해외
기술

시스템레벨 大規模論理 시뮬레이션 方法

1. 머리말

LSI의 고집적화가 진전되어 수백만게이트를 집적하는 LSI가 등장하고 있다. 이와 같은 고집적 LSI를 多用한 계산기시스템에서는 논리회로의 거의 전부가 LSI내에 들어가기 때문에 불량상태의 요인도 LSI내부에 기인하게 된다. 따라서 불량상태 요인을 규명하는 것이 대단히 어려우며, 원인이 명확한 경우에도 불량상태의 수정작업은 LSI의 재제작으로 이어진다. 이 때문에 설계에 필요한 코스트가 방대해진다. 설계코스트를 삭감하고 기간을 단축하기 위해서도 LSI를 제조하기 전의 설계단계에서 논리검증을 할 때 불량상태의 원인을 완전히 제거하고 시뮬레이터로 동작확인을 마칠 필요가 있다.

종래부터 행하고 있는 LSI單体 검증에서는 버스의 競合동작·에러처리 등 복수 LSI의 동시동작, 또는 시스템으로서의 동작 검증을 할 수 없기 때문에 상기와 같은 요구에 충분히 응할 수가 없다. 그 때문에 시뮬레이션환경을 실제의 시스템에 가깝도록 하여 보다 큰 회로규모에서 많은 테스트데이터를 사용하여 시뮬레이션할 필요가 있다.

동사의 계산기개발에서는 1982년에 패스해석에 의한 타이밍검증과 시뮬레이션에 의한 논리검증을 분리한 이래, 市販시뮬레이터를 사용하여 대규모 논리시뮬레이션의 회로규모의 확대를 도모하여 왔다. 1984년에는 CPU레벨을, 1990년부터는 다단계기억층과 주변장치를 포함한 시스템레벨을 검증하고 있다. 이것은 시뮬레이터 자신의 진보와 호스트머신의 고속화에 크게 의존한다. 그러나 시뮬레이터가 취급할 수 있는 회로규모와 속도는 아직 충분치가 않다.

따라서 시뮬레이터의 성능을 최대한 살린 모델화기술과 계산기시스템의 동작상태를 고려함으로써 효율좋은 시험데이터를 작성하는 아키텍처에 정통한 시험기술이 중요하다.

이 논문에서는 시스템레벨 大規模論理 시뮬레이션에 관하여 시스템레벨 검증수법, 시스템레벨 시뮬레이터의 개요와

해외기술

구성, 시뮬레이션 방법에 대하여 기술한다. 다음으로 網羅性을 높이기 위하여 채용한 랜덤테스트에 대하여 기술하고 마지막으로 최근의 적용사례를 소개한다.

2. 시스템레벨 검증방법과 개요

2.1 시스템레벨 검증

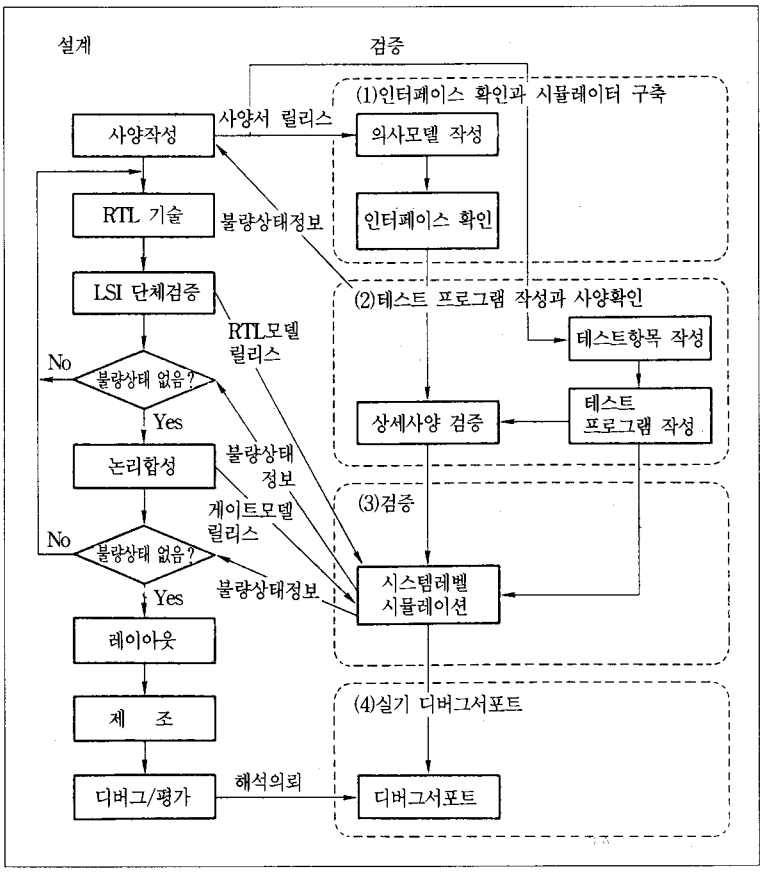
LSI單体檢証, 계산기시스템의 시스템레벨검증과 실시스템에 의한 디버그(이하 "실기디버그"라 함)의 비교를 표1에 나타낸다.

