High-speed Software-based Platform for Embedded Software of a Single-chip MPEG-2 Video Encoder LSI with HDTV Scalability

Katsuyuki OCHIAI, Hiroe IWASAKI, Jiro NAGANUMA, Makoto ENDO and Takeshi OGURA
NTT Human Interface Laboratories
1-1, Hikarinooka Yokosuka-shi, Kanagawa, 239-0847 Japan
E-Mail: {kach,hiroe,jiro,endo,ogura}@nttvdt.hil.ntt.co.jp

Abstract

This paper proposes a high-speed software-based platform for embedded software and evaluates its benefits on a commercial MPEG-2 video encoder LSI with HDTV scalability. The platform is written in C/C++ languages without any hardware description languages (HDLs) for high-speed simulation. This platform is applicable before writing up complete HDLs. The simulation speed is very fast and more than 600 times faster than compiled HDL simulators using RTL description. Fifty percent of the bugs in the final embedded software were located efficiently and quickly, and the design turn-around time was shortened by more than 25%. This platform provides sufficient performance and capability for validating practical embedded software.

1 Introduction

Recently, embedded system LSIs (system LSI) [1], which consist of a core CPU and application-specific hardware, have received considerable attention for their use in implementing various multimedia applications [2, 3]. These advanced applications require system LSIs that are larger in scale, perform more complex functions, and have higher performance than general LSIs. Due to the increasing complexity, the design turn-around time(TAT) of system LSIs have been on the increase, owing, in particular, to the longer time required for the validating embedded software on a core CPU as well as that of the hardware.

The following commercially available hardware description language (HDL) simulation tools for hardware and embedded software of system LSIs are available: compiled HDL simulators [4], hardware accelerators [5], field programmable gate array(FPGA)-based ASIC emulators [6], and hardware/software co-simulators [7]. However, these tools require HDLs, such as register transfer level (RTL) or gate-level hardware description. As a result, the embedded software of system LSIs is especially difficult or impossible to validate before writing up complete HDLs of the hardware.

On the other hand, general purpose programming languages such as C and C++ languages may provide a way to simulate system LSIs independently of any HDLs. These languages may also provide high-speed simulation capability because they have simple and fast codes without event scheduling which is an essential part of an HDL simulator. However, there is no established way to specify hardware behavior models with concurrency and synchronization.

To solve these problems, we propose a simple model to specify the hardware of system LSIs as well as propose a high-speed software-based platform for validating embedded software. This model based on the communicating sequential process(CSP) [8] specifies hardware block concurrency and synchronization by exploiting the guard mechanism. The platform based on this model, which consists of an instruction-level and a function-level simulator for a core CPU and application-specific hardware, respectively, is applicable before writing up complete HDLs. It is written in C/C++ languages without any HDL for hardware/software concurrent design and high-speed simulation.

We evaluated its benefits on a commercial MPEG-2 video encoder LSI with HDTV scalability [9, 10]. The simulation speed was very fast and more than 600 times faster than compiled HDL simulators using RTL description, and it had sufficient performance for simulating several to several ten of millions cycles of practical embedded software within one hour. Fifty percent of the bugs in the final embedded software were located efficiently and quickly by using interactive debugger and visualization tools. The design turn-around time was shortened by more than 25%. The software-based platform provides sufficient performance and capability for validating practical embedded software.

Structure of the paper. Section 2 shows the system LSIs and their concurrent design. Section 3 describes the new model and the software-based platform. Section 4 describes the evaluation results of applying the platform to a

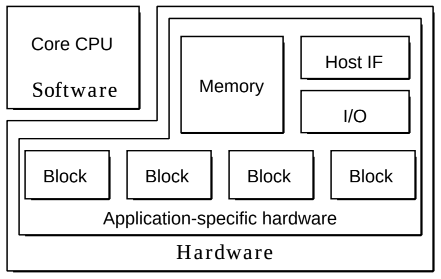


Figure 1. A system LSI architecture

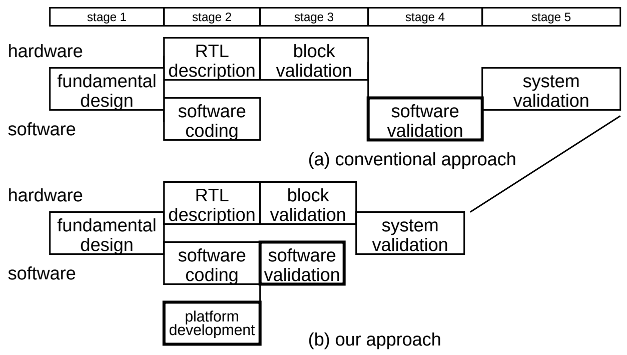


Figure 2. Concurrent design flows

commercial MPEG-2 video encoder LSI.

