Software-Cooperative Power-Efficient Heterogeneous Multi-Core for Media Processing

Hiroaki Shikano[†,††], Masaki Ito[†], Kunio Uchiyama[†], Toshihiko Odaka[†], Akihiro Hayashi[††], Takeshi Masuura[††], Masayoshi Mase[††], Jun Shirako[††], Yasutaka Wada[††], Keiji Kimura[††] and Hironori Kasahara[††]

†Hitachi, Ltd., Tokyo, Japan
††Dept. of Computer Science, Waseda University, Tokyo, Japan
E-mail: hiroaki.shikano.gm@hitachi.com

Abstract— A heterogeneous multi-core processor (HMCP) architecture, which integrates general purpose processors (CPU) and accelerators (ACC) to achieve high-performance as well as low-power consumption with the support of a parallelizing compiler, was developed. The evaluation was performed using an MP3 audio encoder on a simulator that accurately models the HMCP. It showed that 16-frame encoding on the HMCP with four CPUs and four ACCs yielded 24.5-fold speed-up of performance against sequential execution on one CPU. Furthermore, power saving by the compiler reduced energy consumption of the encoding to 0.17 J, namely, by 28.4%.

I. Introduction

Advancement of semiconductor manufacturing technology no longer offers performance improvement because of power limitations. Consequently, chip multiprocessors (CMPs) have been attracting much attention. Recently, multi-core processor products such as Fujitsu's FR-V [1], ARM's MPCore [2], and IBM, Sony, Toshiba's Cell [3] were released onto the market. However, getting full performance from a CMP is troublesome. This is because ordinal programs are composed in sequential order, and therefore the programs should be carefully examined, parallelized and tuned up for efficient use of the CMP. The architecture should therefore be easily hand-tunable and supportable by optimization tools such as automatic parallelizing compilers [4, 5]. Furthermore, processors for embedded systems now demand even higher performance and further lower power consumption.

To cope with these demands, the authors have developed a heterogeneous multi-core processor (HMCP) architecture. The architecture possesses different types of processor cores, including general-purpose processors and special-purpose processors that accelerate specific types of processes. It supports heterogeneous parallel processing with support of the Optimally Scheduled Advanced Multiprocessor (OSCAR) multigrain parallelizing compiler [6, 7]; thus, operating frequency can be lowered while maintaining high performance. It also features a hierarchical memory architecture, data transfer units, and power-control registers, which can be utilized for software optimization (in terms of both power and performance) by the compiler.

This paper describes the heterogeneous multi-core processor (HMCP) architecture, which supports effective parallelized software executions, and presents evaluation results of the architecture incorporated in a fabricated test chip and a simulator using AAC and MP3 audio encoders. The paper is organized as follows. The power-performance optimized HMCP architecture is described in Section II. A parallelizing compiler that divides a program into parallelized tasks and schedules them onto processor cores in the compiling stage is explained in Section III. Evaluation conditions and results of AAC and MP3 encoders on the HMCP are presented in Section IV.

II. Power-efficient multi-core architecture

The proposed heterogeneous multi-core processor (HMCP) architecture is described. Performance of the HMCP is maximized with software supports including parallelized program execution, memory management, and power control. In other words, the goal of the architecture is to maximize performance by utilizing heterogeneous processor cores and by reducing execution overhead of parallelized programs, such as memory access time, synchronization time, and data transfer time, as well as to reduce power consumption by fine-grained software power control.

The HMCP architecture integrates multiple general-purpose processors as well as special-purpose processors, such as dynamically reconfigurable processors (DRP) or digital signal processors (DSP), that process a specific type of program very efficiently, as described in Figure 1. These processor cores are connected to each other with an interconnection network. The architecture has an on-chip centralized shared memory (CSM) on a chip, and various types of high-speed local memories in every processor core.

A. Hierarchical memory architecture

The architecture is equipped with high-speed local memories, including a local data memory (LDM), a distributed shared data memory (DSM), and a local program memory (LPM) in every processor core. It also possesses a centralized shared memory (CSM) on a chip or off the chip. All the memories are mapped within a global address space, and their coherence is managed by software. LDM stores private data accessed by a nearby processor core. DSM preserves data shared with other processor cores for inter-core communication and synchronization. LPM stores a program for the nearby processor core.

