System-level Function and Architecture Codesign for Optimization of MPEG Encoder

Jinku CHOI[1], Nozomu TOGAWA[2], Masao YANAGISAWA[1], and Tatsuo OHTSUKI[1]
[1] Department of Electronics, Information and Communication Engineering Waseda University
3-4-1 Okubo, Shinju-ku, Tokyo 169-8555, Japan.
Tel: +81-3-5286-3396, Fax: +81-3-3203-9184
E-mail: choezg@ohtsuki.comm.waseda.ac.jp
[2] Department of Information and Media Sciences, The University of Kitakyushu
1-1 Hibikino, Wakamatsu-ku, Kitakyushu 808-0135, Japan.
Tel: +81-93-695-3264, Fax: +81-93-695-3368

Abstract: The advanced in semiconductor, hardware, and software technologies enables the integration of more complex systems and the increasing design complexity. As system design complexity becomes more complicated, System-level design based on the IP block and processor model is more needed in most of the RTL level or low level. In this paper, we present a novel approach for the system-level design, which satisfies the various required constraints and an optimization method of image encoder based on codesign of function, algorithm, and architecture. In addition, we show an MPEG-4 encoder as a design case study. The best tradeoffs between algorithm and architecture are necessary to deliver the design with satisfying performance and area constraints. The evaluations provide the effective optimization of motion estimation, which is in charge of an amount of performance in the MPEG-4 encoder module.

1. Introduction

The system design process has become an increasingly difficult problem due to the increasing design complexity, system integration and throughput. At the same time, the market requires more constraints such as a short life cycle of a new product, time-to-market, lower cost, higher throughput, and lower power consumption. To meet these needs, the codesign can be one of the solutions.

The codesign approach is concurrent design of hardware and software to be tightly coupled throughout the design process. The codesign techniques take advantage of design flexibility to create systems that can meet stringent performance requirements with improves design quality, design cycle time, and reduce cost, integration and test time, and bugs.

In recent year, with the increasing system design complexity, System-level codesign will concentrate on the embedded systems including processors, memories, DSP chips, custom hardware or IP components, and custom complete system-on-a-chip.

The successful system-level codesign must address several issues; system-level specification, analysis, partitioning the system into either hardware or software, scheduling the execution of the system's tasks to meet ant any constraints, co-verification etc.

In this paper, we present a design methodology that is a system-level codesign of function and architecture, and algorithms for optimization of image encoder. We describe system design procedure that takes the MPEG encoder processing algorithms specified and provide to perform algorithm and architecture estimation, make tradeoffs based on information from evaluations, and optimize under performance constraints.

This methodology is based on evaluating quantitative computational complexity-area-performance information about the algorithm, functions, and architecture, then combining the information to guide architecture.

2. System-level Codesign

2.1 Previous work

Previous work in hardware-software codesign has been reported in various papers. Polis[1] is described as a network of Codesign Finite State Machines (CFSMs). This model is very well suited for reactive systems, but less suited for signal processing applications involving DSP kernel.

In the Ptolemy [2], heterogeneous modeling, simulation, and design of concurrent systems are studied, with a focus on embedded systems. Although Ptolemy offers some support for code generation, it does not support an explicit mapping of application models onto models of architecture.

Chinook [3] supports an explicit mapping from a behavioral description onto a target architecture and again, a simulation tool is used to simulate the system at different levels of abstraction.

El Greco [4] provides a simulation environment for modeling and validating the functionality of complex heterogeneous systems.

In Y-chart scheme [5], the designer first characterizes a set of applications and chooses an architecture to run that set. Then, each application is mapped onto the architecture and the performance of each application-architecture mapping is evaluated. Depending on the resulting performance numbers, the designer may decide to use that architecture, restructure the application, or modify the mapping of the application to get better performance.

Some codesign approaches like [6], [7], and [8] focus on interactive partitioning, performance estimation, cosimulation, communication synthesis, and code generation. Automated partitioning has been under intense research [6] [8] [9] [10].

Y-chart [11] defined a general scheme for the design of programmable architectures. Target applications are mapped onto the architecture, and their performance is analyzed to obtain performance numbers.

After analysis, the architecture or application is manually tuned by the designer and the process is repeated until a desired system is obtained.

2.2 Motivations

The system-level design is often perceived as a process going from functional specifications of a system, through refinement or optimization steps into an architecture and from there on to a final implementation.

The opportunity to consider tradeoffs in function and architecture performance occurs too late in the design flow for any changes to be made in a timely or cost-effective manner.

The hardware and software design flow can be brought together and pursued in parallel at the earliest point in the design process. The system-level designer has to identifies what a system does, and an architecture model, which identifies how the system is implemented.

