19조 논리회로설계및실험 텀프로젝트 보고서

2024.12.23.

201824607 편경찬, 202155659 한 네이 네인

요 약 텀프로젝트 '이진수 암산 게임'의 설계 과정을 설명하고 외부 및 내부 모듈 설계 원리를 기술한다. 시연을 통해 실험 결과를 보고한다.

• 목차

I 서론

• 제안서 발표 시의 주제 및 구현 계획

1. '2진수 암산 게임'을 주제로 선정
2. Random Number Generator (RNG), 타이머, D2B, 레지스터, 비교기의 모듈 구현
3. 설계한 회로를 FPGA 보드로 구현
4. Keypad를 통한 입력, LED, 7 segment display, Full color LED를 이용해 출력

• 계획된 시나리오

1. 2개의 RNG를 통해 무작위로 생성된 정수를 D2B로 변환
2. 변환된 비트열을 LED 1~4, LED 5~8에 각각 표시
3. LED 표시 직후 타이머가 시작

4. 사용자는 해당 이진수의 10진수 값을 키패드를 통해 연달아 입력
5. 타이머는 사용자가 입력을 완료할 때까지 증가하며 com7, com8에 표시
6. 사용자가 제시간 안에 정답을 입력한 경우
▷ Full Color LED가 초록색으로 변화
7. 제시간에 입력하지 못하거나 틀렸을 경우
▷ Full Color LED가 빨간색 유지

• 계획에서 변경된 부분

1. 생성된 난수에 대해 D2B로 미변환
RNG 내부에서 이미 정수가 아닌 비트열을 출력하도록 설계하였기에 D2B로 재변환할 필요가 없어 제거했다.

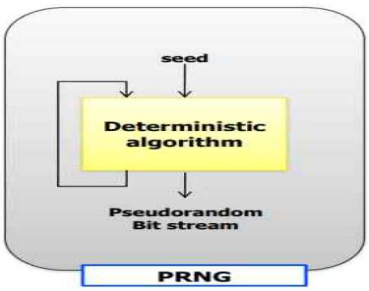
2. 타이머의 카운팅 방식 및 출력 위치 조정
기존 계획에서는 타이머가 00에서 99로 counting up 되기로 계획했으나, 해당 방식보다는 9에서 0으로 counting down 하는 방식이 게임을 진행할 때 더 긴장감을 줄 수 있다고 판단하여 수정했다. 또한 타이머는 com7과 com8에 표시할 계획이었으나 해당 위치는 키패드를 입력하는 부분과 이어져 있기에 사용자가 키패드 입력 시 혼선을 받을 수 있다고 판단하여 single 7 segment에 별도로 표시하기로 했다.

3. RNG의 구체화
PRNG 방식 중 하나인 Linear Congruential Generator (LCG)와 TRNG 요소를 결합하여 난수를 생성하는 방법을 구체화 시켰다.

II 기능 설명

• RNG의 난수 생성 원리

Fig. 1 - PRNG 원리



PRNG의 원리는 고정값 seed로부터 결정적 알고리즘을 사용하여 랜덤 비트열을 출력하는 것이다. 랜덤한 값을 생성하기 위해 PRNG의 요구사항인 비예측성, 임의성을 만족시키고 Seed

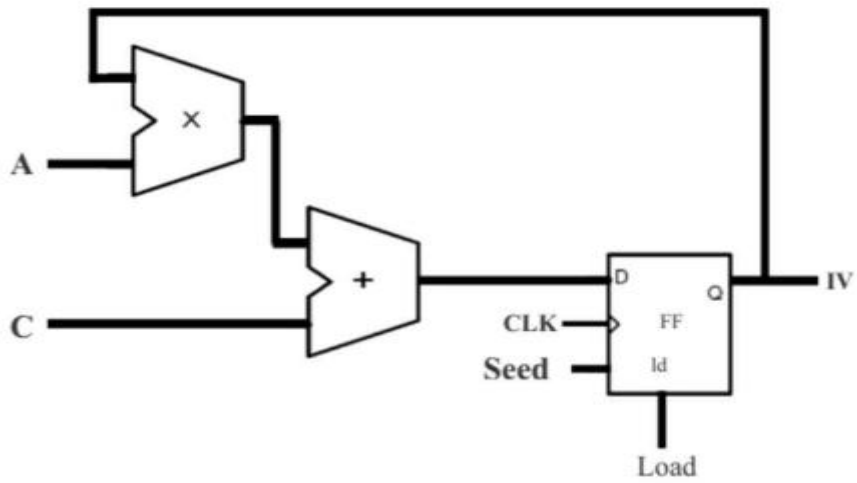
값과 인자들을 예측할 수 없게 설계했다.