實시스템에서는 LSI나 CPU보드는 單体로는 동작하지 않으며 DISK장치 등의 I/O를 접속하여 동작한다. 실기디버그에서는 진단용 프로그램(이하 "진단프로그램"이라 함)을 실행하고, 결과의 확인은 주기억 메모리, 아키텍처레벨의 레지스터 내용, 주변회로에 전송된 데이터 등으로 확인한다.

시스템레벨검증의 목적은 검증대상인 LSI와 CPU보드가 實機와 동일한 동작환경에서 바르게 동작하는 것을 사전에 보증함에 있다. 그 때문에 테스트데이터는 LSI외부핀에의 입력패턴이 아니라 실기디버그에서 사용되는 진단프로그램이나 LSI의 동작을 검증할 수 있는 프로그램을 사용한다.

〈표 1〉 검증의 비교

구 분	LSI 單体檢證	시스템레벨 檢證	실기디버그
대 상	L S I	계산기 시스템	계산기 시스템
목 적	사양대로 되어 있는가를 확인	사양의 정당성을 확인 정상동작을 확인 시스템성능측정 (일부예측)	정상동작을 확인 시스템성능 측정
테스트 방 식	LSI내부레지스터 · 입력신호에 값을 설정	주변회로로부터의 버스 오퍼레이션 및 진단프로그램을 사용	진단프로그램을 사용
기 대 치	LSI내부레지스터 LSI출력신호	주기억 캐시메모리 아키텍처레벨 레지스터 LSI내부레지스터	주기억 캐시메모리 아키텍처레벨 레지스터
관측신호	LSI외부의 신호 LSI내부의 신호	LSI외부의 신호 LSI내부의 신호	LSI외부의 신호



〈그림 1〉 계산기시스템 개발의 흐름

결과 확인은 1클록마다의 동작이 아니라 테스트 종료후의 주기억메모리, 캐시메모리, 아키텍처레벨의 레지스터値 등을 기대치와 비교함으로써 행한다. 버스프로토콜의 검증과 불량상태의 해석수단으로 LSI내부의 신호도 확인할 수 있다.

이와 같이 시스템레벨 검증이란, 종래 실기디버그로 하던 실시스템에 의한 동작 확인작업을 시뮬레이터로 하는 것이다.

2.2 설계작업과 시스템레벨 검증

계산기시스템의 개발의 흐름을 그림 1에 표시한다. 시스

템레벨 검증은 계산기 시스템의 개발에 맞추어 다음의 스텝으로 한다.

