LDLAB 2

수리과학부 2019-16022 박채연

LAB and Discussion

1. Discussion

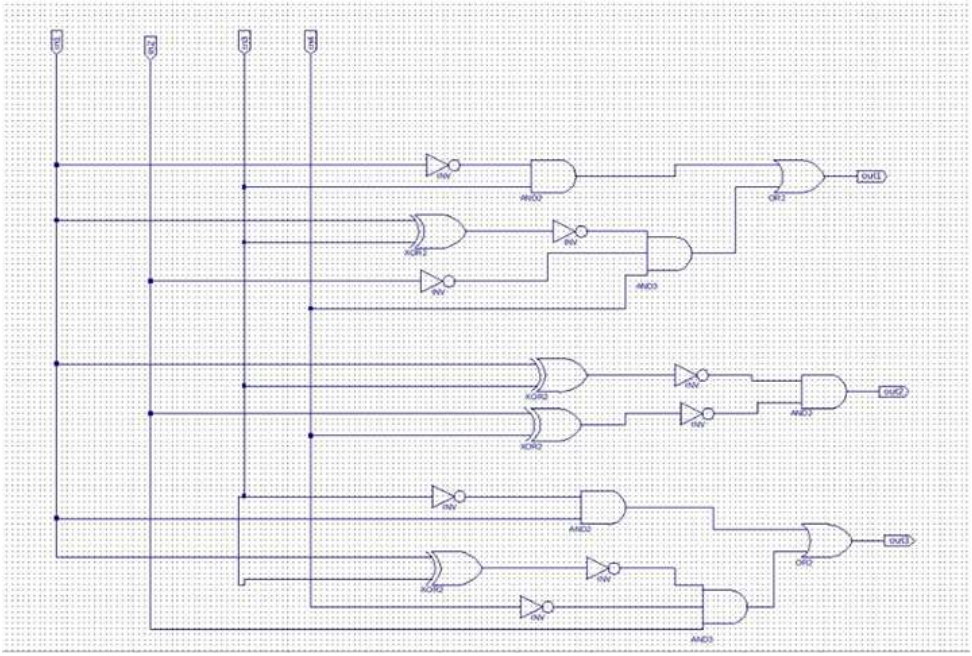
(1) Concepts I learned during the lab session

실제로 회로에 납땜을 하기 전에 Xilinx를 이용하여 구현하고자 하는 회로가 정상적으로 작동하는지 확인하는 과정을 거치면 시행착오를 줄일 수 있음을 배웠다. 또한, Xilinx 조작 방법과 Schematic Simulatoin Code를 작성하는 방법을 배웠다.

Xilinx에서 정상 작동함을 확인한 GT로직을 Universal board에 납땜하여 구현하는 과정에서 Xilinx와 Breadboard의 필요성을 배웠다. Universal board의 경우 Breadboard와 다르게 직접 납땜을 통해 전선과 소자를 연결해야하는데, 이는 오랜 시간이 걸리고 중간에 수정하기가 어렵다. 그러나 Breadboard의 경우에는 납땜이 필요 없고 게이트 등을 꽂기만 하면 되기 때문에 수정하기가 쉽다는 장점이 있으며, Xilinx의 경우에는 컴퓨터만 있으면 구현하고자 하는 회로가 정상적으로 작동하는지 여부를 쉽게 확인할 수 있다는 장점이 있다는 것을 배웠다.

(2) Any errors I made

처음 LT, EQ, GT에 대한 Boolean 식을 구현할 때 minimization을 진행하지 않았고, 이를 그대로 Xilinx로 구현하였다. 아래와 같이 XOR, NOT, AND, OR 게이트를 사용하였는데, 다른 조들은 그보다 적은 종류의 게이트를 이용하여 구현했다는 것을 듣고 minimization를 먼저 진행해야 한다는 것을 깨달았다. 아래는 minimize하지 않은 LT, EQ, GT 로직을 Xilinx로 구현한 이미지이다.



또한, 팀 과제를 수행하면서 초반에 Universal board가 어떻게 동작하는지 잘 이해하지 못해서 어려움을 겪었다. Breadboard의 경우 양 끝은 세로로 연결되어있고 중앙선 양 옆은 가로로 연결되어있었는데, Universal board는 외관이 breadboard와 매우 달랐다. Univresal board에서 VCC와 GND를 제외한 나머지 부분은 어떤 방향으로 연결되어있는지 몰라 어려움을 겪었다.

처음에 한 팀원이 납땜 편의 상 전선은 길게 자르고 피복은 최대한 많이 벗겨서 사용하는게 좋을 것 같다고 하였다. 전선을 두세개 정도 납땜해보니 전선이 길면 전선이 차지하는 공간이 많아지고, 피복을 많이 벗기면 다른 전선과 부딪혔을 때 전류가 통해 원하는 결과를 얻지 못할 가능성이 있다는 것을 알게 되었다.

