Evaluation of Power Consumption at Execution of Multiple Automatically Parallelized and Power Controlled Media Applications on the RP2 Low-power Multicore

Hiroki Mikami[1], Shumpei Kitaki[1], Masayoshi Mase[1], Akihiro Hayashi[1], Mamoru Shimaoka[1], Keiji Kimura[1], Masato Edahiro[1], and Hironori Kasahara[1]

Dept. of Computer Science, Waseda University 3-4-1 Ohkubo, Shinjuku-ku, Tokyo, 169-8555, Japan
{hiroki,kitaki,mase,ahayashi,shimaoka}@kasahara.cs.waseda.ac.jp, kimura@apal.cs.waseda.ac.jp, eda@etrl.jp, kasahara@kasahara.cs.waseda.ac.jp

Abstract. This paper evaluates an automatic power reduction scheme of OSCAR automatic parallelizing compiler having power reduction control capability when multiple media applications parallelized by the OSCAR compiler are executed simultaneously on RP2, a 8-core multicore processor developed by Renesas Electronics, Hitachi, and Waseda University. OSCAR compiler enables the hierarchical multigrain parallel processing and power reduction control using DVFS (Dynamic Voltage and Frequency Scaling), clock gating and power gating for each processor core using the OSCAR multi-platform API. The RP2 has eight SH4A processor cores, each of which has power control mechanisms such as DVFS, clock gating and power gating. First, multiple applications with relatively light computational load are executed simultaneously on the RP2. The average power consumption of power controlled eight AAC encoder programs, each of which was executed on one processor, was reduced by 47%, (to 1.01W), against one AAC encoder execution on one processor (from 1.89W) without power control. Second, when multiple intermediate computational load applications are executed, the power consumptions of an AAC encoder executed on four processors with the power reduction control was reduced by 57% (to 0.84W) against an AAC encoder execution on one processor (from 1.95W). Power consumptions of one MPEG2 decoder on four processors with power reduction control was reduced by 49% (to 1.01W) against one MPEG2 decoder execution on one processor (from 1.99W). Finally, when a combination of a high computational load application program and an intermediate computational load application program are executed simultaneously, the consumed power reduced by 21% by using twice number of cores for each application. This paper confirmed parallel processing and power reduction by OSCAR compiler are efficient for multiple application executions. In execution of multiple light computational load applications, power consumption increases only 12% for one application. Parallel processing being applied to intermediate computational load applications, power consumption of executing one application on one processor core (1.49W) is almost same power consumption of two applications on eight processor cores (1.46W).

1 Introduction

Multicore processors have been widely used in a variety of applications such as embedded systems like mobile devices, games, Digital TV, robots and automobiles, PCs, workstations, and high-performance computers. In embedded systems, various types of multicore processors, for example IBM Toshiba Sony CELL/BE[1], Hitachi Renesas Waseda RP1[2], RP2[3] and RPX[4], ARM NEC MPCore[5], Fujitsu FR1000[5], Panasonic UniPhier and so on, are used for a wide variety of applications such as image and audio processing, and real-time controls. In these embedded multicore platforms, the reduction of power consumption is a crucial problem to extend battery life. Currently, many multicore supports DVFS and/or Power Gating for each processor coare controled by OS. However, OS does not control power status inside an application program pallalelized for multiple cores. The OSCAR compiler has realized an automatic power control scheme using DVFS and Power Gating for each core on a multicore with automatic parallelization of an application program under the constraints of the minimum time execution or the satisfaction of real-time deadline, or real-time execution. However, no paper has evaluated power consumed by multiple application programs parallelized and power controlled by a compiler. This paper evaluates performance and consumed power on a 8-core homegeneous multicore RP2 integrating eight 600MHz SH4A processor cores with power control function of 100%, 50%, 25%, 0% of Frequency statuses in a one clock transition time, 1.4V, 1.2V and 1.0V three levels of voltage states and power gating for individual cores in 5 micro seconds power shut-down and 30 micro seconds power recovery when multiple media applications parallelized and power-controlled by OSCAR comoiler are executed simultaneously. Also, the parallel and power controlled C programs are generated using OSCAR multi-platform API, which are a set of about 20 directives for C and Fortran programs using 4 directives from OpenMP, such as Section, Flush, Critical and Thread-private and new directives for power control, realtime management, DMA transfer, distributed shared memory management, group barrier synchronization and so on. The generated C or Fortran parallel program using OSCAR API[6] can be compiled by ordinary OpenMP compilers. The rest of this paper is organized as follows. Section 2 provides an overview of the OSCAR compiler and its power reduction scheme. Section 3 describes RP2 low-power multicore and characteristic of evaluated applications. Section 4 shows the power consumption evaluation of OSCAR compiler power reduction scheme on RP2. Finally, Section 5 summarizes the main conclusion of this paper.

