iSAVE: A Behavioral Emulator for in-System Algorithm Verification

Seungjong Lee, Moo-Kyung Jung, In-Cheol Park, and Chong-Min Kyung
Dept. of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology,
Taejon 305-701,Korea
E-mail: sjlee@vslab.kaist.ac.kr, icpark@ee.kaist.ac.kr, kyung@ee.kaist.ac.kr

Abstract

This paper presents a behavioral emulation system called iSAVE (in-System Algorithm Verification), which performs in-system verification of the behavioral description in C of a chip in the context of its application board at the early design stage. We were able to significantly increase the emulation speed by modeling the interface of the target chip with both the software part, which runs as thread, and the hardware part, mapped into FPGA logic, we can increase the emulation speed. The proposed idea is validated by demonstrating the behavioral emulation of MP3 decoder chip, as obtained from the public domain MP3 program.

I. INTRODUCTION

With recent VLSI systems which becomes ever more complex, the task of functional verification in the context of target application system has become mandatory for successful silicon at the first run. Structural emulation, where the gate-level model is mapped into FPGA, has been widely used in recent years as an integral part of the verification flow of various ASIC's and processors. Despite the speed and relatively wide functional test coverage, the structural emulation i.e., emulation at the gate-level has a limitation in reducing the time-to-market, as the gate-level model can be only obtained at the late design stage, putting the emulation at the end of the verification flow. Therefore, if some architectural errors are found during the emulation, it usually takes a very long time until the emulation runs again with the bug-fixed.

Another weakness of gate-level hardware emulation based on FPGA box lies in the difficulty of debugging with the application system. Many recent chip design projects the development of its application systems along with the chip design itself. Since the functional correctness of application system cannot be verified until the successful hardware emulation, the designer has to search for possible locations in both the chip and the application system. If it is possible to verify the application system before entering the FPGA-based hardware emulation, saving of the design time can be significant.

Many VLSI system designs, especially in signal processing, start with an algorithm model written in programming languages such as C. After verifying with simulation, the designers translate it into RTL model written in some hardware description language. We present in this paper a behavioral emulation system, which lets designers perform in-system emulation i.e., emulation in the context of the target system with a behavioral model of the chip to be fabricated, not its gate-level model. This technique has become feasible as current high-performance microprocessors usually have enough computing power to process quite complex algorithms within a reasonable, if not real time.

1.1. iSAVE System Overview

iSAVE (in-System Algorithm Verification) system is an emulation system that enables an algorithm or a behavioral chip model of a chip written in C to be verified in the context of its application system[2]. Once the behavioral C model is verified, it can be translated to HDL for synthesis. Originally this system was started from a demand to verify the functional correctness of an instruction-level model of CISC microprocessors with commercial applications in real environment[1], where an instruction set simulator as the behavioral model of the microprocessor begin designed runs with external bus model implemented in FPGA which, in turn, communicates with the outer world.

Figure 1 shows the concept of iSAVE system, which consists of two parts: host computer responsible for the generation of compiled code for the behavioral model or algorithm, and Virtual Chip Module (VCM) which, in turn, consists of Engine Processor (EP) and Pin Signal Generator (PSG). The first task of the chip designer is to write the algorithm or behavioral model of the chip, which runs on the EP in the VCM after compilation in the host computer.

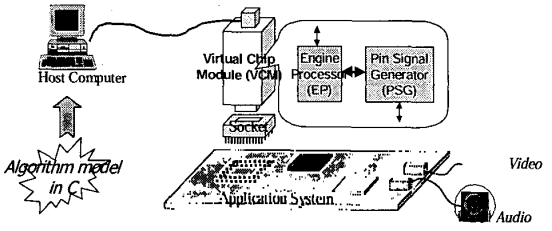

Figure 1. Concept of the iSAVE system.

Another task required by the designer is the description of chip interface models. The designer initially describes the chip interface as Finite State Machine (FSM), which is translated to HDL code by CAD tools. Finally, the interface models are mapped into the Pin Signal Generator (PSG), implemented in FPGA and runs while interacting with the application system.

The bandwidth between the algorithm model and the interface model often becomes the performance bottleneck of the behavioral emulator. In iSAVE system, the algorithm model and the interface model communicate with each other using command and data packet. Compared to signal-level communication, it reduces the communication bandwidth and, therefore, achieves higher emulation speed[1].

The behavioral emulation has the following benefits compared to the structural emulation with FPGA.

- The application system has to be slowed-down to meet clock speed of emulation. In iSAVE, clock is fed to the interface model only. It results that iSAVE can achieve faster external interface than the structural

0-7803-6470-8/00/ $10.00 © 2000 IEEE

emulation where all designs operate with a clock.