2 Concurrent Design of System LSIs

2.1 System LSI Architecture

A typical system LSI architecture is shown in Figure 1. It consists of a core CPU and application-specific hardware that includes memory, host interface, and dedicated hardware blocks. Interactions between the embedded software and the hardware are accomplished by accessing memory-mapped I/O. The embedded software on the core CPU can control application-specific hardware via this memory-mapped I/O. The embedded software on the core CPU provides the required flexibility and the dedicated application-specific hardware supports the performance. A system LSI satisfies the requirements of flexibility and performance for implementing various applications.

2.2 Concurrent Design Flows

The advantages of our concurrent design is apparent form Figure 2 in which our design is compared with a conventional one. In any practical system LSI design, chip designers must determine the hardware and software partitioning as a fundamental design (first stage of Figure 2). After hardware/software partitioning, each hardware description and software coding in the second stage is carried out concurrently.

In the conventional concurrent design, the embedded software can be written up concurrently with the HDL (second stage of Figure 2). However, validation of embedded software must wait until after validating the HDL (third stage of Figure 2) because of necessity of completing the HDLs for hardware interaction. Therefore, the third stage of software design process quite lengthy.

In contrast, in our concurrent design, a software-based platform for validating embedded software is developed while the embedded software and hardware are being written up in the second stage of Figure 2. As a result, the embedded software can be early validated on the platform in the third stage. Thus, our concurrent design has following major features:

- it eliminates the redundant spacing of conventional method (third stage of Figure 2),
- it reduces the time it takes for final system validation because of early embedded software validation with hardware interaction (final stage of Figure 2),
- it accelerates HDL validation by using the expected-value of hardware blocks, which can be extracted from the platform (third stage of Figure 2).

2.3 Requirements of Software-Based Platform

The software-based platform has following function requirements:

- validating embedded software requires mapped I/O access level compatibility with HDLs,
- extracting expected-value of hardware blocks requires a block's top access level compatibility with HDLs.

Moreover, to shorten platform development time, the platform is modeled as follows:

- a core CPU is specified at the instruction-level,
- application-specific hardware is specified at a higher level of abstraction, i.e. a block's top input/output action level.

These models along with the system configuration and function features of the software-based platform are described in the next section in detail.

3 Software-Based Platform

3.1 Simulation Model

The software-based platform consists of a core CPU simulator and an application-specific hardware simulator.

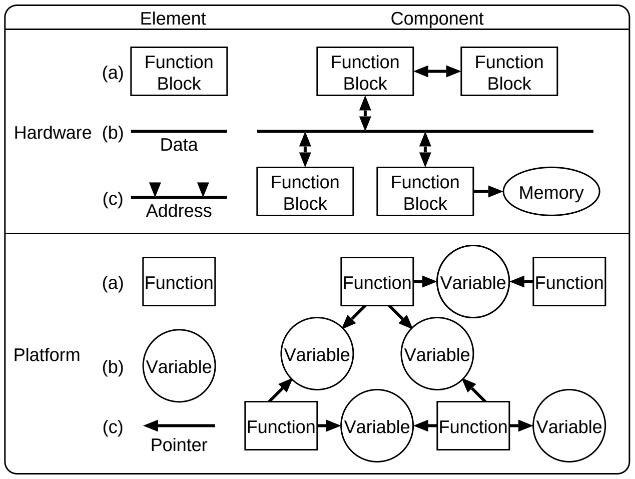


Figure 3. Hardware simulation model

3.1.1 Core CPU

The core CPU simulator is specified at an instruction-level for validating embedded software on the core CPU. Memory-mapped I/Os are general schemes for hardware/software interactions of a system LSI. Memories and hardware/software interface registers included in the application-specific hardware are mapped on the core CPU's data memory address space. The embedded software can access and control hardware blocks in the application-specific hardware via memory-mapped I/Os in the manner it does the data memory address space.

