ESTEREL을 이용한 MESI프로토콜의 정형 설계 및 검증

김민숙° 김진현 최진영

고 려 대 학 교 컴 퓨 터 학 과

{mskim, jhkim, choi} @formal.korea.ac.kr

Formal Design and Verification of MESI Protocol Designed by ESTEREL

Min-Suk Kim° , Jin-Hyun Kim, Jin-Young Choi

Dept. of Computer Science and Engineering, Korea University

요 약

캐쉬 일관성 프로토콜의 하나인 MESI 프로토콜은 다중 프로세싱 환경에서 각각의 프로세서와 메모리 사이의 데이터 일관성을 유지하기 위해 캐쉬, 메모리 등의 통신 개체들을 조정하는 일종의 규칙들 중 하나이다. 프로세서의 수가 많아지고 시스템이 복잡해 질 경우 MESI 프로토콜을 정확하게 설계하고 그 동작을 분석하기는 매우 어렵다. 본 연구에서는 정형기법 도구인 ESTEREL을 이용하여 MESI 프로토콜을 설계하고 그 동작의 안정성을 검증하여, 시스템의 정확성과 안정성을 보장하는 방법에 대해 논한다.

1. 서 론

컴퓨터를 이용하는 응용 분야들 중에는 현존하는 초고속 컴퓨터의 **성능으로도 처리 시간이 매우 오래 걸리는 것들이 많이 있다. 그 예로는 인공지능, 로보틱스, 신호 처리, 유체 역학, 일기 예보 등을 들 수 있다. 컴퓨터의 속도를 결정하는 첫 번째 요소인 프로세서의 속도가 계속 향상되고는 있지만 필요한 정도의 시스템 성능을 얻기에는 여전히 부족하다.** 따라서 최근 대부분의 **컴퓨터 설계에서는 성능 향상을 위한** 방법으로서 병렬처리 기술이 널이 사용되고 있다. 공유-기억장치 구조를 가진 병렬 처리컴퓨터에서는 여러 개의 프로세서들이 동시에 기억장치를 액세스하는 경우가 빈번히 발생하기 때문에 기억장치 충돌에 의한 지연이 발생한다. 이 때 프로세서들간의 여러 개 데이터 복사본 사이의 일관성을 유지하기 위해서, 캐쉬, 메모리 등의 통신 개체들을 조정하는 일종의 규칙을 캐쉬 일관성 프로토콜[1]이라 한다. 현재까지 많은 캐쉬 일관성 프로토콜들이 제안되고 구현되었지만, 정형적으로 검증된 사례는 드물다고 하겠다. 정형 기법을 이용하면 자연어가 내포할 수 있는 애매 모호함이나 불확실성을 최소한으로 줄여 시스템을 구현하기 이전에 그 기능의 정확성을 검사할 수 있어 개발비용과 시간을 줄일 수 있다. 하지만 정형기법[2]을 도입한 설계 및 검증은 수학적 기호나 전문 용어로 기술된 사용자의 요구 명세를 검증한 경우로, 전문가의 설명이나 지식 없이는 이해하기 쉽지 않아 그 효용도가 낮은 편이었다.

본 논문에서 이에 대해 일반 사용자들이 이미 친숙한 범용 언어와 유사한 ESTEREL[3]을 이용하여 지금까지 제안된 여러 가지 캐쉬 일관성 프로토콜 중 1984년 일리노이 주립대에서 제안한 스누핑 프로토콜의 하나인 MESI 프로토콜[4]에 대해 ESTEREL이라는 동기적(Synchronous) Reactive 시스템에 대해 명세 및 검증 할 수 있는 toolset을 이용하는 과정을 보일 것이다. 2장에서는 캐쉬의 일관성을 유지하는 프로토콜인 MESI 프로토콜에 대해, 3장에서는 명세 및 검증 도구로 사용한 ESTEREL toolset에 대해, 4장에서는 ESTEREL을 이용한 MESI 프로토콜의 명세 및 검사, 검증에 대해 논한다. 끝으로 5장에서는 결론 및 향후 과제에 대해 제시한다.

2. 캐쉬 일관성 프로토콜

2.1 캐쉬(Cache)

전형적인 많은 프로그램을 분석하면, 주어진 시간에서의 메모리 참조는 국한된 영역에서만 이루어지는 경향이 있음을 알 수 있다. 이러한 현상을 참조의 국한성이라 한다. 이렇게 자주 참조되는 프로그램과 데이터가 속도가 빠른 조그만 메모리에 저장된다면 평균 메모리 접근 시간이 감소되며 따라서 프로그램의 총 수행시간이 단축된다. 이렇게 빠르고 조그만 메모리를 캐쉬 메모리라 하며, CPU와 주기억 장치사이에 놓인다. 캐쉬의 최적 설계를 위해서는 캐쉬 적중률이 극대화 돼야 하며, 캐쉬 액세스 시간은 최소화되고 주 기억 장치와 캐쉬간의 데이터 일관성 유지 및 그에 따른 오버헤드도 최소화돼야 한다.

2.2 캐쉬 일관성 문제(Cache Coherence Problem)

다중 프로세싱 환경에서는 캐쉬는 캐쉬 일관성 문제를 초래한다. 프로세서 여러 개가 공유 메모리 위치에 있는 내용의 복사본을 각각 캐쉬 할 때, 어떤 개별적인 수정도 전체 메모리의 비 일관성 문제를 가져온다. 캐쉬 일관성을 유지하는 방법은 여러 가지가 있다. 첫째 공유캐쉬를 사용하는 것이다. 둘째 공유 변수는 캐쉬에 저장하지 않는다. 셋째 버스 감시 메커니즘(Bus Snooping)을 이용한다. 이는 시스템 안에 있는 모든 프로세서가 메모리 입출력을 관찰할 수 있다는 특징을 갖는다. 캐쉬 안에 위치한 스누퍼 제어기는 다른 프로세서들에 의한 기억장치 액세스 동작의 주소를 검사하고, 그 결과에 따라 자신의 캐쉬 블록 상태를 조정한다. 이 동작을 위해 각 캐쉬 블록들은 현재의 상태를 나타내는 상태 비트(status bit)를 가지고 있으며, 상태의 수와 종류는 쓰기 방식 및 일관성 유지 프로

토콜에 따라 약간씩 다르다. 본 논문에서는 그 중 Write Back 방식의 MESI Protocol에 대해 명세 및 검증했다.

2.3 MESI Protocol

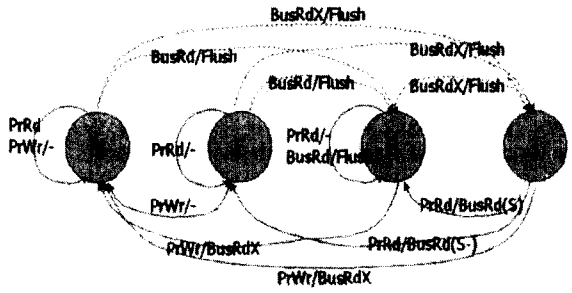
MESI 프로토콜에서는 각각의 캐쉬가 가질 수 있는 상태를 4가지로 나누고 있다.

- M(Modified) : 현재 캐쉬만이 유효(valid)한 값을 갖고 있음
- E(Exclusive) : 현재 캐쉬만이 그 블록을 갖고 있음
- S(Shared) : 현재 캐쉬에서 변경하지 않는 상태로 블록을 갖고 있음
- I(Invalid) : 현재 캐쉬에서 유효하지 않은 블록을 갖고 있음

프로세서는 PrRd(읽기)와 PrWr(쓰기)동작을 요청할 수 있다. 그러면 다른 캐쉬들에게 상태를 알리기 위해 Bus Transition이 발생하게 된다. 그 종류는 다음과 같이 2가지로 나눠볼 수 있다.

- BusRd : 이는 PrRd가 발생했는데 그 블록이 캐쉬에 없는 경우 발생한다. 프로세서는 그 결과로 data를 얻을 수 있기를 기다린다.
- BusRdX : 이는 캐쉬에 없는 블록이나 캐쉬에 는 존재하지만 modified(M)상태가 아닌 블록에 대해 PrWr이 요청되는 경우 발생한다. 이 경우 캐쉬 제어기는 버스에 주소를 올리고 수정상태(M)가 될 독점적인 copy본을 기다린다.

이러한 일련의 MESI 프로토콜의 동작을 표현한 상태 전이 다이어그램(state transition diagram)은 [그림1]과 같다.



[그림1] MESI 프로토콜 상태 전이 다이어그램

3. 정형 기법 도구

3.1 ESTEREL

ESTEREL은 프랑스의 Sophia-INRIA에서 연구 개발되고 있는 정형 명세 및 검증 toolset이다. ESTEREL 명세 언어는 하드웨어나 프로토콜 등을 명세하기 위해 개발된 언어로 Reactive 시스템의 동기적 프로그램을 작성하는데 사용하는 동기적 언어(Synchronous Language)이다. 이러한 동기적 언어는 완벽한 동시성 모델에 근거하며, 이러한 모델 내에서는 동시적 프로세스가 0시간 내에 계산과 정보의 교환이 일어나며, 입력 set에 대한 모든 반응은 시간을 소모하지 않는 instantaneous한 것으로 간주된다. ESTEREL을 사용하게 되면 두 가지 이점을 기대할 수 있다. 첫째, 제어를 처리하는 함수들(control-handling functions)을 표현하기 쉬울 뿐 아니라 명세를 하는 과정에서 발생할 수 있는 실수를 C/C++을 사용하는 것에 비해 최소화시킬 수 있다. 둘째로는 하드웨어 디자인이나 함수적 프로세서(functional process)와 같은 분야에서 시뮬레이션이나 검증하는데 있어 최적화를 시켜준다. XES[5]라는 검사 도구는 ESTEREL로 명세 된 시스템을 설계자나 구현자가 그래픽적으로 시뮬레이션 하는 기법을 제공해준다. 따라서 사용자는 일정한 입력 set에 대해 직접적으로 따라가 볼 수 있는 기능을 제공한다. 이러한 설계기법을 이용하면 설계 단계에서 오류를 찾을 수 있기 때문에 오류가 없는 설계를 통해 구현자는 보다 완벽한 구현을 할 수 있다.

3.1 XEVE(an ESTEREL Verification Environment)

XEVE[6]라는 도구는 implicit 하게 정의된 유한 상태 기계를 대상으로 모델 체킹 기법을 이용하여 시스템을 검증할 수 있다. FSM은 BLIF(Berkeley Logical Interchange Format) 형태로 표현된다. XEVE 는 두 가지 검증 기능을 제공한다. 첫 번째는 바이시뮬레이션 등가 관계를 이용해 최소화된 유한 상태 기계를 이용하는 것이고 두 번째는 위배 신호의 발생 가능 여부를 확인하는 것이다. 만약 시그널이 발생 가능한 경우라면 XES를 통해 그 경우를 따라가 볼 수 있다.

4. MESI 프로토콜의 정형 명세 및 검증

4.1 MESI 프로토콜의 정형 명세

본 논문에서는 3개의 캐쉬로 구성된 다중 프로세싱 환경을 대상으로 한다. 이를 ESTEREL 언어를 사용하여 구현한 결과는 [그림 2]와 같다. 3개의 캐쉬에서는 각각의 프로세서의 읽기 신호(PrRd)와 쓰기 신호(PrWr)를 입력으로 받아들여 캐쉬라는 모듈의 작동을 통해 BusRd나 BusRdX와 같은 시그널들을 브로드캐스팅하고, 나머지 2개의 캐쉬는 이를 감시하고 있다가 버스에 신호가 들어오게 되면 이에 적절한 동작을 0시간 내에 취함으로써 전체 Cache들간의 일관성을 유지시킨다. ESTEREL 언어는 오토마타를 상징적(symbolic)으로 표현함으로써 프로그램의 양을 줄일 수 있다. 그리고 신호들을 브로드캐스팅 시킬 수 있기 때문에 신호들을 넘겨주는 추가적인 동작을 간략화 시킬 수 있다는 장점이 있다.

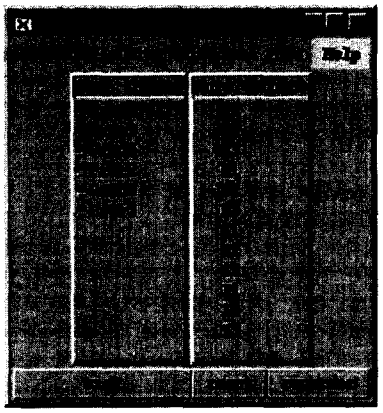
```
module MESI:
input PrRd1, PrWr1, PrRd2, PrWr2,  PrRd3, PrWr3;
output M1, M2, M3, E1, E2, E3, S1, S2, S3, I1, I2, I3;
signal BusRdX1, BusRdX2, BusRdX3, BusRd1, BusRd2, BusRd3,
        Flush1, Flush2, Flush3, Flush_1, Flush_2, Flush_3 in
run Cache[signal PrRd1/PrRd, PrWr1/ PrWr,
      BusRd1/BusRd, BusRd2/BusRd_a, BusRd3/BusRd_b,
      BusRdX1/BusRdX,BusRdX2/BusRdX_a, BusRdX3/BusRdX_b,
      Flush1/Flush, M11/M, E1/E, S1/S, I1/ I]
||
run Cache[signal PrRd2/PrRd, PrWr2/ PrWr,
      BusRd2/BusRd, BusRd1/BusRd_a, BusRd3/BusRd_b,
      BusRdX2/BusRdX,BusRdX1/BusRdX_a, BusRdX3/BusRdX_b,
      Flush2/Flush, M12/M, E2/E, S2/S, I2/ I]
```

```
||
run Cache[signal PrRd3/PrRd, PrWr3/ PrWr,
      BusRd3/BusRd, BusRd1/BusRd_a, BusRd2/BusRd_b,
      BusRdX3/BusRdX,BusRdX1/BusRdX_a, BusRdX2/BusRdX_b,
      Flush3/Flush, M13/M, E3/E, S3/S, I3/ I]
end signal
end module
```

[그림2] ESTEREL로 명세한 MESI 프로토콜의 일부

4.2 MESI 프로토콜의 정형적 검사 및 검증

ESTEREL로 명세된 MESI 프로토콜은 XES를 통해 [그림 3]과 같이 그래픽적으로 결과를 검사해볼 수 있다.



[그림3] XES를 이용한 MESI 프로토콜 시뮬레이션 화면

본 논문에서 MESI 프로토콜이 만족 해야 할 요구 사항은 다음과 같다.

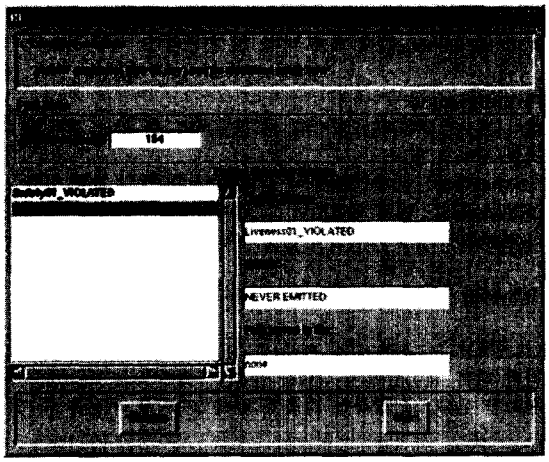
· Safety1: 만약 2개 이상의 캐쉬가 동시에 수정(M)상태에 놓이게 된다면 Safety_VIOLATED라는 신호를 발생해야 한다.

Safety1 :=Always(not (AllPast(M1 and M2))) and
(not (AllPast(M1 and M3))) and
(not (AllPast(M2 and M3)))

· Liveness1: 만약 프로세서가 쓰기 동작을 요청했을 때 자신은 수정(M)상태가 되고 다른 캐쉬들은 무효(I)상태로 변경 시키지 못한다면 Liveness_VIOLATED라는 신호를 발생 해야 한다.

Liveness1:=Always(PrWr1 -> (M1 and I2 and I3)) and
(PrWr2 -> (M2 and I1 and I3)) and
(PrWr3 -> (M3 and I1 and I2))

위와 같이 명세된 특성을 XEVE를 이용하여 검증하면 [그림 4]와 같은 결과를 얻게 된다. 위의 Safety01_VIOLATED신호와 Liveness01_VIOLATED신호가 발생하지 않음을 알 수 있



[그림4] XEVE를 이용한 검증 결과

다. 즉, MESI 프로토콜은 정상적인 캐쉬 일관성 동작을 수행하는 프로토콜임을 증명할 수 있다.

5. 결론 및 향후 과제

지금까지 다중 프로세싱 환경에서 프로세서의 캐쉬들과 메모리 사이의 데이터의 일관성을 유지하기 위한 MESI 프로토콜을 정형 검증 도구인 ESTEREL tool set을 이용하여 명세 및 검사, 검증해 보았다. ESTEREL로 명세한 시스템은 XES를 이용하여 그래픽적인 시뮬레이션을 해볼 수 있었다. 그리고 XEVE를 통한 검증을 통해 3개의 프로세서로 구성된 다중 프로세싱 환경에서 MESI 프로토콜이 Safety와 Liveness를 만족함을 보였다. 향후 과제로는 ESTEREL toolset을 이용한 정형 명세 및 검증 기법을 한국전자통신연구원에서 개발한 PDLSystem을 위한 디렉토리 기반 캐쉬 일관성 프로토콜인 RACE프로토콜에 적용해보고자 한다. 그리고, XEVE로 검증한 결과를 모델 체커인 SMV[7]를 이용하여 검증한 결과[8]와 비교하여 ESTEREL의 효율성에 대해 분석해보고자 한다.

참고 문헌

[1] 김종현, *병렬 컴퓨터 구조론*, 생능 출판사, 1996

[2] Andrew Harry, *Formal Methods-FACT FILE, VDM and Z*, Wiley, 1996

[3] G.Berry, "The Esterel v5 Language Primer Version v5_91", June 5, 2000

[4] David E. Culler ,Jaswinder Pal Singh, *Parallel Computer Architecture*, Morgan Kaufmamn Publishers, 1998

[5] G.Berry and the Esterel Team, " The Esterel v5_91 System Manual", June 5, 2000

[6] Amar Bouali, "XEVE : an Esterel Verification Environment(Version v1_3)", December 1997

[7] Kenneth L. McMillan, *Symbolic Model Checking*, Kluwer Academic Publisher, 1993

[8] 남원홍, "SMV를 이용한 RACE 프로토콜과 CCA보드의 정형 검증 및 테스팅", 2001