(3) How to correct my errors

K-MAP 방법과 Boolean algebra를 이용하여 LT, EQ, GT 로직을 minimize하였다. 그 결과 처음에 Xilinx로 구현했을 때 사용했던 4개 종류의 게이트보다 게이트를 1 종류 덜 사용하도록 GT를 구현할 수 있었다.

또한, 다른 팀원의 설명으로 Universal board는 Breadboard와 다르게 VCC 와 GND를 제외한 나머지 부분은 연결되어있지 않아서 직접 납땜으로 전선을 연결해야 함을 알게되었다. 따라서 VCC와 GND가 연결된 선을 제외하고는 임의로 논리게이트를 위치시킬 수 있음을 알게 되었고, 논리게이트의 Datasheet를 참고하여 납땜을 통해 전선을 이어붙이며 GT 로직을 구현할 수 있었다.

마지막으로, 길이가 길고 피복이 많이 벗겨진 전선 때문에 발생할 수 있는 문제를 예방하기 위해 전선의 길이를 처음보다 짧게 자르고 피복은 최대한 덜 벗겨서 납땜하였다.

2. Result

Implement 2-bit comparator LT, EQ, GT using Xilinx ISE schematic. Simulate its behavior using Xilinx test bench.

우선, K-MAP 방법을 사용하여 LT, EQ, GT 로직을 minimize해야 한다.

LT

AB₩CD	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

따라서 LT = A'C+A'B'D+B'CD = A'C+B'D(A'+C)이다.

EQ

AB₩CD	00	01	11	10

00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

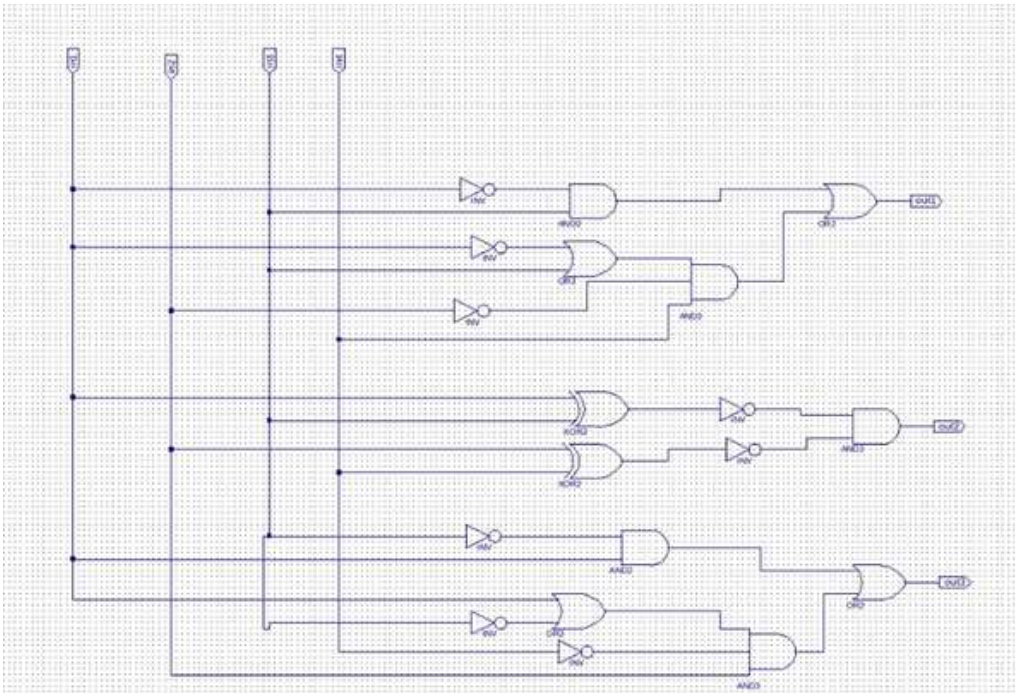
EQ = A'B'C'D'+A'BC'D+ABCD+AB'CD' = A'C'(B'D'+BD)+AC(BD+B'D') = (A'C'+AC)(B'D'+BD) = (A xor C)'(B xor D)'이다.

GT

AB₩CD	**00**	**01**	**11**	**10**
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

따라서 GT = AC'+BC'D'+ABD' = AC'+BD'(C'+A)이다.

지금까지 K-MAP 방법과 Boolean algebra를 통해 minimize한 LT, EQ, GT 로직을 Xilinx를 통해 구현하면 아래와 같다.