2 Multigrain Parallel Processing

The OSCAR compiler exploits multigrain parallelism, coarse grain parallelism, loop level parallelism and near fine grain parallelism from the whole source program. The OSCAR compiler consists of the Fortran77 and restricted C frontend, middle path for multigrain parallelization and several backbends for different target machines. The compiler generates coarse grain tasks called macro-tasks, analyzes parallelism among the macro-tasks by the earliest executable condition analysis,schedules macro-tasks to threads or thread groups statically, apply power reduction scheme, generates parallel code with OSCAR API.

2.1 Macro-Task Generation

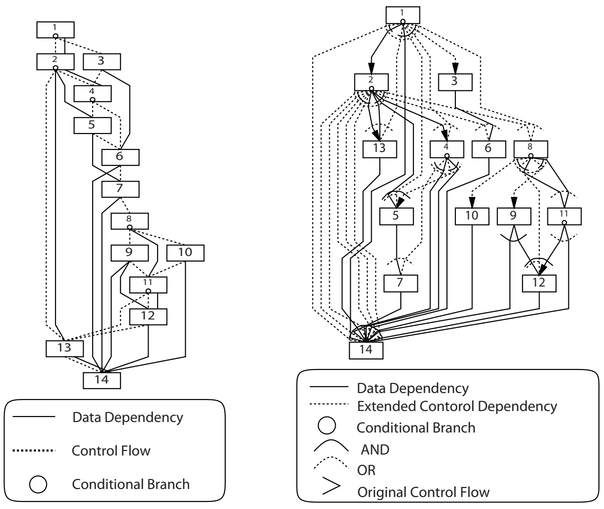
In multigrain parallelization, a program is decomposed into three kinds of coarse grain tasks, or macrotasks (MTs), such as a block of pseudo assignment statements (BPA) like a basic block, a repetition block (RB) like a loop and a subroutine block (SB) like a subroutine [7–11]. Macro-tasks can be hierarchically defined inside each sequential loop which cant be parallelized, and a subroutine block.

2.2 Earliest Executable Condition

After generation of macro-tasks, data dependencies and control flow among macro-tasks are analyzed in each nested layer, and hierarchical macro-flow graphs (MFGs) as shown in Figure 1 (a) representing control flow and data dependencies among macro-tasks are generated[7, 8]. Next, to extract coarse grain task parallelism among macro-tasks, Earliest Executable Condition analysis[7, 8, 12, 13] is applied to each macro-flow graph. It analyzes control dependencies and data dependencies among macro-tasks simultaneously and determines the conditions on which macro-tasks may begin their execution earliest. By this analysis, a macro-task graph (MTG)[7, 8, 12] as shown in Figure 1 (b) is generated for each macro-flow graph. This graph represents coarse grain task parallelism among macro-tasks.

2.3 Macro-task Scheduling

Static scheduling or dynamic scheduling is chosen for each macro-task graph. If a macro-task graph has only data dependencies and is deterministic, static scheduling at compilation time is selected. Generally, static scheduling is more effective than dynamic scheduling since it can minimize data transfer and synchronization overhead without runtime scheduling overhead. If a macro-task graph is non-deterministic by conditional branches among coarse grain tasks, dynamic scheduling at runtime is selected to handle the runtime uncertainties. Dynamic scheduling routines for non-deterministic macro-task graphs are generated by OSCAR compiler and inserted into a parallelized program code to minimize runtime scheduling overhead.



