CCA 보드를 위한 I-Link 버스의 정형 검증[1]

남원홍* °성창훈* 최진영* 기안도** 한우종**
고려대학교 컴퓨터학과*
{wnam, chsung, choi}@formal.korea.ac.kr*
한국전자통신연구원 컴퓨터소프트웨어기술연구소 컴퓨터시스템연구부**
adki@computer.etri.re.kr**, wjhan@com.etri.re.kr**

Formal Verification of I-Link Bus for CCA Board

Won-Hong Nam* °Chang-Hun Sung* Jin-Young Choi*
An-Do Ki** Woo-Jong Han**
Dept. of Computer Science & Engineering, Korea University*
Computer System Department of ETRI-CSTL**

요 약

본 연구는 심볼릭 모델 체커 중의 하나인 SMV(Symbolic Model Verifier)를 이용하여 한국전자통신연구원(ETRI)에서 개발한 CCA (Cache Coherent Agent) 보드를 위한 I-Link Bus(Inside Bus)의 몇 가지 특성(property)들을 검증하여 I-Link Bus의 요구사항(requirement)이 만족됨을 보인다. 이 검증에서는 I-Link Bus의 모델을 SMV 입력 언어로 명세하며, 검증할 특성들을 시제 논리(temporal logic)를 이용하여 나타낸다. 검증을 통해서 I-Link Bus와 PIF(Processor Interface), DC(Directory Controller), RC(Remote access cache Controller) 모듈들이 중재기 우선 순위, send 우선 순위, 중재 요청 신호의 관리, liveness등의 특성들을 만족한다라는 것을 검증하였다.

1. 서론

하드웨어나 소프트웨어 시스템은 불가피하게 규모면, 기능면에서 점점 커지고 복잡해지고 있으며, 이러한 복잡성의 증가로 시스템에서의 에러 존재 가능성은 더욱 높아지고 있다. 덴버 공항 지하 화물 처리 시스템의 실패, 인텔 펜티엄 프로세서의 FDIV 명령어 오류, 아리안 로켓의 폭발 등에서 알 수 있듯이 이러한 에러들은 비용, 시간, 그리고 인력면에서 엄청난 대가를 치러야 했으며, 심지어 인간의 생명에도 위협을 주었다. 특히 펜티엄 프로세서의 FDIV 오류는 이런 종류의 오류가 언제든지 반복해서 발생할 수 있고, 그 손실 역시 매우 크다는 이유로 정형 기법(formal method)의 중요성을 강조 시켰다. 따라서 하드웨어 및 소프트웨어 설계 전문가들은 점차로 기존에 사용되던 시뮬레이션 방법을 대체할 방법으로 정형 검증에 대한 관심을 갖게 되었다[1].

정형 기법은 수학과 논리학에 기반을 둔 방법으로 하드웨어나 소프트웨어 시스템을 명세하거나 검증하는 것이며, 이러한 정형 기법에는 크게 정형 명세(formal specification)와 정형 검증(formal verification)이 있다[2]. 정형 명세는 정형 논리(formal logic) 또는 수리 논리(mathematical logic)에서 사용되는 기호 등을 이용하여 시스템이 동작할 환경에 대한 가정, 시스템이 만족해야 하는 요구사항, 그리고 요구사항을 수행할 시스템 설계 등을 기술하는 것이다. 또한, 정형 검증은 정형 논리 또는 수리 논리에서 제공하는 증명 방법 등을 이용하여 정형 명세를 분석하여 무모순성 및 완전성을 검증하거나, 설계가 주어진 가정에서 요구사항을 만족하는지를 검증하는 기법이다.

정형 검증에는 크게 정리 증명(theorem proving) 방법과 모델 체킹 방법이 있다[2]. 정리 증명 방법에서는 시스템 모델과 명세를 적당한 수학적 논리를 이용한 언어로 표현하고 시스템 모델이 명세를 만족함을 증명을 통해서 보인다. 이 방법은 매우 강력하고 유연한 방법이지만 수많은 논리적 증명을 해야 하므로 매우 어렵고 전문가가 아니면 이용하기가 어렵다. 반면 모델 체킹은 적용분야는 정리증명보다 작지만, 직접 증명할 필요가 없이 도구를 이용하여 손쉽게 그리고 빠르게 검증이 되며 완전히 자동화 될 수 있다[3].

