임베디드시스템을 사용한 시스템온칩

박춘명*

*충주대학교 전기.전자 및 정보공학부 컴퓨터공학전공

The SoC using Embedded Systems

Chun-Myoung Park*

*Computer Engineering Major, School of EEIE, Chungju National University

E-mail : cmpark@cjnu.ac.kr

요 약

본 논문에서는 임베디드세스템에 기초를 둔 시스템온칩을 구성하는 방법을 제안하였다. 제안한 방법은 이전의 방법에 비해 좀 더 콤팩트하고 효과적이다. 이 방법은 높은 수행시뮬레이션을 요구하고 하드웨어/소프트웨어 통합설계 툴을 사용하여 구현을 위한 실행 가능한 규격화된 적절함을 요구한다. 시스템 인터페이스 처럼 이미 존재하고 있는 부품의 재사용은 지원되지만, 작업 이후는 단지 하드웨어/소프트웨어 통합설계 툴의 프로그램에 의해 수행되어진다. 실제 설계 흐름은 모든 프로세스를 통하여 요구되는 구현으로부터 모든 설계 단계 사이의 궤환을 허용하게끔 설명되어진다. 향후 좀 더 진보된 임베디드시스템에 기초를 둔 시스템온칩을 구성하는 방법이 요구된다.

ABSTRACT

This paper presents a method of constructing the system-on-chip(SoC) based on embedded systems. The proposed method is more compact and effectiveness than former methods. The requirements generation start high level performance simulation and then passes to an executable specification suitable for implementation using a hardware/software co-design tool. The reuse of pre-exiting components is supported, as well as synthesis of the system interface, but only after much work is done to program the hardware/software co-design tool.
The actual design flow described allows feedback among all design levels, e.g. from implementation up to requirements, throughout the process. In the future, it is necessary to development the advanced method of constructing system-on-chip based on embedded systems.

키워드

Embedded System, System-on-Chip, Design, Hardware/Software Co-design

I. 서 론

최근에 21세기 IT 분야의 주요 기술로 부각되고 있는 임베디드시스템과 이에 근간을 둔 시스템온칩 구현에 대한 연구가 활발하게 진행되고 있다.[1-3] 특히, 임베디드시스템의 하드웨어/소프트웨어 통합설계는 시스템온칩의 중요한 방법이다.[4-6]
또한, 최근의 분산환경과 모바일 환경에서의 임베디드시스템의 적용은 매우 중요하다.[7-9]
따라서, 본 논문에서는 하드웨어/소프트웨어 통합설계에 대한 내용과 이에 근간을 둔 시스템온칩 구현에 대한 방법에 대해 논의하였다.

II. 연구 배경

2-1. 연구 영역

본 논문에서 제안한 분야는 다음의 3가지로 요약할 수 있다.

□ 모델은 사용가능하지 않다.

간혹 모델은 자세한 저레벨에 있어서 사용 가능하거나 거의 상의 레벨에는 상세하게 되어 있지 않다.

□ 글로벌 분할이 주된 논의 대상이 아니다.

글로벌 분할의 대부분은 이미 준비되어 있으며, 많은 경우의 실제 설계는 다양한 이유로 이미 많은 부분이 존재하고 있다. 분할은 단지 소규모 설계에서만 수행되며 하드웨어는 가끔 이미 설계된 지능형 성질(Intellectual Property)의 형태로 수행되며 쉽게 사용가능하다. 좀 더 많은 유연성은 소프트웨어에만 존재한다.

□ 실제 설계에 있어서, 시스템 아키텍처는 부분적인 분할이 최대 이점으로 적용할 수 있도록 중요한 점을 도출하는 것이 필요하다.

2-2. 시스템온칩 설계

SoC(System-on-Chip)에 대한 설계 방법론은 보드 설계 또는 ASIC의 선형 확장이 아니며, SoC에 대한 아키텍처 논의 영역은 광범위하다.

간혹 이전의 하드웨어와 소프트웨어의 재사용을 팔요로 한다. 즉, 동일한 칩상에 많은 내부 결선을 갖고있지만, 동시에 많은 테스트를 할 수 는 없다는 것이다.

SoC에 있어서 하드웨어와 소프트웨어 사이에 기능을 이동함에 의하여 다양한 아키텍처를 극대화 할 수 있다. 이러한 모든 것들은 좀 더 복잡하고 규모가 큰 부분들을 개선할 수 있으며, 마지막으로 추상모델에 있어서 규모가 큰 서로 다름을 발견할 수 있다.

따라서, 다중의 영역을 상호 동작할 수 있도록 지원되어야 하며 다음과 같은 요소들이 이에 해당된다.