(a) Macro Flow Graph (MFG) (b) Macro Task Graph (MTG)

Fig. 1. Earliest Executable Condition Analysis

2.4 Power reduction scheme[14]

The power reduction scheme determines suitable voltage and frequency for each MT after Macro-task scheduling. Figure 2 (a) shows MTs 1, 2 and 5 are assigned to PE0, MTs 3 and 6 are assigned to PE1, MTs 4, 7 and 8 are assigned to PE2. Edges among tasks show data dependence. OSCAR compiler estimates execution time of each MTs, then decide critical path which is longest execution time of the MTG. Defining execution time of the target MTG, parallel processing of the MTG after DVFS has to satisfy the given deadline. OSCAR compiler with the power reduction scheme decides optimal frequency and voltage of each MT to minimize the whole energy consumption. The detail of voltage and frequency scaling algorithm is described in [14]. After determining voltage and frequency of MTs, OSCAR compiler with the power reduction scheme tries to apply dynamic power shutdown, clock gating or frequency scaling to reduce unnecessary energy consumption including static power consumption by idle processors. OSCAR compiler recalculates MTGs after DVFS like Figure 2 (b) and selects power gating, clock gating, frequency scaling or no control for each idle part, considering the period of idle time and its overhead.

2.5 OSCAR API Code Generation[6]

The OSCAR API is designed on a subset of OpenMP for preserving portability over a wide range of multicore architectures. An OpenMP-based design can support both C and Fortran programs. However, in order to avoid the complexity of a backend compiler and runtime routines, only three directives are

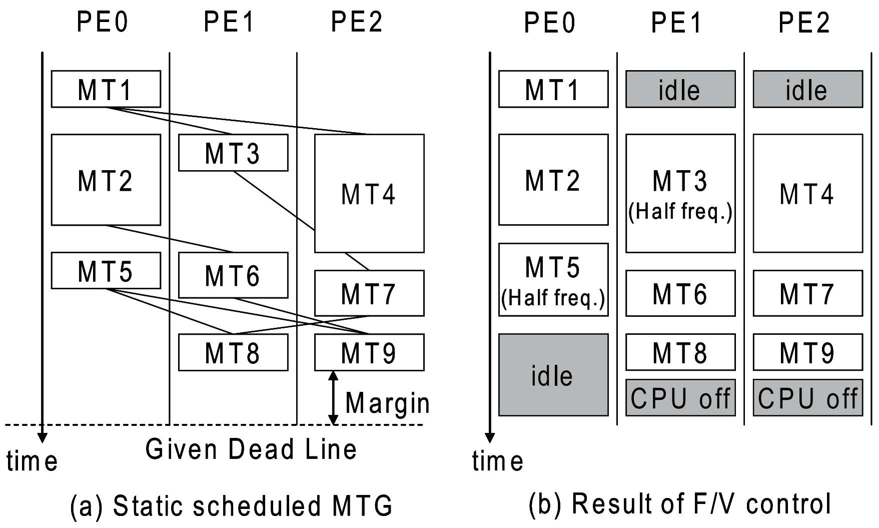


Fig. 2. OSCAR compiler's power control scheme

chosen from the OpenMP, such as parallel sections, flush, and critical, which enable one-time single level thread creation. Note that nested parallelism is not required for the OSCAR API. In addition to these three directives, one OpenMP directive (threadprivate) is extended, and 12 directives are newly added to support the parallel optimizations carried out using the OSCAR compiler, whose specifications are simple as possible.

3 RP2 Multicore and Characteristic of applications

This chapter describes RP2, 8-core multicore processor developed by Renesas Electronics, Hitachi and Waseda University and characteristic of evaluated applications.

3.1 RP2 Multicore

RP2[3] is 8 cores multicore processor developed by Renesas Electronics / Hitachi / Waseda University supported by NEDO Multi core processors for realtime consumer electronics project. RP2 integrate eight SH-4A cores. Figure 3 shows the architecture of RP2. Each processor core has CPU, cache memory, local memory (ILRAM, OLRAM), distributed shared memory (URAM), and DMAC (DTU). RP2 guarantee hardware cache coherence by MESI protocol until 4 cores. However, software must guarantee cache coherence 5 cores and above. Frequency of each core can be changed to 600MHz, 300MHz, 150MHz, and 75MHz independently. In addition supply voltage of entire core can be changed to 1.40V, 1.20V,