The choice of the algorithm for performing a function affects system performance, chip area, and power consumption. We need to analysis to determine whether the functional design satisfies the specification, the mapping of this design to a candidate architecture.

We believe that the key aspect of system design is indeed function-algorithm-architecture codesign.

3. Design flow

The basic of our design methodology is shown in Figure 1. Starting from system specifications and requirements, this step is of course very important and is always necessary very kind of system to develop.

The methodology secondly describes function that the system is intended to provide or a model of the application from design specifications, and then a candidate architecture is selected, which has the potential of realizing the function, including choices of IP blocks in both hardware and software domains.

Algorithm selection assumes the existence of a multiple algorithms for computation of some common functions. We can select an algorithm that maximizes performance while satisfying performance constraints.

The algorithm is then mapped to the architecture, during which time all required hardware-hardware, hardware-software, and software-software partitioning is carried out. Both functional processing is mapped to architectural resources.

During an analysis phase, performance and other studies of the suitability of the architecture to realize the function are carried out, and modifications to the architectural structure or the system functionality are made until the design is brought into a feasible design space.

Following this, communications detail is added to what starts out as a very abstract model of inter-function communications, refining it down to the generic bus transaction level and abstract communications protocol patterns, with specific characteristics of the chosen system buses modeled.

Finally, hardware-software coverification, and implementation for both hardware and software portions of the system are carried out.

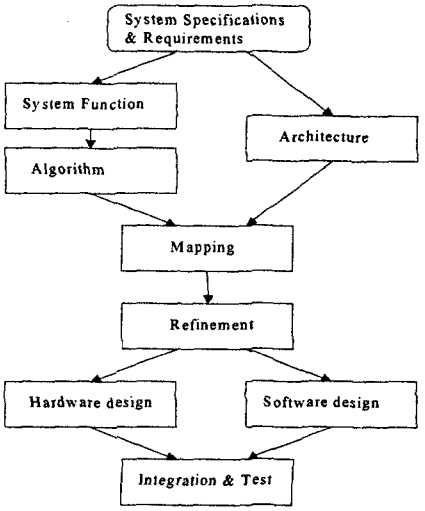

Fig.1 Hardware and software codesign flow

3.1 Function and algorithm

In usually, a system implements a set of functions, where sets of functions are the input and output characterization of the system. The description of function the system has to implement is captured using a particular language that may or may not be formal such as dynamic and static data flows, FSMs for control, SDL models, and C/C++ models. However, these languages are often application dependent and lack the accuracy description of function.

The functional model of an application can be also formed by composition of a variety of models of computation. The system architecture takes the decision on algorithms, which is now to obtain the specified functionality, hardware architectures, which is what is the hardware support required for implementing the functionality with the select algorithm.

3.2 Architecture

The architecture is a set of interconnected components, with a physical dimension that is used to implement a function. In general, a component is an element with explicit context dependency and specified interfaces.

The architecture determines the final hardware implementation and hence it is strictly related to the concept of platform. The architecture for the majority of system designs consists of processors, dedicated hardware blocks, peripherals, and memories.

3.3 Mapping algorithm to architecture

The mapping of algorithm to architecture is an essential step from the specification to the implementation. The mapping is process, where the functions to be implemented is assigned to the components of the architecture.

Mapping functions onto processor resources implies implementation via software tasks running on control processors and DSPs, whereas mappings onto hardware blocks imply hardware-based implementation via custom, ASIC, or reconfigurable logic.

Function-architecture codesign thus permits a wide range of architectural exploration.

When the mapping step is carried out, our choice is dictated by estimates of the performance of the implementation of that algorithm onto the architecture component.

4. Case Study

We described the system design methodology in the previous sections. In this section, we describe an MPEG-4 encoder as a case study.

4.1 MPEG-4 encoder

MPEG-4 improves upon the previously successful MPEGs standards in the areas related to internet streaming, bandwidth scalability, interactivity, and coding efficiency [12]. Compared with H.263 or MPEG-2, arithmetic complexity of MPEG-4 encoders will increase significantly.
This is mainly caused by the overhead introduced for the handling and encoding of multiple video objects.

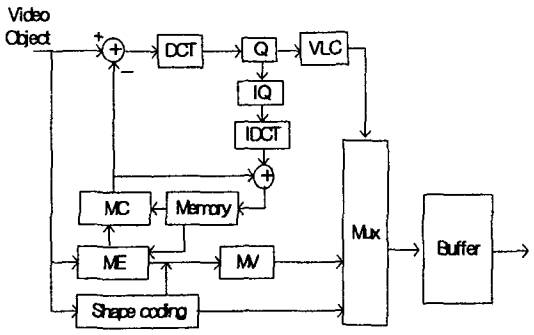

Fig.2 MPEG-4 encoding scheme

Figure 2 describes the main functions and operation of MPEG-4 encoder. The basic video encoding method of MPEG-4 utilizes motion estimation and prediction, DCT/IDCT, Quantization/Inverse quantization, VLC, and others.