978-1-4244-1922-7/08/$25.00 ©2008 IEEE

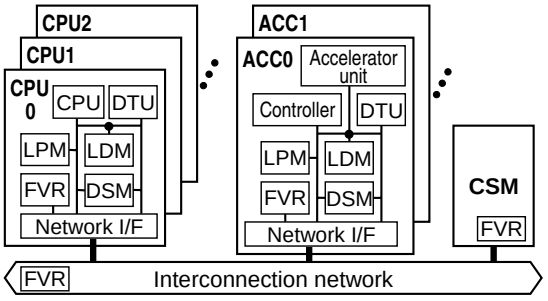


Fig. 1. Heterogeneous multi-core processor architecture

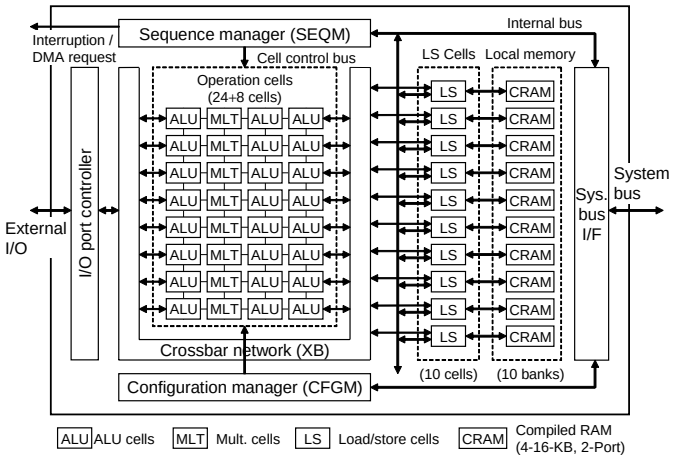


Fig. 2. Block diagram of FE-GA

CSM is used for storage of data shared with all of the processor cores. Utilization of these memories restrains accesses to low-speed off-chip memory; thus, operational performance of the processor is maximized.

B. Data-transfer unit

Each processor core has a data-transfer unit (DTU) attached to an internal bus connected to the local memories. The DTU transfers data between DSMs on different processor cores, between LDM and on- or off-chip CSM, or between on-chip CSM and off-chip CSM behind task executions on processor cores simultaneously. It is also equipped with flag-set and flag-check commands. In flag-set mode, the DTU sets a flag with a number specified in a command. In flag-check mode, the DTU reads a value of a flag and checks its correspondence with the number specified in the command. These commands can be programmed and stored on a memory as a linked list. Different types of transfers can therefore be defined in advance; thus, DTUs can operate independently behind CPUs. Furthermore, the frequency of DTU operations by the CPU is reduced, even if multiple types of data transfers are performed at the same time.

C. Power-control registers

The architecture is equipped with a frequency/voltage control register (FVR). It selects clock frequency and accordingly determined power-supply voltage or power off to functional units of PEs such as processor cores, local memories and a data transfer unit (DTU). Other components such as CSM, an interconnection network, and I/O units are also furnished with an FVR for determining their power mode. The FVR is addressed globally so that any processor cores can set up the power mode of a target unit. Implementation of FVR depends on hardware organizations such as a process technology, functions of the controlled units, etc.

D. Flexible engine/generic ALU array

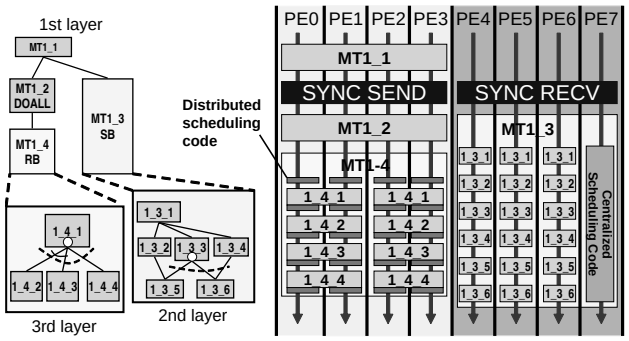
The evaluated HMCP adopts an FE-GA (flexible engine/generic ALU array) [8] which is a type of dynamically reconfigurable processor, as an accelerator. Figure 2 shows the architecture of the FE-GA, consisting of an operation block and a control block. The operation block is composed of two-dimensionally arrayed arithmetic logic unit (ALU) cells, whose functions and connections to neighboring cells are dynamically changeable. It also has multiple banks of a local memory (CRAM) for data storage and a crossbar (XB) network supporting internal data transfer between load/store (LS) cells and CRAMs. The control block consists of a configuration manager (which manages the configuration data for the operation block) and a sequence manager (SEQM) (which controls the state of the operation block). FE-GA is highly optimized in terms of power and performance in media processing for embedded systems.

