行政院國家科學委員會補助專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※※※
※　　　　　　　　　　　　　　　　　　　　　　　※
※　　　對以智財單元為基系統晶片設計之　　　　※
※　　　　　驗證與測試技術開發研究　　　　　　※
※　子計畫一:與組織探索階段互動之系統階層驗證技術　※
※　　　　　　　　　　　　　　　　　　　　　　　※
※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：整合型計畫

計畫編號：NSC 90－2215－E－009－083－

執行期間：　90 年　8 月　1 日至　91 年　7 月　31 日

計畫主持人：董蘭榮

共同主持人：沈文仁

計畫參與人員：　陳宥霖
　　　　　　　　蔡逸凡
　　　　　　　　張智凱
　　　　　　　　簡偉勝
　　　　　　　　黃旭榕

執行單位：國立交通大學電機與控制工程學系

中　華　民　國　91 年　　10 月　　29 日

行政院國家科學委員會專題研究計畫成果報告

對以智財單元為基系統晶片設計之驗證與測試技術開發研究
子計畫一:與組織探索階段互動之系統階層驗證技術
System-Level Verification Interacting with Architecture Exploration

計畫編號：NSC 90－2215－E－009－083－
執行期限：90 年 8 月 1 日至 91 年 7 月 31 日

主持人：董蘭榮　　國立交通大學電機與控制工程學系
共同主持人：沈文仁　　國立交通大學電子研究所
計畫參與人員：
陳宥霖　　國立交通大學電機與控制工程學系
蔡逸凡　　國立交通大學電機與控制工程學系
張智凱　　國立交通大學電機與控制工程學系
簡偉勝　　國立交通大學電機與控制工程學系
黃旭榕　　國立交通大學電機與控制工程學系
Email: lennon@cn.nctu.edu.tw

一、中文摘要

系統晶片設計涵蓋很廣的設計空間。設計者通常需要考量許多可能的系統組織包括選擇演算法則、挑選組織元件、建構候選組織。設計如此的複雜系統誠屬不易，而要設計出能完全符合要求、正確無誤的系統更為困難。設計上的失誤必須要儘早排除，否則在後續階段才發現的失誤將造成耗費耗時的再設計周期。因此，設計者必須面對兩項課題，其一是實現設計程序本身、另一是建立正確的設計結果。其中，設計的正確性將為本計劃的主軸。此子計劃第一年之工作為提出組織元素的成本模型、發展成本評估核心公式、定義效能模型資料結構、發展基礎效能模型。

關鍵詞：效能模型、系統階層驗證、系統晶片

Abstract

The System-On-Chip (SOC) design encompasses a large design space. Typically, the designer explores the possible architectures, selecting algorithms, choosing architectural elements, and constructing candidate architectures. Designing such a complex system is hard; designing such a system which will work correctly is even harder. Design errors should be removed as early as possible; otherwise, errors detected at the later stages will result a costly, time-consuming redesign cycles. Thus, the designer should face two distinct tasks in SOC design; carrying out design process itself and establishing the correctness of a design. Design correctness is the main theme of this project. In the first year, the tasks of this project are: proposing cost models of architecture elements, developing cost estimation engine, defining the data structure of performance modeling, developing the fundamental performance models

Keywords: performance modeling,
system-level verification,
system-on-chip

二、緣由與目的

Intellectual Property (IP) reuse is becoming essential in system-on-a-chip design. It helps designers meet the challenges of increasingly complex applications, highly integrated systems and shorter design cycles. However, effective IP reuse poses several problems, notably in the integration and validation of the foreign IP within the target system. State-of-the-art design methodologies do not address these issues, but rather rely on IP vendors to assist designers with the custom integration of their IP cores. Recently, several EDA companies teamed up with IP vendors to provide some solutions to these problems [4]. The results are interesting, but still rather IP-specific. Our approach is meant to put IP integration technology at the fingertips of the designers themselves, thus reducing the gap between IP vendors and system designers.