본 논문은 심볼릭 모델 체킹 검증 도구의 일종인 SMV를 이용하여 ETRI에서 개발한 CCA 보드를 위한 I-Link Bus를 검증한 사례를 제시한다. 2장에서는 모델 체킹과 검증 도구인 SMV에 대해서, 3장에서는 I-Link Bus에 대해서 알아 본다. 4장에서는 SMV를 이용한 명세 및 검증에 대해서 설명하며, 5장에서 결론 및 향후 과제를 제시한다.

2. 모델 체킹과 SMV

2.1. 모델 체킹

모델 체킹은 상태 탐색(state exploration)을 기반으로 하는 정형 검증 기법이다. 상태 전이 시스템(state transition system)과 특성이 주어지면, 모델 체킹 알고리즘은 시스템이 주어진 특성을 만족하는지를 알아 보기 위해서 전체 상태 공간(state space)을 검사한다. 시제 논리(temporal logic) 모델 체킹은 검증하려고 하는 하드웨어나 소프트웨어 시스템을 모델링한 상태 전이 시스템과 특정한 시제 논리로 표현된 특성을 입력 받아서, 시스템이 그 특성을 만족하는지를 검증하는 것이다. 이러한 모델 체킹에서 일반적으로 사

[1] 본 연구는 1999년 한국전자통신연구원 위탁과제 99379의 결과임.

용되는 논리에는 명제 논리(propositional logic)에 시간 연산자(temporal operator)를 넣어서 확장 시킨 branching-time Computation Tree Logic(CTL)과 Linear Temporal Logic (LTL)이 있다.

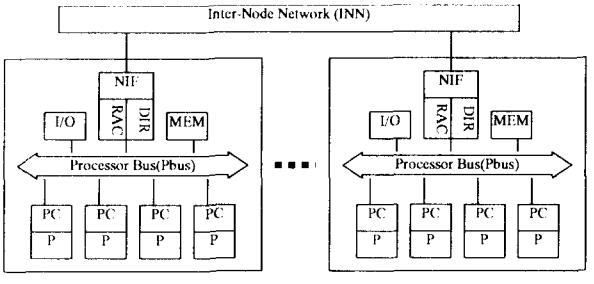
2.2. SMV(Symbolic Model Verifier)

SMV 시스템은 유한 상태 시스템(finite state system)이 temporal logic으로 표현된 요구 명세를 만족하는지 검증하는 정형 검증 도구이다. SMV의 입력 언어는 유한 상태 시스템을 명세하기 위해 만들어 졌다. 시스템은 이 입력 언어를 통해서 동기적인 밀리 머신(Mealy machine)이나, 비동기적인 네트워크로 손쉽게 명세 될 수 있다. SMV 언어가 유한 상태 시스템을 위해 만들어진 것이기 때문에 SMV 언어는 Boolean, scalar, fixed array등만을 제공한다. Temporal logic은 safety, liveness, fairness, dead lock freedom 을 포함한 풍부한 종류의 시간적 특성(temporal property)들을 간단한 문법을 이용하여 표현하는 것이 가능하다. SMV는 입력 언어로 나타내어진 모델이 temporal logic으로 표현된 요구 명세를 만족하는지의 여부를 효율적으로 검사하기 위해서, BDD(Binary Decision Diagram)[4]를 기반으로 하는 심볼릭 모델 체킹 알고리즘을 사용한다. 이러한 SMV를 이용하여 대형 하드웨어나 소프트웨어 시스템 명세를 검증하는 예가 많이 연구되고 있다[3].

