실험물리학 2

6주차 예비 레포트

<디지털 논리회로의 기초>

이름: 김나현

학번: 20191286

분반: 2분반

담당 교수님: 정명화 교수님

담당 조교님: 소현경 조교님

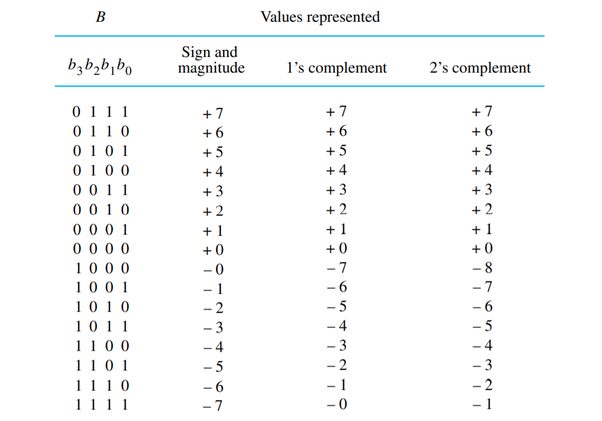
제출일자: 2020년 10월 21일 수요일

1. 실험 목표
2. 반도체 소자를 사용하여 기본 논리 회로인 NOT, AND, OR 회로를 구성한다.
3. 기본 논리 게이트 및 추가 논리 게이트에 대해 이해할 수 있다.
4. 논리 회로의 설계와 해석에 있어서 중요한 부울 대수, 드 모르간의 정리, 카르노 맵에 대해 이해할 수 있다.
5. 실험 이론
6. (2진수) 1의 보수, 2의 보수

+와 -의 기호를 이용하여 양수와 음수를 표현하는 십진수와 달리 이진수에서는 Sign-Magnitude representation이라는 방법 또는 2’s complement(2의 보수)라는 방법을 사용하여 양수와 음수를 구분한다.

우선, Sign-Magnitude representation 방법에 대해서 설명해보면 이 방법으로 이진수를 표현하게 되면 수(크기, magnitude)를 표현하는 부분과 부호(sign)을 나타내는 두 부분으로 나눌 수 있다. 이 방법을 통해서 숫자의 부호를 표현하기 위해서는 단순히 숫자 앞에 양을 의미하는 0 bit를 추가하거나 음을 의미하는 1 bit를 추가하면 된다. 예를 들어, 01010101(2)과 11010101(2)이라는 두 개의 이진수는 공통적으로 1010101으로부터 85라는 크기를 갖는다는 사실을 알 수 있지만 01010101은 앞의 sign bit가 0이므로 양수, 11010101은 앞의 sign bit가 1이므로 음수라는 차이가 있다. 따라서 컴퓨터가 이러한 숫자들을 더하거나 뺄 때 sign과 magnitude를 모두 고려해야 하므로 느리다는 단점이 있다. 또한 Sign-magnitude representation 방법으로는 n자리 이진수의 경우, -(2n-1-1)부터 +(2n-1-1)까지의 숫자를 표현할 수 있으므로 아래 <그림 a>를 보면 n이 4인 4 bit 이진수에 대해 이와 같은 방법으로는 -7부터 +7까지의 숫자를 표현할 수 있다.

