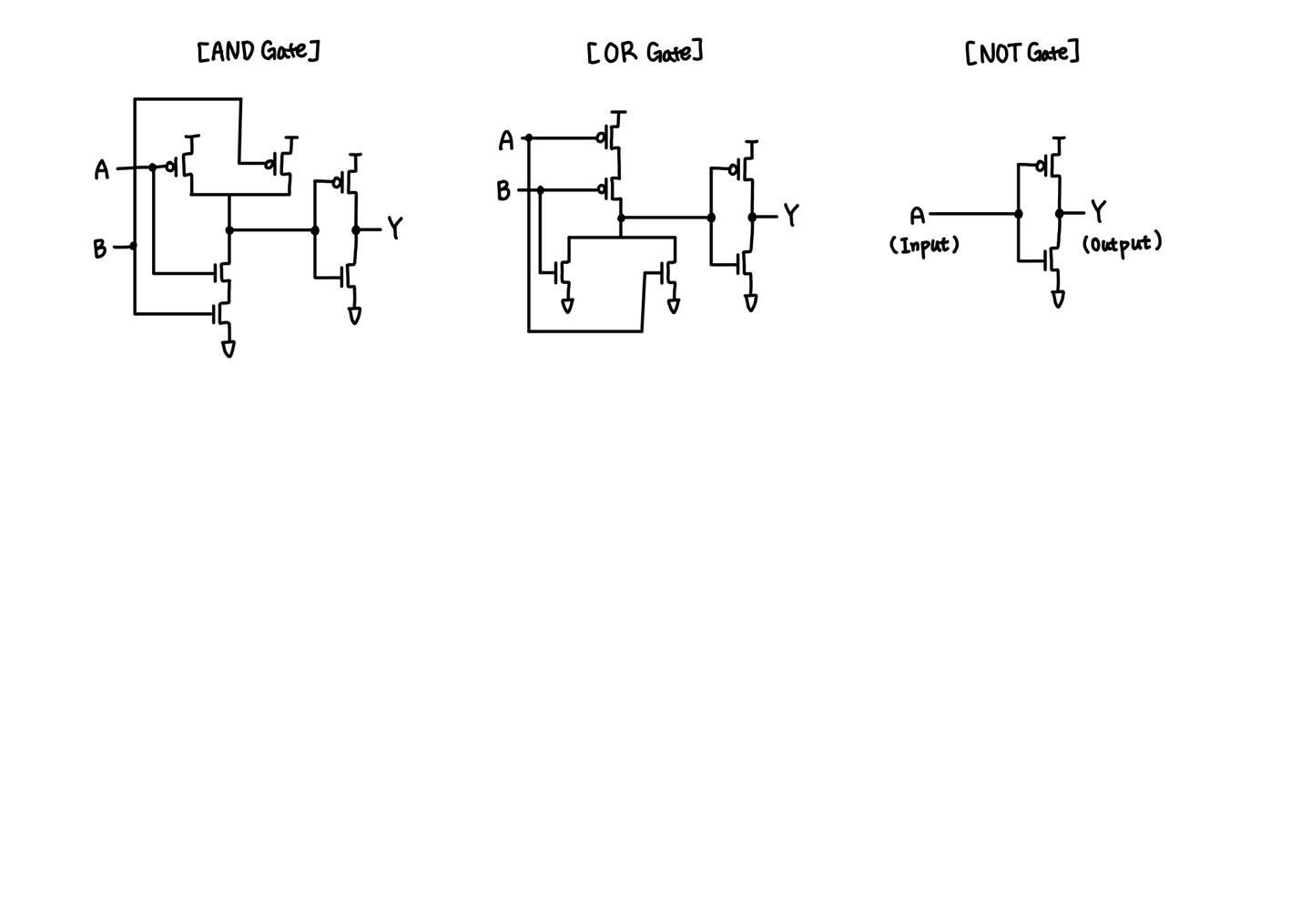
3주차 예비보고서

전공: 아트&테크놀로지학과 학년: 4학년 학번: 20191048 이름: 김도솔

**1.**



**2.**

**1) AND**

2개 이상의 입력을 받으며 1개의 출력을 반환한다. 입력 값이 모두 1인 경우 1을 출력하고 그 외에는 0을 출력한다. Boolean 식에서 곱셈으로 표현된다. 2-Input AND Gate의 진리표는 다음과 같다.

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

**2) OR**

2개 이상의 입력을 받으며 1개의 출력을 반환한다. 입력 값 중 하나라도 1인 경우 1을 출력하고 입력 값에 1이 없는 경우, 즉 모든 입력 값이 0인 경우 0을 출력한다. Boolean 식에서 덧셈으로 표현된다. 2-Input OR Gate의 진리표는 다음과 같다.

