MPSoC 프로그래밍 플랫폼과 재겨냥성 컴파일러 연동을 위한 새로운 응용 기술방법*

김용주, 이종원, 박상현, 윤종희, 조두산, 권용인, 백윤흥
서울대학교 공과대학 전기컴퓨터 공학부
e-mail : yjkim, jwlee, shpark, jhyoon, dscho, yikwon@optimizer.snu.ac.kr, ypaek@snu.ac.kr

New application programming approach for MPSoC programming platform

Yongjoo Kim, Jongwon Lee, Sanghyun Park, Jonghee Yoon, Doosan Cho, Yongin Kwon, Yunheung Paek
School of Electrical Engineering, Seoul National University

요 약

최근들어 MPSoC 프로그래밍 방법에 대한 많은 연구들이 이루어지고 있다. 예전부터 연구가 진행된 모델 기반 프로그래밍 접근이나 UML 같은 모델기반 언어부터 최근에 많이 연구되고 있는 MPI[1] 나 OpenMP[2] 기반의 프로그래밍 방법, 그리고 그 외에도 다양한 접근 방식의 방법론이 연구되어 있다. 하지만 현재까지 대부분의 연구는 최종 결과물이 C 언어 형태로 나오게 되어 있다. 즉 MPSoC 환경을 위한 컴파일러가 따로 제작되어야 하고 이 점은 다양한 이종 MPSoC 환경이 존재한다는 점에서 컴파일러 제작에 많은 부담이 발생한다. 본 논문 본인이 이전에 연구했던 MPSoC 프로그래밍 플랫폼과 플랫폼에서 사용되는 입력 정보의 형태를 설명한다. 그리고 입력정보 형태를 변형하여 재겨냥성(retargetable) 컴파일러와 연동이 가능하게 하여 최종 결과물을 바이너리 형태로 생성할 수 있도록 한다.

1. 서론

오늘날 많은 임베디드 SoC(System-on-Chip) 설계자들이 직면한 대규모 SoC 설계의 어려움, 복잡한 검증 과정, 버그 교정 비용의 증가, 하드웨어/소프트웨어의 후기 통합, 업계 표준의 복잡도 증가 및 변화 등과 같은 문제의 해결 방안으로, SoC 내부에 다중의 프로세서들을 포함하는 MPSoC(Multi-Processor System on Chip) 플랫폼을 채용하는 경향이 늘어나고 있다. 이미 많은 전문가들은 반도체의 미세화 공정에 의존하여 칩의 성능 향상을 지속하는 것이 더 이상 어려워졌음을 주장하며, MPSoC 로의 전환의 필요성을 강조하고 있다. MPSoC 시스템이 가지는 대표적 장점은 고품질의 멀티미디어 서비스 등과 같은 복잡한 병렬 처리 연산을 필요로 하는 임베디드 시스템 응용들을 내부의 멀티프로세서들을 이용하여 효과적으로 처리할 수 있다는 것이다. 또한, 프로세서 상에서 프로그램이 수행되도록 높은 프로그램 수월성(Programmability)을 제공하여 Time-to-market 효과가 증대 된다는 점과 함께, 일반적으로 동일 성능을 내는 고성능의 단일 프로세서 시스템에 비해 전력소모를 줄일 수 있어서, 휴대형 기기 등에서 적합하다는 장점도 가지고 있다.

MPSoC 는 내부 프로세서의 구성에 따라, 크게 동일한 프로세서들로 SoC 를 구성하는 동종(Homogeneous) MPSoC 와 상이한 프로세서들로 구성된 이종(Heterogeneous) MPSoC 로 나뉠 수 있다. 동종 MPSoC 에 비해 이종 MPSoC 에서는 응용프로그램을 구성하는 다양한 작업들을 각기 특성에 맞는 프로세서들을 이용하여 효율적으로 수행하기 때문에 성능/전력소모 비를 최대화 시킬 수 있다. 하지만 이종 MPSoC 의 많은 장점과 최근에 급속히 달성된 여러 MPSoC 하드웨어 개발 성과에도 불구하고, 이들이 현재 널리 사용되지 못하고 있는 가장 큰 원인은 대다수의 전문가들이 지적하듯이 MPSoC 내부에 포함된 상이한 구조를 가지는 여러 프로세서들을 효과적으로 동시에 프로그래밍토록 하는 소프트웨어 방법론이 제대로 확립되어 있지 않기 때문이다.[3] 그래서 이번