2’s complement 방법에 대해 설명하기 전에 보수(complement)라는 용어에 대해 알아보자. 1의 보수란 컴퓨터 내부에서 음의 수가 양의 수와 마찬가지로 계산되도록 표현하기 위해 고안된 것으로, 각 자리를 1에서 빼서 나타낸 것이다. 만약 1010이라는 네 자리 2진수가 있을 때, 이 수의 1의 보수는 단순히 1을 0으로, 0을 1로 전환한 0101이 된다. 1010과 이 수의 1의 보수인 0101을 더하면 1111이므로 0이 아니기 때문에 1의 보수는 원래 수의 음수로의 표현으로는 부족하지만 1의 보수에 1을 더한 수는 음의 수로 사용되기 용이하다. 1010의 1의 보수인 0101에 1을 더하면 0110이 되어 1010과 더했을 때 10000이 되므로 앞의 1이라는 carry를 무시하면 0000이 되므로 0이 된다. 이는 2의 보수가 원래의 수의 음수로의 표현에 적합하다는 것을 아주 잘 보여주는 예시이다. 이때, n자리 이진수는 2’s complement 방법으로 –(2n-1)부터 +(2n-1-1)까지의 숫자만을 표현할 수 있으므로 아래 <그림 a>를 보면 n이 4인 4 bit 이진수에 대해 2’s complement 방법을 사용하면 -8부터 +7까지의 숫자를 나타낼 수 있다.

<그림 a> 4 bit 이진수의 sign-magnitude와 1의 보수, 2의 보수

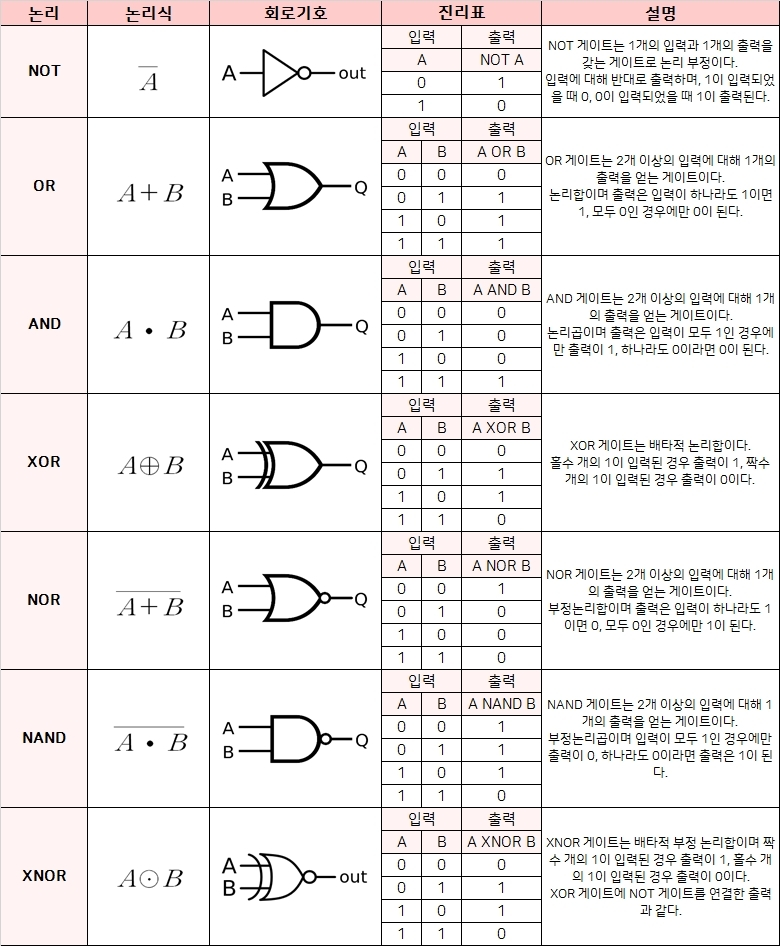
1. 논리 게이트

아래 <그림 b>에는 이후 우리가 다루게 될 논리 게이트들에 대한 정보가 있다. 우선, NOT 게이트는 <그림 b>에 나타난 것처럼 삼각형과 한 꼭짓점에 동그라미를 그린 모양으로 회로 기호를 표현하고 algebra식으로 A’나 의 형태로 표현한다. NOT 게이트는 한 개의 입력과 한 개의 출력을 갖는 게이트로, 입력에 대해 반대로 출력하기 때문에 0을 입력하면 1을 출력하고 1을 입력하면 0을 출력한다.

