Hardware/Software Co-Simulation Test System for Embedded System

Fumihiko Mori

Keywords Embedded, Hardware/software co-simulation, RTL simulator, FPGA, ASIC, LSI

Abstract

At the embedded system product development center, the improvement of development efficiency is continually required to keep up with market trends of expeditiously advancing semiconductor miniaturization and large-scale software development. It is also imperative that our products be of high quality because our products are playing important roles that support social infrastructures.

As a measure against the aforementioned issue of development efficiency, hardware/software co-simulation for embedded systems have been in the spotlight. Under this technique, board operation is simulated before the start of actual board production.

Without using any dedicated co-simulator, we actively utilized the Register Transfer Level (RTL) Simulator in the embedded system development workstations in Japan. As a result, we built a hardware/software co-simulation environment without any new investment.

1 Preface

In the field of embedded system product development, improvement of development efficiency is continually required to keep up with market trends of expeditiously advancing semiconductor miniaturization and large-scale software development. It is also imperative that our products be of high quality because our products are playing important roles to support social infrastructures.

As a measure against the aforementioned issue of development efficiency, hardware/software co-simulation (“HW/SW co-simulation” hereafter) for embedded systems have in the spotlight. Under this technique, the board operation is simulated before the start of the actual board production.

Many dedicated co-simulators are sold by various EDA vendors. These prices are, higher than those of Register Transfer Level (RTL) simulators. We have already introduced RTL simulators. In addition, since the programming language used for the simulator is different from our commonly used ones, it is difficult to introduce these co-simulators.

In order to solve these issues and support the “improvement of development efficiency” and “high quality” required for the above embedded system products, we built the HW/SW co-simulation environment by actively using the RTL simulators. This paper introduces the co-simulation test system for an embedded system where the HW/SW co-simulation environment is made by actively using the RTL simulators.

2 Configuration and Features of the Co-Simulation Test System for an Embedded System

2.1 Central Processing Unit (CPU) Model

This model is devised to simulate the operation of the CPU to be adopted in our embedded system products. It can simulate bus operations using software of actual CPU by using the RTL simulator. In addition, it is possible to simulate functions of peripheral units around the CPU, such as functions of interruption and Direct Memory Access (DMA). The timing specifications are the same as those of an actual CPU, such as an operation clock number like an instruction execution number, an interrupt response clock number, and also a bus signal delay time.

For software implementation, an execution file is used. It is generated by the assembler file gener-

ation tool or the hand-coded assembler support tool to be described later.

2.1.1 Timing Monitoring Function

Violation of CPU timing specifications in regard to setup and holding time can be detected.

2.1.2 Dedicated Instructions for Debugging

Instructions to be supported by a CPU model come in the instructions of an actual CPU and also in the dedicated instructions for debugging. Some examples are shown below.

(1) PRINT instruction

Character strings can be arbitrarily displayed in the console window of the RTL simulator. This function is useful for software operation analysis.

(2) WAIT instruction

At the time of debugging or in the case of time adjustment for software execution, it is possible to produce execution waiting time for the CPU model. Reliable testing is possible for timing competition in cases of an interrupt and 2-port memory access.

2.1.3 Initialize Setup Function

Since necessary setups required for CPU operation, such as bus controller setting, have been incorporated in advance, no setup software is needed. After the reset condition is released, the CPU model can execute a required function immediately.

2.2 Peripheral Model

Similarly as for the CPU model described in **2.1** above, this model can simulate an operation of a memory or a peripheral Large Scale Integration (LSI) with the RTL simulator. The operation clock number and timing specifications are the same as those of actual devices. In addition, the timing monitoring function is also devised to be the same as for **2.1.1** above.

The memory model can set up the initial values at the start of simulation by virtue of the memory initialize function (to be related later).

2.3 Virtual Board

The CPU and peripheral models, as well as design models (RTL) of Field Programmable Gate Array (FPGA) and Application Specific Integrated Circuit (ASIC), are connected to produce the same configuration of the board. This model is then capable of simulation of board operation by using the RTL simulator. To make this virtual board, we use a block diagram editor incorporated in the RTL simulator Active-HDL made by Aldec., Inc. (“Active-HDL” hereafter).

