A Hardware/Software Concurrent Design for a Real-Time SP@ML MPEG2 Video-Encoder Chip Set

Mitsuo Ikeda†, Tsuneo Okubo†, Tetsuya Abe†, Yoshinori Ito‡,
Yutaka Tashiro‡ and Ryota Kasai†

† NTT LSI Laboratories
3-1, Morinosato Wakamiya, Atsugi,
243-01, Japan

‡ NTT Human Interface Laboratories
1-2356, Take, Yokosuka,
238-03, Japan

Abstract

This paper presents a design for a real-time MPEG2 SP@ML video-encoder chip set. Its main features are: hardware/software partitioning based on a software encoder analysis, and a pipeline architecture where hardware and software interact closely and smoothly. We use a hardware/software concurrent design technique with fast verification to avoid major modifications at architectural and RTL levels. The chips were successfully fabricated with 0.5-μm CMOS technology.

1 Introduction

The adoption of MPEG2, the international video compression standard[1], along with advances in ATM technology, have brought multimedia communication services closer to reality. The MPEG2 video-encoder system[2] in particular, provided it can be implemented at a reasonable cost, promises users widespread bi-directional communication services. The MPEG2 encoder chip sets developed so far [3][4], however, do not reach the required level of performance and usability for practical applications.

We recently proposed a two-chip encoder with a "simple profile and main level (SP@ML)" function[5]. This function requires several GOPS of various processing in MPEG2 technology such as "motion compensation", "discrete cosine transform" and "Huffmann coding". The main feature of our encoder was the introduction of a three-step hierarchical telescopic search algorithm for motion estimation to reduce the number of operations. However, a critical issue remained the creation of an architecture that can achieve field programmability for rate control of the bitstream-output while maintaining the hardware efficiency. Because smooth rate-control is crucial to good picture quality, its algorithm has to be as flexible as applications demand. To solve these problems, we use a hardware/software concurrent design approach that starts with an analysis of the software encoder where the MPEG2 core algorithm is described in C language. From the analysis, we decide what functions should be assigned to the core processor or other hardware modules, i.e., software/hardware partitioning[6][7], and determine how closely and smoothly the hardware and software cooperate with each other. Using this information, we developed a flexible macroblock[1]-level pipeline architecture controlled by a high-performance RISC core processor. This architecture achieves a "main-level" encoding throughput of 40,500 macroblock/sec with complete rate-control.

From the viewpoint of "turn around time"(TAT) and re-engineering, our focus of concern is how to avoid modifications at the architectural, functional and RTL design levels. We therefore address the estimation of hardware and software module at each abstraction level. Precise estimation at each level makes it possible to design the chips without major modifications back at the upper level.

Section 2 presents the design methodology. The analysis of encoding functions and hardware/software partitioning are presented in Section 3. The chip architecture is described in Section 4. Section 5 presents verification and implementation results. Section 6 pro-

[1] A basic unit in MPEG2. Composed of 16×16 pixels

ED&TC '96
0-89791-821/96 $5.00

vides concluding remarks.

2 Methodology

Figure 1 shows the design methodology. The objective is the concurrent design of hardware and software and the evaluation of hardware/software cooperation.

