전공: 컴퓨터공학 학년: 2학년 학번: 20161603 이름: 신민준

1. HDL이 무엇인지 조사하고 Verilog 이외의 HDL에 대하여 조사하시오.

HDL은 Hardware Description Language의 약자이며, 하드웨어 기술 언어로 번역된다. HDL은 전자 회로를 정밀하게 기술하는 데 사용하는 컴퓨터 언어이다. HDL은 원하는 동작을 회로에 기술할 수 있고, 원하는 회로 구조를 기술할 수도 있으며, 기술한 내용을 시뮬레이션을 통해 정상적으로 동작하는지 검증할 수도 있다.

HDL의 설계는 두 가지 종류의 시스템을 설계하기 위해 만들어졌다. 하나는 HDL 프로그램을 시뮬레이션 할 수 있는 능력이다. 이 HDL 프로그램은 PLD와 같은 장비에 업로드되어 해당 프로그램에 대한 테스트를 거치거나, 프로그램을 칩으로 제작하여 테스트 할 수 있다. 하지만, 이런 테스트들은 보통 높은 비용을 요구하고, 걸리는 시간이 길기 때문에, 대부분의 디버깅 및 테스트들은 시뮬레이터를 통해 수행한다. 이 시뮬레이터는 실제 디지털 기기의 클럭과 유사하게, 리셋 가능한 클럭을 유지하면서 설계자가 코드를 디버깅 하는 데 있어 특정 시간의 수많은 레지스터의 값들을 볼 수 있도록 해주는 역할을 한다.

HDL 시뮬레이션에 있어 HDL 설계에 가장 중요한 부분은 HDL 프로그램을 시뮬레이션 할 수 있는 능력이라고 볼 수 있다. HDL 프로그램의 테스트는 PLD과 같은 장비에 업로드를 하거나, 이를 칩으로 제작하는 방식으로 할 수 있다. 하지만, 이렇게 테스트를 하는 과정은 보통 큰 비용이 들거나, 오랜 시간이 걸리므로, 대다수의 테스트, 디버깅들은 시뮬레이션을 통해 이루어진다. 이 때 사용되는 시뮬레이터들은 디지털 기기에서 실제로 사용되는 클럭들과 유사하게 설정된-하지만 리셋 가능한-클럭을 유지하면서, 설계자가 코드를 디버깅할 때 특정 시간에 사용되는 다양한 레지스터들의 값들을 실시간으로 볼 수 있게 해준다.

HDL은 일반적인 소프트웨어에서의 프로그래밍 언어와 유사점을 같지만, 둘 사이에는 약간씩의 차이가 존재한다. 두 언어는 공통적으로 – HDL의 경우에는 보통 Synthesizer라고 불리기는 하지만 – 컴파일러에 의해 처리된다. 하지만, HDL의 컴파일 – Synthesizer의 경우, 다음과 같은 단계를 거쳐서 컴파일된다. 먼저, HDL 자체의 고유한 포맷으로 이루어진 칩 내부의 Logic을 보이는 파일을 생성한 후, 산업 표준 – Industry-standard인 EDIF 포맷으로 이를 변환한다. 이후, 마지막으로 JEDEC 포맷 파일로 변환하는데, 이 파일은 PLD를 프로그래밍 하는 데 있어 PLD 프로그래머가 필요로 하는 명령어들을 담고 있다.

이와는 반대로, 소프트웨어 컴파일러들은 그들의 마이크로프로세서들에서 직접적으로 실행되는 명령어를 생산해낸다. 하지만, 이에 따라서, HDL과 소프트웨어 프로그래밍 언어 사이에 존재하는 차이는 두 언어의 특징을 결합하면서 탄생한 reconfigurable system이 대두되면서부터 그 차이의 경계가 모호해지는 중이다. 따라서, 현재는 회사들의 각자의 고유한 포맷들을 이용하는 것 보다는 VHDL이나 Verilog로 대표되는 표준 HDl로 그 포맷이 넘어가는 추세에 놓여 있는 상황이다.

대표적으로 HDL은 Verilog HDL과 VHDL이 존재하는데, 여기에서 VHDL은 VHSIC Hardware Description Language의 약자로, 디지털 회로 및 혼합 신호 – mixed-signal을 표현하는 데 쓰이는 Hardware Description Language - HDL이다. 아날로그 신호, 즉 Mixed-signal은 실제 활용할 때 디지털 회로에 대부분 적용된 상태로 사용한다.

