논문 00-25-11B-14 한국통신학회논문지 '00-11 Vol.25 No.11B

IEEE 1149.1을 기반으로 하는 테스트 시스템을 위한 테스트 버스 콘트롤러의 설계 및 구현

정회원 조 용 태*, 정 득 수*, 송 오 영*

Design and Implementation of A Test Bus Controller for IEEE 1149.1- Based Test System

Yong Tae Cho*, Deuk Soo Jung*, Oh Young Song* *Regular Members*

요 약

본 논문은 보드 레벨 테스팅 및 경계주사기법의 응용을 위한 테스트 버스 콘트롤러의 설계와 구현에 관해 다룬다. 테스트 버스 콘트롤러는 프로세서와 인터페이스를 통하여 IEEE 1149.1 테스트 버스를 제어하기 위한 칩이다. 최근 들어 IEEE 1149.1은 여러 분야에서 응용되어지고 있어서 다양한 응용분야에 적합한 테스트 버스 콘트롤러의 설계가 요구된다. 보드 레벨 테스팅을 위해서 SVF에 정의된 테스트를 수행할 수 있어야 하며, System-on-a-Chip (SoC) 설계 방식에서 내장되어지기 위해서는 작은 칩 크기와 높은 고장 검출률을 가져야 한다. 본 논문에서 구현된 칩은 기존의 테스트 장비에서 널리 쓰이는 SVF에 정의된 테스트를 모두 지원하며, 12k 게이트 정도의 크기를 가진다. 또한 독립적인 칩으로 쓰일 경우는 테스트 버스 콘트롤러가 버스 슬래이브로 쓰일 수 있으므로 IEEE 1149.1 테스트 회로를 가지도록 설계하였다.

ABSTRACT

This paper describes on a design and implementation of test bus controller for board-level testing and boundary scan test application. The test bus controller is a chip for interfacing between a processor and IEEE 1149.1 test bus. Since IEEE 1149.1 is used for many application fields recently, the design of test bus controller for these is essential. The test bus controller must perform SVF commands fully for board-level testing or ISP, and have a small chip area(small gate count) and high fault coverage to be embedded in a System-on-a-Chip (SoC) design methodology. The test bus controller implemented in this paper supports all test defined in SVF, which is broadly used for commercial ATE(Automatic Test Equipments). It has about 12k 2-input nand gates. Also it has an IEEE 1149.1 test logic for the case of bus slave.

Ⅰ. 서 론

최근 들어 VLSI칩의 급격한 고집적화, 멀티칩 모듈의 등장 및 새로운 칩 패키징 기술의 출현 등으로 기존의 ICT(In-Circuit Tester)혹은 기능적 보드 테스터(Functional Board Tester)로는 높은 고장 검출률을 갖는 보드 테스팅이 매우 어렵게 되었다. 이런 문제점을 해결하기 위한 사람들이 모여서 JTAG (Joint Test Action Group)을 결성하고 표준화 작업을 진행시켰다. 이를 토대로 IEEE에서 경계주사 구조(Boundary Scan Architecture)와 프로토콜을 정의한 국제 표준인 IEEE 1149.1이 1990년도에 만들어졌다[1,2,3]. 이 표준은 흔히 JTAG이라 불린다.

IEEE 1149.1은 보드레벨 테스팅을 위한 표준이

* 중앙대학교 전자전기공학부(song@jupiter.cie.cau.ac.kr)
논문번호 : 00213-0619, 접수일자 : 2000년 6월 19일
※ 본 연구는 정보통신부지원의 대학정보통신연구센터 육성지원사업(정보통신연구진흥원)에 의하여 수행되었습니다.

지만 최근 들어서는 여러 분야에서 응용되어지고 있다. 기존에는 범용 프로세서나 DSP칩 등을 설계 시 그 구조의 검증이나 응용프로그램의 하드웨어 기반 디버깅을 위하여 ICE(In-Circuit Emulator), ROM Monitor등의 방법을 사용하였다. 그러나 최근에는 JTAG을 이용하여 하드웨어 에뮬레이션을 수행함으로서 내부 레지스터에 쉽게 접근할 수 있게 되었다[4,5]. 다른 응용 분야는 ISP(In-System Programming)이다. 데이터 통신 분야처럼 칩의 수명이 짧은 시장에서는 칩 제조기간을 단축하기 위해서 FPGA나 CPLD를 사용하는데, 이런 PLD를 보드에 장착된 상태에서 프로그래밍하거나 제조 후 업그레이드를 위해서 JTAG을 이용한 ISP가 널리 쓰이고 있다[6]. 또한 시스템 레벨의 테스트에 적용하여 ASP(Addressable Scan Port)을 제어하는데 적용할 수 있다. Shadow protocol를 이용하여 ASP는 시스템 백플레인에 존재하는 여러 개의 보드 중에 테스트하고자 하는 보드를 선택하여 테스트를 할 수 있을 뿐만 아니라 보드간의 연결 테스트도 가능하다 [8, 9]. 그림 1은 ASP를 이용한 시스템 레벨의 테스트 블록 다이어그램이다. 따라서 보드 테스팅 뿐만 아니라 하드웨어 기반 디버깅 시스템, ISP, ASP등을 위해서 테스트 버스 콘트롤러 설계는 그 의의를 지닌다.

