경북대 IDEC 경진대회

Black Jack with Memorable Adder

팀명 : 뉴런(New Run)

CONTENTS

01
과제 선택 이유

Hazard의 발생을 억제하여

정밀한 작업을 요하는 분야에서

오류가 나는 것을 방지한다.

02
회로 설계 과정

1. 랜덤 카드 추출기
2. 드로우 차단기
3. 기억형 가산기
4. 비교기
5. 디코더

03
실제 동작

HBE-LogicLab-D를 활용하여

블랙젝 게임 구동성공

Hazard의 발생을 제거했다.

04
Q&A

Thanks

1. 과제 선택 이유

과제 선택 이유

기존의 제품의 **단점인 노이즈를 개선한 신제품을 개발**하는 것이 목표이다.

기본적인 논리회로 게이트 Full-Adder는
기억능력이 없는 소자이다.

그래서 기억능력을 더해주기 위해서는 Flip-Flop으로 구성된 레지스터가 필요하다.

하지만, 입력과 출력의 계산이 즉각적으로 이루어지는 Full-Adder와 다르게
레지스터는 입출력에 지연이 생긴다.

이것은 작동시간이 어긋나는 부분을 만들고
Hazard를 발생시킨다.

ms단위의 Hazard이기에 무시할 수 있지만
정밀한 동작을 원하는 분야에서는 이러한 노이즈조차 큰 문제로 이어질 수 있다.

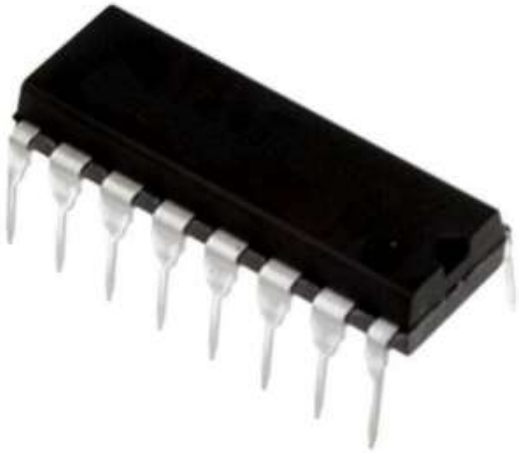
1. 과제 선택 이유

Full-Adder

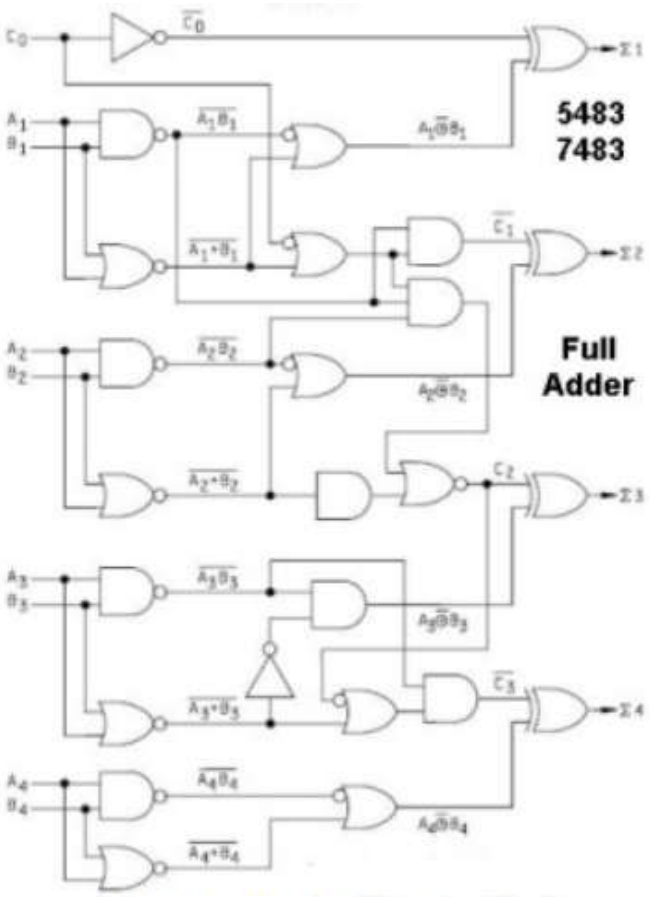
- 기본적으로 제공하는 Full-Adder는 기억능력이 없는 소자이다.
- 블랙잭 게임을 만들기 위해서는 뽑았던 카드의 숫자를 기억하고 지속적으로 더해주는 작업이 필요하다.