- Running the algorithm in software gives the designer more flexibility in debugging. Furthermore, time to restart emulation after bug fixing is significantly smaller than the structural emulation, as the algorithm or behavioral model is easier to diagnose and debug than the structural or gate-level model.

How to interconnect the algorithm-part of the chip model executed sequentially and the interface-part of the chip model executed in parallel is one of the largest design issues in the behavioral emulation.

1.2. Managing Chip Interface

While most algorithm models assume that every input is immediately available i.e., file access, the time to access data from external devices in the application system cannot actually be ignored in the real system. In translation process, these inputs in the algorithm-part of the model are replaced with code that handles the blocking input. Blocking input denotes the 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 valid inputs are ready. Moreover, for the case of multiple independent interfaces, the ratio of waiting time over total execution time can be significant, although it is not observable in real VLSI systems because this overhead is hided due to the nature of concurrent execution in the hardware architectures related to each interface. To obtain high-speed emulation, it is indispensable to make interface processing invisible to the algorithm.

In this paper, we propose a method to mange interface models of VLSI system to reduce the overhead of handling blocking inputs. Section 2 describes the related works to combine the algorithm and the hardware. In section 3, we propose a method to manage the chip interface model in the context of its cooperation with the algorithm-part of the whole chip model. In section 4, we validate our idea with emulating an MP3-decoder chip modeled in C and successfully verified in its application system using the proposed iSAVE system.

II. RELATED WORKS

There has been a need for rapid prototyping systems for many years[5,7]. Most of them were proposed as codesign tools as the means to profile performance or to verify a synthesized system. System description is partitioned into two parts: software and hardware.

Chou et al describes the interface synthesis between hardware and software[6], where device driver and interface logic are synthesized from the hardware specification. However, the interface mentioned is different from chip interfaces in the following aspects:

1. In[6], the software always controls the hardware without allowing the hardware giving commands to the software. In short, the hardware always runs as slave to the software. However, in case of actual chip interfaces, the interface is bi-directional, i.e., it is possible for the interface (hardware) to request data from the algorithm (software).
2. Interface in the codesign need not consider synchronization while HW/SW partitioning as it was already specified in the system model. In our case, chip interface model is responsible for synchronization with the application system.

These differences make interface design for the behavioral emulation different from that of codesign.

Another approach of behavioral emulation comes as an extension of structural emulator. Quickturn Mercury system[4] contains additional processors which executes RTL models of the system, while FPGAs executes gate-level models. Two models are connected with signals and simple logic is attached to interface software and hardware, while we uses commands and more complex logic including FSM and buffer.

III. INTERFACE IMPLEMENTATION

3.1. Interface Modeling

Since the algorithm model has been written without considering interfaces to the application system, the designer has to describe the operations of the interface as FSM. The interface model acts as bridge between the algorithm and the application system. It implies that the interface model can access variables of the algorithm while it generates signals on the pins for external connection to application system.

Because of strong connection between the chip model and the interface, the interface model has been implemented in software[9]. When the chip model sends a command with API(Application Programming Interface) functions, 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 hardware is longer than in the cosimulation. We need to consider following cases in implementing the interface model. The first is the bandwidth of the chip interface. Outgoing pin signals sent from the chip model is translated into real pin signals on the chip boundary with the simple interface model. These signals pass the bus where the processor and the interface hardware are connected. As the operating frequency of the bus is about x10 slower than that of the microprocessor operation, time consumed for communicating between the chip hardware and the simple interface model becomes the bottleneck of behavioral emulation. Hence, it is desirable to reduce the traffic due to this communication.

The other is the latency of the chip interface. Assuming the bus cycle that the application system sends a signal and waits its response from the emulated chip, the response time(T_R) can be calculated as the following:

$$T_R = 2 \times T_I + T_G$$

where T_I and T_G denotes the time taken between the pin and the interface model, and time to generate signal in the interface model, respectively.

Some interface cycles require the chip to response within the specified time. If cannot not meet, special application system needs to be designed. It is better to implement the interface model in hardware than in software in terms of the latency.

In the behavioral emulator, the pin signal model (PSM) is implemented in the hardware, as shown in Figure 2. The chip interface model(CIM) described in FSM is mapped in FPGA after being translated to HDL code in the behavioral emulator. PSM is responsible for understanding cycles of the interface and storing the acquired data in the buffer. When the application system requires some acknowledge signals from the chip, PSM also generates appropriate pin signals.

