ELSEVIER

Journal of Systems Architecture 48 (2002) 49–57

JOURNAL OF SYSTEMS ARCHITECTURE

www.elsevier.com/locate/sysarc

Interface synthesis between software chip model and target board

Seungjong Lee [a,*], Ando Ki [b], In-Cheol Park [a], Chong-Min Kyung [a]

[a] *Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology, 305-701 Daejon, South Korea*

[b] *Dynalith Systems Co. Ltd., Mirae Asset Venture Tower, 996-17 Daechi-Dong, Kangnam-Gu, 135-280 Seoul, South Korea*

Abstract

This paper reports on the synthesis of interface between software chip model and target board in a behavioral emulation system called in-system algorithm verification engine (iSAVE). iSAVE performs in-system verification of the behavioral description of a chip in such high-level languages as C in the context of its application board at the early chip design stage. The interface between the target chip and the target board is implemented as two parts; software part running on a microprocessor in a multi-thread fashion and hardware part mapped into field programmable gate array logic. The proposed idea is validated by successfully demonstrating the behavioral emulation of MP3 decoder chip, i.e., running the MP3 decoding algorithm written in C along with the real MP3 player board minus the MP3 decoder chip itself through the proposed interface scheme.

Keywords: Emulation; In-system verification; Real-time operating system; Interrupt controller; MP3

1. Introduction

In designing a chip as a part of complex information processing systems comprising multiple chips and other system modules, in-system functional verification of the chip model, i.e., functional verification of the design data of each chip in the context of its target application system is very crucial. In-system verification is especially important in multimedia applications where the functional correctness of the chip being designed can be conveniently checked by directly connecting the chip model to the real hardware environment in the application system such as camera, video screen, speaker, and microphone. On the contrary, relying only on traditional software simulation using artificial test/verification vectors can leave functional bugs undetected until the actual silicon chip becomes available, due to the difficulty of covering all possible corner cases.

Traditional hardware emulation, i.e., structural emulation in register-transfer level (RTL) or gate-level where the gate-level model is mapped into field programmable gate array (FPGA), has been widely used in recent years as an integral part of the verification flow of various ASICs and

* Corresponding author.

E-mail addresses: sjlee@vslab.kaist.ac.kr (S. Lee), adki@dynalith.com (A. Ki), icpark@ee.kaist.ac.kr (I.-C. Park), kyung@ee.kaist.ac.kr (C.-M. Kyung).

1383-7621/02/$ - see front matter
PII: S1383-7621(02)00065-6

microprocessors. Despite the speed and relatively wide functional test coverage, the structural emulation, i.e., emulation at RTL or gate-level has a limitation in reducing the time-to-market, as the RTL or gate-level model can only be obtained at the end of the design stage, putting the emulation at the end of the verification flow. Therefore, if some architectural errors are found during the emulation, it generally takes a very long time until the emulation runs again with identified bugs fixed.

Many VLSI chip designs, especially in signal processing area, are started with the algorithm model written in high-level programming languages such as C, C++ and SystemC [1]. After the C model is verified with simulation, the designer translates it into RTL model written as hardware description language (HDL).

Fig. 1 shows the trend of IC design verification. HDL-based simulation is not fast enough for complex chip. System-level, or behavioral simulation allows not only faster model execution but also implementation in mixed-HW/SW system by adopting high level languages (HLLs) such as C/C++. On the other hand, hardware emulation, or structural emulation has been used for the past 15 years as a design sign-off, as it allows wider set of test vectors to be applied. Similarly, in-system verification is nearly mandatory in verifying the system-level models. Driving forces behind this trend towards in-system verification, or behavioral emulation are ever-growing chip complexity and short design turn-around requirement.

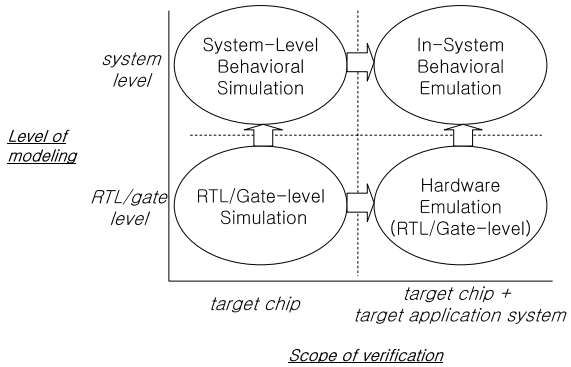

Fig. 1. Trend of IC verification is from RTL/gate-level to system-level in the level of modeling (vertical) axis, and from target-chip only to in-system, i.e., target chip + target application system, in the scope of verification (horizontal) axis.