↓

Full-Adder를 단독으로 사용하는 것은 불가능



기억능력이 없는 Full-Adder

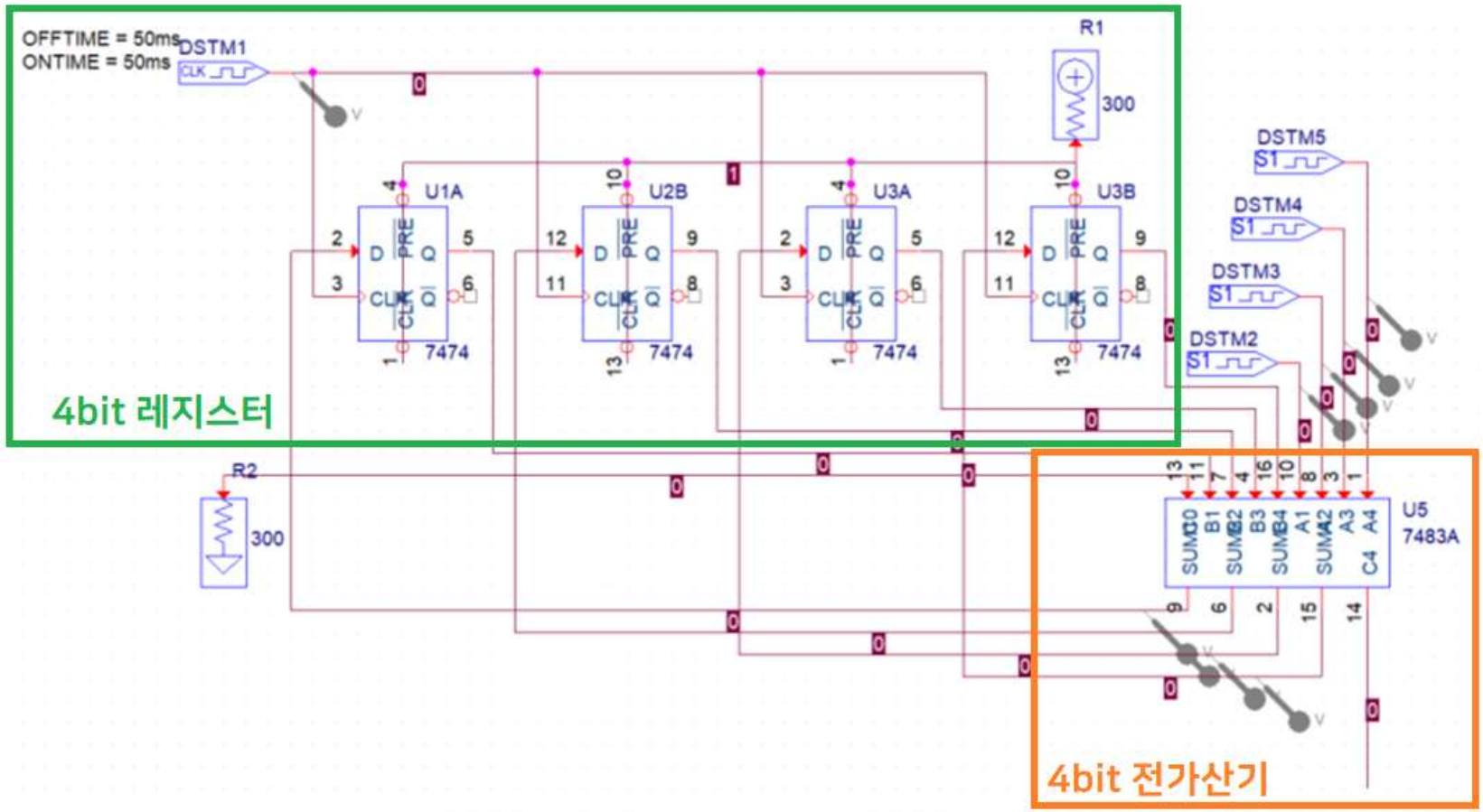


5438 4-Bit Full Adder Chip

1. 과제 선택 이유

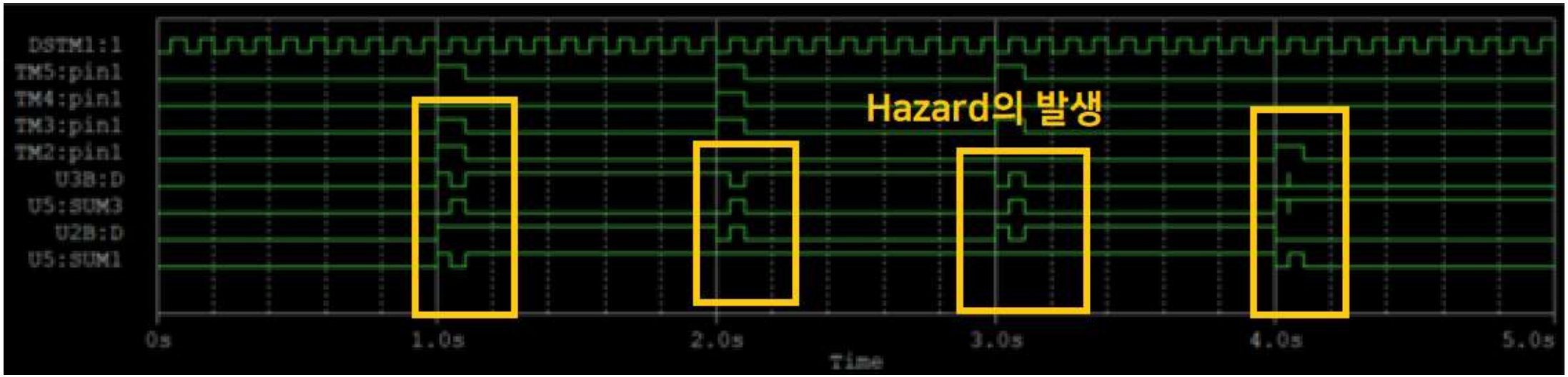
기존의 방식

- 기억능력이 있는 n개의 D플립플롭을 활용하여 n-bit 레지스터를 만든다
- 가산기의 출력을 다시 D플립플롭의 입력으로 루프를 형성하여 기억능력이 있는 가산기를 제작한다.
- 우리조는 Pspice를 활용하여 간단한 4-bit 가산기를 제작하여 시뮬레이션을 돌려보았다.



1. 과제 선택 이유

시뮬레이션 결과



- Full-Adder와 레지스터의 작동에지가 일치하지 않아서 Hazard가 발생
- ms 단위의 오차이기 때문에 블랙잭 게임을 완성하는데는 무리가 없는 수준
- **정밀한 분야에 사용 시 문제가 발생할 가능성**이 있음.

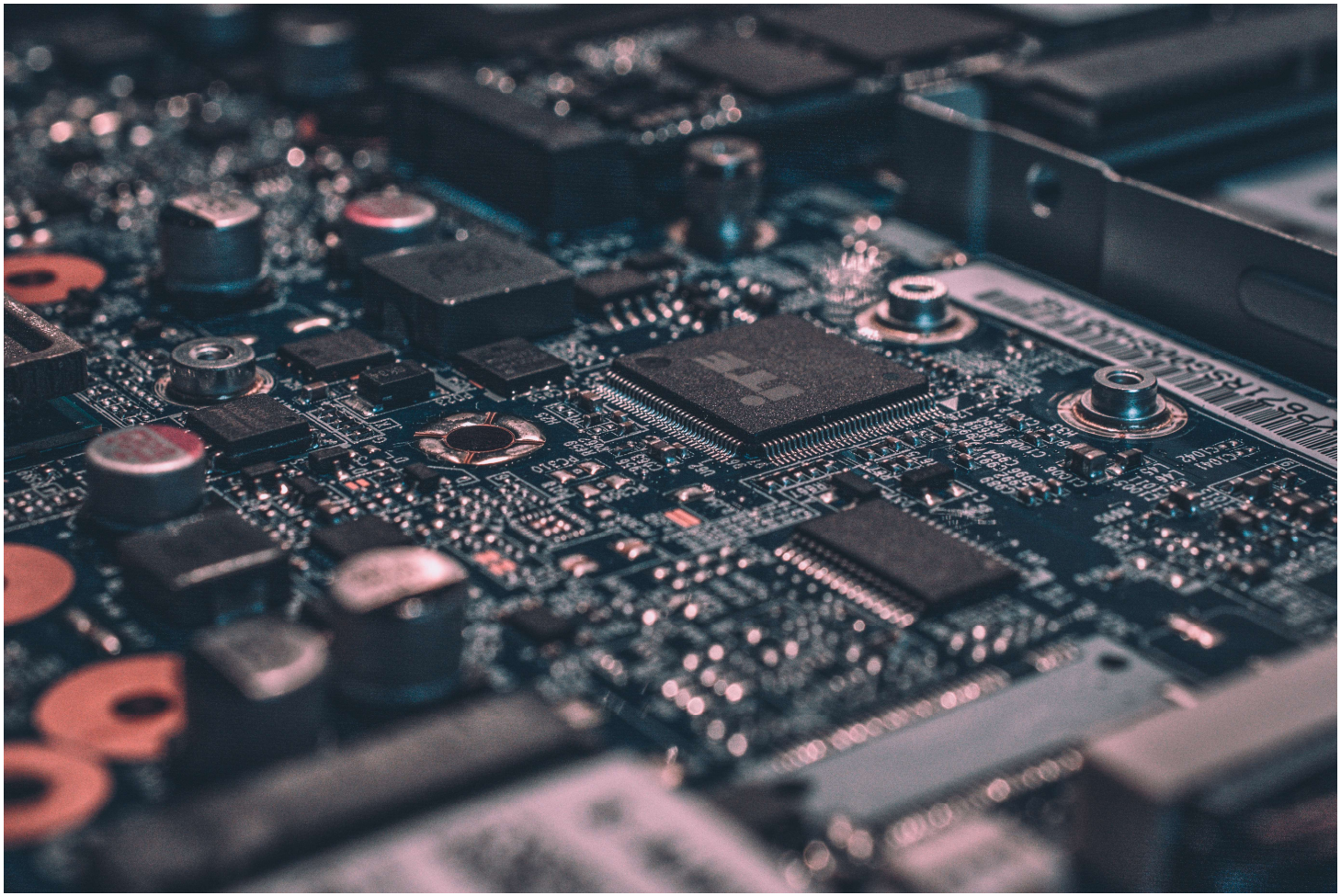
1. 과제 선택 이유

해결방법 고안

- 기억능력을 얻기 위해서는 플립플롭이 필수적이다.
- 다양한 요인이 있지만, 해저드는 플립플롭과 전가산기의 작동에지의 불일치로 인해 발생한다.

↓

- **가산기 자체를 플립플롭으로 구성**할 수 있다면 기억능력이 있는 가산기를 만들 수 있을 것이다.



2. 회로 설계 과정

2. 회로 설계 과정

카드 카운터(1)

블랙잭에서 숫자 카드는 자체의 숫자로 세지만 **페이스카드(Face Card)는 모두 10으로 간주**한다.
그래서 일반적으로 한 슈트에서 뽑을 수 있는 '숫자'는 다음과 같다.

"A-2-3-4-5-6-7-8-9-10-10-10-10"

우리조는 카드가 나올 수 있는 확률을 동일하게 맞추기 위해 난수 발생기를 사용하지 않고 카운터를 활용하였다.