또한, test 코드는 아래와 같다.

```
1  // Verilog test fixture created from schemati
2
3  `timescale 1ns / 1ps
4
5  module comparator_comparator_sch_tb();
6
7  // Inputs
8      reg in1;
9      reg in2;
10     reg in3;
11     reg in4;
12
13 // Output
14     wire out1;
15     wire out2;
16
17 // Bidirs
18
19 // Instantiate the UUT
20    comparator UUT (
21       .in1(in1),
22       .in2(in2),
23       .in3(in3),
24       .in4(in4),
25       .out1(out1),
26       .out2(out2),
27       .out3(out3)
28    );
29 // Initialize Inputs
30
31     initial begin
32         in1 = 0;
33         in2 = 0;
34         in3 = 0;
35         in4 = 0;
```

```
38         in1 = 0;
39         in2 = 0;
40         in3 = 0;
41         in4 = 1;
42         #50;
43
44         in1 = 0;
45         in2 = 0;
46         in3 = 1;
47         in4 = 0;
48         #50;
```

```
50         in1 = 0;
51         in2 = 0;
52         in3 = 1;
53         in4 = 1;
54         #50;
55
56         in1 = 0;
57         in2 = 1;
58         in3 = 0;
59         in4 = 0;
60         #50;
61
62         in1 = 0;
63         in2 = 1;
64         in3 = 0;
65         in4 = 1;
66         #50;
67
68         in1 = 0;
69         in2 = 1;
```

```
        in3 = 1;
        in4 = 0;
        #50;

        in1 = 0;
        in2 = 1;
        in3 = 1;
        in4 = 1;
        #50;

        in1 = 1;
        in2 = 0;
        in3 = 0;
        in4 = 0;
        #50;

        in1 = 1;
        in2 = 0;
        in3 = 0;
        in4 = 1;
        #50;

        in1 = 1;
        in2 = 0;
        in3 = 1;
        in4 = 0;
        #50;

        in1 = 1;
        in2 = 0;
        in3 = 1;
        in4 = 1;
        #50;
```

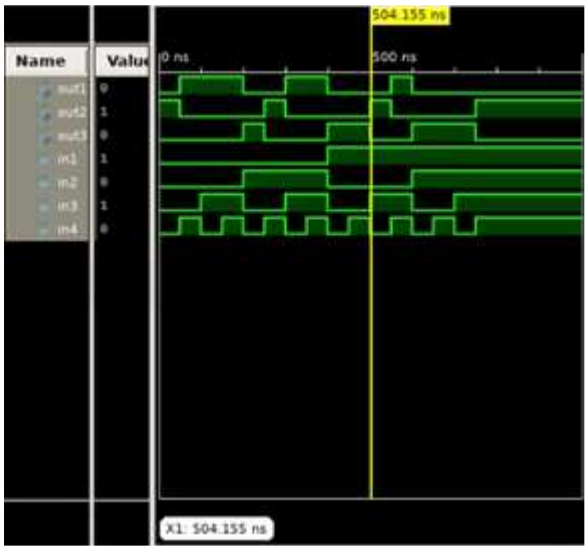
```
        in1 = 1;
        in2 = 1;
        in3 = 0;
        in4 = 0;
        #50;

        in1 = 1;
        in2 = 1;
        in3 = 0;
        in4 = 1;
        #50;

        in1 = 1;
        in2 = 1;
        in3 = 1;
        in4 = 0;
        #50;

        in1 = 1;
        in2 = 1;
        in3 = 1;
        in4 = 1;
        #50;
    end
endmodule
```

즉, in1, in2, in3, in4를 아래와 같은 순서로 입력하였다.

순서	In1	In2	In3	In4
1	0	0	0	0
2	0	0	0	1
3	0	0	1	0
4	0	0	1	1
5	0	1	0	0
6	0	1	0	1
7	0	1	1	0
8	0	1	1	1

9	1	0	0	0
10	1	0	0	1
11	1	0	1	0
12	1	0	1	1
13	1	1	0	0
14	1	1	0	1
15	1	1	1	0
16	1	1	1	1

이에 대한 시뮬레이션 결과는 아래와 같다.



위에서 제시한 입력 순서에 따른 LT, EQ, GT 결과를 표로 정리하면 아래와 같은데, 정확하게 시뮬레이션 결과와 일치하는 것을 확인할 수 있다.