3. CCA 보드를 위한 I-Link Bus

3.1. 개요

CC-NUMA 의 일종인 PDLSystem(Physically-Distributed but Logically-Shared memory system)은 [그림 1]과 같이 inter-node 네크워크에 연결되어 있는 여러 개의 노드들로 구성되어 있다. 각각의 노드들은 캐쉬를 가지고 있는 프로세서와 공유 메모리의 일부분인 메모리 모듈, I/O, RAC(Remote Access Cache)와 디렉토리로 구성된 네트워크 인터페이스를 갖는다[5].

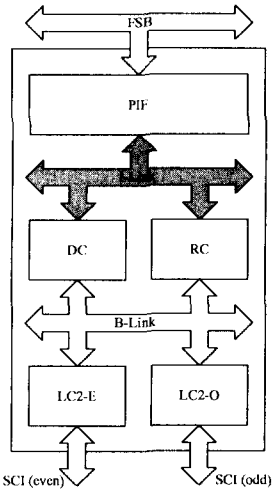


[그림 1] PDLSystem : P(Processor); PC(Processor Cache); NIF(Network Interface); DIR(Directory); RAC(Remote Access Cache)

노드 안의 프로세서 캐쉬들은 snoopy 방식의 일종인 MESI 프로토콜에 의해 일관성이 유지되며, 노드 간의 일관성은 directory 기반의 RACE 프로토콜(Remote Access Cache coherency Enforcement Protocol)[5]에 의해 유지된다. 프로세서는 프로세서 캐쉬에 없는 데이터 접근 시 해당 캐쉬 라인을 가져오기 위해서 프로세서 버스에 요청을 보낸다. 만약, 그 주소가 노드 안의 로컬 메모리에 해당한다면 로컬 메모리가 요청한 데이터를 제공한다. 그렇지 않은 경우, RAC가 그 요청을 처리하게 된다. 즉, RAC가 해당 데이터를 유효한 상태로 가지고 있다면 바로 그 데이터를 제공하게 되고, 그렇지 않으면 inter-node network으로 연결된 외부 노드(remote node)의 메모리로부터 데이터를 가져오게 된다. 각 노드들은 외부 노드들의 접근에 관한 공유 정보들을 관리하기 위해서 디렉토리를 가지고 있다.

3.2. CCA(Cache Coherent Agent) 보드

CCA보드는 [그림 1]의 NIF를 나타내며, [그림 2]와 같이 프로세서 버스인 FSB(Front Side Bus)와 SCI(Scalable Coherent Interface)를 연결한다. 이를 위해 보드 내부에는 FSB와의 인터페이스를 담당하는 PIF(Processor Interface), RACE 프로토콜을 위한 DC(Directory Controller)와 RC(Remote access cache Controller), SCI 전송계층 프로토콜을 담당하는 LC2로 구성된다. 특히 이중 SCI 네트워크를 위해 LC2는 LC2-E(LC2-Even)와 LC2-O(LC2-Odd) 두개로 구성된다. 이들 내부 모듈들(PIF, DC, RC, LC2-E, LC2-O) 사이는 버스로 연결되는데 PIF와 DC/RC 사이는 I-Link(Inside Link), DC/RC와 LC2-E/O 사이는 B-Link(Backside Link)라 한다[6].



[그림 2] CC-Agent Board Block Diagram

3.3. I-Link Bus

- 동기형 버스 : 모든 신호는 버스 클럭에 동기 된다.
- 패킷 전송 프로토콜 : 모든 전송은 패킷 형태로 전송된다.
- 분리형 프로토콜 : 명령어와 주소와 경우에 따라 데이터를 전송하는 요청 패킷과, 상태와 경우에 따라 데이터를 전송하는 응답 패킷은 독립적으로 진행된다.
- 64-비트 데이터 폭 : 제어 신호를 제외한 모든 정보는 8-바이트 버스를 통해 전송된다.
- 패리티 방어 : 데이터 버스에 전송되는 정보는 패리티로써 오류를 검출하며, 필요 시 재시도하는 기법을 사용한다.
- 중앙 집중형 중재 : 우선 순위를 이용하는 중앙 집중형 중재기법을 사용한다.

4. SMV를 이용한 명세 및 검증

4.1. 명세

4.1.1. 가정

SMV를 이용한 검증에서는 I-Link Bus에 다음과 같은 가정을 설정한다.