We present in this paper a behavioral emulation system, which lets designers perform in-system verification, i.e., verification of chip model written in HLLs such as C, C++ or SystemC in the context of the target application system.

1.1. System overview

In-system algorithm verification engine (iSAVE) is a system that enables an algorithm or a behavioral chip model of a chip written in C to be verified in the context of its target application system [2]. Once the behavioral C model is verified, it can be used as a golden reference model with which the lower-level design translation/implementation is to be compared. Furthermore, C model which does not have such constructs as pointer and file I/O can be translated to HDL for synthesis. This system was also successfully demonstrated to verify the functional correctness of an instruction-level model of CISC microprocessors with commercial applications in real environment [3], where an instruction set simulator as the behavioral model of the microprocessor runs along with external bus model implemented in FPGA which, in turn, communicates with the outer world, i.e., its motherboard.

Fig. 2 shows the iSAVE-based emulation system consisting of two parts: host computer responsible for the generation of compiled code for the behavioral model, and the iSAVE system which, in turn, consists of processing engine (PE) and pin signal generator (PSG). The first task of the chip designer is to write the algorithm or behavioral model of the chip, which is translated into a code directly interacting with PSG after the designer specifies the software variables responsible for communicating with the external hardware [2]. The generated code is compiled in the host computer and is downloaded to PE where the behavior of the target chip, i.e., chip to be designed, is verified.

Another task required by the designer is to describe the chip interface model. The designer ini-

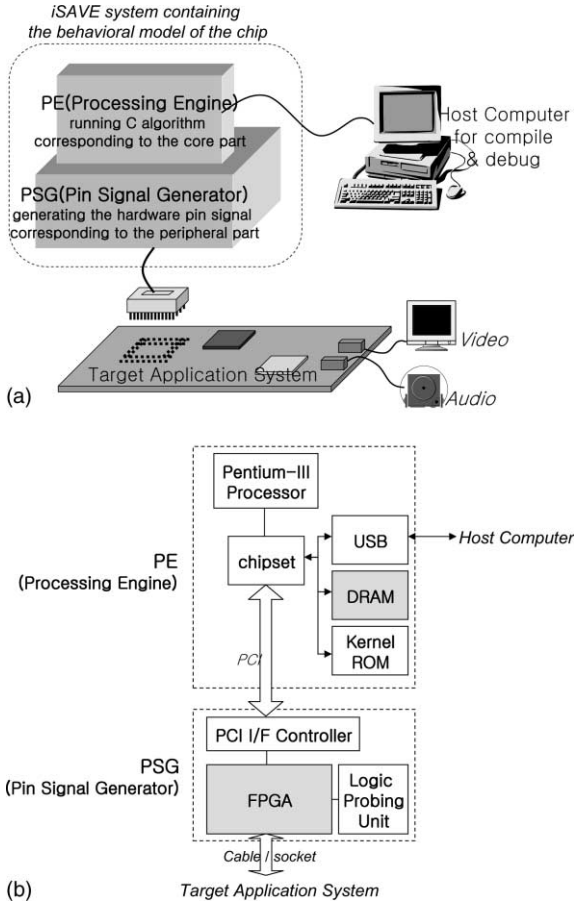

Fig. 2. (a) Overall structure of the proposed behavioral emulation system which consists of iSAVE system, host computer and target application system, and (b) details of hardware architecture of the iSAVE.

tially describes the chip interface as a finite state machine (FSM), which is translated to HDL code. Finally, the interface models are mapped into the PSG implemented in FPGA which generates/receives real hardware signals to/from the target application system via cable and/or the same socket as will be used by the target chip when it is fabricated and packaged.

The iSAVE system was implemented with Pentium-III processor and its chipset. Custom peripherals, such as kernel ROM, LCD driver and specially designed interrupt controller, as shown in Fig. 2(b), are implemented with an FPGA chip. The PSG consists of three parts: (i) PCI interface for communicating with PE, (ii) FPGA chip for implementing the relevant I/O interface protocol and generating pin signals compatible with the socket pins in the target application system, and (iii) logic probing unit for monitoring various I/O interface signals.