Fig. 2 - LCG Equation

$$Xn+1 = (aXn + c) \bmod m$$

PRNG의 종류 중 하나인 LCG는 해당 수식을 기반으로 동작한다. RNG는 4비트를 사용하기에 모듈러는 16(2^4)으로 설정한다. LCG가 가능한 모든 값을 한 번씩 생성하는 최대 주기를 가지도록 a에 대해 a-1이 m의 모든 소인수로 나누어지도록 설정하고 c와 m은 서로소로 설정한다.

Fig. 3 - LCG Circuit



LCG는 최초의 시드 값을 기준으로 Fig. 2 식에 의해서 새로운 난수를 만들어낸다. 선형 합동 생성기는 Linear 하기에 인자들과 마지막으로 생성된 난수를 알면 그 뒤에 만들어질 모든 난수를 예측할 수 있어서 암호학적으로 안전한 난수 생성기는 아니다. 또 Pseudo이기 때문에 프로그램을 실행하면 우리가 최초에 설정한 seed 값에 의해 매번 동일한 숫자를 생성해낼 것이다. 즉 시드가 6, a가 5, c가 1일 경우, *'6, 15, 12, 13, 2, 11, 8, 9, 14, 7, 4, 5, 10, 3, 0, 1, 6 ...'*의 순서로 퀴즈를 제시할 것이다.
우리 조는 이런 패턴화되는 문제를 막기 위해 일정 시간 동안 매우 빠른 클락을 발생시키고 그 클락이 발생한 횟수만큼 Fig 3. 회로에 연산을 진행해 출력된 값을 난수로 추출하기로 했다. 사용자에게 항상 start 버튼과 stop 버튼을 눌러서 새로운 게임을 진행할 수 있도록 유도하고 해당 딜레이 시간 동안 발생하는 클락만큼 LCG를 여러 번 연산시켰다. 즉 Pseudo와 linear 하다는 단점을 실제 랜덤한 소스인 TRNG적인 요소를 이용해 패턴화되는 랜덤 수열 중에서 어떤 부분을 시작하는지 모르게 만들었기 때문에 사용자는 물론이고 시드값을 설정하는 원작자도 예상할 수 없는 완전한 랜덤을 구현했다.

