2018년 2학기 컴퓨터공학실험Ⅱ
CSE3016-05반 4주차 예비 보고서

학번: 20171665
이름: 이 선 호

2018. 09. 28

목 차

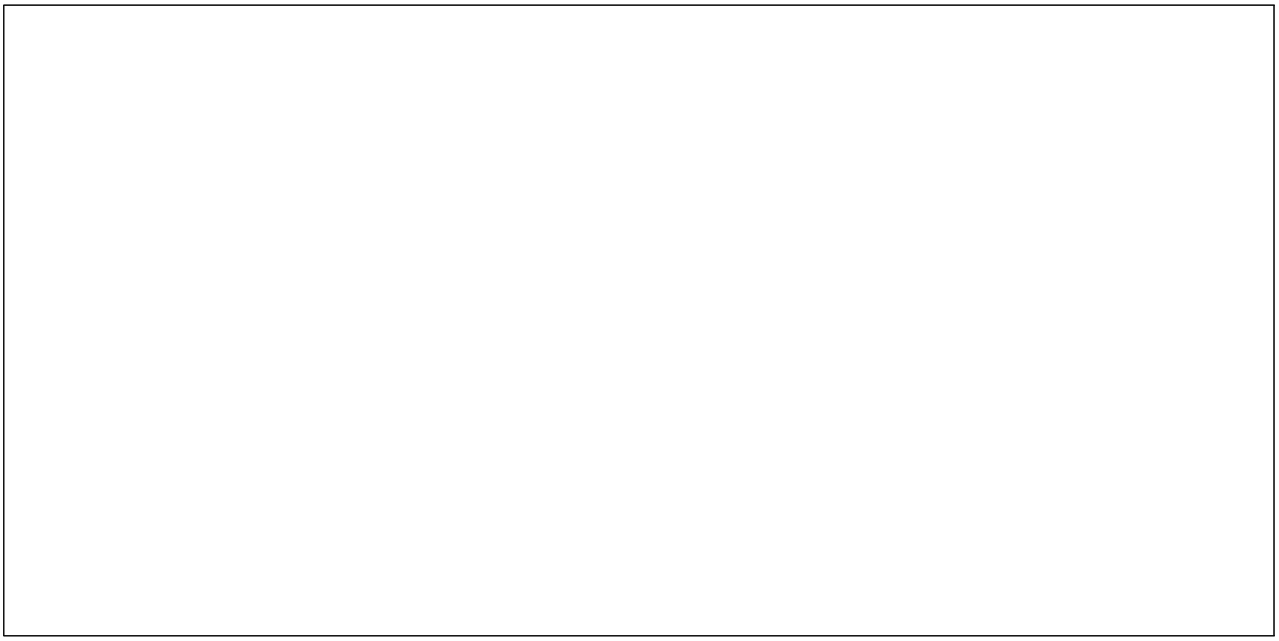
Ⅰ Gate 구조를 Transistor Level로 그리기

NAND, NOR, XOR Gate의 구조를 항목별로 나눠서 Transistor-Level로 그렸다.