* 본 연구는 교육과학기술부/한국과학재단 우수연구센터육성사업(R11-2008-007-01001-0), 지식경제부 출연금으로 ETRI, SoC 산업진흥센터에서 수행한 ITSoC 핵심설계인력양성사업, 서울시 산학연 협력사업, 2008 년도 정부(교육과학기술부)의 재원으로 한국과학재단의 국가지정연구실사업(R0A-2008-000-20110-0), 지식경제부 및 정보통신연구진흥원의 대학 IT 연구센터 지원사업(IITA-2008-C1090-0801-0020), 교육과학기술부 도약연구 지원사업 (R17-2007-086-01001-0), 지식경제부 및 정보통신연구진흥원의 IT 원천기술개발사업[과제관리번호: 2006-S-006-02, 과제명: 유비쿼터스 단말용 부품/모듈] 의 지원을 받아 수행되었습니다.

논문에서는 본인이 예전에 연구한 소프트웨어 프로그래밍 방법론을 소개하고 재겨냥성 컴파일러[4]와의 연동을 위해 아키텍처 정보 기술을 확장하는 방법에 대해 논의하겠다. 2 장에서는 MPSoC 프로그래밍 framework 을 간단히 설명하고, 3 장에서는 재겨냥성 컴파일러를 위한 아키텍처 정보를 변경하는 방법에 대해 논의 하고, 4 장에 결론을 맺는다.

2. CIC

MPSoC 프로그래밍 framework 은 그림 1 과 같은 과정을 거친다. 처음의 입력은 CIC(common intermediate code) 라는 우리가 제안한 입력형태로 들어간다. CIC 는 2 부분으로 구성되어 있는데 C 언어 함수 형태로 응용의 알고리즘을 기술한 작업 코드 부분과 xml 형태로 목표 아키텍처의 정보를 기술한 아키텍처 부분으로 나뉜다.

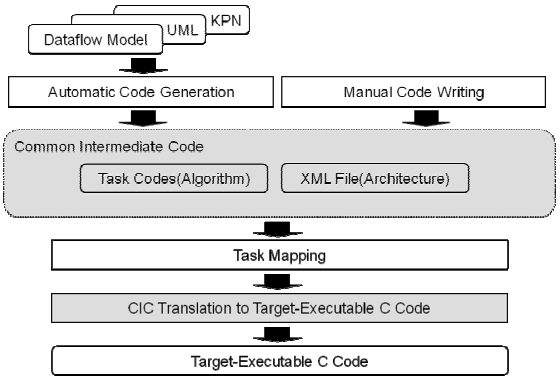

그림 1 병렬 프로그래밍 소프트웨어 생성을 위한 framework

작업 코드 부분은 전체 응용을 함수형태의 작은 작업으로 만들어서 각 작업마다 하나씩 기술해 준다. 그림 2 는 h.263 복호기를 6 개의 작업으로 분리해 낸 예제이다. 예제에서는 복호기를 VLD 와 3 개의 Macroblock 복호화, 움직임 보상, 출력으로 작업을 분리하였다. 각각의 작업은 C 코드로 생성되어 독립적인 C 코드로 만든다. 이 C 코드는 목표 독립적인 코드로 아직 컴파일 될 수 없는 형태이다. 추후 CIC 변환기를 거치면서 목표 의존적인 코드로 바뀌어 컴파일 될 수 있게 된다. 각 작업들은 다른 작업들과 독립적으로 기술되어 있으며, 작업간의 연결관계와 데이터 이동은 아키텍처 부분에서 기술된다.

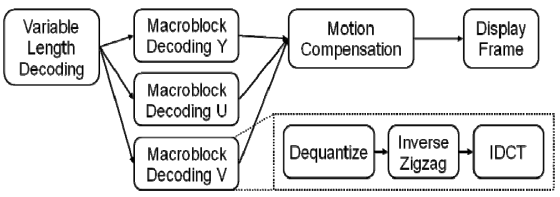

그림 2 H.263 복호기 작업 기술 예제

아키텍처 부분은 알고리즘 코드가 수행될 목표 아키텍처에 대한 정보가 기술되는 부분이다. 이 부분에서는 하드웨어에 대한 기술과 작업의 제약사항에 대한 기술, 그리고 통신 구조에 대한 기술로 나누어져 기술되어 있다. 하드웨어 기술 부분은 프로세서의 종류와 개수, 메모리 주소와 영역, 하드웨어 IP 의 주소와 위치, OS 유무 등의 정보를 포함한다. 제약사항의 기술은 작업의 주기나 데드라인 같은 시간적 제약정보, power 소모에 대한 제약, 메모리 사용에 대한 제약 등의 정보를 포함한다. 통신구조에 대한 기술은 각 프로세서와 메모리가 어떠한 구조로 어떻게 연결되어 있는지, 그리고 작업들의 종속성과 데이터 이동을 기술한다.