3.1.2 Application-Specific Hardware

An application-specific hardware is specified at a higher level of abstraction, i.e. a block's top input/output action level by exploiting the following three elements.

The model shown in Figure 3 consists of three hardware elements: (a) a *function block* element, (b) a *data* element and (c) a *address* element. A *function block* element is a functional set of hardware block. A *data* element is a read/write value of an I/O port. An *address* element is an I/O port address. In the platform, each hardware element is described as: (a) a *function* element, (b) a *variable* element, or (c) a *pointer* element.

In this model, data flow between hardware blocks and/or a core CPU is achieved by manipulating handshake flags as shown in Figure 4. In the write-enable mode, the left *function* can write a value to a *variable* but the right *function* cannot read it. In the read-enable mode, the right *function* can read a value from a *variable* but the left *function* cannot write it.

In this way, the data flow from the left *function* to the right *function* is accomplished by sharing the *variable*. This

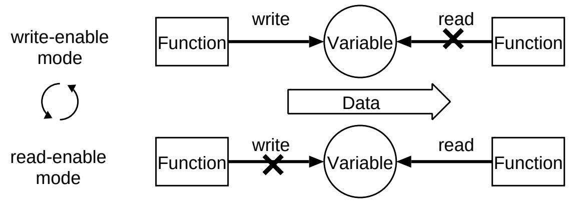


Figure 4. Communication using handshake flag

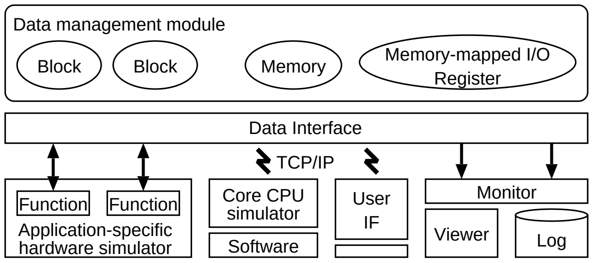


Figure 5. Software-based platform overview

mechanism is based on the guard mechanism of the communicating sequential process (CSP) [8]. Each *function* element is only maintained locally of its handshake flags; however, all *function* elements run concurrently with synchronization of the shared *variable*.

The data flow based on the guard mechanism is completely independent of the scheduling order of *function* elements. This enables a simulator to exploit a simple and fast round-robin scheduling without expensive event-driven scheduling, which is an essential part of a HDL simulator.

3.2 Software-Based Platform Overview

The overview of the software-based platform is shown in Figure 5. It consists of a core CPU, an application-specific hardware simulator with several function modules, a user interface, a monitoring module, a data management module, and a data interface.

The core CPU simulator written in C++ language is connected to the application-specific simulator using TCP/IP. The application specific-hardware simulator is written in C language based on the above-mentioned modeling without any HDL. The C function for the hardware block is designed independently of hardware functions except for only input/output actions. The platform runs on a UNIX machine.

Our approach using the C/C++ language for software-based platform design is sufficiently general that it can be applied to any system LSI simulation.

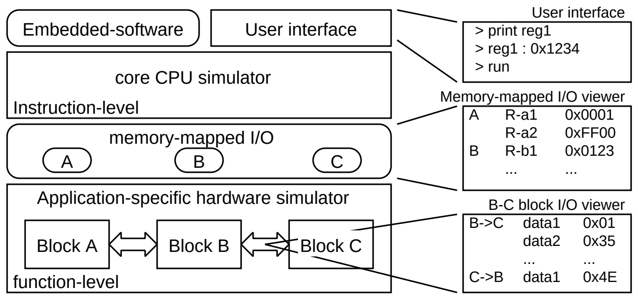


Figure 6. Functions of interactive debugger

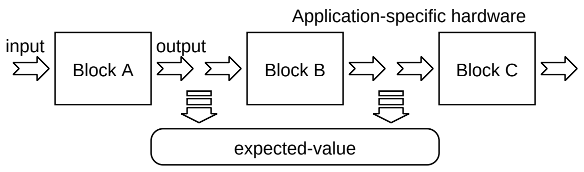


Figure 7. Extraction of expected-value

3.3 Function Features

The function features for implementing this platform are shown in Figure 6. They are a user interface and viewers of memory-mapped I/O and block I/Os. In order to validate embedded software, a designer handles the user interface to control the core CPU commands interpretively and looks at the viewer of memory-mapped I/O and block I/Os. Monitoring data, such as memory-mapped I/O and block I/Os, is stored as a log file. Moreover, the data can be visualized and displayed on viewers in real time using X-Window. These visualization tools are particularly useful for validating embedded software of multimedia system LSIs.

