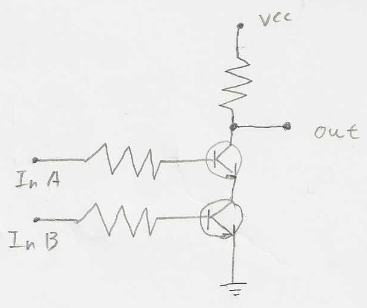
4주차 예비보고서

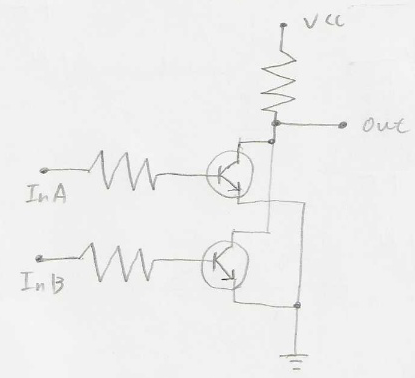
전공: 기계공학과 학년: 3학년 학번: 20191820 이름: 김형준

**1.**

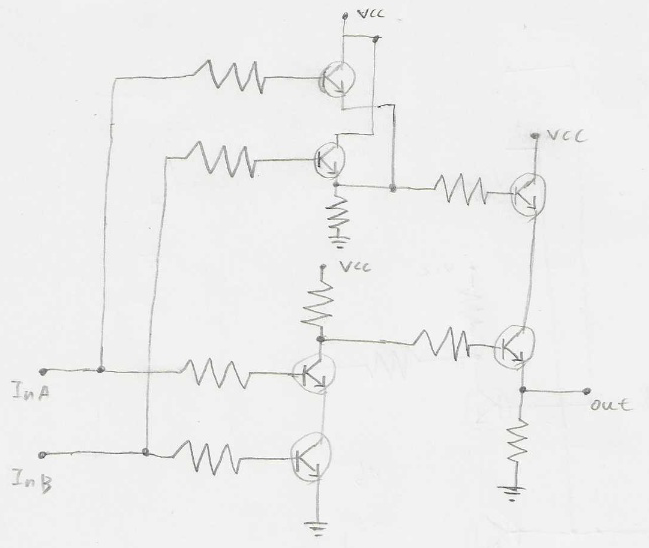
1-1. 논리게이트 NAND (Transistor-Level)



1-2. 논리게이트 NOR (Transistor-Level)



1-3. 논리게이트 XOR (Transistor-Level)

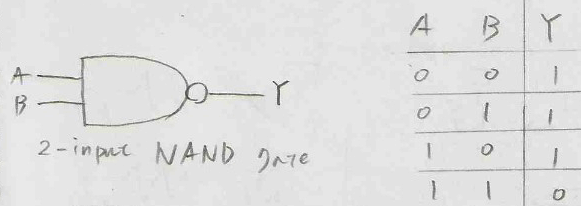


**2.**

2-1. NAND Logic

NAND Logic은 NOT AND의 의미로, AND Logic의 부정이다. 주어진 두 명제가 모두 참일 때만 거짓을 반환하는 논리 연산이고, Boolean 식에서는 로 나타낸다.

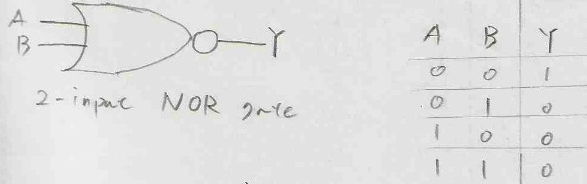
NAND Logic을 논리회로로 구현한 것을 NAND Gate라 하며, 모든 입력(Input)이 High(1)일 때만 출력(Ouput)으로 Low(0)를 반환하는 기본 논리소자이다. 이외의 모든 입력에 대해서는 High(1)를 반환하는 특성을 지닌다. 어떠한 논리 함수도 NAND 게이트만으로 구현할 수 있기 때문에, NAND 게이트는 함수의 완결성(Functional Completeness)을 지닌다. 입력이 두 개인 2-input NAND Gate의 다이어그램과 진리표는 아래 그림과 같다.



2-2. NOR Logic

NOR Logic은 NOT OR의 의미로, OR Logic의 부정이다. 주어진 두 명제 중 하나라도 참일 때 거짓을 반환하는 논리 연산이고, Boolean 식에서는 로 나타낸다.