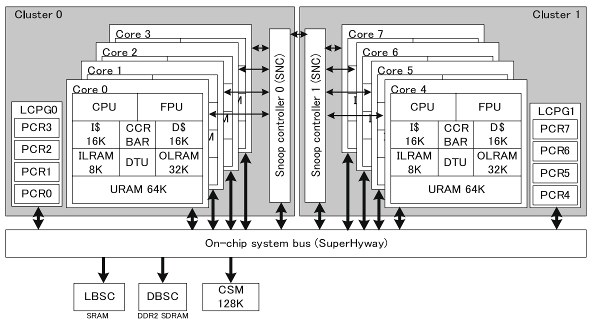


Fig. 3. Architecture of RP2 Multicore

and 1.00V. Figure 4 shows RP2 power status. Light Sleep stop CPU clock supply. Normal Sleep stop clock supply of processor core except URAM and DMAC. Resume Standby stop URAM clock supply and shutdown processor core except URAM. CPU off shutdown entire core.

Status	Clock Gating	Power Shutdown	Power Consumption [W]
FULL (600MHz,1.40V)	-	-	5.99
MID (300MHz, 1.20V)	-	-	2.61
LOW (150MHz,1.00V)	-	-	1.27
VERYLOW (75MHz, 1.00V)	-	-	1.00
Normal Sleep	CPU, cache, ILRAM, OLRAM	-	0.725
Resume Standby	URAM	CPU, cache, ILRAM, OLRAM, DTU	0.563
CPU off	-	CPU, cache, ILRAM, OLRAM, DTU, URAM	0.554

Fig. 4. Power Status of RP2 Multicore

3.2 Evaluated Applications

This section describes specifications of evaluated applications.

AAC Encoder This program read audio data and process Filter bank, MS stereo, Quantization, and Haffman coding for each frame. Each frame can be processed parallel. Encoding process are unrolled by number of processor. Deadline of power control is 23ms per one frame. This AAC encoder is implemented by Parallelizable C, which is referred to AAC-LC encoding program of Renesas Electronics and Hitachi.

MPEG2 Decoder MPEG2 decoder stages are Variable Length Decoding (VLD), Motion Compensation, Inverse Quantization and Inverse DCT. MPEG2 decoder has slice parallelism and macroblock parallelism. Each parallelism has sequential execution on VLD. In this paper, the code of MPEG2 decoder is implemented with reference of MediaBench[15] with the description explained in Section5.1. Furthermore, VLD for a slice is divided into searching a slice header, called Prescanning[16], and decoding a slice. Decoding slices is executed in parallel, so that parallel execution part is increased. The OSCAR compiler extracts slice level parallelism. Deadline of power control is 33ms per one frame.

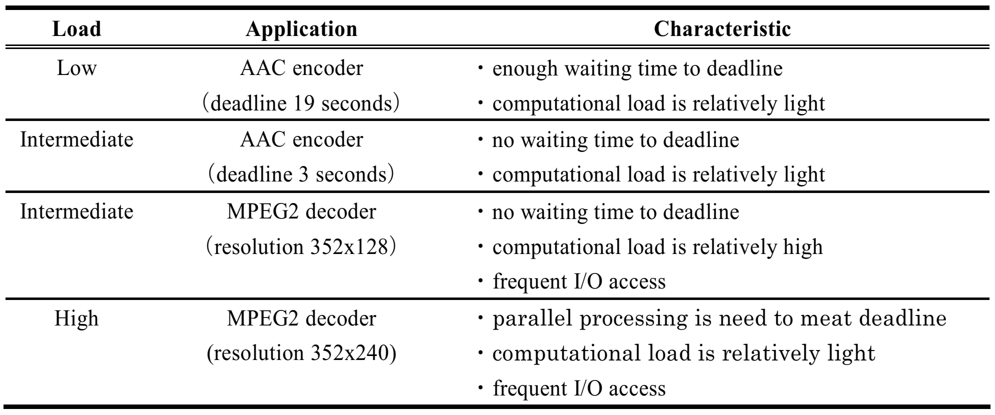
Characteristics of Application Figure 5 shows characteristics of each application. As a light computational load application, AAC encoder is selected. AAC encoder can fulfill deadline by VERYLOW (75MHz) power status. 19 seconds audio data is inputted. AAC encoder process 19 seconds audio data by 2.7 seconds. AAC encoder has enough waiting time for deadline. As a middle computational load application, AAC encoder (deadline 3 seconds) and MPEG2 decoder (resolution 352x128) is selected. AAC encoder need 2.7 seconds by one core, so this AAC encoder must run at FULL power status when using only one core. MPEG2 decoder process 352 pixel x 128 pixel video by 8.3 seconds by using one core. In addition continuous I/O has been issued by bit processing of Prescanning. As a high computational load application, MPEG2 decoder (resolution 352x240) is selected. MPEG2 decoder process 352 pixel x 240 pixel video by 17.6 seconds by using one core and 11.1 seconds by using two cores. MPEG2 decoder must use two cores or above to fulfill deadline (15 seconds). This paper execute these applications multiple and mesure power consumption of entire chip.