순서	LT	EQ	GT
1	0	1	0
2	1	0	0
3	1	0	0
4	1	0	0
5	0	0	1
6	0	1	0
7	1	0	0
8	1	0	0
9	0	0	1
10	0	0	1

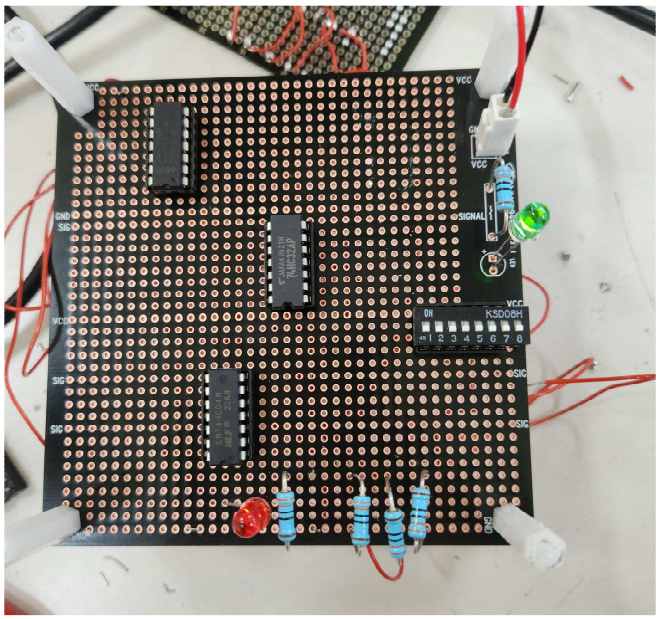
11	0	1	0
12	1	0	0
13	0	0	1
14	0	0	1
15	0	0	1
16	0	1	0

Homework

1. implement 2-bit comparator GT logic on the universal board using DIP switch and primitive logic gates(INV, AND, OR, XOR, NAND…). Try your best to minimize the number of logic gates.

앞에서 제시한 minimized GT logic인 Y= AC'+BD'(A+C')을 사용하여 구현하였다. 이때, 사용하는 논리게이트의 종류를 최소화해야 하므로 BD'(A+C')를 3-input AND 게이트로 구현하지 않고, 중첩된 2-input AND 게이트로 구현하였다.

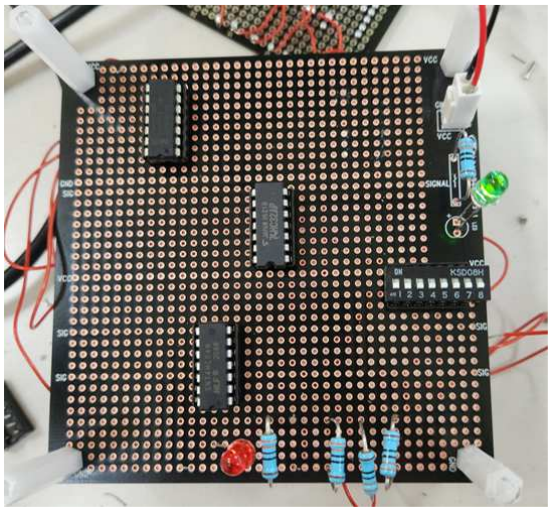
NOT 게이트, AND 게이트, OR 게이트를 이용하여 아래와 같이 구현하였다. 사진에서 잘 보이지 않지만 전원에 연결했을 때 초록색 발광 다이오드의 불이 켜진 상태이다.



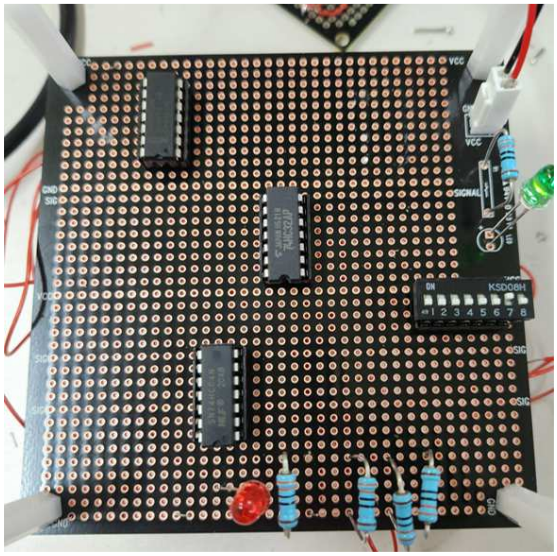
4개짜리 스위치 대신 8개짜리 스위치를 사용하였는데, 1번 스위치가 in1, 3번 스위치가 in2, 5번 스위치가 in3, 7번 스위치가 in4 를 의미한다. 또한, 발광 다이오드에 직접적으로 5V의 전압을 걸어주면 깨질 수 있어서 저항을 연결해주어야 하는데, 빨간색 다이오드에 대한 저항은 납땜 편의상 뒤쪽에서 연결하였다.

결과를 보이면 아래와 같다. (편의상 'in1=0, in2=0, in3=0, in4=0인 경우'를 '0000인 경우'와 같이 표기하였다.)

