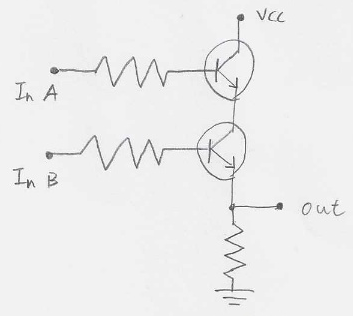
3주차 예비보고서

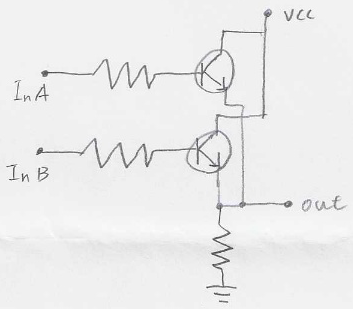
전공: 기계공학과 학년: 3학년 학번: 20191820 이름: 김형준

**1.**

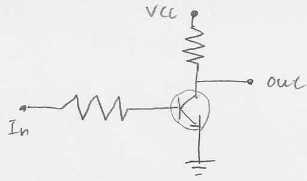
1-1. 논리게이트 AND (Transistor-Level)



1-2. 논리게이트 OR (Transistor-Level)



1-3. 논리게이트 NOT (Transistor-Level)

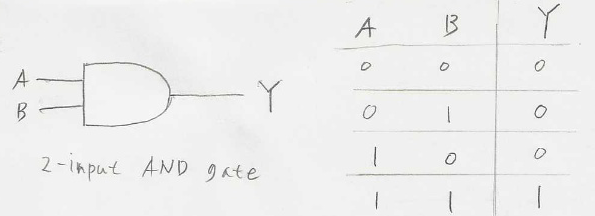


**2.**

2-1. AND Logic

AND Logic은 논리곱을 의미하며, 주어진 두 명제가 모두 참일 때만 참을 반환하는 논리 연산이다.

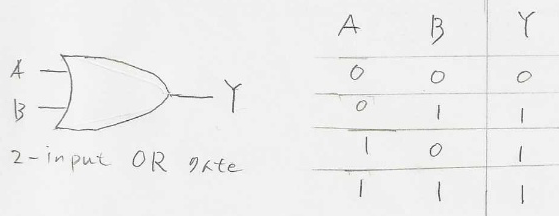
Boolean 식에서는 A∧B, A∙B, A&B로 나타낸다. AND Logic을 논리회로로 구현한 것을 AND Gate라 하며, 모든 입력(Input)이 High(1)일 때만 출력(Ouput)으로 High(1)를 반환하는 기본 논리소자이다. 이외의 모든 입력에 대해서는 Low(0)를 반환하는 특성을 지닌다. 입력이 두 개인 2-input AND Gate의 다이어그램과 진리표는 아래 그림과 같다.



2-2. OR Logic

OR Logic은 논리합을 의미하며, 주어진 두 명제 중 하나라도 참일 때 참을 반환하는 논리 연산이다.

Boolean 식에서는 A∨B, A+B, A|B로 나타낸다. OR Logic을 논리회로로 구현한 것을 OR Gate라 하며, 모든 입력(Input) 중 하나라도 High(1)일 경우, 출력(Ouput)으로 High(1)를 반환하는 기본 논리소자이다. 입력 중 High(1)가 하나도 없는 경우에는 Low(0)를 반환하는 특성을 지닌다. 입력이 두개인 2-input OR Gate의 다이어그램과 진리표는 아래 그림과 같다.



2-3. NOT Logic

NOT Logic은 부정을 의미하며, 주어진 명제가 참이면 거짓을, 거짓이면 참을 반환하는 논리 연산이다. Boolean 식에서는 ¬A, , ~A로 나타낸다. NOT Logic을 논리회로로 구현한 것을 NOT Gate라 하며, 입력(Input)을 반전시킨 논리값을 반환하는 기본 논리소자이다. 입력이 High(1)인 경우 Low(0)를, 입력이 Low(0)인 경우 High(1)를 반환하는 특성을 지닌다. NOT Gate의 다이어그램과 진리표는 아래 그림과 같다.