4 Performance of simultaneous execution of multiple application programs parallelized and power-controlled by OSCAR compielr

This section evaluates execution performance and consumed power on the RP2 eight core homogeneous multicore with DVFS and power gating capabilities when multiple media application programs, which are automatically parallelized and power-controlled by OSCAR compiler, are executed simultaneously sharing eight cores.

4.1 Performance of simultaneous execution of multiple low-processing-load applications

This sub-section describes consumed power on RP2 when low-processing-load applications shown in Figure 5 namely each application program is a real-time



Load	Application	Characteristic
Low	AAC encoder （deadline 19 seconds）	・enough waiting time to deadline ・computational load is relatively light
Intermediate	AAC encoder （deadline 3 seconds）	・no waiting time to deadline ・computational load is relatively light
Intermediate	MPEG2 decoder （resolution 352x128）	・no waiting time to deadline ・computational load is relatively high ・frequent I/O access
High	MPEG2 decoder (resolution 352x240)	・parallel processing is need to meat deadline ・computational load is relatively light ・frequent I/O access

Fig. 5. Characteristic of Application

AAC encoder to encode a 16 seconds music file in 16 seconds, are executed in parallel. Figure 6 shows consumed power when one to eight light-load AAC encoders are executed in parallel using different numbers of processor cores on the RP2. The vertical axis shows the consumed power and the horizontal axis shows number of processor cores, or PEs, to execute the two AAC encoders. In each number of PEs, AAC encoder programs less than number of PEs are executed. For example, on one PE, just one AAC encoder is executed sequentially with a non power controlled mode shown in left bar and a power controlled mode shown in right bar. Also, on eight PEs, the left bar shows consumed power when one AAC encoder is executed in parallel, each of which uses one PE, with non power controlled mode. The second left bar shows the consumed power when one AAC encoder is executed on 8 PEs in parallel with the power control. The third left bar shows the consumed power when two AAC encoders are executed in parallel each of which uses 4 PEs respectively. The forth bar shows the power when 4 AAC encoders are executed in parallel, each of which uses 2 PEs respectively. The fifth left bar shows the power when 8 AAC encoders are executed in parallel, each of which uses 1 PE. In other words, if "N" application programs are executed on "M" PEs, "M/N" PEs are assigned to each application programs in each number of PEs. In this light-load AAC encoder, a PE can easily execute the AAC encoder keeping real-time deadline. On the1 PE, though one AAC encoder without power control, or an ordinary single core execution with 100% frequency (600 MHz) and the highest voltage (1.4V), consumes 1.89 W, the power controlled AAC encoder just requires 0.59 W since OSCAR compiler chose 1/8 frequency (75 MHz) and the lowest voltage (1.0V) for waiting the dead line. Namely, OSCAR compiler gives us 69% of power reduction when one AAC encoder is executed on one PE. On 2 PEs, one AAC encoder parallelized on 2 PEs without power control consumes 2.19 W. One AAC encoder parallelized to 2 PEs with power control consumes 0.59 W that is the same as on the 1 PE since OSCAR compiler applies appropriate DVFS control in nano-second order during execution and power gating to the second PE and lowest frequency and voltage power states to the first PE during waiting for the deadline. Also, two AAC encoders on 2 PEs, in which each AAC encoder is executed on 1 PE with power control, consumed just 0.66 W. In the two AAC encoder real-time execution, OSCAR compiler reduces power by 70% from 2.19W to 0.66W. On 4 PEs, one AAC encoder parallelized for 4 PEs without power control consumes 2.75 W. One AAC encoder parallelized to 4 PEs with power control consumes 0.59 W, or the same as on 1 PE. The two AAC encoders on 4 PEs, in which each AAC encoder is executed on 2 PEs with power control, consumed just 0.68 W. Also, four AAC encoders on 4 PEs, in which each AAC encoder is executed on 1 PE with power control, consumed just 0.78 W. In other words, OSCAR compiler reduces the power by 72% from 2.19W to 0.66W when four AAC encoders are executed on 4PEs. On 8 PEs, one AAC encoder parallelized for 8 PEs without power control consumes 3.47 W. One AAC encoder parallelized to 8 PEs with power control consumes 0.60 W, or the almost same as on 1 PE. The two AAC encoders on 8 PEs, in which each AAC encoder is executed on 4 PEs with power control, consumed just 0.69 W that is similar to 0.66 W on 2PEs and 0.68 W on 4 PEs. The four AAC encoders on 8 PEs, in which each AAC encoder is executed on 2 PEs with power control, consumed 0.82 W that is just 0.04 W larger than on 4 PEs. Also, eight AAC encoders on 8 PEs, in which each AAC encoder is executed on 1 PE with power control, consumed just 1.01 W. In other words, OSCAR compiler reduces the power by 71% from 3.47 W to 0.66 W when eight AAC encoders are executed on 8 PEs. Here, we should pay attentions that the eight AAC encoder execution on 8 PEs just requires the just 0.13 W for one AAC encoder though the one AAC execution on 1 PE without power control consumes 1.89 W, namely the eight core execution gives us 93% power reduction, and the one AAC execution on 1 PE with power control consumes 0.59 W, namely the eight core execution gives us 78% power reduction. These results show the simultaneous execution of multiple low-load application programs with OSCAR compiler's power control gives us huge reduction in a single application average power consumption.