- 동기 데이터 흐름
- 제어 영역 반응 시스템
- 아나로그

아키텍처 뿐만 아니라 SoC의 타당성은 여러 가지 이유로 동일한 것은 아니며, 주요한 타당성으로써 원형(Prototype)을 사용할 수 은 없으며 그 이유는 다음과 같다.

- 만일, 시스템에 에러가 있다면, 경비는 매우 높아질 것이다.
- 하드웨어와 소프트웨어 부품 사이의 복잡한 상호동작은 칩상의 핀으로 부터 볼 수 있는 것은 아니다.

두번째로는 ASIC의 경우에 수행되는 타당성으로써는 다음과 같은 것들을 들 수 있다.

- 완벽한 크기
- CPU 제어 ASIC
- 독립적인 부분에 있어서 각각의 부품을 검증
- 2개의 대형 상태 공간의 상호동작

세 번째로서는 추상 레벨로 부텅의 상세함이나 밀어놓기 등을 할 수 있다는 것이다. 이러한 추상레벨과 더불어 규모가 큰 문제는 이미 작업을 하고 있는 상세한 모델에 대한 부품의 상위 레벨을 필요로 하는 설계이다.

III. 설계흐름에 기초를 둔 작업흐름

다음 그림 3-1은 잘 정돈된 설계와 설계흐름에 기초를 둔 작업 흐름 사이의 다름 점들을 보여준다.

그림 3-1에서 전자는 CAD 중심적이고 설계 행위의 이상적인 면을 보여주지만, 후자는 산업계에서 사용되는 설계흐름을 보여준다.

즉 작업의 자세한 공정 설명과 어떻게 수반되는 방법들에 관련된 것이다.[7] 작업흐름에서의 각각의 개별적인 단계는 활동적이다.

활동적인 것은 인간뿐만 아니라 프로그램 또는 상호동작적인 프로그램을 사용하여 수행된다. 작업흐름 패러다임은 특히 SoC에 대해 유효하다.

일찌기 SoC 설계의 국면은 주로 인간의 작업을 포함하고 있는 반면에 후자는 주된 프로그램으로 포함하는 단계이다.

향 후 제공되는 잘 정돈된 작업에 대한 실행 가능한 작업 흐름은 동적이다.[8] 본 논문에서는 Coware간의 동작적인 규격을 요구하는 시스템 레벨사이의 반복적인 정립을 할 수 있는 시스템 레벨 시뮬레이션에 대한 매뉴얼을 설명한다.

설계흐름에 기반을 둔 작업 흐름과 더불어 시스템 설계 팀은 고장난 설계 공정의 다양한 부분에 대한 같은 환경을 다룬다. 예를 들자면, 설계 재사용을 고려하여 단지 게이트 레벨 설명에 대한 블록의 재사용을 필요로 한다.

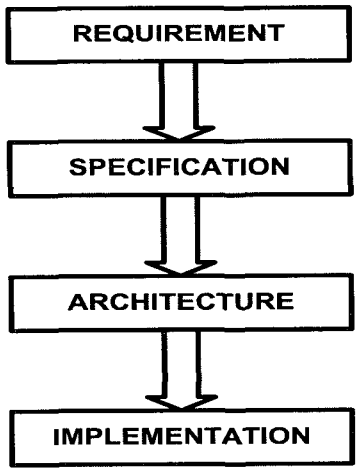
그러한 재사용 시스템의 시뮬레이션을 위하여, 이상적인 시뮬레이션 속도를 성취하기 위해 높은 레벨 설명의 생성을 해야 한다.

이것은 또 다른 문제를 야기하며, 이는 재사용된 게이트 레벨 설명에 반하여 새로운 높은 레벨의 타당성이다.

