2주차 예비보고서

전공: 컴퓨터공학과 학년: 4학년 학번: 20191583 이름: 김태곤

**1. HDL이 무엇인지 조사하고 Verilog이외의 HDL에 대하여 조사하시오**

1) HDL이란

HDL(Hardware Description Language)은 디지털 회로 및 시스템의 설계와 모델링을 위해 사용되는 전문 프로그래밍 언어이다. 이 언어는 하드웨어의 기능 및 동작을 상세하게 기술하며, 설계자가 물리적 하드웨어를 제작하기 전에 회로의 동작을 정의하고 시뮬레이션할 수 있게 해준다. 복잡한 디지털 시스템과 반도체 장치를 효율적으로 설계하고 검증하는 데 필수적인 도구로, 마이크로프로세서, 메모리 배열, 사용자 정의 로직 블록 등 다양한 구성 요소를 포함한다.

HDL의 사용은 하드웨어 설계의 복잡성을 관리하고, 설계 과정을 자동화하여 개발 시간과 비용을 절감할 뿐만 아니라, 더 높은 수준의 재사용성과 검증 가능성을 제공한다. 가장 널리 알려진 HDL로는 VHDL(VHSIC Hardware Description Language)과 Verilog가 있다.

2) Verilog 이외의 HDL

(1) VHDL(VHSIC Hardware Description Language)은 초고속 집적회로(Very-High-Speed Integrated Circuits) 설계를 위해 개발된 하드웨어 기술 언어로, 디지털 회로의 설계, 검증, 구현에 사용되는 프로그래밍 언어이다. 복잡한 전자 회로를 정밀하게 기술하는 데 사용되며, 시간과 동시성을 표현할 수 있는 독특한 표기법을 갖추고 있다. VHDL은 일반적인 소프트웨어 프로그래밍 언어와는 다른 특성을 지니며, 이러한 차이점은 VHDL이 특히 하드웨어 설계의 자동화에 중점을 둔 언어이기 때문이다.

(2) MyHDL은 Python 프로그래밍 언어를 기반으로 한 하드웨어 기술 언어(HDL)이다. Jan Decaluwe에 의해 개발되었으며, 디지털 회로의 설계, 시뮬레이션, 검증, 합성 과정에 사용된다. MyHDL은 Python의 강력한 기능과 유연성을 활용해 하드웨어 설계를 더 쉽고 효율적으로 만든다. MyHDL로 작성된 코드는 파이썬 스크립트로 실행되며, 디지털 로직을 모델링하고 시뮬레이션할 수 있다. 또한 MyHDL은 VHDL이나 Verilog와 같은 전통적인 HDL로 변환될 수 있어, 이를 통해 FPGA나 ASIC과 같은 실제 하드웨어로 합성될 수 있다.

**2.Verilog의 역사와 발전 과정을 조사하시오**

Verilog는 1980년대 초반, 디지털 회로의 설계 자동화를 목적으로 개발된 하드웨어 기술 언어(HDL)이다. Phill Moorby에 의해 처음 개발되어, 1983년 Prabhu Goel이 설립한 Gateway Design Automation 회사에 의해 공개되었다. 이 언어는 당시 복잡해지고 있는 디지털 회로의 설계와 검증 과정을 간소화하고 효율적으로 만들기 위한 도구로써, 게이트 레벨의 시뮬레이션을 주목적으로 하고 있었다.

1985년, Verilog-XL 버전과 해당 시뮬레이터가 공개되었고, 이는 Verilog 언어의 사용범위를 확장시키는 데 중요한 역할을 했다. 그 후 1989년 Cadence Design Systems에 의해 Gateway Design Automation이 인수되면서, Verilog는 업계 표준 HDL로 자리매김하기 시작했다. 1995년, IEEE는 Verilog를 공식적으로 IEEE 1364 표준으로 채택하며 Verilog-95로 알려지게 되었다. 이 표준화 과정은 Verilog의 호환성과 사용성을 크게 향상시켜 주었다.

2001년에는 Verilog-2001 업데이트가 발표되었다. 이 업데이트는 2의 보수 표현, 새로운 변수 타입의 지원, 파일 입출력 기능 개선, C 스타일의 함수 등 새로운 문법과 기능을 추가하여 사용자가 더 복잡한 디지털 시스템을 설계하고 모델링할 수 있도록 해주었다.

디지털 시스템 설계에서 검증의 필요성이 점점 커지면서, 이를 지원하기 위한 대대적인 확장이 필요하게 되었다. 이러한 요구를 충족시키기 위해 2005년 SystemVerilog가 IEEE 표준으로 채택되었다. SystemVerilog는 기존 Verilog의 기능에 시스템 수준 모델링과 검증을 위한 다양한 기능을 추가하여, 설계와 검증 분야에서 더욱 강력한 도구로 자리잡았다.

