JTAG 기반 SoC의 개선된 온 칩 디버깅 유닛 설계

준회원 윤 연 상*, 류 광 현*, 정회원 김 용 대*, 한 선 경*, 유 영 갑*

Advanced On-Chip Debugging Unit Design for JTAG-based SoC

Yeon sang Yun*, Kwang hyun Ryoo* *Associate Members*
Yong dae Kim*, Seon kyoung Han*, Young gap You* *Regular Members*

요 약

JTAG 기반 SoC의 디버깅 성능향상을 위한 온 칩 디버깅 유닛(On-chip debugging unit)을 제안하였다. 제안된 디버깅 유닛은 JTAG 모듈, 코어브레이커로 구성된다. JTAG 모듈은 기존의 IEEE 1149.1 표준을 변형하여 효율적으로 설계하였다. SoC 시스템의 집적도가 높아질수록 1회의 디버깅 사이클을 실행하기 위한 반복적인 TAP 명령의 인가가 예상된다. 제안된 디버깅 유닛이 TAP 명령 인가과정의 불필요한 클럭 소모를 최소화하였다. 성능분석 결과 기존의 방식과 비교하여 14% 정도의 디버깅 성능의 증가를 보였고 TAP 컨트롤러 회로의 게이트 수는 50% 정도 감소하였다.

Key Words : SoC debugging; IEEE 1149.1; JTAG; TAP controller.

ABSTRACT

An on-chip debugging unit is proposed aiming performance enhancement of JTAG-based SoC systems. The proposed unit comprises a JTAG module and a core breaker. The IEEE 1149.1 standard has been modified and applied to the new JTAG module. The proposed unit eliminates redundant clock cycles included in the TAP command execution stage. TAP execution commands are repeatedly issued to perform debugging of complicated SoC systems. Simulation on the proposed unit shows some 14% performance enhancement and 50% gate count reduction compared to the conventional ones.

Ⅰ. 서 론

SoC 설계 기술의 발전으로 전체 시스템 개발시간이 단축되고 있지만 전체 개발시간 중 디버깅이 차지하는 비중은 증가하고 있다. 이는 시스템 개발시 디버깅 및 테스트에 소비되는 금액이 총 개발비의 50% 내지 70%를 차지한다는 한 조사결과에 의해 입증되었다[1]. 시스템 설계는 점차 보드 수준(board level)의 설계에서 칩 수준(chip level)의 설계로 집적화되고 있다. 테스트 기법 또한 기존의 bed-of-nails와 같은 보드 수준의 방식이 아닌 JTAG 기반 방식이 널리 사용되고 있다[2-4]. JTAG 기반 테스트 방식은 온 칩 디버깅 유닛(이하 디버깅 유닛이라 함)을 타겟 프로세서에 탑재하는 구조를 갖는다. 이 구조는 타겟의 출력만을 모니터링 하는 것이 아니라 테스트 입력의 인가 및 타겟의 파이프라인 실행 등, 디버깅 기능을 실행하기 용이한 장점을 갖는다[5,6].

현재 JTAG 기반 디버깅 시스템은 호스트의 디버거 프로그램과 연동하여 사용되고 있다. 따라서 호스트에서 제공되는 병렬 또는 직렬 데이터 전송속도가 전체 디버깅 성능의 제약요소가 된다. 일례로

* 충북대학교 정보통신공학과 통신회로 및 시스템 설계 연구실 (ysyun@hbt.chungbuk..ac.kr)
논문번호 : KICS2004-05-008, 접수일자 : 2004년 5월 14일
※본 연구는 사업자원부의 지역혁신 인력양성사업의 연구결과로 수행되었다.

400Mbit/s 전송속도의 PC 인터페이스를 통해 400MHz로 동작하는 고속 프로세서의 출력을 실시간으로 모니터링하고자 한다면 프로세서의 출력 중 1비트 관측이 가능하다. 32비트의 출력을 실시간으로 감지하기 위해서는 10Gbit/s의 전송속도가 요구되며 이는 PC 인터페이스 구조의 혁신적 변화가 있기 전까지는 불가능하다. 디버깅 성능을 향상시키기 위한 노력은 계속되고 있다. 근래의 연구에서는 디버깅 시스템을 변형하거나 디버깅 유닛의 하드웨어 사이즈를 줄임으로써 디버깅 성능향상을 꾀하였다[7,8].

본 논문에서는 기존의 IEEE 1149.1에서 제시한 JTAG 모듈을 효율적으로 변형함으로써 디버깅 성능 향상 및 회로사이즈를 감소시켰다. 본 논문의 구성은 다음과 같다. II장은 JTAG 기반 테스트의 전반적 개요 및 기존의 디버깅 유닛에서 발견된 개선사항을 제시하였다. III장에서는 본 논문에서 제안한 디버깅 유닛을 설명하였다. IV장은 본 연구를 위해 테스트 목적으로 설계된 프로세서에 제안된 디버깅 유닛을 탑재하였을 경우의 성능을 분석하였다. 끝으로 V장에서 본 논문의 결론을 맺었다.

II. 디버깅 시스템

본 장은 JTAG 기반 디버깅 시스템의 전반적인 개요를 다루었다. 또한 기존의 디버깅 유닛의 분석을 통하여 개선사항을 제시하였다.

2.1 디버깅 시스템

일반적인 디버깅 시스템은 그림 1과 같이 호스트와 프로토콜 변환기 그리고 테스트 타겟으로 구성된다. 호스트는 사용자를 위한 GUI를 통한 디버깅 기능을 제공한다. 프로토콜 변환기는 호스트와 타겟 사이의 인터페이스를 담당하며 최종적으로 TAP(test access port)을 통해 타겟으로 데이터를 전송한다. TAP은 TCK(test clock), TMS(test mode select), nTRST(test reset), TDI(test data input) 그리고 TDO(test data output) 핀으로 구성된다[5].

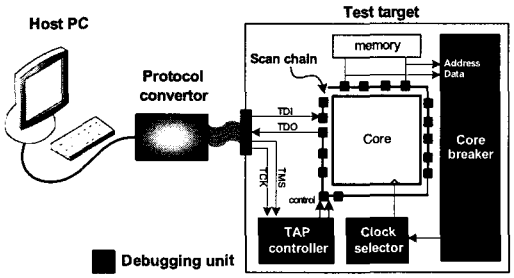


그림 1. 디버깅 시스템

테스트 타겟 내부는 주요 프로세싱을 담당하는 코어 뿐 아니라 디버깅 유닛을 포함한다. 다시 디버깅 유닛은 JTAG 모듈과 코어 브레이커로 구성된다. JTAG 모듈은 TAP 컨트롤러와 스캔체인으로 구성된다. 코어의 입출력을 TDO 핀으로 시프트하기 위해 설치된 스캔체인은 TAP 컨트롤러에 의해 제어된다. 코어 브레이커는 코어를 정지시킬 시점을 미리 프로그램 할 수 있도록 한다. 만약 코어가 정상동작을 수행하던 중 코어 브레이커의 내부에 미리 프로그램 된 주소에서 명령을 패치하려 한다면 코어는 정지된다. 정지된 코어는 TAP 컨트롤러의 제어에 의해 파이프라인을 진행하기 된다. 정상 동작시의 시스템 클럭을 디버깅 클럭으로 전환시키는 과정은 클럭 선택기(clock selector)에서 이루어진다.

코어와 디버깅 유닛을 이용한 전체 디버깅 순서를 그림 2에서 나타내었다. 코어 브레이커에 프로그램 된 주소는 정상모드에서 메모리의 주소 값과 비교된다. 비교 결과 '일치(match)'가 이루어지면 시스템은 디버깅 모드로 진입된다. 디버깅 모드에서 TAP 컨트롤러는 스캔체인 및 코어의 클럭을 제어하여 테스트 입력의 인가, 저속의 내부 동작 실행 그리고 실행결과의 출력 시프트의 3단계 과정을 거쳐 1회의 디버깅 사이클을 실행하도록 한다.

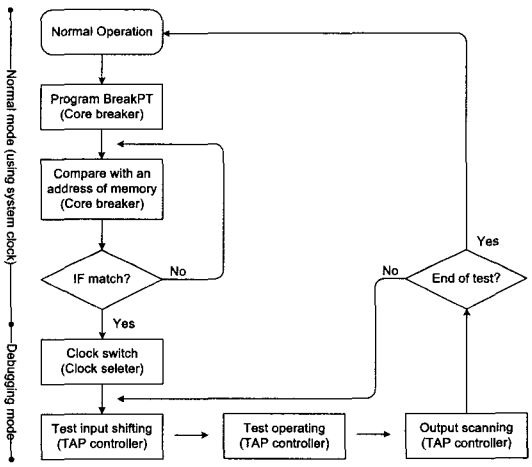


그림 2. 임베디드 시스템의 디버깅 순서

2.2 기존의 TAP 컨트롤러의 개선사항 도출

기존의 TAP 컨트롤러는 TMS와 TCK 신호를 입력으로 하는 16개의 유한상태머신(finite state machine)으로 설계할 수 있으며 그림 3의 상태도에

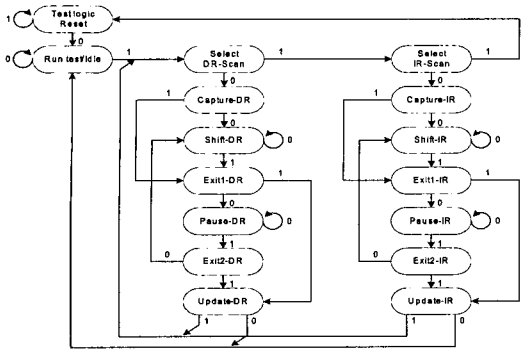


그림 3. 기존의 TAP 컨트롤러 상태도[5]

따라 제어신호를 출력한다. 각각의 상태는 TCK가 '0'에서 '1'로 변할 때 천이되고 TMS 신호에 따라 경로가 결정된다.

IR(instruction register)은 4비트의 TAP 명령들을 명령 디코더(TAP instruction decoder)로 전달하기 위한 4비트 시프트 레지스터로 구성된다. 명령 디코더는 DR 선택신호를 생성한다. DR 선택신호는 TAP 명령에 따라 어떤 스캔체인의 데이터를 TDO로 시프트 할 것인지를 결정하는 신호이다. TAP 컨트롤러의 상태에 따른 출력은 그림 4와 같다. TAP 컨트롤러의 출력신호들은 IR 및 DR 각각의 스캔 셀을 제어한다.

4비트의 TAP 명령이 명령 디코더로 인가되는 과정을 TAP 명령 인가과정이라고 정의하였다. 기존의 TAP 컨트롤러의 제어에 의하여 실행되는 TAP 명령 인가과정을 그림 5의 타이밍도로 나타내었다. 기존의 TAP 컨트롤러는 clockIR, updateIR의 신호를 출력하여 IR의 스캔 셀을 제어한다. 4비트의 TAP 명령을 디코더로 저장하기 위해 총 10클럭의 시간이 요구된다. 일반적으로 시프트 되어 입력된 4비트

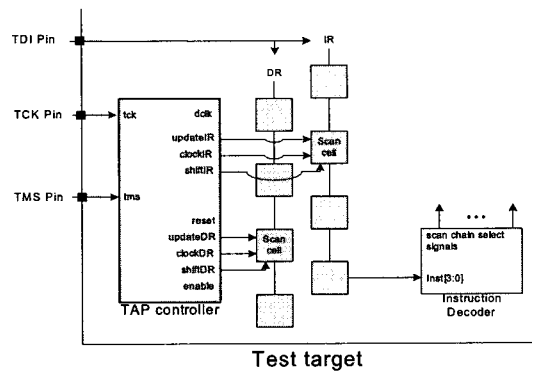


그림 4. 기존의 TAP 컨트롤러의 출력신호

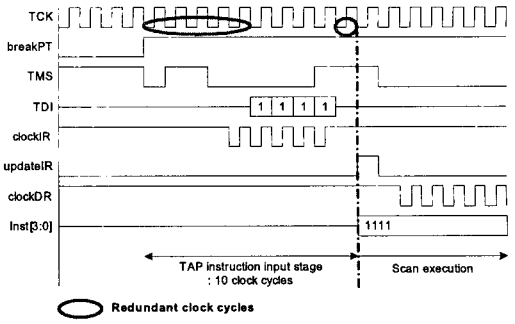


그림 5. 기존의 TAP 컨트롤러의 제어 타이밍도

의 데이터를 레지스터에 저장하기 위한 클럭 회수는 5회이다. 기존의 TAP 컨트롤러는 내부 상태천이 과정으로 인하여 5클럭이 소모되었음을 확인하였다.

Ⅲ. 제안된 TAP 컨트롤러

기존의 TAP 컨트롤러를 이용한 TAP 명령 인가과정의 경우 불필요한 클럭 소모가 발생됨을 확인하였다. 본 절에서는 이를 해결하기 위하여 개선된 TAP 컨트롤러를 제안하였다. 그리고 제안된 TAP 컨트롤러로 대체되면서 기존의 TAP 명령 인가과정을 위해 요구되는 보상 회로를 제안하였다.

3.1 개선된 TAP 컨트롤러

TAP 명령어 저장과정에서의 불필요한 클럭 소모를 제거하기 위한 개선된 TAP 컨트롤러의 상태도는 그림 6과 같다. 개선된 TAP 컨트롤러를 위해 기존의 TAP 컨트롤러가 갖는 IR을 제어하기 위한 상태를 모두 제거하였다. 이로 인해 기존의 방식과 다른 두 가지 변화가 발생한다. 첫째로 기존의 상태도에서 TMS 신호가 TCK의 5클럭 이상 HIGH를 유지하면 임의의 상태에서 Test logic Reset 상태로 천이되었지만 개선된 상태도에서는 TCK의 4클럭 이상이면 Test logic Reset 상태로 천이가 가능하다. 이는 사용자의 요구에 의해 좀 더 빨리 디버깅 시스템을 초기화할 수 있음을 의미한다. 두 번째 변화는 Run test/Idle 상태를 반복하는 최단 주기의 변화이다. 기존의 상태도에서 4클럭이었던 최단 주기는 제안된 상태도를 이용할 경우 3클럭으로 감소하였다. Run test/Idle 상태는 테스트(또는 디버깅) 모드에서 타겟 코어를 실행하기 위한 테스트 클럭이 생성되는 시점이다[6]. 제안된 상태도에서는 Run test/Idle 상태의 반복주기가 감소되었으므로 테스트 클럭의 속도가 25% 증가한 결과를 낳았다.

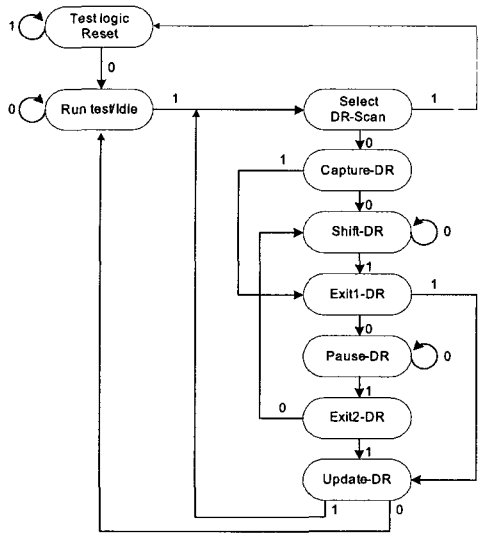


그림 6. 개선된 TAP 컨트롤러의 상태도

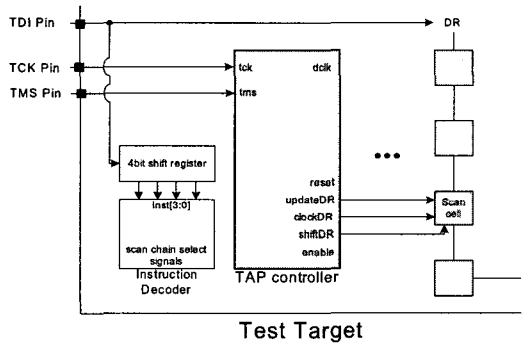


그림 7. 개선된 TAP 컨트롤러의 출력신호

기존의 TAP 컨트롤러의 출력신호 중 IR을 위한 제어신호는 그림 7과 같이 모두 제거되었다. 또한 IR 대신 4비트 시프트 레지스터로 대체되었으며 기존의 TAP 명령디코더는 유지하였다.

제안된 시스템에서 TMS 신호는 TDI로 입력되는 데이터가 TAP 명령임을 나타내는 이네이블 신호역할을 한다. TMS 신호가 '1'을 유지하고 TCK가 연속 4회 이상 '0'에서 '1'을 반복하는 경우 TAP 컨트롤러는 리셋상태가 된다. 이 때 별도로 설치된 mod4 카운터는 기존의 updataIR 신호를 대신하여 update 신호를 발생시켜 시프트 되어진 TAP 명령을 명령레지스터로 저장하게 된다. 개선된 TAP 명령 인가과정은 5 클럭 내에 수행될 수 있다. 이는 기존의 시스템과 비교하여 5클럭을 절약한 결과이다.

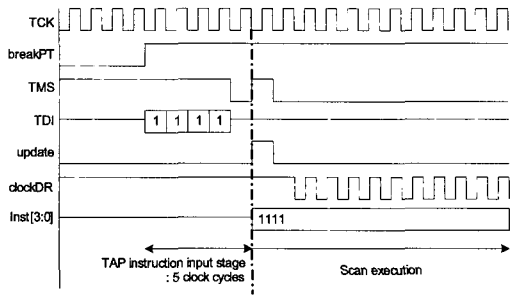


그림 8. 개선된 TAP 컨트롤러의 제어 타이밍도

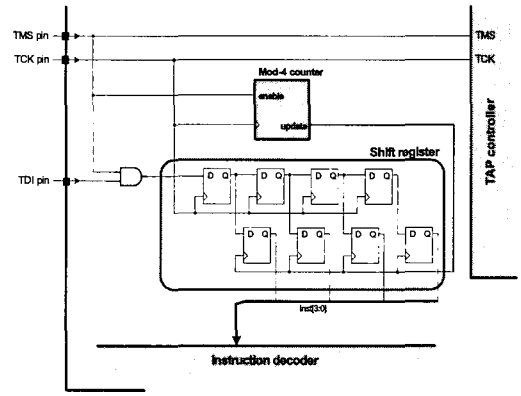


그림 9. 개선된 TAP 명령인가 회로

3.2 개선된 TAP 명령 인가 회로

개선된 TAP 컨트롤러를 이용하는 구조에서는 IR이 제거되었으므로 TAP 컨트롤러의 제어 없이 자체적으로 TAP 명령인가과정을 수행하여야 한다. 이 과정을 수행하기 위한 보상 회로를 그림 9에서 나타내었다. 4비트의 TAP 명령은 기존의 방식과 같이 TDI 핀으로 입력된다. TAP 컨트롤러의 Shift-IR 상태에서 TDI 입력이 TAP 명령임을 인식하였지만 개선된 구조에서는 TAP 컨트롤러의 Test logic Reset 상태에서 TAP 명령임을 인식하도록 한다. 즉, 리셋 상태에서 JTAG 모듈이 초기화 상태로 머무는 동안 TAP 명령은 TDI 핀을 통해 디코더로 저장된다. 제안된 구조에서 TMS 신호는 TDI 핀으로부터 TAP 명령을 시프트하기 위한 이네이블 신호 역할을 한다. TAP 컨트롤러 리셋 상태가 TCK의 4클럭 동안 지속된다면 mod4 카운터는 update 신호를 발생시키고 이 신호로부터 동기화되어 시프트레지스터에 저장된 TAP 명령이 명령 디코더로 저장된다.

Ⅳ. 데모시스템

본 절은 개선된 TAP 컨트롤러를 갖는 디버깅 유닛의 설계 결과를 설명하였다. 디버깅을 위한 타겟 프로세서는 본 연구를 위해 제작된 코어이다. 본 논문에서는 제안된 디버깅 유닛을 코어 위에 탑재한 데모시스템(demonstration system)을 구현하였고 기존의 테스트 시스템과의 비교를 통해 디버깅 성능을 분석하였다.

성능분석을 위한 데모시스템은 그림 10과 같다. 테스트 코어는 본 연구실에서 제작된 32비트 RISC 코어인 CB32TC를 이용하였다[9]. 코어가 정상모드로 동작되고 있을 때, 디버깅 시점을 코어브레이커로 프로그램 한다. 이때의 디버깅 시점은 32비트의 메모리 주소 값이다. 주소 값은 TDI로 입력되어 스캔체인 2번을 통해 코어브레이커로 입력된다. 그 다음 테스트 코어의 명령 패치단계에서 코어브레이커로 프로그램 해 놓았던 주소의 명령을 참조하고자 한다면 코어브레이커는 일치되었다는 표시로 breakPT 신호를 HIGH 상태로 만든다. 이제 테스트 코어의 클럭은 TAP 컨트롤러에서 출력되는 dclk으로 전환된다. Dclk 신호는 TAP 컨트롤러가 Run test/Idle 상태일 경우에만 HIGH을 나타낸다. 즉, 이 시점부터 코어는 TAP 컨트롤러에서 발생시키는 dclk에 트리거되어 코어의 연산을 수행한다. 스캔체인 0과 1은 각각 코어로 테스트 명령을 인가하고 코어로부터 출력을 확인하기 위해 설치하였다.

데모시스템은 표 1의 TAP 명령을 지원한다. IDCODE와 BYPASS는 각각 ID 레지스터와 bypass 레지스터로 TDI와 TDO를 연결하게끔 한다. EXTEST는 제시된 시뮬레이션 회로의 스캔체인 0을 TDI와 TDO로 연결하여 내부 스캔체인의 연결

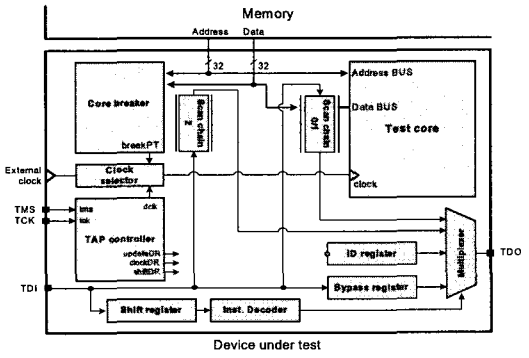


그림 10. 데모시스템 : 디버깅 유닛을 탑재한 테스트 코어

표 1. TAP 명령 및 기능

TAP 명령어	Binary	기 능
IDCODE	0000	타겟의 고유 ID 확인
BYPASS	0001	지정된 타겟의 테스트 생략
EXTEST	0010	외부시스템과의 입출력상태 확인
SC1	0011	스캔체인 1의 데이터를 TDO로 출력
SC0	0100	TDI로부터 스캔체인 0 선택
SC2	0101	TDI로부터 스캔체인 2 선택

Debugging step	Test vector	Description
Step 1. Break point programming	A. TDI << 0101 B. TDI << 0x00000004	A. Select scan chain 2 B. Shift 32 bits address
Core halt		
Step 2. Input test instructions	A. TDI << 0100 B. TDI << 0xC0000001	A. Select scan chain 0 B. LDR R0, x"00000001"
Step 3. Execute pipeline	TAP controller - Repeat Run test/Idle state	Input dclk to the test core
Step 4. Output scanning	TDI << 0011	Select scan chain 1

그림 11. 디버깅 절차에 따른 입력 테스트 벡터

상태를 확인한다. 이상의 명령들은 IEEE 1149.1에서 제시한 필수 명령이며 SC0, SC1, SC2 명령은 본 데모 시스템에 적용하기 위한 명령이다. SC1은 테스트 코어의 데이터버스의 32비트 데이터를 시프트 하여 출력하기 위한 명령이다. SC0과 SC2는 TDI를 통해 입력되는 데이터를 각각 코어의 데이터 버스와 코어브레이커로 인가하기 위한 명령이다.

데모시스템의 성능분석을 위한 테스트 벡터의 인가순서는 그림 11과 같다. 제시된 테스트 벡터는 코어가 메모리의 0x04번지 명령까지 실행한 뒤, 0x01번지의 데이터를 확인하는 목적을 갖는다. 단계 1에서는 TDI 핀으로 SC2 명령(0101)을 인가한 뒤 0x04번지 주소를 프로그램 한다. 0x04번지까지 실행을 끝낸 후 코어는 정지되고 0x05번지의 명령이 아닌 테스트 명령(0xC1000001)이 코어로 인가된다(단계 2). 제시된 테스트 명령은 코어로 하여금 0x01번지의 데이터를 레지스터 R0로 로드한다(단계 3). 이 때 0x01 번지의 데이터는 메모리버스로 이동되고 이 데이터를 TDO로 시프트 시킨다(단계 4).

개선된 디버깅 유닛의 성능분석을 위하여 테스트 코어에 기존의 디버깅 유닛을 장착한 뒤 시뮬레이션을 통해 결과를 확인하였다. 본 연구에서 제안된 회로는 HDL을 이용하여 설계하였으며 Modelsim 5.7d로 회로의 동작을 검증하였다. 제시된 그림 12

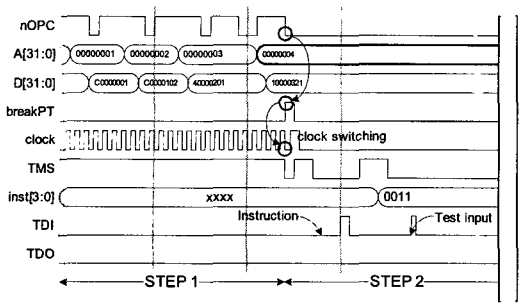


그림 12. 시뮬레이션 결과

의 파형은 기존의 디버깅 유닛을 장착한 테스트 코어의 디버깅 과정 중 단계 1과 단계 2의 일부만을 나타낸다.

전체 시뮬레이션 결과는 표 2에서 요약하였다. 단계 1에 해당되는 디버깅 시점을 프로그램 하는 과정은 성능분석에 포함시키지 않았다. 단계 2와 단계 4에서는 개선된 디버깅 유닛을 이용함으로써 TAP 명령 인가과정에서 기존의 방식에 비해 5 클럭 절약할 수 있었다. TAP 컨트롤러의 상태 축소로 인해 부수적으로 Run test/Idle 상태의 반복주기가 1 클럭 감소되었다. 이로 인해 파이프라인의 실행을 위해 dclk을 생성하는 주기도 1 클럭으로 감소하였으며 그 결과, 단계 3에서 모두 6 클럭이 절약되었다. 개선된 디버깅 유닛을 이용하였을 경우 기존의 방식에 비해 총 14%의 성능증가를 보였다.

표 2. 전체 시뮬레이션 결과

(단위 : 클럭회수)

디버깅 단계	단계 2	단계 3	단계 4	전 체	성능증가
기존의 방식	45	25	47	117	100%
개선된 방식	40	19	42	101	114%

표 3. 회로 합성 결과

(단위 : 게이트 수)

	테스트 코어	디버깅 유닛			Cell library
		코어 브레이커	TAP 컨트롤러	스캔 체인	
기존의 방식	14,363	25	47	117	Equivalent gate count with Xilinx library
개선된 방식	14,363	19	42	101	

개선된 디버깅 유닛의 회로 합성 결과를 표 3에서 정리하였다. 회로합성 결과는 기존의 디버깅 유닛과 비교할 때 TAP 컨트롤러와 스캔체인에서 차이를 보였다. TAP 컨트롤러는 50% 정도 게이트 수가 감소하였다. 반면 스캔체인의 경우 보상회로의 추가로 인해 3%의 하드웨어 게이트 수가 증가하였다.

V. 결 론

기존의 TAP 컨트롤러를 포함하는 디버깅 유닛은 TAP 명령 인가과정에서 불필요한 클럭 소모의 원인이 되었다. 제안된 디버깅 유닛은 이를 최소화하기 위하여 기존의 IR을 제거하였고 그 결과 TAP 컨트롤러의 동작상태가 절반으로 감소하였다. 데모 시스템에 적용한 성능분석결과 개선된 디버깅 유닛은 32비트 RISC 코어일 경우 14%의 디버깅 속도 향상을 나타냈다. 또한 TAP 컨트롤러 회로의 합성결과 게이트 수는 기존에 비해 50% 정도 감소하였다.

SoC 시스템은 하나의 칩 내부에 많은 종류의 코어 IP들을 탑재한다. 이로 인해 각각의 IP들의 동작을 검증하기 위한 테스트 요구가 증가하고 따라서 코어 주변의 스캔체인 수가 많아지게 된다. 이러한 스캔체인을 제어하기 위한 TAP 명령 역시, 비트수가 커지거나 4비트의 고정된 길이를 유지하며 단계화 될 수밖에 없다. 이미 ARM의 경우 스캔체인 선택레지스터(scan chain select register)를 개발하여 TAP 명령의 단계화를 시작하였다. TAP 명령이 단계화 된다면 한 번의 디버깅 사이클을 위하여 더욱 반복적으로 TAP 명령이 인가되어야한다. 이러한 시점에서 제안된 디버깅 유닛이 TAP 명령 저장과정에서의 불필요한 클럭 소모를 최소화 하였다는 점을 감안할 때, 점차 그 효율성이 부각되리라는 전망이다.

참 고 문 헌

[1] B. Hailpern, P. Santhanam, "Software debugging, testing, and verification," *IBM Systems Journal*, vol. 41, pp. 4-12, 2002

[2] A.J. Albee, M. Ellis, "Basic boundary-scan for in-circuit test," *IEEE Proceedings of ETC 1994*, pp. 349-354, Apr. 1993

[3] K. P. Parker, *The Boundary Scan Handbook*, Kluwer Academic Publishers, 2003

[4] J. Beck, R. Rose, "Integrated test logic for

video IC's," *IEEE Proceedings of Test Conference 1998*, pp. 744-751. Sept. 1998

[5] IEEE Std 1149.1-2001, Test Port and Boundary-Scan Architecture, IEEE, 2001

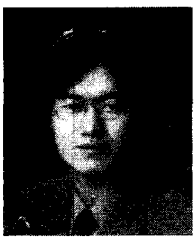
[6] Advanced RISC Machines, ARM7TDMI Data Sheet, Document Number: ARMDDI 0029E, http://www.arm.com

[7] I. Huang, C. Kao and H. Chen, "A retargetable embedded in circuit emulation module for microprocessors," *IEEE Design & Test of Computers*, vol. 19, pp. 28-38, Aug. 2002

[8] C. MacNamee, D. Heffernan, "Emerging on-chip debugging techniques for real-time embedded systems," *IEE Computing & Control Eng.*, vol. 11, no. 6, pp. 295-303, Dec. 2000

[9] 윤연상, 최종화, 김용대, 한선경, 유영갑, "재이식 가능한 32비트 테스트코어 설계," *컴퓨터정보통신연구소 논문지*, 제 12권, 제 1호, 9-15쪽, 2004년 5월

윤 연 상(Yeon sang Yun) 준회원



2004년 2월 충북대학교 전기전자공학부 학사

2004년 3월~현재 충북대학교 정보통신공학과 석사과정

<관심분야> 디지털 회로설계 및 테스트, 임베디드 프로그래밍, 암호 시스템

류 광 현(Kwang hyun Ryoo) 준회원

1998년 2월 충북대학교 정보통신공학과 학사

2005년 2월 충북대학교 정보통신공학과 석사

<관심분야> 전자공학, 이미지 처리, 암호 시스템

김 용 대(Yong dae Kim) 정회원

1990년 2월 충북대학교 정보통신공학과 학사

1993년 2월 충북대학교 컴퓨터공학과 석사

1989년~1998년 신흥기술연구소 팀장

2000~현재 충북대학교 정보통신공학과 박사과정

<관심분야> Computer arithmetic, ASIC 설계, 암호 시스템

한 선 경(Seon kyoung Han) 정회원

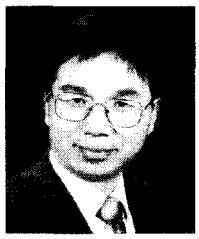
1991년 2월 충북대학교 정보통신공학과 학사

1993년 2월 충북대학교 정보통신공학과 석사

2004년 8월 충북대학교 정보통신공학과 박사

<관심분야> Computer arithmetic, Cryptographic system, ASIC 설계

유 영 갑(Young gap You) 정회원



1975년 서강대학교 전자공학과 학사

1975~1979년 국방과학연구소 연구원

1981년 Univ.of Michigan, Ann Arbor 전기전산학과 석사

1986년 Univ.of Michigan, Ann Arbor 전기전산학과 박사

1986~1988년 금성반도체 (주) 책임 연구원

1993~1994년 아리조나 대학교 객원교수

1998~2000년 오레곤 주립대학교 교환교수

1988~현재 충북대학교 정보통신공학과 교수

<관심분야> VLSI 설계 및 테스트, 고속 인쇄회로 설계, 암호학