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

**3) NOT**

1개의 입력을 받고 1개의 출력을 반환한다. 즉 입력 값이 0인 경우 1을 출력하고, 입력 값이 1인 경우 0을 출력한다. 입력값을 반전시켜 출력하는 특성이 있어 Inverter라고도 부른다. NOT Gate의 진리표는 다음과 같다.

|  |  |
| --- | --- |
| Input | Output |
| 0 | 1 |
| 1 | 0 |

**3.**

**Fan-out**은 디지털 논리 회로에서 사용되는 용어로, 특정 출력 신호가 하나 이상의 입력으로 전달되는 횟수를 나타낸다. 즉, 특정 출력이 몇 개의 입력으로 들어가는지를 나타내는 개념이다. Fan-out이 크다는 것은 한 출력이 다수의 논리 게이트에 입력으로 사용된다는 것을 의미한다. Fan-out이 높아질수록 신호가 중간에 노이즈나 간섭을 받을 가능성이 커지며 전체 회로의 지연 시간이 늘어날 수 있다. 따라서 너무 큰 Fan-out은 회로의 성능을 저하시키거나 신호 전달의 불안정성을 초래할 수 있으므로 이를 관리할 필요가 있다. 하나의 출력 신호를 두 개 이상으로 분할하여 처리하거나 논리 회로를 트리 구조로 설계하는 등의 방식으로 Fan-out을 줄일 수 있다. Fan-out 관리는 디지털 논리 회로 설계의 중요한 측면 중 하나이며, 성능 및 안정성을 유지하는 데 핵심적인 역할을 한다.

**4.**

**전파 지연(Propagation Delay)**은 디지털 논리 회로에서 입력이 변경된 후 출력이 해당 변화를 반영하기까지 걸리는 시간을 말한다. 즉 신호가 논리 게이트와 전선을 통해 전파되는 데 걸리는 시간을 측정한 것이다. 전파 지연의 발생 원인으로는 논리 게이트 지연, 전선 전파 지연 등이 있다. 먼저 각 논리 게이트는 입력 신호를 처리하고 출력을 계산하는 동안 일정한 시간이 걸리는데, 이를 논리 게이트의 전파 지연이라고 한다. 다양한 게이트 유형(AND, OR, NOT 등) 및 기술(CMOS, TTL 등)에 따라 전파 지연 정도가 달라질 수 있다. 다음으로 전선 전파 지연은 전선에서 발생하는 전파 지연을 말한다. 신호는 전선을 따라 전파되며, 전선의 길이 및 소재에 따라 전파 속도가 다르다. 따라서 더 긴 전선은 더 많은 전파 지연을 유발할 수 있다. 전파 지연은 디지털 논리 회로의 동작과 성능에 큰 영향을 미친다. 논리 회로가 더 짧은 전파 지연을 가질수록 입력 변경에 대한 출력 반응이 더 빨라진다는 점에서 이를 알 수 있다. 따라서 디지털 회로 설계 시 성능 최적화를 위해 논리 게이트 및 전선의 선택, 논리 회로 배치 등이 고려되어야 한다.

**5.**

**Verilog의 task와 function**은 C 언어의 함수와 유사한 개념으로, 코드의 재사용성을 높이고 코드를 간결하게 만드는 데 사용된다. 이러한 코드 구성 요소는 큰 디지털 회로의 설계에서 특히 유용하며, 프로젝트 내에서 코드의 가독성을 향상시키고 코드의 재사용성을 높이는 데 도움이 된다.

1. **task:** Verilog에서 task는 모듈 내에서 언제든지 호출 가능한 서브루틴 키워드이다. task는 자신이 정의된 모듈 안에서 호출될 수 있을 뿐만 아니라, 다른 모듈에서도 호출될 수 있다. task를 사용할 때는 module을 선언하는 것과 동일하게 task와 endtask로 감싸서 사용한다. 세부 특징은 다음과 같다.

* 시간 지연과 연관된 기능(ex. delay, posedge clk 등)을 포함할 수 있다.
* 시간 지연기능을 사용하지 않는 경우, 합성(synthesis)이 가능하다.
* 하나 이상의 input(입력), output(출력), inout(입출력) 등의 인수를 가지거나 인수를 하나도 가지지 않을 수도 있다. return value를 가질 수 있고, output(출력), inout(입출력)의 인수를 통해 다수의 값을 전달할 수 있다.
* 내부에서 task와 function 호출이 가능하다.
* non-zero 시뮬레이션 타임에 실행된다.

|  |
| --- |
| 예시 코드 |
| task adder;  input a, b;  output sum;  begin  sum = a + b;  end  endtask |

1. **function:** Verilog에서 function은 task와 비슷하지만 약간의 차이점이 존재한다. 사용할 때는 function과 endfunction으로 감싸서 사용한다. 세부 특징은 다음과 같다.

* 시간 지연과 연관된 기능을 포함할 수 없다.
* 합성이 가능해 조합 회로 모델링에 사용될 수 있다.
* 적어도 하나 이상의 input(입력) 인수를 가져야 하며, output(출력), inout(입출력)의 인수를 가질 수 없다. 자신의 이름으로된 하나의 return value만 가진다.
* 내부에서 다른 function을 호출할 수 있지만, task 호출은 허용되지 않는다.
* 항상 0 시뮬레이션 타임에 실행된다.

|  |
| --- |
| 예시 코드 |
| function adder;  input a, b;  begin  adder = a + b;  end  endfunction |