The bandwidth between the algorithm model and the interface model often becomes the performance bottleneck of the behavioral emulator. In iSAVE, the algorithm model and the interface model communicate with each other using a predefined set of commands and data packet. Compared to signal-level communication, it reduces the communication bandwidth and, therefore, achieves higher emulation speed [3].

The behavioral emulation has the following benefits compared to the structural emulation:

- In structural emulation, target application system often needs to be slowed down to level with the slow speed of the emulation system itself. In iSAVE, clock is fed to the interface model only. Consequently, iSAVE can achieve much faster external interface than the structural emulation where all parts must operate with a clock. [1]
- Running the algorithm in software gives the designer more flexibility in debugging. Furthermore, time to restart emulation after bug fixing is significantly less compared with the structural emulation, as the algorithm or behavioral model is easier to diagnose and debug than the structural or gate-level model.

One of the most crucial issues in the design of behavioral emulation is how to interconnect the algorithm part of the chip model executed sequentially and the interface part of the chip model executed in parallel.

[1] In structural emulation (or hardware emulation), the clock speed of the target system, the FPGA-mapped internal logic and the interface circuit in between must be identical, which is below 1 MHz for the gate count of about 500 000 gates or above, while in behavioral emulation, due to high-speed functional execution, the clock speed of the interface easily reaches 50 MHz or above.

1.2. Hardware/software interface synthesis

There has been a need for rapid prototyping systems for many years [9]. Most of them were proposed as HW/SW codesign tools to profile performance or to verify a synthesized system. System description is partitioned into two parts: software and hardware.

Behavioral emulation is mostly regarded as an extension of structural emulator [5], and contains a number of processors executing RTL models of a part of the system and FPGA's executing gate-level models of the rest. Two models are connected with simple logic attached to interface software and hardware.

There are some research results reported in the interface synthesis between hardware and software [6]. Chou et al. [7] describes the interface synthesis between hardware and software in the context of microcontroller-based design, where I/O ports are automatically assigned and the relevant interface logic and software module as required for the interface as generated from templates. In [7], the software module always controls the hardware block without allowing the hardware giving commands to the software, i.e., the hardware always runs as a slave to the software. In case of actual chip interfaces, however, the interface is bi-directional, i.e., it is possible for the interface (hardware) to request data from the algorithm (software). Moreover, Chou et al's result is not generally applicable as the interface design solution since it does not consider the synchronization during HW/SW partitioning as originally specified in the system model. In our case, on the other hand, event handler is available which is responsible for synchronization between the chip model and the target application system.

Basu et al. [8] included event handler in the design of embedded systems, where concurrently executed blocks are modeled as interrupt service routines (ISRs). ISRs are generated based on the control data flow graph while considering stack overflow and return position of ISR. These considerations made the system design process a very complex one.

In iSAVE, the event handler is not a part of the chip model provided by the user, but a module automatically generated to manage the bandwidth and latency of interfaces of the model. ISRs as the event handler in iSAVE has simpler structure than one in [8] because it is handled in the dedicated operating system (OS).

While most algorithm models assume that every input is immediately available, time to access data from external devices in the application system cannot actually be ignored in the real system. These inputs in the algorithm part of the model are replaced with the code that handles the blocking input. Blocking input denotes input signal which halts the execution of the algorithm model until it is handled. If the bandwidth of external interface is lower than the internal processing speed, the algorithm model must wait until all valid inputs are ready. Moreover, in case of multiple independent interfaces, the fraction of waiting time among total execution time can be significant, although it is not observable in real VLSI systems because this overhead, i.e., waiting time, is hidden due to the concurrent execution of the interface hardware. To simulate such concurrent event, it is necessary to make interface processing invisible to the execution of the algorithm.

In this paper, we propose a method to manage interface models of VLSI system to reduce the overhead of handling blocking inputs. In Section 2, we propose a method to manage the chip interface model especially in the context of its coworking with the algorithm part of the whole chip model. In Section 3, an MP3 decoder chip modeled in C was successfully verified in its target application system using the proposed behavioral emulation system.

2. Interface implementation

2.1. Interface modeling

The algorithm model of a chip being designed is generally written without considering the interface with the target application system of the chip. It is then necessary that designer describes the operations of the interface as FSM. The interface model, as a bridge between the algorithm and the application system, accesses variables of the algorithm

as well as generates signals on the pins for direct connection to the target application system.