다음으로 OR 게이트와 AND 게이트에 대해 알아보자. OR 게이트와 AND 게이트 모두 NOT 게이트와는 달리 두 개 이상의 입력에 대해 1 개의 출력을 갖는 게이트라는 공통점이 있지만 OR 게이트는 두 개 이상의 입력 중 단 하나라도 1이면 1을 출력하지만 AND 게이트는 입력 모두가 1이 아닌 경우를 제외하면 0의 입력이 하나라도 있으면 0을 출력한다는 차이점이 있다. 입력이 A와 B라고 할 때, OR 게이트는 A+B라고 표현하며 AND 게이트는 A•B라고 표현하고 각 회로 기호는 아래 <그림 B>를 참조하면 된다.

OR 게이트에 NOT을 붙인 NOR 게이트에 대해 알아보면, NOR 게이트는 OR 게이트의 정반대 출력을 가지므로 입력이 모두 0일 때 1의 출력을 갖고 그 경우를 제외하면 모두 0의 출력을 갖는다. 이는 말그대로 OR 게이트의 회로 기호의 앞에 동그라미를 붙여서 표현하고 (A+B)’=A’B’라고 표현한다. 이와 마찬가지로 AND 게이트에도 NOT을 붙인 NAND 게이트가 있는데 이는 (AB)’=A’+B’라고 표현하며 AND 게이트 앞에 동그라미를 붙인 기호를 사용하고, 입력이 모두 1일 때만 0의 출력을 갖고, 입력이 하나라도 0이면 1의 출력을 갖는다.

마지막으로 이보다는 조금 심화된 XOR와 XNOR 게이트에 대해서 알아보자. XOR 게이트는 A와 B의 입력에 대해 A⊕B이라는 논리 식으로 표현되는데 이는 AB’+A’B라는 식으로 표현할 수 있다. 두 개의 입력이 서로 같으면 0을 출력하고 두 개의 입력이 서로 다르면 1을 출력한다. XNOR 게이트는 XOR 게이트에 NOT을 붙여서 만든 게이트이므로 XOR 게이트와 정반대의 출력 값을 갖기 때문에 두 입력이 서로 같으면 1을, 두 입력이 서로 다르면 0을 출력한다. 이때, XNOR 게이트에서 A와 B 두 개의 입력에 대해 출력은 A⊙B라는 기호로 표현할 수 있고 이는 다르게 AB+A’B’라는 식으로 표현 가능하다.



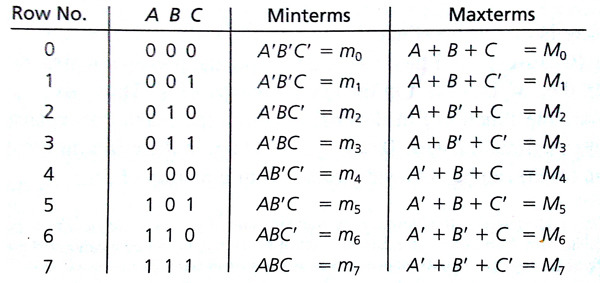
<그림 b> 논리 게이트

1. 부울 대수(Boolean algebra), 드 모르간의 법칙(DeMorgan’s theorem), 카르노 맵(Karnaugh map)

