**컴퓨터공학 설계 및 실험Ⅱ**

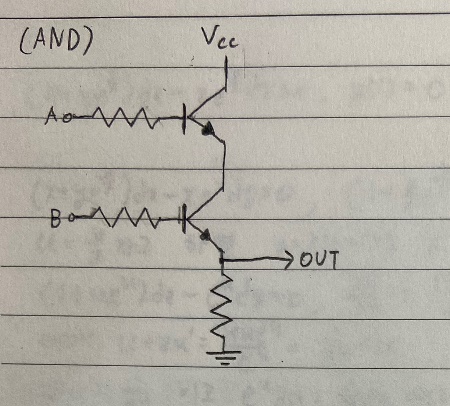
3주차 예비보고서

서강대학교 공학부 컴퓨터공학 전공

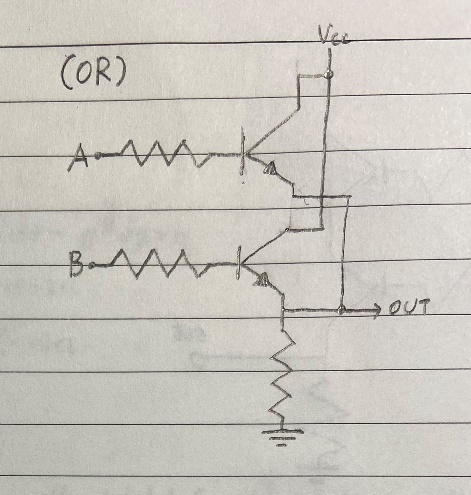
20171646 박태윤

**1. 논리게이트 AND/OR/NOT의 구조를 Transistor-Level로 그리시오.**

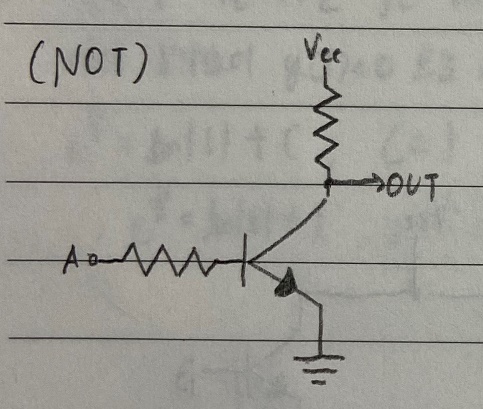
-AND 게이트



-OR게이트



-NOT게이트



**2. AND/OR/NOT Logic의 특성에 조사하시오.**

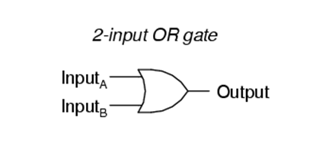
-AND게이트

AND게이트는 논리곱(C = A ∙ B)을 표현한다. 이는 즉 입력이 모두 High(1)인 경우에만 High신호를 출력하고 그 이외의 경우에는 모두 Low(0)신호를 출력한다는 것이다. 진리표와 논리회로 기호는 다음과 같다.

|  |  |  |
| --- | --- | --- |
| A | B | Output |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

|  |
| --- |
|  |

-OR게이트

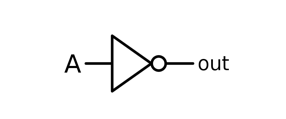
 OR게이트는 논리합(C = A + B)을 표현한다. 입력 중 하나 이상 High(1)인 경우 High신호를 출력하고 모든 입력이 Low(0)인 경우 Low신호를 출력한다. 진리표와 논리회로 기호는 다음과 같다.

|  |  |  |
| --- | --- | --- |
| A | B | Output |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

|  |
| --- |
|  |

-NOT게이트

NOT게이트는 입력값과 반대되는 값을 출력시킨다. Inverter(인버터)라 부르기도 한다. 추가적으로 NOT게이트를 두개 연결한 BUFFER게이트도 존재한다. BUFFER게이트는 입력값을 변화시키지는 않지만 신호의 세기를 강화시키는 역할을 한다. NOT게이트의 진리표와 논리회로기호는 다음과 같다.