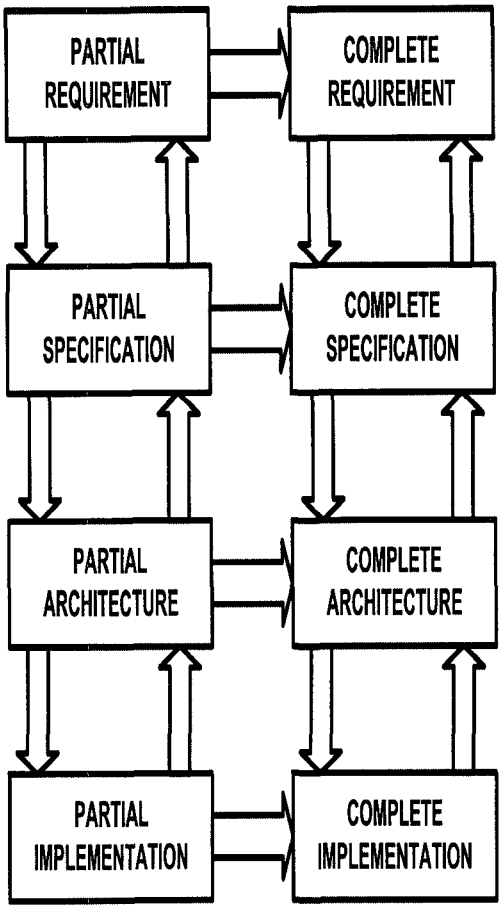
설계 흐름에 기반을 둔 작업 흐름 역시 재정렬 또는 규격으로 부터의 높은 레벨의 생성과 같은 설계흐름에 기반을 둔 재정렬과 같은 동일한 단계를 지원한다.

IV. CoWare를 수반하는 하드웨어/소프트웨어 통합 설계

모든 시스템의 부분으로써 분석되어 져야 할 노드 제품의 플래너(Planner)는 소프트웨어 프로그램이 가능한 CPU 코어(Core) 상에서 제일 좋게



(a) Refinement based design flow



(b)Workflow based design flow

그림 3-1. 설계 흐름의 비교
Fig. 3-1. The comparison of design flow

실행된다. 그리고 그 기능은 하드웨어 칩상에 가장 잘 구현된다.

이들 아키텍처 결정은 요구되는 칩상의 메모리 총 용량, 전체적인 다이(die) 크기, 시스템 수행, 전력 소비 등에 극적인 충격을 가질 것이다. 설계 과정에 늦게 시스템 분할의 지연은 아키텍처 결정이 게이트들과 더불어 만들어 진다.

CoWare 하드웨어/소프트웨어 통합설계 환경은 존재하고 있는 언어를 사용하여 하드웨어/소프트웨어 부품의 상호 규격을 허용한다.[6-7]

CoWare는 하드웨어와 소프트웨어 사이의 인터페이스의 명백한 규격을 제공하며, 하드웨어 인터페이스와 디바이스 드라이버를 생성함에 인해 하드웨어와 소프트웨어의 인터페이스를 정확하게 합성한다.

CoWare는 통신과정의 모델에 기초를 두며, 상위 레벨 하드웨어의 점진적인 섬세함을 지워한다.

모델은 재사용을 지원하며, 시스템 부품의 통신 동작 사이의 분할을 명백하게 함에 의한 하드웨어와 소프트웨어 캡슐화를 지원한다.

이러한 규격과 통신 과장에 기초를 둔 데이터 모델에 대한 방법이 기본이다. 설계는 기능과 통신 동작 사이의 엄격한 분할을 지원한다.

설계는 그들의 통신 동작의 절대적인 모델을 유지하는 동안 그들의 기능적인 동작을 묘사함에 의해 재사용을 만든다.

설계가 실제로 시스템에 사용되어 질 때, 규격 방법은 추상 통신 모델을 시스템 문맥속의 좀 더 적절한 상세한 동작으로의 정렬을 허용한다.

동일한 규격의 방법은 자기 자신 프로그램이 가능한 것으로 사용되어지며, 이들 모델은 독립적인 하드웨어/소프트웨어 통합설계 방법론으로서 사용된다.

일단, 글로벌 시스템 아키텍처가 정의되어지면, 합성 툴과 컴파일러는 모든 프로세서, 가속기와 메모리 부품을 구현할 수 있다.

CoWeare 설계 환경은 링크하는 인터페이스를 자동으로 생성함과 시스템 규격으로 구성되는 자기 자신 프로세서를 차단함에 dmp해 생성되는 인터페이스에 의해 존재하는 설계 기술을 제공한다.

CoWare 환경으로 구성하는 시스템 설계는 다음과 같은 단계가 있다.[9]

• 기능 규격 :

시스템은 채널을 통해 데이터를 교환하는 통신 프로세스의 수단에 의해 규정된다. 프로세스의 동작은 호스트 언어를 사용하여 삽입된다.

• 아키텍처 정의 :

최적의 할당 프로세스, 가속기, 메모리, 기능 규격으로의 바인딩.

이러한 상호 동작 할당된 바인딩 단계는 하드웨어/소프트웨어 분할을 포함하고 있다.