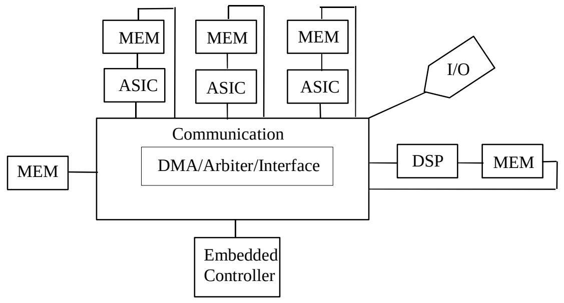

Figure 1 A typical SoC architecture

We describe a system-on-a-chip architecture in terms of IP cores. Figure 1 depicts a typical embedded architecture composed of: a communication cloud connecting all system components, an embedded controller, a DSP, ASIC blocks, memory components, I/O, and a DMA. Each component in the system can be an in-house or third-party Intellectual Property core. Each IP core can be accompanied by its own different set of verification and modeling tools (e.g., an instruction set simulator (ISS), an HDL model, a C model, or an emulator board), of varying speed, accuracy, and level of abstraction. We allow the designer to treat IP cores of heterogeneous provenance as black boxes characterized by their I/O interfaces; if the unified design environment is VHDL-based, for example, the IP cores look like VHDL entities. The reuse considered in this paper is at the architecture level. Other levels of reuse (e.g., core level) are described in [3].

We use our own front-end system level verification environment, developed last year [9], to identify the appropriate architecture, and IP cores for the target application. Once the architecture and the appropriate IP cores are identified, we use our co-simulation environment to generate a system co-simulation test-bed by producing all the required interfaces and wrappings between the different IP cores. Our interfacing approach deals with the I/O interfaces. It is independent of the communication protocols adopted between IP cores[1]. We do not intend to synthesize the communication device interfaces between the hardware and software [6], but rather to focus on synthesizing the means to allow plug-and-play IP integration and validation. In order to accomplish this, we have developed a multi-layered wrapping scheme for IP cores. The innermost layer is an Application-Programming-Interface (API) layer. It allows access to core resources without the necessity for detailed knowledge of the core itself, thus simultaneously simplifying integration and protecting the owner of the IP. The second wrapper layer is the communication layer; it handles the exchange of required information between the individual IP core and the overall simulation environment, allowing for heterogeneous cosimulation. The outermost layer is the synchronization layer. It synchronizes the interactions, timing, and collection of results between the IP core and the system simulation environment.

Our approach extends the work described in [7]. It allows for integration and validation of IP cores of different forms (i.e., not limited to C programs), and proposes an optimized synchronization scheme that can run in both event-driven and lock-step modes (based on the different phases of interactions between the IP core and the rest of the system). The API layer adds to the generality and flexibility of the proposed wrapping scheme, and protects the IP core.

三、結果與討論

The system-level co-simulation technology described in this report has been validated successfully on several applications, and at different design stages of several telecommunications and multimedia products. We have applied our wrapping technology on a H.263 CODEC, an ADSL application, a High-Density Central Site Modem (HDSM), and an MPEG-4 decoder. Within all these applications, the embedded architecture looks much like the architecture of Figure 1. What varies from one application to the other is the number of ASICs, the type of DSP, the version of the embedded controller, and the communication interconnect and protocols. In all cases, we had to deal with in-house and third-party IP cores. Some of the in-house IP cores were still under development, and our wrapping scheme helped plug them into the system prior to, or upon, their completion. It also helped us, when we started developing the wrapping scheme, to work first on in-house IP cores. This gave us the opportunity to develop the API functions for the cores ourselves, after collaboration with the (in-house) IP core designers.

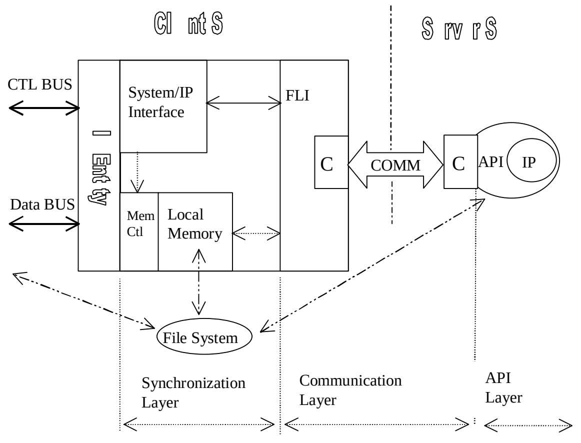

Figure 2 HW/SW co-verification model

Based on the model described earlier, most of the wrapped IP cores ended up looking like the model of Figure 2. The IP core is visible to the system simulation environment through its entity. The entity is composed of a control bus and a data bus. If a VHDL model of the IP core is used, no wrapping is required. Otherwise, the IP entity must interface to the three layers: Synchronization, Communication, and API. Those layers, as described earlier, allow the system simulation environment to break seamlessly into heterogeneous environments and run in a co-simulation mode. The heterogeneous co-simulation is based on a Client/Server model. Thus, every IP has both a client and a server side. The client is the IP instantiation model within the system simulation environment. It calls upon a server IP model (that executes in a foreign environment) whenever an execution of the IP is required.