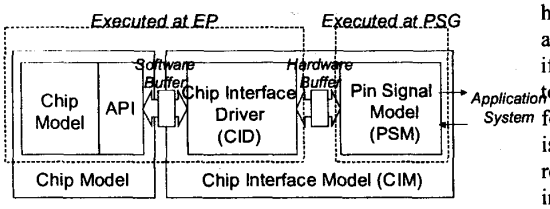

Figure 2. Connection between Algorithm and Interface Model

Besides the portion of the interface model implemented in hardware, e.g. FPGA, there are software part responsible for fetching data from the buffer in the hardware and assign it in the appropriate variable in the algorithm. This software part is called Chip Interface Driver (CID). However, not every CID executes such a unidirectional operation. Considering the interface that the application system reads a variable in the chip model with a command, CID returns data to PSM after it fetches and interprets the command. The whole CIM needs to be properly partitioned into CID and PSM such that the traffic between CID and PSM is minimal.

3.2. Buffer Assignment

As the signal rate of the chip interface is different from that of the algorithm, the algorithm and the chip interface need to communicate via buffer.

CIM has a role of the bridge between the chip model and the chip interface in the hardware, as shown in Figure 3, with two buffers. One buffer, named hardware buffer, is implemented in hardware and responsible for connecting CID and PSM. The other buffer, named software buffer, is implemented as array in software and responsible for connecting the algorithm and CID.

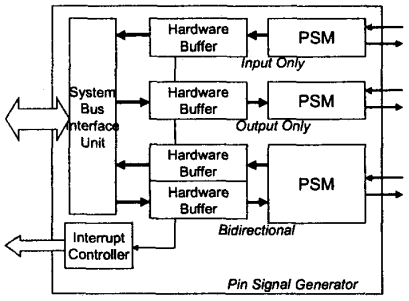

Figure 3. Block diagram of Pin Signal Generator (PSG)

Hardware buffer synchronizes between the application system and the system bus where the PSM hardware is attached. On the other hand, software buffer synchronizes between the algorithm and CIM. CID fetches the data from the hardware buffer and stores them in the software buffer, and then the chip model access the software buffer with the API function. The operation of the CID is described in Section 3.3.

The size of the software buffer should be larger than the amount of the data which the chip model can fetch at once. While the software buffer is flexible to change its size, the hardware buffer is limited in increasing the size due to the hardware resource constraints. The size of the hardware buffer also affects the execution time of the chip model. For example, if the hardware buffer is too small, it requires many CID calls to fill the software buffer and wastes large computing power for thread switching. On the other hand, if the hardware buffer is too large, it takes a long time to fill up and empty the buffer, resulting in slow context switching to the case of multiple interacting I/O devices, with separate buffers for each. To help the designer to determine the size of the hardware buffer, the interface synthesis algorithm is proposed.

3.3. Managing Chip Interface Driver with Thread

The chip interface driver (CID) runs independent of the chip model in EP, therefore, CID can simultaneously access the same variable with the chip model. This property enables us to implement CID with thread. The chip model and CIDs for each chip interface are allocated to different threads to be executed concurrently. The difference from traditional threaded program is that synchronization is not necessary in the behavioral emulator.

The chip model is usually executed in EP, but it is switched to the associated CID routine when CID needs to access the hardware buffer. It is called demand scheduling. Figure 4 shows the sequence of demand scheduling.

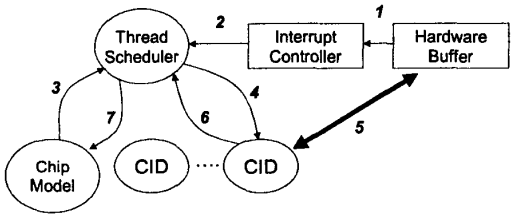

Figure 4. Sequence of Demand Scheduling

When the hardware buffer has enough data, the hardware buffer generates the event and marks the event register at the interrupt controller (1). If at least one event occurs, the event controller alarms the processor with an interrupt signal (2). Then, the event handler determines which hardware interface has generated the event and makes the chip model switch to the assigned CID (3,4). The called CID moves data in the hardware buffer to the software buffer (5). The processor returns to the execution of the chip model after CID updates the pointer of the software buffer (6,7).

IV. EXPERIMENTAL RESULTS

4.1. Implementation

ISAVE system was implemented using a Pentium-II processor and Altera FLEX series FPGA, which executes the algorithm model and the interface model, respectively. The kernel controls the communication with host computer and thread-level switching.

The development system is built with GCC as C compiler. Each CID is programmed as an independent C function but is possible to access the software buffer in the chip model. The kernel initializes each thread with the memory addresses of each CID routine before it executes the chip model.

The hardware buffer was implemented with embedded memory block available in the FPGA.

4.2. Application: MP3 Decoder Chip

An MP3 (MPEG-1 Audio Layer-3) decoder chip, MAS3507D was used as a vehicle to demonstrate the validity of the proposed method. The chip has three serial interfaces and extra pins, such as reset, power-pins[8]. The MP3 ISO model available in the public domain was used as the algorithm model of the chip.

