2000年度 大韓電子工學會 秋季綜合學術大會 論文集 第23卷 第2號 2000/11

AE32000의 기능 검증

이 종 욱, 오 형 철
고려대학교 전자정보공학과
전자우편 : {rabbi,hyeong}@atlas.korea.ac.kr

Functional Verification of AE32000

Jong-Wook Lee, Hyeong-Cheol Oh
Dept. of Elec. & Info. Engineering, Korea University
E-mail : {rabbi,hyeong}@atlas.korea.ac.kr

Abstract

This paper presents a technique used for verifying the design of AE32000, a 32-bit microprocessor core. We follow the commonly used verification procedure while speeding up and completing the debugging process by adopting a reverse engineering scheme.

Ⅰ. 서론

최근 공정 기술과 회로 기술의 급격한 발달로 고성능 마이크로프로세서의 출현이 가능케 되었으나, 마이크로프로세서의 성능이 향상될수록 더욱 증가하게 되는 하드웨어의 복잡도 때문에, 이러한 하드웨어의 기능을 검증하는 일이 전체 설계에 소요되는 비용과 시간면에서 많은 부담이 되고 있다. 따라서, 마이크로프로세서의 설계에 있어서 기능 검증은 매우 중요한 과정으로 대두되고 있다.

기능 검증의 방법에는 여러 가지가 있는바, 그 중에 시뮬레이션 기반의 검증 방법은 가장 일반적이면서 정확한 검증 방법으로 평가되고 있다. 그러나, 긴 수행시간을 요구하고, 점점 복잡해지는 하드웨어의 모든 상태에 대한 측정이 어렵다는 단점이 있다.

본 실험에서 검증 모델로 사용한 32비트 마이크로프로세서인 AE32000[1]은 (주)ADChips의 EISC Family CPU Core로 시뮬레이션 기반의 일반적인 검증 절차를 사용하여 검증을 수행하고, 효과적인 디버깅 과정을 위하여 역엔지니어링(Reverse Engineering) 기법[2]을 도입하여 좀 더 빠르고 완벽한 디버깅 과정을 수행하고자 한다.

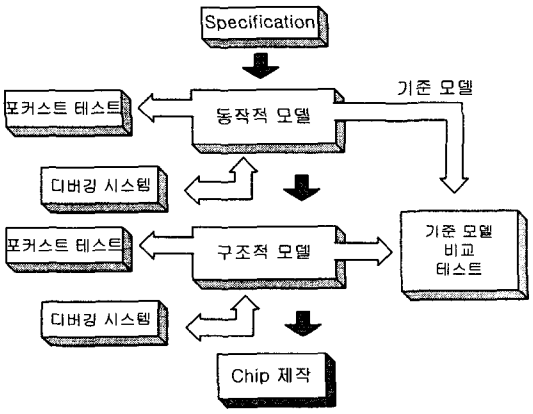
본 논문의 구성은 다음과 같다. 제Ⅱ절에서는 AE32000의 설계 검증을 위해 본 논문에서 사용하는 기능 검증 기법의 개요를 설명하고, 제Ⅲ절에서는 실제로 AE32000 모델을 적용한 실험 및 결과에 대해 설명을 하고, 마지막으로 제Ⅳ절에서 결론을 맺는다.

Ⅱ. AE32000의 기능 검증

2.1 기능 검증 방법

기능 검증의 방법에는 여러 가지 다양한 방법들이 있으며, 그러한 방법들에는 시뮬레이션(simulation) 기반의 검증 방법과 에뮬레이션(emulation) 기반 검증 방법 그리고, 정형 분석을 기반으로 하는 검증 방법 등으로 크게 나눌 수 있다. 시뮬레이션을 기반으로 하는 검증 방법은 가장 일반적이면서 정확한 검증이 가능하나, 긴 수행 시간을 요구하고, 점점 복잡해지는 시스템의 모든 상태에 대한 측정이 어렵다는 단점이 있다. 이 시뮬레이션을 이용한 검증 방법에는 포커스트(Focused) 테스트와 기준 모델 비교 테스트 등이 있다[3]. 에뮬레이션 기반의 검증 방법은 설계된 모델의 기능을 소프트웨어나 하드웨어로 직접 구현하여 검증하

