2주차 예비보고서

전공: 경영학과 학년: 4학년 학번: 20190808 이름: 방지혁

1.

HDL은 하드웨어 기술 언어로 Hardware Description Language의 약자이다. 디지털 회로와 혼합 신호를 표현하는데 사용됩니다. 다른 일반적인 프로그래밍 언어들과 다르게 소프트웨어를 프로그래밍하는 것이 아니라 HDL은 디지털 논리 회로를 설계자가 설계, 검증 및 simulation 하는데 사용됩니다. 또한, 컴파일에 차이가 존재합니다. 보통의 프로그래밍 언어의 컴파일 과정은 source, compile, 실행파일의 과정을 거치지만, HDL의 컴파일은 source, compile, simulation, synthesis, hardware의 과정을 거친다. HDL은 유연하고 수정하기 좋으며, 재사용성이 높고, 알고리즘을 다루는데 유리합니다.

많은 이들이 Verilog를 널리 사용하지만 여러 HDL들이 존재합니다. 우선 VHDL이 있다. VHDL은 정부 주도로 개발되었으며, Verilog에 비해 더 엄격하고 문법이 길어 복잡하지만, 오류를 예방하고 복잡한 시스템을 정확하게 설계하는 데 유리합니다.

또한, AHDL이 존재합니다. Alter에서 만든 HDL로 Altera의 Quartus 개발 환경에서 사용 가능하며 다른 FPGA 제조사에서 사용되지 않습니다. 단순히 회로의 상세 구현이 가능한 것뿐만 아니라, 더 높은 레벨에서 설계의 추상화가 가능합니다.

2.

1983년에 Gateway Design Automation 회사의 창업자인 Prabhu Goel가 Verilog를 만들면서 시작되었습니다. 기존에는 대부분의 디지털 회로 설계가 트랜지스터 수준에서 이루어졌었으며 설계자가 자동화하는 툴이 부족하였습니다. 이러한 과정을 자동화하고, 설계자가 고수준에서 회로를 설계하고 시뮬레이션을 통해 검증할 수 있도록 개발된 것입니다. 또한, Verilog의 형태가 C 언어와 비슷하여 사람들이 빨리 배울 수 있게 되었고, 널리 퍼지게 되었습니다.

1990년대에 들어 이 회사가 Cadence Design System이라는 회사에 인수되었고 VHDL과 함께 업계의 표준으로 자리매김하였습니다. 이후 사람들은 표준화의 필요성을 제기하기 시작하였습니다. 일관성과 상호 운용성이 보장되어야 했기 때문입니다. 그렇게 1995년 IEEE 표준으로 제정되었고 ASIC, FPGA 설계에서 중요하게 자리 잡았습니다.

또한, SystemVerilog가 등장하게 됩니다. 하드웨어 설계가 복잡해지면서 새로운 언어가 필요하자 verilog의 기능을 확장한 것입니다. 이는 설계 뿐만 아니라 하드웨어 검증에 필요한 고급기능들이 존재하였습니다. 전반적으로 정리하자면 Verilog는 최근의 반도체까지 오기까지 여러 기여를 했다는 것을 알 수 있습니다.

3.

Verilog에서는 우선 모듈이 기본적인 설계 단위입니다. 모듈 중에 메인이 되는 모듈을 TOP 모듈이라고 하며, 이는 C의 main 함수와 비슷하다고 보면 됩니다. 모듈은 머리부, 선언부, 몸체부로 나눌 수 있습니다. 처음으로 머리부에서는 모듈 이름을 설정하고, 포트 이름을 나열합니다. 마지막으로는 'endmodule'로 끝을 알려야 합니다. 다음으로 선언부에서는 포트의 방향, 비트의 폭, 레지스터(reg)나 매개변수들을 선언합니다. 이는 C언어에서 변수를 선언하는 것과 비슷하다고 볼 수 있습니다. 몸체부에선 회로의 동작을 표현하는 구문 등으로 구성되어 있습니다.

또한, 비트연산자, 논리연산자, 관계연산자, 등가연산자, 조건연산자, 결합연산자, 시프트연산자 이렇게 다양한 연산자를 가지고 있습니다. 비트연산자에는 ~(bitwise NOT), &(bitwise AND), |(bitwise OR), ^(bitwise XOR) 등이 있습니다. 논리연산자에는 &&(logical AND), ||(logical OR), !(negation)이 있고, 관계연산자에는 >, < 등이 있습니다. 등가연산자에는 ==,!= 가 있고, 조건 연산자에는 result?a:b 의 구문이 있습니다. 결합 연산자는 비트를 연결하거나 복제하는 연산자로 assign을 사용합니다. 시프트연산자에는 <<과 >>이 있습니다.

always 구문은 반복적으로 실행됩니다. @(sensitivity_list)에서 제어합니다. initial 구문은 나열된 순서대로 한 번만 실행됩니다. if 문은 여러 줄일 경우 begin과 end로 묶어야 하며, 반드시 always 문 내에서 사용되어야 합니다. 또한 case연산자도 반드시 always문 내에서 사용되어야 합니다. while문은 조건문이 false가 될 때까지 수행합니다.