• 타이머의 동작 방식

Fig. 4 - Timer & Clock

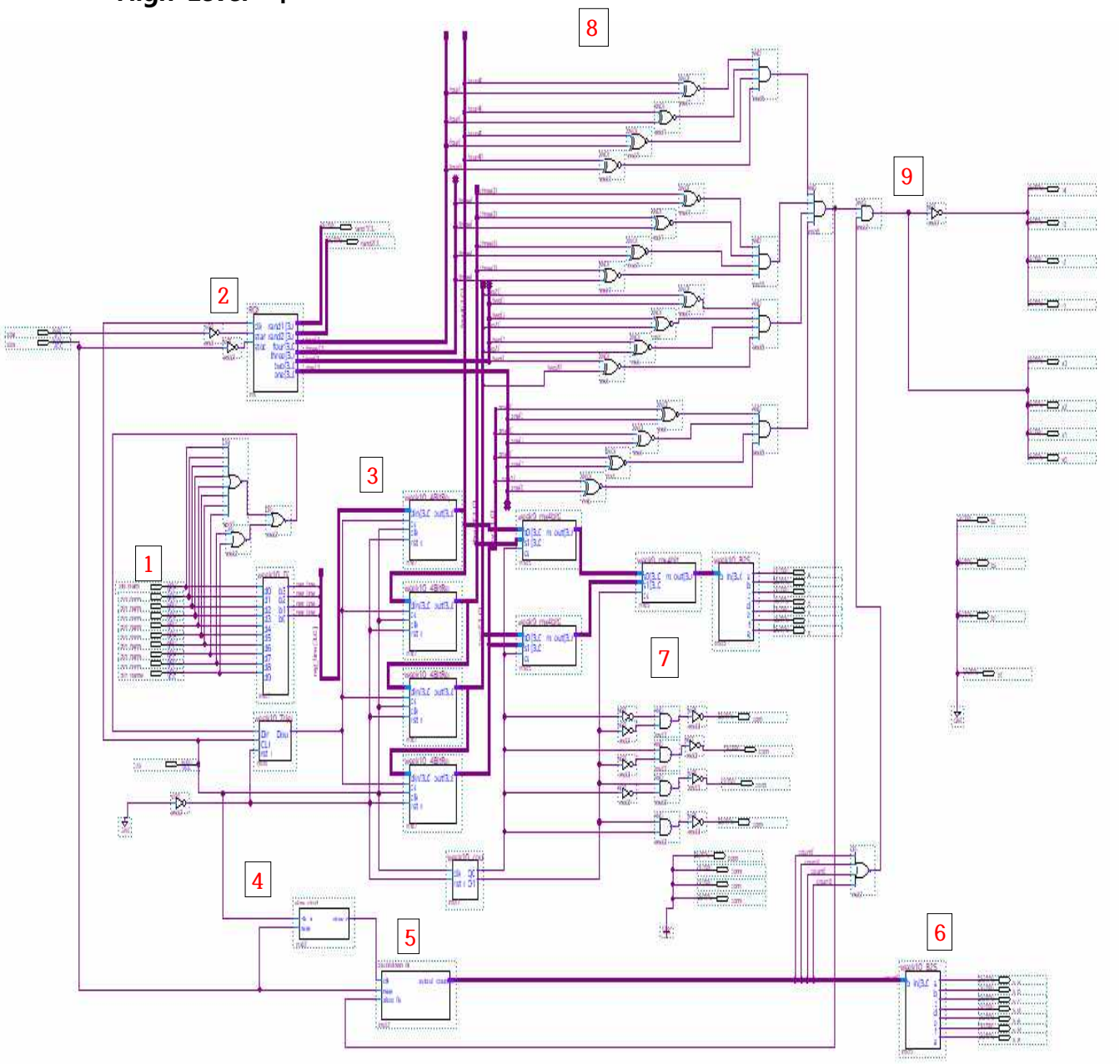
타이머는 클락 500Hz에 맞춰 9에서 시작해 0으로 카운트 다운되도록 설계했다. Clock 분주를 이용해 실제 초와 동기화되도록 설정했다. LED의 이진수가 표시된 후부터 키패드로 정답을 입력 완료할 때까지 타이머가 동작하도록 구현했다. Fig. 4는 좌측의 싱글 7 segment에 '7초'가 디스플레이 된 모습을 볼 수 있다.

• 기타 입출력 기능

3x4 키패드를 사용하여 표시된 이진수에 해당하는 십진수를 사용자로부터 입력받는다. 입력된 십진수는 7 segment array의 com1 ~ com4에 표시되도록 구현했다. 사용자의 입력값을 생성된 난수와 비교하여 입력이 정확하면 Full Color LED가 녹색, 입력이 잘못되었거나 타이머가 0이 되어 제한 시간 안에 정답을 입력하지 못하면 빨간색을 표시한다.

Ⅲ 회로도 및 모듈 기능 설명

- **High Level 회로도**



1번 섹션에서 키패드 입력을 감지하면 D2B로 변환시켜 3번 섹션의 레지스터에 차례로 저장한 뒤 7번 섹션에서 결정된 com 자리에 의해 8 array segment에 표시된다. 2번 섹션은 RNG로써 난수를 생성하며 LED에 표시한다. 4번 섹션은 클락을 분주시켜 5번 섹션의 카운트 다운 타이머가 6번 섹션의 Single 7 segment에 표시되도록 한다. 8번 섹션은 난수 비트와 키패드 입력 비트를 비교해 정답으로 인정될 시 5번 섹션의 타이머의 입력 비트에 1을 주고 9번 섹션에서 Full Color LED로 빨간색과 녹색 중 하나를 결정하도록 만들었다.