실제로 랜덤은 아니지만, 매우 빠른 속도로 카운팅을 하기 때문에 카드를 뽑는 입장에서는 랜덤과 같다

A-2-3-4-5-6-7-8-9-10-10-10-10 카운터

	T						T+1						
	F	E	D	C	B	A	F	E	D	C	B	A	
0	0	0	0	0	0	0	0	0	0	0	0	1	
1	0	0	0	0	0	1	0	0	0	0	1	0	2
2	0	0	0	0	1	0	0	0	0	0	1	1	3
3	0	0	0	0	1	1	0	0	0	1	0	0	4
4	0	0	0	1	0	0	0	0	0	1	0	1	5
5	0	0	0	1	0	1	0	0	0	1	1	0	6
6	0	0	0	1	1	0	0	0	0	1	1	1	7
7	0	0	0	1	1	1	0	0	1	0	0	0	8
8	0	0	1	0	0	0	0	0	1	0	0	1	9
9	0	0	1	0	0	1	0	0	1	0	1	0	10
10	0	0	1	0	1	0	0	1	1	0	1	0	10
10	0	1	1	0	1	0	1	0	1	0	1	0	10
10	1	0	1	0	1	0	1	1	1	0	1	0	10
10	1	1	1	0	1	0	0	0	1	0	1	1	A/11
A/11	0	0	1	0	1	1	0	0	0	0	1	0	2

카드 카운터(2)

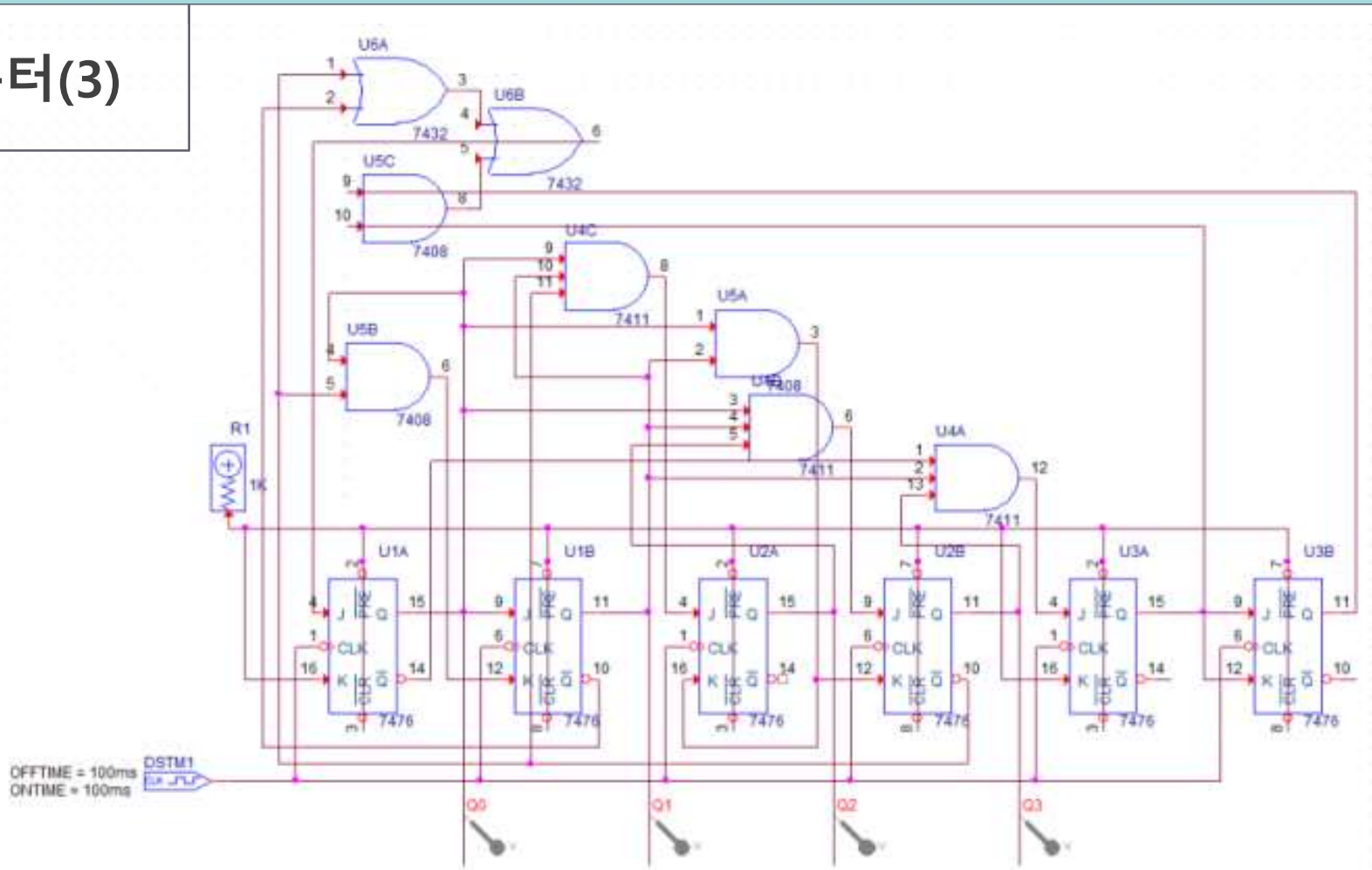
비트	입력	논리식	비트	입력	논리식
F	J	E	C	J	ABD'
	K	E		K	AB
E	J	A'BD	B	J	A
	K	1		K	AD'
D	J	ABC	A	J	D'+B'+EF
	K	AB		K	1

JK 플립플롭으로 구현한 진리표

F		E		D		C		B		A	
J	K	J	K	J	K	J	K	J	K	J	K
0		0		0		0		0		1	
0		0		0		0		1			1
0		0		0		0			0	1	
0		0		0		1			1		1
0		0		0			0	0		1	
0		0		0			0	1			1
0		0		0			0		0	1	
0		0		1			1		1		1
0		0			0	0		0		1	
0		0			0	0		1			1
0		1			0	0			0	0	
1			1		0	0			0	0	
	0	1			0	0			0	0	
	1		1		0	0			0	1	
0		0			1	0			0		1

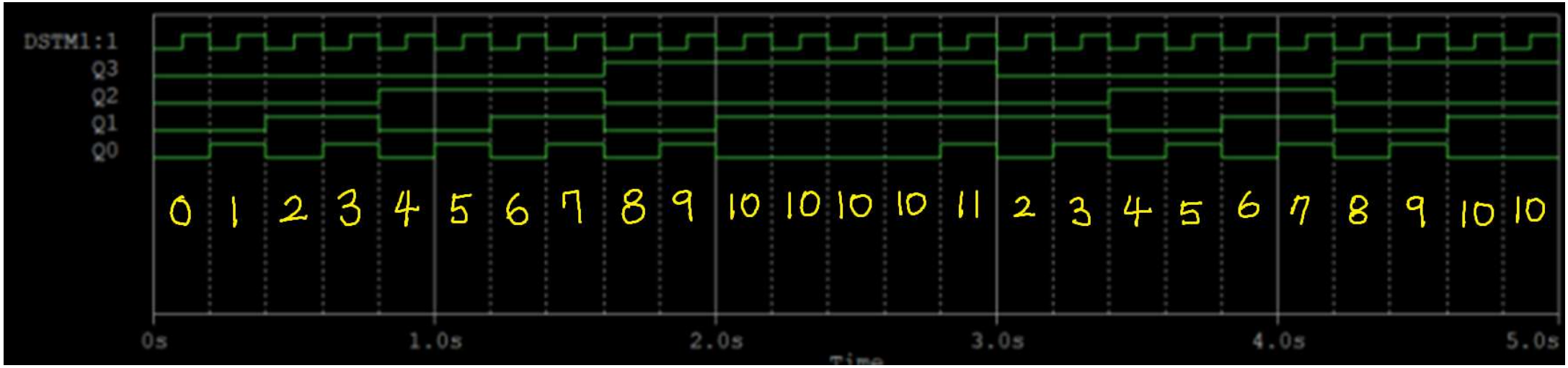
2. 회로 설계 과정

카드 카운터(3)



2. 회로 설계 과정

카드 카운터(4)



위의 그림과 같이 **“0-1-2-3-4-5-6-7-8-9-10-10-10-10-11-2-3…”** 순으로 숫자가 성공적으로 카운팅됨을 알 수 있습니다.

드로우 차단기(1)

카드를 선택하는데 스위치를 쓸 것이다.

하지만, 스위치를 쓸 때 아무리 빨리 눌러도
카운터에서 하나의 숫자만 선택할 정도로 빠른 것은 아니다.