부울 대수란 영국의 수학자 G.Boole에 의해 창시된 논리수학으로, ‘A 또는 B이다’와 같은 인간의 표현을 기호를 통해 대수적으로 취급할 수 있도록 한 것이다. 이와 같은 Boolean algebra의 특성 중 두 개의 입력 a와 b가 있다고 할 때, a+b=b+a이고 ab=ba라는 것을 통해 OR 게이트와 AND 게이트에서 교환법칙이 성립하고 세 개의 입력 a, b, c가 있을 때도 a+(b+c)=(a+b)+c이고 a(bc)=(ab)c라는 것을 통해 결합법칙이 성립한다는 것을 알 수 있다. 또한 a와 a의 보수, a’에 대해서 0+a=a+0=a이고 1+a=a+1=1임을, a•0=0•a=0이고 a•1=1•a=a임을, a+a’=a’+a=1이고 aa’=a’a=0임을 알 수 있다. a’의 보수, (a’)’=a이고 a+a=a, a•a=a라는 특성 또한 알 수 있다. 이보다 조금 더 심화된 내용으로는 부울 대수에서 a(b+c)=ab+ac, a+bc=(a+b)(a+c)라는 분배법칙이 성립한다는 것과 ab+ab’=a, (a+b)(a+b’)=a라는 식이 성립함을 알 수 있다. 이를 앞에서 본 부울 대수의 특성을 이용해서 증명해보면 (a+b)(a+c)=a•a+ab+ac+bc=a+ab+ac+bc=a(1+b+c)+bc=a+bc이고, ab+ab’=a(b+b’)=a•1=a, (a+b)(a+b’)=aa+ab+ab’+bb’=a+ab+ab’=a(1+b+b’)=a이다. 부울 대수에서는 a+a’b=a+b, a(a’+b)=ab라는 simplification에 도움이 되는 특성도 있는데 a+a’b=(a+a’)(a+b)= aa+aa’+ab+a’b=a+b(a+a’)=a+b이고, a(a’+b)=aa’+ab=ab이다. absorption이라고 불리는 특성도 있는데 a+ab=a, a(a+b)=a라는 것이고, 이는 a+ab=a(1+b)=a이고 a(a+b)=aa+ab=a+ab=a(1+b)=a인 것이다. (a+c)(a’+b)=aa’+a’c+ab+bc=a’c+ab+bc=a’c+ab+(a+a’)bc=a’c+ab+abc+a’bc=ab(1+c)+a’c(1 +b)=ab+a’c이므로 ab+a’c=(a+c)(a’+b)이다. 이를 정리해 놓은 것이 아래 ①부터 ⑪이고 ⑫는 DeMorgan 법칙으로 (a+b)’=a’b’, 즉 NOR 게이트는 두 입력을 각각 NOT 게이트를 거쳐서 AND 게이트에 입력한 출력 결과와 같다는 것이고 (ab)’=a’+b’, 즉 NAND 게이트는 두 입력을 각각 NOT 게이트에 거쳐서 OR 게이트에 입력한 출력 결과와 같다는 법칙이다.

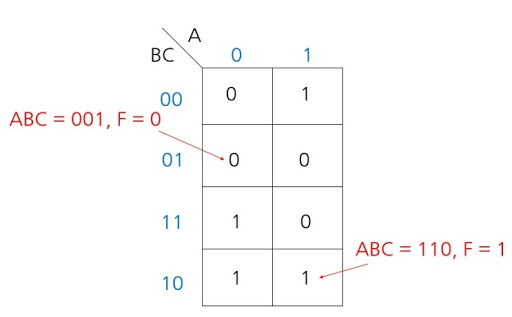
1. a+b=b+a, ab=ba
2. a+(b+c)=(a+b)+c, a(bc)=(ab)c
3. a+0=0+a=a, 1+a=a+1=1, a•0=0•a=0, a•1=1•a=a
4. a+a’=a’+a=1, aa’=a’a=0
5. a+a=a, a•a=a
6. (a’)’=a
7. a(b+c)=ab+ac, a+bc=(a+b)(a+c)
8. ab+ab’=a, (a+b)(a+b’)=a
9. a+ab’=a+b, a(a’+b)=ab
10. a+ab=a, a(a+b)=b
11. ab+a’c=(a+c)(a’+b)
12. (a+b)’=a’b’, (ab)’=a’+b’ (DeMorgan)

아래 <그림 c>처럼 모든 변수가 항상 한번씩 사용된 product term을 minterm이라고 하고 모든 변수가 항상 한번씩 사용된 sum term을 maxterm이라고 한다.



