2주차 예비보고서

전공: 컴퓨터공학과/생명과학과 학년: 9학기 학번: 20181435 이름: 박다희

**1.**

HDL은 Hardware Description Language로 컴퓨터 언어 중 하나이다. 일반적인 프로그래밍 언어인 C언어나 java등과 큰 차이점은 HDL은 시간적 개념을 가지고 있다는 점이고, 그렇기에 컴퓨터의 고급 언어에 디지털 시스템의 기능이나 하드웨어의 특징을 접합 시켜 회로를 원하는 방향으로 기술할 수 있도록 해준다. 따라서 HDL을 사용하면 디지털 논리 회로의 동작을 설명하고 시뮬레이션을 통한 검증 그리고 합성, 구현까지 가능하며 gate level에서 뿐만 아니라 구조적이나 동작 level에서도 하드웨어를 나타낼 수 있다.

Verilog이외의 HDL의 종류에는 VHDL이나 MyHDL, Chisel등이 있는데 먼저 VHDL은 VHSIC Hardware Description Language로 VHSIC의 하위 언어이다. 혼합 신호나 디지털 회로를 표현하는 언어로 추상화 수준이 뛰어나 복잡한 디지털 시스템을 구현하는 데 적합한 언어이다. 또한 VHDL은 Pascal언어와 Ada언어를 바탕으로 하고 있어서 C언어와 유사한 Verilog와는 차이점이 있다. 그 다음 MyHDL은 python을 바탕으로 한 언어로 python 언어를 사용하여 원하는 하드웨어를 설계할 수 있게 해준다. 그래서 가독성 있고 간결한 코드를 작성할 수 있다는 장점이 있다. 마지막으로 스칼라 언어를 사용하여 하드웨어를 설계하는 Chisel로 DSL(Domain-Specific Language)이다. 이 언어는 여러 종류의 하드웨어 시스템을 정의하는 데에 있어서 유연성을 제공하는 언어이다.

**2.**

Verilog의 역사와 그 발전 과정은 다음과 같다. 1980년대 초반에 기존의 회로 simulation 언어였던 C언어를 기반으로 한 C-based Simulator로 시작하여 회로의 동작을 보여주고 디버깅할 수 있게 도와주었다. 1985년엔 Gateway Design Automation에서 Verilog가 개발되기 시작하였고, 시뮬레이션 도구로써 초기 Verilog 상용 구현 중 하나였던 Verilog-XL이 출시되었다. 1990년엔 IEEE(미국 전기전자 기술 학회)에서 Verilog를 IEEE 1364표준으로 인정하였고, 이 후부터 표준화된 Verilog는 여러 분야의 업체에서 사용되었다. 1995년에는 디지털과 아날로그 시스템을 함께 설계하는데 사용될 수 있도록 디지털 회로와 아날로그 회로를 다 아우를 수 있는 기능을 추가한 Verilog-AMS(Analog and Mixed Signal)이 개발되었고 2000년대에 들어와서는 하드웨어를 설계하기 위한 언어로의 입지를 강화하며 단지 전자 회로를 시뮬레이션하는 것 뿐만 아니라 합성과 FPGA/ASIC디자인에도 사용하기 시작됐다. 여기에 SystemVerilog가 개발되어 Verilog를 더욱 더 보완하고 더 뛰어난 기능을 제공하도록 했다. 이러한 발전 과정을 거쳐 현재의 Verilog와 SystemVerilog는 ASIC과 FPGA의 설계적인 분야에서 매우 큰 역할을 하고 있으며 다양한 분야의 디자인과 시뮬레이션 도구에서 지원되고 있고 SystemVerilog가 2005년에 IEEE표준으로 만들어졌다.