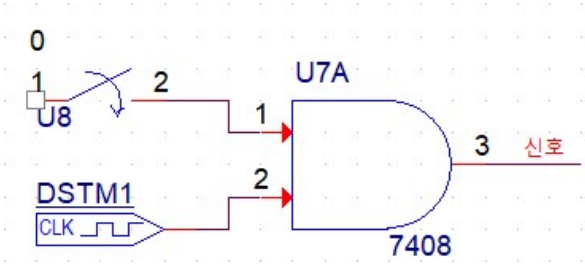
즉, 버튼을 누를 때 다수의 신호가 전달되어

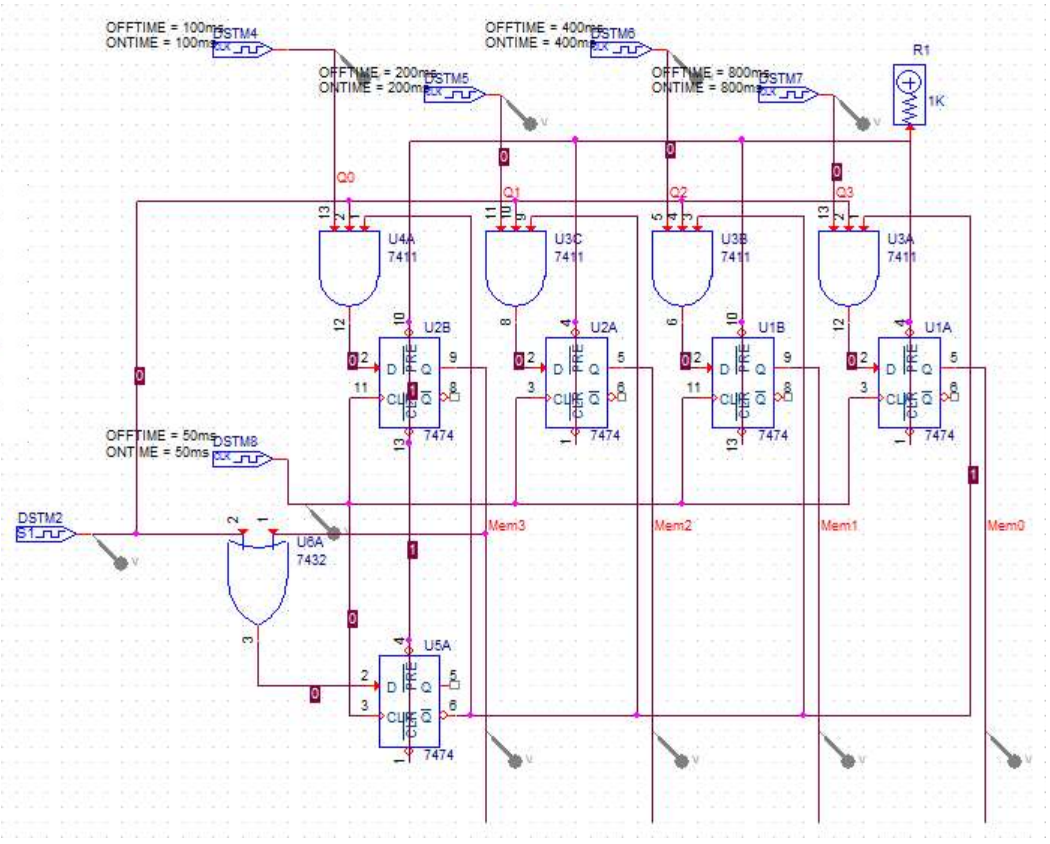
몇 가지의 수가 계속하여 인식될 것이다.

이런 현상을 방지하기 위해 버튼을 누르는

시간과 상관없이 하나의 신호만 출력되도록

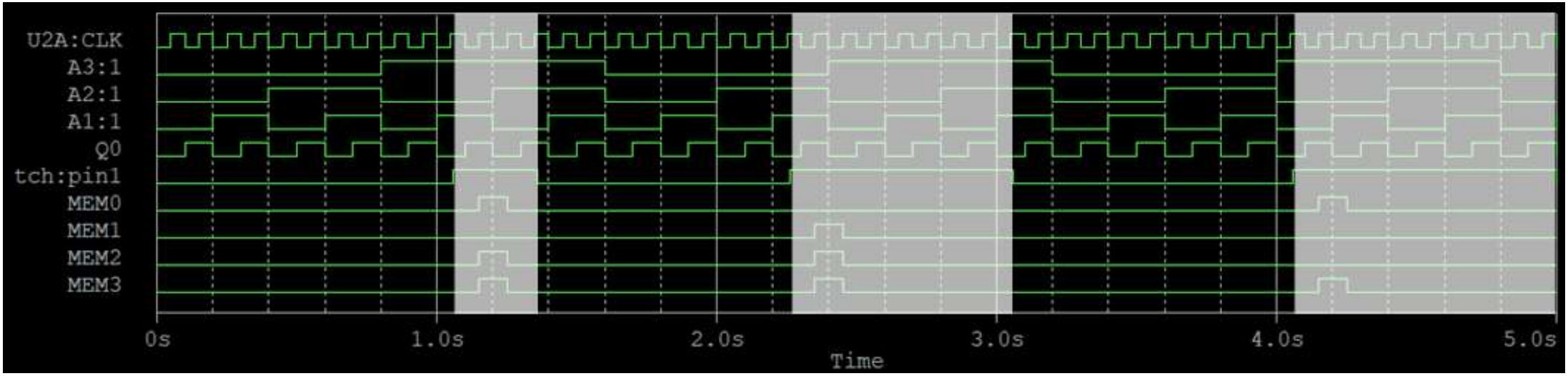
설계가 필요하다.





2. 회로 설계 과정

드로우 차단기(2)



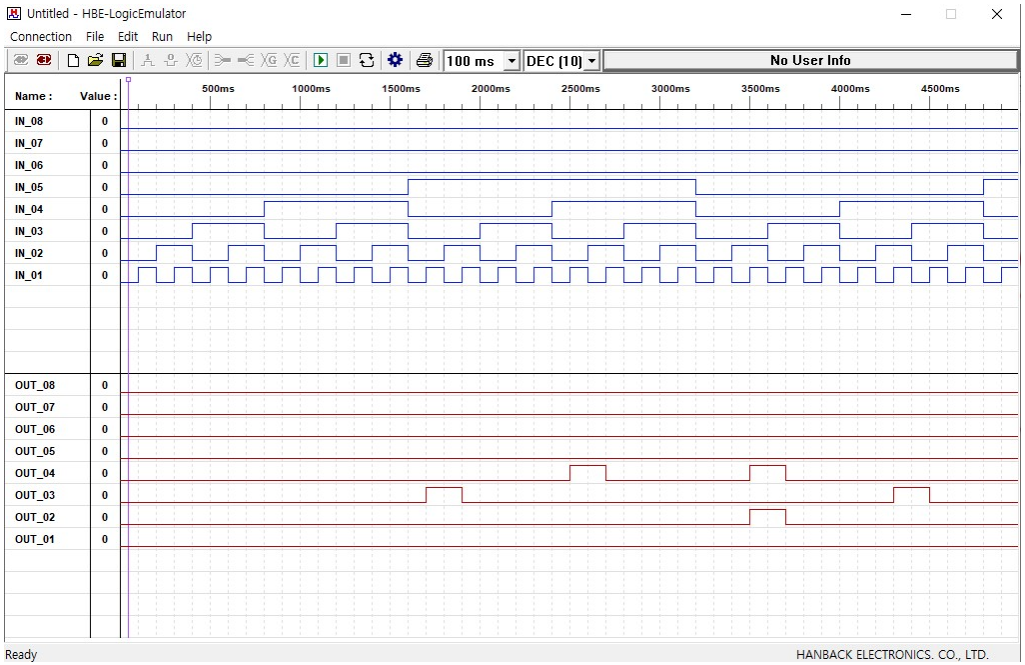
스위치는 임의의 구간을 선택하여 입력을 주었으며 위 결과에서 알 수 있듯이 스위치를 눌렀을 때 그 주기에서 카운팅하던 숫자가 출력됨을 알 수 있다.

