3주차 예비보고서

전공: 아트엔테크놀로지 학년: 3학년 학번: 20191172 이름: 함승우

1.

논리 게이트에서 숫자 1은 전류가 흐르고 있음을 뜻하며, true를 뜻한다. 논리 게이트에서 숫자 0은 전류가 흐르고 있지 않음과 false를 뜻한다.

Transistor-Level은 트랜지스터의 설계를 통해서 그릴 수 있다. 트랜지스터는 전자 신호 및 전력을 스위칭 또는 증폭하는 데 사용되는 반도체 장치이다. 외부 회로에 연결하기 위해 기본적으로 3개의 단자가 있는 반도체 재료로 구성된다. 트랜지스터는 켜지는 신호에 따라서 다양한 종류가 있지만, 그리기 전에 NPN 트랜지스터로 통일하겠다.

첫번째로 AND 게이트를 그리자면,

그림, 스케치, 친필, 일러스트레이션이(가) 표시된 사진

자동 생성된 설명

트랜지스터가 직렬로 연결되어 있고, A와 B에 연결되어 출력한다.

두번째, OR 게이트를 그리자면,

스케치, 그림, 친필, 라인 아트이(가) 표시된 사진

자동 생성된 설명

트랜지스터가 병렬로 연결되었다.

마지막으로 NOT gate는

친필, 상징, 흑백이(가) 표시된 사진

자동 생성된 설명

트렌지스터의 베이스에 전류가 흐르면 출력 단자는 전류가 흐르지 않게 된다.

2.

첫번째로 AND 게이트에서는 2개 이상의 입력 값을 받고 하나의 출력 값을 반환해주는 게이트이다. 입력한 값이 모두 0인 경우 0을 출력하고, 입력 값 중 0이 하나라도 존재할 경우 0을 출력한다. 두 입력 값이 모두 1일 경우에만 1을 출력한다.

Transistor-Level에서는 전류가 흐를 때 1, 흐르지 않을 때 0이라고 한다. Transistor가 직렬로 연결되어 있어서 직렬 연결의 특성에 따라서 Transistor가 모두 전류가 흘러야 output에 1이 나타나게 된다.

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

두개의 입력이 있다고 가정할 때의 진리표를 위와 같이 작성할 수 있다. Boolean 식에서 곱셈으로 표현할 수 있다.

두번째로 OR 게이트에서도 역시 2개 이상의 입력 값을 받고 하나의 출력 값을 반환해준다. 입력한 값이 모두 0이라면 결과값으로 0을 출력하고, 입력값이 모두 1일 경우 결과적으로 1을 출력한다. 입력 값 중 1이 하나라도 존재하는 경우에도 결과적으로 1을 출력한다.

Transistor가 병렬로 연결되어 있어서 위 그림의 두 개의 Transistor중 하나라도 전류가 흐르는 경우에 output이 1이 나타나게 된다.

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

두 개의 입력값에서 위 진리표를 작성할 수 있다. 위 같은 특성으로 Boolean 식에서 덧셈으로 표현할 수 있다.

마지막으로 NOT 게이트는 입력 값을 하나만 받고 그와 반대는 신호를 출력값으로 반환해준다.

입력 값이 0이라면, 1을, 1이라면 0으로 반환해준다. 위 1번에서 트래지스터의 스위칭 기능을 역으로 이용해 트랜지스터의 베이스에 전류가 흐르게 되면 출력 단자에 전류가 흐르지 않게 되고, 베이스에 전류가 흐르지 않는다면, 출력 단자에 전류가 흐르게 된다.

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

3.Fan-out이란 논리 회로에서 하나의 논리 게이트의 출력이 얼마나 많은 논리 게이트 입력으로 사용되었는지 나타내는 말이다. 팬 아웃이 크다는 말은 하나의 논리 게이트로부터 나온 출력이 많은 논리 게이트의 입력으로 사용될 수 있다는 것을 말한다. 표준논리소자들은 1개의 출력신호에 접속할 수 있는 입력 신호의 수에 제한이 있다. 어떤 소자의 팬 아웃이라고 함은 그 소자의 출력에 연결할 수 있는 소자의 최대 개수이다. 즉 제한이다. 만약 Fan-out 이상의 소자를 연결하면 정확한 신호 연결이 되지 않을 수도 있고, 회로 자체가 망가질 수 있다. 한 소자를 거쳐 보낼 수 있는 전류량에는 한계가 있는데, 그 소자의 출력에 여러 개의 입력을 연결하면 전력을 입력의 개수에 비례해서 전력을 많이 사용하게 된다. 이러한 문제점으로 팬 아웃이 설정되어 있다.

4. 전파 지연이란 신호 값의 변화가 입력에서 출력까지 전달되는데 걸리는 평균시간을 의미한다.

전파 지연 시간에는 두 가지가 있는데 하나는 tPLH(propagation delay time from low to high)(입력 신호에 반응하여 출력이 0에서 1로 변화하는 데 걸리는 시간), tPHL(propagation delay time from high to low)(입력 신호에 반응하여 출력이 1에서 0으로 변화하는 데 걸리는 시간)으로 총 두개가 있다. 위 두개의 평균값을 보통 전파 지연시간으로 칭한다. 회로를 구성하는 게이트의 개수가 늘어날수록 전파 지연시간이 증가한다. 현실적으로 보았을 때 논리 회로의 전파 지연을 아예 없앨 수는 없으나 전파 지연을 잘 고려해서 회로를 설계해야 한다. 만약 전파 지연이 일관되지 않는다면 데이터 에러가 발생할 수 있다.

5. Verilog의 task는 코드의 반복을 줄이기 위해 사용된다. 다른 문법과 비슷하게도 task와 endtask로 감싸서 사용한다는 것을 알 수 있다. 이 task의 내부에서는 다른 task와 function도 사용 가능하다. 또한 Verilog의 HDL의 시간 지연과 관련된 기능을 포함시킬 수 있다.

non-zero 시뮬레이션 time에도 실행가능하며, input과 output을 가질 수 있다.

task를 사용하는 예시는 이렇다.

*task task\_name;*

*input …;*

*output y…*

*begin*

*…*

*end*

*endtask*

이렇게 task 구문을 사용하고 후에 task를 호출할 때

*task\_name (…)*

이런 식으로 호출하게 된다.

0개 이상의 input, output, inout의 인수를 포함하고, 값을 return하지 않을 수 있다, 또한 output과 inout의 특성을 살려 다수개의 값을 return할 수 있다.

Verilog의 function은 코드의 반복을 줄인다는 점에서 task와 비슷하지만, 약간의 차이가 있다. 우선 사용하는 방법은 module을 선언하는 것과 비슷하게 function과 endfunction으로 감싸서 사용한다. task와 다르게 반드시 1개의 output이 있어야 하고, 시간 지연과 관련된 #delay, posedge clk, negedge clk 등 이런 기능을 포함시킬 수 없다. 또한 내부에 function만 호출 가능하다는 점에서 task와 다르다.

function을 사용하는 예시는 이렇다,

*function func\_name;*

*input…;*

*…*

*begin*

*func\_name = …;*

*end*

*endfunction*

또한, 함수(function)은 nonblocking할당을 가질 수 없고, task도 호출할 수 없다는 점에서 task와 다르다. task와 function을 각각 이해해 알맞은 상황에 사용할 수 있어야 한다.