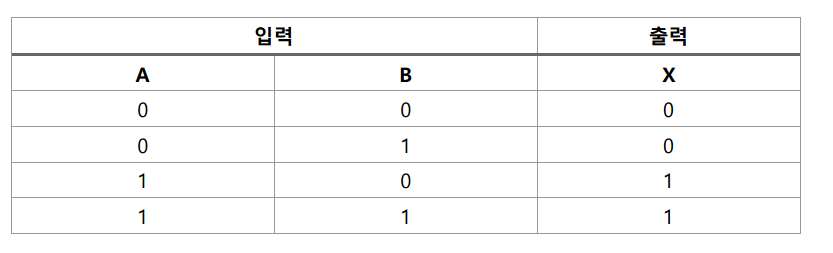
<그림 c> 변수 A,B,C에 대한 minterm과 maxterm

하지만 진리표를 작성하지 않고 minterm이나 maxterm으로 이루어진 함수에 관한 식을 얻고 싶을 때 우리는 아래 <그림 d>와 같은 카르노 맵이라는 것을 활용할 수 있다.



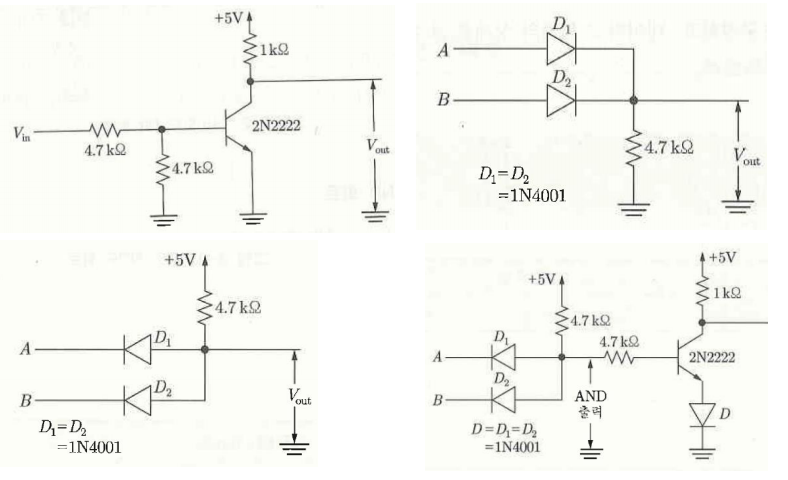
<그림 d> 변수 A, B, C에 대한 함수 f의 카르노 맵

위의 <그림 d>는 세 가지 변수에 대한 카르노 맵을 그린 것으로 가로 축은 A를 세로 축은 BC의 곱을 나타내었다. 이처럼 n개의 변수가 있을 때 카르노 맵의 칸은 2n개가 있고 변수에 개수를 적절히 나누어 가로와 세로 축을 구성하면 된다. 특이한 점은 BC를 나타내는 세로 축에 BC가 위에서부터 각각 00, 01, 10, 11이 아니라 00, 01, 11, 10이라는 것인데 이는 앞뒤로 한 개의 bit만이 다르게 하기 위해 나열한 방법이다. 위의 <그림 d>에서는 minterm이 1이 되는 부분이 AB’C’, A’BC, A’BC’, ABC’일 때인데 이를 모두 사용하여 함수 f에 대한 식 f(A,B,C)=AB’C’+A’BC+A’BC”+ABC’라고 표현하는 것이 아니라 k=0, 1 ,2, 3,..일 때 2k개의 묶음(implicants)으로 묶고, 이 중 더 큰 묶음에 포함되지 않는 묶음들(prime implicants)과 다른 묶음에 들어가지 않는 부분이 있는 묶음(essential implicants)를 찾아내어 최대한 간략하게 나타내어야 한다. 이러한 방법으로 <그림 d>에서는 f(A,B,C)=AC’+A’B라는 minimize된 형태로 함수 f를 나타낼 수 있다.

