제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003.5)

FPGA 상의 설계 검증을 위한 논리 분석기 소프트웨어 및 하드웨어 구현

황수연, 정성헌, 장경선
충남대학교 컴퓨터공학과
e-mail:charisma@ce.cnu.ac.kr

The Implementation of Logic Analyzer Software & Hardware for Design Verification on FPGA board

Soo-Yeon Hwang, Sung-Heon Jung, Kyoung-Son Jhang
Dept of Computer Engineering, Chung-Nam University

요 약

FPGA 보드를 이용하여 디지털 논리 설계를 검증하려면 고가의 논리 분석기 장비를 필요로 한다. 본 논문은 FPGA 설계에 대한 검증을 PC에서 직접 입력 데이터를 FPGA 보드 쪽으로 전달하고 그 결과를 다시 PC 쪽에서 GUI 형태로 확인할 수 있도록 구성된, 논리 분석기 기능을 갖는 VHDL 모듈과 소프트웨어의 구현에 관한 것이다. 이와 같은 VHDL 모듈과 소프트웨어 모듈을 활용함으로써 추가 비용 없이 검증 과정을 수행할 수 있다.

1. 서론

FPGA 실험용 Kit는 고등학교나 대학교에서 디지털 회로 실험, 디지털 시스템 설계 실험 등에 광범위하게 사용되고 있다. 그러나 설계 결과를 하드웨어적으로 검증하기 위해서는 Kit 자체보다 훨씬 고가인 논리 분석기를 필요로 한다. 따라서, 저렴한 비용으로 설계 결과를 하드웨어적으로 검증할 수 있는 방법을 필요로 하고 있다. 본 연구에서 구현한 논리 분석기를 기존에 이미 구입한 FPGA 보드와 함께 사용함으로써 고가 장비 구입 없이 설계결과를 하드웨어적으로 검증 해 볼 수 있다.

본 연구에서 구현한 논리 분석기는 크게 두 부분으로 구성되어 있다. 한 부분은 하드웨어 부분인 LAPG (Logic Analyzer and Pattern Generator) 이고, 다른 한 부분은 소프트웨어 부분인 SLAPG (Software Logic Analyzer and Pattern Generator) 이다. 또한 하드웨어와 소프트웨어 사이의 통신을 위해 Parallel Port [1] 를 사용하였다.

2. 논리 분석기

2.1 LAPG

2.1.1 개요

LAPG는 PC로부터 Parallel Port를 통해 전송되는 제어신호에 의해 구동되는 FSM (Finite State Machine) 이다. (VHDL로 구현) 즉, PC에서 사용자가 원하는 결과를 얻기 위해 적당한 컨텐츠들 (제어신호, 데이터) 을 Parallel Port를 통해 FPGA 보드 쪽으로 보내면, FPGA 보드에 다운로드된 LAPG가 그 신호들을 입력받아 해당 동작을 수행하는 VHDL 모듈이다.

2.1.2 구현 시 고려사항

2.1.1과 같이 PC와 LAPG 사이의 Parallel Port를 통한 데이터 전송에는 몇 가지 문제점이 있다. 이런 문제점의 주요 원인은 Parallel Port를 통해 LAPG로 전송되는 컨텐츠들의 속도와 FPGA 보드에 다운로드된 LAPG의 속도가 서로 다르다는 점이다. 즉, 두 모듈이 서로 비동기적으로 동작해야 한다는 점이다.

이러한 문제를 해결하기 위해 본 연구에서는 하나의 프로토콜을 제안했다. 즉, busy 신호를 이용한 두 모듈 사이의 핸드 쉐이크 (Handshake) 프로토콜이다. 프로토콜에 대한 자세한 내용은 3.1에 소개된다.

2.1.3 기능

LAPG는 기본적으로 PC, FPGA 보드에 장착된 SRAM, 사용자 디자인 사이에서 중개 역할을 하는 VHDL 모듈이다. 즉, PC 쪽에서 유효한 데이터를 LAPG 쪽으로 보내주면, LAPG는 그 데이터를 SRAM에 저장하고, 다시 저장된 데이터를 사용자 디자인의 입력 값 (Test Pattern) 으로 사용한다. 사용자 디자인의 입력 값에 대한 결과 값은 LAPG에 의해 SRAM에 저장된다. 이때 입력 값이 저장된 SRAM의 주소와 결과 값을 저장할 SRAM의 주소는 사용자가 설정할 수도 있고, 설정하지 않았다면 서로 겹치지 않도록 디폴트 값이 설정된다. 마지막으로 PC 쪽에서 사용자는 결과 값이 저장된 SRAM을 access하여 유효한 값을 읽을 수 있게 된다. 이와 같은 흐름으로 전체 동작이 이루어진다. 이처럼 LAPG는 기능 (Functionality) 면에서 아주 중요한 역할을 담당하게 된다.