보드 테스트 혹은 유지 보수를 위한 콘트롤러 (MMC : Module test and Maintenance Controller)의 일반적인 구조는 [10]에서 제안되었다. 테스트 채널(Test Channel)이라 불리는 회로가 테스트 버스를 제어하도록 하였다. 프로세서는 테스트 채널을 통해서 보드에 테스트 패턴을 가하거나 보드로부터 결과를 읽어오도록 하였다. 그러나 최근 들어 IEEE Std 1149.1을 기반으로 하는 시스템에서는 SVF (Serial Vector Format)가 널리 쓰이고 있어서, SVF의 명령어 처리 및 내부레지스터 프로그래밍의 용이성을 고려한 테스트 버스 콘트롤러의 설계가 요구된다. 또한 최근에는 집적 기술의 발달로 시스템을 한 IC에 실현하는 SoC(System-on-a-Chip) 기술이 등장하기에 이르렀다. SoC 설계 방식에서 내장되어질 수 있도록 작은 칩의 크기와 높은 고장검출률을 가지는 설계가 요구된다. 본 논문에서 구현된 테스트 버스 콘트롤러는 SVF 명령어를 지원하며, 12k 게이트 수준의 크기를 가진다. 또한 SCAN 설계를 수행 후 ATPG에 의해 생성된 테스트 패턴에 대해여 고장 시뮬레이션을 해본 결과 98%이상의 고장검출율을 가진다.

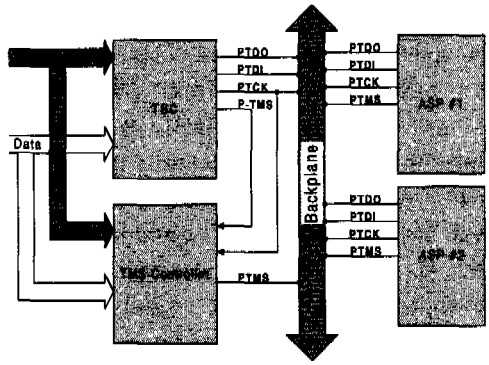


그림 1. ASP(Addressable Scan Port)를 이용한 시스템 레벨의 테스팅

본 논문의 구성은 Ⅱ장에서 테스트 버스를 제어하는 방법 및 설계 의의에 대해서 설명하고 Ⅲ장에서는 테스트 버스콘트롤러의 구조와 동작에 대해서 설명한다. Ⅳ장 Ⅴ장에서는 구현에 관해 설명하고 결론을 맺는다.

Ⅱ. 기존의 테스트 버스 콘트롤러

IEEE 1149.1이 적용된 PCB 보드에 대한 보드 레벨 테스팅 및 ISP를 위해서는 각각의 테스트에 적합한 테스트 버스 신호를 생성하여야 한다. 이런 JTAG Binary Data Stream을 생성하는 방법은 그림 2와 같다. 테스트 패턴 생성 프로그램 및 ISP 프로그램을 간단히 하고 구조화하기 위해서 테스트를 상위 레벨에서 기술하는 중간 단계의 파일들을 사용한다. 그림 2에서 보여지는 SVF 또는 JAM 파일은 그 대표적인 것이다. SVF(Serial Vector Format)는 1991년 Texas Instrument와 Teradyne에 의해서 테스트 패턴과 테스트 결과 분석용 정보 등을 표현하기 위해서 개발된 언어이며, IEEE Std

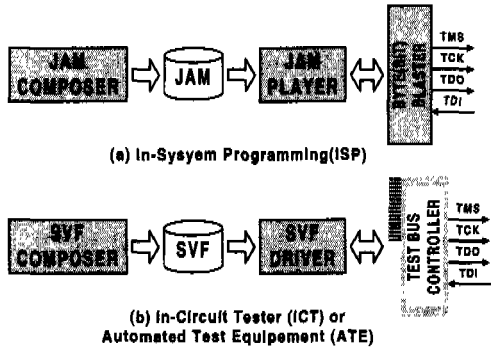


그림 2. 테스트 버스를 제어하기 위한 방법

1149.1을 기반으로 하는 테스터와 소프트웨어에서 광범위하게 채택하고 있는 테스터용 언어이기도 하다[7]. JAM은 ISP를 위한 새로운 표준 파일 형식이며 IEEE 1149.1 JTAG 인터페이스를 사용하는 ISP 디바이스를 프로그래밍 하거나 검증하기 위한 프로그래밍 및 테스팅 언어이다. 그림 2 (a)의 경우처럼 JAM 파일을 이용한 방법은 부가적인 하드웨어 장치인 테스트 버스 콘트롤러가 필요 없어 하드웨어가 간단해지는 장점이 있다. 그러나, 데이터 전송 속도 등에 한계가 있어서 ISP에만 한정되어 사용되어 진다. 그러나 그림 2 (b)의 경우처럼 SVF 파일과 테스트 버스 콘트롤러라는 부가적인 하드웨어 장치를 이용하는 방법은 PCB 상에서의 보드 레벨 테스팅 및 ISP를 지원하는 PLD들을 프로그래밍하거나 검증하는 등 여러 분야에서 응용이 가능한 이점이 있다. 따라서 거의 모든 ATE 제조회사들은 자기만의 테스트 버스 콘트롤러를 가지고 있으며 SVF 파일만 주어진다면 ATE에 내장된 테스트 버스 콘트롤러가 자동적으로 JTAG Binary Data Stream 생성하도록 하고 있다. 이런 방식을 취하는 대표적인 ATE 제조회사에는 Hewlett Packard, GenRad Inc, Teradyne, ASSET InterTech, JTAG Technologies 등이 있다.