NOR Logic을 논리회로로 구현한 것을 NOR Gate라 하며, 모든 입력(Input) 중 하나라도 High(1)일 경우, 출력(Ouput)으로 Low(0)를 반환하는 기본 논리소자이다. 입력 중 High(1)가 하나도 없는 경우에는 High(1)를 반환하는 특성을 지닌다. 어떠한 논리 함수도 NOR 게이트만으로 구현할 수 있기 때문에, NOR 게이트는 함수의 완결성(Functional Completeness)을 지닌다. 입력이 두 개인 2-input NOR Gate의 다이어그램과 진리표는 아래 그림과 같다.

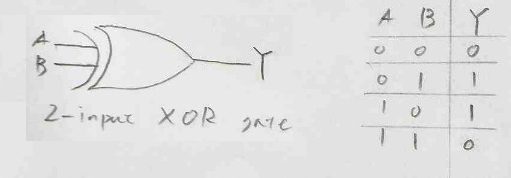


2-3. XOR Logic

XOR Logic은 Exclusive OR의 의미로, 베타적 논리합을 의미한다. 주어진 두 명제 중 참인 명제가

1개일 때 참을, 아닌경우 거짓을 반환하는 논리 연산이고, Boolean 식에서는 로 나타낸다.

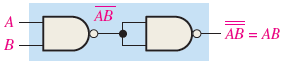
XOR Logic을 논리회로로 구현한 것을 XOR Gate라 하며, 모든 입력(Input) 중 High(1)의 개수가 홀수인 경우, 출력(Ouput)으로 High(1)를 반환하는 기본 논리소자이다. 입력 중 High(1)의 개수가 짝수인 경우에는 Low(0)를 반환하는 특성을 지닌다. XOR 게이트는 로도 나타 낼 수 있다. 입력이 두 개인 2-input XOR Gate의 다이어그램과 진리표는 아래 그림과 같다.



**3.**

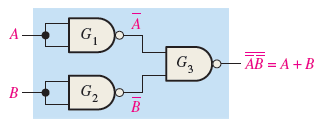
NAND 게이트와 NOR 게이트는 함수의 완결성을 지니고 있으므로 어떠한 논리 함수도 NAND 또는 NOR게이트만으로 만들 수 있다. 따라서 NAND 게이트 또는 NOR 게이트만 사용해서 AND, OR, NOT 게이트를 만들 수 있는데, 이렇게 한 종류의 게이트만을 사용하여 회로를 구성하면 로직 개수는 증가하지만, 설계나 제조가 매우 쉬워지는 장점이 있다.

3-1. NAND로 AND게이트의 구현



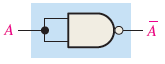
NAND게이트를 직렬로 두 개를 연결해서 AND게이트를 구현할 수 있다.

3-2. NAND로 OR게이트의 구현



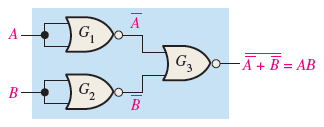
NAND게이트를 병렬로 연결해서 각각의 입력을 반전시킨 다음(G1, G2), 각각의 출력을 NAND게이트(G3)의 입력으로 넣으면 OR게이트를 구현할 수 있다.

3-3. NAND로 NOT게이트의 구현



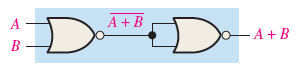
주어진 입력을 NAND게이트의 두 입력에 동일하게 넣으면 NOT게이트를 구현할 수 있다.

3-4. NOR로 AND게이트의 구현



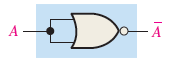
NOR게이트를 병렬로 연결해서 각각의 입력을 반전시킨 다음(G1, G2), 각각의 출력을 NOR게이트(G3)의 입력으로 넣으면 AND게이트를 구현할 수 있다.

3-5. NOR로 OR게이트의 구현



NOR게이트를 직렬로 두 개를 연결해서 OR게이트를 구현할 수 있다.

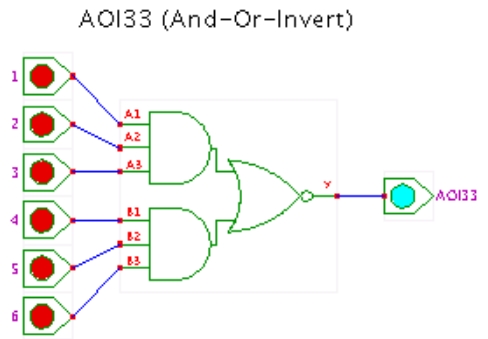
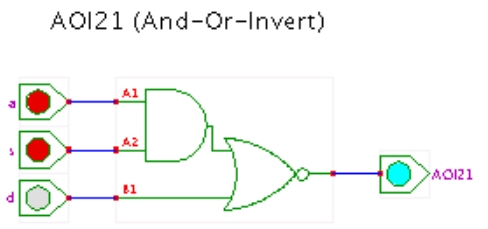
3-6. NOR로 NOT게이트의 구현



주어진 입력을 NOR게이트의 두 입력에 동일하게 넣으면 NOT게이트를 구현할 수 있다.