(1) 인터페이스확인과 시뮬레이터구축

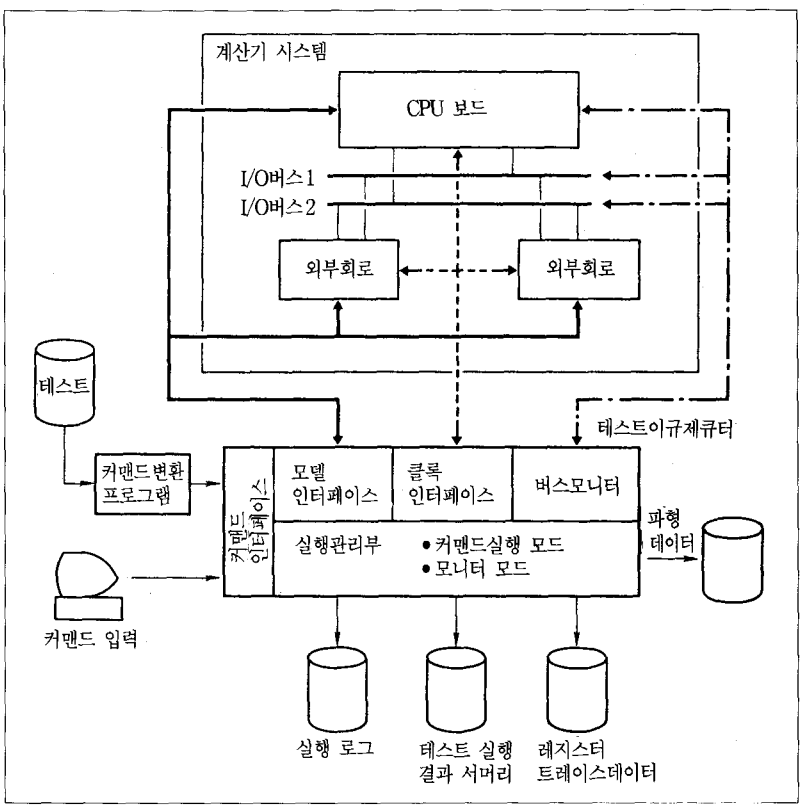
설계측이 작성한 각 LSI의 사양서를 기초로 검증측은 인터페이스신호와 기본적인 기능이 동작하는 LSI의 의사모델 및 CPU 보드모델을 작성한다. 또 시스템사양에 기초하여 검증에 필요한 주변회로를 작성하고 의사모델과 CPU보드모델을 조합하여 시스템레벨 시뮬레이터를 구축한다. 여기서는 각 LSI간 인터페이스 및 CPU보드 내외의 인터페이스의 확인과 시스템레벨 시뮬레이터의 구축을 목적으로 하고 있다.

(2) 테스트프로그램 작성과 사양확인

각 LSI의 사양서를 기초로 검증측은 인터페이스확인용 의사모델의 기능을 더욱 충실하게 하여 거의 모든 LSI의 기능이 동작하는 사양확인용의 의사모델을 작성한다. 시스템레벨검증용의 테스트사양 작성도 병행하여 행한다. 테스트사양으로부터 테스트프로그램을 작성하고 사양확인용의 의사모델을 사용한 시스템레벨 시뮬레이터상에서 디버그한다. 테스트사양 작성·테스트프로그램 작성에서 분명히 LSI사양에 문제가 있으면 설계측에 보고한다. 이 모델의 목적은 테스트프로그램작성과 LSI의 상세한 사양을 검증하는데 있다.

(3) 검 증

설계측에서 單体검증이 종료된 LSI의 RTL(Register Transfer Level)모델을 사용하여 시스템레벨 시뮬레이터를 구축한다. 복수LSI의 개발에서는 각 LSI의 개발진척이 다르다. 이 경우에는 일부의 LSI에 의사모델을 사용하여



〈그림 2〉 시스템레벨 시뮬레이터

RTL과 의사모델의 믹스드레벨 시뮬레이터를 구축한다.

