3주차 예비보고서

전공: 컴퓨터공학과 학년: 3학년 학번: 20211547 이름: 신지원

**1.**

도표, 라인, 폰트, 평면도이(가) 표시된 사진

자동 생성된 설명도표, 폰트, 라인, 디자인이(가) 표시된 사진

자동 생성된 설명

순서대로 AND, OR, NOT Gate 의 Transistor-Level 을 그렸다.

**2.**

- AND Logic

AND 는 논리곱을 의미하며 따라서 입력이 모두 1이 되어야 출력도 1이 나올 수 있다. 시퀀스 회로에서 직렬을 의미하며 값을 표로 나타내면 그림 1-1과 같다.

라인, 텍스트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

<그림1-1. AND Logic 의 진리표>

AND logic 을 Gate를 사용하여 표기할 때 1-2와 같은 기호를 사용한다.

도표, 라인, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

<그림1-2. AND Gate의 기호>

그림 2-1의 결과를 이해하여 그림 2-2의 기호를 사용하여 펄스 입력에 따른 OR 게이트의 출력과 각 입력에 따른 출력을 기호로 나타내면 그림 2-3와 같다.

- OR Logic

OR 는 논리합을 의미하며 따라서 입력이 하나라도 1이 되면 출력은 1이 나올 수 있다. 시퀀스 회로에서 병렬을 의미하며 값을 표로 나타내면 그림 2-1과 같다.

라인, 번호, 텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

<그림2-1. OR Logic 의 진리표>

OR logic 을 Gate를 사용하여 표기할 때 2-2와 같은 기호를 사용한다.

도표, 스케치, 라인, 기술 도면이(가) 표시된 사진

자동 생성된 설명

<그림2-2. . OR Gate의 기호>

그림 2-1의 결과를 이해하여 그림 2-2의 기호를 사용하여 펄스 입력에 따른 OR 게이트의 출력과 각 입력에 따른 출력을 기호로 나타내면 그림 2-3와 같다.

도표, 라인, 그래프이(가) 표시된 사진

자동 생성된 설명

<그림2-3. OR Logic 의 출력>

- NOT Logic

NOT은 입력이 1이면 출력은 0이되고, 입력이 0이면 출력은 1이 되며 반대의 값을 출력한다고 생각할 수 있다. 따라서 보수기라고 불리며 시퀀스회로에서는 B접점을 표현한다.

라인, 스크린샷, 텍스트, 번호이(가) 표시된 사진

자동 생성된 설명

<그림3-1. NOT Logic 의 진리표>

NOT logic 을 Gate를 사용하여 표기할 때 3-2와 같은 기호를 사용한다.

도표, 라인, 스케치, 종이접기이(가) 표시된 사진

자동 생성된 설명

<그림3-2. NOT Gate의 기호>

**3.**

Fan-out 이란 논리 회로에서 하나의 논리 게이트의 출력이 얼마나 많은 논리 게이트의 입력으로 사용되는지 서술할 때 쓰인다.(출처, 위키백과) Fan-out 을 서술하는 이유는, 우선 입력신호의 수를 제한하기 위하여 사용한다. Fan-out 을 초과하여 접속하면 출력전류가 지나치게 커지게 되어 회로가 손상될 수 있다. 또한 전압강하 때문에 신호가 도달하지 못하여 신호의 논리 상태를 보장할 수 없게된다. 따라서 입력신호의 수를 제한하여 위와 같은 상황을 방지한다. 그 다음 이유에는 시간을 빠르게 하기 위해서 사용한다. Input단이 늘어날수록 propagation delay 가 커지게 된다. 따라서 출력, 대기 신호를 늦추게 되고 시스템의 오작동을 유발할 수 있기 때문에 사용한다.

위와 같은 상황을 방지하기 위해서 사용한다면, Fan-out 을 적절하게 지정하는 것이 중요할 것이다. Fan-out 은 아래와 같은 식을 사용하여 결정된다. 입력신호와 출력신호를 통하여 결정되는 것이다.

텍스트, 폰트, 화이트, 라인이(가) 표시된 사진

자동 생성된 설명

<Fan-out 을 결정하는 식>

Fan-out 이 크다는 것은 하나의 출력이 많은 논리게이트의 입력으로 사용된다는 뜻이다. 따라서 Fan-out이 지나치게 크다면 아래 사진과 같이 버퍼나 NOT 게이트 두 개를 연결하여 해결할 수도 있다.