III. OSCAR PARALLELIZING COMPILER

In multi-core systems, programs should be parallelized to attain their full performance. Accordingly, we developed an OSCAR (optimally scheduled advanced multiprocessor) parallelizing compiler [6, 7] that extracts parallelism from a program and schedules parallelized parts of the program automatically. Furthermore, it also controls power modes (i.e., clock frequency and power supply voltage) of each processor core by utilizing its parallelized scheduling result at a compiling stage.

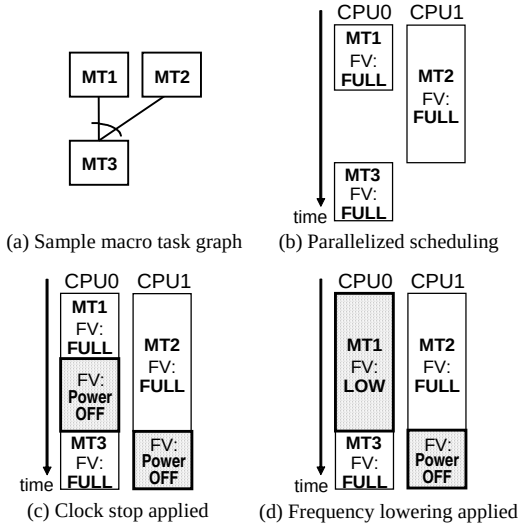
A. Multi-grain parallel processing

Multi-grain parallelization utilizes three types of structural parallelism among coarse grain tasks, such as loops, subroutines, and basic blocks [6]. A program is decomposed into macro tasks, or coarse-grain tasks, such as repetition blocks (RBs) or loops, subroutine calls (SBs), and blocks of pseudo assignment statements (BPAs) or fused basic blocks. Macro tasks such as SBs and RBs may include other macro tasks inside; therefore, tasks are hierarchically defined. Control flow and data dependencies among tasks are then analyzed and earliest executable condition analysis is performed. As a result, a macro task graph (MTG), as shown in Figure 3(a), which represents coarse-grain task parallelism, is generated. After generating a MTG, tasks on the MTG are scheduled onto processors. At this time, if no undetermined conditions, such as conditional branches exist in the MTG, the compiler adopts static scheduling that assigns the tasks onto the processors statically at compiling time. The static scheduling optimizes synchronizations and data transfers at compiling time; therefore, scheduling overhead at runtime is minimized. In case undeter-



(a) Sample macro task graph (b) Scheduling result

Fig. 3. Multi-grain parallelization scheme of the OSCAR compiler



(a) Sample macro task graph (b) Parallelized scheduling

(c) Clock stop applied (d) Frequency lowering applied

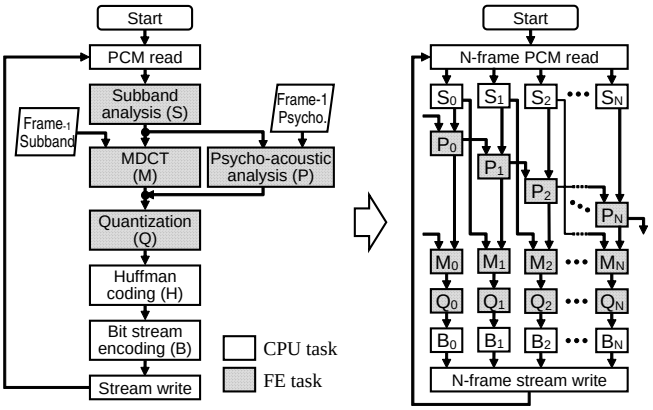
Fig. 4. Basic concept of power saving scheme

mined conditions exist, the compiler adopts dynamic scheduling, which generates a task-scheduler code managing run-time task scheduling. Figure 3(b) shows the scheduling result of the MTG described in Figure 3(a), whose macro tasks are assigned to eight hierarchically grouped processors.

