3주차 예비보고서

전공: 컴퓨터공학과 학년: 3학년 학번: 20191599 이름: 송경호

**1.**

**1-1) AND**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Out** |
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

도표, 그림, 원, 스케치이(가) 표시된 사진

자동 생성된 설명

**1-2) OR**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Out** |
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

도표, 그림, 원, 스케치이(가) 표시된 사진

자동 생성된 설명

**1-3) NOT**

|  |  |
| --- | --- |
| **A** | **Out** |
| 1 | 0 |
| 0 | 1 |

상징, 도표, 원, 흑백이(가) 표시된 사진

자동 생성된 설명

**2.**

**2-1) AND**

AND는 논리곱을 의미한다. 두 입력이 모두 참일 경우에만 출력이 참이 된다. 즉, AND 게이트는 입력 전압이 모두 High인 경우에만 출력 역시 High가 된다. A, B가 입력일 때, 논리식은 AxB 혹은 A•B로 표현된ㄷ다.

**2-2) OR**

OR은 논리합을 의미한다. 두 입력이 모두 거짓일 경우에만 출력이 거짓이 된다. 즉, OR 게이트는 입력 전압이 하나라도 High인 경우 출력 역시 High가 된다. A, B가 입력일 때, 논리식은 A+B로 표현된다.

**2-3) NOT**

NOT은 부정을 논리부정을 의미한다. 입력이 참인 경우 출력은 거짓이, 입력이 거짓인 경우 출력이 참이 된다. 즉, NOT 게이트는 입력된 신호에 대한 반전 신호를 출력으로 내보낸다. A가 입력일 때 논리식은 ~A로 표현된다.

**3.**

Fan-out이란, 디지털 논리 회로를 구성할 때 중요한 개념으로, 하나의 출력신호에 대하여 다음 logic의 input으로 들어갈 때, 접속 가능한 입력의 수에 제한을 거는 것을 일컫는다. Fan-out 지정을 통해 입력의 수를 제한하는 이유는 크게 두가지로 Signal Quality와 Timing 때문이다.

먼저, Fan-out의 제한을 통해 입력 신호의 품질을 보장할 수 있다. 이상적인 논리 게이트의 경우 원하는 수의 입력과 출력이 가능하겠지만 현실의 논리 게이트에는 출력에 흐를 수 있든 최대 전류의 한계가 존재한다. 따라서 fan-out을 통해 입력의 수에 제한을 걸지 않고 많은 수의 입력을 연결하면 출력 전류의 한계를 넘어 회로가 손상될 수 있다. 또한 입력 신호가 한계를 넘어서면 출력이 불가능하거나 로직 레벨이 감소하는 경우가 발생하여 예상과 다른 동작이 발생할 수 있다.

다음으로, fan-out의 제한은 시간적인 측면에서 중요하다. 논리 게이트의 입력에는 저항과 용량이 존재하며, 이러한 용량은 전파 지연을 발생시킨다. 이때, 입력의 수가 증가하면 용량 역시 증가하므로 전파 지연이 더욱 커져 시스템의 동작 속도에 큰 영향을 미친다. 따라서 원치 않는 동작이 발생하는 것을 막고자 fan-out을 통해 입력의 수를 제한한다.

**4.**

전파 지연이란 입력 신호가 논리 게이트를 거쳐 출력 결과로 나타날 때까지 걸리는 시간을 의미한다. 이는 논리 회로의 동작 속도를 결정하는 핵심적인 지표이다.

전파 지연은 논리 게이트의 복잡성, 구성 요소의 전파 속도 등 다양한 요인에 따라 달라지게 된다. 일반적으로 전파 지연(T\_PD)는 입력 신호가 1에서 0으로 변하는 데 걸리는 시간인 T\_PHL과 0에서 1로 변하는 데 걸리는 시간인 T\_PLH의 평균으로 나타낸다. **T\_PD = (T\_PHL + T\_PLH) / 2**

전파 지연은 논리 회로가 얼마나 빠르게 동작할 수 있는지에 대한 직접적인 영향을 미치는 만큼 논리 회로 설계 시 전파 지연을 최소화해야 한다. 또한 여러 개의 논리 게이트를 거치면 전파 지연이 누적될 수 있으므로 이를 고려하여 논리 회로를 효율적으로 최적화 해야 한다.

**5.**

Verilog에서는 코드의 반복 작업을 최소화하고 재사용성을 높이기 위해 프로그래밍 언어의 함수와 유사한 역할을 하는 **task**와 **function**을 사용한다.

Task와 function은 공통적으로 로컬 변수, 레지스터 등을 가질 수 있지만 net 자료형을 사용할 수 없다는 특징이 있다. 또한 always와 initial과 같은 구문을 사용할 수 없다.

Task와 function은 차이점 역시 존재한다.

먼저, 시간 지연의 측면에서 task 내에서는 시간 지연을 포함할 수 있으나 function은 포함할 수 없다. 따라서 function은 항상 0 시뮬레이션 타임에 실행된다. 다음으로 입출력 인수 개수의 측면에서 Task는 인수가 없을 수도 있는 반면 function은 적어도 하나 이상의 입력을 가져야 하며 하나의 출력 인수만을 가진다. 또한 task 내에서는 task와 function 모두를 호출할 수 있는 반면 function 내에서는 오직 function만을 호출할 수 있다. 마지막으로 function은 합성이 항상 가능하지만 task는 시간 지연을 사용하지 않는 경우에만 합성이 가능하다.