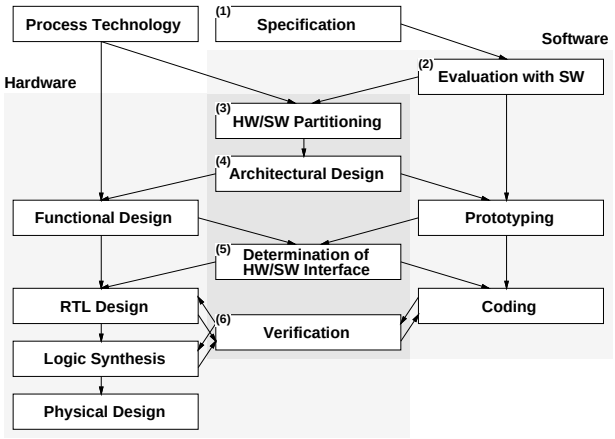

Figure 1: Methodology Model

First of all, (1) specifications are made for encoder functions: SP@ML with three-step rate control (*RC*) which fully conforms to the international standards. This is followed by the (2) characterization of each function that should compose the target LSIs using a software encoder developed to evaluate the algorithm. Some measures to quantify the functions are introduced. The evaluations with these measures have to be accurate and carefully considered to avoid re-engineering at the next design stage. The next step is (3) partitioning into hardware and software parts according to the characterization. This is followed by (4) architectural design. Certain features of the core processor, such as instruction sets and the bit-width of data, are determined at this phase by analyzing the processing assigned to software. (5) Determining the interface between the hardware and software and designing the pipeline architecture follow. The timing of communication between modules is a key issue. The pipeline architecture for the whole encoding is designed at this phase. Up to this point, C language and its utilities are the main tools used. The hardware modules are described in an RTL language, SFL (Structured Function description Language). The last stage is the (6) verification of the architecture using the RTL simulator SECONDS, commercially available netlist level simulators and an ASIC emulator.

3 Analysis of Encoding and HW/SW Partitioning

3.1 Analysis of MPEG2 Encoding

Figure 2 is a diagram of the functions of the MPEG2 SP@ML encoder, which is implemented by a software model in C language. The values in parenthesis indicate the performance requirement extracted from the model.

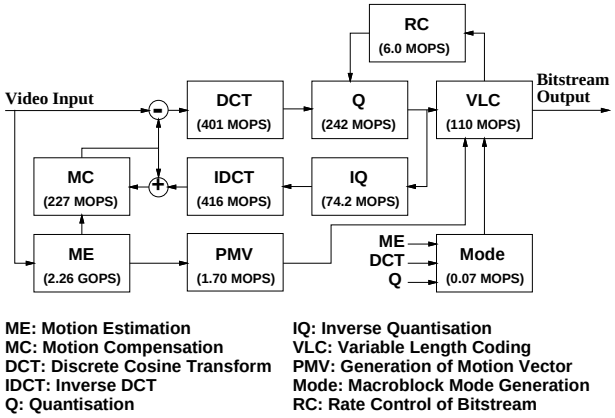

Figure 2: Functions of the MPEG2 Video-encoder

First, the functions are qualitatively classified into three categories in terms of flexibility, or fixed-unfixed criteria:

1. Functions that should be flexible because of the possibility of modifications in design or programmability requirements in the field (*RC*, *PMV* and *Mode*).
2. Functions where that processing is fixed (*DCT*, *IDCT*, *MC* and *IQ*).
3. Functions that have both properties 1 and 2.

A quantitative estimation is then performed to characterize the functions more clearly. The following measures are used for this step.

1. The total number of operations (called "*throughput*" hereafter). This represents the suitability for hardware implementation.

2. The frequency of branch operations ("*complexity*"). This reflects the complexity of processing and helps in estimating the difficulty of hardware implementation.

These measures are estimated by summing instructions processed in the software. Since the instruction sets of the core processor are not determined at this phase, the number of executed instructions are measured using a DLX[8] assembler. The results are shown in Fig. 3.