Especially for HMCPs with different types of processor core, the compiler should consider processor types suitable for each macro task. The target processor types are designated with directives in a program by users, and the compiler utilizes them. At the scheduling stage in compiling, the compiler estimates finishing time of each macro task in the case that it is processed either on a CPU or on a designated accelerator. It then chooses a processor that finishes processing of the task earlier [9].

B. Compiler control power saving

Multi-grain parallel processing exploits parallelism. However, not all the processor cores are always in operation simultaneously. This is because some parts of a program do not have enough parallelism that exploits all the processor resources due to data dependency or control dependency among tasks. Figure 4 shows a sample macro task graph (a) and its scheduling result on two PEs (b). Because of data dependency among the three tasks, CPU0 is in idle mode between MT1 and MT3 when MT1's execution cost is smaller than MT2's. The compiler generates power-control codes, setting up an FVR such



(a) Processing flow of MP3 encoder (b) N-frame parallelized structure

Fig. 5. Processing flow of MP3 encoder

that clock and power supply are shut off while CPU0 is in idle mode, Figure 4(c), or clock frequency and supply voltage of CPU0 are lowered during MT1 execution, Figure 4(d). The selection of FV modes for MT1 in Figure 4(d) is performed by calculating CPU0 idling time after MT1 to the next dependent MT3 determined by MT2 execution cost estimated by the compiler. In this way, to optimize total power consumption while maintaining performance enhancement by parallelization, the compiler applies power-saving control and then inserts an FVR accessing code among tasks. Two modes for the power-saving schemes by the compiler are defined [10]. One is the fastest execution mode, which applies a power-saving scheme to tasks, except for tasks on a critical path of a program. This mode guarantees the fastest execution time, since the execution of tasks on a critical path is not retarded by the power control. The other mode is real-time processing with a deadline constraint. This mode minimizes total power consumption within a given deadline.

IV. Evaluation

In the evaluation of the HMCP architecture, MP3 audio encoder is applied. We implemented MP3 encoders on the HMCP architecture simulator that accurately models the architecture described in Section II. Task scheduling and low-power control of MP3 encoder is performed by the compiler described in Section III.

A. MP3 encoder

MP3 encoding is used as an audio-compression technique. The process of the encoding consists of sub-band analysis, psycho-acoustic analysis, modified discrete cosine transform (MDCT), quantization, Huffman coding, and bit-stream encoding as shown in Figure 5(a). We used an UZURA MP3 encoder [11], whose implementation abides by the MP3 encoding standard, as a target encoding program. Profiling of the program shows that sub-band analysis and quantization shares almost 90% of the total encoding time. Sub-band analysis, MDCT, quantization, and psycho-acoustic analysis were designated as FE-GA tasks.

For automatic parallelization of the program, we modified the structure of the program. The original program has multi-

TABLE I
SIMULATION PARAMETERS

LDM latency	1 cycles
Remote DSM latency	4 cycles
CSM latency	16 cycles
Network arbitration	2 cycles
Frequency transition overhead	30,000 cycles
Power transition overhead	60,000 cycles
Operation frequency (CPU, FE and bus)	300 MHz
Power supply voltage	1.0 V

TABLE II
POWER CONTROLLING MODE

State	FULL	MID	LOW	OFF
Frequency	1	1/2	1/4	0
Voltage	1	0.87	0.71	0
Dynamic energy	1	3/4	1/2	0
Static energy	1	1	1	0

[Normalized value to FULL mode]

ple processing stages in a loop that corresponds to the number of the input audio frames as shown Figure 5(a). We utilized inter-frame parallelism so that a unit of the parallelized frames is grouped and looped up as shown in Figure 5(b). In the evaluation, the number of parallelized units was set as 16.

B. HMCP simulator

Evaluation of HMCP with an MP3 encoder was performed by utilizing an HMCP architectural simulator for accurately modeling the HMCP described in Figure 1. The number and types of processor cores are variable, with arbitrary memory size and latency. The CPU core is modeled as an SH-4A processor [12]. For an accelerator core, we adopted a simple implementation as a pseudo-core. The functions of such a core were implemented solely for MP3 encoding in advance. The accelerator function ID was inserted in an input program, and the compiler converts the specified part of the program into a code that calls the determined function on a pseudo-core. The core then starts calculation of data placed on the LDM and returns an execution result on the LDM when a specified number of execution cycles for the function elapses. We developed FE-GA programs for sub-band analysis, psycho-acoustic analysis, MDCT, and quantization, and we evaluated execution cycles on an FE-GA cycle-accurate simulator introduced to the pseudo-core simulation module.