서론에서 언급한 바와 같이 IEEE 1149.1은 하드웨어 기반 소프트웨어 디버깅 시스템에서 응용될 뿐만 아니라 그 응용은 시스템 레벨 테스팅 등 최근에도 계속하여 연구가 되고 있다. 따라서 여러 분야에 응용 가능한 독자적 테스트 버스 콘트롤러를 설계는 의의를 지닌다. 무엇보다도 기존의 제품화된 테스트 버스 콘트롤러는 System-on-a-Chip(Soc) 설계 방식에서 내장되어 질 수가 없다. 크기가 작고 고신뢰도를 가지는 테스트 버스 콘트롤러라면 범용 프로세서나 DSP(Digital Signal Processor) 등에 내장되어 보드의 유지 보수가 가능하게 된다. 따라서 SVF 호환 가능하며 크기가 작고 고신뢰도를 가지는 독자적인 테스트 버스 콘트롤러의 설계는 의의를 지닌다.

Ⅲ. 테스트 버스 콘트롤러의 설계

테스트 버스 콘트롤러는 프로세서와 테스트 버스 인터페이스를 제공한다. 또한 프로세서에 의해 제어되어지며 테스트 버스 상의 4개의 신호선들에 대한 실질적 제어 시퀀스를 생성한다. 테스트 클록은 시스템 클록과 독립적으로 사용될 수 있어서 양쪽 인터페이스가 비동기일 수 있다.

본 논문에서 설계한 테스트 버스 콘트롤러의 블록 다이어그램은 그림 3과 같다. 총 5개의 기능 블록을 가지고 8개의 16 비트 레지스터를 가지고 있다. 내부 레지스터들의 위치는 그림 3에서 나타내었다.

3.1 테스트 버스 콘트롤러 인터페이스

3.1.1 프로세서 인터페이스

프로세서와 인터페이스를 통하여 내부레지스터에 접근하게 된다. 8개의 내부레지스터에 접근하기 위하여 3 비트의 어드레스 버스가 필요하다. 프로세서 인터페이스는 다음과 같이 구성된다.

- 3-Bit 어드레스 버스 : A0 ~ A2
- 16-Bit 데이터 버스 : D0 ~ D15
- 읽기 제어 신호 : $\overline{WR}$
- 쓰기 제어 신호 : $\overline{RD}$
- 칩 선택 신호 : $\overline{CS}$

본 논문에서 구현한 IEEE 1149.1 테스트 버스 콘트롤러는 그림 4와 같이 프로세서에 연결될 수 있다. 일반적으로 프로세서와 인터페이스를 위한 보드에서는 어드레스 디코더 및 버퍼 등의 TTL로 구성된다. 어드레스 디코더는 PAL로 구현될 수 있다. 그러나 SVF의 TRST 명령어 처리 및 소프트웨어에 의한 가변적인 TCK 생성을 위해서 PAL 대신에 보드 콘트롤러(Board Controller)를 따로 설계하였다. 구현된 보드 콘트롤러는 프로세서가 접근 가능한 8-Bit 내부레지스터와 어드레스 디코더, 클록 분주기 등으로 구성된다. 보드 콘트롤러는 TRST 신호를 생성하고 어드레스를 디코딩하여 $\overline{CS}$(Chip

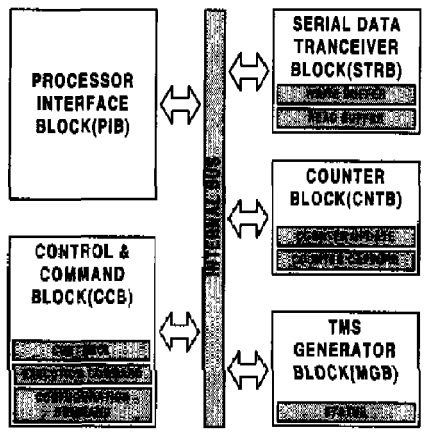


그림 3. 테스트 버스 콘트롤러의 블록다이어그램

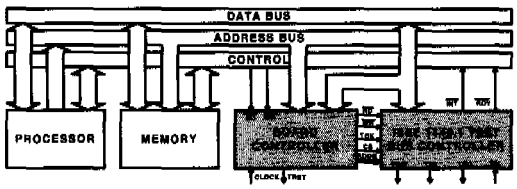


그림 4. 프로세서 인터페이스

Select) 신호를 생성한다. 또한 클록 분주기를 통해서 1Mhz~10Mhz사이의 가변적인 TCK를 생성시킬 수 있다.