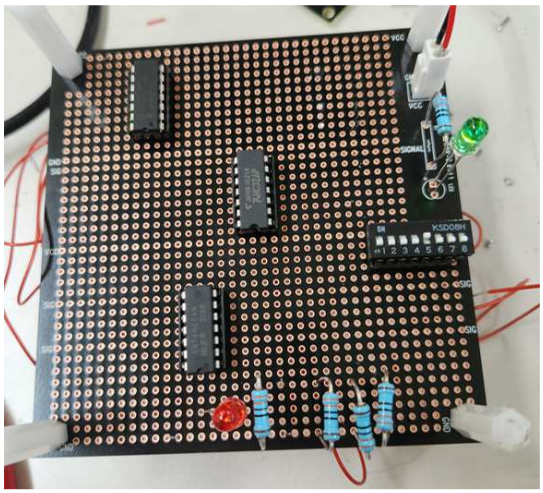
(1) 0000인 경우



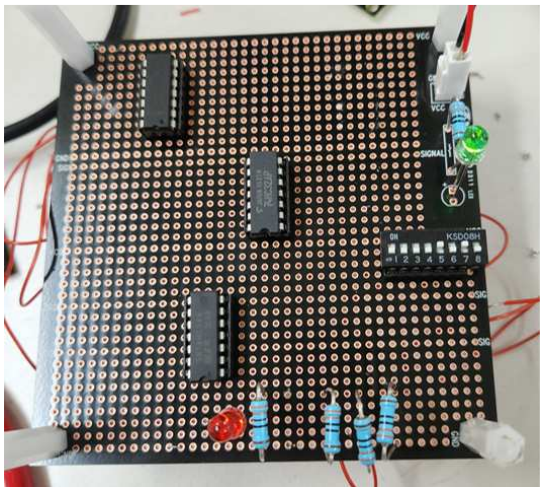
(2) 0001인 경우



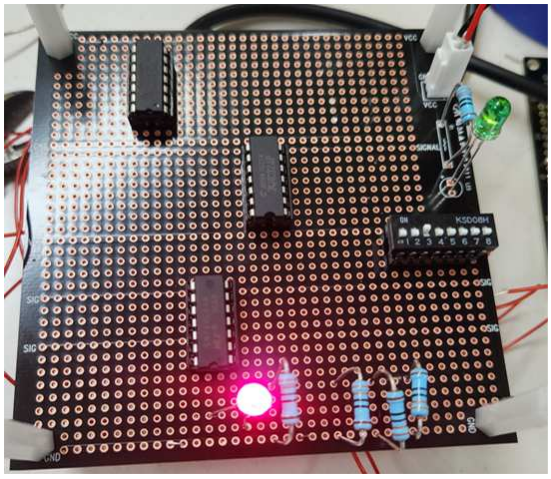
(3) 0010인 경우



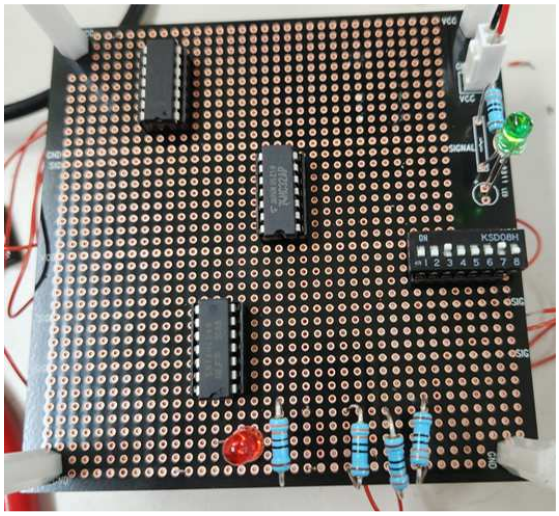
(4) 0011인 경우



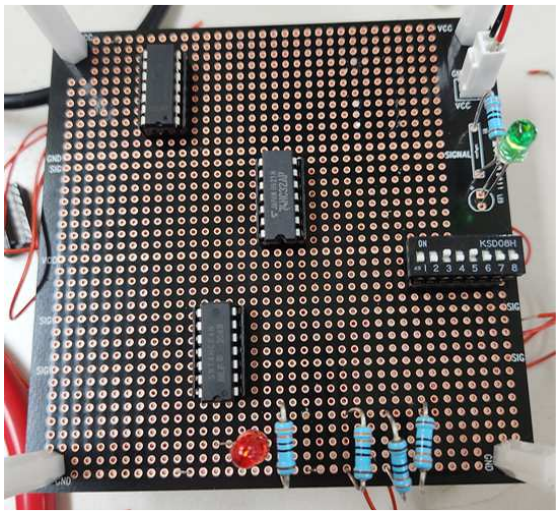
(5) 0100인 경우



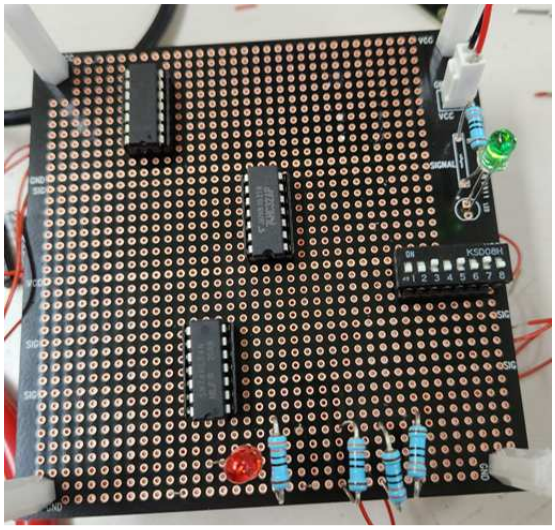
(6) 0101인 경우



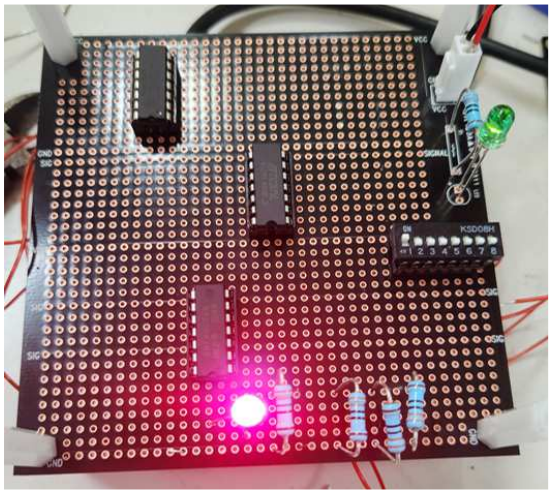
(7) 0110인 경우



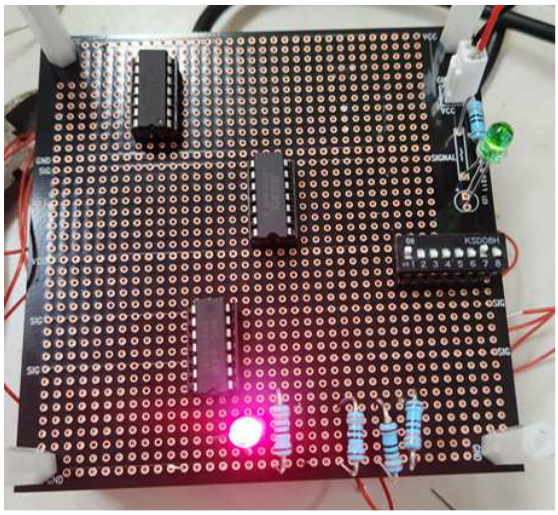
(8) 0111인 경우



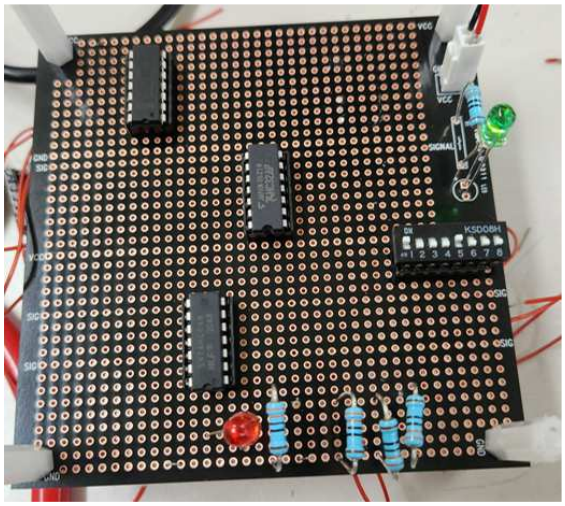