1. 실험 장비 및 재료
2. 실험 장비
3. NI ELVIS
4. 오실로스코프: PHILIPS 60 MHz Digital Storage Oscilloscope PM3335
5. 함수발생기: EZ FG-8002
6. DC power supply
7. 실험 재료
8. 저항, 커패시터
9. 1N4001, 2N2222A
10. 7400, 7402, 7403, 7404, 7408, 7420, 7432, 7486
11. 실험 방법
12. 반도체 소자 논리 회로와 기본 논리 소자

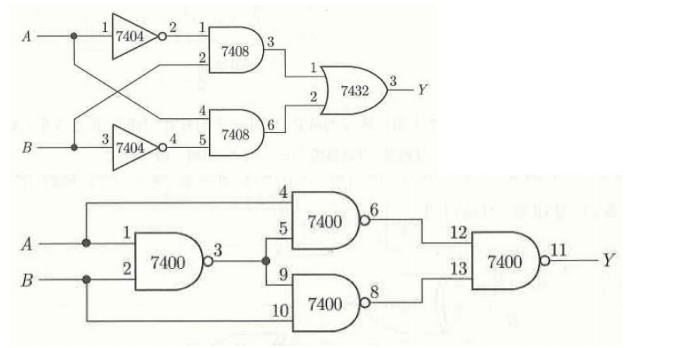
<그림 0> 진리표 예시

이전에 배웠던 다이오드와 트랜지스터를 사용하여 기본 디지털 논리 회로를 구성하고 입력에 대한 출력 상태를 위의 <그림 0>과 같이 진리표로 나타내어 기록해본다. 진리표 작성은 디지털 논리 회로에 관한 실험에서 중요한 부분이므로 본 실험을 통해 숙지할 수 있도록 한다.

1.  아래 <그림 1>과 같이 회로를 구성한 후, 위의 <그림 0>처럼 진리표를 작성하여 <그림 1>의 회로들이 각각 어떠한 논리를 구현한 회로인지를 알아본다. 입력에서의 0과 1의 상태는 본 회로에서 각각 0 V와 5 V를 인가해주어 표현하고, 출력에서의 0과 1은 LED를 연결하여 LED가 꺼진 상태를 0, 켜진 상태를 1이라고 정의한다.

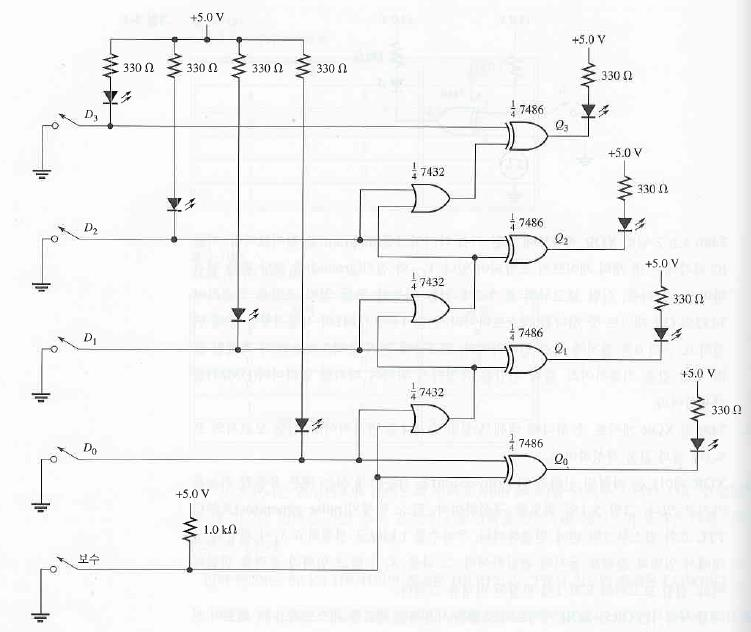
<그림 1> 반도체 소자 논리 회로

1. 7404, 7432, 7408, 7402, 7400 소자들의 매뉴얼을 참고하여 공통적으로 7번 핀에는 접지를, 14번 핀에는 +5 V를 연결하여 각각 소자에 대해 진리표를 작성해본다. 아래 <그림 2>의 회로를 구성하고 진리표를 작성하여 <그림 2>의 회로들이 어떠한 논리를 구현하고 있는지를 알아보고, <그림 2>의 회로 중 아래 회로에 대해 12번과 13번 단을 7400 소자가 아닌 7408 소자로 바꾸어 연결한 후, 진리표를 작성하여 이전의 경우와의 차이점에 대해 알아보도록 한다.