Usually the motion estimation consumes most of the time required by encoding. The DCT/IDCT unit performs discrete cosine transform during coding and its inverse transform during reconstruction. In addition, the unit performs quantization (Q) and inverse-quantization (IQ) for the DCT coefficients.

The MPEG-4 encoding includes several compute intensive algorithms such as discrete cosine transform (DCT) and motion estimation(ME).

4.2 Motion estimation

The motion estimation is a technique to eliminate temporal redundancy of image sequences and usually it has most computational complexity (about 60-80% of total computation time), and affects image encoding systems performance greatly for a given bit rate.

The motion estimation for extract motion vector is not standardized in the digital image standardizations. The requirements of applications differ significantly for motion estimation algorithms. So improving its functionality can optimize the whole system a great deal. The hardware or software implementation may be selected for the motion estimation in one application.

The motion estimation algorithms usually have relatively simple arithmetic operations, but require a high memory bandwidth. Therefore, computational complexity and quality are optimized of motion estimation at algorithm and architecture level.

We need to effect of interactions between tradeoffs within these parameters. We implemented estimation algorithms by software and designed an architecture it.

5. Implementation and Experimentation

5.1 Specifications

To implementation MPEG-4 encoder, we assume target design specifications, listed in Table 1. We can decompose several function blocks from specification as figure 1.

These function blocks were described by behavior adaptation, and communication interface among modules.

Table 1 MPEG-4 Encoder design specifications

MPEG-4 Simple Profile Levels 0,1,2 and 3
Resolution up to CIF encoding
at 30 frames per second
4:2:0 Ycbcr input
DCT and IDCT compatible with IEEE Std 1180-1990
Quantization and Inverse Quantization (H.263)
Zigzag-SCAN
Run length coding (RLC)
VLC and RVLC
ME (Motion estimation)
• Full search or three-step search strategy
• Range of motion vectors 16 pixels
• Half pixel resolution
• 4 MV / macro block

5.2 Target Architecture

The architecture is to determine type of programmable processors, the memory capacity, the local bus type, hardware modules, and software or operating system, which optimally meet design specification and performance.

The hardware architecture should allow for variety of processing elements and support variety images encoder. Furthermore, there is a need to accommodate scalability for future expansions and for efficiency.

Figure 3 shown our target MPEGs encoder architecture consisting based on a programmable processor(ARM processor), memories, and some hardware modules with local memory.

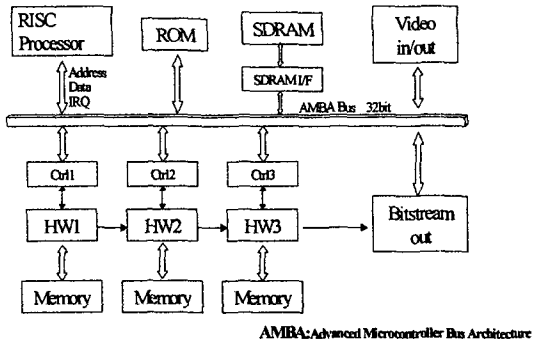

Fig.3 MPEG-4 target system

The processor and hardware modules(HW1, HW2, HW3) connected to on ARM local bus using main memory. Each hardware module has local memory and control unit(Ctrl1/2/3), which interface to ARM local bus.

5.3 Implementation

We evaluate four kinds of motion estimation algorithms; FS(full search), TSS(Three Step Search), FSS(Four Step Search), NTSS(New Three Step Search) and implement them by software, and analyze computational complexity, image quality, and CPU speed based on software.

The comparison in terms of computational complexity is based on execution time and count instructions. And, we evaluate image quality in terms of PSNR(peak-signal-noise rate). Table 2 listed average PSNR, CPU speed, and total number of arithmetic.

We selected one algorithm, which was the TSS among evaluated motion estimation algorithms, and the TSS was mapping to architecture(Figure 4). Table 2 summarized hardware implementation results for the TSS algorithm.

There are many solutions of motion estimation architectures and approach hardware design. In general, architecture of the motion estimation consists of a PE(processing element) array, a memory and a memory controller, and an MV(motion vector) detector module.

The PE array is major part of motion estimation, thus it needs to tradeoffs and optimize in terms of parallelism, memory bandwidth, control cost, and computational cost. We implement these PE array using VHDL and SYSNOPSYS design compiler tools. The PE array presents combination the PE's structure and hardware comparison. For number of PE element, we analyze chip area, number of cycles, and memory bandwidth for each other in the Table 3.