1. NAND Gate

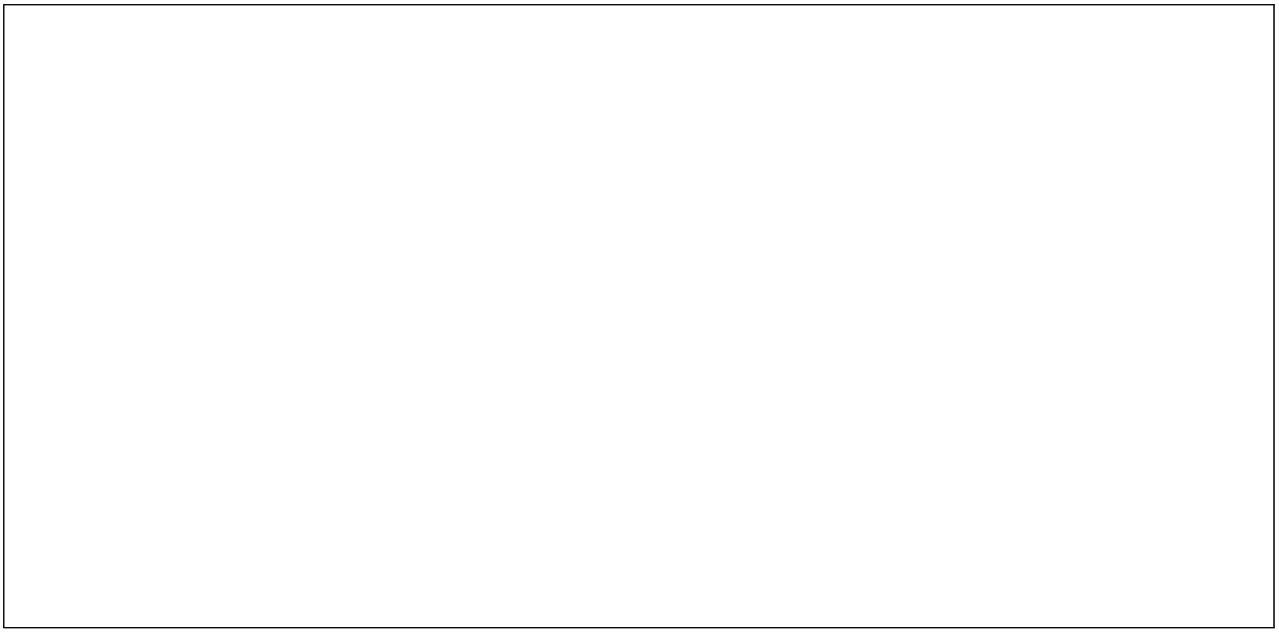
[그림 Ⅰ-1]

2. NOR Gate



[그림 Ⅰ-2]

3. XOR Gate



[그림 Ⅰ-3]

Ⅱ NAND / NOR / XOR Logic

1. NAND / NOR / XOR Logic의 특성

NAND Logic은 input 값들이 모두 1일 때 output으로 0을 출력하며, 그렇지 않을 때에는 1을 출력하는 논리이다. 다시 말해서, AND Logic과 input에 따른 output이 완전히 상반된다. 논리식으로 표현하면 $\sim(A \bullet B)$이고 논리 회로 기호로 표현하면 [그림 Ⅱ-1]과 같으며, Functional Expression으로 나타내면 $F=(xy)'$이다. 또한 Transistor-Level로 표현하면 [그림 Ⅰ-1]이고, 이 때 사용되는 트랜지스터 개수는 4개이며 전파 지연(Gate Delay) 시간은 약 1.4 나노초다. 진리표로 나타내면 [표 Ⅱ-1]과 같다.

NOR Logic은 input 값들 중에 하나라도 1이 있을 때 output으로 0을 출력하며, 그렇지 않고 input 값들이 모두 0일 경우에는 1을 출력하는 논리이다. 다시 말해서, OR Logic과 input에 따른 output이 완전히 상반된다. 논리식으로 표현하면 $\sim(A+B)$이고 논리 회로 기호로 표현하면 [그림 Ⅱ-2]와 같으며, Functional Expression으로 나타내면 $F=(x+y)'$이다. 또한 Transistor-Level로 표현하면 [그림 Ⅰ-2]이고, 이 때 사용되는 트랜지스터 개수는 4개이며 전파 지연 시간은 약 1.4 나노초다. 진리표로 나타

내면 [표 Ⅱ-2]와 같다.

XOR Logic은 두 개의 input 가운데 1개만 값이 1이면 output을 1로 출력한다. 다시 말해서, 두 개의 input 값이 서로 같으면 output으로 0을, 서로 다르면 1을 출력하는 것이다. 예를 들어, input 값으로 0과 1이 들어오면 output으로 1을, 반대로 input이 모두 0 또는 1이면 output으로 0을 출력한다. 논리식으로 표현하면 $A \oplus B$이고 논리 회로 기호로 표현하면 [그림 Ⅱ-3]과 같으며, Functional Expression으로 나타내면 $F = x \oplus y$이다. 또한 Transistor-Level로 표현하면 [그림 Ⅰ-3]이고, 이 때 사용되는 트랜지스터 개수는 14개이며 전파 지연 시간은 약 4.2 나노초다. 진리표로 나타내면 [표 Ⅱ-3]과 같다.