• 통신 선택 :
프로세서를 만들기 위한 필요한 소프트웨어와 하드웨어 자동 생성과 다른 환경과의 통신
- 통신 블록(CB)은 가속기 사이의 파이프라인과 동기를 제공한다.
- ARM 포로세서에 대한 하드웨어와 소프트웨어 사이의 통신은 좀 더 복잡하다.
- ARM 인터페이스는 어드레스 복호기, DMA 채널, 인터럽트와 I/O 포트를 포함한다.
- ARM 내에 소프트웨어 드라이버는 반드시 합성되어져야 되며, ARM 상에서 구동되는 프로세스를 링크시킬 수 가 있어야 한다.
• 부품 구현
가속기 프로세서, 인터페이스 하드웨어와 소프트웨어, 메모리, 프로세서 코어상에서 구동되는 소프트웨어 등과 같은 시스템안의 모든 부품, 그리고 디버깅 블록은 설계 환경을 사용하여 구현된다.

CoWrae는 ARM C 컴파일러와 상업적인 같은 다른 부품 컴파일러를 설계 환경으로 임베디드한다.

V. 결론

본 논문에서는 최근에 21세기 차세대 IT 기술의 주요 기술중에 하나인 임베디드시스템에 기반을 둔 시스템온칩 구성에 대한 방법에 대해 논의하였다.
시스템 레벨 설계 툴의 산업계 채택에서의 주요한 단계는 혼합 절대 레벨을 가로지르는 IP 블록의 정당성과 모델 생성의 개선이다.
특별한 블록에 대한 가장 표현 가능한 언어 형태에서의 존재하고 있는 블록이 설명되어지므로, 많은 형식(Format)과 코딩 스타일이 사용되어진다. 이러한 이종간의 실현 스터일은 어려운 절대 부품의 라이브러리의 생성 과정을 만든다.
존재하고 있는 IP 블록의 시스템 레벨 설명의 손으로 하는 코딩은 가격이 비싸고 실수하기 쉬운 점이 있다.
필요한 것은 정당성의 방법을 개선시키고, 자동적으로 생성하고, 시스템 레벨 프레임 작업안에 존재하는 블록을 캡슐화 하는 것이다. 존재하고 있는 실현 레벨 모델에 반한 손으로 생성된 정대색인에 대한 새로운 정당성을 위한 방법과 좀 더 좋은 툴은 라이브러리 생성 과정에 대한 속도를 필요로 한다.
이 경우에 시스템 레벨 시뮬레이션은 결정적으로 중요하다. 향 후 손으로 생성하는 과정은 실현 시뮬레이션 의 동작의 관찰로부터 시스템 레벨 모델을 생성하는 자동 툴에 의해 수행되어질 것이다.
동시에 실행 시간에 레벨 정합을 수행 할 수 있는 랩퍼(Wrapper) 속에 실현 모델을 캡슐화 하는 좀 더 효과적인 방법은 손으로 생성하는 시스템 모델과 자동으로 생성하는 시스템 모델 사이의 간격을 설계에 기반을 둔 시스템을 채택하는 속도와 마찬가지로 채울 수 있다.
제안한 방법은 기존의 방법에 비해 효과적이며, 규칙성과 신뢰성이 개선되었다. 하지만, 향 후 좀 더 진보된 방법의 연구를 필요로 한다.

참고문헌

[1] Edward A. Lee, "What's Ahead for Embedded Software?," IEEE Computer, September 2000, pp.18-26

[2] Wayne Wolf, "What Is Embedded Computing?", IEEE Computer, pp 136-137, January, 2002.

[3] Steve Furber, ARM System-on-chip Architecture, Adison-Wesley, 2000.

[4] G. De Micheli and M. Sami, hardware/Software Co-Design. Kluwer Academic Publishers, Norwell, MA, 1996.a

[5] F. Balarin, M. Chiodo, P. Giusto, H Hsieh, A. Jurecska, L. Lavagno, C. Passerone, A. Sangiovanni-Vicentelli, E. Sentovich, K. Suzuki and B. Tabbara, hardware-Software Co-Design of Embedded Systems - The Polis Approach, Kluwer Academic Publishers, MA, 1977.

[6] R. K. Gupta, Co-Synthesis of hardware and Software for Digital Embedded Systems, Kluwer Academic Publishers, Boston, MA, 1995.

[7] Sybase, Synchronization Technologies for Mobile and Embedded Computing, A White paper

[8] M. Kamath and K. Ramamrithan, Modeling, correctness & systems issues in supporting advanced database applications using workflow management systems, Umass Cs Technical report 96-07, Jan. 1996, Available from

[9] H. Lavana, A. Khetawat, F. Brglez, and K. Kozminski, Executable workflow : A paradigm for collaborative design on the internet, Proceedings of the 34th Design Automation Conference, pp.553-558, June 1997,