|  |  |
| --- | --- |
| A | Output |
| 1 | 0 |
| 0 | 1 |

|  |
| --- |
|  |

**3. Fan-out에 대하여 조사하시오.**

디지털 회로에서 TTL 또는 CMOS와 같은 표준논리소자들은 1개의 출력신호에 접속할 수 있는 입력신호의 수가 제한적이다. 이 입력신호의 수를 제한시키고 지정하는 것이 Fan-out(팬아웃)이다. Fan-out이 지정되는 이유는 각 소자의 출력단에 최대로 흐를 수 있는 전류에 제한이 있기 때문이고, 만일 지정된 Fan-out을 초과하여 접속하게 되면 출력전류가 과대하여 회로가 소손될 수 있으며 출력단의 전압강하 때문에 출력전압이 규정치로 유지되지 못하여 그 다음에 입력되는 신호의 논리상태를 보장할 수 없게 되기 때문이다. 그러므로 Fan-out을 초과하여 부하를 접속하게 될 경우에는 중간에 출력전류 용량이 큰 버퍼를 쓰거나, 여러 개의 버퍼를 사용하여 부하를 분담시키는 것이 바람직하다. Fan-out은 다음과 같은 식에 의해서 결정된다.

**IOH : Output High Current**

**IIH : Input High Current**

**IOL : Output Low Current**

**IIL : Input Low Current**

**4. 전파지연에 대하여 조사하시오.**

전파지연이란 신호 값의 변화가 입력에서 출력까지 전달되는데 걸리는 평균시간을 의미한다. 경우에 따라 입력 신호가 변경되었을 때 출력이 최종 출력 수준의 50%에 도달하는 데 걸리는 시간을 의미하기도 한다. 논리게이트의 지연시간 및 개수 등에 영향을 받는다. 전파지연을 감소시키면 데이터를 빠르게 처리할 수 있고 전반적으로 시스템 성능을 향상시킬 수 있다. 다음과 같이 계산할 수 있다.

**TPD : 전파지연시간**

**TPHL : 입력신호에 반응하여 출력이 1에서 0으로 변화하는데 걸리는 시간**

**TPLH : 입력신호에 반응하여 출력이 0에서 1로 변화하는데 걸리는 시간**

**\* 1/2**

**5. Verilog의 task 및 function에 대해 조사하시오.**

Verilog에서 task와 function은 마치 C언어의 함수처럼 사용된다. behavioral구문만을 포함할 수 있으며 always, initial구문을 포함할 수 없거나 wire변수를 가지지 못하는 등 공통점도 존재하지만 몇 가지 부분에서 차이점을 보인다. 이를 정리하면 다음과 같다.

|  |  |
| --- | --- |
| **task** | **function** |
| 다른 task와 function  모두 호출할 수 있다. | 다른 function은 호출할 수 있지만  다른 task는 호출할 수 없다. |
| 여러 인수를 가지거나 가지지 않을 수도 있다. | 적어도 하나 이상의 input인수를 갖는다. Output, inout 인수를 가질 수 없다. |
| timing 제어문을 통해 non-zero 시뮬레이션 타임에도 실행될 수 있다. | 항상 0시뮬레이션 타임에 실행된다. |
| delay, timing, event 제어 구문을 쓸 수 있다. | delay, timing, event 제어 구문을 쓸 수 없다. |
| Output등의 인수를 통해 여러 값 전달가능. | 단 하나의 return값을 가짐. |

task와 function의 기본 형태는 다음과 같다.

-task의 기본 형태

task <name\_of\_task>;

<parameter\_declaration>

begin

<statement\_or\_null>

end

endtask

-function의 기본 형태

function <range\_or\_type>? <name\_of\_function>;

<parameter\_declaration>

begin

<statement>

end

endfunction