결 과 보 고 서

과제명: 디지털 IC 로 구성한 기억력 테스터기

팀 명: Antidementia

경북대학교 IDEC 2022 창의 회로설계 챌린지

I. 과제 개요

최근 기회가 되어 노인복지관에 봉사활동을 나가게 됐는데, 거기서 각종 치매 예방 프로그램에 많은 분들이 지원하는 것을 보았다. 그리고 방문하시는 어르신들과 대화를 나누었는데, 자신이 죽는 것보다 두렵고 더 걱정 되는것은 치매에 걸리는 것이라고 하신 말씀들이 인상 깊어 치매 문제에 관심을 가지게 되었다. 대한민국은 현재 고령화 사회로 점점 더 진입해가고 있다. 고령자들의 증가에 따라서 대두되고 있는 문제가 바로 '치매'이다. 보건복지부의 자료에 따르면 2021 년 65 세 이상 노인 인구는 총 857 만 명이고, 이중에서 치매 환자 수는 88 만 명에 이를 것이라고 한다. 이런 식으로 계속 증가를 하다 보면 2030 년에는 치매 환자 수가 100 만 명이 되는 시대가 올 것이다.

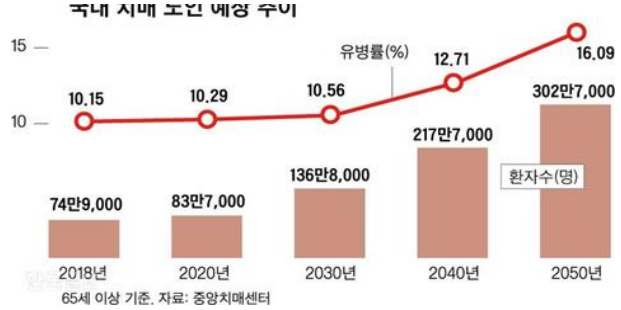


그림 1

모든 병이 그렇듯, 치매 또한 초기에 증상을 파악하고 적절한 치료를 받는다면, 치매 발병 시기를 늦추거나 안 오게 할 수가 있다. 치료 중에는 약물 치료와 비약물치료가 있는데, 비약물치료에서 주로 하는 것이 '기억력 테스트'이다. 기억력 테스트는 치매의 단계별 진료를 받을 수 있고, 중증 상태로 진행되는 걸 억제하거나 조기에 발견할 수 있는 중요한 검사로 알려져 있다. 이러한 기억력 테스트기를 논리 회로를 이용하여 만들어 보고자 한다.

II. 설계 목표

우리의 목표는 회로 소자만으로 기억력 테스트기를 구현하고, 그 과정에서 마주하게 되는 문제들 통해 창의적이고 혁신적인 문제해결 능력을 배양하는 것이다. 본 작품은 Start 버튼을 누르면 문자 ㄱ,ㄴ,ㄷ, ㅁ 랜덤한 순서대로 문자 표시 LED(Light Emitting Diode)에 출력되고 사용자가 LED 로 출력 된 문자들을 기억한 후 제한 시간 내에 알맞은 문자를 순서에 따라 버튼을 누르면 초록색 LED 를 통해 사용자에게 정답 여부를 알려주도록 설계 되었다. 아래 그림 3 에 나온 FLow chart 로 요약된다.

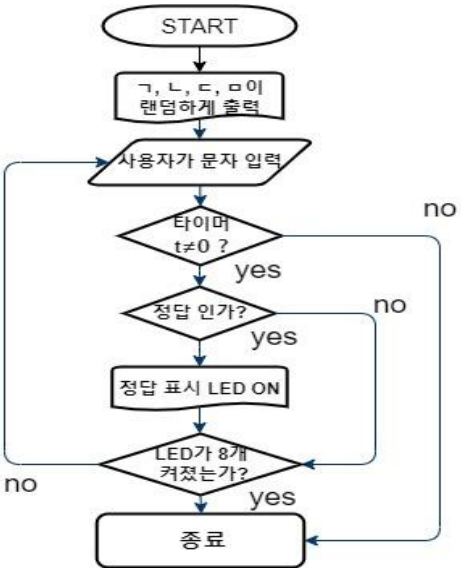


그림 2

III. 설계 및 시뮬레이션

해당 장치는 크게 **입력 모듈, 제어처리 모듈, 출력 모듈**로 구분된다.

1. 입력 모듈

입력 모듈은 그림 3 와 같이 구성되며, 각각의 기호 ㄱ, ㄴ, ㄷ, ㅁ에 해당되는 택트 스위치와 스위치들의 눌림을 감지하는 4bit OR, 레지스터 초기화를 위한 Reset 스위치, 테스트를 시작하는 Start 스위치으로 구성된다. 4x2 Encoder 를 이용하여 ㄱ,ㄴ,ㄷ, ㅁ에 해당되는 버튼들을 각각 11, 00, 10, 01 로 대응되도록 하였다. start 스위치를 DFlipFlop clk 에 연결하고 입력 D 에 1 을 입력해 start 신호가 한번 들어오면 reset 을 누르기 전까지 1 을 출력한다.

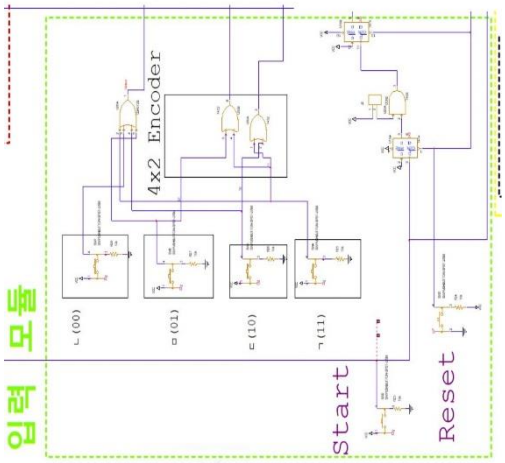


그림 3

2. 제어처리 모듈

제어처리 모듈은 사용자가 입력한 순서가 맞는지 틀렸는지 판별하고 알려주는 Enable 회로, 동작 시간을 제한하는 타이머 회로, CLK 회로 등으로 구성되며 전체 회로 입력, 출력 모듈을 연결하고 제어처리 하는 부분이다.

A. Enable 회로

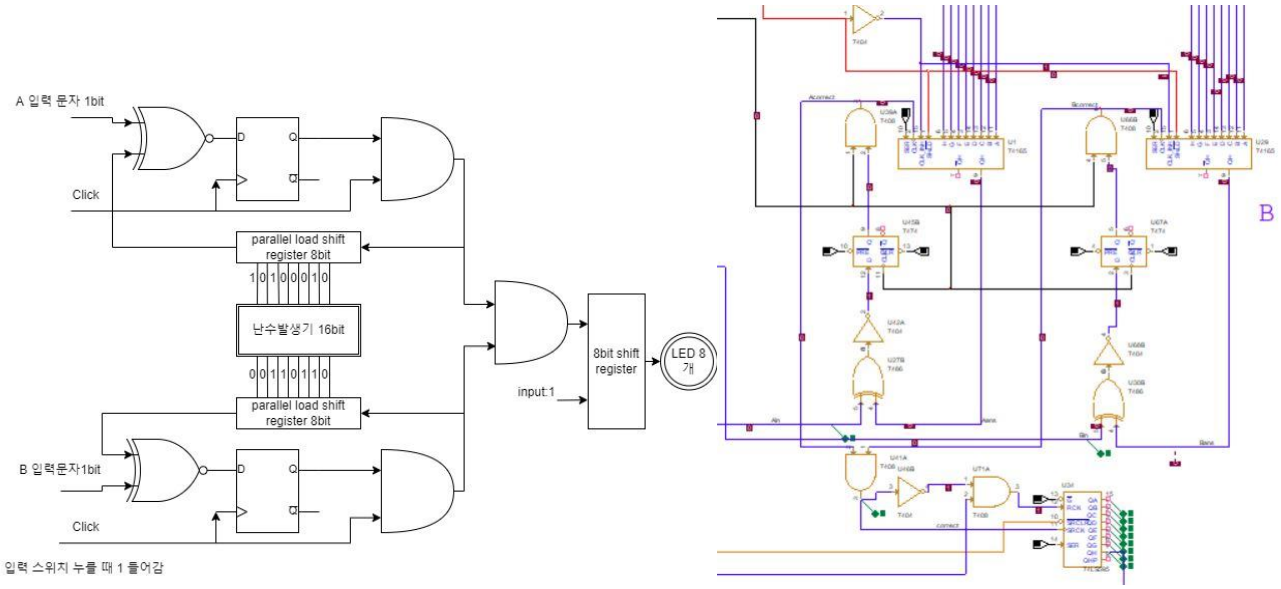
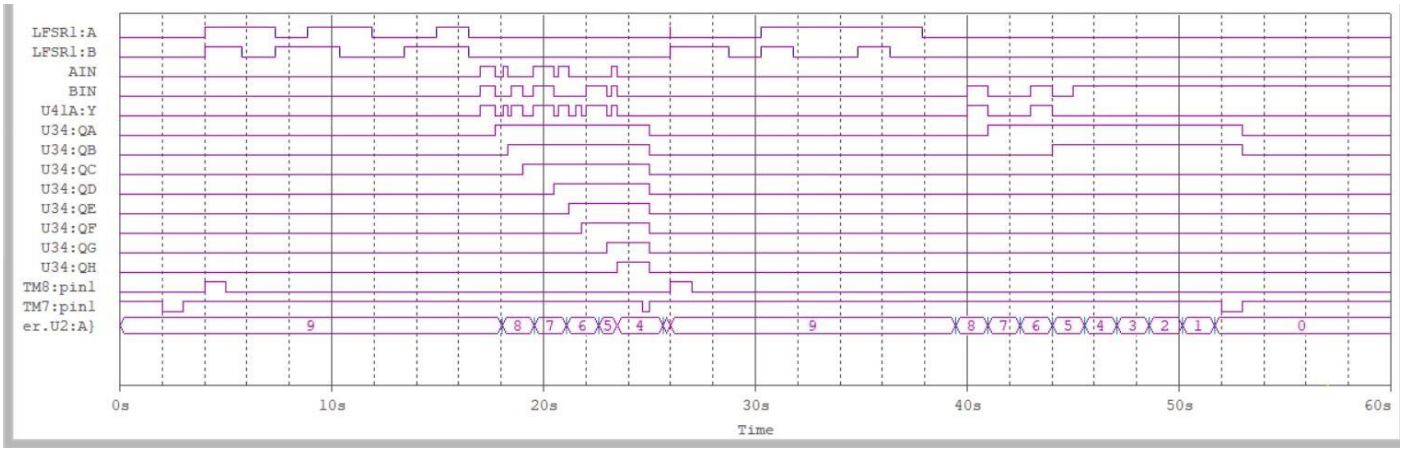