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

Serial Output (SO)

CID moves the data in the software buffer to the hardware buffer, and then PSM 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 correct sound.

Serial Input (SI)

When the software buffer does not enough data, the chip must assert the DEMAND pin before it reads the data stream. CID controls the DEMAND pin along with loading the data from the hardware buffer. PSM aligns the pin signals in parallel and stores them in the hardware buffer.

I2C

The application system initializes the internal registers in the chip via this interface. In contrast to the other two interfaces described in FSM, its asynchronous operation makes us describe it in HDL code. Another special consideration is that state machine to interpret the cycles is located in the CID while CIM make the serial data in 8-bit word.

It was observed that although the processor consumes about 10% of its computing power for MP3 decoding, we failed to obtain the correct sound without concurrent interface management, since the hardware buffer assigned to serial output starves during MP3 decoding and filling of the hardware buffer assigned to serial input. We filled the hardware buffer assigned to serial output with highest priority among the chip interfaces and succeeded to hear seamless sound with concurrent interface management.

Figure 5 shows the prototype iSAVE system running with the MP3 player board. We emulated the MP3 decoder chip using a source code available from the public domain.

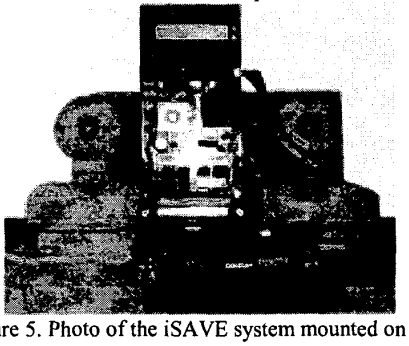

Figure 5. Photo of the iSAVE system mounted on the MP3 player board for in-system verification of the MP3 algorithm.

V. CONCLUSIONS

In this paper, we presented a behavioral emulation system, named iSAVE which consists of VCM(Virtual Chip Module) and host computer. ISAVE allows behavioral emulation of a chip to be designed in the context of its target system, which allows early verification the functional correctness of the algorithm to be implemented as a chip. We suggested a method of concurrently executing the core algorithm and the chip interface. It gives the following additional benefits: First, the interface can be described without modifying the algorithm. Second, as the overhead of checking the chip interface can be borne by the CID, the processor can concentrate on the core algorithm execution. Thus, in connection with the behavioral model running makes the system run significantly further than conventional FPGA-based emulation.

We demonstrated successful prototype iSAVE system by the emulation of MP3 decoder chip as an application. It is remarkable that 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.

For more convenient and fast emulation, we are currently working on a software tool which, once given the CIM description, automatically partitions CID and PSM.

REFERENCES

[1] N. Kim, H. Choi, S. Lee, S. Lee, I.C. Park, and C.M. Kyung, "Virtual Chip: Making Functional Models Work on Real Target System," in Proc. Design Automation Conf., 1998

[2] C.J. Park, S. Lee, B.I. Park, H. Choi, J.G. Lee, Y.I. Kim, M.K. Jung, I.C. Park, and C.M. Kyung, "Early In-System Verification of Behavioral Chip Models," in Proc. High-Level Design Verification and Testing, 1998

[3] B. Clement, R. Hersenmeule, E. Lantreibecq, B. Ramanadin, P. Coulomb, and F. Pogodalla, "Fast Prototyping: a system design flow applied to a complex System-On-Chip multiprocessor design," in Proc. Design Automation Conf., 1999

[4] J. Bauer, M. Bershteyn, I. Kaplan and P. Vyedin, "A Reconfigurable Logic Machine for Fast Event-Driven Simulation," in Proc. Design Automation Conf., 1998

[5] R. Gupta, C. Coelho, and G. Micheli, "Synthesis and Simulation of Digital Systems Containing Interacting Hardware and Software Components," in Proc. Design Automation Conf., 1992

[6] P. Chou, R. Ortega, and G. Borriello, "Synthesis of the HW/SW interface in microcontroller-based systems," Proc. Intl. Conf. on Computer-Aided Design, 1992

[7] W. Wolf, "Hardware-Software Co-Design of Embedded Systems," Proceedings of the IEEE, July, 1994

[8] MAS3507D MPEG 1/2 Layer 2/3 Audio Decoder Manual, Macronas Intermetall, 1998

[9] L. Guerra, J. Fitzner, D. Talukdar, C. Schlager, B. Tabbara, and V. Zivojnovic, "Cycle and Phase Accurate DSP Modeling and Integration for HW/SW Co-Verification," in Proc. Design Automation Conf., 1999