프로세서와 인터페이스를 위한 쓰기/읽기 타이밍 다이어그램은 각각 그림 5, 6과 같다. 그림 5는 테스트 버스 콘트롤러의 내부 레지스터에 값을 쓸 때의 타이밍 다이어그램이다. 그림 5에서 보여지는 것처럼 데이터 버스의 값은 $\overline{WR}$ 신호의 positive edge에서 유효한 값이어야 한다. 그림 6은 테스트 버스 콘트롤러의 내부 레지스터에 값을 읽을 때의 타이밍 다이어그램이다. 그림에서 보여지는 것처럼 테스트 버스 콘트롤러 $\overline{RD}$ 신호의 negative edge에서 데이터 버스에 실을 값을 준비해야 하며, positive edge에서는 유효한 값을 가져야 한다.

3.1.2 테스트 버스 인터페이스

IEEE 1149.1 표준에 따라서 테스트 버스 인터페이스는 다음으로 구성된다.

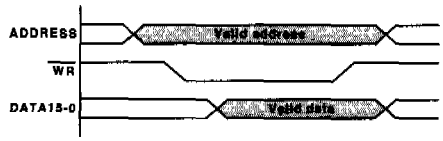


그림 5. 프로세서의 쓰기 타이밍 다이어그램

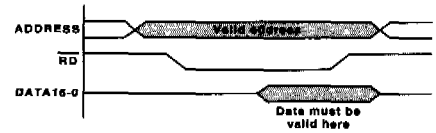


그림 6. 프로세서의 읽기 타이밍 다이어그램

- 테스트 클록 출력 : TCK
- 두 개의 테스트 데이터 입력 : TDI0 ,TDI1
- 네 개의 TMS 출력 : TMS0, TMS1, TMS2, TMS3
- 테스트 데이터 출력 : TDO
- 테스트 클록 입력 : CLOCK
- 테스트 버스를 high impedance로 보내기 위한 TOFF (TEST OFF)
- 테스트 버스 콘트롤러 및 모든 테스트 회로를 리셋하기 위한 TRST(TEST RESET) 입력

CLOCK 신호는 보드 콘트롤러에서 분주 되어 들어 올 수도 있고 시스템 클럭이 바로 들어 올 수도 있다. 이 신호는 테스트 버스 콘트롤러 및 모든 테스트 회로를 위한 테스트 클록(TCK)으로 쓰여진다. 멀티 체인을 가지는 보드를 위하여 TMS 신호를 네 개를 두었다. TDI와 각 체인의 TDO는 쉬프트 동작(SHIFT-DR 또는 SHIFT-IR 상태)에만 유효한 값을 지니고 그 외의 상태에서는 high impedance를 가지기 때문에 하나만 있으면 된다. 그러나 TDO가 버퍼링 되는 경우는 high impedance를 가질 수 없기 때문에 이런 경우를 위하여 두 개의 TDI를 두었다.

3.2 내부 레지스터

그림 3에서 보여지는 것처럼 테스트 버스 콘트롤러는 8개의 내부레지스터를 가진다. : 콘트롤 레지스터(Control Register), 설정 명령 레지스터(Configuration Command Register), 실행 명령 레지스터(Execution Command Register), 카운터 업데이트 레지스터(Counter Update Register), 카운터 캡춰 레지스터(Counter Capture Register), 상태 레지스터(Status Register), 읽기 버퍼(Read Buffer), 쓰기 버퍼(Write Buffer). 내부 레지스터 각각의 기능은 표 1에 나타내었다. 또한 각 레지스터의 각 비트의 기능은 그림 7과 같다.

3.3 기능 블록

3.3.1 프로세서 인터페이스 블록

프로세서 인터페이스 블록은 프로세서 혹은 PC와 인터페이스를 하기 위한 블록이다. 프로세서 인터페이스 블록은 시스템 버스와 인터페이스를 위한 데이터 버스 버퍼(Data Bus Buffer)와 읽기/쓰기 회로(Read/Write Logic)로 구성된다. 데이터 버스 버퍼는 16 비트 양방향 삼상 버퍼를 사용하였다. 읽기/쓰기 회로는 내부 레지스터로 접근을 위해 프로세서로부터 발생되는 어드레스를 디코딩하고, 다른 기능 블록을 제어하기 위한 제어신호를 생성한다.

프로세서 인터페이스 블록의 블록 다이어그램은 그림 8과 같다. 테스트 버스 콘트롤러 칩이 선택되고($\overline{CS}$ 신호가 '0'), 읽기/쓰기 동작이 수행 될 때($\overline{RD}$ 신호 또는 $\overline{WR}$ 신호가 '0')만 데이터

버스 버퍼를 활성화시킨다. 칩이 선택되지 않으면 ($\overline{CS}$ 신호가 '1') 내부 버스와 입력 데이터 버스는 완전히 분리된다. 세 비트의 어드레스 신호는 디코딩되어 8개의 내부레지스터를 접근 가능하게 한다.

앞서 설명한 보드 콘트롤러는 칩 선택 신호를 생성시킨다. 칩 선택 회로는 PAL등이 쓰이지만 다양한 클록의 제공과 TRST 신호를 제어하기 위하여 보드 콘트롤러를 설계하였다.

3.3.2 제어 및 명령 블록

제어 및 명령 블록의 구조는 그림 9와 같다. 3개의 내부 레지스터와 명령어 디코더 회로로 구성된다. 세 개의 내부 레지스터는 콘트롤 레지스터(Control Register), 설정 명령 레지스터(Configuration Command Register), 실행 명령 레지스터(Execution Command Register)이다. 설정 명령 레지스터는 SVF 명령을 수행하기 위하여 STATE, RUNTEST, SIR, SDR 등의 테스트 종류를 지정하고 스캔 동작의 마지막 상태를 지정한다.