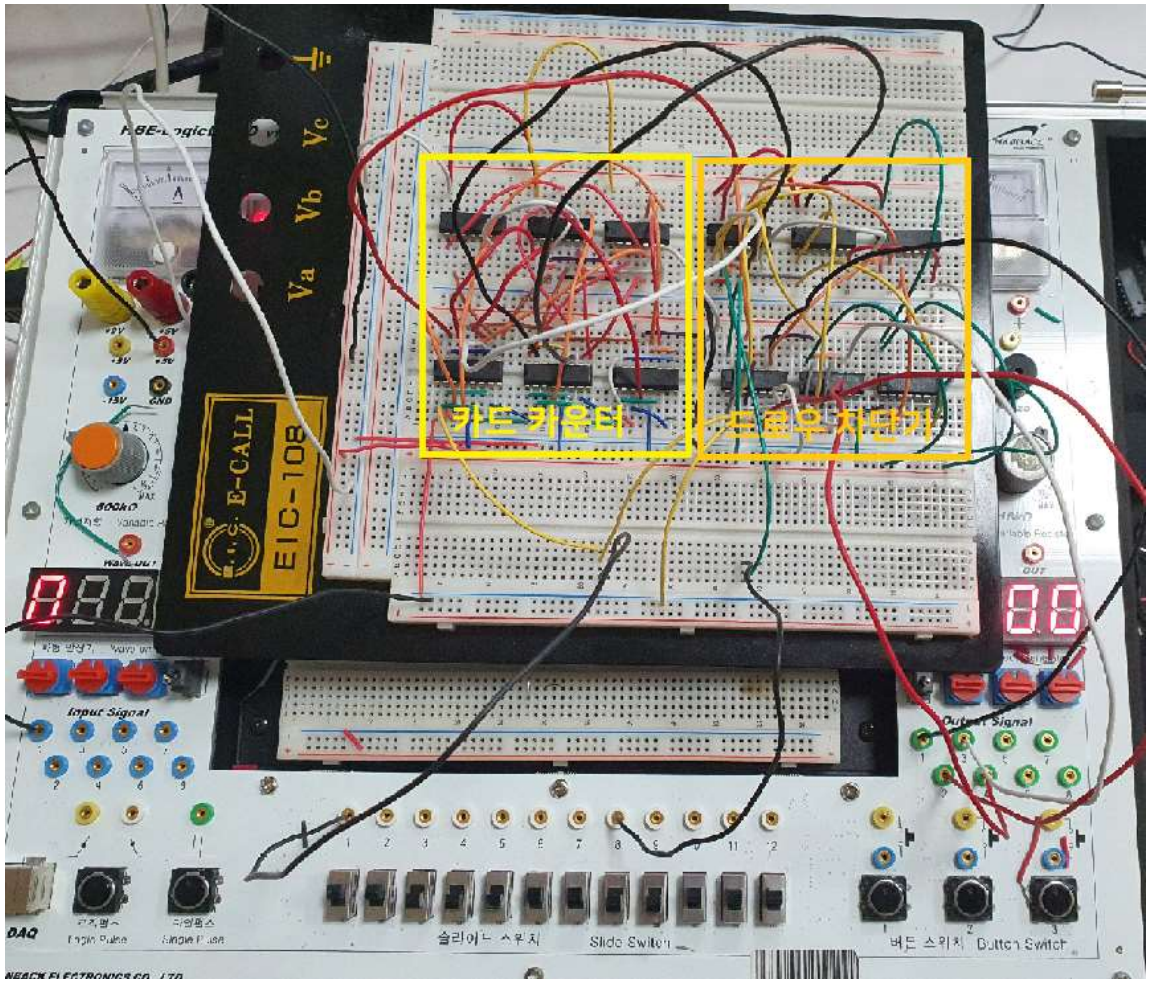
또한 스위치를 누른 직후 다음 CLOCK의 **상승에지에서 스위치가 자동으로 OFF**되는 것을 알 수 있다.

이 출력결과는 가산기의 입력으로 들어가서 합산 후 디코더로 들어가게 된다.

2. 회로 설계 과정

카운터+차단기 회로





2. 회로 설계 과정

기억성 가산기(1)

Full-Adder 소자를 사용하면 해저드가 발생하여 **J-K 플립플롭을 활용한 기억성 가산기**를 제작하였다.

Carry의 존재 유무에 따른 가산기의 동작특성을 설정하고
T플립플롭의 **토글스위치를 활용**해서 이전의 상태에 대한 반전으로 기억성을 구상했다.

마지막으로 원하는 진리표를 작성 후 카르노맵을 활용하여 논리회로식을 도출해냈다.

입력 1	입력 2	캐리	상태 변화	변화 원리		특이사항
0	0	0	0 에서 0	리셋	유지	
0	1	0	0 에서 1	셋팅	반전	
1	0	0	1 에서 1	셋팅	유지	
1	1	0	1 에서 0	리셋	반전	캐리 1

캐리	2	1	반전	캐리
C	B	A	T	C'
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	0	1
1	1	1	0	1

입력 1	입력 2	캐리	상태 변화	변화 원리		특이사항
0	0	1	0 에서 1	셋팅	반전	
0	1	1	0 에서 0	리셋	유지	캐리 1
1	0	1	1 에서 0	리셋	반전	캐리 1
1	1	1	1 에서 1	셋팅	유지	캐리 1

A₩BC	00	01	11	10
0	0	1	0	1
1	0	1	0	1

T

A₩BC	00	01	11	10
0	0	0	1	0
1	0	1	1	1

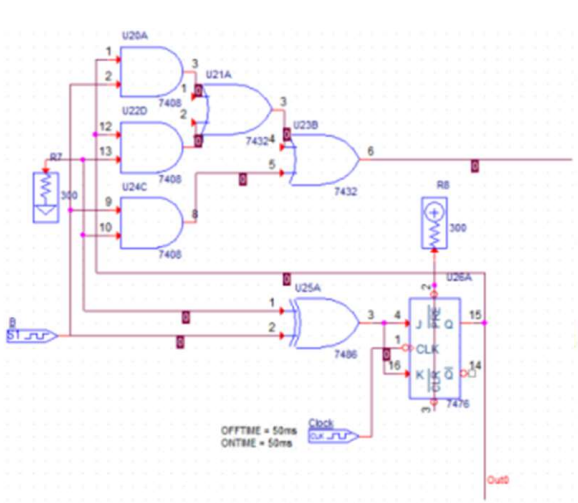
C

T = B'C + BC'

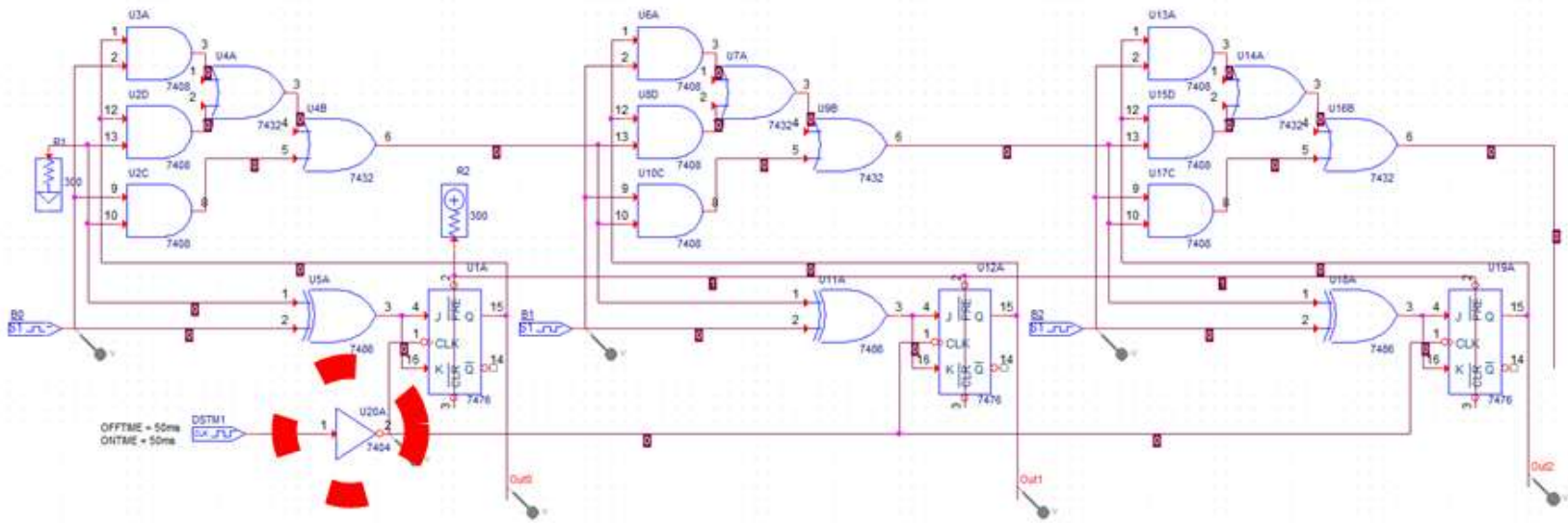
C = AB + AC + BC

2. 회로 설계 과정

기억성 가산기(2)



1Bit 가산기



4Bit 가산기

좌측의 1Bit 가산기를 총 4개 연결하여 만들어야 하지만, 무료 버전의 피스파이스는 **소자 수에 제한이 있어서** 4Bit 가산기로 테스트를 해보았다.