• Module Level 회로도 & 모듈 기능

Fig. 5 - RNG

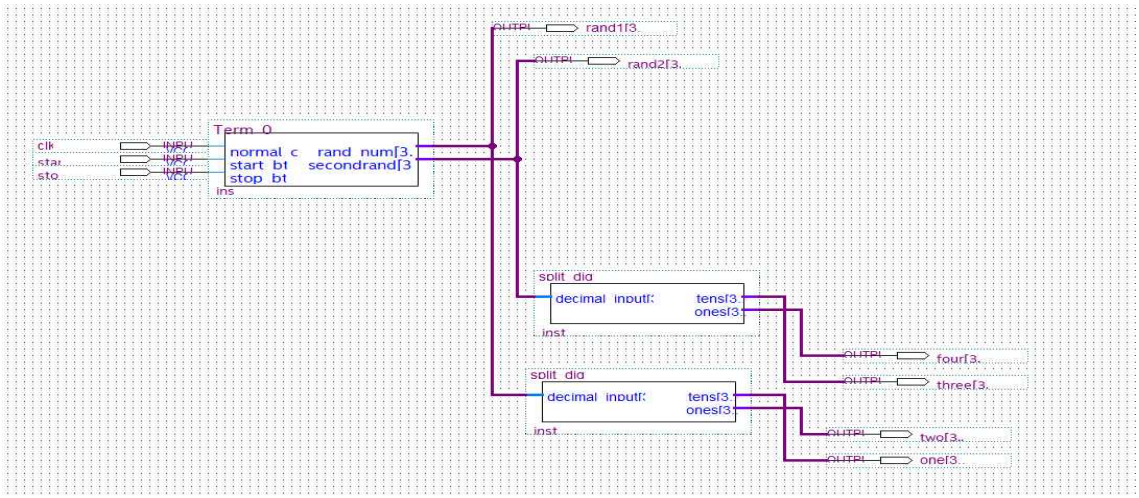


Fig. 6 - RNG > Term_05

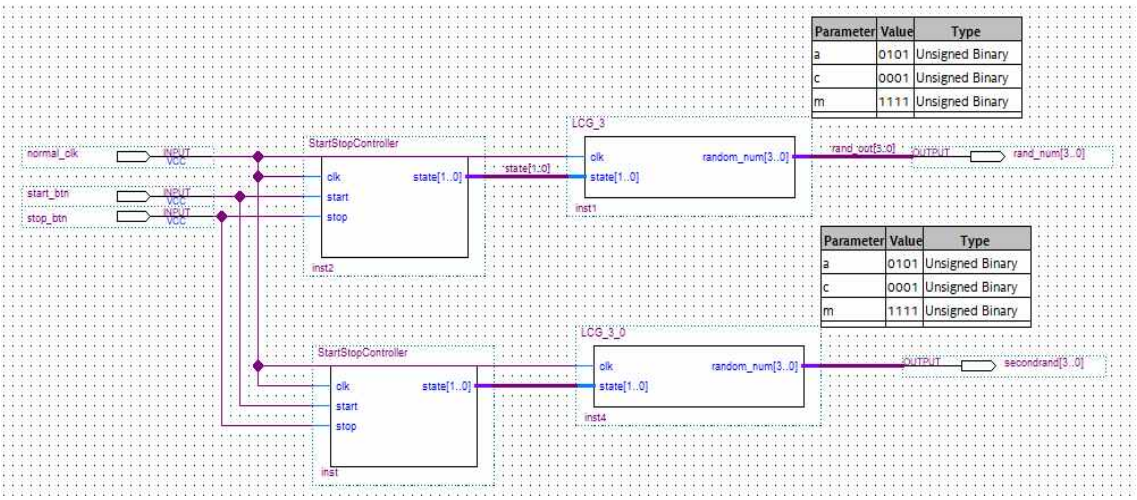


Fig. 7 - RNG > Term_05 > StartStopController

```
module StartStopController (
    input clk,
    input start,
    input stop,
    output reg [1:0] state
);

    initial begin
        state <= 1;
    end

    always @(posedge clk) begin
        if(!start)
            state <= 1;
        else if (!stop)
            state <= 0;
    end

endmodule
```

Fig. 8 - RNG > Term_05 > LCG3 / LCG3_0

```
module LCG_3 (
    input clk,
    input [1:0] state,
    //input [3:0] seed,
    output reg [3:0] random_num
);

parameter a = 4'b0101;
parameter c = 4'b0001;
parameter m = 4'b1111;
reg [3:0] current_num;

initial
   current_num <= 4'b0110;

always @(posedge clk) begin
    if (state==1) // start_btn pressed
       current_num <= (a * current_num + c) & m;
    else if (state==0) // stop_btn pressed
       random_num <= current_num;
end

endmodule
```