오늘날 Verilog와 그 확장판인 SystemVerilog는 전 세계적으로 디지털 회로 설계와 검증에 널리 사용되는 언어 중 하나로, 그 역할과 중요성은 계속해서 증가하고 있다.

**3.Verilog의 기본적인 구조와 문법에 관하여 조사하시오**

1) Verilog의 기본적인 구조

Verilog는 C언어와는 달리, 코드 블록의 시작과 끝을 나타내기 위해 중괄호 대신에 'begin'과 'end'를 사용한다. 전체 구조는 크게 세 부분, 즉 머리부, 선언부, 몸체부로 나누어진다.

머리부 : 머리부는 Verilog 모듈의 시작을 알리는 부분으로, 모듈의 이름과 모듈의 포트(입력, 출력, 입출력)를 선언한다. 모듈은 Verilog에서 하나의 디지털 회로 단위로 간주되며, 이 부분에서는 해당 회로가 외부 세계와 어떻게 연결되는지를 정의한다.

선언부 : 선언부에서는 모듈의 포트에 대한 더 자세한 정보를 제공한다. 이는 포트의 방향(입력, 출력, 입출력)과 데이터 타입(비트의 너비와 타입 등)을 명시하는 부분이다. 이 부분에서는 또한 모듈 내부에서 사용할 변수나 신호들을 선언할 수도 있다.

몸체부 : 몸체부는 모듈의 실질적인 기능을 구현하는 부분이다. 여기에는 할당문(assignment), 조건문(if, case), 반복문(for, while) 등 Verilog의 다양한 구문을 사용하여 특정 기능을 수행하는 로직이 포함된다. 몸체부에서는 입력 포트로부터 받은 데이터를 처리하고, 그 결과를 출력 포트로 보내는 등의 연산이 수행된다. 또한, 인스턴스화(instantiation)를 통해 다른 모듈을 호출하여 사용하는 것도 이 부분에서 이루어진다.

2) Verilog 문법

\* 데이터 타입

• wire: wire 데이터 타입은 연결선을 나타내며, 주로 외부 포트나 다른 회로 요소 간의 연결에 사용된다. 이는 값이 저장되지 않는 경로로서, 신호가 흐를 수 있는 경로를 제공한다. wire에 할당된 값은 연속 할당을 통해 정의되며, 신호의 변화가 있을 때마다 자동으로 업데이트 된다.

• reg: reg는 내부적으로 값을 저장할 수 있는 레지스터를 나타낸다. 주로 always 블록 내에서 사용되며, 순차적인 로직 구현이나 상태 기계 설계에 필수적이다. reg 변수는 조건에 따라 값이 변경될 수 있으며, 비연속 할당이 가능하다.

• integer, real: 각각 정수와 실수 값을 저장하는데 사용된다.

• time : 시뮬레이션의 시간 값을 나타내는 time 데이터 타입은 시간 지연 값이나 시뮬레이션 중의 특정 시간을 기록하는 데 사용된다.

\* 할당문

• 연속 할당: 연속 할당문은 assign 키워드를 사용하여 wire 타입 변수에 값을 지속적으로 할당하는 방식이다. 이 방식은 주로 조합 회로의 구현에 사용되며, 할당된 신호는 입력 신호가 변할 때마다 자동으로 업데이트된다. 연속 할당은 하드웨어의 연결을 표현하는 데 사용되며, 신호의 논리적인 연산 결과나 다른 신호의 값을 wire 타입 변수에 연결하는 데 활용된다.

*assign wireName = expression;*

• 비연속 할당: 비연속 할당은 always 및 initial 블록 내에서 사용되며, reg 타입 변수에 값을 할당하는 방식이다. 비연속 할당은 조건문 또는 반복문과 함께 사용되어 순차적인 로직이나 상태 기반의 동작을 구현하는 데 적합하다. 할당된 값은 특정 조건이 충족되거나 블록 내의 코드가 실행될 때 변경될 수 있으며, always 블록은 일반적으로 클럭 신호의 변화에 반응하여 동작한다.

*always @(posedge clk) begin*

*regVariable <= expression;*

*end*

\* 연산자

기본적인 언어들과 동일하다.

• 산술 연산자: +, -, \*, /, % 등

• 논리 연산자: &&, ||, !

• 비교 연산자: ==, !=, >, <, >=, <=

• 비트 연산자: &, |, ^, ~, <<, >>

• 조건 연산자: ? : (삼항 연산자)

\* 제어문

• if 문: 조건에 따라 다른 명령을 실행한다.

*if (condition) begin*

*// 조건이 참일 때 실행할 명령*

*end else begin*

*// 조건이 거짓일 때 실행할 명령*

*end*

• case 문: 여러 조건 중 하나를 선택해 실행한다.

*case (variable)*

*value1: // 값1에 대한 처리*

*value2: // 값2에 대한 처리*

*default: // 기본 처리*

*endcase*

• for, while 문: 반복적인 작업을 수행한다.

*for (initialization; condition; increment) begin*

*// 반복 실행할 명령*

*end*