****

**3.**

Fan-out 이란 논리회로에서 정상동작에 영향을 주지 않고, 하나의 논리 게이트의 출력에 연결될 수 있는 논리 게이트 입력의 최대 개수를 의미한다. 이는 각 소자의 출력단에는 최대로 흐를 수 있는 전류에 제한이 있기 때문인데, Fan-out을 초과해서 연결시키면 출력전류가 지나치게 커져서 소자의 회로가 손상 될 수 있다. 그렇지 않은 경우에는 출력단의 전압강하 때문에 출력전압 레벨이 기준치에 도달하지 못해 연결되어있는 다음 논리 게이트의 입력단에 신호가 제대로 전달되지 않아 시스템 오동작의 원인이 될 수 있다.

Fan-out을 초과하는 경우를 방지하려면, 회로 중간에 출력 전류의 용량이 큰 버퍼 게이트를 사용하거나, 여러 개의 버퍼를 사용하여 부하를 나눠서 분담시키는 등의 방법을 사용해야 한다.

Fan-out은 출력단의 출력전류와 입력단의 입력전류의 비로 결정할 수 있는데,

Fan-out 또는 Fan-out 으로 계산할 수 있다.

**4.**

전파지연(propagation delay)은 논리회로에서 입력 신호값의 변화가 입력에서 출력까지 전달되는데 걸리는 평균시간을 의미한다.

전파 지연은 (입력신호에 반응하여 출력값이 0(Low)에서 1(High)로 변할 때 걸리는 시간)와 (입력신호에 반응하여 출력값이 1(High)에서 0(Low)으로 변할 때 걸리는 시간) 두 가지로 구분할 수 있다. 전체 전파 지연 은 와 의 산술평균으로 구할 수 있다.

논리회로의 전파지연은 회로를 구성하고 있는 소자들의 전파지연을 모두 합친 것이기 때문에

회로의 지연시간은 논리 게이트들의 지연시간과 개수에 영향을 받으며, 입력 신호가 여러 개의 게이트를 통과할수록 전체 지연시간이 길어지므로 회로 설계시 전파지연을 고려해야 한다.

**5.**

Verilog의 function과 task는 동일한 기능을 반복적으로 사용해야 할 때, 코드의 반복을 줄이기 위해 사용한다. 이를 통해, 소스 코드의 가독성을 올리고 디버깅을 용이하게 만들 수 있다.

function과 task는 공통적으로 행위 수준의 문장만을 포함 할 수 있고, 모듈과 다르게 initial문과 always문을 포함할 수 없다. 또한, 동시에 두 곳 이상에서 하나의 task 또는 function을 호출하는 경우, task 또는 function 키워드 바로 뒤에 automatic 키워드를 추가해 사용중인 내부 변수들의 값이 공유되는 문제를 해결할 수 있다.

function은 function과 endfunction 키워드로 감싸서 정의하며, 특성은 다음과 같다.

∙ 시간 지연을 포함할 수 없어 zero delay를 갖는 조합회로의 구현에 사용된다.

∙ 내부에서 다른 function을 호출 가능하다. (task는 호출 불가능)

∙ delay, event, timing control 등 시간지연과 관련된 keyword를 포함 할 수 없다.

∙ output, inout 인수를 가질 수 없고, 하나 이상의 input 인수를 가져야만 한다.

∙ 출력 변수는 함수명과 동일하며, 항상 1개의 결과값을 반환한다.

task는 task와 endtask 키워드로 감싸서 정의하며, 특성은 다음과 같다.

∙ 시간 지연을 포함할 수 있어 delay, event, timing문을 사용할 수 있다.

∙ 내부에서 다른 function과 task를 호출 가능하다.

∙ 0개 이상의 input, output, inout 인자를 사용할 수 있어 한번에 여러 개의 값을 반환할 수 있다.

∙ non-zero 시뮬레이션 시간에 수행될 수 있다.

∙ 시간지연 기능을 사용하지 않을 경우에만 합성이 가능하다.