x	y	(xy)'
0	0	1
0	1	1
1	0	1
1	1	0

[표 Ⅱ-1]

x	y	(x+y)'
0	0	1
0	1	0
1	0	0
1	1	0

[표 Ⅱ-2]

x	y	x⊕y
0	0	0
0	1	1
1	0	1
1	1	0

[표 Ⅱ-3]

[그림 Ⅱ-1]

[그림 Ⅱ-2]

[그림 Ⅱ-3]

2. 기본 논리게이트(AND/OR/NOT)와 변환 관계

실제로 논리회로를 설계할 때에는 AND, OR, NOT 게이트보다 NAND, NOR 게이트가 주로 사용되는데, 그 이유는 AND나 OR 게이트에 비해 NAND나 NOR 게이트로 제작하는 것이 상대적으로 쉽고 효율적이기 때문이다. 게이트에 사용되는 트랜지스터 개수만 봐도 NAND와 NOR 게이트는 4개이고 전파 지연이 1.4 나노초인데 반해 AND, OR 게이트는 6개가 사용이 되고 전파 지연이 2.4 나노초다. 그리고 NOT의 트랜지스터 사용 개수는 2개이고, 전파 지연은 1 나노초인데, 이 비용이 AND와 OR가 같이 사용되면 소모량이 더 늘어나게 된다. 그래서 NAND와 NOR 게이트를 AND, OR, NOT 게이트로 변환하거나 또는 그 반대의 경우가 생길 수 있다.

NAND, NOR 게이트와 기본 논리 게이트 사이의 변환 과정에서는 드모르간의 법칙이

자주 사용된다. 만일 회로를 NAND 게이트만을 이용하여 표현하고자 한 경우에는 논리식이 곱의 합 형태로 표현되어 있어야 효율적이다. 예를 들어서 어떤 두 input A와 B, 그리고 C와 D가 각각 AND 논리로 연결되고 각각의 output이 다시 OR 논리로 연결된다면, 이는 모두 NAND 논리의 사용으로 바꿀 수 있다. 예를 들어, $F = ab + cd$라는 논리식은 $ab + cd = \overline{\overline{ab}\,\overline{cd}}$와 같이 식이 성립하므로 모든 논리 연산을 NAND 연산으로 고치는 것이 가능하다. 반면에 $(a+b)(c+d) = \overline{\overline{(a+b)} + \overline{(c+d)}}$와 같이 POS(Product of Sum) 식의 연산을 모두 NOR를 사용한 식으로 변환할 수 있다.

Ⅲ AND-OR-INVERT와 XOR Logic

1. AND-OR-INVERT Logic의 응용

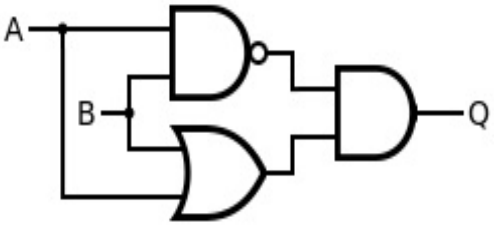
AND-OR-INVERT Logic(AOI)은 두 개 이상 또는 한 개의 AND 게이트와 NOR 게이트를 합쳐서 구성한 두 단계 논리함수를 뜻한다. 이를 논리식으로 표현하면 $F = \overline{(A \wedge B) \vee (C \wedge D)}$ 또는 $F = \overline{A \vee (B \wedge C)}$로 나타내며, 4개 이상의 input이 들어올 때 모든 input 값을 알지 않아도 output을 예측 가능한 특징을 지닌다.

AOI는 CMOS(Complementary Mental-Oxide-Semiconductor, 집적회로의 한 종류)를 제작하는 데 효율적인데, 그 이유는 AOI 게이트에 사용되는 트랜지스터의 총 개수가 AND, NOT, 또는 OR 게이트를 따로 사용하여 실행할 때보다 상대적으로 적기 때문이다. 예컨대, AOI 게이트가 CMOS에서 6개의 트랜지스터로 제작 가능한 반면에 NAND 게이트 2개와 inverter 1개로 총 10개, 또는 NOR 게이트 2개로 총 8개 등 상대적으로 트랜지스터 사용 총 개수가 더 많아진다.