그림 4,5

Start 신호가 입력되면서 랜덤 문자 시퀀스를 8 Bit parallel load shift register 에 저장하고 문자 점등이 끝나면 shift mode 로 변경한다. 사용자가 답을 입력하면 저장된 문자 시퀀스와 XNOR 논리로 일치여부를 판단하여 정답일 경우 1 을 DFF 에 저장하도록 한다. DFF 출력은 serial input 이 1 인 8 bit shift register clk 으로 입력되어 정답을 맞출 때마다 LED 를 하나씩 켜도록 하고, parallel load shift register clk 으로 입력되도록 하여 다음 순서 정답을 비교하도록 한다.



시뮬레이션 1

Enable 회로의 시뮬레이션 1 을 확인한 결과 정답 8 개를 모두 맞춘 경우인 20s~30s 에서는 사용자가 정답(Ain, Bin)을 입력할 때마다 정답 LED U34:QA~U34:QH 가 순차적으로 켜져 시간 제한 타이머(U2:A)가 4 초일 때 8 개 모두 켜지는 것을 확인할 수 있으며, 제한 시간 동안 정답을 두번만 맞춘 경우인 40s~50s 에서는 U34:QB 까지만 켜지는 것을 확인하였다. 정답 LED 들은 reset 신호(TM7: pin1)이 입력되면 모두 꺼지도록 하였다.

B. CLK 회로

CLK 회로는 난수 발생기를 제어하기 위한 0.6kHz 고주파 Clock 신호와 정답 문자 점등 회로, 타이머를 제어하기 위한 0.6Hz 저주파 Clock 신호를 만드는 2 개의 발진회로로 구성된다. 우리는 555ic 를 사용하여 CLK 회로를 구현하였으며, 세 가지 동작 영역 중 끊임없이 두 상태를 오가는 불안정 모드를 사용하여 발진 신호를 출력하도록 구현하였다.

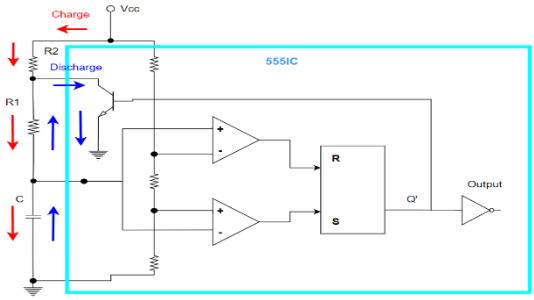


그림 6

555ic 의 내부 회로도가 포함된 불안정 모드의 회로를 살펴보면 그림 6 과 같다. 초기 상태에 플립플롭의 출력이 0 이므로, npn BJT(Bipolar Junction Transistor)의 Base 에 1 이 인가된다. 따라서 커패시터는 파란색의 Discharge 경로를 따라 방전된다. 이 과정에서 커패시터의 전압이 Vcc/3 까지 떨어지게 되면 하단부의 비교기 출력이 1 이 되어 플립플롭의 Q'가 0 이 된다. 이렇게 되면 npn BJT 는 동작하지 않게 되고, 커패시터는 다시 Charge 의 경로를 따라 충전된다. 충전 과정 또한 커패시터의 전압이 2Vcc/3 가

되면 상단부 비교기의 출력이 1 이 되어 플립플롭이 RESET 되며, 위의 과정을 계속 반복하게 된다. CLK 회로의 발진 주기는 충전 시간, 방전 시간의 합으로 정의되는데, 충전, 방전 경로가 그림 6 에 표시된 것과 같이 RC 직렬회로를 거친다. 따라서 아래 과도응답 방정식으로 충전 방전 시간을 계산하여 발진 주파수를 구한다.

$$Vcc(1-e^{-\frac{a}{\tau}}) = \frac{1}{3}Vcc, Vcc(1-e^{-\frac{b}{\tau}}) = \frac{2}{3}Vcc$$
$$b-a = \tau(ln2) \cong 0.693 \times \tau$$
$$T = 0.693x(\tau_r + \tau_f)$$

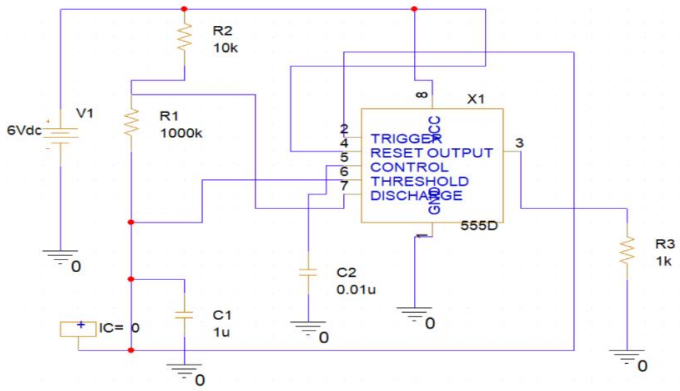
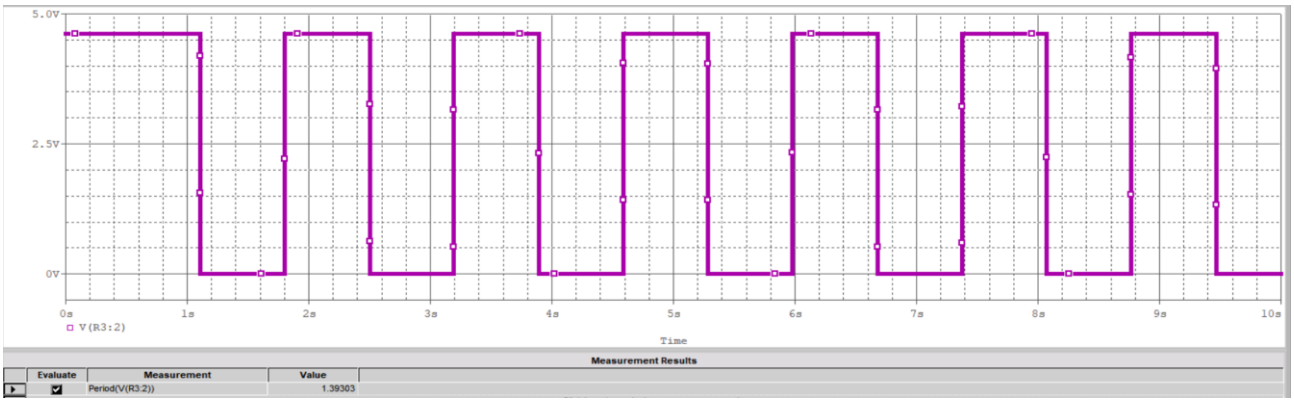


그림 7



시뮬레이션 2

시뮬레이션 2 를 통해 주기가 1.393s 인 구형파가 생성되는 것을 확인하였다.

C. 타이머 회로

타이머는 회로의 남은 테스트 시간을 출력하는 역할을 하며, 제한 시간이 끝나거나, 제한 시간이 끝나기 전 테스트를 완료하게 되면 타이머를 포함한 모든 회로의 동작을 정지시킨다. 해당 회로는 아래와 같이 설계되었다.

