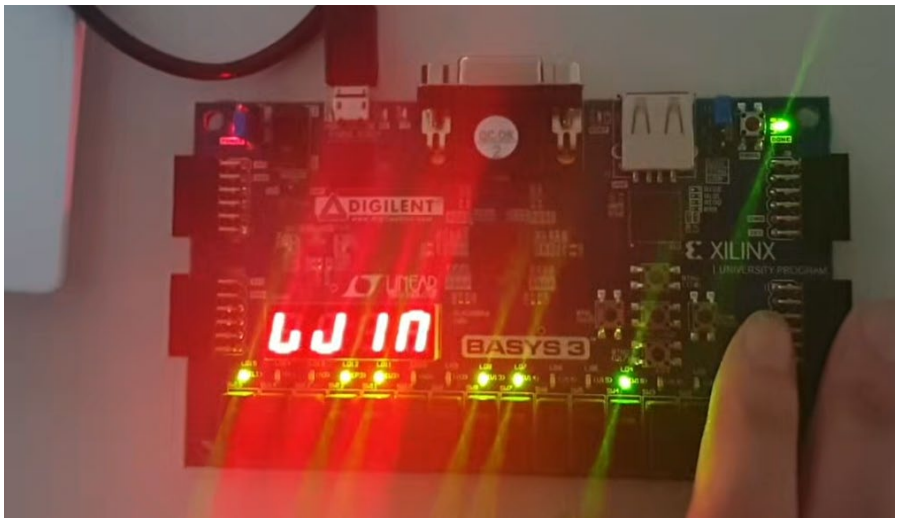
사용자가 왼쪽 버튼을 눌러 1이라는 값이 입력되었고 7-segment display에는 현재 누적 값을 표시하는 곳이므로 01이라는 값이 업데이트 되었다.



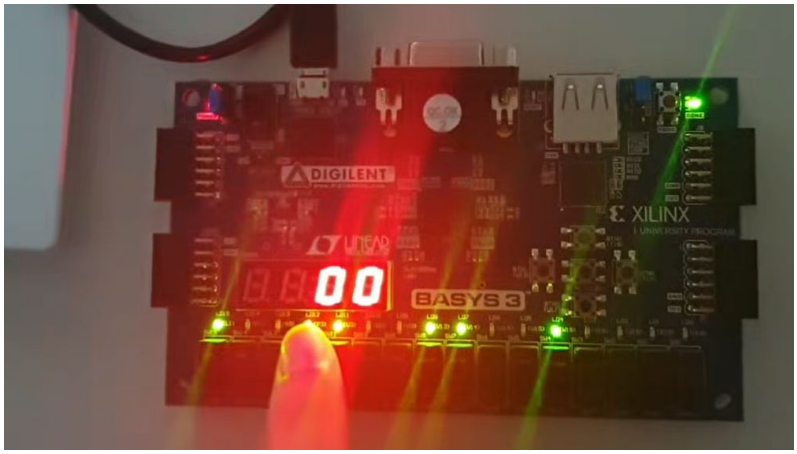
사용자의 입력 후에는 컴퓨터가 랜덤으로 1부터 3 사이의 숫자를 집어넣는다. 위의 사진에서는 사용자가 1을 넣고 컴퓨터가 3이라는 값을 집어넣었으므로 현재 누적값이 04로 업데이트 되어 표시됨을 알 수 있다.



이렇게 컴퓨터와 플레이어가 숫자를 번갈아가며 집어넣는 과정을 반복한다. 만약 사용자의 차례에서 입력을 받은 후 누적 값이 31 이상이 된다면 LOSE라는 단어를 display에 표시하여 사용자의 패배를 알린다.



만약 컴퓨터의 차례에서 컴퓨터가 입력값을 넣은 후 현재 누적 값이 31 이상이 된다면 WIN이라는 단어를 표시하여 사용자의 승리를 알린다.



LOSE 혹은 WIN이 출력된 후에는 다시 초기값 00으로 리셋되어 새롭게 게임을 플레이할 수 있다.

ㅇ. 토론

FPGA 연결을 위해 스스로 코드를 짜보는 과정은 처음이어서 FPGA를 작동시키는데 많은 착오가 있었다. 특히 FPGA에서 초기 상태에 00을 표시하는 것은 가능했지만 버튼을 눌렀을 때 플레이어의 값을 집어넣는 부분에서 어려움이 있었는데, 이는 Non-Blocking assignment 코드를 이용하여 컴퓨터가 입력을 바로바로 처리하지 않고 하나의 sequence가 끝난 후 한꺼번에 assign 과정을 수행하기 때문에 버튼을 누를 때 바로바로 값이 처리되지 않는다는 것을 알게 되었다. 또한 counter를 어떻게 초기화시켜야 값이 올바르게 변화되는지도 알 수 있었고, game_over의 조건문을 통해 WIN과 LOSE가 번갈아가며 출력되는 문제도 해결할 수 있었다.

FPGA 코드를 베릴로그를 이용해 처음 짜보는 과정이 많이 힘들고 어려웠지만 이번 실험

을 통해 베릴로그와 FPGA에 대해 더 잘 이해할 수 있게 되었다. 또한 팀원들과 함께 서로 도우며 베스킨라빈스 31 게임을 완벽하게 구현하였다는 점에서 매우 의미가 있는 실습이었다.