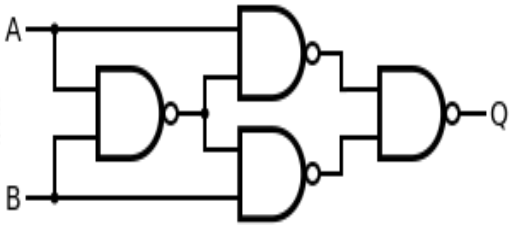
2. XOR Logic 구현 방법

위에서 살펴본 바와 같이 XOR Logic은 회로에서 input 값이 상호 배타적인 경우 output으로 1을, 그 외에는 0을 갖는다는 것을 알고 있다. 그래서 XOR 게이트는 두 개 이상의 input 값을 모두 알아야 결과를 알 수 있다. XOR는 AND, OR, NOT 등 기본 게이트를 통해서 여러 방법으로 구현가능한데, 이는 XOR의 논리식을 자세히 살펴보면 그 방법들을 유추할 수 있다.

XOR의 논리식은 $F = x \oplus y = x\bar{y} + \bar{x}y$인데 여기서 $x(\bar{x}+\bar{y}) + y(\bar{x}+\bar{y}) = \bar{x}y + x\bar{y}$이므로 NOT 게이트 2개, OR 게이트 2개, 그리고 AND 게이트 2개로 구현할 수 있다. 또한 위의 식을 드모르간의 법칙을 써서 논리곱의 식으로 정리하면 $x(\bar{x}+\bar{y}) + y(\bar{x}+\bar{y}) = x(\overline{xy}) + y(\overline{xy}) = (\overline{xy})(x+y)$임을 알 수 있다. 그래서 NAND 게이트 1개, AND 게이트 1개, 그리고 OR 게이트 1개를 혼합하여 XOR 회로를 구현할 수 있다. 이를 구현한 회로는 [그림 Ⅲ-1]을 보면 알 수 있다. 그리고 드모르간의 법칙을 이용하여 $x(\overline{xy}) + y(\overline{xy}) = \overline{x(\overline{xy})y(\overline{xy})}$와 같이 나타낼 수 있는데, 이는 다시 말해서 NAND 게이트 4개만으로도 XOR 게이트 구현이 가능하다는 것을 알 수 있다. 이를 구현한 회로는 [그림 Ⅲ-2]를 보면 알 수 있다.



[그림 Ⅲ-1]



[그림 Ⅲ-2]

Ⅳ 기타 이론

1. 패리티 비트(Parity Bit)

XOR Logic은 주로 기기 간의 통신 오류를 검출할 때 사용하는 패리티 비트에 이용할 수 있다. 어떤 통신 방법이든지 간에 모든 통신에는 오류 또는 누락이 발생할 수 있는데, 예를 들어 중간에 하나의 비트가 누락될 수도 있고, 노이즈 등 다른 요인으로 인해 신호가 잘못 읽혀질 수도 있다. 그래서 데이터 통신에는 오류 검출 단계가 포함되는 경우가 대다수인데, 이 때 대표적으로 사용하는 방법은 바로 XOR Logic을 이용한 패리티 비트를 추가하는 방법이다.

패리티 비트 사용 방법에는 홀수 패리티 비트(Odd Parity Bit)와 짝수 패리티 비트(Even Parity Bit)가 있다. 여기서 홀수 패리티 비트 방법은 이는 1의 개수가 홀수일 때 패리티 비트를 0, 짝수일 때 패리티 비트를 1로 설정하여 전체 데이터에 있는 1의 개수가 홀수가 되도록 하는 것이다. 짝수 패리티 방법인 이와 반대된다. 만일 전송 과정에서 오류가 발생했다면 패리티 비트의 값이 틀리게 작성되었을 여지가 있는데, 여

기서 오류를 검출하기 위해 패리티 비트와 전송 데이터의 각 자리의 비트를 서로 XOR 연산해 나간다. 홀수 패리티 방법에서 패리티 비트와 데이터의 각 자리 비트를 차례대로 XOR 연산한 결과가 1이 나오면 오류가 없는 것이고, 0이 나오게 되면 오류가 발생했다는 의미이다.

패리티 비트의 단점은 오류 검출이 상대적으로 정확하지 않다는 것이다. 전송 과정에서 단일 비트가 아니라 복수개의 비트가 오류가 발생한다면 1의 개수가 홀수인지 또는 짝수인지 그 사실은 처음과 달리지지 않기 때문이다.