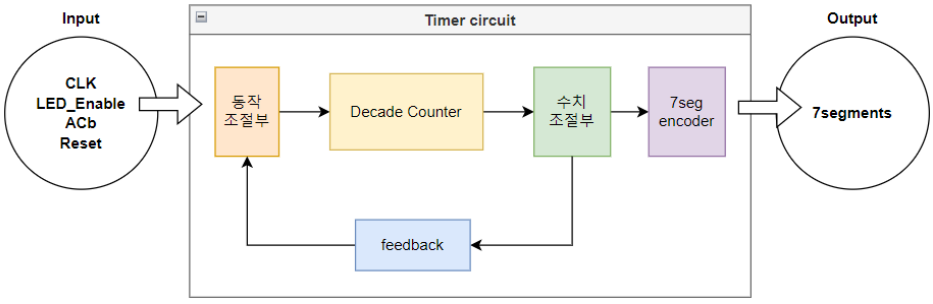


그림 8

동작 조절부는 외부의 Input 과 feedback 값을 입력 받아 타이머가 동작을 해야 하는지 중지해야 하는지 판단하여 전반적인 회로의 동작을 조절한다. 타이머가 동작을 해야 하는 상황이면, 입력 신호는 다음 블록인 Decade Counter 로 넘어가고, 매 CLK 마다 값이 1 씩 증가하는 신호를 만든다. 수치 조절부는 Adder 를 이용해 1001 (9)에서 Decade counter 값을 빼는 연산을 수행하며, 타이머의 동작에 맞게 숫자가 1001(9)부터 0000(0)까지 역순으로 줄어들도록 한다. 그 다음 수치조절부 출력을 7segment 신호로 인코딩하고, 숫자가 0000(0)이 되면 동작 조절부로 피드백 되는 값이 0 이 되어 타이머와 전체 회로를 중지시킨다.

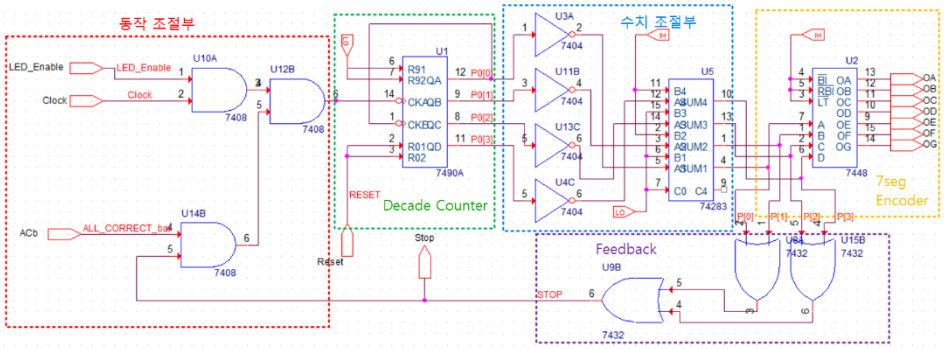
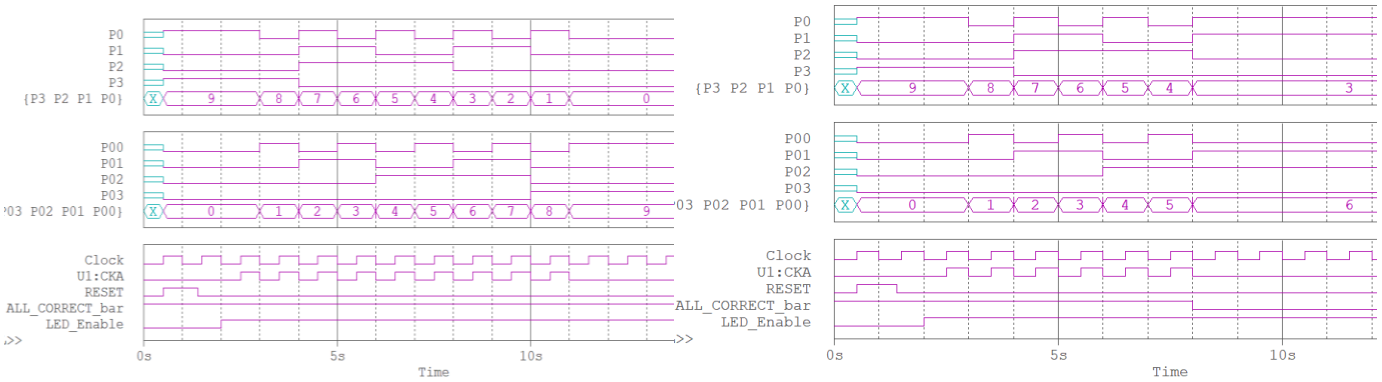


그림 9



시뮬레이션 3,4

시뮬레이션 3 은 제한 시간이 끝났을 때 타이머와 전체 회로가 정지하는 것을 보인 시뮬레이션이다. 동작 초반 0.5~1.4s 동안 RESET 을 눌러 회로를 초기화 시키면, Decade counter 는 값이 0 으로 유지된다. 2s 의 시점에 정답 점등 회로의 동작이 끝났다는 LED_Enable 신호가 들어올 때 타이머의 동작이 시작되며 Decade counter 의 출력이 매 CLK 마다 1 씩 증가한다. 이후 수치 조절부를 통해 타이머 회로의 출력 값이 9 부터 1 씩 떨어지다가 타이머의 동작이 끝나는 11s 시점부터 계속 0 의 출력이 유지되며 타이머 회로의 동작이 정지한 것을 확인하였다.

시뮬레이션 4 은 제한 시간이 끝나기 전 테스트를 완료했을 때 타이머와 전체 회로 동작이 정지하는 것을 보이기 위한 시뮬레이션이다. 시뮬레이션 5 과 동일하게 초반에 RESET 을 통해 회로를 초기화 하고, 2s 에 LED_Enable 신호가 켜진다. 8s 시점에 모든 문자가 일치하였다는 신호로 ALL_CORRECT_bar 값이 0 으로 변하면 타이머의 출력이 0011(3)으로 유지되는 것을 확인하였다.

3. 출력 모듈

출력 모듈은 최대 길이 8 의 랜덤한 문자열들을 생성하고 저장, 이를 랜덤한 순서대로 LED 에 출력하는 역할을 하며 LED 점등회로, 난수 발생기로 구성된다. 3X3 행렬로 배열된 LED 에 ㄱ, ㄴ, ㄷ,ㅁ 문양을 아래 그림 10 과 같이 랜덤한 순서대로 보여준 뒤 꺼진 상태를 유지한다.

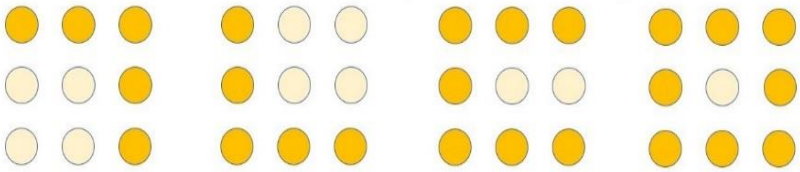


그림 10

A. LED 점등 회로

아래 그림 11 와 같이 LED 를 4 개의 영역으로 나눈 뒤 각 영역 내에 있는 LED 끼리 연결하여 LED 묶음을 만들면 각 4 개의 LED 묶음의 조합으로 원하는 문자 ㄱ, ㄴ, ㄷ, ㅁ을 모두 출력할 수 있다. 가령 문자 ㄱ을 출력 할 경우 1 번, 3 번, 4 번 LED 묶음이 동시에 켜지도록 하고, 문자 ㄴ을 출력하고자 할 경우 2 번, 4 번 LED 묶음을 동시에 켜지도록 한다.

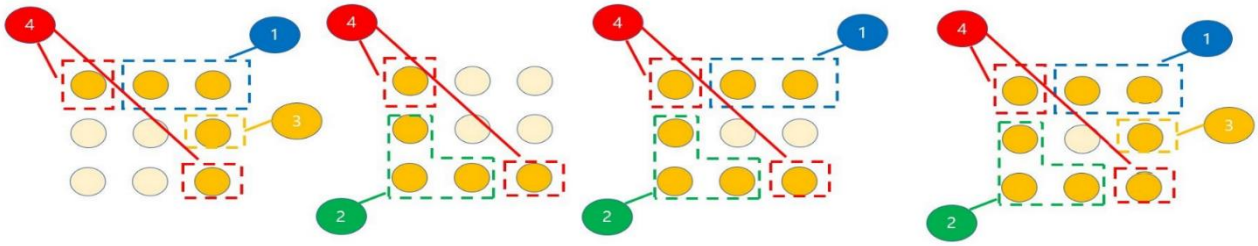


그림 11

이러한 방식을 통해 각 LED 를 별도로 제어하는 대신 4 개의 묶음으로 제어하여 필요한 제어 회로의 개수를 줄일 수 있으며, 각 LED 묶음에 대한 제어 회로는 아래 그림 12 에서 제시된 진리표와 회로도 같이 간단하게 구성할 수 있다.