Because of strong connection between the chip model and the interface, the interface model has been implemented in software [11]. When the algorithm sends a command to the interface using a predefined function, the interface model interprets it and generates appropriate pin signals. In case of behavioral emulation, time consumed to communicate between the algorithm and the interface model is longer than in cosimulation. We need to consider the bandwidth and latency issues in implementing the interface model.

- *Interface bandwidth:* Outgoing pin signals sent from the algorithm is translated into real pin signals on the chip boundary with the simple interface model. These signals pass through the bus where the processor and the interface model are connected. As the operating frequency of the bus is typically about 10 times slower than that of typical microprocessors, time consumed for communicating between the chip model and the simple interface model becomes the bottleneck of the behavioral emulation. Hence, it is desirable to reduce the traffic due to this communication.
- *Interface latency:* With the bus cycle where the application system sends a signal and waits for its response from the emulated chip, the response time (T_R) can be calculated as Eq. (1):

$$T_R = 2 \times T_i + T_G \quad (1)$$

where T_i denotes the time of flight between the pin and the interface model, and T_G denotes the time to generate signal in the interface model. Some interface cycles strictly require the chip to respond within the specified time. In iSAVE, the interface model is implemented in hardware rather than in software to better deal with the latency issue.

The emulation system consists of three parts—algorithm, interface handler and interface model as shown in Fig. 3. The algorithm which runs on the PE is augmented with functions which access the software buffer called S-buffer as shown in Fig. 3 [2]. The interface model running on the PSG generates the pin signals of the emulated chip.

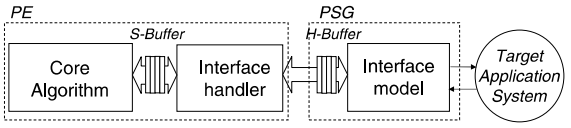

Fig. 3. Logical model showing the interconnection between the core algorithm running in PE and the interface model running in PSG through the interface handler running in PE. A software buffer called S-buffer is embedded in the PE along with the core algorithm and the interface handler, while a hardware buffer called H-buffer is embedded in the PSG.

Fig. 3 shows that the interface handler which runs concurrently with algorithm on PE acts as a bridge between the algorithm and the interface model. As the signal rate of the interface model is generally different from that of the algorithm, it is necessary to include a hardware buffer called H-buffer in FPGA. Therefore, two buffers are included at the interface of iSAVE. The role of interface handler is to manage data stream between the two such that the required data throughput is sustained within the allowable latency.

Fig. 4 shows the block diagram of the PSG. It contains a number of interface models each of which emulates a group of I/O signals of the emulated chip. The data generated by the interface model is delivered, via system bus and H-buffer, to the algorithm running on PE via system bus and the H-buffer.

2.2. Buffer assignment

Hardware buffer called H-buffer as implemented in FPGA is responsible for connecting the

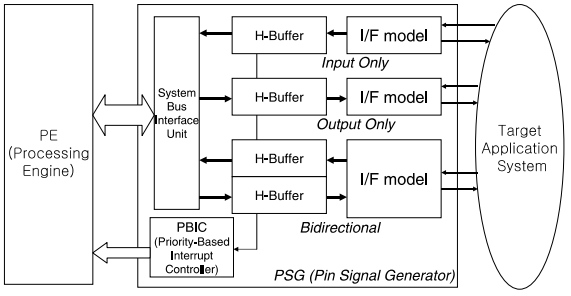

Fig. 4. Block diagram of the PSG consisting of a set of hardware buffer (H-buffer), with various interface models in the target application system side, and system I/F unit with PBIC in the PE side. Details of PBIC is shown in Fig. 5.

interface handler and the interface model. The other buffer, S-buffer, is implemented as a software array and responsible for connecting the algorithm and the interface handler.

H-buffer is a speed buffer between the target application system and the system bus where the interface model is attached, while S-buffer is a speed buffer between the algorithm and the interface model. The interface handler fetches the data from the H-buffer and stores them in the S-buffer, and then the algorithm accesses the S-buffer with the relevant API functions. Operation of the interface handler is described in Section 2.4.

Size of the S-buffer should be larger than the amount of data needed to be fetched by the algorithm at once. While the S-buffer is flexible to change its size, the H-buffer is limited in size due to the hardware resource constraint. The size of the H-buffer also affects the execution time of the algorithm. For example, if the H-buffer is too small, it results in excessive calls to interface handler to fill the S-buffer and wastes large computing power for thread switching among multiple threads. On the other hand, if the H-buffer is too large, it takes a long time to fill up and empty the buffer, resulting in slow context switching in case of multiple interacting I/O devices with separate buffers for each. The size of the buffer can be calculated from the bandwidths of the interface model and the system bus [10].

