컴퓨터공학실험2 3주차 예비 보고서

전공: 컴퓨터공학 학년: 2학년 학번: 20201635 이름: 전찬

**1. 목적**

논리 게이트 AND/OR/NOT의 구조를 Transistor-Level로 표현해본다. AND/OR/NOT logic을 파악한다. Fan-out에 대해 조사한다. 전파 지연을 이해하며 Verilog의 task와 function을 알아본다.

**2. AND/OR/NOT의 logic과 Transistor-Level에서의 표현**

logic operation 이란 True, False두 가지 상태에 대한 연산이다. 대표적으로 and, or, not이 기본적인 논리 연산에 해당되며 이를 통해 다양한 연산을 만들어낼 수 있다. 추가적으로 회로에서 NAND, NOR를 통해 AND, OR, NOT 게이트를 표현할 수 있기 때문에 두 게이트를 범용 게이트(Universal Gates) 라고 하기도 한다.

첫 번째로 논리 연산자 AND는 두 개 이상의 input이 모두 참(True)일 때 참(True)를 output으로 가지는 연산자이다. 이 이외의 경우에는 거짓(False)을 output으로 가진다.

두 번째로 논리 연산자 OR은 두 개 이상의 input 중 하나 이상의 input이 참(True)라면 참(True)을 output으로 갖는 연산자이다.

마지막으로 논리 연산자 NOT은 위와 다르게 하나의 input을 받는다. 이 input이 참(True)이라면 거짓(False)을, 거짓(False)이라면 참(True)을 output으로 갖는 연산자이다. 추가적으로 회로에서 NOT 게이트를 Inverter이라고 하기도 한다.

**3. 전파 지연**

전파 지연이란 논리 회로가 입력신호를 받은 이후 출력 결과가 변화하는 데에 걸리는 시간을 의미한다. 예를 들어, NOT 게이트에서 입력신호가 0->1로 바뀌었을 때 이에 따라 출력 신호도 1->0으로 변화할 것이다. 이때 입력신호가 0->1로 바뀌었을 때부터 출력 신호가 1->0으로 바뀔 때까지의 딜레이가 생길 것인데, 이를 전파 지연이라고 할 수 있다. 하지만 어떤 논리 회로는 출력이 1->0으로 바뀌는 시간과 0->1로 바뀌는 시간이 다를 수 있다. 따라서 실제로 전파 지연 시간 = {tPLH(입력이 바뀐 시간부터 출력이 0에서 1로 바뀌는 지연 시간) + tPHL(입력이 바뀐 시간부터 출력이 1에서 0으로 바뀌는 지연 시간)}/2 로 정의한다.

**4. Fan-out**

Fan-out은 간단히 말해서 성능을 위해 input에 제한을 두는 것을 말한다. ‘성능’이란 말을 풀어보면 Quality(정확도)에서의 성능과 Time(시간)에서의 성능이라고 할 수 있다.

우선 성능에서 Quality란 Input과 회로를 거쳐 Output이 적절하게 발생하는지 판단하는 것이다. Input의 개수가 늘어남에 따라 Quality를 저해하는 다양한 상황이 발생할 수 있다. 그중 대표적인 두가지가 부하와 전압강하이다. 우선 첫 번째로 Input이 너무 많게 되면, 그에 따라서 많은 전류가 흐르며 전자 회로에 부하를 일으킬 것이다. 이는 전자 회로에 안좋은 영향을 미칠 수 있다. 두 번째로 모든 전선은 효율이 100%가 아니기 때문에 전압 강하가 발생하게 되는데, 이에 따라서 1(High)였던 신호가 0(Low)로 변하는 상황이 존재할 것이다. Input이 많아지게 된다면, 전압 강하의 빈도도 증가할 것이다. 이에 따라서 회로의 출력에 문제가 발생할 것이다. 따라서 Input의 수를 조절해야 한다.

Time에서의 성능은 위에서 설명한 전파 지연과 관련이 있다. Input이 증가함에 따라서 전파 지연도 더욱 크게 발생할 것이며, 이 또한 여러 문제를 야기할 수 있다.

위와 같은 이유로 적절한 Input의 수를 유지해주어야 한다.

(fan-out 식 설명하기)

**5. Verilog의 task와 function**

C 언어에서 macro 와 function을 통해서 재사용 가능한, 한 기능을 수행하는 함수를 정의할 수 있는 것처럼, Verilog에서는 task와 function을 통해서 한 기능을 수행하는 함수를 정의할 수 있다.

우선 task는 module을 정의할 때와 비슷하게 task ~~ endtask 형태로 정의한다. task에는 여러가지 특징이 존재하는데, output을 여러 개 가질 수 있다는 점이 존재한다.

function을 정의할 때도 비슷하게 function ~~ endfunction 형태로 정의할 수 있다. function은 output을 하나만 가질 수 있다.