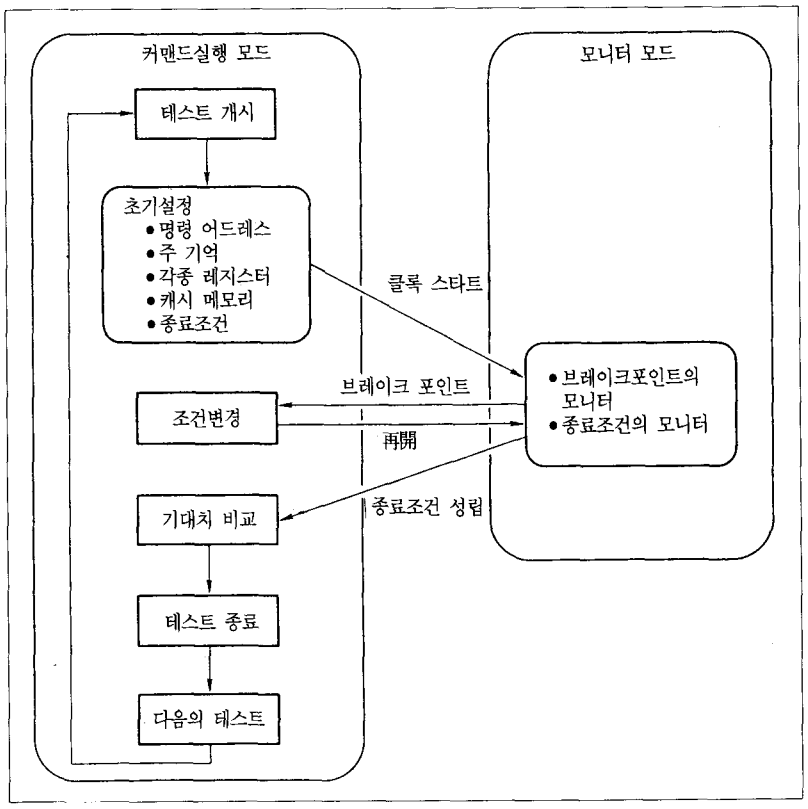
설계측은 單体검증을 완료한 후 論理合成하여 LSI의 게이트모델을 작성한다. 검증측은 게이트모델의 릴리스를 받아 시스템레벨 시뮬레이터를 구축하고 검증을 실시한다. 검증은 單一오퍼레이션의 확인에 관계되는 테스트로부터 시작하여 연속동작, 진단프로그램, 경합테스트, 에러테스트의 순으로 한다.

검증측에서 LSI의 타이밍개선이 종료된 모델을 검증하고 버그가 존재하지 않으면 LSI의 레이아웃 설계에 들어간다.

(4) 實機디버그서포트

實機디버그에서 불량상태가 생긴 경우에는 實機상에서 채취한 신호파형을 기초로 시스템레벨 시뮬레이터상에서 불량

해외기술



〈그림 3〉 테스트프로그램 실행 플로

상태의 상황을 재현하여 LSI의 내부동작을 해석한다. 實機상에서는 관측할 수 없는 상세정보를 얻을 수 있기 때문에 대단히 유효한 해석수단이다.

2.3 모델化 方法

시뮬레이터의 성능을 최대한으로 살리기 위한 모델화기술의 하나로서 논리검증에서는 지연정보를 제거한다. 타이밍검증은 靜的인 버스해석으로 대용한다. 그 결과 상세한 타이밍 정보를 고려한 시뮬레이션에 비하면 시뮬레이션에 필요한 메모리를 1/2로 삭감할 수 있고 또 약 5배의 시뮬레이션속도를 확보할 수 있었다.

또 게이트모델을 검증의 대상으로 한 이유는 다음과 같다.

(1) 논리합성툴이 작성한 게이트모델과 RTL모델과의 등가성을 검증할 필요가 있다.

(2) RTL모델을 사용하는 것보다는 게이트모델을 사용한 쪽이 약 4~5배 시뮬레이션속도가 빠르다.

검증에서 불량상태가 발견된 경우의 해석은 RTL모델을 사용하여 실시한다. RTL모델은 논리합성툴이나 라이브러리에 의존하지 않으며 게이트모델에 비하여 해석이 용이하기 때문이다.[1)]

3. 시뮬레이터의 모델과 그 기능

3.1 시스템레벨 시뮬레이터의 구성

이번에 사용한 시스템레벨 시뮬레이터는 그림 2에 나타낸 것과 같이 계산기시스템의 모델과 검증을 실행관리하는 테스트이그제큐터의 두 부분으로 구성된다. 시스템레벨 시뮬레이터는 하드웨어기술언어인 Verilog HDL(주1)을 사용하여 기술하고 있다.