2.3. Priority-based interrupt controller (PBIC)

The interrupt controller included in the PSG decides which hardware buffer needs to be serviced with priority. The priority-based scheme in the PBIC, by controlling the latency of data transfer between hardware and software, helps reduce the time to transfer data especially for the high-priority service request.

The priority for the hardware buffer is determined by Eq. (2).

$$P_{\text{buf}} = P_{\text{base}} + \alpha S_{\text{buf}} + \beta T_{\text{wait}} \tag{2}$$

where α, β are some constants.

The base priority, P_{base}, is defined for each H-buffer. Generally, higher base priority is assigned to a hardware buffer attached to an interface which requires faster I/O response time.

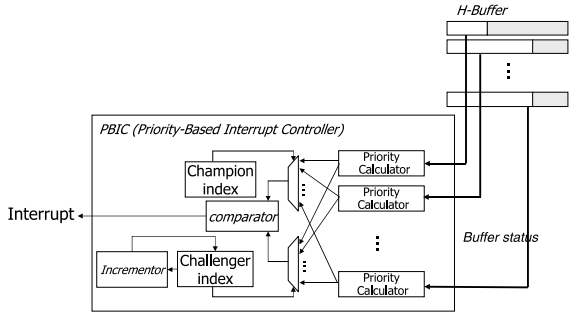

Fig. 5. Detailed block diagram of the PBIC shown at the lower left corner of Fig. 4.

S_{buf}, which denotes the filled portion of the H-buffer, is included to prevent the 'buffer full' or 'buffer overflow' condition. Finally, T_{wait}, which denotes the time H-buffer spent in waiting, is included to prevent 'excessive starvation'.

Fig. 5 shows the interrupt controller implemented in the PSG. The priority for each H-buffer is calculated in the *priority calculator* using the Eq. (2) with S_{buf} and T_{wait} coming from the H-buffer. A small counter is embedded in each priority calculator to measure the waiting time, T_{wait}, for each buffer.

The index of H-buffer which has the highest priority so far is stored as the *champion index*. Current priority of each H-buffer, as scanned by the challenger index being incremented by one, is compared with that of champion H-buffer, i.e., H-buffer designated by the champion index, to update the champion index if necessary.

2.4. Managing interface handler with thread

The interface handler runs independent of the algorithm model of the target chip in PE, and therefore, can simultaneously access the same variable with the algorithm in PE. This property allows implementing the interface handler as thread. The algorithm and interface handler for each chip interface are treated as separate thread executed concurrently. Unlike traditional multi-thread program synchronization or precedence relation among threads is not necessary in the behavioral emulator.

The PE normally executes the chip model, and switches to the associated handler routine when

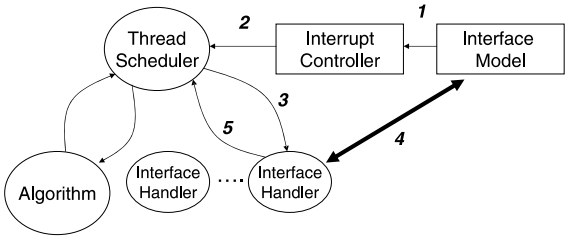

Fig. 6. Sequence of demand scheduling using thread scheduler.

the handler needs to access the H-buffer, which is called demand scheduling.

The sequence of demand scheduling is explained below according to the sequence diagram in Fig. 6. Algorithm runs with fetching data in the S-buffer: (1) When enough data from the interface model are stored in the H-buffer, the H-buffer generates the event and marks the event register at the interrupt controller. (2) If at least one event occurs, the event controller alarms the processor with an interrupt signal and then the thread scheduler is invoked. The thread scheduler identifies the hardware interface which has generated the event. (3) The thread scheduler invokes the assigned handler as a new thread which runs concurrently with the algorithm thread. (4) The called interface handler then moves data in the H-buffer to the S-buffer. (5) After data movement, the interface handler notifies its completion to the thread scheduler.