2.2 SLAPG

SLAPG는 PC에서 Parallel Port를 통해 FPGA 보드에 다운로드된 LAPG를 구동시키는 소프트웨어 모듈이다. (Visual C++로 구현) 즉, 사용자 디자인을 테스트하기 위해 Parallel Port를 통해 LAPG 쪽으로 테스트 벡터 값과 제어신호를 보내주고, LAPG에 의해 테스트된 결과 값을 받아 다시 화면에 디스플레이 해 주는 역할을 한다. 또한 SLAPG는 Stimulus 편집 기능과 Waveform Viewer 기능을 가지고 있다.

3. 논리 분석기 구현

3.1 프로토콜

그림 1은 PC (Software 영역: SLAPG) 와 FPGA (Hardware 영역: LAPG) 가 Parallel Port를 통해 통신할 때 사용되는 프로토콜을 보여준다.

PC 쪽의 SLAPG는 어떠한 명령을 시작할 때 항상 busy (active low 신호) 신호를 체크한다. busy 신호가 '1' (not busy) 이면 원하는 명령을 시작하고, '0' (busy) 이면 '1'이 될 때까지 대기하게 된다. 즉, not busy인 경우에만 제어신호를 LAPG 쪽으로 보내준다. 이와 같은 동작이 그림 1의 Parallel Port 영역에 잘 나타나있다. 또한 FPGA 쪽의 LAPG는 제어신호를 받으면 바로 busy 신호를 '0' (busy) 으로 assert 하고, 제어신호에 해당하는 동작을 수행한다. 수행이 완료되면 busy 신호를 '1' 로 de-assert 해 주고 다음 명령을 받는다. 이와 같은 동작을 반복함으로써 PC와 FPGA 사이의 비동기적인 문제를 해결할 수 있다.

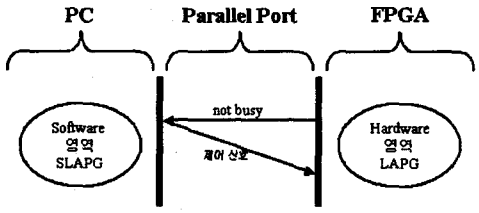

그림 1. 프로토콜

3.2 제어신호 및 동작

제어신호는 사용자가 LAPG를 구동시키기 위한 명령으로써 표 1에 정리된다.

신 호	명 령	신호 설명
"0001"	WRITE	SRAM에 데이터를 쓰기위한 명령어
"0010"	READ	SRAM에 저장된 데이터를 읽기 위한 명령어
"0011"	CONTINUE 1	명령이 계속 됨을 의미하는 명령어 1
"0100"	CONTINUE 2	명령이 계속 됨을 의미하는 명령어 2
"0101"	CONFIGURATION	Configuration Mode를 설정하기 위한 명령어
"0110"	TESTING	사용자가 설계한 하드웨어 모듈을 작동시키기 위한 명령어
"0111"	EOF	명령이 끝났음을 의미하는 명령어
"1000"	RESET	LAPG를 리셋 시키기 위한 명령어

표 1. 제어신호 정의

표 1에서 CONFIGURATION 모드에는 4가지가 있다. RB는 "Read Begin address"로서 읽기 위한 SRAM의 시작 주소를 의미한다. RL은 "Read Length"로서 SRAM에 저장된 데이터를 몇 번 읽을

것인가를 결정하는 횟수를 의미하며, WB는 "Write Begin address"로서 쓰기 위한 SRAM의 시작 주소를 의미한다. 마지막으로 WL은 "Write Length"로서 SRAM에 데이터를 몇 번 쓸 것인가를 결정하는 횟수를 의미한다. 또한 TESTING 과정은 분석하고자 하는 사용자 설계 모듈에, SRAM에 이미 저장된 테스트 벡터 값을 적용시켜서 그 결과 값을 다시 SRAM에 쓰는 단계까지를 의미한다.

표 1의 제어신호를 바탕으로 전체 동작의 순서를 정리하면 다음과 같다.