Fig. 9 - RNG > Term_05 > split_digits

```
module split_digits (
    input [3:0] decimal_input,  //
    output reg [3:0] tens,      //
    output reg [3:0] ones       //
);

    always @(*) begin
        // Split the input into te
        tens = decimal_input / 10;
        ones = decimal_input % 10;
    end

endmodule
```

두 개의 독립적인 LCG 인스턴스 중 하나인 Fig. 8의 LCG3는 '0110'인 6, LCG3_0은 '0100'인 4를 시드값으로 갖고 Fig. 2의 원리에 따라 a는 5, c는 1를 갖는다. state가 1인 상태는 사용자가 start 버튼을 누른 상황이며 stop 버튼을 눌러 state가 0이 될 때까지 'a * current_num + c'의 모듈러 값을 'current_num'에 저장한다. stop 버튼을 누르면 딜레이 시간 동안 연산이 반복된 'current_num' 값이 출력 레지스터인 random_num 변수에 고정적으로 저장되며 출력한다. start 버튼과 stop 버튼은 Fig. 7의 'StartStopController'에 의해 제어된다. 최종 난수 비트열은 Fig. 5와 같이 LED1~4 및 5~8과 'split_digits'의 입력으로 들어간다. Fig. 8의 'split_digits'은 4비트 이진수를 십의 자리와 일의 자리로 구분해 출력한다.

Fig. 10 - slow_clock

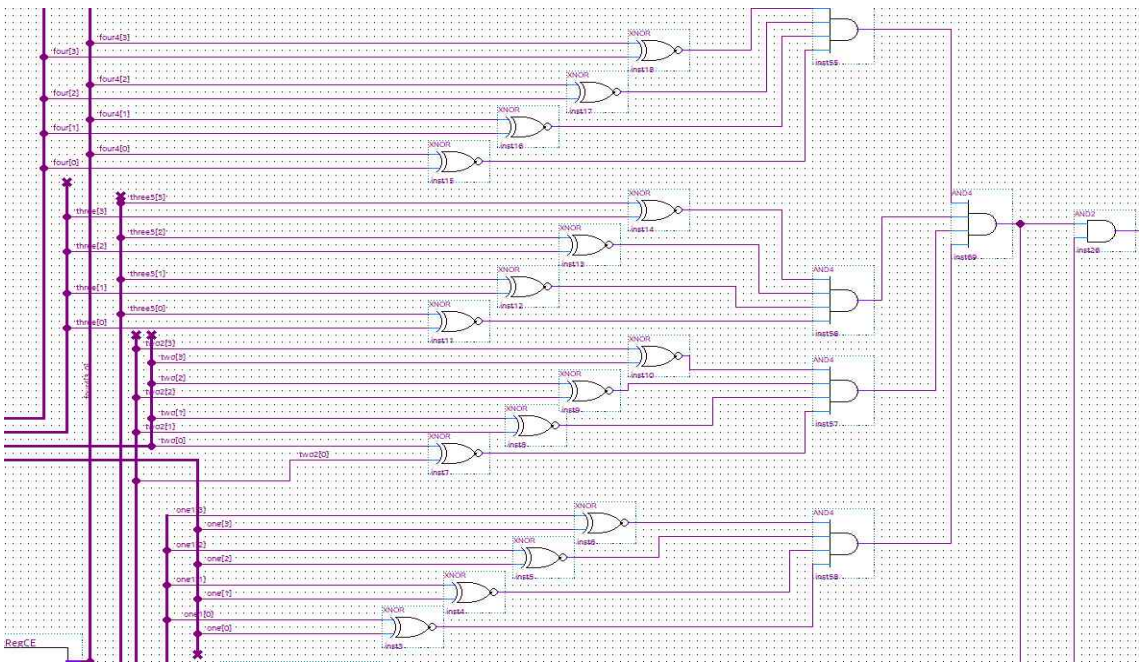
```
module slow_clock9 (
    input wire clk_in,
    input wire reset,
    output reg slow_clk
);

    reg [8:0] counter;

    always @(posedge clk_in or posedge reset) begin
        if (reset) begin
            counter <= 9'b0;
            slow_clk <= 1'b0;
        end else if (counter == 9'd499) begin  // Ac
            counter <= 9'b0;
            slow_clk <= ~slow_clk;
        end else begin
            counter <= counter + 1;
        end
    end

endmodule
```

