3주차 예비보고서

|  |
| --- |
| - FPGA란?  - HDL이란?  - Verilog의 Net, Reg 자료형  - AND/OR/NOT 구조 그림  - AND/OR/NOT Logic의 특성  - Fan-Out?  - 전파지연  - Verilog의 task와 function |

20141196 김성희

1. FPGA가 무엇인가? 왜 어떻게

- FPGA는 ‘Field Programmable Gate Array’의 약어로 현장에서 고칠 수 있는 gate array라는 뜻이다.

- ASIC에 비해 값이 싸며 고칠 수 있어서 ASIC이전에 프로토타입을 만들 때도 쓰인다.

- HDL이라는 언어를 사용하여 프로그래밍을 한다.

**2. HDL** Hardware Description Language **이 무엇이며 verilog외에 무엇이 있는가?**

- HDL은 두 가지 종류의 시스템을 설계하기 위해 만들어졌다.

ㅁ. 특정 직접회로를 설계하기 위해 사용된다. 이 경우 회로의 동작을 예측하는 모델을 제공한다.

ㅁ. FPGA같은 논리 회로를 기록하는 디바이스를 프로그램하기 위해 사용한다. 코드를 작성하고 컴파일을 한 후 기기에 올린다.

- HDL은 여러가지가 있지만 주로 Verilog와 VHDL을 사용한다.

ㅁ. verilog에 비해 VHDL은 문법제약이 강해서 학교, 연구실에서 많이 사용된다. 반면 Verilog는 문법제약이 약해서 유연하고 효율성이 높아 회사에서 주로 사용한다.

**3. Verilog의 Net 자료형과 Reg자료형**

- Reg 자료형

ㅁ. 절차적 할당(procedural assignment)으로 할당할 때 쓰인다. always문이나 if문에서 쓰인다.

ㅁ. 값을 잠깐 동안 보관할 수 있다.

ㅁ. 쉽게 말하자면 데이터를 ‘처리’할 때 쓰인다.

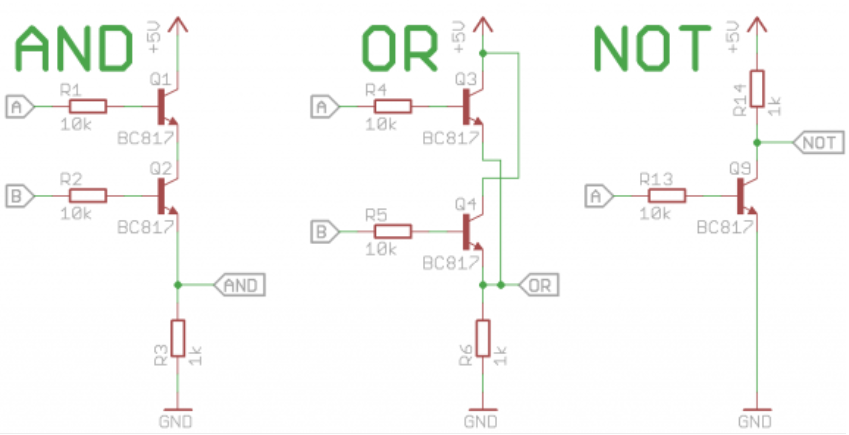
- Net 자료형

ㅁ. 논리 게이트에서 물리적 연결을 나타내기 위해 사용한다.

ㅁ. 연속적 할당(continuous assignment)으로 할당할 때 쓰인다. Assign을 쓸 때 사용한다. Wire y; reg x; wire z; assign y=x&z;

ㅁ. 쉽게 말하자면 데이터를 ‘전달’할 때 쓰인다. (주로 출력 값을 입력 값으로 전달할 때 쓰임.)

**4. 논리게이트 AND/OR/NOT 구조 (transistor-level)**



- A, B가 입력 값이며 이 출력 값이다.

**5. AND/OR/NOT Logic의 특성**

- AND: 입력 값이 2개다. 2개 모두 1인 경우 출력 값이 1이 된다. 그 외의 경우 출력 값이 0이다.

- OR: 입력 값이 2개다. 2개 모두 0인 경우 출력 값이 0이 된다. 그 외의 경우 출력 값이 1이다.

- NOT: 입력 값이 1개다. 입력 값의 반대의 경우를 출력 값으로 가진다. (0은 1로, 1은 0으로)

**6. Fan-out이 무엇인가?**

- 디지털 회로에서 널리 사용되는 표준논리소자들은 1개의 출력신호에 영향을 줄 수 있는 입력신호의 수가 정해져 있다. 이러한 제한을 fan-out이라 한다.

- 신호의 수가 정해져 있는 이유는 각 소자의 출력회로에 흐를 수 있는 전류가 제한돼 있기 때문이다.

**7. 전파지연**

- 회로에서 전류가 흐르는 시간을 말한다. 즉 위 AND게이트를 보면 입력 A에서 출발하여 transistor까지 그리고 transistor에서 출력까지 전류가 도착하는 시간들을 전파지연이라고 한다.

- 전파지연을 생각하지 않고 회로를 설계하다 보면 원하는 clock 주기에 데이터가 쓰여지지 않는 사태가 발생할 수 있어 유의해야 한다.

**8. Verilog의 task와 function**

- 둘 모두 c언어의 함수와 비슷하나 분명한 차이가 있다.

- **task**

ㅁ. task 내에서 다른 task와 function을 사용할 수 있다.

ㅁ. delay, timing, event 제어가 가능하다.

ㅁ. input 인수를 받지 않을 수 있다.

ㅁ. output 인수를 통해 출력 값을 전달한다. 출력 값이 없어도 되고 다수 존재해도 된다.

- **function**

ㅁ. function 내에서 다른 function을 사용할 수 있으나 task는 사용 불가하다.

ㅁ. delay, timing, event 제어가 불가능하다. (항상 시뮬레이션이 0에서 시작)

ㅁ. 하나 이상의 input 인수를 꼭 받아야 한다.

ㅁ. output 인수가 아닌 함수 이름을 통해서 출력 값을 받아야 한다. (output 인수가 없다.) 출력 값이 반드시 존재해야 하며 2개 이상은 출력 불가하다.

**사용법**

|  |  |
| --- | --- |
| task example;  input x, y;  output z;  begin  z = x&y;  end  endtask | function example;  input x, y;  begin  example = x&y;  // example을 call하면 x&y 값 반환  end  endfunction |