VHDL의 방식으로 표현되어 있는 회로에서 실제로 동작하는 기능적 소자로 변환하기 위해서는 합성 – Synthesis 과정을 거쳐야 실제로 동작이 가능한 회로가 완성된다. 이 때, FPGA, ASIC등의 환경에 따라 실제로 합성된 회로의 소자들의 내용이 달라지므로, 칩을 설계할 때 사용되는 개발도구는 목적에 맞는 소자가 사용되어 합성되도록 하게 만들어진다. 예로, FPGA를 판매하는 회사는 일반적으로 회로 입력, 즉, 심볼 및 VHDL코드에서부터 시작해서, 제작한 VHDL코드의 Synthesis, 그리고 Module들의 배치 등을 지원하는 도구를 기본적으로 지원하는 관계에 있다.

1. Verilog의 역사와 발전 과정을 조사하시오.

Verilog은 Phil Moorby에 의해 모의 시험용 언어의 목적을 가지고 개발된 언어이다. 개발 당시, Verilog는 단어 “Verification”과 “Logic”의 합성어로 그 이름의 유래를 찾아볼 수 있다. 이후, Verilog는 Cadence Systems에 의해 인수가 되었고, 그 후 업계 표준의 HDL – Hardware Descriptive Language로 자리잡게 되었다.

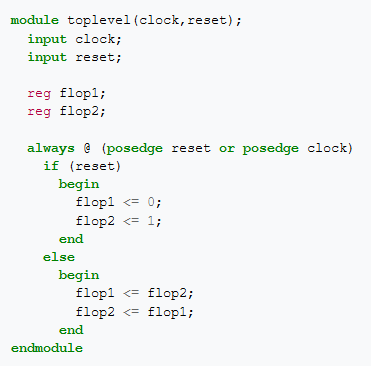
1995년에 IEEE표준이 만들어진 후, Verilog-2001이라 일컫어지는 2001년의 변경이 있었지만, 이후 디지털 시스템을 설계할 때의 검증의 역할이 커지게 되어, 이를 지원하기 위한 기능의 확장의 필요성이 대두되기 시작했다. 이에 따라 SystemVerilog가 IEEE표준으로서 2005년에 만들어지게 되었다.

하지만, SystemVerilog는 실제 업계에서는 검증용의 테스트 벤치를 작성할 때 외에는 잘 쓰이진 않는데, 이 이유는 C에서 쓰이는 enum, class, struct, union 등의 데이터 타입들이 실제 하드웨어로 Synthesis할 때 그 결과물로 나오는 netlist에 대해 직관적인 예측이 힘들게 하고, 도구 사이의 문법적인 해석에 있어 차이가 있을 때 netlist를 기반으로 한 시뮬레이션 결과를 토대로 RTL 디버깅을 실시간으로 실시하기 굉장히 까다롭기 때문이다. 당연히, Verilog에서의 wire나 register 데이터 타입들은 소스코드에서 명시된 그대로 wire나 register로 합성되는 것은 아니나, SystemVerilog에 비하면 netlist를 직접 고치는 것이 가능할 정도이다.

1. Verilog의 기본적인 구조와 문법에 관하여 조사하시오.

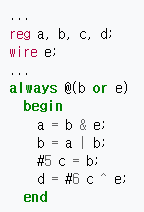
Verilog는 Ada 기반의 VHDL과 달리 C언어와 비슷한 문법을 가지고 있는 것이 특징이라고 볼 수 있다. If, for, while문과 같은 제어 구조에 있어서도 서로 동일하며, 연산자나 출력 루틴들도 거의 비슷하단 점 또한 사용자들의 접근성을 높이는 역할을 한다. 하지만, case문의 문법 형태나, 블록의 시작과 끝을 ‘{‘, ‘}’ 대신 Begin, End로 사용하는 것은 Pascal과 유사하다.

두 Flip-flip을 구현하는 Verilog 코드는 다음과 같다:



Verilog에서의 할당 연산자인 ‘<=’는 일반적인 C언어와 같은 언어와는 다른 기능을 수행한다. 이 할당 연산자는 non-blocking(논-블로킹)이라 불리며, clock이 발생할 때, 해당 할당 연산자가 병렬적으로 동시에 실행됨을 의미한다. 따라서, 위의 Flip-flop 코드에서 flop1과 flop2는 clock cycle에 따라 서로 그 값이 바뀌게 되는 것이다.

Non-blocking이라고 불리는 또 다른 할당 연산자는 ‘=’으로, 이는 clock이 발생할 때, 각 명령이 순차적으로 실행되고, 따라서 먼저 할당된 값이 나중에 할당된 값에 순차적인 영향을 주게 된다는 것을 의미한다. 만약 위 Flip-flip의 예에서 ‘<=’ 할당 연산자가 ‘=’ 할당 연산자로 바뀌어지게 된다면, flip1과 flip2는 그들의 값이 변하지 않은 채 그 값을 유지할 것이다.



Delay에 대한 예로는 위와 같은 코드가 있다: