멀티 프로세서 시스템-온-칩(MPSoC)을 위한 버스 매트릭스 구조의 빠르고 정확한 성능 예측 기법

김성찬° 하순회
서울대학교 전기컴퓨터공학부
sungchan.kim@iris.snu.ac.kr, sha@iris.snu.ac.kr

Fast and Accurate Performance Estimation of Bus Matrix for Multi-Processor System-on-Chip (MPSoC)

Sungchan Kim° Soonhoi Ha
School of Electric Engineering and Computer Science, Seoul National University

1. 서론

반도체 공정의 발달로 인해 멀티 프로세서 시스템-온-칩 (MPSoC)을 위해 점점 많은 수의 프로세서 및 메모리 코어들을 하나의 칩에 집적하는 것이 가능해졌다. 이로 인해 칩 내부의 프로세서 코어들을 연결하는 통신 구조는 성능, 전력 소모 및 하드웨어 면적 등의 측면에서 큰 영향을 끼치게 되었고 이의 최적화가 MPSoC 설계에서 핵심 단계 중의 하나가 되었다 [1]. 버스 매트릭스 기반 통신 구조는 현재 널리 사용되는 버스 프로토콜과의 호환성을 유지하면서 고성능을 제공할 수 있는 통신 구조로서 이미 상용화된 IP들이 활발하게 이용되고 있다.
공유 버스 기반의 통신 구조의 탐색에 대한 기존 연구들에 의하면, 다양한 인자들에 의해 매우 방대한 설계 공간이 형성되기 때문에 주어진 개발 시간 내에 최적의 통신 구조를 얻기 위해서는 각 구조들에 대해 정확하고 빠르게 성능을 예측하는 것이 필요하다 [2][3].
그러나 현재까지 개발된 버스 매트릭스 통신 구조의 설계 공간 탐색 방법은 시간이 오래 걸리는 시뮬레이션에 의존하고 있다 [4][5]. 이를 해결하기 위하여 본 논문에서는 주어진 응용의 메모리 접근 패턴의 통계 정보에 기반한 큐잉 이론을 이용하여 버스 매트릭스 통신 구조의 성능 예측 기법을 제시하고자 한다.
기존 연구들과 비교해 본 논문은 두 가지 의의를 갖는다. 첫 번째, 제시된 성능 예측 기법은 multiple-outstanding transaction 및 메모리 시스템의 접근 시간 모델에 general distribution을 사용하여 실제적인 버스 매트릭스 구조 분석을 가능하게 한다. 또한 multiple-outstanding transaction을 지원하는 버스 매트릭스에서 동시에 실행되고 있는 transaction들의 평균 개수를 예측할 수 있어, 버스 매트릭스 구현에서 transaction을 저장해야 하는 버퍼의 개수를 최소화할 수 있도록 해준다. 두 번째, 제안하는 성능 예측 기법은 사이클 단위의 정확도를 갖는 시뮬레이션과 비교해 훨씬 빠르면서 매우 작은 성능 예측 오차를 갖기 때문에 빠른 설계 공간 탐색을 가능하게 해 준다.

2. 버스 매트릭스의 성능 예측 기법

버스 매트릭스는 2가지 측면에서 고성능을 제공한다. 첫 번째로 각 프로세서 코어(마스터)와 메모리 코어(슬레이브) 사이에 독립적인 버스를 사용하여 데이터 접근 경로를 제공할 수 있다. 이를 통해 각 프로세서 코어들의 병렬적인 메모리 접근을 가능하게 해 준다. 두 번째로 multiple-outstanding transaction을 사용하여 버스 사용 효율을 높이는 것이다. 각 마스터들은 transaction을 수행하기 위해 버스가 해당 마스터의 transaction을 수행하기 시작했다는 응답을 받아야 한다. 만약 버스가 동시에 하나의 transaction만 처리할 수 있을 때, 각 마스터들은 현재 처리되고 있는 transaction의 수행이 끝날 때까지 기다려야 한다. 반면 여러 개의 transaction들이 동시에 수행될 때 마스터들은 현재 transaction의 해당 슬레이브를 접근하는 동안 자신의 transaction에 대한 응답을 버스로부터 즉시 받을 수 있어 버스 사용 효율을 높일 수 있다.
본 논문에서 제안하는 성능 예측 기법은 2개의 입력을 필요로 한다. 하나는 마스터들의 집합 $\Phi_M=\{m_i \mid i=0,.., N_M-1\}$과 슬레이브들의 집합 $\Phi_S=\{s_i \mid i=0,..,N_S-1\}$를 갖는 버스 매트릭스이고, 다른 하나는 실행되는 응용에 의해 생기는 메모리 접근에 대한 통계적인 정보이다. 분석의 편의를 위해 하나의 버스는 하나의 슬레이브만 갖고 있다고 가정한다. 메모리 접근에 대한 통계 정보는 주어진 응용이 해당 프로세서 코어에서 실행될 때의 메모리 트레이스가 통신 구조에 의해 실행 시간에서의 어떠한 지연도 발생하지 않는 것을 가정하여 생성된다. 마스터 m_i이 슬레이브 s_j에 접근하는 transaction들의 간격을 나타내는 확률 변수를 $V_{i,j}$로 나타내며, 이는 $fv_{i,j}(v)=P\{V_{i,j}=v\}$로 정의되는 확률 밀도 함수를 갖는다. 또한 마스터 m_i이 슬레이브 s_j에 접근할 때 이에 대한 서비스 시간을 나타내는 확률 변수는 $L_{i,j}$이며, $fl_{i,j}(l)=P\{L_{i,j}=l\}$로 정의되는 확률 분포 함수를 갖는다. 임의의 확률 변수 A에 대해서 $E[A]$와 $E[A^2]$는 A의 평균(또는 기대값)과 분산으로 정의한다. 그러므로 $E[L_{i,j}]$는 마스터 m_i가 슬레이브 s_j를 접근할 때의 지연 시간 (또는 서비스 시간)에 대한 확률 변수 $L_{i,j}$의 평균을 나타내며, 마찬가지로 $E[V_{i,j}]$는 $V_{i,j}$의 평균을 나타낸다.
하나의 슬레이브 s_0를 갖는 버스 매트릭스를 고려한다. 표현의 편의를 위해 마스터 m_i가 슬레이브 s_0를 접근하는 간격과 접근시의 서비스 시간을 나타내는 확률 변수 $V_{i,0}$와 $L_{0,i}$대신 V_i와 L_i를 사용하기로 한다. 따라서 확률 밀도 함수 $fv_{i,0}(\cdot)$와 $fl_{0,i}(\cdot)$ 대신 $fv_i(\cdot)$와 $fl_i(\cdot)$를 사용한다.
본 논문에서는 버스 매트릭스 구조를 모델하기 위하여 M/G/1 큐잉 시스템을 이용한다. 'M'은 마스터들이

transaction을 발생시키는 간격이 exponential distribution이라는 것을 나타내며, 'G'는 서버(메모리 시스템)의 서비스 시간의 확률 분포(본 논문에서는 $fl_i(\cdot)$)가 general distribution임을 나타낸다. 마지막 '1'은 서버의 개수가 한 개임을 나타낸다. m_i이 transaction을 생성한 후 서버로부터 서비스를 받기 전까지 (즉 메모리에 대한 접근할 시작할 때까지) 기다려야 하는 확률 변수를 W_i라고 하자.이제 마스터 m_i가 기다려야 하는 평균 대기시간 $E[W_i]$ 을 구한다. 마스터 m_i가 transaction을 생성할 때 이전의 다른 마스터들로부터의 transaction들은 서비스를 받고 있거나 FIFO에서 대기하는 두 가지 경우 중의 하나이다. $P(B_i)$를 현재 마스터 m_i가 서비스를 받고 있을 확률이라고 하자. $E[R_i]$은 마스터 m_i가 서비스를 받고 있을 때 남아 있는 서비스 시간을 나타낸다. 그리고 N_i은 m_i이 평균적으로 발생한 transaction의 개수를 나타낸다. 마스터들은 하나의 transaction만을 생성하므로 이는 transaction이 발생되었는지 아닌지의 확률로 생각할 수 있다. 따라서 마스터 m_i가 기다려야 하는 시간을 나타내는 확률 변수 W_i의 평균 $E[W_i]$은 다음과 같다.

$$E[W_i] = \sum_{j=0,\forall j\neq i}^{N_M-1} \left(E[N_j]\cdot E[L_j] + P(B_j)\cdot E[R_j]\right)$$

다음으로, 서비스를 기다리는 모든 transaction의 개수를 나타내는 확률 변수를 N이라고 할 때 $N=\Sigma N_i$이고 $i=0,\ldots,N_M-1$이다. 현재 서비스를 받고 있는 transaction까지 고려한다면, 시스템에는 $E[N]+1$의 transaction들이 있고, 버스가 최대의 성능을 얻기 위해 필요한 (즉 transaction의 처리 시간과 다른 transaction들의 응답 시간을 최대로 중첩 시키기 위해 필요한) issue capability의 하계 값 B_L은 아래와 같다.

$$B_L = \lceil E[N]+1 \rceil$$

3. 실험 및 결과

다양한 마스터 개수(2부터 16)와 transaction 생성 비율(0.1부터 0.3)을 갖는 버스 매트릭스 구조에 대해 제안한 성능 예측 기법의 정확도를 분석한 결과가 그림 1에 나타나 있다. 마스터 개수와 transaction 생성 비율의 조합에 대해, 각 마스터들은 100,000번의 transaction을 갖도록 메모리 트레이스를 무작위로 100번씩 생성하였다. 각 transaction들은 2, 4, 8번의 메모리 접근들로 이루어질 수 있으며 이는 uniform distribution을 가정하여 선택된다. 따라서 transaction들에 대한 서비스 시간은 2, 4, 8 사이클 중의 하나가 된다. 각 마스터들은 병렬적으로 transaction들을 수행하며 모든 transaction들이 종료될 때까지의 시간을 성능 예측 기법으로 얻은 후 이에 대한 시뮬레이션과의 오차를 비교하였다. 예측 오차는 최대 6%를 넘지 않는다.

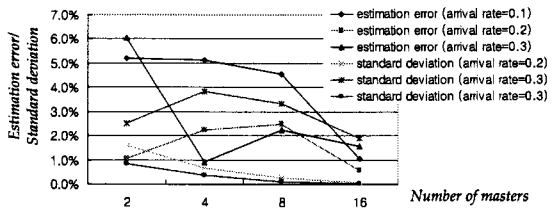

그림 1. 버스 매트릭스 구조에 대해 제안하는 예측 기법으로 얻어진 성능 예측 결과를 시뮬레이션과 비교한 오차 및 오차들의 표준 편차.

4. 감사의 글

본 연구는 BK21 프로젝트, 과학기술부 도약연구지원사업(R17-2007-086-01001-0), 삼성전자에 의해 지원되었다. 또한 서울대학교 컴퓨터신기술연구소와 IDEC은 본 연구에 필요한 기자재들을 지원해 주었다. 본 연구는 또한 한국전자통신연구원의 SoC 핵심설계인력양성사업에 의해 부분적으로 지원되었다.

5. 참고 문헌

[1] L. Benini and G. De Micheli, "Networks on Chips: A new SoC paradigm," Computer, vol.35. no.1, pp.70-78, Jan. 2002.
[2] S. Kim and S. Ha, "Efficient Exploration of Bus-Based System-on-Chip Architectures," IEEE TVLSI, vol.14, no.7, pp.681-692, July 2006.
[3] K. Lahiri, A. Raghunathan, and S. Dey, "Design space exploration for optimizing on-chip communication architectures," IEEE TCAD, vol.23, no.6, Jun. 2004
[4] S. Pasricha, N. Dutt, and M. Ben-Romdhane, "Constraint-driven bus matrix synthesis for MPSoC," ASP-DAC, pp.30-35, Jan. 2006.
[5] O. Ogawa et al, "A practical approach for bus architecture optimization at transaction level," DATE, pp.176-181, Mar. 2003.