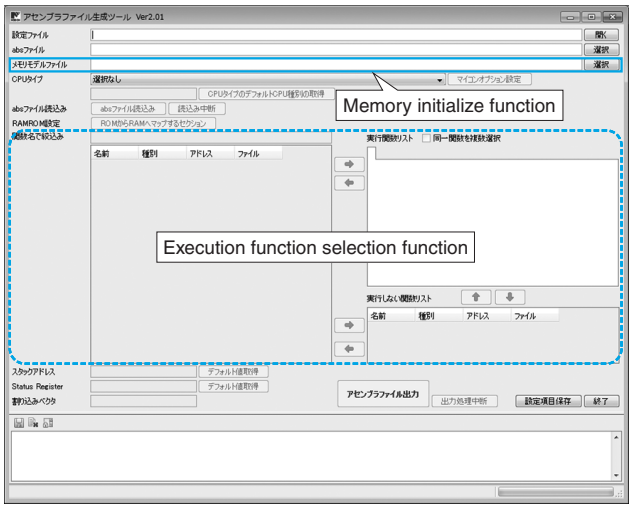


Fig. 1 **Assembler File Generation Tool**

The startup screen of the assembler file generation tool is shown.

2.4 Assembler File Generation Tool

Fig. 1 shows a screen of the assembler file generation tool. An executable object generated by the software development environment for CPU (to be discussed later) is converted into a format that can be executed by the CPU model, and an execution file is generated. This tool offers the following features:

2.4.1 Execution Function Selection Function

As stated in the **2.1.3**, no software processing for initial setup is needed for the CPU model. After the reset condition is released, the CPU model can execute a required function immediately. When a debugging function is selected, it becomes possible to generate an execution file that can execute only the required function. As a result, the execution time needed for reset start processing can be reduced and debugging efficiency can be improved.

2.4.2 Memory Initialize Function

A function selected by the execution function selection function is capable of an execution without pretreatment. In some cases, it requires that certain specified contents have to be in the memory as the initial values. When initial values for the memory are described in the memory, the assembler file generation tool generates an execution file combined with memory initial values. The memory model of the virtual board acquires and sets up memory initial values from the execution file. After that, the selected execution function can set up the required initial values in the memory model before execution.

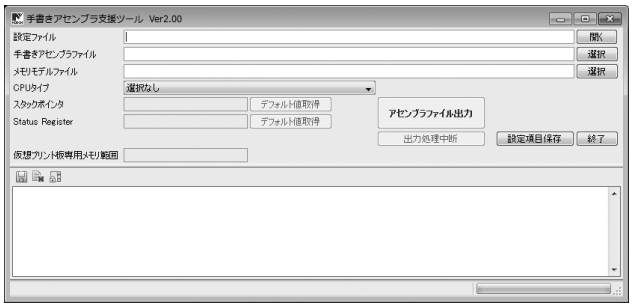
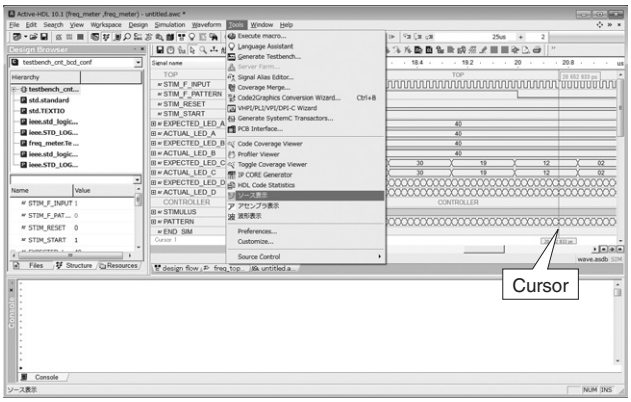
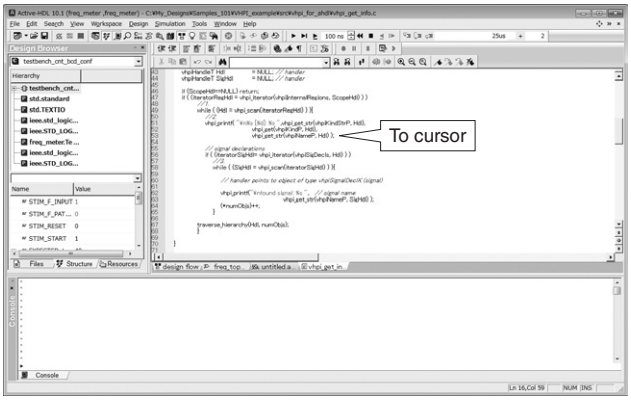


Fig. 2 Hand-Coded Assembler Support Tool

The startup screen of the hand-coded assembler support tool is shown.



(a) Waveform window



(b) Text editor

Fig. 3 Source Display Function

(a) Shows an arbitrary operating part of a simulation waveform.
(b) Shows the relevant part of a software source file.
※All information about designing is imagery.