- FSB에서 PIF로 들어 오는 요청은 PIF의 PCPU(Pseudo CPU)가 non-deterministic하게 생성한다.
- B-Link에서 DC나 RC로 들어 오는 요청은 DC나 RC의 PCPU가 non-deterministic하게 생성한다.
- 각 모듈의 PCPU는 SD의 request queue가 가득 찬 경우는 더 이상 request를 발생하지 않는다.
- 각 모듈의 PCPU는 명령어 플릿만 있는 요청만을 생성한다. 즉, 각 모듈은 bus를 한 사이클 동안만 사용한다.
- RACE 프로토콜과 관계되는 DIR과 RAC의 state는 I-Link bus 부분만으로는 state 변화를 완전하게 할 수 없으므로 non-deterministic

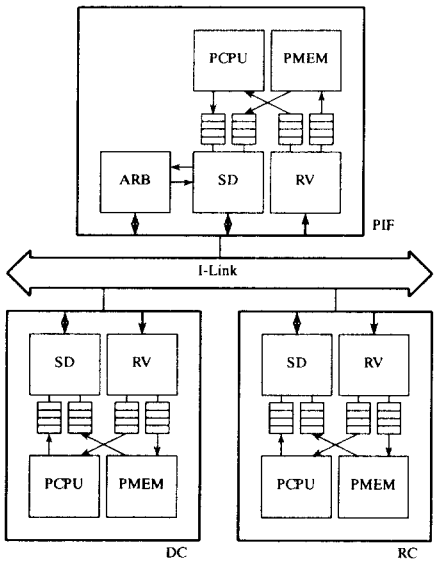
하게 변화한다고 가정한다.

• 각 모듈로 들어오는 요청들은 PMEM(Pseudo Memory)가 외부 모듈들을 대신해서 처리하며, 그에 대한 응답을 SD의 reply queue에 삽입한다.

4.1.2. 추상화 모델

[그림 3]은 위의 가정을 기반으로 한 I-Link bus의 추상화 모델이다. 검증 대상은 I-Link Bus와 PIF, DC, RC 모듈로 구성된다.

PIF는 버스를 중재하는 ARB(Arbiter)와 I-Link Bus로 패킷을 보내고 받는 역할을 하는 SD(sender)와 RV(receiver), 외부 모듈들을 대신하는 PCPU, PMEM과 SD와 RV가 사용하는 4개의 큐로 구성된다. DC와 RC는 PIF에서 ARB를 제외한 다른 요소들은 동일하며, 추가로 DIR과 RAC의 state 값을 가지고 있는 변수를 가지고 있다.



[그림 3] I-Link Bus의 abstract model

4.2. 검증

4.2.1. 검증한 property

(1) Queue overflow

각 모듈 안의 큐가 overflow되는지를 검사한다. 4개의 큐 중에서 SD의 request 큐는 full인 경우 PCPU가 더 이상 요청을 발생시키지 않도록 설계했으므로, 검사할 필요가 없다. 또한 RV의 reply 큐와 request 큐는 PCPU와 PMEM가 패킷이 들어 오는 족시 처리하므로 overflow가 발생하지 않는다. 그러므로, SD의 reply 큐만을 검사 대상으로 한다.

(2) 중재 우선순위

버스 중재기는 요청보다 응답을 높은 우선 순위를 주며, 응답 시에는 RC, DC, PIF순으로, 요청 시에는 PIF, RC, DC순으로 중재를 한다. 이러한 우선순위가 항상 잘 지켜지는지를 검사한다.

(3) Send Priority

각 모듈의 SD는 요청과 응답 패킷을 모두 송신해야 할 경우 응답 패킷에 우선순위를 준다. 이러한 특성이 만족하는지를 검사한다. 즉, request queue와 reply queue에 패킷이 모두 있는 경우 응답 패킷을 먼저 처리해야 한다.