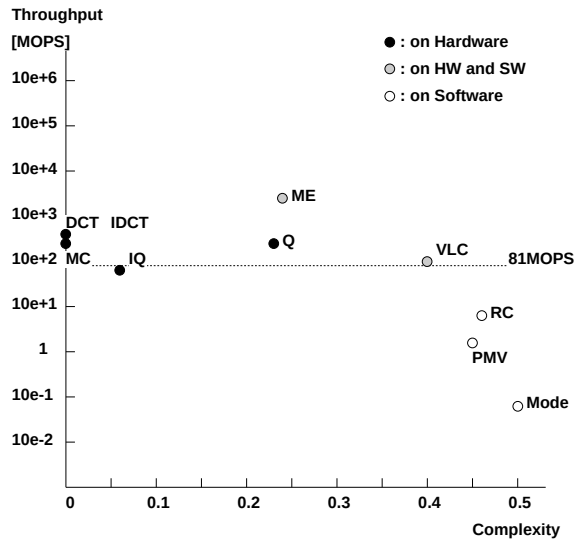

Figure 3: Estimation of Encoder Functions

3.2 Hardware/Software Partitioning

From the qualitative properties and quantitative estimation, the partitioning of the main functions is done under the following assumptions: (1) The clock of the encoder chip is 81 MHz. This is because the system clock of the CODEC board is 27 MHz and the operating clock frequency should be a multiple of that value and the 0.5-μm CMOS technology we provide is more practical at a clock frequency of less than 100 MHz. (2) The built-in core processor has an ordinary Harvard architecture and no specific operations tuned for image signal processing.

Details of the partitioning are as followings:

1. A function that deals with picture pixel data directly is not allocated on just the software module for the following reasons. Essentially, such a function has the potential of about 10 MOPS for a 30 frame/sec image sequence with 720 × 480 pixels, because one operation of a pixel data under this condition brings $30 \times 720 \times 480$ operations per second for 8-bit data. Among the main functions, *ME*, *MC*, *DCT* and *IDCT* are categorized in such functions.

2. The *RC* function is given top priority for software allocation, due to the intense programmability requirement and its low "*throughput*" and high "*complexity*". The same is true for *PMV* and *Mode*.

3. *DCT*, *IDCT*, *MC* and *IQ* are clearly allocated to hardware because of their high "*throughput*" and low "*complexity*".

4. *VLC* features about 100 MOPS "*throughput*" and comparatively high "*complexity*" because parameters tend to vary widely according to the applications. Therefore, it is implemented in both software and hardware. Partitioning of the *VLC* functions is as shown in Fig. 4. Header-bitstream generation functions such as *SEQ*, *GOP*, *PIC* and *SLC* have low "*throughput*" under 0.1 MOPS and comparatively high "*complexity*". In addition, they have a large variety of parameters. They are therefore on software. On the other hand, Huffmann coding functions *BIT*, *COEF* and *MB* have "*throughput*" over 10 MOPS and comparatively low "*complexity*". They are partitioned into the hardware module.

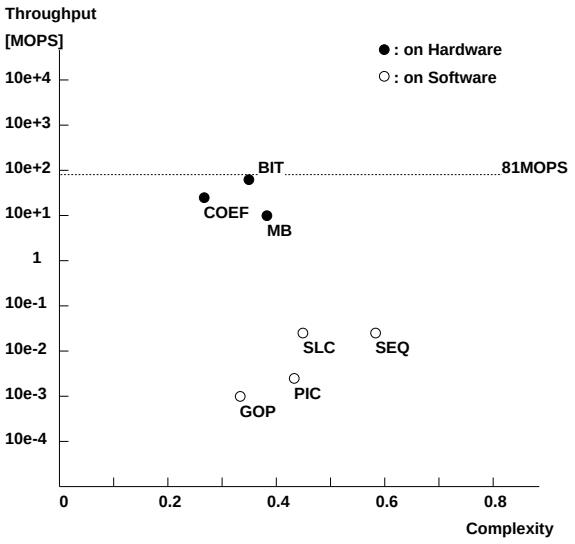

Figure 4: Estimation of VLC Functions

5. The Q function has relatively high "*complexity*" because it has an adaptive processing in choosing a quantisation step according to RC. However, we allocate it to hardware because of its similarity to IQ.
6. The ME algorithm, the three-step hierarchical telescopic search method, requires complex control to combine its subfunctions, each of which requires high "*throughput*". We therefore allocate this to both hardware and software to avoid re-engineering due to its complexity.