The following describes a scenario for heterogeneous cosimulation (master/slave model) using our wrapping technology:

1) *Initialization phase:* The master drives the control bus with address and data. The interface technology (synchronization layer) translates this information into a register index and a programming value. This information is used to program the IP core.
2) *Execution phase:* The master drives the control bus to start the slave. The interface model then dumps the program/data memory spaces into program/data files. It also issues an execution order to the IP core. The IP core starts processing based on the program/data files.
3）*Termination phase:* Upon completion of execution, the IP core signals the completion of the task to the IP interface. The interface model then updates the data memory space (loading from the result files), and issues a time-accurate interrupt request to the master.

Our hardware/software co-simulation technology is verified to be an efficient and fast way to run heterogeneous cosimulation, and quickly incorporate foreign IP cores into embedded systems. Substantial productivity gains and design-time reductions have resulted from its use. In a high-density central site modem (HDCSM) application, we have been able to wrap several in-house IP cores and set up a heterogeneous cosimulation system platform in less than two weeks. This includes the development of the API functions for the IP cores, and the interaction models with the system. We have developed and tested the heterogeneous cosimulation IP models with little knowledge about the cores themselves. Applying different speedup techniques, we have been able to accelerate the heterogeneous system cosimulation substantially (often by three orders of magnitude). For example, we have been able to speed up the system simulation of the HDCSM application from three instructions per second (RTL) to 1600 instructions per second. Thus far, the IP-based synthesis technology is only partially automated. The communication layer and the client/server interfaces are generated automatically. However, the designer's assistance is needed to define the interaction model, and the API functions.

四、成果自評

本計畫第二年已建立軟硬體共模擬環境，可有助於組織探索階段完成軟硬體互動之驗證工作。本計畫之研究成果已發表下列兩篇國際會議論文與一篇國內會議論文：

1. Ting-Hsun Wei, Shiuh-Rong Huang, and Lan-Rong Dung, 2002, "An Automated IP Synthesizer for Limited-Resource DWT Processors," ICASSP 2002
2. Shiuh-Rong Huang and Lan-Rong Dung, 2002, "A MAC-level Synthesis of Resource-Constrained DWT SIP," VLSI/CAD Symposium 2002
3. Shiuh-Rong Huang and Lan-Rong Dung, 2002, "VLSI Implementation for MAC-Level DWT Architecture," ISVLSI 2002

另外，部分研究成果正投稿于 IEEE 期刊。

經由本計畫之執行已培養四名碩士畢業生。該四名碩士畢業生目前服務於系統晶片相關之高科技企業。

五、參考文獻

[1] Lan-Rong Dung, Vijay K. Madisetti, and John W. Hines, "Model-Based Architectural Design and Verification of Embedded DSP Systems," 1996 VLSI Signal Processing Workshop, October 1996

[2] Lan-Rong Dung, Mohamed Ben-Romdhane and Marius Vassiliou, "IP-Based Architecture Exploration," DesignCon99, February, 1999

[3] Mohamed Ben-Romdhane, Marius Vassiliou and Lan-Rong Dung, "Rapid Prototyping of Multimedia Chip Sets", ICASSP-99, March,1999

[4] Richard Goering, "New Tools will Force Embedded Designers to Link Hardware/Software Efforts – Codesign turns workplace on its head," EETimes, Issue:988, January 12, 1998

[5] J.K. Adams and D.E. Thomas, "The Design of Mixed Hardware/Software Systems," Proc. Of 33rd DAC, 1996, pp.515-520

[6] Lan-Rong Dung and Vijay K. Madisetti, 1996, Fall, "Conceptual Prototyping of Scalable Embedded DSP Systems," IEEE Design and Test of Computers, pp. 54-65

[7] C. A. Valderrama, Fran□ois Na□abal, Pierre Paulin, Ahmed Amine Jerraya, "Automatic Generation of Interfaces for Distributed C-VHDL cosimulation of Embedded Systems: an Industrial Experience," 7th IEEE International Workshop on Rapid Prototyping, June 1996