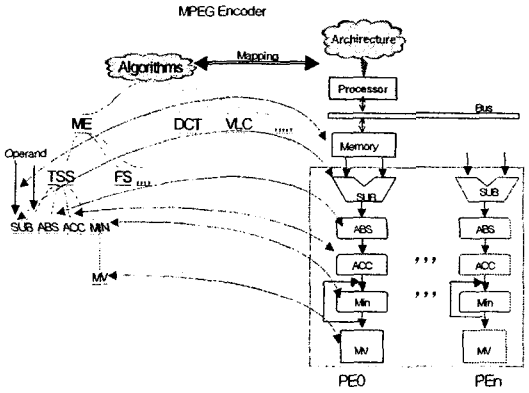

Fig.4 Algorithm to architecture mapping.

Table 2 the average PSNR, arithmetic, and speed comparisons (The speed runs on an Ultrasparc processor 200Mhz/128MB)

Images	Algorithms	Average PSNR	Speed (sec)	Arithmetic
Football (CIF, 80frame)	FS	62.89	221.66	5765
	TSS	61.24	10.42	893
	NTSS	61.04	7.1	977
	FSS	61.23	8.89	1106

Table 3 hardware comparisons for TSS

No. PEs	No. Cycles	Memory Bandwidth	Chip area (hitachi 0.35μm technology)
3	2304	3072	0.21262 mm^2
9	768	7680	0.64268mm^2
18	256	9728	1.99284 mm^2
64	64	16640	6.05824 mm^2

6. Conclusions

In this paper, we introduced an implementation method of MPEG-4 encoder based on codesign of function, algorithm, and architecture. The best tradeoffs between each module are necessary to deliver the design satisfying performance and area constraints. With evaluations and measurements at an early stage of the design process, we also optimized motion estimation for MPEG-4 encoder with respect to maximal performance and minimal area

References

[1] F. Balarin, E. Sentovich, M Chiodo, P. Giusto, H. Hsieh,B Tabbara, A. Jurecska, L. Lavagno, C. Passerone, K. Suzuki, and A. Sangiovanni-Vincentelli, *Hardware-Software Co-design of Embedded Systems – The POLIS approach*, Kluwer Academic Publishers, 1997.

[2] Edward A. Lee et al., "Overview of the Ptolemy project,"ERL Technical Report UCB/ERL M99/37, University of California, Berkeley, July 1999.

[3] P. Chou, R.B. Ortega, and G. Boriello, "The Chinook Hardware/Software Co-Synthesis System," *Proc. Intl. Symposium on System Synthesis*, 1995.

[4] J. Buck, R. Vaidyanath, "Heterogeneous Modeling and Simulation of Embedded Systems in El Greco," *Proc. CODES*,March 2000.

[5] P. Lieverse, P. Van der Wolf, E. Deprettere, and K. Vissers, "A Methodology for Architecture Exploration of Heterogeneous Signal Processing Systems," to appear in *Journal of VLSI Signal Processing.*

[6] Balarin, F., Chiodo, M., Giusto, P., Hsieh, H., Jurecska, A., Lavagno,L., Passerone, C., Sangiovanno-Vincentelli, A., Sentovich, E., Suzuki,K., Tabbara, B.,*Hardware-Software Co-Design of Embedded Systems – The POLIS Approach*, Kluwer, Norwell, USA, 1997.

[7] Madsen, J., Grode, J., Knudsen, P., Petersen, M., Haxthausen, A.,*LYCOS: The Lyngby Co-Synthesis System, Design Automation for Embedded Systems*, Volume 2, Issue 2, Kluwer, Norwell, USA, 1997.

[8] Gajski, D., Vahid, F., Narayan, S., Gong, J.,"SpecSyn: An Environment Supporting the Specify-Explore-Refine Paradigm for Hardware/Software System Design," *IEEE Trans. on VLSI Systems*, Vol. 6, No.1, pp. 84-100, 1998.

[9] Vahid, F.,"Modifying Min-Cut for Hardware and Software Functional Partitioning," 5th International Workshop on *Hardware/Software Codesign*,March24-26, Braunschweig,1997

[10] Hollstein, T., Becker, J., Kirschbaum, A., Glesner, M.,"HiPART: A New Hierarchical Semi-Interactive HW/SW Partitioning Approach with Fast Debugging for Real-Time Embedded Systems," 6th International Workshop on *HW/SW Codesign*, March 15-18, Seattle,1998.

[11] P. Lieverse, P. van der Wolf, E. Deprettere, K. Vissers.,"A Methodology for Architecture Exploration of Heterogeneous Signal Processing Systems," *IEEE Workshop on Signal Processing Systems*, Taipei, October 1999.

[12] Overview of the MPEG-4 Standard (ISO/IEC JTC1/SC29/WG11N3747). http://www.cselt.it/mpeg/standards/mpeg-4/mpeg-4.html.