A power calculation is implemented in the simulator. For the CPUs, a power model is based on Wattch [13], and Wattch parameters are determined from RTL-based power simulation on an SH-4A processor core. For FE-GAs, average power consumption of each programmed function is calculated by multiplying utilization rate of the operation cells and the maximum power consumption which was evaluated on an FE-GA evaluation chip when all the cells were in operation. Power-control registers are also implemented in the simulator. The register controls clock frequency and power-supply voltage of the processor cores. Power-control modes of CPU and FE-GA cores are presented in Table II. In the evaluation, the power mode of FE-GA is limited to either FULL or OFF because accelerators should execute specific parts of the program as fast as possible.

TABLE III
IMPROVED PERFORMANCE BY FE-GA ON ONE-FRAME ENCODING

Process	On CPU	On FE-GA	Speed-up
Sub-band analysis	21,942 Kcyc	305 Kcyc	71.9x
Psycho-acoustic analysis	464 Kcyc	4 Kcyc	108.2x
MDCT	2,852 Kcyc	48 Kcyc	59.7x
Quantization	39,954 Kcyc	9,044 Kcyc	4.4x
Total	65,203 Kcyc	9,402 Kcyc	6.9x

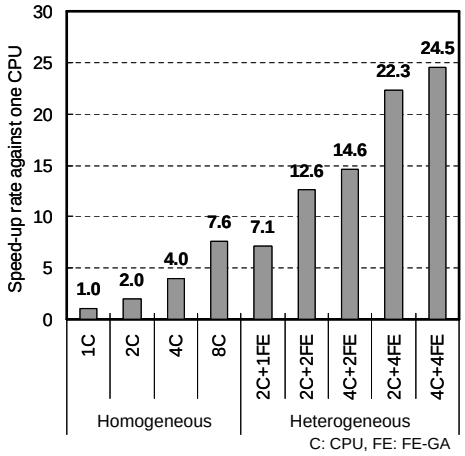


Fig. 6. Performance improvement of MP3 encoder

C. Improved performance with FE-GAs

We developed the FE-GA programs for the sub-band analysis, psycho-acoustic analysis, MDCT, and quantization. Execution cycles of one-frame encoding were measured both on an FE-GA and on a single CPU by simulator, as shown in Table III. Introducing FE-GAs to the above four processes yielded 71.9-, 108.2-, 59.7-, and 4.4-fold speed-up in performance compared to processing speed with sequential execution on a CPU respectively.

D. Performance evaluation

Figure 6 shows speed-up ratio against sequential execution on one CPU when sixteen-frame encoding is performed under various processor configurations. For homogeneous multi-cores, the speed-up rate is proportional to the number of CPUs supplied. When four CPUs and eight CPUs are used, the speed-ups are 3.97 and 7.61, respectively. For heterogeneous configurations, when two CPUs and one FE-GA are utilized, the speed-up is 7.13. As more processor cores are supplied, that is, two CPUs and two FE-GAs, four CPUs and two FE-GAs, two CPUs and four FE-GAs, and four CPUs and four FE-GAs, the speed-ups reach 12.57, 14.62, 22.31, and 24.48 respectively. Introducing FE-GA is a key for improving performance and the speed-up rate is highly improved as the number of FE-GAs increases. This is because quantization, which takes up much of the encoding process, is assigned and efficiently processed on FE-GAs. Figure 7 shows a Gantt chart of the execution on two CPUs and two DRPs. The chart shows that tasks from MT187 to MT201 are quantization, which is assigned to the two FE-GAs.