Moreover, this platform can extract expected-values of input/output pairs on each hardware block as shown in Figure 7. By comparing these values with a HDL simulation result, HDL block validation can be accelerated.

4 Evaluation and Discussion

4.1 Target System LSI

The target system LSI is an advanced commercially available single-chip MPEG-2 MP@ML video encoder LSI with HDTV scalability that is capable of being used in a multiple-chip MP@HL video. Its block diagram is shown in Figure 8. The features of this LSI are shown in Table 1. This LSI is a key component for digital satellite broadcast systems or digital storage systems such as DVD recorders.

It consists of a core CPU and application-specific hardware dedicated to MPEG-2 video that includes a search engine block and a SIMD processor block for motion compensation, and a variable length code (VLC) block. The embedded software on the core CPU controls these hardware blocks in real-time via one hundred and several dozen memory-mapped I/Os. The embedded software is written in the C language and its size is about 10k lines.

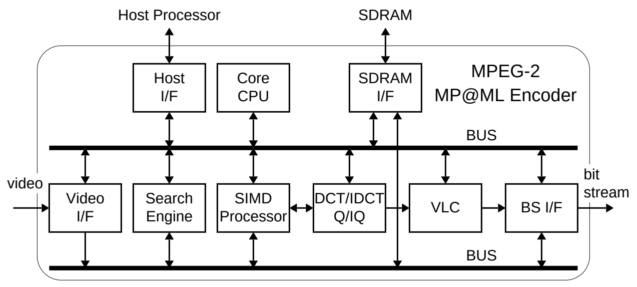


Figure 8. Block diagram of MPEG-2 video encoder LSI

Table 1. Features of LSI

Technology :	$0.25\mu m$ 4-level metal CMOS
Gates :	5.0 million transistors
Area size :	$9.8 \times 9.8mm^2$ die size
Clock :	$81MHz$
Power :	$1.5W@2.5V/3.3V$
Package :	208-pin QFP
Extension :	multi-chip MP@HL encoder

4.2 Validation of Embedded Software

The embedded-software of MPEG-2 video encoder LSI was validated by using interactive debugger and the visualization tools of the software-based platform. The debugging was accomplished hierarchically from a small unit to a large unit according to the MPEG-2 video coding unit shown in Figure 9. The simulation time for each unit is shown in Table 2. The designer was able to debug the small units of the embedded software interpretively and efficiently using memory-mapped I/O registers and block I/O values because small units of a macroblock or a slice take only a few seconds to simulate. Moreover, the designer was able to validate the large units (such as a picture or a GOP) with rate control by using a bit-stream analysis as a simulation log.

The validation using the platform in X-window on a workstation is shown in Figure 10. The user interface which operates the platform and some viewers of memory-mapped I/O and block I/Os, are filled with some X-window utilities, such as xeyes and login windows. Fifty percent of the total 230 bugs in the final embedded software were able to be located efficiently and quickly by using interactive debugger and visualization tools with simulating several to sev-

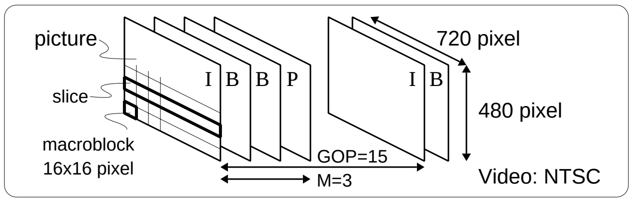


Figure 9. MPEG-2 video coding unit

Table 2. Simulation time

Coding unit	CPU time		NTSC video
macroblock	0.04	second	(720×480 pixel)
slice	1.9	seconds	M=3, GOP=15
picture	57	seconds	Sun Ultra 60
GOP	14	minutes	(300 MHz×2CPU)

eral tens of millions of cycles of practical embedded software. These implemented functions were sufficient to validate practical embedded software.

4.3 Simulation Performance

Our platform's simulation speed for 30 NTSC pictures is compared with those commercially available simulation tools in Table 3. The simulation speed of the platform was very fast despite being only software. It was more than 600 times faster than compiled HDL simulators using RTL description, and more than 40 times faster than hardware accelerators using gate-level descriptions. The simulation speed reached about 1/10th the speed of an FPGA-based ASIC emulator which is the fastest except for a real chip. This level of performance is sufficient to simulate several tens of millions of cycles of practical embedded software.