4 Architecture of the Encoder

4.1 Chip Architecture

Based on the results of partitioning, we decided on the chip-level architecture shown in Fig. 5. Hardware modules such as $DCT/IDCT$, Q/IQ, VLC

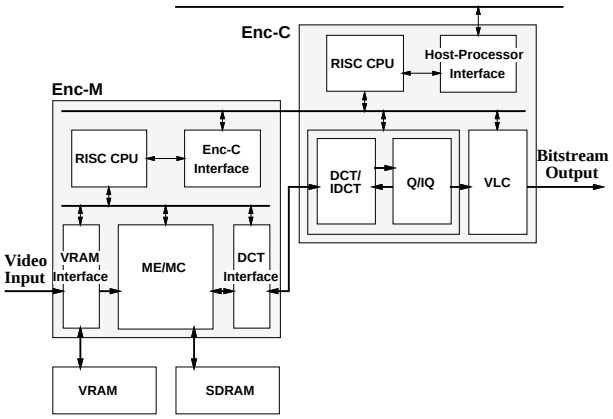

Figure 5: Chip Architecture

and ME/MC are controlled by a core processor with a flexible macroblock-level pipeline architecture. The processor also processes RC, $Mode$ and PMV. Since ME is a complex procedure that requires a huge amount of hardware, we implement ME on an independent chip (Enc-M) with MC. Other modules are accommodated in another chip (Enc-C). Another processor is introduced into Enc-M to shorten the design TAT for the complex control. From the viewpoint of the control structure, the Enc-M chip is a module controlled by the processor in the Enc-C. To control each hardware module with the processor, each hardware module has a memory-mapped I/O for the processor. In other words, the storage on the hardware module interfaced to the processor is treated as a memory-mapped storage for software.

4.2 Pipeline Architecture

A prototype software is developed to determine the timing of interactions between the hardware and software. The number of steps required when the functions are executed is calculated for each software and hardware module. The number of variables and instructions in the prototype software are estimated to determine the size of the data memory and instruction memory on the core processor.

In the MPEG2 standard, a macroblock is treated as a basic unit of encoding. Therefore, it is natural that a macroblock-level pipeline architecture is introduced for the processing. Table 1 shows the cycles required to perform a macroblock process on each hardware and software module. Required throughput corresponds to a 2,000 clock/macroblock for the 81-MHz clock rate. Since ME/MC uses 1,920 clocks, if

Table 1: Operation Cycles for a Macroblock

Module	Average	Maximum
ME/MC	1,920	1,920
DCT/IDCT	1,872	1,872
Q/IQ	1,872	1,872
VLC	1,152	4,608
Software	878	1,214

more than an 80-clock waiting period occurs in a macroblock process, the pipeline becomes clogged. Similarly, if a macroblock waiting period occurs in a slice (i.e. a row of macroblocks), the same thing happens. Moreover, the VLC module costs a maximum of over 2,000-clock cycle. From the viewpoint of hardware cost, it is wasteful to construct a pipeline architecture with a fixed cycle of 2,000. We therefore granted some flexibility in cycles for the macroblock-level pipeline.

Figure 6 shows the pipeline schedule of a three-macroblock sequence and the data dependency between main functions. The timing of interaction between modules is determined on the architecture. This pipeline architecture has the following features:

1. Each hardware module sends (or receives) data to (or from) the core processor once a macroblock. To

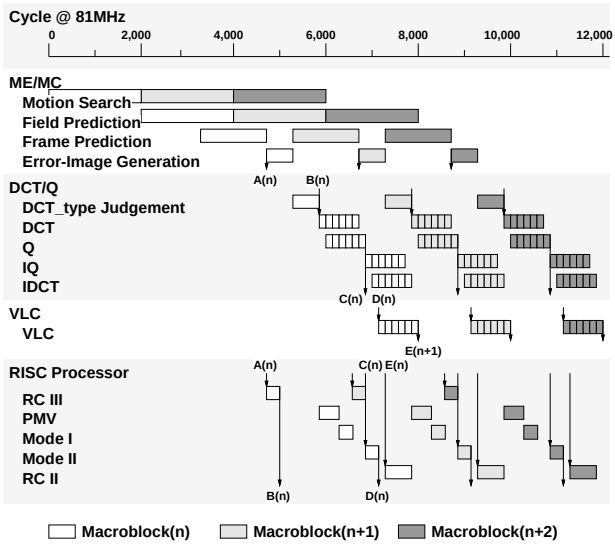

Figure 6: Pipeline Schedule of Macroblocks
(The arrows indicate main data dependencies.)

avoid a hardware handshake between the ME/MC module and the VLC module, data such as motion vectors transferred from ME/MC to VLC are handled by software.

2. *DCT* and *IDCT* processing share the hardware resource, as do *Q* and *IQ* processing.

3. There are some memories between ME/MC and DCT/Q and between DCT/Q and VLC for macroblock-data buffering. Even if the software on the core processor performs other tasks and stops a hardware module process, other modules continue their processing until all the memories become full.

4.3 Architecture of the Core Processor

For the design of the core processor architecture, the prototype software for each function allocated on software is estimated in detail. An evaluation of operations for a macroblock (Table 2) shows that a more than or equal to 16-bit-wide datapath and the multiplication and division functions are needed for the processor. A comparison of 16- and 32-bit datapaths (Table 3) shows that a 16-bit datapath is suitable because of its low area cost and the small increase in program steps on the application software. As a result, we decided upon the built-in core processor architecture shown in Fig. 7. 24-bit-wide instructions are used to minimize hardware cost. It has five pipeline-stages

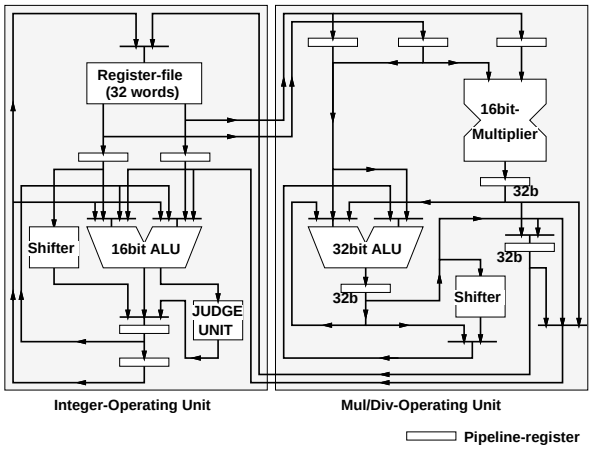

Figure 7: Datapath on the Core Processor

and operates at a frequency of 81 MHz and achieves a maximum of 81 MIPS.

Table 2: Operations for a Macroblock

Operation	# Operations	# Cycles
16bit Integer	690	690
16bit Mul./Div.	21	184
32bit Mul./Div.	0.03	2.7
Total	711	874

Table 3: Comparison of 16/32bit datapath

Datapath	16 bit	32 bit
Gates	9.5K gates	17.1K gates
Area of CPU	8.84 mm^2	16.46 mm^2
Instruction Mem.	120Kb	160Kb
Steps†	4,720	4,500

†:Static Steps of Software

5 Verification and Implementation

The methodology for hardware/software co-verification is shown in Fig. 8. To verify the functions on hardware and software efficiently, software modules are separated into two groups: modules for control of hardware and modules like those for *RC* calculations,