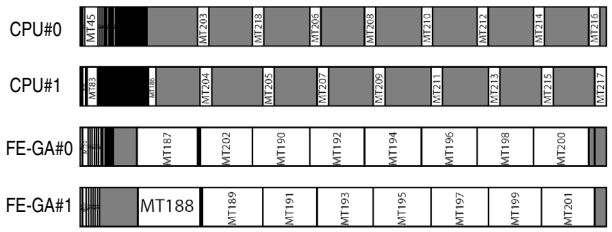


Fig. 7. Trace Gantt chart of MP3 16-frame encoding on CPU×2+FE×2

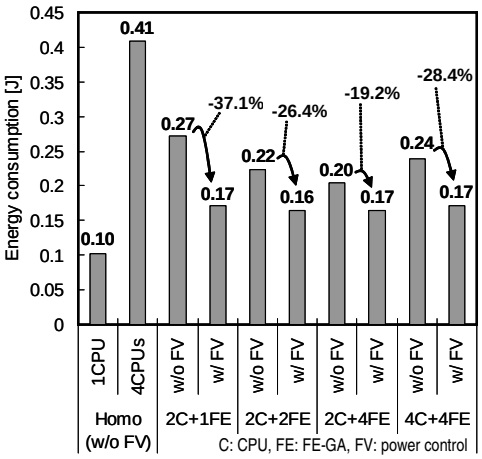


Fig. 8. Simulated energy consumption of MP3 encoding

E. Power evaluation

The power-controlling scheme described in Section III is applied to MP3 encoding, and the energy of the encoding process is evaluated using a simulator. The fastest controlling mode is adopted with use of the parallelized scheduling by the compiler. Figure 8 shows the energy consumption with various configurations of processor cores evaluated on the simulator. In the case of two CPUs and one FE-GA, the energy with the power control applied is 0.17 J, which is reduced by 37.1% from the energy in the case without control. When the number of the processor cores increase, namely, to two CPUs and two FE-GAs, two CPUs and four FE-GAs, and four CPUs and four FE-GAs, the energies in the case that power control is applied are 0.17 J, 0.16 J, 0.17 J, and 0.17 J respectively, which are reduced by 26.4%, 19.2%, and 28.4%, respectively, from those values in the case without the power control.

The graph shows that processing on two CPUs and four FE-GAs is the most efficient in energy terms in the case without the power control. However, energy consumption when power control is applied under various PE configurations is almost the same. This is due to energy cut-off by the control during the idling time of the processors. Figure 9 shows a Gantt chart of the encoding on two CPUs and two FE-GAs when the power control is applied. The power control is applied to CPU tasks from MT203 to MT218, which are bit-stream encoding processes. The execution time of these tasks is shorter than quantization time, which is executed in parallel on FE-Gas, as shown in Figure 7, without power control. Moreover, Figure 9 (the case with the power control) shows idling time of the CPUs is reduced in bit-stream encoding processes (which are executed more slowly by the power control). The overhead of the power control over the processing time is increase by about 0.5% compared to executions without the power control.

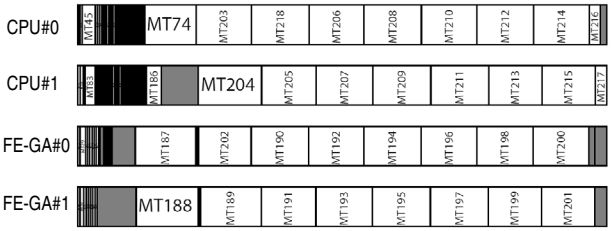


Fig. 9. Trace Gantt chart of MP3 16-frame encoding on CPU×2+FE×2 with power control applied

V. Conclusions

A heterogeneous multi-core processor (HMCP) architecture, which integrates general-purpose processors (CPUs) and accelerators (ACCs), was developed. HMCP achieves both high-performance and low-power for embedded systems by exploiting accelerators and efficient parallel processing with support of a parallelizing compiler. To evaluate the HMCP architecture, MP3 encoding is implemented on an HMCP simulator with multiple CPU cores and DRP cores with power control in conjunction with a parallelizing compiler. The evaluation of MP3 encoding shows that compared to sequential execution on one CPU, four CPUs and four DRPs with the proposed architecture speed-up the execution by 24.5 times. The energy of the encoding is 0.17 J with the power control of the fastest execution mode applied by the compiler, which is reduced by 28.4% from that without the control.

Acknowledgements

A part of this research has been supported by NEDO “Advanced Heterogeneous Multiprocessor”. and STARC “Automatic Parallelizing Compiler Cooperative Single Chip Multiprocessor”.