(9) 1000인 경우



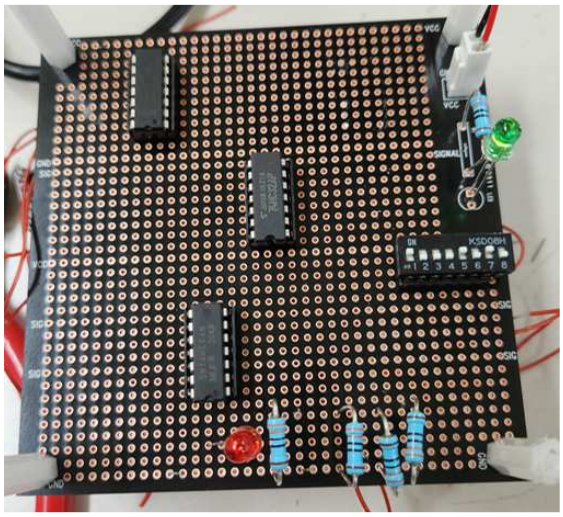
(10) 1001인 경우



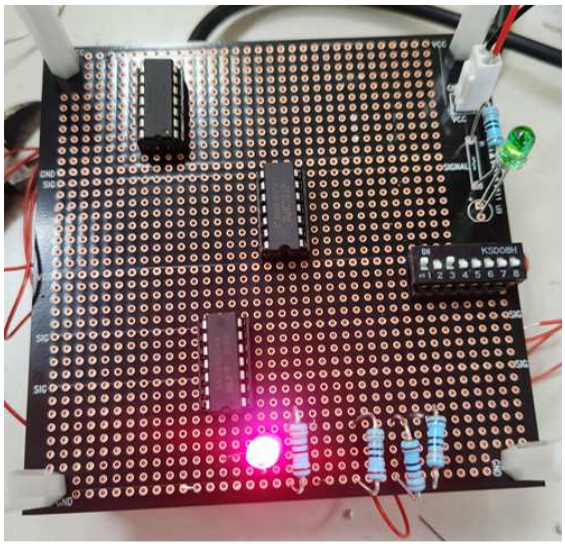
(11) 1010인 경우



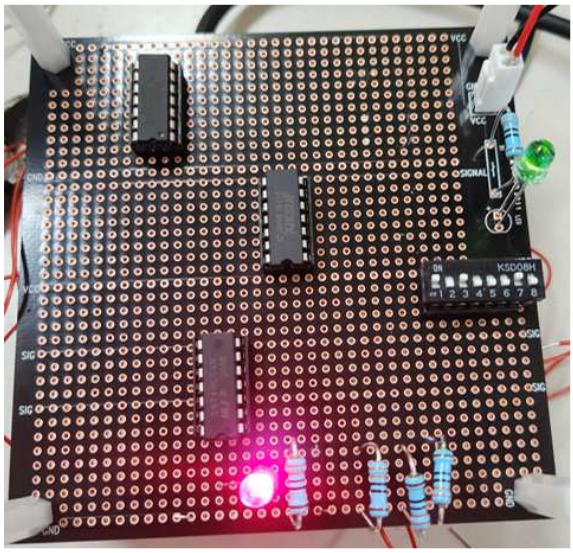
(12) 1011인 경우



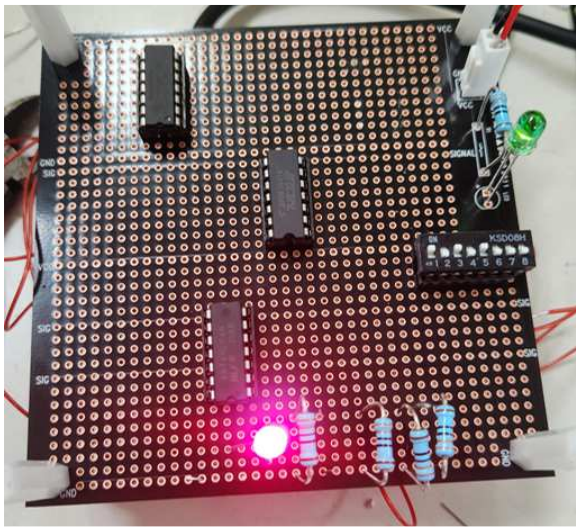
(13) 1100인 경우



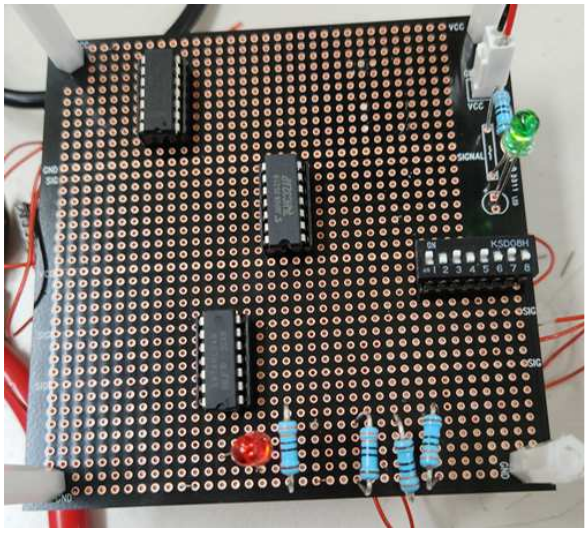
(14) 1101인 경우



(15) 1110인 경우



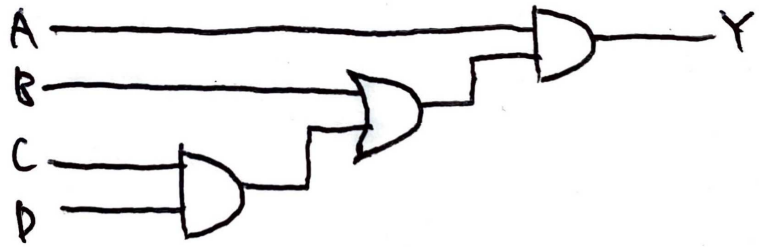
(16) 1111인 경우



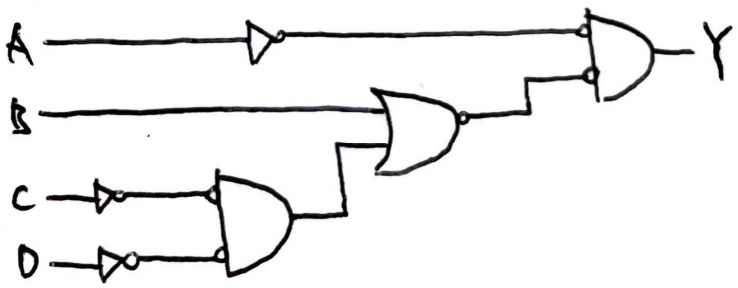