계산기시스템의 모델은 복수의 대규모 LSI가 탑재된 검증대상인 CPU보드와 비검증대상의 주변외부회로로 구성된다.

비검증대상의 주변외부회로는 실물과 꼭 같은 기능을 가질 필요는 없다. 실물과 동등한 주변회로를 준비하면 그것을 제어하기 위한 프로그램이 복잡하게 된다. 그렇기 때문에 각종 I/O버스의 기본적인 오퍼레이션과 기본적인 에러처리를

(주1) "Verilog-HDL"은 미국 Cadence Design Systems, Inc.의 등록상표이다.

할 수 있는 의사모델만을 갖추었다. 주변외부회로를 제어하는 프로그램은 I/O버스의 오퍼레이션에 대응시켰다.

검증대상인 LSI, CPU보드 자체는 LSI의 개발스텝에 맞추어 검증측이 준비한 의사모델, 설계자가 작성한 논리합성전의 RTL모델, 합성후의 게이트모델을 사용한다.

각 모델에는 테스트이그제큐터와의 인터페이스를 부가하여 검증시와 불량상태해석시의 각 모델과 테스트이그제큐터와의 정보의 受渡를 쉽게 하고 있다. 각 레벨에서의 LSI모델과 테스트이그제큐터의 인터페이스를 동일하게 함으로써 각 LSI의 공정에 영향을 받지 않으며 RTL/게이트모델이 릴리스되기 전에 테스트프로그램을 디버그할 수 있다.

3.2 테스트이그제큐터

테스트이그제큐터는 시스템레벨 시뮬레이션의 모니터기능과 시뮬레이션환경의 구성을 제어하는 것으로 실기디버그시의 시스템콘솔, 제어패널, 로직레코더, 진단프로그램의 모니터부에 상당한다.

구체적으로는 테스트프로그램을 해석하는 커맨드인터페이스, 계산기시스템의 모델과의 인터페이스인 모델인터페이스, CPU보드클록이나 주변회로에의 클록을 공급하는 클록인터페이스, CPU보드의 내부버스, 외부버스를 모니터링하는 버스모니터, 테스트실행을 관리하는 테스트실행관리부가 있다.

테스트이그제큐터는 다음의 기능을 갖고 있다.

(1) 계산기시스템 구성정보의 설정

테스트이그제큐터내의 구성제어스위치를 설정함으로써 계산기시스템의 구성, CPU보드내의 구성(프로세서의 개수, 캐시메모리용량, 주기억용량 등), 시뮬레이션시의 각종 제어정보를 변경할 수가 있다.

(2) 디버그 기능

시뮬레이션시의 디버그 기능을 강화하기 위하여 주로 다음의 것을 표시할 수가 있다.

- 아키텍처레벨의 레지스터
- LSI의 내부레지스터
- 각종버스(I/O버스, 프로세서버스, 시스템버스)
- 캐시메모리, 주기억메모리
- 시뮬레이션 중의 구성정보

테스트이그제큐터의 커맨드를 會話的으로 입력하는 기능에 의하여 상기 내용을 간단하게 변경할 수가 있다. 또 명령어드레스와 마이크로프로그램 어드레스에 대한 브레이크포인트나 1클록씩의 스텝실행기능을 갖게 하고 있다. 그 때문에 지정한 조건이 발생하였을 때 테스트프로그램의 실행을 일시적으로 멈추게 하여 CPU의 상태를 변경하여 테스트프로그램의 실행을 재개할 수가 있다.

(3) 파형해석, 트레이스 기능

각종 버스, 아키텍처레벨의 레지스터를 트레이스하는 기능과 캐시메모리의 내용을 덤프하는 기능을 가지며 해석이나 버그레포트에 필요한 정보를 출력한다. 버스의 트레이스결과는 따로 작성한 프로그램을 사용하면 파형출력으로 변환될 수 있는 포맷으로 하고 있다. 에러신호에 대해서도 트레이스기능을 설치하여 에러보고패스를 검증할 수 있다.