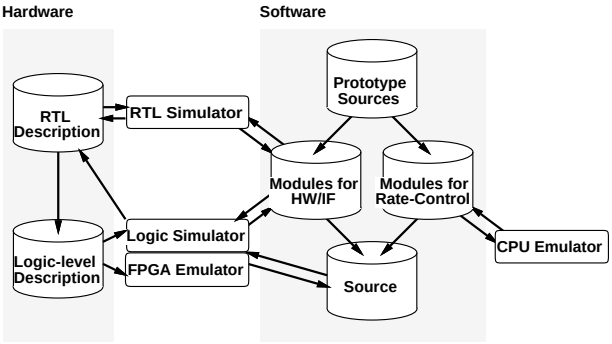

Figure 8: Co-verification Model

which run independent of the hardware modules. The former is verified and debugged by co-simulation. The latter is validated using the software emulator for the RISC core processor. For the validation of software and hardware, the expected outputs of each hardware module and of the whole encoding process are generated by the software encoder.

Cooperation of hardware and software on the pipeline architecture was validated by RTL and logic-level simulators. Figure 9 shows a simulation result for an encoding sequence. The quantiser scale code (QSC), the signal/noise ratio (SNR) of the output, and the total number of bits encoded are shown in macroblocks. QSC is the output of the RC function and a variable which dominates the quantisation error and effects SNR directly. As another effect of QSC, the total number of bits is controlled to meet the given bit-rate condition. The bitstream-output and SNR are estimated by the software encoder and the software decoder developed. The whole sequence of encoding was validated using an FPGA emulator.

The Enc-C and Enc-M were fabricated using a 0.5-μm triple-metal CMOS technology. The characteristics of the LSIs are shown in Tables 4 and 5. Figures 10 and 11 are the photographs of Enc-C and Enc-M, respectively.

6 Conclusion

A real-time MPEG2 SP@ML encoder chip set was designed with hardware/software concurrent design techniques. Starting with an analysis of a software encoder, a flexible macroblock-level pipeline architecture was devised. The chips were designed without major

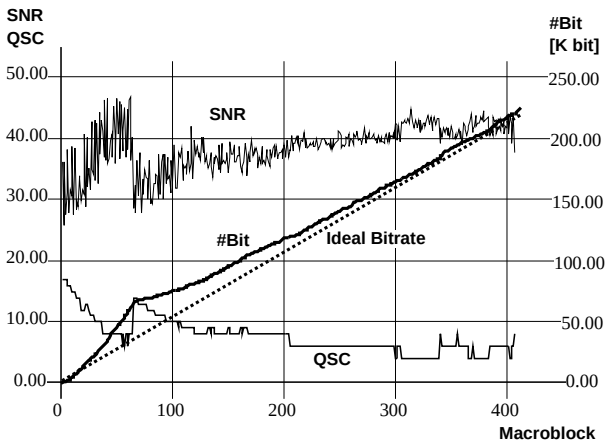

Figure 9: Result of an Encoding Sequence

Table 4: Characteristics of LSIs

	Enc-C	Enc-M
Device	0.5-μm triple-metal CMOS	
Clock	81MHz	
Transistors	1.3M Tr.	2.0M Tr.
Program Memory	24b × 5Kword	24b × 3.5Kword
Data Memory	16b × 1Kword	16b × 1Kword
Other Memories	20Kb	91Kb
Area	$14.0 \times 14.0 mm^2$	$16.5 \times 16.5 mm^2$
Power	2.5W @ 3.3V	3.5W @ 3.3V
Package	204-lead CQFP	340-lead CQFP

Table 5: Functional Features of LSIs

Standard	Simple profile @ Main level
Frame Size	720×480/720×576/etc.
Frame Rate	Up to 30 frames/sec
Bit Rate	Up to 15 Mbps
ME Range	-32/+31.5 × -32/+31.5
Encoding Delay	$\leq$ 85 ms