**3.**

Verilog의 기본적인 구조와 문법은 다음과 같다. Verilog에서 기본 설계 단위는 module로 Verilog의 디자인은 하나 이상의 module로 이루어져 있다. 이 module을 통해 다른 module을 포함하는 계층적 구조를 구성할 수 있다. Verilog을 구성하는 기본 구조는 머리부, 선언부 그리고 몸체부로 크게 3가지로 나눌 수 있는데, 먼저 머리부는 ‘module (모듈이름) (port 목록);’으로 나타낼 수 있는데 키워드와 같은 이름은 사용할 수 없다. 마지막에는 endmodule를 작성하여 module의 끝임을 나타내야 한다. 선언부는 머리부에서 선언한 module에 필요한 것들을 선언하는 부분인데 C언어에서 변수를 선언하듯이 reg, port, 비트 폭, parameter등을 선언부에서 선언한다. 마지막으로 몸체부에서는 회로의 구조나 동작 그리고 기능을 구현하는 구문들로 이루어져 있다. 몸체부에 따라 논리합성용 구문인지 시뮬레이션용인지 라이브러리 설계용인지 구분할 수 있다. Verilog의 자료형에는 여러 종류가 있는데 크게 두 가지 종류로 나눌 수 있다. 하나는 추상적인 저장 장치를 나타내는 register 변수이고, 또 다른 하나는 디바이스 간의 물리적인 연결을 나타내는 net 변수이다. 먼저 register변수의 종류로는 절차형 할당문인 initial이나 always 등에 의해 값을 할당 받는 객체인 reg, 정수형 변수인 integer, 실수형 변수인 real 그리고 시간에 따라 값을 변경하고 싶은 상황처럼 시간의 흐름을 체크하여 처리해야 하는 역할인 time 또는 realtime 변수 등이 있다. 그 다음 net 변수에는 tri-state net의 경우에 선을 연결할 때 사용하는 tri 변수와 module안에서 선언한 변수들이 어떻게 연결되어 있는 지 나타내는 변수인 wire가 있다.

Verilog에서 상수를 표현할 때는 여러 방법이 있는데 먼저 일반적인 표현으로는 (bit 수)(n진수)(숫자)로 표현할 수 있다. 따라서 만약 bit수를 4비트로 지정한 경우에는 4bit 2진수 1111을 4’b1111로 표현할 수 있다. 이와 같이 12bit 16진수 abc를 나타내면 12’habc이다. 만약 bit 수를 지정하지 않은 경우 32bit 10진수 214를 표현할 때는 214로, 32bit 16진수 a2를 표현할 때는 ‘ha2로 표현하면 된다. 음수를 표현할 때는 2의 보수로 처리하여 저장되는데 예를 들어 3과 2의 보수로써 음수로 -6’d3으로 표현하면 된다. Verilog의 연산자 중 산술연산자는 +, -, /, \*, %로 각각 덧셈, 뺄셈, 나눗셈, 곱셈, 나머지연산자고 관계 연산자로는 >, <, >=, <=, ==, !=로 각각 크다, 작다, 크거나 같다, 작거나 같다, 같다, 다르다를 의미한다. 논리연산자로는 and의 &&, or의 ||, not의 !, 비트단위 not의 ~, 비트단위 and의 &, 비트단위 or의 |, 비트단위 xor의 ^, 비트단위 xnor의 ^~ 또는 ~^ 등이 있다. Shift 연산자로는 오른쪽으로 이동하는 >> 연산자와 왼쪽으로 이동하는 << 연산자가 있다. 예를 들어 0110>>1을 수행하면 0011의 결과가 나오고 1100<<1을 수행하면 결과는 1000이다. 결합 연산자로는 {}가, 반복 연산자로는 {{}}}가 있다.

Verilog에서는 또한 여러 가지 구문을 가지고 있는데 C언어와 유사한 if, for, while, case문 뿐만 아니라 timescale, always, assign, initial, Blocking, Non-Blocking 등의 구문도 가지고 있다. 먼저 timescale 구문은 사용할 때 ‘timescale (시간단위) / (정밀도)’로 나타내는데 코드에 구현하는 모든 시간 단위를 작성한 (시간 단위)로 지정한다는 의미이고 (정밀도)는 시간단위 관련해 소수점의 허용 범위를 나타낸다. Assign 구문은 net 자료형 변수에 값을 할당하는데 사용하고 assign 구문으로 할당한 변수의 영향을 없앨 때는 deassign 구문을 사용한다. Always 구문은 반복적으로 실행되는 구문으로 ‘always @ (sensitivity list)’로 선언하며 begin~end 안에 실행하고자 하는 구문을 적으면 된다. Sensitivity list는 always문의 실행을 컨트롤하는 역할을 한다. Initial구문은 딱 한 번 실행되며 사용할 때는 initial을 작성하고 begin~end안에 실행하고자 하는 구문을 작성하면 된다. Blocking구문은 always나 initial 구문의 begin~end구문을 실행하는데 계산하고 바로 저장까지 이루어지는 반면 non-blocking 구문은 begin-end안의 모든 구문을 다 계산을 한 뒤에 한꺼번에 저장이 이루어진다.