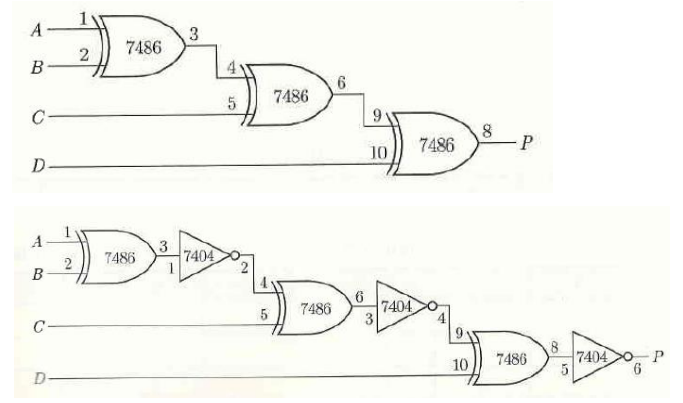
<그림 2> XOR 회로

1. 아래 <그림 3>의 회로를 구성하고 진리표를 작성한다. 이때, 보수(complement) 스위치를 개방한 상태에서 진리표를 작성해보고, 보수 스위치를 닫은 상태에서 진리표를 작성해본 후 두 가지 경우에서의 차이점에 대해 알아보도록 한다.



<그림 3> OR과 XOR을 사용한 보수 실행 회로

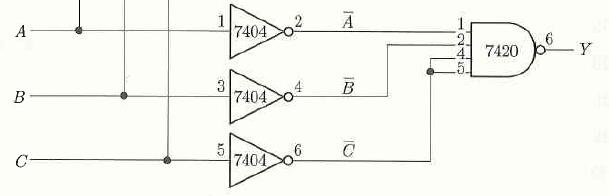
1. 아래 <그림 4>의 회로를 구성하고 진리표를 작성한다. 이때, 아래 <그림 4>의 회로들은 1 비트 에러 검출을 하는 데에 사용되는 회로로, 1의 총 개수가 짝수인지를 검출하는가, 홀수인지를 검출하는가에 따라 짝수 패리티 회로와 홀수 패리티 회로로 나뉘는 패리티 회로들이므로 실험을 통해 이에 대해 이해해본다.



<그림 4> 4비트 짝수, 홀수 패리티 검사기 회로

1. 부울 대수와 드 모르간의 정리

부울 대수는 1847년 George Boole이 논리의 수학적 해석을 위해 제안한 것으로 논리학 및 기초 수락에 응용되었다. 드 모르간의 정리는 부울의 절친한 수학자 DeMorgan이 부울 대수에서 중요한 역할을 하는 두 가지 정리를 제안한 것이다. 부울 대수와 드 모르간의 법칙을 이용하여 아래 <그림 5>의 회로의 진리표를 작성해본 후, 회로를 실제로 구성하여 진리표를 작성하여 이 두 가지 진리표를 비교해보면서 부울 대수와 드 모르간의 정리를 이해하도록 한다.



<그림 5> 부울 대수와 드 모르간 법칙의 이해를 위한 회로

1. 참고문헌

-Earl Gates, 전기전자공학, 1판, 북스힐, 2018년, pg. 241-263

-Alan B. Marcovitz, Introduction to logic design, 3판, McGraw-Hill Higher Education, 2009년, pg. 9-213