3주차 예비보고서

전공: 컴퓨터공학과/생명과학과 학년: 9학기 학번: 20181435 이름: 박다희

**1.**

텍스트, 도표, 친필, 스케치이(가) 표시된 사진

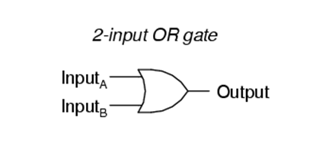
자동 생성된 설명

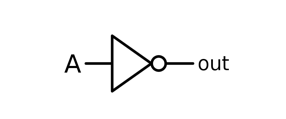
**2.**

먼저 AND gate는 논리곱(C = A ∙ B)을 표현하는 gate로 입력 값이 다 1인 경우에만 출력 값이 1이 되고 그 이외의 입력 값에는 출력 값으로 0이 나온다. AND gate의 논리 회로 그림은 다음과 같다.

텍스트, 폰트, 도표, 화이트이(가) 표시된 사진

자동 생성된 설명

OR gate는 논리합(C = A + B)을 나타내는 gate로, 입력 값이 다 0인 경우에만 출력 값이 0이 되고 그 외의 경우에는 출력 값이 1이 나오는 gate이다. OR gate의 논리 회로 그림은 다음과 같다.

 마지막으로 NOT gate는 들어온 값에 대해 반대의 값을 내보내는 gate로 1이 들어오면 0을 출력 값으로 보내고, 0이 들어오면 1을 내보내는 gate이다. Inverter라고 부르기도 하는데 NOT gate를 두 개 이어서 연결하면 buffer gate라고 한다. 이는 입력 값을 그대로 내보내지만 신호의 세기를 키우는 역할을 한다. NOT gate의 논리 회로 그림은 다음과 같다.

**3.**

Fan-out은 논리 회로에서 하나의 출력 신호가 다른 논리 게이트의 입력 신호로 얼마나 많이 분기되거나 또는 전달되는 지 나타낼 때 사용하는 개념이다. 하나의 출력 신호가 여러 개의 다른 게이트의 입력으로 사용될수록 fan-out이 증가한다. Fan-out을 지정하는 이유는 최대로 흐를 수 있는 전류의 제한이 각각의 소자 출력단에 있고, 만약 지정된 fan-out보다 fan-out이 더 커지면 회로의 신호가 전달될 때에 오류가 생기거나 회로 자체가 손상될 수도 있다. 또한 전달 될 때에도 더 많은 시간이 걸려서 회로의 동작 속도가 감소될 수 있고, 회로의 전체 지연 시간이 늘어날 수도 있다. 따라서 만약 fan-out을 지정된 것보다 넘어서 사용하게 될 때에는 중간에 출력전류의 용량이 큰 버퍼를 사용하거나 또는 여러 개의 버퍼를 써서 분담시키는 것이 좋다. Fan-out은 이와 같은 식으로 결정되는데, IOH는 Output High Current를, IIH는 Input High Current를, IOL은 Output Low Current를 그리고 IIL은 Input Low Current를 의미한다. 이렇게 fan-out을 최적화하도록 관리한다면 회로의 안정성을 유지하고 성능을 높일 수 있고 이는 전체적인 시스템의 효율성을 더 향상시킬 수 있다.

**4.**

전파 지연은 논리 회로의 입력 신호가 출력 신호로 전달되는 데에 걸리는 평균 시간을 의미한다. 또는 상황에 따라 입력 신호가 바뀌었을 때 출력 신호가 최종 출력 수준의 50%까지 되는 데에 걸리는 시간을 나타내기도 한다. 전파 지연은 논리 게이트의 개수나 논리 게이트의 지연 시간 등에 영향을 받으며 만약 전파 지연을 감소시킨다면 데이터를 보다 빠르게 처리할 수 있고 전체 시스템의 성능을 높일 수 있다. 전파 지연을 구하는 식은 이고, (Propagation Delay)는 전파 지연 시간을, (Propagation delay time from High to Low)은 출력이 입력 신호의 영향을 받아서 논리 1에서 논리 0으로 바뀌는데 걸리는 시간을 의미하고, (Propagation delay time from Low to High)는 출력이 입력 신호에 영향을 받아서 논리 0에서 논리 1로 바뀌는데 걸리는 시간을 나타낸다.

**5.**

Verilog에서의 task와 function은 코드의 재사용성을 높이고 module화 하는 것을 높이기 위해 기능을 제공하는 block이다. 따라서 한 프로젝트에 동일한 작업이 많다면 task와 function을 사용하여 코드를 더욱 간결하게 바꿀 수 있다. Verilog의 task와 function은 always나 initial 구문을 사용할 수 없고, behavioral 구문만을 사용해야 하며 또한 다른 변수들은 사용할 수 있으나 wire 변수는 사용하지 못한다는 공통점이 있다. Task와 function의 차이점은 다음과 같다. 먼저 task의 특징부터 서술하자면, task는 또 다른 task와 function을 호출할 수 있으며 delay, timing, event같은 제어 구문을 가질 수 있다. 또한 시작할 때 task~endtask으로 시작과 끝을 명시하며 선언할 수 있고 여러 개의 인수를 가지거나 혹은 하나도 없을 수도 있다. Non-zero simulation time에 실행될 수 있으며 시간 지연 기능을 사용하지 않는 경우에는 synthesis도 가능하다. Task는 표현식 내부에서는 사용할 수 없지만 명령문으로는 호출이 가능하다. 반면에 function은 또 다른 task는 호출을 할 수 없고, 다른 function만 호출이 가능하다. 선언할 때는 function~endfunction으로 시작과 끝을 명시할 수 있고 delay, timing, event와 같은 제어구문은 사용할 수 없다. 인수로는 최소 1개 이상의 input인수를 가져야 하고 output이나 inout인수를 가질 수 있는 task와는 달리 input인수만 가질 수 있다. 또한 단 1개의 반환값만을 가지는데 return문을 사용하여 값을 반환해야 하고, function은 항상 zero-simulation time에 실행된다. Function은 합성이 가능하여 조합 회로를 모델링 할 때에 사용할 수 있고 function 내부에서는 function만 호출 가능하다.

다음은 task와 function의 코드 기본 형태이다. 먼저 task의 기본 형태는 아래와 같다.

task (task 이름);

(parameter선언)

begin

(구문 또는 NULL)

end

endtask

Function의 코드 기본 형태는 다음과 같다.

function (범위 또는 type) ? (함수 이름);

(parameter선언)

begin

(구문)

end

endfunction