표 1. 내부 레지스터의 종류와 기능

레지스터	기능	접근
콘트롤 레지스터	내부 콘트롤 환경	읽기/쓰기
설정 명령 레지스터	수행할 테스트 정의	읽기/쓰기
실행 명령 레지스터	칩의 동작을 명령하는 명령어	읽기/쓰기
카운터 업데이트 레지스터	쉬프트 할 데이터 수	읽기/쓰기
카운터 캡쳐 레지스터	현재의 카운터 값	읽기 전용
상태레지스터	칩의 상태	읽기 전용
읽기 버퍼	TDI를 위한 데이터	읽기 전용
쓰기 버퍼	TDO를 위한 데이터	쓰기 전용

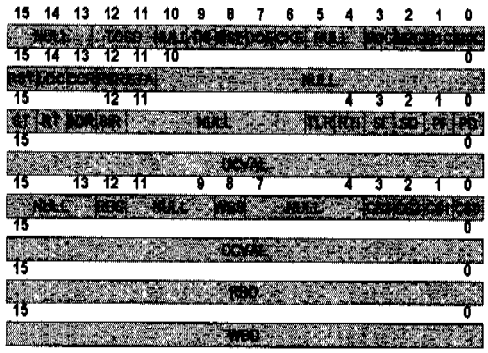


그림 7. 내부 레지스터 각 비트의 기능

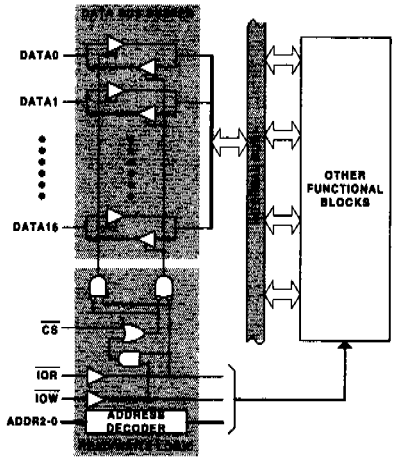


그림 8. 프로세서 인터페이스 블록

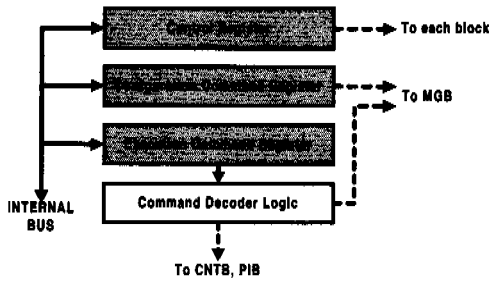


그림 9. 제어 및 명령 블록

실행 명령 레지스터에 테스트 시작이라는 명령이 들어오기 전에는 설정 명령 레지스터에 정의된 테스트를 수행하지 않는다. 실행 명령 레지스터에 테스트 시작이라는 명령을 쓰게 되면 비로소 설정 명령 레지스터에 정의된 테스트를 수행하게 되고 TMS 생성 블록 FSM이 활성화되어 실질적인 테스트가 수행된다. 명령어 디코더 회로(Command Decoder Logic)는 실행 명령 레지스터에 값이 쓰일 때마다 실행 명령 레지스터의 값을 디코딩하여 실행 명령을 수행하기 위한 제어 신호를 생성한다. 여기서 생성된 제어 신호는 설정 명령 레지스터에 정의된 STATE, RUNTEST, SDR, SIR등의 명령을 실질적으로 수행하게 하거나 테스트 버스 콘트롤러를 리셋시킨다. 또한 카운터, 카운터 캡춰 레지스터, 상태 레지스터를 갱신하게 한다.

3.3.3 카운터 블록

TAP의 16개 상태 중 데이터를 쉬프트하기 위한 상태는 SHIFT-IR과 SHIFT-DR 상태이고, BIST (Built-In Self Test)는 RUN-TEST/IDLE 상태에서

수행된다. 카운터 블록은 이런 상태의 사이클 수를 세기 위한 블록이다. 카운터 블록의 구조는 그림 10과 같다. 그림에서 보여지는 것처럼 카운터 블록은 16 비트 백워드 카운터, 카운터에 업데이트 할 데이터를 저장할 카운터 업데이트 레지스터, 진행중인 상태를 확인하기 위한 카운터 캡쳐 레지스터로 구성된다. 프로세서는 카운터 업데이트 레지스터와 카운터 캡쳐 레지스터만 접근할 수 있다.

3.3.4. 직렬데이터 송수신 블록

IEEE 1149.1이 적용된 PCB 보드에 전송될 명령어와 데이터를 공급하고, 또한 PCB 보드의 TDO로부터 나오는 데이터를 수집한다. 직렬 데이터 송수신 블록의 블록 다이어그램은 그림 11과 같다. 쓰기 버퍼, 읽기 버퍼, 쉬프트 레지스터(Shift Register) 등으로 구성된다. 프로세서는 쓰기 버퍼와 읽기 버퍼만 접근이 가능하다. 그림 11에서 보여지는 실선은 데이터의 흐름을 나타내고 점선은 제어선을 나타낸다. TDO는 '0' 또는 '1' 만 내보는 경우를 위하여 멀티플렉서로 선택을 할 수 있게 하였다.