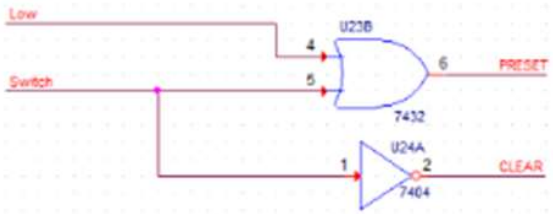
2. 회로 설계 과정

비교기(1)

기본적으로 4Bit 비교기가 제공되지만 우리는 21과 숫자를 비교해야 한다.
21을 2비트로 나타내면 "10101" 이다.
22는 2비트로 "10110"이다.
그래서 이것을 모두 비교하기 위해서는
총 5Bit를 비교해야 한다.
하지만, 22를 기준으로 비교를 한다면
1의자리 수의 비트는 상관이 없다.
5비트 중 맨 앞의 **4자리가 1011을 넘는다면, 그 수는 모두 21크기 때문**이다.

또한, 매 게임이 끝났을 때 뽑았던 카드를 리셋해주기 위한 스위치를 함께 설계하였다.

H	S	PRESET	CLEAR	상태	결과
0	0	1	1	(합 < 22) (스위치 OFF)	P/C OFF
0	1	1	0	(합 < 22) (스위치 ON)	Clear - 초기화
1	0	0	1	(합 > 21) (스위치 OFF)	BUST 출력
1	1	1	0	(합 > 21) (스위치 ON)	Clear - 초기화



P = (HS')' = H'+S

C = S'

I. 스위치가 OFF, High가 0 : Preset과 Clear가 모두 1이 되어 현재 상태를 유지한다.

II. 스위치가 ON, High가 0 : Clear가 0이 되어 초기화를 시킨다.

III. 스위치가 OFF, High가 1 : 합이 21보다 크기 때문에 BUST를 출력해야 한다.

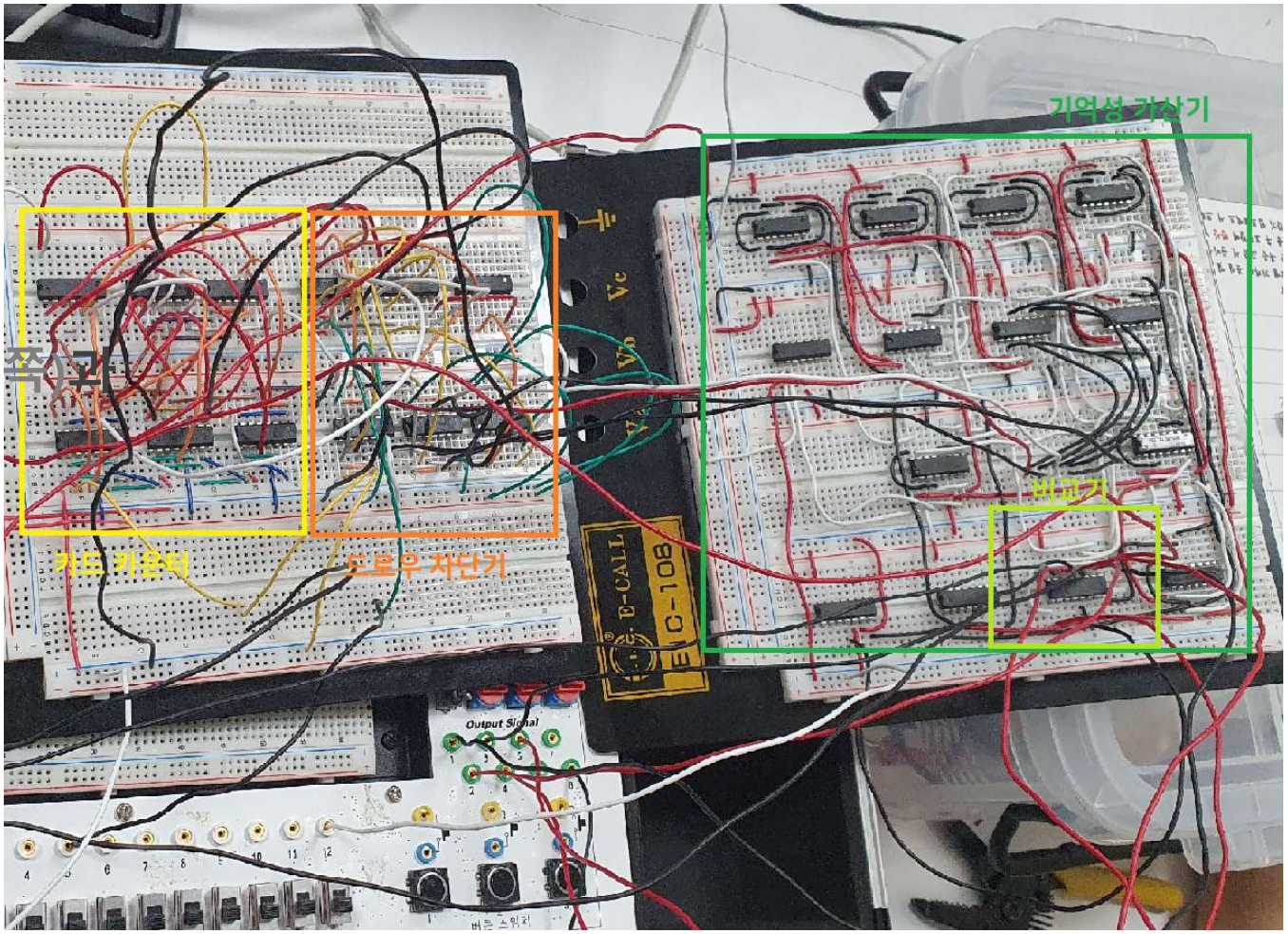
IV. 스위치가 ON, High가 1 : 게임이 끝난 후 초기화를 시켜야 하므로 Clear가 0이 된다.

2. 회로 설계 과정

가산기+비교기

오른쪽 그림과 같이
앞에서 소개했던 카운터+차단기 (왼쪽)과

가산기+비교기 (오른쪽)을 연결하여
테스트를 했다

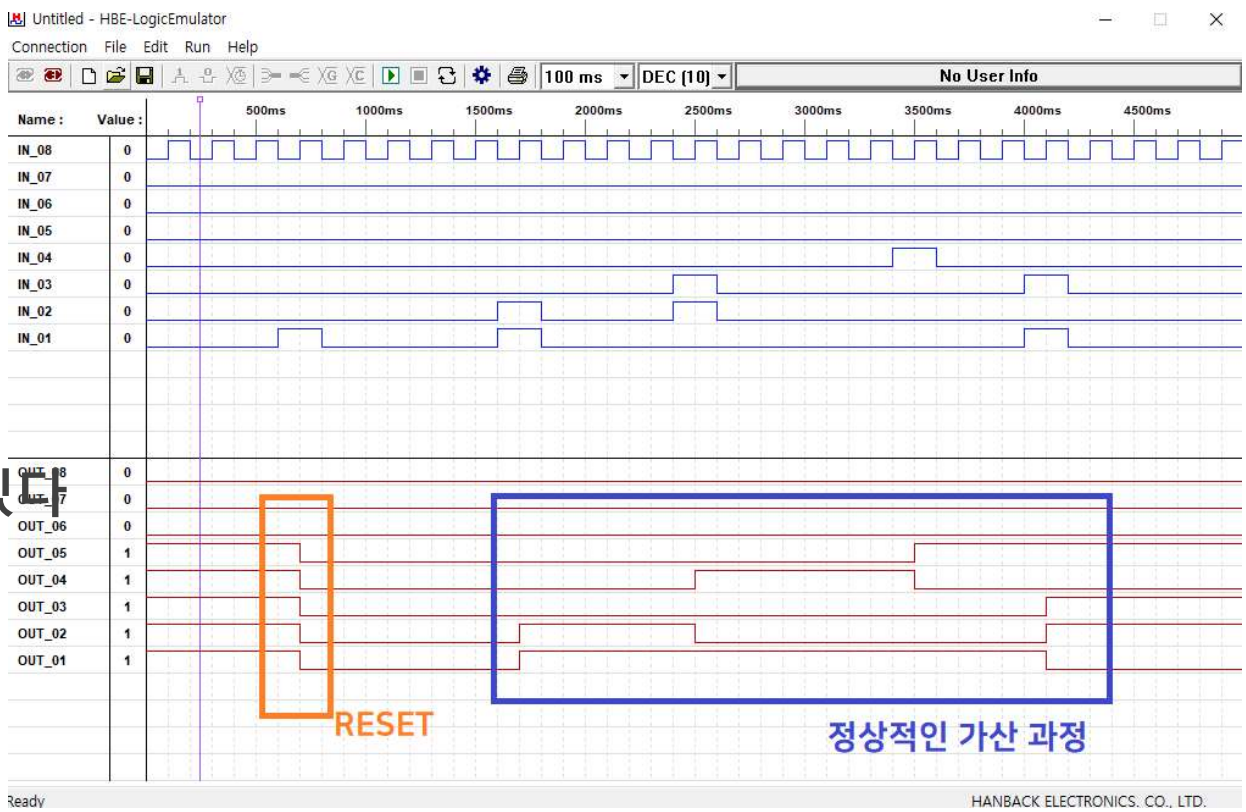


기억성 가산기(3)

설계했던 의도대로

스위치를 작동할 때 마다

임의의 숫자가 가산되는 것을 알 수 있다



2. 회로 설계 과정

디코더(1)

가산기에서 출력되는 2진수 결과를 그대로 연결할 수 있는 LED-Display가 실험실에 존재하지 않음

그래서 저희는 7-Segment LED 2개를 사용하여 십의 자리와 일의 자리 LED를

각각 컨트롤 하는 방식을 채택.