2. Draw a circuit diagram for the below formula using only NOR and NOT gates. You can draw it by hand or using any of a computer program.

Y=A(B+CD)

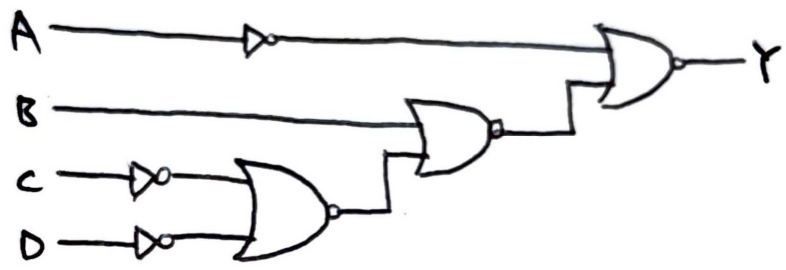
우선 Y=A(B+CD)의 circuit diagram을 NOR 과 NOT 게이트 만을 사용하라는 조건을 고려하지 않고 그려보면 아래와 같다.



이때, AND와 OR 게이트를 NOR 게이트로 바꾸기 위해 NOT 게이트와 버블을 추가하면 아래와 같다.



따라서, 최종적으로 아래와 같이 NOR과 NOT 게이트만을 이용하여 표현 가능하다.



3.

(1) Make a truth table

Y=AB+ABC+A'B+AB'C에 대한 진리표는 아래와 같다.

A	B	C	AB	ABC	A'B	AB'C	Y
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	1	0	1