4.4 Short Design Turn-Around Time

The concurrent design enables a designer to overlap the embedded-software validation stage and the hardware block validation stage. This platform eliminates the extra time required for the redundant spacing that is characteristic of the conventional way (third stage of Figure 2-(a)). Moreover, it reduces the time required in the final system validation step because of early embedded software validation with hardware interaction (final stage of Figure 2-(b)). This platform can also accelerate hardware block validation using expected-value of hardware blocks (third stage of Figure 2-(b)). In these ways, a design's turn-around time can be shortened by more than 25% by using this platform.

Table 3. Comparison with other simulation tools

Simulator/Emulator	time (30 pictures)	
Real chip	1	second
ASIC emulator (QT)	3	minutes
Software-based platform	28	minutes
Hardware Accelerator (Zycad)	20	hours
Event simulator (VCS)	12	days

5 Conclusion

This paper has proposed a high-speed software-based platform for validating embedded software and has evaluated its benefits by applying to a commercial MPEG-2 video encoder LSI. The platform is written in C/C++ languages without any HDLs for hardware/software concurrent design and high-speed simulation. This platform is applicable before writing up complete HDLs. The simulation speed is very fast and more than 600 times faster than compiled HDL simulators using RTL description.

Fifty percent of bugs in the final embedded software were located efficiently and quickly by using interactive debugger and visualization tools. The design turn-around time was shortened by more than 25%. The platform has provided sufficient performance and capability for validating practical embedded software of the commercial MPEG-2 video encoder LSI with HDTV scalability.

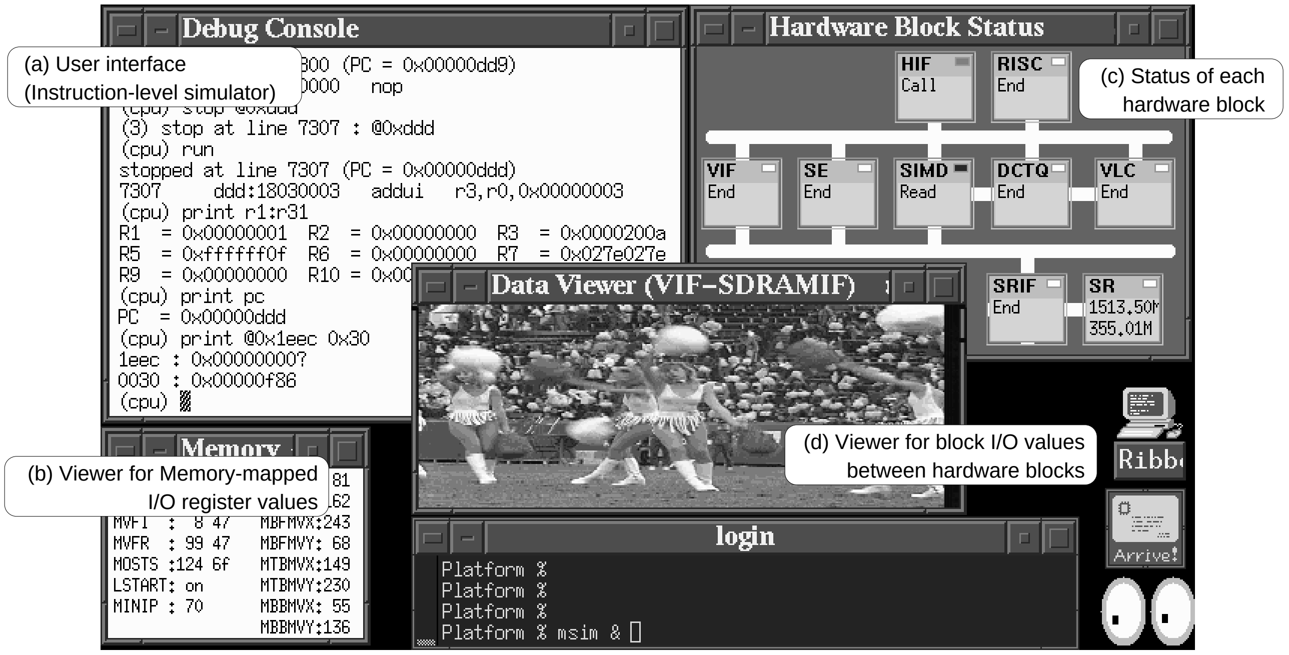
In the near future, we will study the performance limitations of the software-based platform, and we will investigate ways to expand it to a multiprocessor environment or network-based parallel and distributed environment for high-speed simulation. The approach of the software-based platform promises to be an important step towards embedded software validation of future system LSIs.