(4) 실행결과 관리기능

테스트프로그램의 실행로그에 더하여 테스트프로그램 종료시에 각 테스트프로그램마다의 기대치와 실측치와의 비교결과를 출력하기 때문에 배치로 복수의 테스트프로그램을 실행하였을 때는 후에 테스트프로그램의 패스/페일을 종합하여 확인할 수가 있다.

해외기술

3.3 테스트프로그램 실행플로

각 테스트프로그램은 그림 3에 표시하는 플로로 실행한다. 테스트이그제큐터는 테스트 개시시에는 커맨드실행모드이며 필요에 따라 레지스터, 캐시메모리, 명령어드레스 레지스터 등에 초기치를 세트한다. 시스템클록을 움직여 테스트이그제큐터는 모니터모드가 된다. 모니터모드에서는 각 테스트프로그램에 기술되어 있는 테스트프로그램의 종료조건과 브레이크포인트를 모니터한다. 다음과 같은 종료조건이 성립되었을 때는 테스트프로그램을 종료한다.

- 명령어드레스가 지정한 어드레스와 일치
- 주변회로로부터의 종료조건
- 프로세서의 상태가 지정한 상태와 일치

테스트프로그램을 실행할 때는 배치기능을 갖게 하여 복수테스트프로그램의 연속실행을 가능케 하였다. 또 다음의 조건이 발생하였을 때는 테스트프로그램을 강제로 종료한다.

- 각종 버스의 행업
- 테스트의 실행 클록수가 지정치를 초과
- 테스트의 실행명령수가 지정치를 초과
- 한 명령의 실행클록수가 지정치를 초과
- 예기치 않은 에러 발생

테스트프로그램의 실행이 종료되면 테스트이그제큐터는 커맨드실행모드로 되돌아 오고 테스트프로그램에 기술된 기대치와 시뮬레이션 모델에서의 실측치의 비교로 테스트의 패스/페일을 판정한다. 페일시에는 버그해석에 필요한 정보를 파일에 덤프한다.

이상이 테스트이그제큐터의 특징이다. 테스트이그제큐터를 사용한 시뮬레이션은 아키텍처레벨의 검증에는 적합하나 버스오퍼레이션의 검증에는 적합치 않다. 왜냐하면 버스오퍼레이션의 검증과 같이 많은 조합조건이 존재하는 경우에는 개개의 조건에 대하여 1件1葉으로 테스트를 기술하게 되면 방대한 시간이 걸리며 조건누설이나 한쪽 편중이 생기기 때문이다.

그래서 우리들은 랜덤한 조건으로 테스트프로그램을 자동생성하는 툴을 개발하여 테스트이그제큐터에 부가하였다.

3.4 테스트프로그램 自動生成툴

테스트프로그램 자동생성툴은 버스오러페이션의 競合動作과 CPU의 상태에 착안한 조건을 조합한 테스트프로그램을 자동생성하는 시뮬레이션툴이다.

테스트프로그램 생성방식은 랜덤하게 시험조건을 설정하는 방식과 사람의 손에 의하여 시험조건을 지정하는 방식이 있다. 이들 두가지 방식을 단독으로 선택하거나 조합하여 선택할 수가 있다. 테스트프로그램개발자는 단순한 시험항목에 대하여는 시험조건을 지정하기만 하면 테스트프로그램의 작성이 가능하게 된다. 복수의 조건을 조합하는 시험항목에 대하여는 주된 시험조건만을 지정하고 기타의 조건을 랜덤하게 함으로써 테스트프로그램의 작성이 가능하다.