1. CONFIGURATION 동작: CONFIGURATION 명령과 CONTINUE 명령을 이용하여 사용자가 원하는 WB, RB, WL, RL을 순서대로 LAPG 쪽으로 보낸다. 이때 마지막 정보인 RL을 보낸 후에 EOF 명령을 LAPG 쪽으로 보낸다. 그럼으로 해서 LAPG는 CONFIGURATION 동작이 완료되었음을 인식한다. 또한 CONFIGURATION 동작에서 정해진 WB, RB, WL, RL은 2, 3, 4 동작에서 필요로 하는 중요한 정보이다.

2. WRITE 동작: WRITE 명령과 CONTINUE 명령을 이용하여 사용자가 원하는 테스트 벡터 값을 연속해서 FPGA 보드상의 SRAM에 저장한다. 저장하는 횟수와 SRAM의 시작 주소는 1의 CONFIGURATION 동작에서 정해진 정보를 이용한다. 이때도 마찬가지로 마지막 테스트 벡터 값을 저장한 후 EOF 명령을 LAPG 쪽으로 보냄으로써 WRITE 동작이 완료되었음을 알려준다.

3. TESTING 동작: TESTING 명령을 이용하여 분석하고자 하는 사용자 디자인을 테스트하는 동작이다. 이때도 마찬가지로 LAPG는 1의 CONFIGURATION 동작에서 정해진 정보를 이용한다.

4. READ 동작: READ 명령과 CONTINUE 명령을 이용하여 3의 TESTING 동작에서 얻어진 테스트 결과 값을 읽는 동작이다. 이때도 마지막 결과 값을 읽은 후 EOF 명령을 LAPG 쪽으로 보낸다. 따라서, LAPG는 모든 동작이 완료되었음을 인식한다.

3.3 논리 분석기의 전체 구조

그림 2는 본 연구에서 구현한 논리 분석기의 전체 구조를 보여준다. 또한 데이터의 전체 흐름도 보여주고 있다.

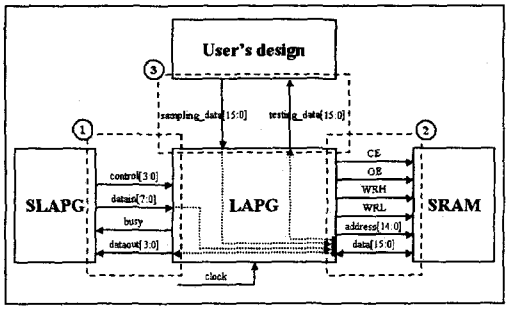

그림 2. 전체 블록도

그림 2에서 SLAPG는 PC상의 소프트웨어이고, LAPG는 FPGA 보드에 다운로드된 하드웨어이다. User's Design은 분석하고자 하는 사용자 디자인이며, SRAM은 FPGA 보드에 물리적으로 장착된 메모리이다. 그림 2의 각 번호는 해당하는 통신을 표시하고 있다. 1번은 Parallel Port의 인터페이스를 이용한 통신이며, 2번은 SRAM을 access하기 위한 통신이다. 마지막으로 3번은 사용자 디자인을 테스트하기 위한 통신이다.

4. 실험 결과

이번 장에서 실제 실험 결과를 보이겠다. FPGA 보드에 다운로드 하기 위해 Altera사에서 제공하는 MAX+PLUS II 10.1 버전 [2] 을 사용하였으며, 실험 대상이 되는 FPGA 칩탑 모듈 또한 Altera사에서 제공하는 FLEX 10K (EPF10K30RC208-3) 디바이스 [3] 를 사용하였다.

4.1 조합 회로 (Combinational Logic)

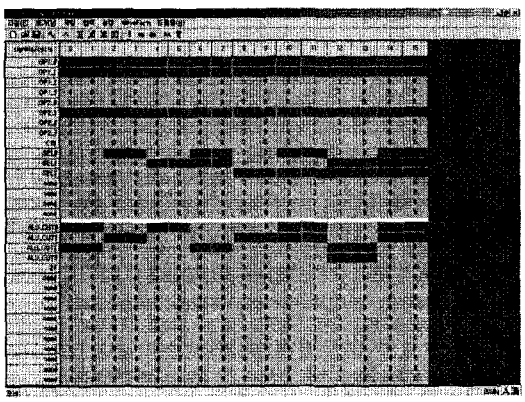

그림 3. 조합 회로 예제

그림 3의 사용자 디자인은 ALU이다. OP1[3:0]과 OP2[3:0]는 입력 값이며 (OP1:3, OP2:2), CIN은 Carry Input 이다. 또한 SEL[2:0]은 선택 신호이다. ("000": OP1 + OP2, "001": OP1 - 1, "010": OP1 - OP2, "011": OP1 + 1, "100": OP1 and OP2, "101": OP1 or OP2, "110": not OP1, "111": OP1) 출력 값으로 ALU_OUT[3:0]은 SEL[2:0] 신호에 따라 동작하는 결과 값이며 ZF는 Zero Flag 신호이다. none으로 표시된 값은 사용하지 않는 핀을 의미한다. 그림 3에서 6, 7 사이클을 보면 SEL 신호의 값이 "011"이다. SEL 신호가 "011"이면 OP1 + 1의 연산을 취한다. 따라서 OP1의 값 3에 1을 더한 값으로 4가 (ALU_OUT[3:0] = "0100") 출력됨을 볼 수 있다. 다른 부분도 이와 마찬가지로 검증해 보면 된다.