3. Experimental results

3.1. Implementation

iSAVE system executes the algorithm model and FPGA chip which executes the interface model. The kernel controls the communication with host computer and thread-level switching. The development system is built with GCC as C compiler. Each handler is programmed as an independent C function but it is possible to access the S-buffer in the algorithm. The kernel initializes each thread with the memory addresses of each handler routine before it executes the algorithm. The H-buffer is implemented with the embedded memory block available in the FPGA.

3.2. Application: MP3 decoder chip

An MP3 (MPEG-1 audio layer-3) decoder chip, MAS3507D is used as a vehicle to demonstrate the behavioral emulation based on the proposed interface synthesis. The chip has three serial interfaces and extra pins such as reset and power-pins [12]. The MP3 ISO model available in the public domain [13] is used as the algorithm model of the chip.

We implemented the interface model for each interface as the following:

Serial output (SO): The handler moves the data in the S-buffer to the H-buffer, and then the interface model serializes the 16-bit data in the hardware buffer. The hardware buffer must not starve, i.e., it needs to be filled instantaneously to generate the sound uninterrupted.

Serial input (SI): When the S-buffer does not have enough data, the chip must assert the DEMAND pin before it reads the data stream. Handler controls the DEMAND pin along with loading the data from the H-buffer. The interface model aligns the pin signals in parallel and stores them in the H-buffer.

I2C: The application system initializes the internal registers in the chip via I2C interface. In contrast to other two interfaces described in FSM, its asynchronous operation can be better described in HDL code. Another special consideration is that the state machine to interpret the cycles is located in the handler while the interface model generates 8-bit serial data.

iSAVE system automatically connects the file related to I2SI and I2SO to corresponding interface models and connects the variable which control a MP3 data stream to the DEMAND pin.

Table 1 shows the amount of time consumed in the MP3 algorithm decoding, SI and SO, respectively for data transfer for a 60-s MP3 data stream.

It was observed that although the processor consumes only about 30% of its computing power for MP3 decoding, correct sound cannot be produced without concurrent interface management, since the H-buffer assigned to SO starves during MP3 decoding and filling of the H-buffer assigned to SI. By filling the H-buffer assigned to SO with highest priority among the chip interfaces,

Table 1
Execution time for algorithm execution (MP3 decoding), SI and SO for a 60-s MP3 stream data

	Without handler (s)	With handler (s)
Algorithm	26	26
SI	35	3.8
SO	69	12.4
Total playing time	130	60 (idle over 17.8 s)

uninterrupted sound can be produced with the concurrent interface management.

4. Conclusion

In this paper, we presented a behavioral emulation system, called iSAVE. iSAVE allows behavioral emulation of a chip to be performed in the context of its target system. We suggested a method of concurrently executing the core algorithm and the chip interface, which gives the following additional advantages: First, the interface can be described without modifying the algorithm. Second, as the overhead of checking the chip interface can be borne by the handler, the processor can concentrate on the core algorithm execution.

We successfully demonstrated prototype iSAVE system by the emulation of MP3 decoder chip as an application. We were able to run the whole MP3 decoder system by preparing only the interface model with the MP3 algorithm obtained from the public domain, which can be modified or replaced by the algorithm developer. The algorithm model and the application system need no modification for the emulation regardless of the hardware requirement of the application system.

5. For further reading

The following reference may also be of interest to the reader: [4].

References

[1] K. Wakabayashi, T. Okamoto, C-based SoC design flow and EDA tools: an ASIC and system vendor perspective, IEEE Tr. CAD 19 (12 December) (2000).

[2] C.J. Park, S. Lee, B.I. Park, H. Choi, J.G. Lee, Y.I. Kim, M.K. Jung I.C. Park, C.M. Kyung, Early in-system verification of behavioral chip models, in: Proceedings of High-Level Design Verification and Testing, 1999.

[3] N. Kim, H. Choi, S. Lee, S. Lee, I.C. Park, C.M. Kyung, Virtual chip: making functional models work on real target system, in: Proceedings of Design Automation Conference, 1998.

[4] B. Clement, R. Hersenmeule, E. Lantreibecq, B. Ramanadin, P. Coulomb, F. Pogodalla, Fast prototyping: a system design flow applied to a complex system-on-chip multiprocessor design, in: Proceedings of Design Automation Conference, 1999.

[5] J. Bauer, M. Bershteyn, I. Kaplan, P. Vyedin, A reconfigurable logic machine for fast event-driven simulation, in: Proceedings of Design Automation Conference, 1998.

[6] A. Rajawat, M. Balakrishnan, A. Kumar, Interface synthesis: issues and approaches, in: Proceedings of VLSI Design, 2000.