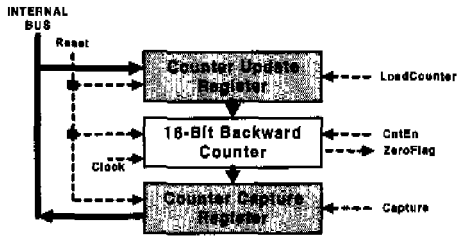


그림 10. 카운터 블록

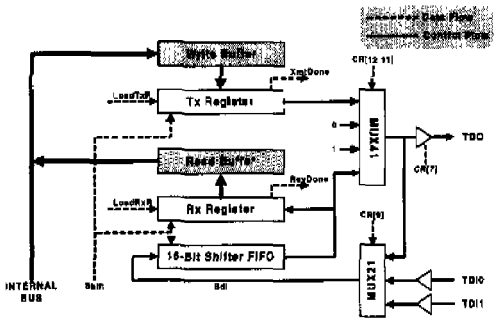


그림 11. 직렬 데이터 송수신 블록

PCB 보드로 나가는 데이터는 TCK의 negative edge에서 전송되어지고, PCB보드로부터 들어오는 데이터는 TCK의 positive edge에서 샘플링되어진다. IEEE 1149.1 테스트 버스 콘트롤러 내의 데이터 패스의 자체 테스트와 안정적인 데이터 송신을 위하여 쉬프터 FIFO를 두었다. 프로세서가 읽기 버퍼 또는 쓰기 버퍼에 접근하기 전에는 상태 레지스터를 읽어서 읽기/쓰기 버퍼의 상태를 확인한다. 읽기 버퍼 플래그와 쓰기 버퍼 플래그는 읽기 버퍼와 쓰기 버퍼에 데이터 유무의 상태를 나타낸다.

3.3.5. TMS 생성 블록

실제 TMS 신호와 칩 내의 제어 신호를 생성시키는 핵심 부분이다. 이 블록은 크게 다음 두개의 서브 블록으로 구성된다. 그 하나는 시퀀스 인식 블록(Sequence Accepter Block)이며, 또 하나는 시퀀스 생성 블록(Sequence Generator Block)이다. 시퀀스 인식 블록에는 시퀀스 생성 블록에서 생성된 TMS 신호가 들어온다. 이 TMS 신호를 참조하여 보드 내의 상태를 계속적으로 추적하게 된다. 이러한 기능을 하도록 하기 위해서 TAP 콘트롤러와 유사한 기능을 하도록 구현하였다. 이렇게 알아낸 보드 상태는 상태 레지스터(Status Register)에 기록된다. 시퀀스 생성 블록은 설정 명령 레지스터에 정의된 각 명령에 따라 그에 적절한 TMS 신호 및 제어 신호를 생성한다. 그림 12는 TMS 생성 블록의 블록 다이어그램이다.

3.4 내부 레지스터 프로그래밍 절차

프로세서는 단순히 테스트 데이터와 수행될 명령을 테스트 버스 콘트롤러의 내부 레지스터에 써주기만 하면 된다. 테스트 버스 콘트롤러는 내부 레지스터에 쓰여진 값에 따라 자동적으로 동작을 수행하게 된다. 또한 그 결과 데이터는 테스트 버스 콘트롤러의 내부 레지스터에 기록된다. 프로세서는 그 결과를 읽어들여서 그 결과를 이용하여 PCB 보드의 고장을 검출 또는 진단 등을 할 수 있게 된다.

SVF 파일을 이용한 보드 레벨 테스트 및 ISP 시스템에서는 SVF 드라이버가 SVF 파일을 입력으로 받아서 테스트 버스 콘트롤러의 내부 레지스터를 자동적으로 프로그래밍 하거나 읽어오게 된다. SVF 드라이버의 입력이 되는 SVF 파일은 상용화된 거의 모든 CAD 툴에서 생성 가능하다. 보드 테스팅의 경우는 칩 제작 시에 주어지는 각각의 칩에 대한 BSDL과 PCB보드의 네트리스트 정보(EDIF)를 이용하여 테스트 패턴 생성 프로그램에 의하여 생성되며 ISP의 경우에는 PLD 제조회사의 CAD 툴에서 각 디바이스에 맞는 SVF를 생성한다. SVF의 명령어는 크게 테스트 패턴을 정의하는 명령어와 테스트 동작에 관련된 명령어로 구분된다. 테스트

동작에 관련된 명령어는 각 레지스터에 적재하는데 필요한 scan 명령, 상태 천이를 위한 명령, BIST를 위한 명령으로 구분되어 있다. Scan 명령에는 데이터 레지스터(DR) scan을 위한 패턴을 정의하고 데이터 레지스터 scan을 수행하여 테스트 패턴을 주입하는 역할을 하는 SDR(Scan Data Register)와 명령 레지스터(IR) scan을 위한 패턴을 정의하고 명령 레지스터 scan을 수행하여 명령어를 주입하는 역할을 하는 SIR(Scan Instruction Register)로 되어있다.