0	1	1	0	0	1	0	1
1	0	0	0	0	0	0	0
1	0	1	0	0	0	1	1
1	1	0	1	0	0	0	1
1	1	1	1	1	0	0	1

(2) Minimize # of operators

연산자를 최소화하기 위한 계산은 아래와 같다.

Y = AB + ABC + A'B + AB'C

= AB(1+C) + A'B + AB'C

= AB + A'B + AB'C

= (A+A')B + AB'C

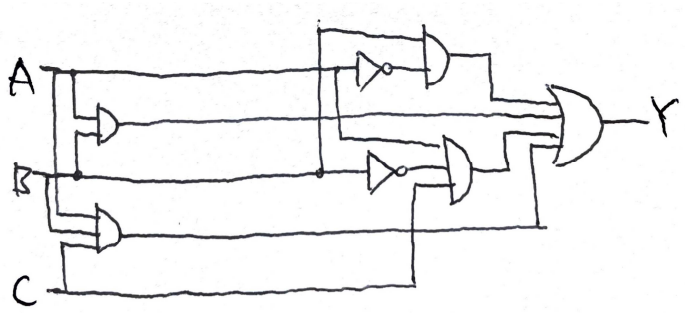
= B + AB'C

= B + AC

따라서 Y=AB+ABC+A'B+AB'C의 연산자를 최소화하면 Y=B+AC임을 알 수 있다.

(3) Draw a circuit diagram

Y=AB+ABC+A'B+AB'C의 circuit diagram을 그리면 아래와 같다.



한편, Y의 연산자를 최소화하면 Y=B+AC임을 앞에서 보였는데, 이를 circuit diagram으로 나타내면 아래와 같다.