Fig. 11 - countdown_timer

```
module countdown_timer (
    input clk,
    input reset,
    input stop_flag,
    output reg [3:0] output_count
);

    reg [6:0] count = 7'b0001001;

    always @(posedge clk or posedge reset) begin
        if (reset) begin
            count <= 7'b0001001; //0
        end else if (!stop_flag) begin
            if (count > 0) begin
                count <= count - 1;
            end else begin
                count <= 7'b0000000;
            end
        end
    end

    always @(*) begin
        output_count = count;
    end

endmodule

```

Fig. 10의 slow_clock은 입력된 시스템 클럭을 500배 느리게 만들어 출력 클럭을 생성하는 모듈이다. counter 값이 499에 도달하면 counter를 0으로 리셋하고 slow_clk를 토글한다. 만약 reset 신호가 활성화되면 counter와 slow_clk이 초기화 된다. 시스템 클럭이 500Hz 이기에 1/500s = 2ms 이며 1초 간격은 1/2ms이기에 500으로 셋팅해야 카운트 다운이 초 단위로 동기화 된다. Fig. 10의 countdown_timer는 9에서 시작해 stop_flag가 true가 되기 전 까지 0으로 카운트 다운한다. 향후 stop_flag는 비교기에서 정답 여부 비트를 받아와 타이머를 멈출지 결정한다. reset이 활성화되면 타이머는 다시 9초로 초기화된다.

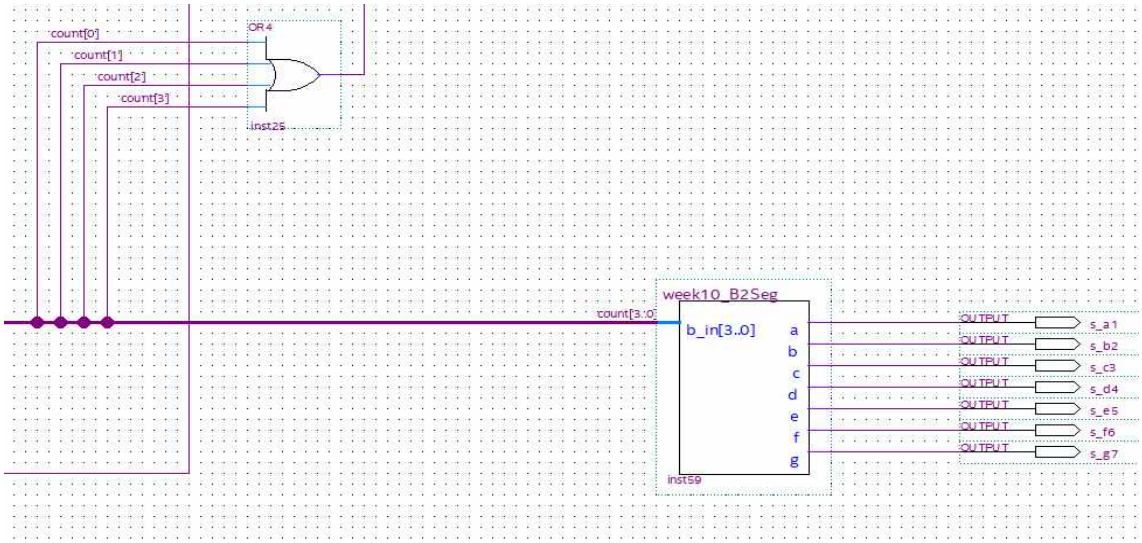
Fig. 12 - Comparator



비교기는 4비트 난수 두 개와 키패드로 입력한 4자리 값의 비트를 각각 XNOR로 비교해 모두

일치할 때 AND에 의해 비트 1을 출력한다. 이후 해당 비트를 반전시킨 비트를 빨간색 Full Color LED에 연결해 기본적으로 빨간불이 들어오도록 설정하고, 정답이 입력되면 녹색불이 켜지도록 설정했다.