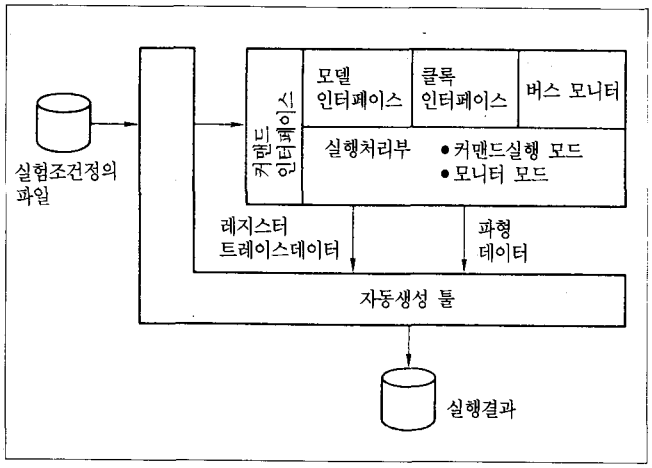
또 시험항목을 설정하지 않고 모든 시험조건을 램덤하게 한 테스트프로그램의 작성도 가능하다. 그러나 조건을 축소하여 불량상태의 검출효율을 올리는 것이 통상적인 사용방법이다.

테스트프로그램의 실행은 생성한 테스트케이스를 기초로 주변외부회로의 제어프로그램과 CPU의 초기상태를 설정한 후에 시험대상을 동작시킨다. 오퍼레이션 실행후에는 기대치와 실행결과를 비교함으로써 시험대상의 정당성을 확인한다.

테스트프로그램 자동생성툴의 구조와 동작을 다음에 기술한다.

(1) 구 조

그림 4에 표시하는 것과 같이 테스트이그제큐터에 자동생성툴을 부가하여 시뮬레이션환경을 구축한다. 시험조건이 지정된 외부파일을 기초로 테스트프로그램을 생성하는 기능, 생성한 테스트프로그램에서 시뮬레이터환경에 맞추어 테스



〈그림 4〉 프로그램 자동생성툴을 부가한 테스트이그제큐터

트이그제큐터의 커맨드를 생성하는 기능, 시험실행후의 기대치를 비교하는 기능으로 구성되어 있다.

(2) 동 작

생성하는 테스트프로그램은 아래에 표시한 버스의 동작조건과 CPU의 상태를 랜덤하게 조합한 것이다.

(a)버스동작의 조건

- 리퀘스트의 타이밍
- 버스의 오러페이션
- 어드레스와 데이터의 패턴
- 割込信號

(b) CPU의 상태

- 캐시메모리의 상태
- 스토어버퍼의 상태
- 메모리의 상태

또 자동생성툴의 테스트프로그램 실행후의 기대치생성에 더하여 각종 버스에 대한 타임의 체크기능을 가지고 있다.

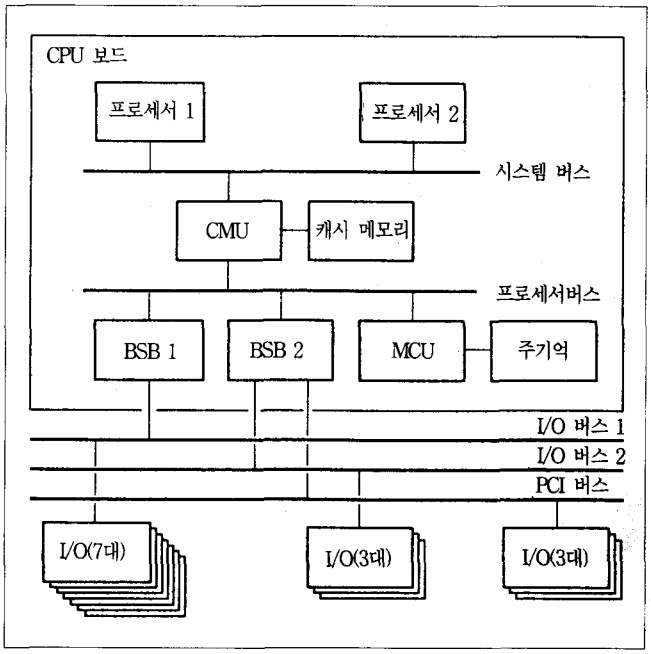
이 기능에 의하여 시험대상에서 의도한 시험이 행해지고 있음을 확인할 수가 있다.

4. 적용사례

시스템레벨 시뮬레이션과 자동생성툴을 적용한 시스템의 사례를 그림 5에 표시한다. CPU보드에서는 MCU(메모리 관리유닛), CMU(캐시메모리유닛)와 프로세서가 프로세서버스와 시스템버스로 결합되어 있다.