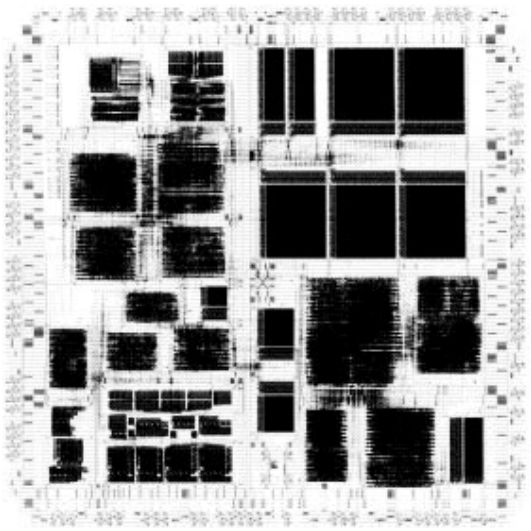

Figure 10: Photo of Enc-C chip

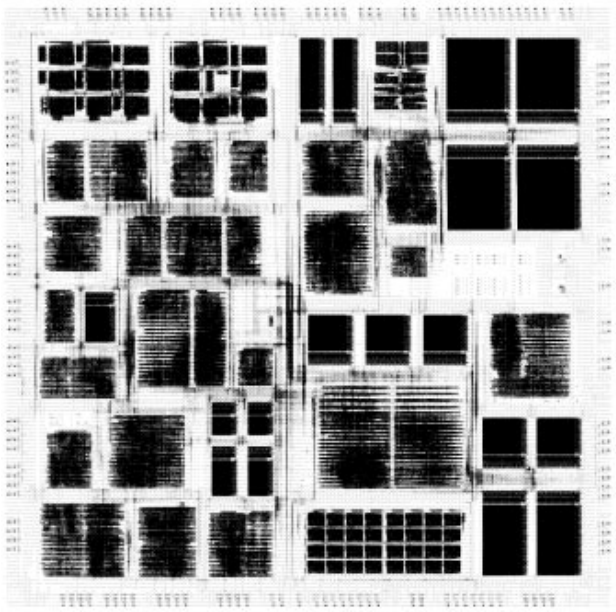

Figure 11: Photo of Enc-M chip

modifications at the architectural, functional and RTL design levels. They were successfully fabricated with 0.5-μm CMOS technology and perform encoding of 30-frames/sec image sequences for 720 $\times$ 480 pixels at an operating frequency of 81 MHz. They are used on MPEG2 system boards with an ATM interface for a CATV service and PC boards for a bi-directional communication service.

Acknowledgements

We would like to thank Osamu Karatsu of the NTT LSI Laboratories for supporting this work. Thanks are also due to Hironori Yamauchi of the NTT Human Interface Laboratories and the members of the Advanced LSI laboratory for their helpful suggestions.

References

[1] "INFORMATION TECHNOLOGY – GENERIC CODING OF MOVING PICTURES AND ASSOCIATED AUDIO ISO/IEC 13818-1 International Standard (System)," November 11, 1994.

[2] "INFORMATION TECHNOLOGY – GENERIC CODING OF MOVING PICTURES AND ASSOCIATED AUDIO ISO/IEC 13818-2 International Standard (Video)," November 11, 1994.

[3] T. Matsumura et al. "A Chip Set Architecture for Programmable Real-Time MPEG2 Video Encoder," IEEE Custom Integrated Circuits Conference, 1995.

[4] J. Armer et al. "A Chip Set for MPEG2 Video Encoding," IEEE Custom Integrated Circuits Conference, 1995.

[5] T. Kondo et al. "A Two-Chip Real-Time MPEG2 Video Encoder with Wide Range Motion Estimation," Symposium HOT Chips VII, 1995.

[6] R. Ernst, J. Henkel and T. Benner, "Hardware-Software Co-synthesis for Microcontrollers," IEEE Design & Test, pp.64-75, Dec. 1993.

[7] J. Gong, D.D. Gajski and S. Narayan, "Software Estimation Using A Generic-Processor Model," Proc. of the European Design and Test Conference, 1995.

[8] J. Hennessy and D. Patterson, "Computer Architecture : A Quantitative Approach," Morgan Kaufmann, 1990.