	A	B	Start	1	2	3	4
소등	x	x	0	0	0	0	0
ㄴ	0	0	1	0	1	0	1
ㅁ	0	1	1	1	1	1	1
ㄷ	1	0	1	1	1	0	1
ㄱ	1	1	1	1	0	1	1

1 =(A+B)start 2 $=\overline{AB}$ 3 =B 4 =start

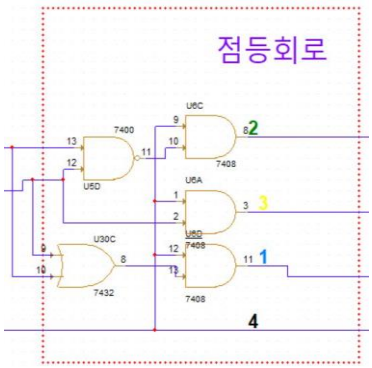


그림 12

B. 난수 발생기

난수 발생기는 랜덤한 비트 시퀀스를 출력하는 회로로 출력 모듈에서 생성해야 하는 랜덤한 문자들과 그 문자들의 출력 순서를 정한다. 랜덤성이란 어떤 결과 값을 예측할 수 없는 무작위성을 의미하며 반복 주기가 무한에 가까운 것을 말하는데, 이러한 특성을 가진 회로는 현실적으로 구현할 수 없다.

따라서 반복 주기가 유한하지만 이산균등분포 특성을 가지는 의사랜덤(Pseudo Random) 비트 시퀀스를 출력하는 LFSR(Linear Feedback Shift Register)로 회로를 구현하고 랜덤성을 최대화하기 위해 반복 주기를 최대화한다. 선형 피드백 함수는 XOR 이 이용되기 때문에 모든 비트가 0 으로 구성된 경우에는 피드백 값이 항상 0 이 되어 동작하지 않기 때문에 초기 값 seed 설정 시 해당 경우는 제외해야 하며 이로 인해

LFSR 에서 만드는 수열의 최대 길이는 n 개의 레지스터가 사용될 경우 $2^n - 1$ 된다. 또한 반복 주기는 피드백 선형 함수가 원시 다항식 형태를 가져야 최대 값이 얻어진다. 즉 LFSR 시스템의 반복 주기는 seed 값과 피드백 선형 함수 종류에 따라 결정된다. 본 작품에서는 아래 그림 13 과 같은 16 비트 피보나치 LFSR 을 차용하였으며, 특성 다항식은 $y = x^{16} + x^{14} + x^{13} + x^{11} + 1$ 으로 표현된다. 홀수 자리 출력과 짝수 자리 출력들이 각각 LED 점등 회로 A, B 로 순차적으로 입력되어 문자가 랜덤한 순서로 8 번 출력되도록 하였다. 발생할 수 있는 랜덤 문자 시퀀스의 반복 주기는 가능한 최대 반복 길이인 65,535 인 것을 파이썬으로 확인하였다.

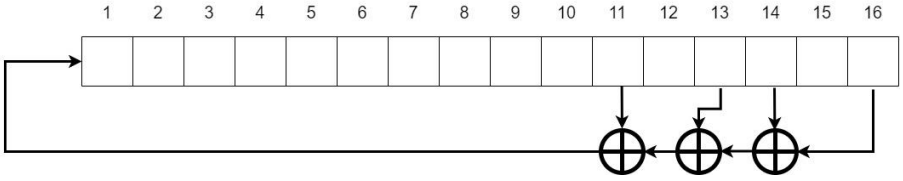


그림 13

XOR 피드백 이용 LFSR 은 seed 값이 모두 0 이면 안 되지만, 시뮬레이션에서 사용되는 레지스터, Counter 등의 소자는 초기 값이 0 일 때 정상 동작을 하는 문제점이 있다. 이를 해결하기 위해 XNOR 사용해 seed 값이 모두 1 인 경우 동작하지 않고 모두 0 인 경우 정상 동작하는 LFSR 로 바꾸어 다른 소자들과 함께 잘 동작할 수 있도록 하였으며, 회로와 발생 시퀀스, 시뮬레이션 결과는 각각 그림 14 과 시뮬레이션 5 과 같다.

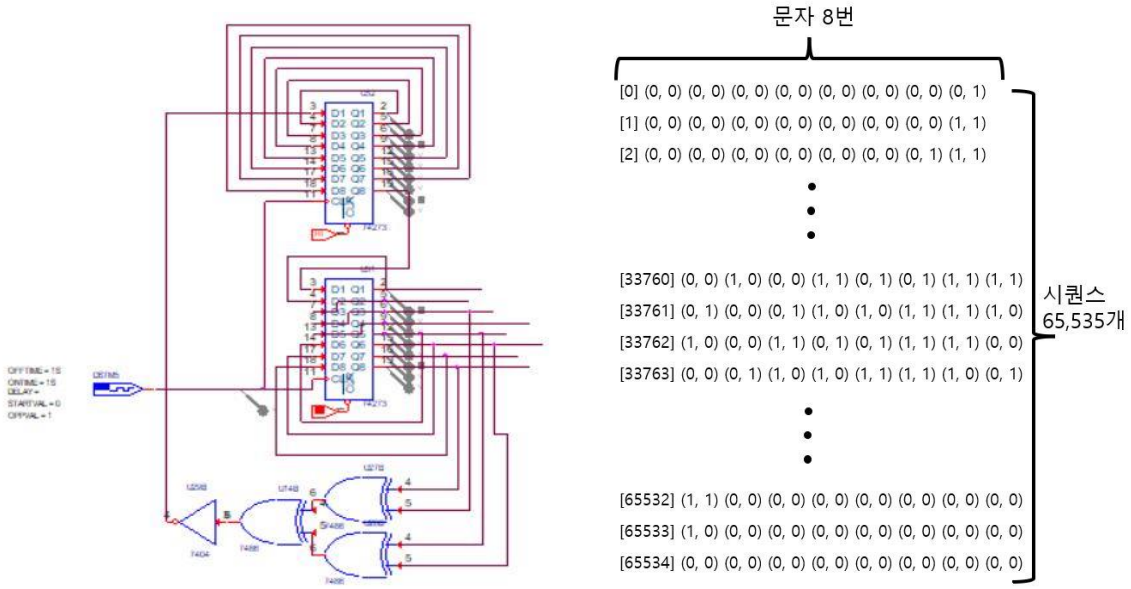
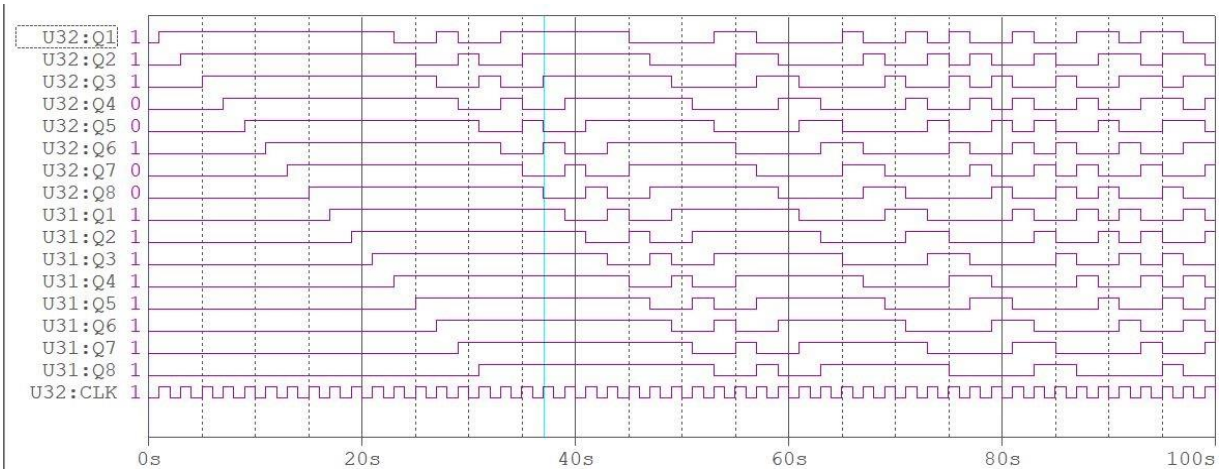


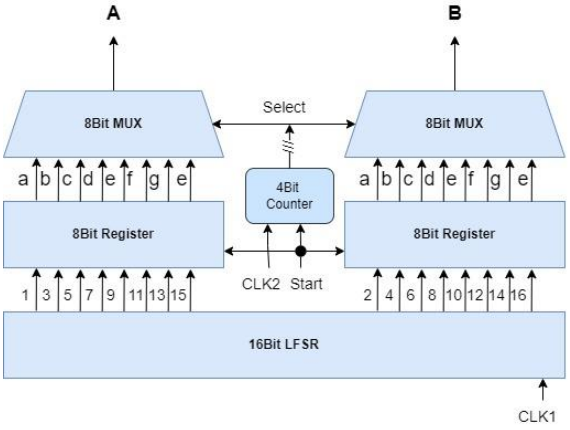
그림 14



시뮬레이션 5

C. 출력 모듈의 동작

LFSR 과 LED 점등회로은 그림 15 에 제시된 도면과 같이 8 비트 레지스터, MUX 로 연결된다. 전원이 켜지면 LFSR 에서 고주파 CLK1 신호마다 랜덤한 16 비트 시퀀스를 생성하고, 8 비트 MUX(Multiplexer) 2 개는 00 으로 초기화한다. 시작 버튼을 눌러 Start 신호 입력하면 생성된 난수의 홀수, 짝수 번째 비트들이 각각 8 비트 레지스터에 저장되고 MUX select 에 입력된 counter 출력이 저주파 CLK2 에 따라 1 씩 증가하면서 점등 회로 1,2,3,4 출력으로 문자가 순서대로 켜지도록 하였다. 문자가 8 번 출력 되어 counter MSB 에서 '1'이 출력 될 경우 저주파 CLK2 가 차단되어 counter 동작이 중지된다. 해당 동작은 아래 시뮬레이션 6 결과에서 확인할 수 있다.



4Bit 카운터	1234@ 0~20s	1234@ 25~40s
0	1011 ㄱ	1111 ㅁ
1	1101 ㄷ	1111 ㅁ
2	1111 ㅁ	0101 ㄴ
3	1011 ㄱ	1011 ㄱ
4	1101 ㄷ	1101 ㄷ
5	0101 ㄴ	1101 ㄷ
6	1111 ㅁ	1011 ㄱ
7	1011 ㄱ	1101 ㄷ

그림 15.출력 모듈 도면& 시뮬레이션 결과 정리 표

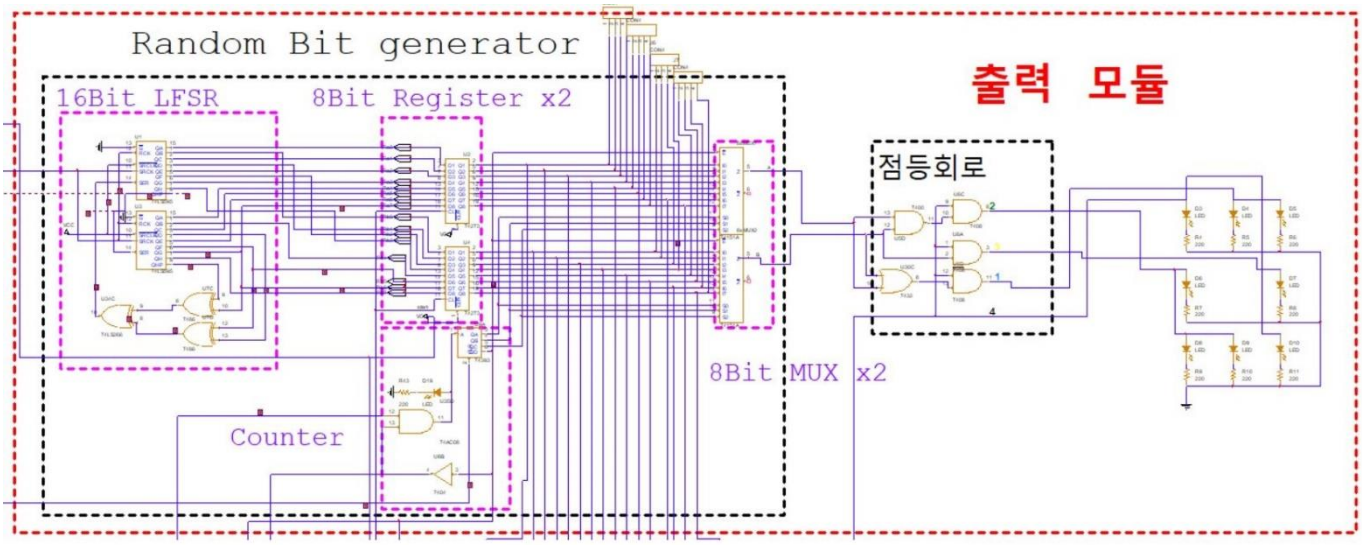
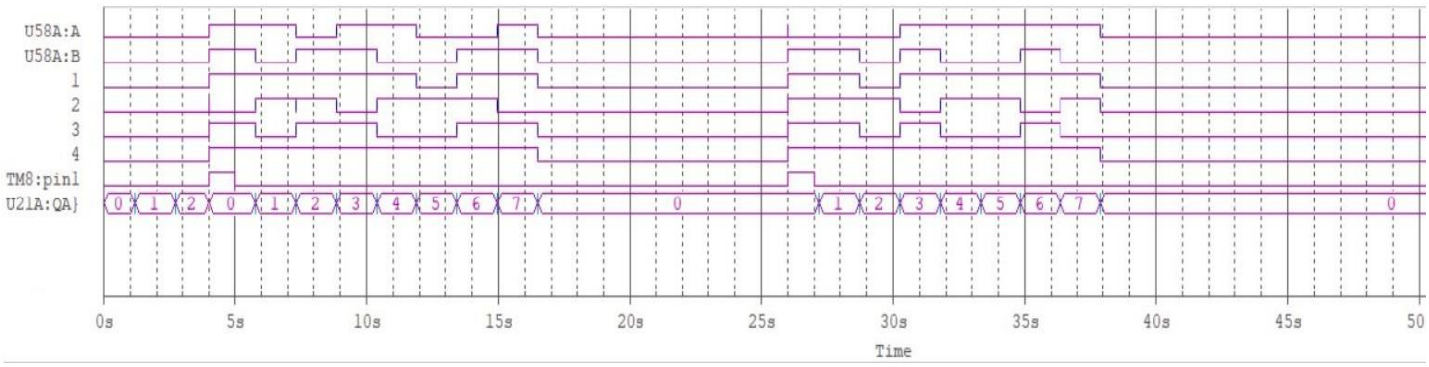


그림 16



시뮬레이션 6

IV 제작 결과

입력, 출력, 제어처리 모듈을 모두 연결한 회로는 그림 17 과 같으며, 해당 회로의 PCB 설계도면은 그림 18, 제작된 PCB 기판은 그림 19 이다.