표 2는 내부레지스터를 프로그래밍하기 위한 순서도이다. 테스트 데이터를 보드에 쓰고 읽는 경우(SIR, SDR)의 프로그래밍 순서이다. SVF에 정의된 테스트 중 STATE, RUNTEST의 경우의 내부 레지스터 프로그래밍 절차는 기본적으로 이와 유사하다.

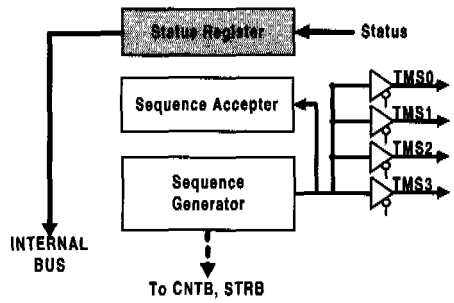


그림 12. TMS 생성 블록

표 2. 프로그래밍 절차

절차	대상 레지스터	접근	동작
1	실행 명령 레지스터	쓰기	소프트웨어 리셋
2	콘트롤 레지스터	쓰기	제어환경 설정
3	카운터 업데이트 레지스터	쓰기	데이터 수 저장
4	실행 명령 레지스터	쓰기	카운터 업데이트
5	설정 명령 레지스터	쓰기	테스트 정의
6	실행 명령 레지스터	쓰기	테스트 수행
7	상태 레지스터	읽기	상태 확인(반복)
8	쓰기 버퍼	쓰기	테스트 데이터
9	상태레지스터	읽기	상태 확인(반복)
10	읽기 버퍼	읽기	결과 데이터
7~10의 과정을 반복			

이 중에서 SIR/SDR 명령어의 경우 프로그래밍 절차를 자세히 살펴 보면 다음과 같다. 테스트 버스 콘트롤러는 테스트를 수행하기 전에는 항상 소프트웨어 리셋이 수행되어 칩이 초기화되어야 된다. 그래서 가장 먼저 실행 명령 레지스터에 소프트웨어 리셋명령을 쓴다. TMS 출력 신호의 연결 상태나 TDO로 나가는 데이터의 종류 선택을 위해서 콘트롤 레지스터에 제어환경 설정값을 쓴다.

SHIFT-DR 또는 SHIFT-IR 상태에서 쉬프트 되어질 데이터의 개수를 세기 위한 카운터 값을 카운터 레지스터에 쓴다. 카운터 값을 백워드 카운터에 업데이트하기 위해서 실행 명령 레지스터에 카운터 업데이트 명령 값을 쓴다. 설정 명령 레지스터에 원하는 테스트와 테스트 종료 후 마지막 상태를 지정해주는 값을 쓴다. 설정 명령 레지스터에 정의한 테스트를 수행하기 위해서 실행 명령 레지스터에 테스트 수행 명령 값을 쓴다. 또한 쓰기 버퍼에 데이터를 쓰거나 읽기 버퍼의 데이터를 읽기 전에는 반드시 상태레지스터를 읽어보아야 한다. 이렇게 함으로써 대상 보드는 어떤 상태에 있는지 쓰기 버퍼, 읽기 버퍼에 있는 값이 유효한지를 판단하게 된다. 그래서 상태 레지스터를 읽어보고 유효한 값이 존재하는지 확인한 다음 쓰기 버터나 읽기 버퍼에 들어있는 값을 쓰고 읽는 것을 반복하게 된다. 테스트 대상 보드가 SVF 파일의 ENDDR에 정의된 상태임을 확인함으로써 테스트가 종료됨을 알 수 있다.

Ⅳ. 실 험*

본 논문에서 구현된 테스트 버스 콘트롤러를 이용하여 실제 보드에 적용을 하기 위해서 ATPG (Automatic Test Pattern Generator)툴을 이용하여 테스트 입력 백터인 SVF파일을 생성하였다. ATPG 툴은 IEEE Std 1149.1에 근거한 boundary scan 규정을 따라서 설계된 보드에 대하여, boundary scan cell 사이의 연결구조를 테스트하는데 필요한 테스트 패턴을 자동으로 생성하게 하는 툴이다. ATPG 툴은 보드 데이터 파일과 보드 정보 파일을 입력으로 받아 여러 가지 테스트에 필요한 테스트 패턴을 자동으로 생성해서 SVF파일로 생성하게 된다. 그림 13은 ATPG툴의 블록 다이어그램을 나타낸 것이다. 테스트 버스 콘트롤러는 이 SVF파일을 SVF드라이버 프로그램을 통해서 입력으로 받아 드리게 된다. SVF드라이버 프로그램은 3.4에서 기술한 레지스터 프래그래밍 절차에 맞게 구현되어진 프로그램이다. 테스트 버스 콘트롤러는 SVF에 기술된 명령을 수행하고 테스트 데이터를 테스트 보드에 보내고 테스트 결과 파일을 생성하게 된다. 생성된 결과 파일을 입력으로 받아 테스트 보드에 어떤 결함이 있는

지를 진단 툴을 이용하여 분석하고 테스트를 마치게 된다. 진단 툴은 진단을 위해 필요한 정보와 결과 파일을 입력으로 받아 진단 알고리즘에 따라 진단을 수행하고 그 결과를 출력하게 된다. 그림 14는 진단 툴의 블록 다이어그램을 나타낸 것이다. 우리는 이 모든 툴을 직접 구현하여 테스트를 성공적으로 수행하였다.