그 이외에도 조합 회로들을 검증해 본 결과 이상 없이 작동함을 관찰하였다.

4.2 순차 회로 (Sequential Logic)

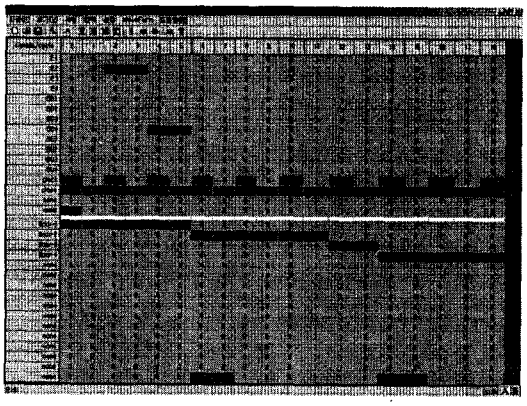

그림 4. 순차 회로 예제

그림 4의 사용자 디자인은 클럭에 의해 동작하는 4층 엘리베이터 제어기이다. E1부터 E4는 엘리베이터 내부에 있는 층수 버튼이며, BD과 BU은 각 층에 있는 Button Down, Button Up 버튼이다. clock은 클럭 신호이고 reset은 리셋 신호이다. 또한 close는 엘리베이터의 문을 닫는 버튼이다. floor-1부터 floor-4는 엘리베이터가 현재 몇 층에 있는지를 알려주는 출력 신호이며, Open 신호는 사용자가 원하는 층수에 도달했을 때, 문을 열어주는 출력 신호이다. 그림 4에서 사용자가 엘리베이터 내부에서 2층 버튼을 눌렀다. (2, 3 사이클) 한 사이클 뒤에 4층에서 사용자가 Button Down 버튼을 눌렀다. (4, 5 사이클) 결과적으로 출력 값을 보면 2층에서 한 사이클 동안 문이 열렸고, (6, 7 사이클) 다시 문을 닫은 후 4층에 엘리베이터가 도착했을 때 한 사이클 더 엘리베이터의 문이 열렸다. (14, 15 사이클) 따라서 엘리베이터가 정상적으로 동작함을 검증할 수 있다.

그 이외에도 순차 회로들을 검증해 본 결과 이상 없이 작동함을 관찰하였다.

5. 결론

본 연구에서 구현한 논리 분석기를 사용함으로써 교육 기관이나 학교에서 논리 분석기와 같은 고가 장비에 대한 구입 부담을 줄일 수 있을 것이다. 또한 FPGA 보드를 실험용 장비로 사용하는 고등학교나 대학교의 디지털 회로 실험, 디지털 시스템 설계 실험 시간에 논리 분석기 장비를 대신하여, 본 연구에서 구현한 논리 분석기를 사용함으로써 설계의 동작 여부를 하드웨어적으로 손쉽게 확인해 볼 수 있을 것이다. 따라서 디지털 회로 설계 교육에 많은 도움을 줄 것으로 기대된다.

현재는 일부 보드에서만 지원되지만, 추후에는 여러 보드에 동작하는 버전을 만들 계획이다. 해당 소프트웨어와 하드웨어는 binary 형태이지만 게시판에서 download 받아 사용해 볼 수 있다.

게시판의 URL은 다음과 같다.

- http://eda.ce.cnu.ac.kr
- 상위 메뉴 중, "LAPG 게시판"을 클릭하면 해당하는 게시판을 볼 수 있다.

참고문헌

[1] Programming the Parallel Port, Interfacing the PC for Data Acquisition and Process Control, Chapter 3, Chapter 4, Dhananjay V.Gadre, R&D Books, Lawrence, KS 66406

[2] MAX+PLUS II Manual: http://www.altera.com/literature/lit-mp2.html

[3] FLEX 10K EPF 10K30 Device Pin Table: http://www.altera.com/literature/lit-dp.html#FLEX10k