는 방법으로서 비교적 많은 검증 시간을 들이지 않고도 정확한 검증을 할 수 있는 장점이 있으나, 에뮬레이션을 위한 별도의 에뮬레이션 시스템이 필요하므로 비용이 많이 드는 단점이 있다[4]. 정형 분석 기반의 검증은 기준 모델과 합성된 모델의 동일성 정도를 체크하며, 비교적 적은 검증 시간을 가지고, 정확한 검증을 할 수 있다는 장점이 있으나, 적용이 어렵고 많은 시간을 필요로 하며 실제로 적용 가능한 회로의 사이즈가 제한적인 단점이 있다[5]. 이 밖에 최근에 제안된 방법들로서 파이프라인의 가장 깊은 단계부터 통합하면서 통합이전 모델과 비교하는 것으로 검증을 수행하는 방법[6]과, Full RTL 구현 모델의 검증을 위해 역엔지니어링 기법을 이용하는 방법[2]들이 있다.

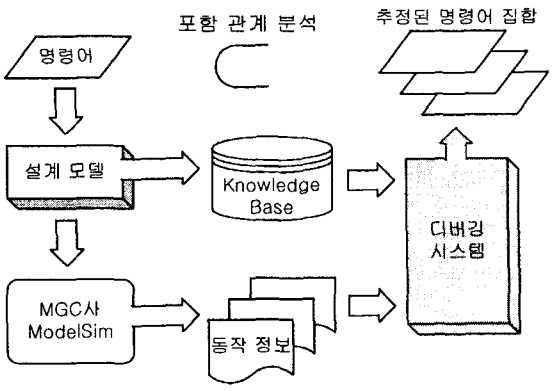


[그림 1] 기능 검증 메커니즘의 흐름도

이러한 새로운 방법들 중 본 연구에서는 역엔지니어링 기법을 이용한 검증 방법에 관심을 가지고 보다 효율적이고 검증의 성능을 향상시킬 수 있는 방안을 모색해 보고자하였다. [그림 1]은 본 연구에서 사용한 검증 메커니즘(mechanism)에 대한 전체 흐름도를 보여준다.

2.2 디버깅 시스템

마이크로프로세서의 설계에 있어서 실제로 설계된 구조에 대한 지식이 없이는 내부 구조만으로 명령어 집합의 형태를 정확히 파악하기 어려우며, 특히 내부 연결 경로가 매우 복잡한 시스템의 경우에 모든 제어 신호 및 데이터 경로에 대한 추정이 어렵다. 따라서, 본 연구에서는 역엔지니어링 기법을 도입하여 설계된 모델로부터 내부의 데이터 및 제어 신호의 연결 경로를 파악하고 파이프라인 마이크로프로세서의 각 파이프라인 단계별 동작에 따른 연결 정보를 분석하여 지식 기반(Knowledge Base)을 생성한다.



[그림 2] 디버깅 시스템의 전체 구성도

[그림 2]의 C 언어로 작성된 디버깅 시스템은 위의 과정에서 생성된 지식 기반과 임의의 명령어를 수행시켜서 얻어진 모델의 동작 정보를 통해 설계된 모델의 내부를 동작시킨 명령어 집합을 추정하여 추정된 명령어 집합과 실제 사양(Specification)상의 명령어 집합의 포함 여부를 확인하는 방식을 사용한다. 따라서 기존의 명령어 수행의 정오를 가리기 위해 해당 명령어가 수행될 때 모델의 내부에서 동작해야 하는 제어신호나 레지스터의 값의 변화를 보는 것을 좀 더 효과적으로 사용한다. 그러므로 임의의 명령어에 대해 내부적으로 동작하는 모델의 내부 모듈을 파악하고 어떠한 명령어가 수행되었는지를 추정한 후 추정된 명령어 군에 대해 실제 수행된 명령어와 포함 관계가 성립되는지를 확인하여 설계된 모델에 대한 디버깅을 수행하게 된다. 이러한 방식은 내부적으로 동작된 모듈에 대한 검증이 가능하고, 궁극적으로 동작되지 않은 모듈에 대해 동작을 시킬 수 있는 명령어 군이 파악됨으로 Corner Case에 대한 검증이 수행될 수 있으며, 보다 작은 테스트 명령어 집합으로 Code Coverage를 높일 수 있는 방법이 된다. 이를 위해 AE32000을 모델로 사용하여, 여기에 사용되는 84개의 명령어 집합을 5개의 클래스로(Load, Store, Branch, Data processing, etc.) 분류하여 명령어 군을 형성하였다

단계 / 클래스	IF	ID	EX	MEM	WB	클래스 판별
Load	O	O	O	O	O	LOAD
Store	O	O	O	O		
Branch	O					
Data P.	O	O		O	O	
etc.	O	O	O			

[표 1] 파이프 단계 동작에 따른 클래스 판별

[표 1]은 5개의 클래스로 구분된 명령어 집합에 대해 해당 파이프라인 단계에서 동작이 수행되었는지를 점검하여 동작이 이루어진 부분에 동작 여부를 표시하였다. 그러므로, 만약 임의로 수행된 명령어가 Load 명령어 집합에 포함되는 명령어라면, 설계된 모델은 각 단계별로 해당 명령어가 수행될 때 동작하는 모듈을 위주로 동작이 이루어지게 된다. 그러므로, 해당 클래스가 모두 동작하거나 가장 많은 동작을 하게 되는 해당 클래스가 추정되고, 그것을 토대로 좀 더 세부적인 모듈의 분석을 통해 명령어가 판별된다.

단계 / 명령어	IF	ID	EX	MEM	WB	명령어 판별
LDB	O	O	O		O	LD
LDS	O	O	O		O	
LD	O	O	O	O	O	
LDBU	O	O	O		O	
LDSU	O	O	O		O	
POP	O	O				

[표 2] 클래스 판별 후 명령어 판별

[표 2]에서 세부적으로 판별된 명령어는 실제로 해당 명령어가 수행되면서 파생될 수 있는 여러 가지 조건에 따라 세부적으로 나눠지므로 같은 LD 명령어 중에서도 여러 형태의 파생 명령어 집합을 형성하게 된다. 이것을 실제 임의로 입력한 명령어와 비교해서 추정한 명령어 집합 내에 입력한 명령어가 포함되어 있으면, 기능상의 문제가 없는 것이고, 만약 추정한 명령어 집합 내에 입력한 명령어가 포함되지 않으면, 기능상의 문제가 있는 것으로 판단하게 된다. 문제가 있는 것으로 판단이 되면, 내부적으로 해당 명령어가 동작을 시킨 부분을 다시 점검하여 내부의 오류를 쉽게 찾을 수 있게된다.

Ⅲ. 실험 및 결과

3.1 실험 과정

본 연구에서는 Mentor Graphics사의 상용 시뮬레이션 도구인 Modelsim 시뮬레이터를 이용하여, 보다 효율적이면서 신뢰성 있는 검증을 수행하였다.

먼저 설계된 모델에 대한 포커스트 테스트에서는 100개의 특정 기능을 검증하도록 제작된 테스트 프로그램을 사용하여 개별 명령어와 조합 명령어에 대해 시뮬레이션 기반의 테스트를 하였다. 이 과정에서 기능상의 문제를 발생시키는 명령어에 대해 디버깅 시스템을 사용하여 디버깅을 수행하였다. ModelSim의 Code Coverage는 시뮬레이션이 수행되는 동안 설계된 모델 HDL code의 해당 라인이 실행되는 경우, 해당 라인이 동작한 횟수를 알려주는 기능을 수행한다. 이러한 기능을 사용하면 임의의 명령어 프로그램이 수행될 경우 해당 모듈의 동작 여부를 파악할 수 있어서 시스템의 내부의 동작을 파악할 수 있다. 그러므로, 이 방법을 시간으로 나누어 적용하면 해당 클럭에 따라 동작되는 구조를 파악할 수 있어서 내부의 동작을 파악하는 효과를 극대화 할 수 있다. 또한 실험에서 적용 대상으로 삼고자 하는 파이프라인 마이크로프로세서의 경우는 각 클럭 단위로 단계별 동작을 하는 구조이므로, 여기서 사용한 방법이 큰 효과를 거둘 수 있다.

단계	동작 기술	동작 특징	판별

IF	ppc = nppc;
	iaddr = ppc;
	nppc = ppc + 4;
		일반적인 명령어 패치 동작	L/S/B/D/E

ID	r1_idx=ir[3:0];
	abus=rf[r1_idx];
	imm={4'b0,ir[7:4],2'b0};
	widx=ir[11:8];
		Load 및 Store를 위한 명령어 디코딩	L/S/D

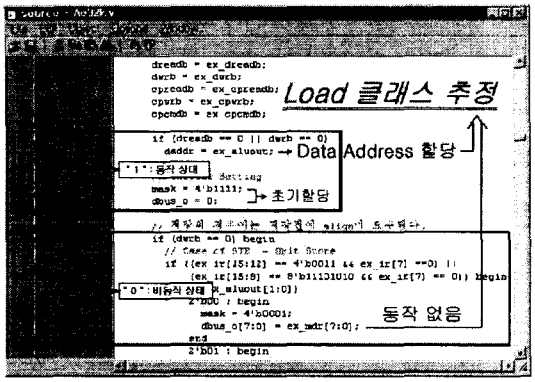
EX	ex_aluout = alu_in1 +{26'b0,imm[5:0]};	일반적인 주소 연산	L/S
MEM	daddr=ex_aluout; bus_k=4'b1111; dbus_o=0;	Load 추정	L
WB	cbus <= dbus_i; rf[mem_widx]<=cbus	Data P. 탈락 Store 탈락.	L

[표 3] AE32000동작적 모델에 LD명령어를 적용

디버깅 시스템을 사용하는 실험에서는 포커스트 테스트에서 사용한 테스트 프로그램을 그대로 사용하였으며, AE32000에 적용하여 모델 내부에서 동작하는 부분에 대한 정보와 데이터 및 제어 신호에 관한 정보를 파악하였다. 이를 바탕으로 명령어 클래스를 생성하고 생성된 클래스가 실제로 입력된 명령어 클래스에 포함 여부를 확인한 후 비교를 통해서 설계 오류를 찾아내었다.

[표 3]은 Load 명령어에 대한 추정 과정이다. 각 단계별 동작을 살피고 각 단계에서의 동작 특성으로 클래스를 판별한 결과, 모든 단계에 추정이 가능한 클래스가 존재하는 Load 명령어로 결정됨을 나타낸다.

[그림 3]은 [표 3]의 MEM 단계에서 동작 기술을 추출해 내고 Load 명령을 추정하는 것을 보여주는 실험 예제이다. [그림 3]에서 Data Address 할당이라고 표시된 daddr=ex_aluout; 라인은 메모리에 데이터 어드레스를 보내는 것으로 Load나 Store시 메모리에 주소값을 주기 위한 동작이다. 또한, 비동작 상태의 해당



[그림 3] MEM 단계에서 클래스 판별

라인이 동작하지 않았다는 것은 Store 클래스의 명령이 아님을 의미하므로 Load 클래스가 추정된다. 이렇게 추정된 클래스는 나머지 파이프라인의 단계에 대한 분석 과정을 통하여 명령어 클래스내의 유사 명령어가 추정된다.

3.2 오류 검출 결과

명령어 군	명령어 수	오류수	오류분포
데이터 처리 관련	38 개	10 개	50 %
분기 관련	18 개	3 개	15 %
읽기 관련	15 개	3 개	15 %
저장 관련	6 개	1 개	5 %
기타 관련	7 개	3 개	15 %
계	84 개	20 개	100 %

[표 4] 동작적 모델에 대한 오류 검출 결과

[표 4]는 AE32000에 대한 오류 검출 결과이다. 총 84개의 명령어에 대해 20개의 설계 오류의 검출 및 디버깅을 수행하였다. 오류의 분포는 가장 많은 명령어 수를 포함하는 데이터 처리 관련 명령어 군이 가장 높은 50%를 차지하였고, 저장 관련 명령이 가장 낮은 5%의 오류 분포를 나타내었다.

Ⅳ. 결론

본 연구에서는 32비트 마이크로프로세서인 AE32000의 기능 검증을 수행하였다. 효율적인 기능 검증을 위해 역엔지니어링 기법을 도입한 빠르고 완벽한 디버깅 시스템을 개발하였다. 기능 검증 방법은 시뮬레이션 기반의 일반적인 검증 절차를 사용하였으며, 구현된 모델을 가지고 임의의 명령어에 따른 동작의 상태를 분석하여 실제 입력된 명령어를 추정한 후 실험을 통해 추정된 명령어와 실제 입력한 명령어를 비교하는 것으로 디버깅을 수행하였다. 이와 같은 디버깅 과정에서는 테스트 프로그램을 사용하여 해당 명령의 동작을 살피는 기존의 방법에서는 쉽게 발견되기 어려운 Corner Case를 발견할 수 있었다. 또한 모든 가능한 경로에 대한 값을 확인하는 것이 아니라, 내부에서 동작하는 경로를 파악함으로써 오류를 단 시간에 쉽게 발견 할 수 있는 장점이 있었다.

Verilog-HDL로 설계된 AE32000의 동작적 모델에 대해 테스트를 하였으며, 현재 합성이 가능한 구조적 모델에 대한 검증을 본 논문에서 제안한 방법으로 진행 중이다. 또한 이렇게 검증이 완료된 동작적 모델을 기준 모델로 삼고 합성 가능한 구조적 모델과 동일한 테스트 벡터를 입력하여 내부의 레지스터와 제어 신호들의 변화를 비교하여 검증을 수행하는 기준 모델 비교 테스트를 수행할 것이다.

감사의 글

본 논문은 COSAR의 연구비 지원과 IDEC의 설계 도구 지원에 의하여 연구되었습니다.

참고 문헌

[1] H.-C. Oh et al., "AE32000: An Embedded Microprocessor Core", in Proc. AP-ASIC2000, pp.255--258, Aug. 2000.

[2] J. D. Hauke, "Design Verification Using Reverse Engineering." Master Thesis, Computer Science and Engineering, The University of Michigan, May, 1999.

[3] M. Kantrowitz et al., " I'm done simulation; now what? Verification coverage analysis and correctness checking of the DECchip 21164 Alpha microprocessor. ", in Proc. DAC, pp.325-330, 1996.

[4] G. Ganapathy et al., "Hardware emulation for functional verification of microprocessors", in Proc. DAC, pp.305-310, 1996.

[5] Toshihiro Hattori et al., "Design Methodology of 1200Mhz Superscalar Microprocessor:SH-4", in Proc. DAC, pp.246-249, 1998.

[6] J. Levitt and K. Olukotun, "Verifying correct pipeline implementation for microprocessors," in Proc. ICCAD, pp.162-169, 1997.