2주차 예비보고서

전공: 신문방송학과 학년: 4학년 학번: 20191150 이름: 전현길

1. HDL이 무엇인지 조사하고, Verilog 외의 HDL에 대하여 조사하시오.

HDL(Hardware Description Language)은 하드웨어를 기술하는 언어로, 전자 회로의 정밀한 동작, 구조를 설계하고 회로를 시뮬레이션하는 데서 사용되는 언어이다. 게이트의 수가 커지고 회로가 복잡해질수록 물리적인 설계가 어려워지므로, 설계를 HDL로 표현하여 언어화함으로써 검증 및 설계를 더욱 편리하게 할 수 있다.

대표적인 HDL 언어로 Verilog가 있지만, 그 외에도 여러 다른 HDL이 존재한다. Altera EDA Tool에서 사용할 수 있는 AHDL, 자바를 기반으로 한 FPGA 기술 언어인 JHDL, 파이썬 환경에서 FPGA를 기술할 수 있는 MyHDL, 그리고 Verilog를 제외한 HDL 중 가장 범용적으로 사용되는 VHDL(VHSIC Hardware Description Language)이다. VHSIC은 Very-High-Speed Integrated Circuits의 약자로, 초고속집적 회로를 의미한다.

Verilog HDL이나 SystemVerilog가 c언어와 유사한 문법을 갖고 있는 것과 달리, VHDL은 Ada, pascal과고 유사한 문법을 갖고 있다. VHDL은 명령어 방식으로 동시성(concurrency)을 처리하는 Verilog와 달리, data flow 방식으로 동시성(concurrency)을 처리하기 때문에 시스템 수준의 보다 복잡하고 추상적인 설계에 특화되어 있다.

또한 SystemVerilog가 객체지향형 프로그래밍을 통해, Verilog HDL은 모듈화 설계(module construct)를 통해 재사용성을 갖는 것에 비해, VHDL은 다양한 패키지와 라이브러리를 통해 재사용성을 높인다.

*모듈화 설계: 독립된 기능적 구성요소인 모듈(module)이 서로 관계를 맺으며 확장하는 구조의 설계

2. Verilog의 역사와 발전 과정을 조사하시오.

Verilog는 1983년 Gateway Design Automation사에서 개발된 HDL로, 1990년 Cadence Design System에 기업이 인수되면서 업계 표준으로 자리잡게 되었다. 최초에 베릴로그에는 논리 회로 설계 및 하드웨어 기술에 관련한 기능이 없었으며, 오로지 회로의 시뮬레이션을 위한 언어로 설계되었다. 이후 Cadence 사가 Verilog를 퍼블릭 도메인으로 이전한 뒤, IEEE에 제출하면서 Verilog-95(IEEE Standard 1364-1995)가 IEEE 표준으로 공인되게 되었다.

이후에도 IEE 표준 1364로 Verilog-95, Verilog 2001, Verilog 2005가 개발되었

고, SystemVerilog와 Verilog의 언어 표준을 합성한 SystemVerilog 2009가 IEEE Standard 1800-2009로 새로 공인되었다. SystemVerilog란 회로의 모델링, 설계, 시뮬레이션, 구현에 사용되는 HDL 및 HVL(하드웨어 검증 언어; Hardware Verification Language)로, 더욱 현실적이고 효과적인 하드웨어 검증을 위해 기능이 확장된 언어이다. SystemVerilog standard 역시 2012년, 2017년, 2023년 업데이트가 이뤄지는 등 꾸준히 발전을 거듭하고 있다.

3. Verilog의 기본적인 구조와 문법에 대하여 조사하시오.

앞서 설명했듯이 Verilog는 C언어와 유사한 문법을 가지고 있으며, if문, while문과 같은 제어문과 출력 루틴, 기본 연산자들이 거의 유사하다. 단, C언어와 달리 블록의 시작과 끝을 중괄호, {} 대신 begin, end문을 이용해 구분한다는 점, 동시성 및 타이밍을 제어할 수 있다는 점 등에서 구별점이 있다.

Verilog는 **독립적인 기능을 하는 모듈을 통해 하드웨어를 기술**하는 언어로, 모듈은 크게 **머리부, 선언부, 몸체부**의 3가지로 구성된다. 머리부는 다음과 같이 **모듈의 이름, 모듈의 포트 목록**으로 구성된다.

module *module_name* ({port_list}):

선언부는 포트 목록에 나열된 **포트들의 방향, 비트 폭(bit width), reg과 wire의 선언, parameter 선언** 등 모듈에서 필요로 하는 구성요소들을 선언한다. 마지막으로 몸체부는 회로의 기능, 동작, 구조 등을 표현하는 다양한 Verilog 구문을 포함하는데, always/initial/assign문, function 및 task의 정의 및 호출, 하위 모듈 등이 해당된다. 한 개의 모듈은 다른 다양한 하위 모듈 개체(instance)들을 포함할 수 있다.

Verilog에는 크게 두 가지 자료형(data type)이 존재하는데, 추상적 저장 장치(=값을 저장)인 **register**와 디바이스의 물리적 연결을 담당하는 **net**이다.

register 자료형은 대표적으로 always, initial 등의 절차적 할당문에 의해 값을 받는 객체인 **reg**, 정수형 변수 **integer**, 실수형 변수 **real**, 시간형 변수 **time, realtime** 등이 있다. 다음으로 **net 자료형**은 논리적 동작·기능이 있는 자료형과 없는 자료형으로 분리할 수 있다. 논리적 동작·기능이 없는 자료형으로 **wire**와 3상태(tri-state)가 될 수 있는 wire인 **tri**가 있으며, 있는 자료형의 경우 대표적으로 **wand(wired AND), wor(wired OR)**가 있다. 이외에도 다양한 자료형이 존재한다.

Verilog의 **상수 선언**은 C언어에 비해 비교적 복잡하지만, 원리를 알면 간단하다. **(bit 수)'(입력 형식)(입력 값)**의 형식을 갖는데, 이 때 입력 형식을 바탕으로 입력 값을 읽어 bit로 변환한다.

a = 8'h13

예를 들어 8'h13의 경우 16진수로 13을 읽어 8bit로 저장하므로, 입력된 값은 10진

수로 19이고 00010011이 저장된다. 만약 8'13을 입력한다면, 13을 10진수로 읽으므로 00001101이 저장된다. 이처럼 값을 선언할 때 입력 형식에 주의할 필요가 있다. 단, 앞에 붙은 bit size를 꼭 붙이지 않아도 선언은 가능하다.

Verilog의 연산자는 3항 연산자, shift 연산자, 산술/논리 연산자를 아울러 C언어와 거의 유사하므로, ~&(NAND 연산), ^~(XNOR), **결합 연산자, 반복 연산자 등** Verilog에서만 사용되는 연산자에만 주의하면 큰 어려움이 없다.

이외에도 연속 할당문, 절차형 할당문, blocking/nonblocking 문법 등 Verilog에서 다루는 기초 문법이 더욱 다양하게 존재하지만, 결과보고서에서 묻는 내용과 중복되므로 예비보고서에는 생략하고, 결과보고서에서 다시 작성하고자 한다.