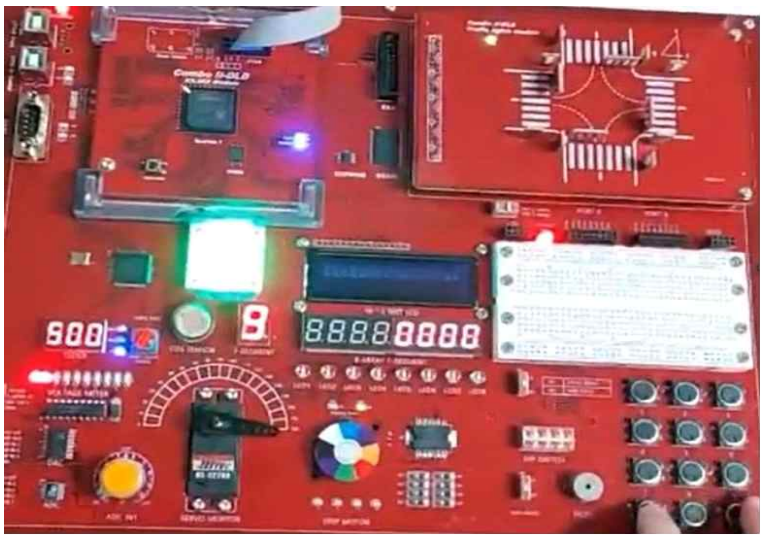
Fig. 13 - Single 7 Segment display



상단의 OR4는 count[0..3] 값을 OR 연산 후 Fig. 12의 비교기 AND2의 입력 비트 중 하나로 전달한다. 즉 count의 모든 비트가 0이면 AND는 0을 출력하도록 해 제한 시간 안에 입력을 못하면 정답 여부와 상관없이 빨간불이 들어오도록 설정했다. 우측의 B2Seg 모듈은 4비트 입력값을 single 7 segment의 각 세그먼트(a~g)에 매핑하여 숫자를 표시하는 디코더이다.

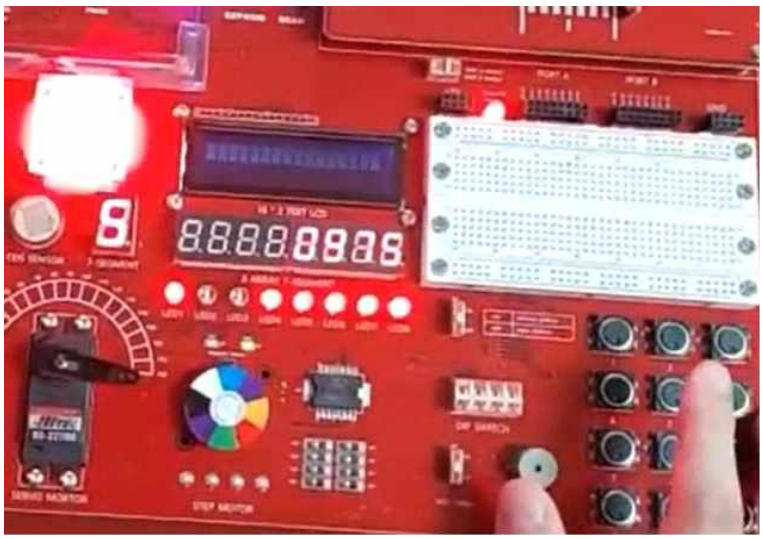
IV 동작 결과

Fig. 14 - 프로그램 실행 후 대기모드



프로그램을 실행하면 대기모드에서 사용자가 start 버튼과 stop 버튼을 누를 때까지 기다리게 된다.

Fig. 15 - 오답 입력



stop 버튼을 누르면 LED1~4와 LED5~8에 두 개의 난수가 퀴즈로 제시되며 Full Color LED는 빨간색을 유지한 채 카운트 다운을 시작한다. Fig. 15의 상황에서 입력으로 '0915'를 입력해야 하지만 오답으로 '0975'를 입력하면 카운트 다운은 계속 진행되며 정답으로 인식되지 않는다.