**4.**

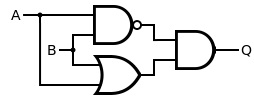
AND-OR-INVERT logic (AOI logic)은 1개이상의 AND게이트와 그 뒤에 연결되어있는 NOR게이트로 이루어진 2-level 복합논리 회로이다. AOI에 붙는 숫자는 회로에 입력되는 값의 개수를 의미하며, AND게이트에 들어오는 입력의 개수를 각각 표기한다. (AND게이트가 없는 경우는 1로 표기)



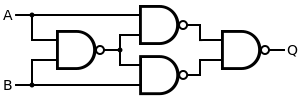
예를 들어, 왼쪽 그림처럼 첫 번째에 2개의 입력, 두 번째에 1개의 입력인 경우는 AOI21 게이트로 표기하며, 오른쪽 그림처럼 첫 번째에 3개의 입력, 두 번째에 3개의 입력인 경우는 AOI33 게이트로 표기한다. AOI 게이트는 CMOS회로에서 쉽게 구현할 수 있고, AOI 게이트의 기능을 AND, OR, NOT 게이트를 각각 따로 사용하여 구현하는 것보다 사용되는 총 트랜지스터의 개수가 적다는 장점이 있다. 예를 들면 AOI21 게이트의 경우 CMOS에서 6개의 트랜지스터로 구현할 수 있는데, 이를 NAND, NOR, NOT게이트의 조합으로 구현하면 NAND게이트(4) + NOR게이트(4) + NOT게이트(2) = 10개의 트랜지스터를 사용해야 하므로 비효율적이다. 따라서 AOI 게이트를 사용하는 경우 사용하는 총 트랜지스터의 개수가 감소하므로, 전체 회로의 속도 증가 및 전력 감소, 제조비용 감소 등의 효과를 볼 수 있다.

**5.**

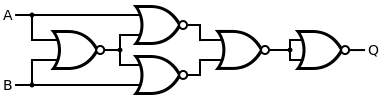
XOR 게이트는 Boolean 식으로 로 표기하며, , 등으로 나타낼 수 있다. 아래는 기본 논리 게이트의 조합으로 XOR 게이트를 구현한 예시들이다.



위 그림은 AND, OR, NAND게이트를 사용해 XOR 게이트를 구현하였다.



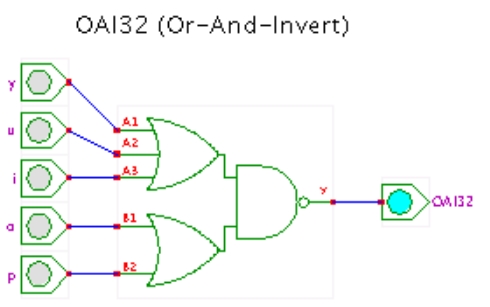
위 그림은 NAND게이트만을 사용해 XOR 게이트를 구현하였다. (NAND 게이트의 함수의 완결성 사용)



위 그림은 NOR게이트만을 사용해 XOR 게이트를 구현하였다. (NOR 게이트의 함수의 완결성 사용)

**6.**

OR-AND-INVERT logic (OAI logic)은 AND-OR-INVERT logic (AOI logic)의 보수(complement)이며, 1개이상의 OR게이트와 그 뒤에 연결되어있는 NAND게이트로 이루어진 2-level 복합논리 회로이다. OAI에 붙는 숫자는 AOI 게이트와 마찬가지로, 회로에 입력되는 값의 개수를 의미하며, OR게이트에 들어오는 입력의 개수를 각각 표기한다. (OR게이트가 없는 경우는 1로 표기)



예를 들어, 위 그림처럼 첫 번째에 3개의 입력, 두 번째에 2개의 입력인 경우는 OAI32 게이트로 표기한다. OAI 게이트는 CMOS회로에서 쉽게 구현할 수 있고, AOI 게이트와 마찬가지로 OAI 게이트의 기능을 AND, OR, NOT 게이트를 각각 따로 사용하여 구현하는 것보다 사용되는 총 트랜지스터의 개수가 적다는 장점이 있다. 따라서 OAI 게이트를 사용하면, 전체 회로의 속도 증가 및 전력 감소, 제조비용 감소 등의 효과를 볼 수 있다.