(4) 중재 요청 신호

각 모듈은 마지막 데이터 전송 주기 한 사이클 이전까지만 iREQ를 구동 시켜야 한다. 즉, 한 개의 버스 클럭만큼 데이터를 전송하는 경우 iGRT 신호를 받으면 바로 다음 클럭에서 iREQ 신호를 걸어 내야 한다.

(5) Liveness

Liveness 특성은 I-Link Bus에 각 모듈이 연결되어서 각 모듈에서 발생한 요청 패킷이 starvation이 발생하지 않고, 버스 허가를 받아서 receive 모듈에 전달되고, 또한 요청 패킷이 receive 모듈의 PMEM에서 처리되어 응답 패킷이 다시 버스 허가를 받아서 요청을 발생시킨 모듈까지 돌아오는지를 확인하는 특성이라 하겠다.

4.2.2 검증결과

검증 결과 4.2.1에서 설명한 다섯 가지 property들이 모두 만족함을 알 수 있었다. 따라서, I-Link Bus와 각 모듈들이 중재기 우선순위, send 우선 순위, 중재 요청 신호의 관리, liveness등의 property들을 만족한다라는 것을 검증하는 것이다. [그림 4]는 검증에 사용된 리소스의 양을 보여 준다.

```
Resources used
==============
user time.........................14856 s
system time.......................1.94 s
BDD nodes allocated...............3345143
data segment size.................1788976
```

[그림 4] 사용된 리소스

5. 결론

정형 기법은 정형 논리와 수학에 기반을 둔 방법으로 자연어가 내포하는 애매모호함과 불확실성을 배제할 수 있으며, 시스템이 어떤 특성을 만족하는지를 검증하기 때문에 최소한 검증된 특성에 대해서는 완전히 믿을 수 있다. 특히, BDD를 기반으로 한 심볼릭 모델 체킹은 대형 하드웨어나 소프트웨어 시스템에서도 잘 적용될 수 있으며, 다양한 특성들을 검증할 수 있다.

본 연구는 하드웨어 검증에 있어서 정형 검증 도구를 이용함으로써 시뮬레이션의 한계를 극복할 수 있는 방법론을 제시하였으며, 정형 검증 기법이 하드웨어 개발 시간의 단축면에서 많은 효과가 있다는 점을 제안한다.

SMV를 이용한 I-Link Bus의 검증 결과 I-Link Bus와 PIF, DC, RC 모듈들이 중재기 우선 순위, send 우선 순위, 중재 요청 신호의 관리, liveness등의 특성들을 모두 만족한다라는 것을 검증하였다.

향후 과제로는 모델 체킹을 비롯한 정형 검증 기법을 다양한 분야의 하드웨어나 소프트웨어 검증에 적용하는 것이라 하겠다.

6. 참고 문헌

[1] David L. Dill, John Rushby, Acceptance of Formal Methods : Lesson from Hardware Design, *IEEE Computer*, vol. 29, no. 4, pp. 16-30, April 1996.

[2] Edmund M. Clark and Jeannette M. Wing, "Formal Methods: State of the Art and Future Directions", *ACM Computing Surveys*, pp.626-643, Dec. 1996.

[3] Kenneth L. McMillan, *SYMBOL MODEL CHECKING*, Kluwer Academic Publisher 1993.

[4] R. E. Bryant, Graph-Based Algorithms for Boolean Function Manipulation, IEEE Transaction Computers, vol. 35, no. 6, pp.677-691, Aug. 1986.

[5] Ki, Ando., *et al.*, "RACE Protocol, Ver 1.1", Computer System Dept. Computer Software Tech. Lab. ETRI TM-3100-1999-012, V1.1 Draft, Aug. 1999.

[6] Ki, Ando., *et al.*, "CCA 보드에서 사용하는 전송형태와 패킷 구성", Computer System Dept. Computer Software Tech. Lab. ETRI TM-3100-1999-036, V1.1 Draft, Aug. 1999.

[7] Ki, Ando., *et al.*, "I-Link 버스 인터페이스 구현", Computer System Dept. Computer Software Tech. Lab. ETRI TM-3100-1999-071, V1.0 Draft, May 1999.