[7] P. Chou, R. Ortega, G. Borriello, Synthesis of the HW/SW interface in microcontroller-based systems, in: Proceedings of International Conference on Computer-Aided Design, 1992.

[8] A. Basu, R.S. Mitra, P. Marwedel, Interface synthesis for embedded applications in a codesign environment, in: Proceedings of VLSI Design, 1998.

[9] W. Wolf, Hardware-software co-design of embedded systems, in: Proceedings of the IEEE, July, 1994.

[10] B. Svantesson, S. Kumar, A. Hemani, A methodology and algorithms for efficient interprocess communication synthesis from system description in SDL, in: Proceedings of VLSI Design, 1998.

[11] L. Guerra, J. Fitzner, D. Talukdar, C. Schlager, B. Tabbara, V. Zivojnovic, Cycle and phase accurate DSP modeling and integration for HW/SW co-verification, in: Proceedings of Design Automation Conference, 1999.

[12] MAS3507D MPEG 1/2 layer 2/3 audio decoder manual, Macronas Intermetall, 1998.

[13] MPEG-1 and MPEG-2 audio draft software, Universitaet Hannover, ftp://ftp.tnt.uni-hannover.de/pub/MPEG/audio.

Seunjong Lee received the B.S. and the M.S. degrees in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST) in 1993 and 1995, respectively. He is currently working toward the Ph.D. degree in the Department of Electrical Engineering, KAIST. He is currently a research engineer with Dynalith Systems, Korea, where he has been since 2000. His current research interests include computer architecture and system verification. He received the Best Paper Award in the AP-ASIC in 2000.

Ando Ki received a B.S. degree in electronics engineering in 1986 from Hanyan University, a M.S. degree in electrical engineering in 1988 from KAIST, and a Ph.D. degree in computer science in 1997 from the University of Manchester. He was awarded the ETRI scholarship and the British Council scholarship for his study in UK. Before he pursued his Ph.D. degree, he worked for the ETRI as a senior researcher. He is currently the Chief Engineer at the Dynalith Systems.

In-Cheol Park received the B.S. degree in electronics engineering from Seoul National University in 1986, the M.S. and Ph.D. degrees in electrical engineering from KAIST in 1988 and 1992, respectively. Since June 1996, he has been an Assistant Professor and now an Associate Professor in the Department of Electrical Engineering and Computer Science at KAIST. Prior to joining KAIST, he was with IBM T.J. Watson Research Center, Yorktown, New York from May 1995 to May 1996, where he researched on high-speed circuit design. He received the Best Paper Award at the ICCD in 1999, and the Best Design Award at the Asia and South Pacific Design Automation Conference (ASP-DAC) in 1997. His current research interests include CAD algorithms for high-level synthesis and VLSI architectures for general-purpose microprocessors. He is a senior member of the IEEE.

Chong-Min Kyung received the B.S. degree in electronic engineering from Seoul National University, Korea, in 1975, and the M.S. and Ph.D. degrees in electrical engineering from KAIST, Korea, in 1977 and 1981, respectively. After graduation from KAIST, he worked at AT&T Bell Laboratories, Murray Hill, NJ, from April 1981 to January 1983 in the area of semiconductor device and process simulation. In February 1983, he joined the Department of Electrical Engineering at KAIST, where he is now a Professor. His current research interests include microprocessor/DSP architecture, chip design and verification methodology. He is Director of the Integrated Circuit Design Education Center (IDEC) established to promote the VLSI design education in Korean Universities through CAD environment setup, chip fabrication services, and providing various educational materials and media related with integrated circuits and systems design. He is also Director of Center for High-Performance Integrated Systems (CHiPS) located in KAIST. During 1993–1994, he served as the Asian Representative in the International Conference on Computer-Aided Design (ICCAD) executive committee. He also served as Vice Chairman of the 1999 COOLChips II held in Kyoto, Japan, and as Co-chair of the program committee of ASP-DAC 2000. He received the Most Excellent Design Award, and Special Feature Award in the University Design Contest in the ASP-DAC 1997 and 1998, respectively. He received the Best Paper Award in the 36th Design Automation Conference (DAC) held in NewOrleans, LA in June 1999, the 10th International Conference on Signal Processing Application and Technology (ICSPAT), Orlando, FL, in November 1999, and the International Conference on Computer Design (ICCD), Austin, TX in October 1999.