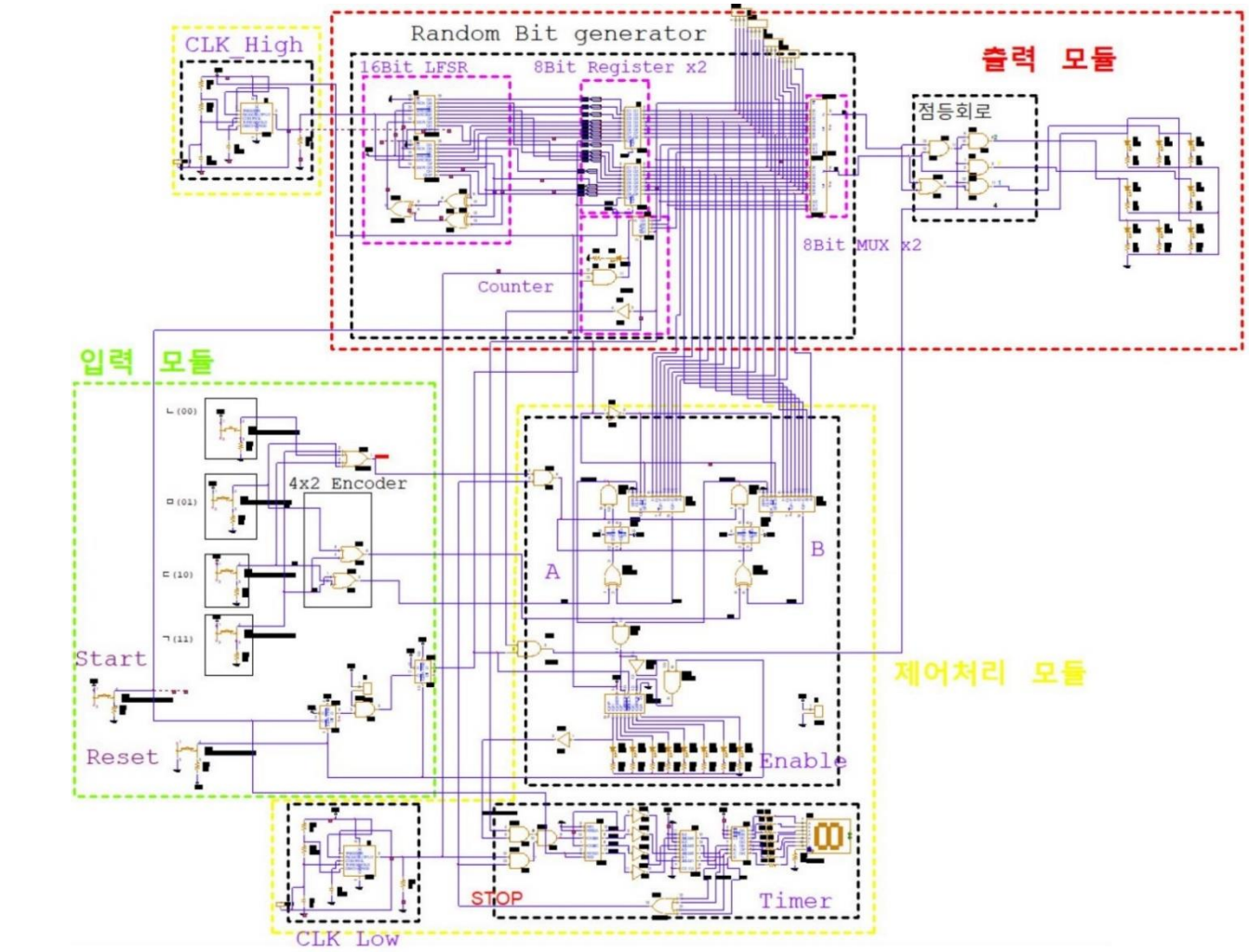


그림 17

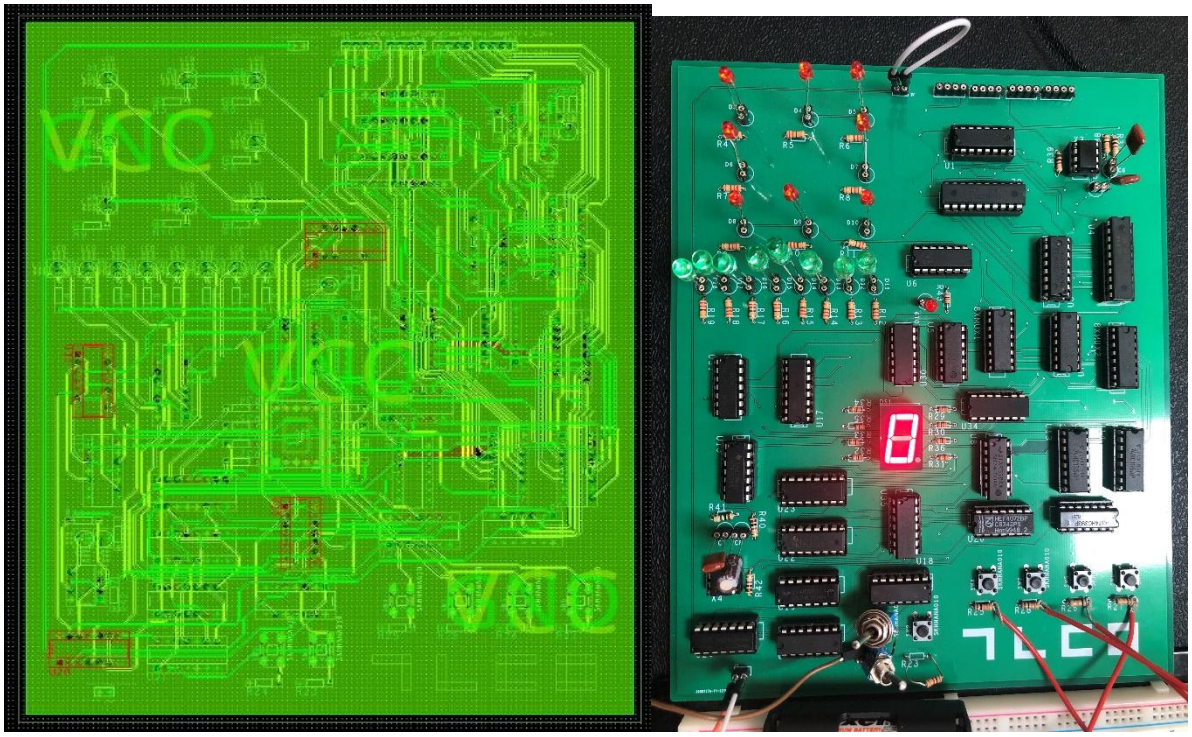


그림 18, 19

동작시연영상 링크: https://youtube.com/shorts/y3B2bPPi0kM

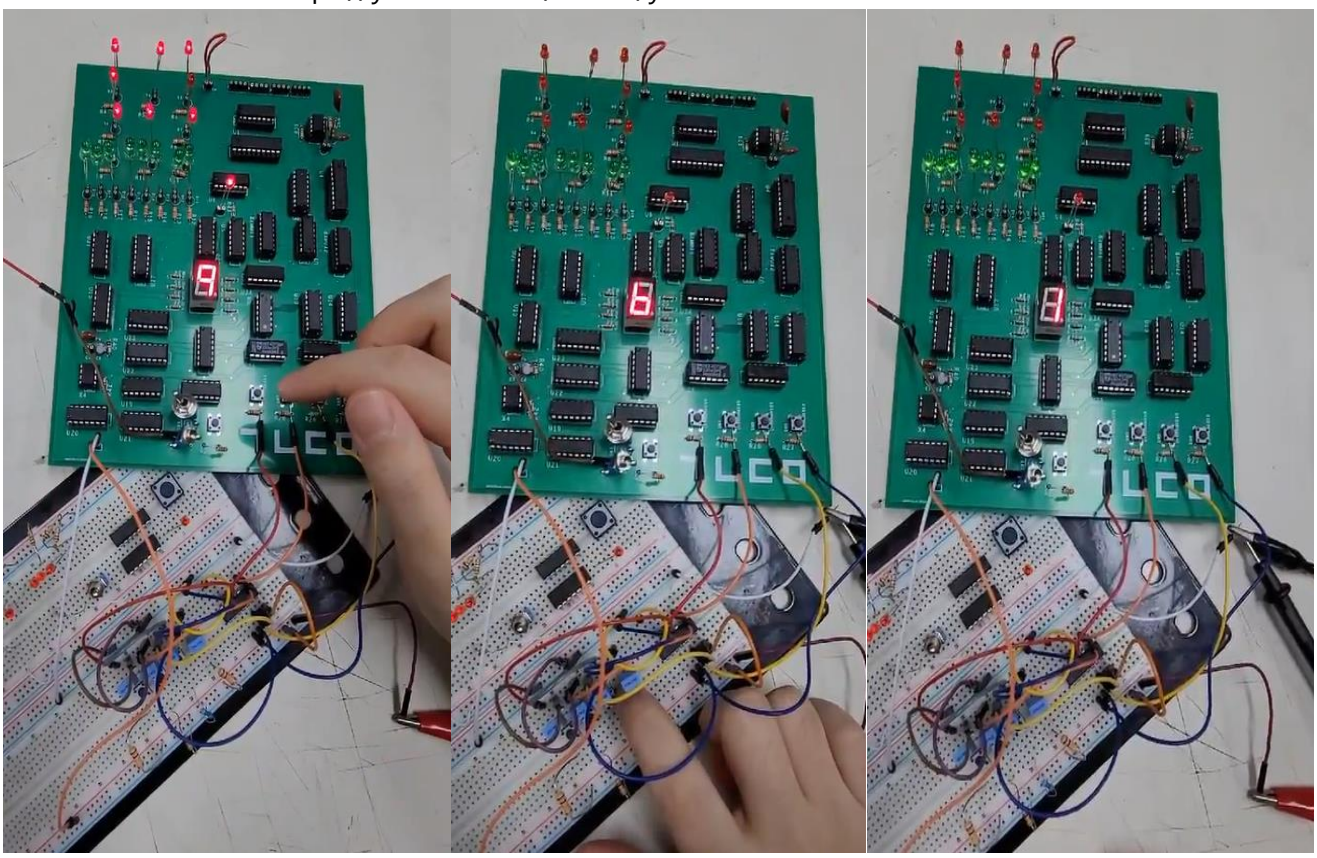


그림 20

그림 20-1) START 를 누름과 동시에 난수발생기에서 생성된 8 개의 정답문자열이 출력되는 것을 확인하였다. 우측에 연결된 단일 led 를 통해 연속된 문자를 구분할 수 있다.
그림 20-2) 정답문자열의 출력이 끝난 후 정답점등 LED 와 단일 LED 가 꺼지며, 타이머의 동작이 시작되는 것을 확인 가능하다. 정답 점등회로 하단의 녹색 LED 를 통해 입력의 일치 현황을 알 수 있다.
그림 20-3) 정답문자열이 모두 입력되면 녹색 LED 8 개가 모두 켜지며, 타이머가 정지하는 것을 확인하였다.

V. 시행착오

1. 개선 사항

기존의 구조에서는 문자 점등회로 입력 00 비트에는 문자를 할당하지 않고 무효값으로 설정하여 게임 시작 전과 끝난 후 LED 를 모두 끄도록 설계하였는데, LFSR 에서 출력되는 난수에도 00 이 포함 되므로 문자열이 짧아지는 테스트기 난이도 저하 문제를 일으킨다. 또한 기존 회로에서 아무런 입력을 받지 않았음에도 불구하고 정답처리가 되는 것을 방지하기 위해 발생한 문제를 해결하기 위한 Enable 회로를 추가 했는데 이로 인해 필요한 소자 수가 늘어나는 문제점이 생겼다.

이를 해결하기 위해 문자 점등회로 출력과 시작 신호를 AND 로 연결하여 테스트 시작 버튼을 눌렀을 때만 LED 가 켜지도록 구조를 변경하였고, 입력 00 에 문자 ㄴ을 할당하고 새로운 문자 ㅁ을 추가하였는데, 이로써 더 간단한 Enable 회로 구성이 가능해졌다. 기존에는 16 비트의 정답 문자에 대한 8 비트 일치 신호 출력을 위해 사용자가 입력한 답을 테스트 종료 때까지 저장하였고, 저장된 8 개의 정답 비트 모두에 대해 각각 비교기 XOR 를 할당하였는데, 이로 인해 많은 수의 메모리 와 논리소자가 추가로 필요하였다. 개선된 Enbale 회로는 Parallel shift register 와 두개의 비교기 xnor 을 도입해 사용자로부터 정답을 입력 받을

때마다 1 비트씩 순차적으로 처리 하도록 하여 저장해야할 데이터를 줄였고, 46 개 사용되던 소자를 13 개로 줄여 필요한 메모리 및 논리 소자의 71.7% 를 감소시켰다.