일의 자리 수는 0부터 9까지 수를 반복

십의 자리 수는

0~9의 수에서는 0

10~19의 수에서는 1

20~21는 2가 출력되도록 설정했다.

또한, Bust의 경우에는 별개로 **1011로 고정**하여 출력하였다.

						십의 자리				일의 자리			
수	A	B	C	D	E	Z	Y	X	W	P	O	N	M
0	0	0	0	0	0			0	0	0	0	0	0
1	0	0	0	0	1			0	0	0	0	0	1
2	0	0	0	1	0			0	0	0	0	1	0
3	0	0	0	1	1			0	0	0	0	1	1
4	0	0	1	0	0			0	0	0	1	0	0
5	0	0	1	0	1			0	0	0	1	0	1
6	0	0	1	1	0			0	0	0	1	1	0
7	0	0	1	1	1			0	0	0	1	1	1
8	0	1	0	0	0			0	0	1	0	0	0
9	0	1	0	0	1			0	0	1	0	0	1
10	0	1	0	1	0			0	1	0	0	0	0
11	0	1	0	1	1			0	1	0	0	0	1
12	0	1	1	0	0			0	1	0	0	1	0
13	0	1	1	0	1			0	1	0	0	1	1
14	0	1	1	1	0			0	1	0	1	0	0
15	0	1	1	1	1			0	1	0	1	0	1
16	1	0	0	0	0			0	1	0	1	1	0
17	1	0	0	0	1			0	1	0	1	1	1
18	1	0	0	1	0			0	1	1	0	0	0
19	1	0	0	1	1			0	1	1	0	0	1
20	1	0	1	0	0			1	0	0	0	0	0
21	1	0	1	0	1			1	0	0	0	0	1
BUST	1	1	1	1	1	1	0	1	1	1	0	1	1

십의 자리수 0

십의 자리수 1

십의 자리수 2

모두 1011로 고정하여
b가 출력되도록 설정

십의 자리 수 :Z = AB

Y = 0

X = AC

W = BD + BC + AC'

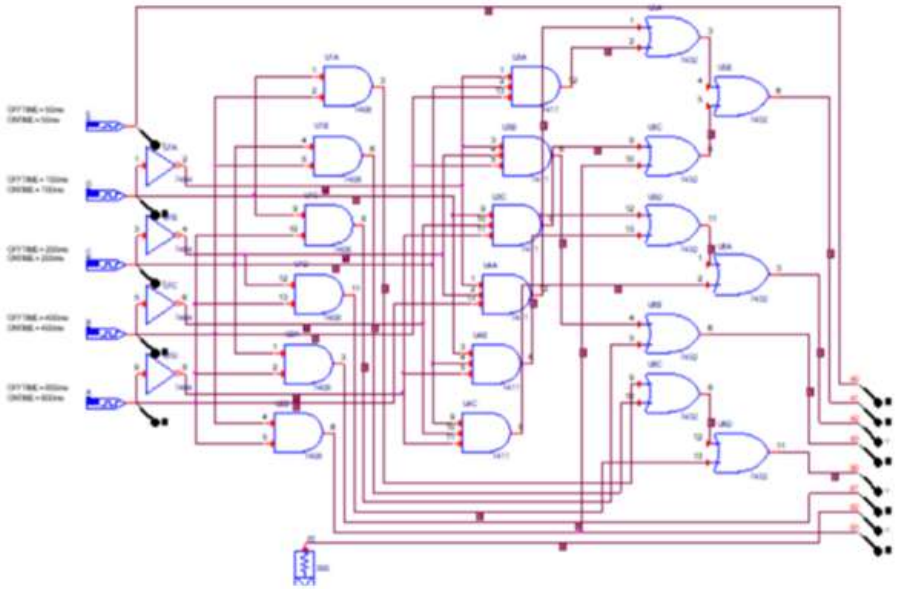
일의 자리 수 :P = AD + BC'D'

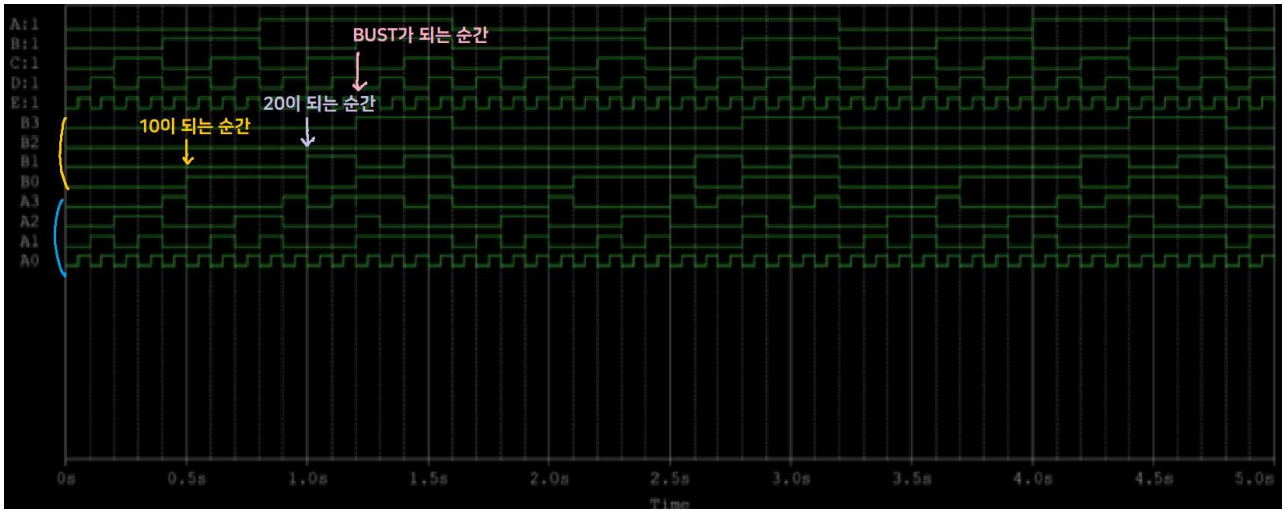
O = A'B'C + A'CD + AC'D'

N = AB + A'B'D + BCD' + AC'D

M = E

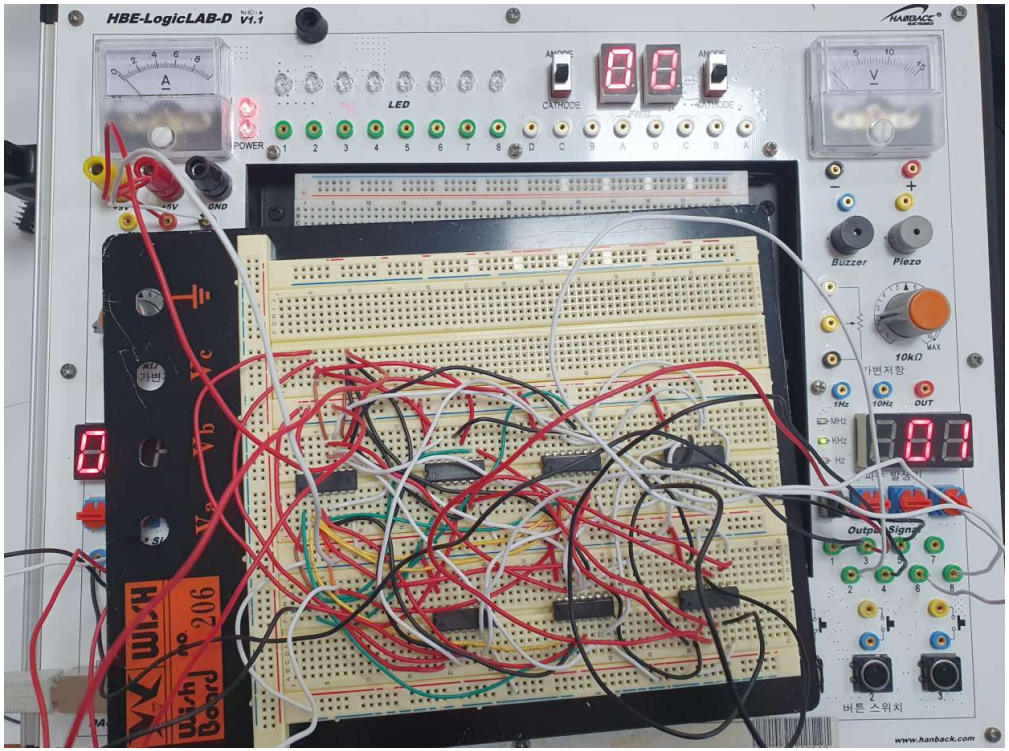
디코더(2)