입력된 CIC 코드는 CIC 변환기를 거쳐서 목표 독립적인 코드에서 목표 의존적인 코드로 변환이 된다. 변환기는 아키텍처 코드를 읽고 분석하여 작업 코드에 어떤 정보를 반영해야 할지를 추출한다. 추출한 정보를 바탕으로 알고리즘 코드가 목표 아키텍처에서 정상적으로 수행될 수 있도록 API 변환이나 통신 코드등을 자동으로 작성하고 최적화된 성능이 나올 수 있도록 작업을 프로세서에 맵핑하는 작업을 수행한다. 그리고 각 프로세서에 할당된 작업을 묶어줄 수 있는 인터페이스 코드를 자동 생성한다. 변환 결과로 생성된 코드는 컴파일 과정만 거치면 목표 아키텍처에서 수행될 수 있는 코드가 된다. 좀 더 자세한 정보는 논문[5]에 설명되어 있다.

3. 확장된 CIC 아키텍처 기술

앞에서 제안한 CIC 를 사용한 프로그래밍 방법은 아키텍처 정보만 변경하여 다른 아키텍처에서 코드를 쉽게 재사용할 수 있고 모델 기반 언어 형태를 띄고 있어서 코드의 이해가 쉽다는 점, 그리고 C 언어로 되어 있어 이전 프로그래머들에게 거부감이 없다는 점에서 MPSoC 용 병렬 프로그래밍 방법으로 충분히 가치가 있다. 하지만 CIC 프로그래밍 방식을 단순한 프로그래밍 방법론에서 그치지 않고 재겨냥성 컴파일러와 같이 사용할 수 있도록 발전시킨다면 MPSoC 용 컴파일러를 개발할 시간을 줄일 수 있어 훨씬 빠른 Time-to-Market 이 가능해 진다.

앞에서 제안한 아키텍처 기술은 하드웨어 정보 기술, 제약정보 기술, 통신구조 기술로 이루어져 있다. 이 정보들은 재겨냥성 컴파일러에서 필요한 정보들을 포함하고 있으며, 추가적으로 필요한 정보는 ISA 정도만 추가적으로 기술해 주면 된다. 그래서 CIC 의 아키텍처 기술 부분에 ISA 정보를 네 번째 기술로 추가한다면 본인의 연구실에서 만든 재겨냥성 컴파일러를 사용해 MPSoC 용 컴파일러를 얻을 수 있다.

4. 결론

확장된 CIC 아키텍처 기술을 사용하여 재겨냥성 컴파일러를 연동한다면 사용자가 중간에 전혀 간섭하지 않아도 수행가능한 바이너리 코드가 생성되도록 하는 MPSoC 용 seamless 컴파일러를 구현할 수 있다.

현재의 재겨냥성 컴파일러는 각각의 프로세서 용으로 컴파일러를 생성한다. 사용자는 이 컴파일러를 사용하여 각각의 작업 코드를 컴파일하여 바이너리로 만들어내게 된다. 하지만 중간과정을 이어주는 부분만 구현한다면 추후 진정한 MPSoC 용 seamless 컴파일러가 구현될 수 있다.

MPSoC 가 최근 각광받은 연구 분야로 떠오르면서 많은 분야에서 연구가 이루어지고 있다. 하지만 아직 제대로 된 병렬 프로그래밍 방법론이 갖추어 지지 않았다. 본 논문은 이런 프로그래밍 방법론 연구의 발전에 기여를 할 수 있을 것이라고 생각한다.

참고문헌

[1] MESSAGE PASSING INTERFACE FORUM. 1994. MPI: A message-passing interface standard. Int. J. Supercomput. Appl. High Perform. Comput. 8, 159–416.

[2] OPENMP ARCHITECTURE REVIEW BOARD. 1998. OpenMP C and C++ application program interface version 1.0. http://www.openmp.org.

[3] Martin, G.,, “Overview of the MPSoC design challenge”, In Proceedings of the 43rd Annual Conference on Design Automation, 274–279. 2006.

[4] 안민욱 , 백윤흥 , 조정훈, 응용프로그램에 특화된 명령어를 통한 고정 소수점 오디오 코덱 최적화를 위한 ADL 기반 컴파일러 사용, 정보처리학회논문지 2006 년 8 월,

[5] S. Kwon, Y. Kim, W. Jeun, S. Ha, Y. Paek, “A Retargetable Parallel-Programming Framework for MPSoC”, ACM Transactions on Design Automation of Electronic Systems(TODAES) 2008