또 보드와 I/O사이는 동사의 독자적인 2종류의 I/O버스외에 PCI버스가 있으며 보드내의 버스브리지 BSB 1, 2로 시스템버스에 접속되어 있다. 시스템버스에 접속되어 있는 거의 모든 LSI가 능동적으로 시스템버스상에 리퀘스트를 출력할 수가 있다. PCI버스에는 3개의 에이전트, I/O버스 1에는 7개의 I/O, I/O버스 2에는 3개의 I/O가 접속되어 각각 능동적으로 리퀘스트를 출력할 수 있다.

목표하는 시험커버리지를 달성하기 위하여 필요한 테스트케이스의 수, 클록의 견적, 머신파워를 산정하였다. 이번에는 약 1주간으로 전 시스템테스트가 완료되도록 동사의



〈그림 5〉 대상의 계산기 시스템

해외기술

EWS "ME/R7500"(SPECmark=77.5) 5대와 "ME/R 7550"(SPECmark=146.8) 3대를 준비하였다.

그림 5의 계산기시스템을 검증할 때 시스템레벨 시뮬레이터에 의한 검증과 자동생성툴을 사용한 검증을 다음과 같은 분담으로 추진하였다.

(1) 시스템레벨 검증

시스템레벨 검증에서는 CPU보드에서 아키텍처레벨의 검증이 주체가 되었다.

- CPU보드에서의 사양서레벨의 검증
- 버스競合의 검증(최대 3개의 리퀘스트가 경합)
- 예외처리, 에러기능의 검증
- 진단프로그램에 의한 검증

시스템레벨검증에서는 종래 프로토機디버그에 미루어 왔던 프로그램디버그기능 예외처리, 에러처리를 완전하게 검증할 수가 있었기 때문에 검증시의 커버리지가 향상되었다.

(2) 자동생성툴을 사용한 검증

자동생성툴을 사용한 검증에서는 버스상에서의 오퍼레이션 경합검증이 주체가 되었다. 또한 하나의 테스트를 기초로 하여 리퀘스트의 타이밍을 램덤하게 변화시킴으로써 보다 높은 檢證度를 올리고 있다.

- 시스템버스상에서의 오퍼레이션의 경합(최대 6개의 리퀘스트를 경합시킴)
- I/O버스上에서의 리퀘스트의 경합

이번에 준비한 테스트케이스의 종류, 테스트케이스의 수, 클록수를 표 2에 표시한다.

의사모델 작성시기부터 테스트이그제큐터와 모델과의 인터페이스를 공통으로 함으로써 빠른 단계에서부터 테스트할 수가 있었다. 또 랜덤테스트를 시간이 허용하는 한 실행함으로써 사람손에 의한 작성에서 빠트리기 쉬운 부분을 커버할 수가 있었다.

〈표 2〉 實施테스트의 내용

테스트의 종류	건수	클 록 수
仕樣書 레벨	800	350만
진단프로그램	3,500	1500만
랜덤테스트	—	2000만
예외·에러처리	500	300만

새로이 개발한 진단프로그램은 시스템레벨 시뮬레이터를 사용하여 동작확인을 하였다.

5. 맺음말

이 논문에서 소개한 시스템레벨검증 및 테스트프로그램 자동생성툴에 의한 랜덤테스트에 의하여 LSI작성전에 實시스템에서 행하는 디버그의 초기단계를 시뮬레이션에 의하여 할 수가 있었다. 이에 의하여 이번에도 LSI를 다시 고쳐만드는 일이 없이 기능진단프로그램 및 시스템진단프로그램에 의한 테스트를 거쳐 오퍼레이팅시스템의 동작을 확인할 수가 있었다.

또 과거의 프로젝트에 있어서도 실시스템에서 발생한 장해를 시스템레벨 시뮬레이터에서 재현하고 LSI의 내부동작을 해석하여 실시스템에서 일어난 동작을 쉽게 또한 확실하게 해명할 수 있음이 입증되었다.