도표, 라인, 기술 도면, 스케치이(가) 표시된 사진

자동 생성된 설명

<Fan-out이 초과되었을 때>

**4.**

전파지연이란 전기 회로에서 신호가 회로 속을 전파할 때 생기는 지연 시간을 의미한다. 신호값이 변화할 때, 입력에서 출력까지 전달되는 데 걸리는 시간을 나타내며 출력의 상승, 하강에 대한 각각의 입력에서 상승 지연 시간과 하강 지연 시간 사이의 평균값을 구하여 전파 지연 시간을 구할 수 있다. 자세한 식은 아래와 같다.

폰트, 친필, 화이트, 라인이(가) 표시된 사진

자동 생성된 설명

<전파지연 시간을 결정하는 식>

구체적인 예시를 통해 살펴보자. 아래 그림에 의하면 input과 output 의 흐름이 있을 때 input이 들어온다고 그 값이 output 에 바로 반영되지 않는다. Input이 있다고 하더라도 output 에 반영되는 데 살짝의 텀이 있는데 이를 전파지연이라고 말한다.

도표, 라인, 스케치, 평면도이(가) 표시된 사진

자동 생성된 설명

<전파지연을 위한 예시>

그렇다면 Verilog에서 전파지연은 어떻게 될까? Verilog에서 전파지연은 특정 경로의 wire의 길이가 다른 wire 보다 많이 길어 전기신호가 늦게 도착하는 경우 발생하며 아래와 같이 선언한다.

폰트, 텍스트, 화이트, 그래픽이(가) 표시된 사진

자동 생성된 설명

<Verilog에서 전파지연 표기 예시>

위처럼 표기하면 작은신호가 오더라도 관계없이 데이터에 전달되기 때문에 전파지연을 대비할 수 있다. 다른 상황에서는 Non-blocking 을 사용하여 전파지연에 관계없이 작업을 수행하도록 지시하여 대비할 수도 있다.

**5.**

task 와 function 은 ‘서브루틴’ 의 종류로 코드의 재사용과 가독성 향상을 위하여 사용한다. 먼저 Task 는 절차적인 코드를 묶어 관련 작업을 수행하는데 사용되며, 입력 매개변수를 받아 작업에 사용하는 데이터를 전달할 수 있다. Task 내에서는 비동기식 블록을 사용할 수 있어 fork 및 join을 사용하여 병렬 작업을 수행할 수 있다. 주요 특징에는 내부에 다른 task와 function을 사용할 수 있고, non-zero 시뮬레이션 시간에 수행될 수 있으며delay, 타이밍 제어 문장 등을 포함할 수 있다. 또한 입출력의 수에 상관없지만 값을 되돌릴 수는 없다. 구조는 ‘taks- endtask’ 형태로 이루어진다.

**task adder(input a, b, output sum);**

**sum = a + b;**

**endtask**

**//만약 같은 task 를 동시에 호출할 때는 automatic 을 사용한다.**

**task automatic bitwise\_xor(**

**output [15:0] ab\_xor,**

**input [15:0] a, b**

**);**

**begin**

**#delay ab\_and = a & b;**

**ab\_or = a | b;**

**ab\_xor = a ^ b;**

**end**

**endtask**

function의 특징에는 다른 function을 사용할 수 있지만, 다른 task는 사용 할 수 없다는 것이 있다.  
또한 시뮬레이션 시간이 0일때 수행되며 task 와 달리 어떠한 delay, 타이밍 제어 문장 등을 포함할 수 없다. 입출력에 대한 제한도 가지고 있는데, 적어도 하나 이상의 input을 가져야 한다. 구조는 마찬가지로 ‘function – endfuntion’ 로 구성되어 있다.

**module shifter; //Shift 를 사용하여 모듈을 정의한다.**

**…**

**function int multiplier(input a, b);**

**multiplier = a \* b;**

**endfunction**

**6. 참고자료**

https://ko.wikipedia.org/wiki/AND\_게이트

https://ko.wikipedia.org/wiki/팬\_아웃

http://word.tta.or.kr/dictionary/dictionaryView.do?subject=전파+지연