2.5 Hand-Coded Assembler Support Tool

Fig. 2 shows a screen shot of the hand-coded assembler support tool. The assembler file generation tool described in **2.4** calls for an executable object that is generated by the software development environment for CPU (to be discussed later). For this reason, it is necessary to use a high degree of perfection software source that can generate objects. For the embedded system development, development of hardware and software is simultaneously promoted. In some cases, however, executable objects are not always available at the start of hardware debugging. The hand-coded assembler support tool can generate an execution file of the CPU model based on a program described in assembler language without using the above executable object. By virtue of this technique, it is possible to start hardware debugging without waiting for the completion of generating executable objects. Further, by using dedicated instructions for debugging related in the **2.1.2**, we could perform debugging work more efficiently.

2.6 Software Development Environment for CPU

This is a PC application software which generates executable objects for actual CPU. Similarly as for debugging with an actual CPU, we use the same software to be furnished by the CPU supplier.

2.7 RTL Simulator

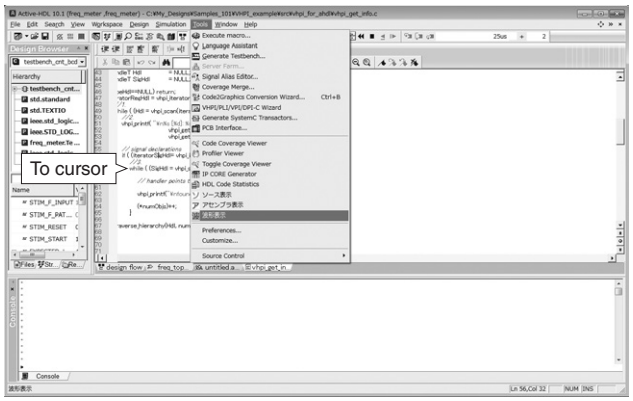
This is a PC simulator software which executes HW/SW co-simulation for virtual boards. We adopted the Active-HDL.

2.7.1 Source Display Function

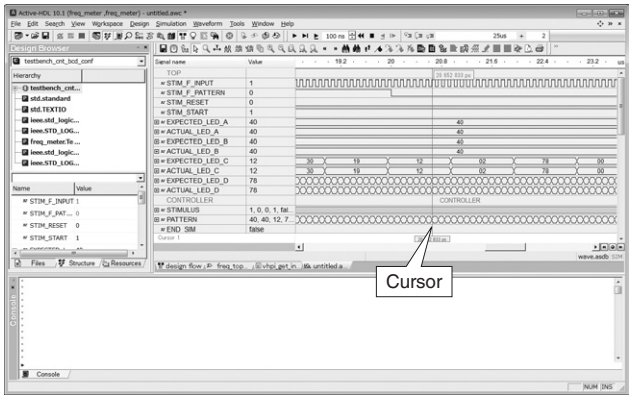
Fig. 3 shows the source display function, whose function is an add-on function made by the company. This source display function utilizes functions of a waveform window and text editor accommodated in the Active-HDL. As shown in (a), this function is executed by indicating any part of a waveform with a cursor in the state that the waveform is showing the result of simulation in the waveform window. Then, as shown in (b), the text editor opens a corresponding software source file and the cursor points at the associated line. In addition to the role of software source files, it is possible to display an assembler file compiled in the software development environment for the CPU. This function utilizes the user tool registration function furnished in the Active-HDL. This function can be executed by actuating a menu or a button on the Active-HDL screen.

2.7.2 Waveform Display Function

Fig. 4 shows the waveform display function. The waveform display function is an add-on function developed by the company. It utilizes a text editor integrated in the Active-HDL and also the wave-



(a) Text editor



(b) Waveform window

Fig. 4 Waveform Display Function

(a) Shows an arbitrary part of a software source file.
(b) Shows the relevant part of a simulation waveform.
※All information about designing is imagery.

form window. In the state that the simulation has been finished, the software source file is opened with the text editor as shown in (a). When an arbitrary point is designated with a cursor and this function is executed, a cursor of the waveform window moves to the designated point as shown in (b). This function utilizes the user tool registration function provided to the Active-HDL. It can be used by means of a menu operation or button operation to be attempted from the Active-HDL screen.

3 Postscript

This paper introduced the individual functions of our co-simulation test system for embedded systems. Thanks to the release of this system, it has become possible to perform board simulation by using the RTL simulator. This allowed us to better debug FPGA and ASIC prior to production of actual products. There is still room for improvement, however, on the simulation speed. In other words, it is difficult to apply this system for software debugging which requires a long simulation time.

Going forward, we will work on improving speed performance and make efforts to expand its application to the software development stage. In so doing, we will contribute further to the “improvement of development efficiency” and the “higher quality.”