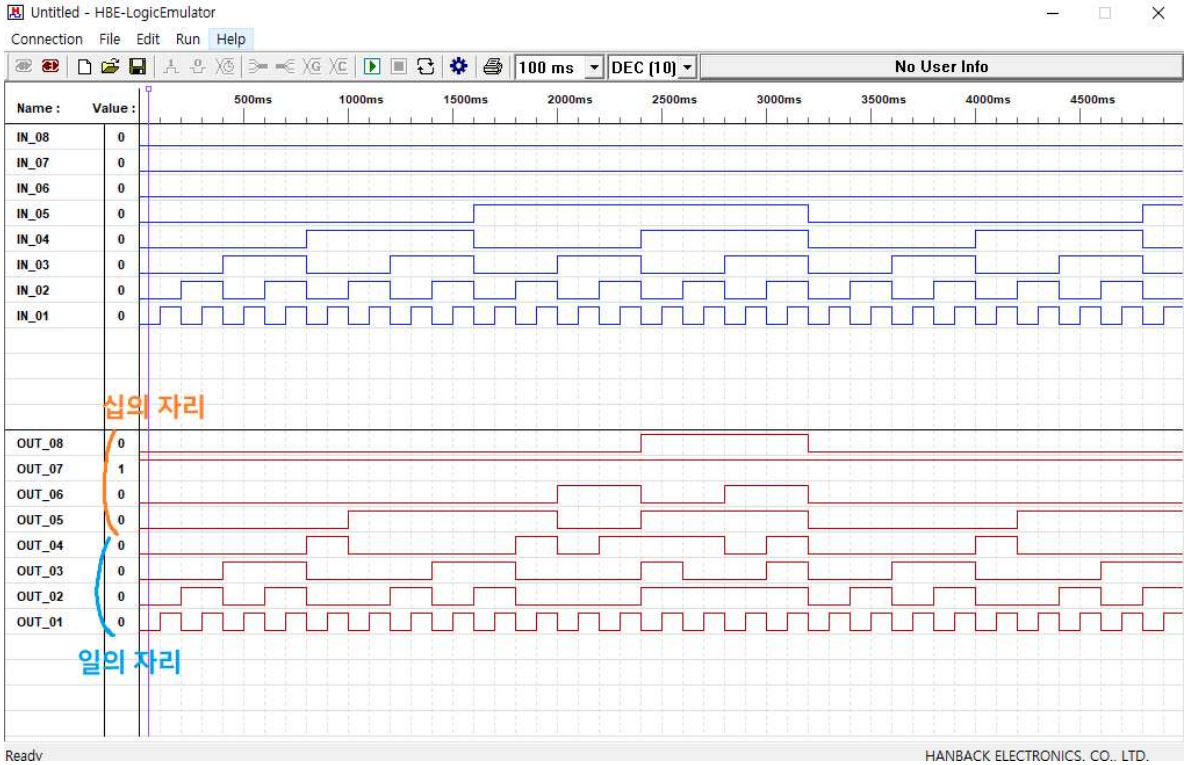



BUST가 되는 순간
20이 되는 순간
10이 되는 순간
A:1
B:1
C:1
D:1
E:1
B3
B2
B1
B0
A3
A2
A1
A0
0s
0.5s
1.0s
1.5s
2.0s
2.5s
3.0s
3.5s
4.0s
4.5s
5.0s
Time

2. 회로 설계 과정

디코더(3)

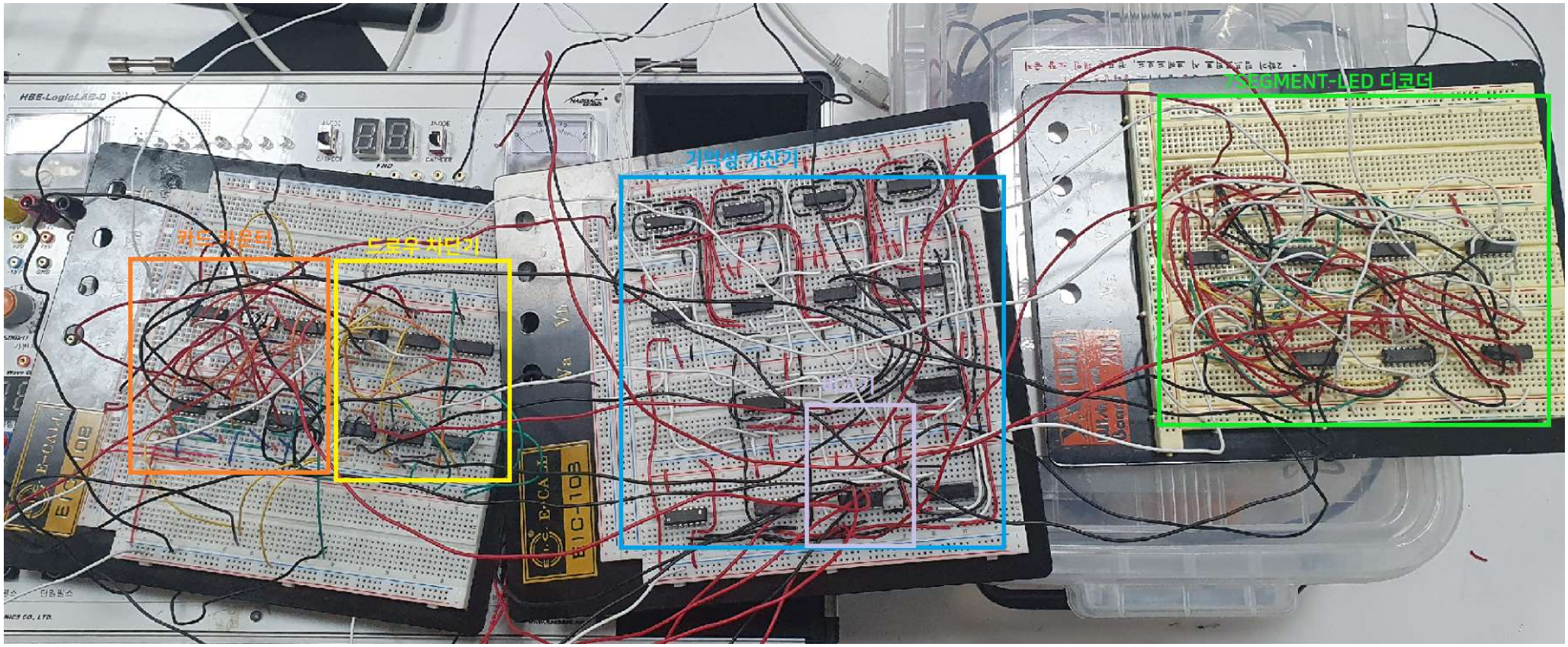




3. 실제 동작 과정

3. 실제 동작 과정

최종 회로 설계



3. 실제 동작 과정

실제 동작

https://youtu.be/WSXG_dV9ZRQ

영상을 첨부하니 용량이 너무 커서 업로드가 되지 않는 관계로 유튜브 링크를 첨부하

기대효과

1.카운터, 드로우 차단기, 기억성 가산기, 비교기, 디코더 각각의 회로는 다른 제품에 활용도가 높아 다양하게 응용될 수 있음

2. 특히 기억성 가산기는 기존의 가산기와는 차별화 된것으로 반복되는 입력들을 빠르게 합산할 때 유의미하게 활용될 것으로 판단됨

Q&A

Thanks

NEW RUN

Thanks