References

[1] T. Shiota, K. Kawasaki, Y. Kawabe, W. Shibamoto, A. Sato, T. Hashimoto, F. Hayakawa, S. Tago, H. Okano, Y. Nakamura, H. Miyake, A. Suga, and H. Takahashi, “A 51.2gops 1.0gb/s-dma single-chip multi-processor integrating quadruple 8-way vliw processors,” in *Proc. of IEEE Internatilnal Solid-State Ciruits Conference (ISSCC2005)*, Feb. 2005.

[2] J. Cornish, “Balanced energy optimization,” in *Proc. of the 2004 International Symposium on Low Power Electronics and Design (ISLPED ’04)*, Aug. 2004.

[3] D. Pham, S. Asano, M. Bolliger, M. N. Day, H. P. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, M. Rilley, D. Shippy, D. Stasiak, M. Suzuoki, M. Wang, J. Warnock, S. Weitzel, D. Wendel, T. Yamazaki, and K. Yazawa, “The design and implementation of a first-generation cell processor,” in *Proc. of the IEEE International Solid-State Circuits Conference (ISSCC 2005)*, Feb. 2005.

[4] R. Eigenmann, J. Hoeflinger, and D. Padua, “On the automatic parallelization of the perfect benchmarks,” *IEEE Trans. on parallel and distributed systems*, vol. 9, no. 1, Jan. 1998.

[5] M. W. Hall, J. M. Anderson, S. P. Amarasinghe, B. R. Murphy, S. Liao, E. Bugnion, and M. S. Lam, “Maximizing multiprocessor performance with the SUIF compiler,” *IEEE Computer*, 1996.

[6] H. Kasahara, H. Honda, A. Mogi, A. Ogura, K. Fujiwara, and S. Narita, “A multi-grain parallelizing compilation scheme on oscar (optimally scheduled advanced multiprocessor),” *Proc. of the 4th Workshop on Language and Compilers for Parallel Computing*, Aug. 1991.

[7] K. Kimura, Y. Wada, H. Nakano, T. Kodaka, J. Shirako, K. Ishizaka, and H. Kasahara, “Multigrain parallel processing on compiler cooperative chip multiprocessor,” in *Proc. of the 9th Workshop on Interaction between Compilers and Computer Architectures (INTERACT-9)*, Feb. 2005.

[8] T. Kodama, T. Tsunoda, M. Takada, H. Tanaka, Y. Akita, M. Sato, and M. Ito, “Flexible engine: A dynamic reconfigurable accelerator with high performance and low power comsumption,” in *Proc. of IEEE Symposium on Low-Power and High-Speed Chips (COOL Chips IX)*, 2006.

[9] Y. Wada, A. Hayashi, T. Iyoku, T. Masuura, J. shirako, H. Nakano, H. Shikano, K. Kimura, and H. Kasahara, “A hierarchical coarse grain task static scheduling scheme on a heterogeneous multicore,” *Technical Report of IPSJ, 2007-ARC-174-17(SWoPP2007) (in Japanese)*, Aug. 2007.

[10] J. Shirako, N. Oshiyama, Y. Wada, H. Shikano, K. Kimura, and H. Kasahara, “Compiler control power saving scheme for multi core processors,” in *Proc. of the 18th International Workshop on Languages and Compilers for Parallel Computing(LCPC2005)*, Oct. 2005.

[11] UZURA3, “MPEG-1 / Layer-3 encoder in FORTRAN90,” http://members.at.infoseek.co.jp/kitaurawa/index_e.html.

[12] F. Arakawa, T. Yoshinaga, T. Hayashi, Y. Kiyoshige, T. Okada, M. Nishibori, T. Hiraoka, M. Ozawa, T. Kodama, T. Irita, T. Kamei, M. Ishikawa, Y. Nitta, O. Nishii, and T. Hattori, “An embedded processor core for consumer appliances with 2.8gflops and 36m polygons/s fpu,” in *Proc. of IEEE Internatilnal Solid-State Ciruits Conference (ISSCC2004)*, Feb. 2004.

[13] D. Brooks, V. Tiwari, and M. Martonosi, “Wattch: A framework for architectural-level power analysis and optimizations,” in *Proc. of the 27th International Symposium on Computer Architecture*, Jun. 2000.