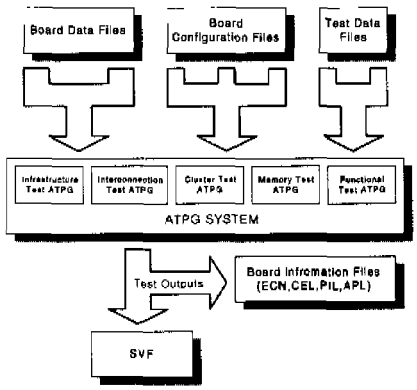


그림 13. ATPG툴의 블록 다이어그램

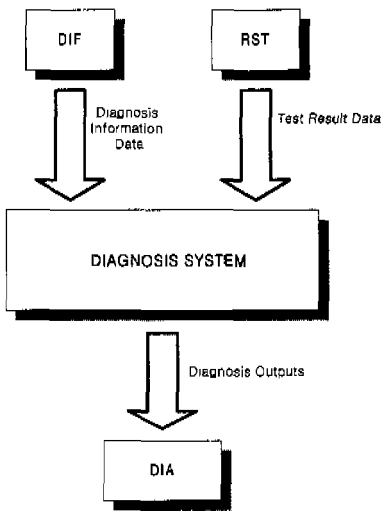


그림 14. 진단 툴의 블록 다이어그램

V. 결 론*

본 논문에서는 IEEE 1149.1 기반 테스트 시스템을 위한 테스트 버스 콘트롤러를 설계하였다. 이 테스트 버스 콘트롤러의 기능은 프로세서와 인터페이스를 통하여 IEEE 1149.1 테스트 버스를 제어한다. 본 테스트 버스 콘트롤러의 특징은 SVF에 정의된 테스트를 수행 가능하며 내부레지스터 프로그래밍이 용이하도록 하였다. 또한 범용 프로세서나 DSP 칩에 내장되어 보드레벨 테스트 및 ISP를 수행할 수 있도록 하기 위하여 고신뢰도를 가지고 크기가 작도록 설계되었다.

* 본 연구의 실험에서 사용한 S/W 및 H/W는 부분적으로 IDEC의 지원에 의한 것임

참 고 문 헌

[1] IEEE Standard Test Access Port and Boundary Scan Architecture, IEEE Std. 1149.1-1990, *IEEE Press*, New York, 1990

[2] Miron Abramovici, Melvin A. Breuer, Arthur D. Friedman, "Digital System Testing and Testable design", *IEEE press*, pp. 395-412, 1990

[3] Colin M. Maunder, Rodham E. Tulloss, "The Test Access Port And Boundary Scan Architecture", *IEEE Computer Society Press*, pp.23-28, 1990

[4] James Gately, Miriam Blatt, Dennis Chen, and Scott Cooke. UltraSPARC-I Emulation, DAC 1995

[5] Steve Furber, "ARM System Architecture", Addison Wesley, pp. 219-241, 1996

[6] Dave Bonnett, In-System Programming: A Major New Application of Boundary Scan, ASSET InterTech, Inc.

[7] Serial Vector Format Specification Revision D, ASSET InterTech, Inc.

[8] Lee, Whetsel, "A Proposed Method of Accessing 1149.1 in a Backplane Environment", *International test conference*, pp. 206-216, 1992

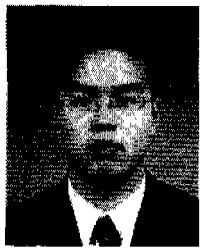
[9] Lee, Whetsel, "Hierarchically Accessing 1149.1 Applications in a System Enviroment", *Proc. of International Test Conference*, pp. 517-526, 1993

[10] Jung-Cheun Lien, Melvin A. Breuer, "A Universal Test and Maintenance Controller for Modules and Boards", *IEEE Transactions on Industrial Electronics*, 1989

조 용 태(Yong Tae Cho) 정회원

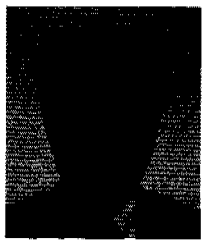
1998년 2월 : 중앙대학교 제어계측공학과 학사
2000년 2월 : 중앙대학교 제어계측공학과 석사
<주관심 분야> VLSI시스템 설 계 및 테스트

정 득 수(Deuksoo Jung) 정회원



1999년 2월 : 중앙대학교 제어계측공학과 학사
1999년 2월~현재: 중앙대학교 제어계측공학과 석사과정
<주관심 분야> VLSI시스템 설계 및 테스트, Channel Coding

송 오 영(Ohyoung Song) 정회원



1980년 2월 : 서울대학교 전기공학과 학사
1982년 2월 : 한국과학기술원 전기 및 전자공학과 석사
1992년 2월 : University of Massachusetts at Amherst, 전기 및 컴퓨터공학과 박사
1982년 3월~1985년 5월 : 국방부 기술연구 사무관
1991년10월~1992년 10월 : Intergraph Corp. Electronics 수석연구원
1992년 1월~1993년 11월 : IBM Microelectronics 수석연구원
1994년 1월~1994년 8월 : 삼성전자 LSI사업부 수석연구원
1994년 9월~현재 : 중앙대학교 전자전기공학부 부교수
<주관심 분야> VLSI시스템 설계 및 테스트