Fig. 16 - 정답 입력

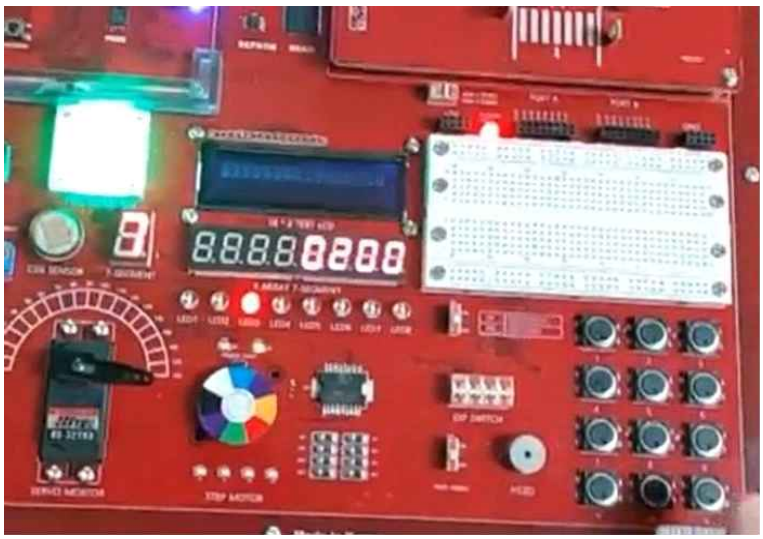


Fig. 16에서 주어진 LED 퀴즈에 따라 정답인 '0200'을 제한 시간 안에 입력 완료하면 Full Color LED는 초록색으로 변화하면 타이머는 멈추게 된다. 새로운 난수를 받아 게임을 다시 실행하기 위해서는 최초의 동작과 동일하게 start 버튼을 누른 후 stop 버튼을 누르면 된다.

V 프로젝트의 추가적인 아이디어

이번 프로젝트에서는 계획서에 제시된 모든 모듈과 기능을 완벽히 구현하였지만, 다음과 같은 아이디어를 추가할 수 있다.

• 난수의 난이도 조절

4비트 + 4비트 조합 대신 5비트 + 3비트 조합을 사용하거나, 하나의 8비트를 출력하도록 설정할 수 있다. 또한, 다중 라운드 형식을 도입해 누적 점수 체계를 추가하는 방식으로 확장할 수도 있다.

• GUI 도입

사용자와의 상호작용 및 모니터링을 위해 LCD를 이용한 그래픽 사용자 인터페이스를 도입할 수 있다.

VI 역할 분배

조원 / 역할	편경찬	한 네이 네인
문서화 및 발표	- 최종 보고서 작성 - 제안서 작성 - 제안서 PPT 제작 - 최종 발표 - 제안서 발표	- 최종 보고서 작성 - 제안서 작성
주요 구현 모듈	- LCG - StartStopController - countdown_timer - slow_clock - B2Seg	- split_digits - comparator - 모듈 연결 및 보드 동작 확인

Ⅶ 결론

이번 텀프로젝트인 '이진수 암산 게임'은 FPGA를 이용한 하드웨어 설계의 다양한 요소를 통합적으로 활용한 프로젝트로, 설계부터 구현, 시연까지 전 과정을 통해 디지털 논리 설계 및 시스템 통합 능력을 향상시킬 수 있었다. 실습에서 사용한 모듈 외에 다른 모듈을 활용해보며 회로 설계뿐만 아니라 수식이 원활히 동작하도록 직접 Verilog 코드를 작성했다는 점에서 의미가 크다. RNG, 타이머, 비교기, 레지스터 등의 모듈 설계와 이를 FPGA에서 구현함으로써 하드웨어 설계의 기본 개념을 심화할 수 있었다. 특히, PRNG 방식인 LCG와 TRNG의 결합은 실제 응용에서 패턴 예측 가능성을 줄이는 창의적인 접근으로, 하드웨어 난수 생성의 실질적인 설계 원리를 경험하는 계기가 되었다. 모듈 간 신호 연결, 클럭 분주를 통한 타이밍 제어, 입력값 비교 및 출력 처리 등의 과정을 거치며 디지털 회로 설계에서 모듈화와 통합 설계의 중요성을 체감할 수 있었다.
게임의 긴장감을 높이기 위해 카운트다운 타이머를 도입하거나, 사용자 입력 혼선을 방지하기 위해 출력 디스플레이 위치를 변경하는 등, UI/UX를 고려한 설계 변경을 통해 단순한 논리 회로 설계에서 나아가 실질적으로 사용 가능한 시스템을 설계하는 과정을 배웠다.

Ⅷ 참고문헌

[1] Majzoobi, Mehrdad, Farinaz Koushanfar, and Srinivas Devadas. "FPGA-based true random number generation using circuit metastability with adaptive feedback control." Cryptographic Hardware and Embedded Systems-CHES 2011: 13th International Workshop, Nara, Japan, September 28-October 1, 2011. Proceedings 13. Springer Berlin Heidelberg, 2011.