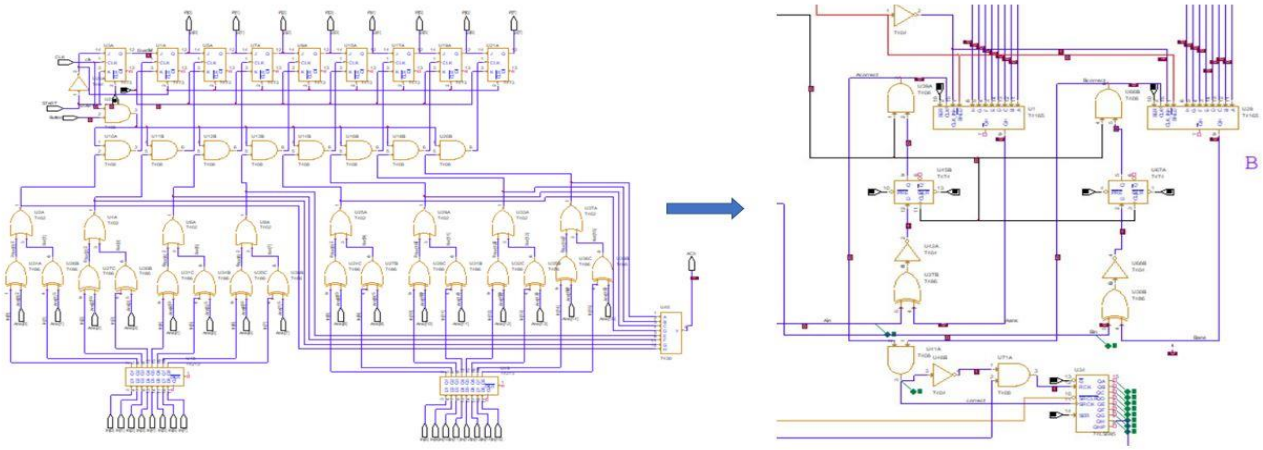


그림 21 기존 Enable, 22 개선된 Enable

2. PCB 제작 중 발생한 문제

사용하는 dip 소자의 개수가 많아 VCC 파워선을 배선하는 것이 어려워서 Top layer 에는 GND, Bottom layer 에는 VCC copper pour 를 도포하였다. 그러나 이 과정에서 그림 23 과 같이 pad 와 line 에 의해 고립된 island 영역에 VCC port 가 존재하여 해당 port 가 floating 이 되어버리는 문제가 발생하였다. 이에 배선작업이 모두 끝난 후에 그림 24 와 같이 island 를 구성하고 있는 line 을 Top layer 로 변경하여 island 를 없애며 문제를 해결하였다.

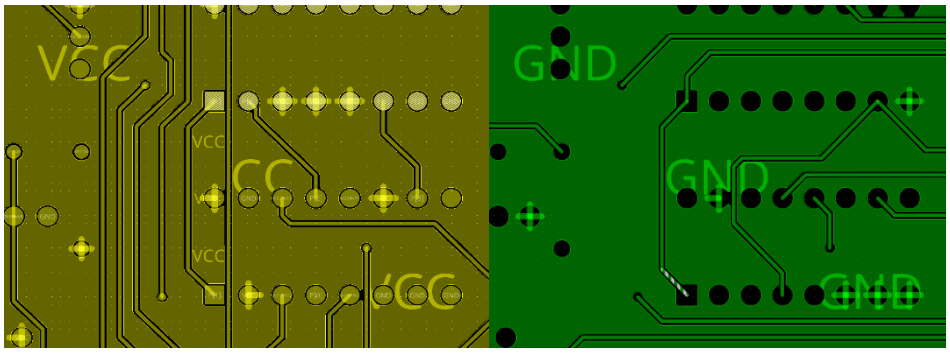
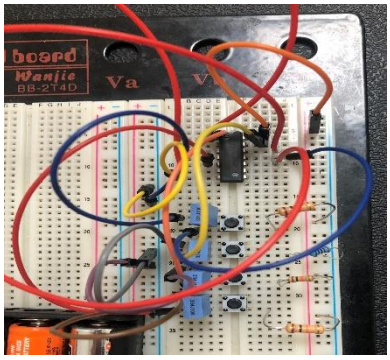


그림 23, 24

3. 스위치의 채터링 노이즈

브레드 보드에 회로를 구현하는 도중, 정답 문자가 연속해서 나온 경우일 때 사용자가 입력을 한번만 눌렀는데, 여러개의 정답판별 LED 가 동시에 켜지는 등 스위치를 한번 눌렀음에도 여러번 누른것처럼 동작하는 현상이 여러차례 관측되었다. 이는 물리적인 스위치의 구조 상, 접점을 개폐하는 과정에서 발생하는 진동에 의한 고주파 노이즈가 추가되면서 신호가 여러번 입력된것처럼 인식되는 채터링 현상이 원인으로 시뮬레이션 상에서 미처 확인하지 못한했던 상황이었다. 채터링 문제를 해결하기 위해 그림 26 과 같이 Low pass filter 를 이용해 고주파 노이즈를 제거하도록 하고 파형을 안정화하기 위해 슈미트트리거를 이용하였다.



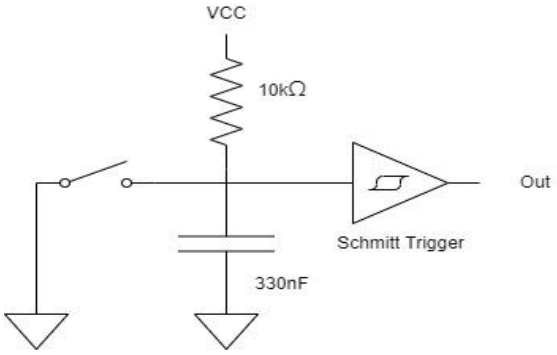


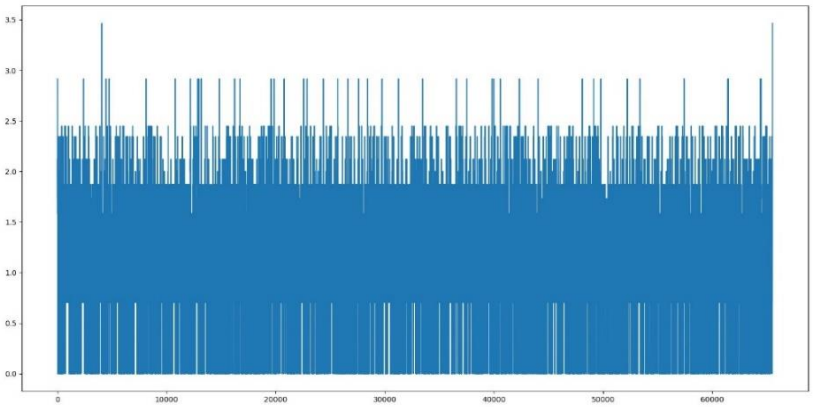
그림 25, 26

VI. 고찰

1. 특정 문자 편중으로 인한 테스트기 난이도 문제

우리가 설계한 LFSR 은 '1'로만 이루어진 비트 시퀀스를 제외한 모든 경우의 비트 시퀀스가 발생 가능하기 때문에 특정 문자만 반복해서 나오는 등 문자의 편중으로 인해 테스트기 난이도가 낮아지는 경우가 존재한다. 기억력 테스트기의 적절한 난이도를 설정하고 성능을 개선하기 위해 전체 경우 중 해당 시퀀스가 얼마나 있는지 파악하고 편중도를 조절하기 위한 회로를 제안하고자 한다.

파이썬을 이용하여 전체 발생 가능한 비트 시퀀스 65,535 개를 구하고 각 시퀀스에서의 ㄱ, ㄴ, ㄷ, ㅁ발생 빈도의 표준편차를 구해 전체 시퀀스 중 문자가 편중된 시퀀스가 얼마나 있는지 구하였다. 아래 그림 27 에 제시된 그래프를 통해 전체 시퀀스에서 편중된 시퀀스가 분포가 랜덤하다는 것을 확인하였으며 각 편중 문자 조합 별 표준편차는 아래 표에 제시되었다.표준편차가 가장 작은 0.0 일 때는 각 문자가 2,2,2,2 번 골고루 나오는 경우이고 가장 높은 3.464 는 특정 문자만 8 번 나오는 경우이다. 우리는 특정 문자가 5 번 이상 나오는 시퀀스들이 테스트기 난이도를 저하한다 판단하였는데, 이에 해당하는 편차 1.7 이상인 시퀀스들의 발생 확률은 11.559%로 꽤 높은 빈도로 발생하여 이를 제한하고자 한다.



표준편차	편중 문자조합
3.464	8 0 0 0
2.915	7 1 0 0
2.449	6 2 0 0
2.345	6 1 1 0
2.121	5 3 0 0
2.0	4 4 0 0
1.871	5,2 1, 0
1.732	5 1 1 1
1.581	4 3 1 0
1.414	4 2 2 0
1.225	4 2 1 1
1.0	3 3 1 1
0.707	3 2 2 1

```
ㄱ : 131064 개  ㄴ : 131072 개  ㄷ : 131072 개  ㅁ 131072 개
편차 1.7이상인 시퀀스 개수: 7575 개
편차 1.7이상 발생확률: 11.559 %
```

그림 27

편중도를 줄이기 위해 다른 선형함수를 사용하는 방안은 편차 1.7 이상인 시퀀스 발생 확률은 여전히 11.5% 안팎을 유지하고 반복 주기의 길이가 줄어드는 문제가 있어 적합하지 않다. 따라서 우리는 난수 발생기에서 특정 문자가 편중되는 경우를 감지하면 난수 발생기 값을 레지스터에 저장하지 않고 다음

시퀀스를 기다렸다가, 편중되지 않은 시퀀스가 나올 때 레지스터에 저장하는 방식과 그에 대한 'BSS(Bias Sequence Skipper)'회로를 제안한다.