Acknowledgments

The authors are grateful to Dr. Susumu Ichinose of the NTT Human Interface Laboratories and Dr. Ryouta Kasai of the NTT System Electronics Laboratories for supporting this work. Furthermore, we would like to acknowledge of the members of the Visual Communication Laboratory for useful discussions during the development of this system.

References

[1] Daniel D. Gaiski, Frank Vahid, Sanjiv Narayan, and Jie Gong, "Specification and Design of Embedded Systems," *Prentice Hall*, 1994.



```
300 (PC = 0x00000dd9)
0000   nop
(cpu) stop @0xddd
(3) stop at line 7307 : @0xddd
(cpu) run
stopped at line 7307 (PC = 0x00000ddd)
7307     ddd:18030003   addui   r3,r0,0x00000003
(cpu) print r1:r31
R1  = 0x00000001  R2  = 0x00000000  R3  = 0x0000200a
R5  = 0xffffff0f  R6  = 0x00000000  R7  = 0x027e027e
R9  = 0x00000000  R10 = 0x00
(cpu) print pc
PC  = 0x00000ddd
(cpu) print @0x1eec 0x30
1eec : 0x00000007
0030 : 0x00000f86
(cpu) ▒
```

```
                  81
                  .62
MVFI  :  8 47    MBFMVX:243
MVFR  : 99 47    MBFMVY: 68
MOSTS :124 6f    MTBMVX:149
LSTART: on       MTBMVY:230
MINIP : 70       MBBMVX: 55
                 MBBMVY:136
```

```
Platform %
Platform %
Platform %
Platform % msim & ▯
```

Figure 10. Software-based platform on X-Window

[2] T. Kondo, K. Suguri, M. Ikeda, T. Abe, H. Matsuda, T. Okubo, K. Ogura, T. Tashiro, N. Ono, T. Minami, R. Kusaba, T. Ikenaga, N. Shibata, R. Kasai, K. Otsu, F. Nakagawa, and Y. Sato, "Two-Chip MPEG2 Video Encoder," *IEEE Micro*, pp. 51-58, April 1996.

[3] M. Inamori, J. Naganuma, H. Wakabayashi, and M. Endo, "A Memory-based Architecture for MPEG2 System Protocol LSIs," *The European Design and Test Conference (ED&TC)*, March 1996.

[4] "Chronologic VCS Reference Manual (Release 4.1)," *Synopsys, Inc.*, March 1998.

[5] "LightSpeed Simulation Server Reference Manual (Version 1.0)," *Zycad Corporation*, November 1996.

[6] "System Realizer Reference Manual (Version 5.1)," *Quickturn Design Systems, Inc.*, December 1997.

[7] "Seamless CVE Reference Manual (Release 2.4)," *Mentor Graphics Corporation*, July 1998.

[8] C. A. R. Hoare, "Communicating Sequential Processes," *Comm. ACM*, Vol. 21, No. 8, pp. 666-677, August 1978.

[9] T. Minami, T. Kondo, K. Nitta, S. Suguri, M. Ikeda, T. Yoshitome, H. Watanabe, H. Iwasaki, K. Ochiai, J. Naganuma, M. Endo, E. Yamagishi, T. Takahashi, K. Tadaishi, Y. Tashiro, N. Kobayashi, T. Okubo, T. Ogura, and R. Kasai, "A Single-Chip MPEG2 MP@ML Video Encoder LSI with Multi-chip Configuration for a Single-board HP@ML Encoder," *Hot Chips 10*, August 1998.

[10] M. Ikeda, T. Kondo, K. Nitta, K. Suguri, T. Yoshitome, T. Minami, J. Naganuma, and T. Ogura "An MPEG-2 Video Encoder LSI with Scalability for HDTV based on Three-layer Cooperative Architecture," *Proc. of the Design, Automation and Test in Europe (DATE)*, March 1999.