2. 문자 편중 문제 해결

BSS 는 특정 문자가 몇 번 감지되었는지 세는 편중감지회로와 감지된 문자 수가 기준을 넘는지 판단하는 비교기로 구성된다. 편중된 경우가 감지되면 입력된 시작 신호가 난수 저장 레지스터 CLK 으로 들어가는 것을 억제하다가 편중되지 않은 경우가 감지되면 시작 신호가 CLK 으로 들어가도록 해 레지스터에 난수를 저장하는 방식으로 동작한다.

A. 편중감지회로

편중감지회로는 8 비트의 문자 시퀀스 중 특정 문자 발생 시 '1'을 입력 받고 '1'이 몇 개 있는지 BCD 로 출력해야 한다. 가령 0010(2)+0100(4)을 입력 받을 때 0110(6)이 출력 되는 것이 아니라, 입력에 1 이 2 개 있다는 뜻에서 0010(2)이 나와야 한다. HA(Half Adder)는 1 비트 이진수 연산을 하는 소자인데, 1 의 개수를 세는 것과 같은 결과를 가지므로 그림 28 과 같이 8 비트를 2 비트씩 4 개의 HA 로 연산을 하고 FA(Full Adder)로 각 값들을 더하도록 하면 입력에 있는 1 의 개수를 얻을 수 있다. 시뮬레이션 9 에서 입력 a0~a7 에 들어온 1 의 개수가 비교기 출력 S 에서 BCD 값으로 출력 되는 것을 확인하였다.

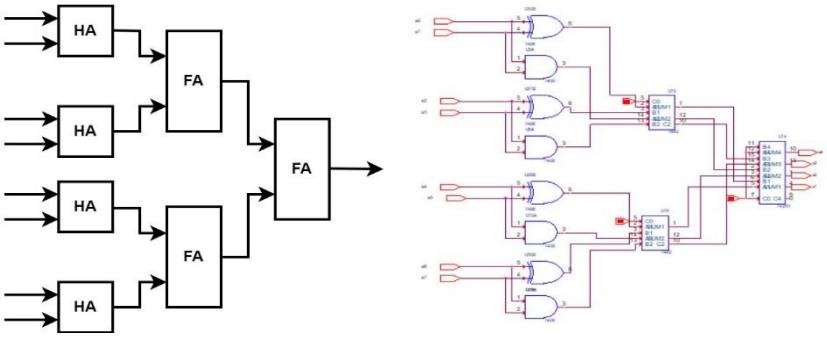


그림 28,29

시뮬레이션 7

B. BSS 적용

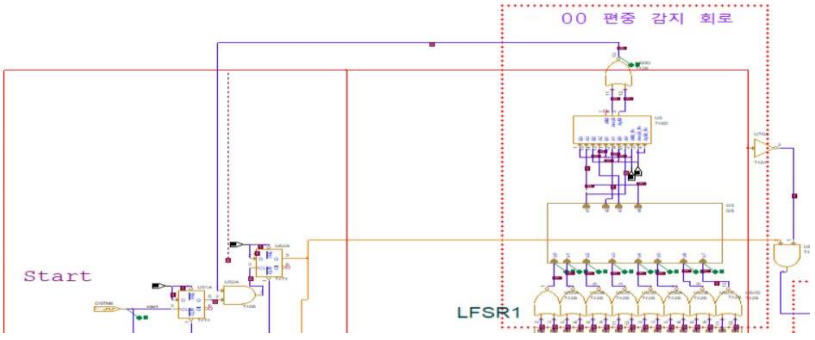
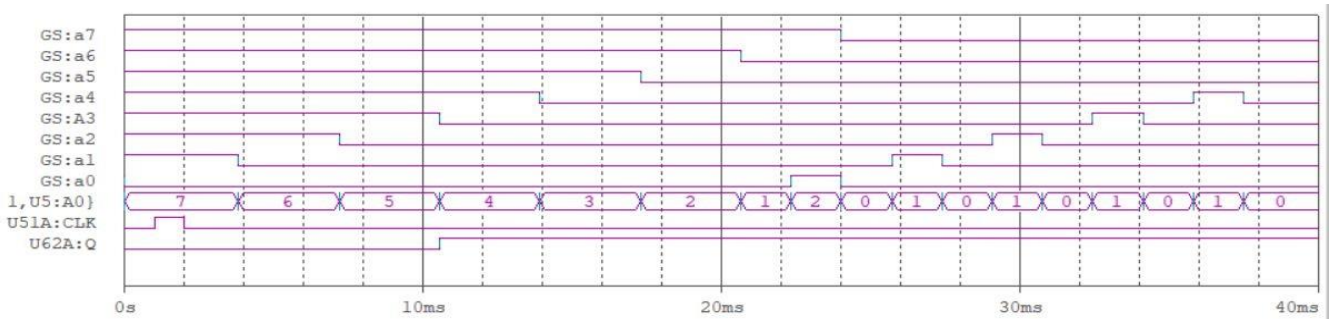


그림 28.BSS 회로



시뮬레이션 8.BSS 시뮬레이션

1~2ms 에 시작 버튼을 눌렀다 떼면 00(ㄴ)이 5 번 이상 감지될 때는 start 입력 신호(U62A:Q)가 들어가는 것을 차단해(skip) 레지스터 값의 모든 비트가 0 으로 유지되다가, 11ms 지점에서 1 의 개수가 4 개 이하가 될 때 start 신호는 레지스터 CLK 을 트리거하도록 1 로 변하면서 DFF 에 저장되는 것을 확인하였다. 편차 1.7 이상인 시퀀스 7,575 개 중 00 값이 5 번 이상 나오는 경우는 1,789 개이며 해당 값들이 skip 돼 레지스터에 저장되지 않는다고 하면 편중 문자(ㄴ) 발생 확률은 11.559%에서 9.077%로 낮아졌다. 이를 통해 한 문자에 BSS 적용시 편중도가 2.482%p 만큼 줄인 것을 알 수 있으며 BSS 를 4 가지 문자 모두에 대해 적용하면 대략 9.928%p 정도 줄일 수 있을 것으로 예상된다.

```
(0,0) 이 5번 이상 나오는 경우:  1789 개
개선시 확률: 9.077 %
```

그림 30

이처럼 BSS 를 무효 값에 적용하면 문자 편중도를 줄여 난이도 조절을 할 수 있다. ㄱ, ㄴ, ㄷ, ㅁ 문자 모두에 대해 적용하고 특정 문자가 많이 나오는 경우를 skip 함으로써 테스트기 난이도 조절과 성능 개선을 할 수 있지만, 회로가 커진다는 단점이 있으므로 Nor gate 를 이용해 8 개의 랜덤 문자 시퀀스에서 00 (ㄴ)이 나오는 경우에만 1 을 출력하는 경우만 예시로 제작하였으며, 난수 감지 논리에 따라 조절 가능하다. 또한 편중 여부 기준은 비교기로 제어하도록 하였다. 00(ㄴ)이 7 번 이상, 3 번 이상 나오는 등의 경우에 감지하도록 변경 가능하며, 아예 한번도 나오지 않도록 조절하는 것도 가능하다. 탈부착식으로 설계하여 사용자 기호에 맞게 난이도 조절이 가능하도록 하였다.

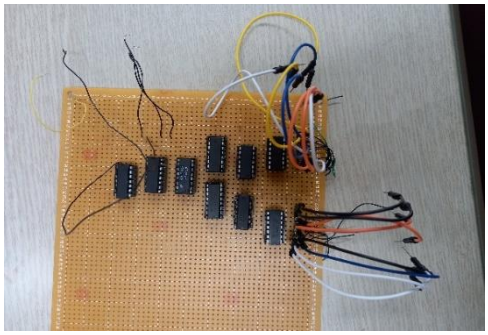


그림 31

VII. 결론

최근 크게 대두되는 이슈인 치매는 환자 본인의 사회 활동 제약으로 인한 고립, 경제적 문제, 가족의 돌봄 부담 등 우리 사회에 큰 문제를 일으키는데, 완치시킬 수 있는 방법은 아직 없어 질병의 진행을 억제하는 더불어 약들과 손상되지 않은 인지 영역을 극대화하는 기억력 훈련을 수행한다. 이를 위해 병원 등에 방문하기에는 고령자들 중 거동이 불편하신 분이 많으시고, 자택에서 컴퓨터나 스마트폰의 응용 프로그램을 사용하는데도 어려움을 느끼는 분들이 많다. 본 기억력 테스트기를 통해 병원을 방문하지 않고 IT 기기에 익숙하지 않은 분들도 쉽게 자택에서 기억력 훈련을 수행하는 효과를 기대한다.