4.2 Performance of simultaneous execution of multiple Intermediate processing-load applications (AAC encoder)

This sub-section describes consumed power on RP2 when intermediate- processing-load applications shown in Figure 5, namely each application program is a super-real-time AAC encoder to encode a 16.0 seconds music file in 3.0 seconds, are executed in parallel. In this application, a single PE needs 2.7 seconds to process and manages to satisfy the real-time deadline with the almost highest clock frequency. Figure 7 shows consumed power when one to four intermediate-load AAC encoders are executed in parallel using different numbers of processor cores on the RP2. Here, the case of eight AAC encoders is not shown because eight AAC encoders on 8 PEs exceeds RP2's memory capacity. The vertical axis shows the consumed power and the horizontal axis shows number of PEs as Figure 7. On the1 PE, one AAC encoder without power control consumes 1.95 W and

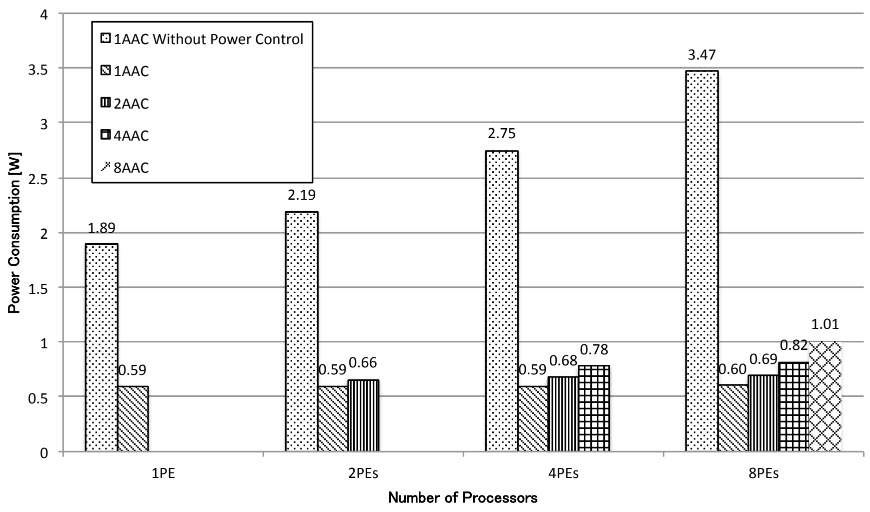


Fig. 6. Power Consumption of Low processing-load applications (AAC encoder)

the power controlled AAC encoder requires 1.78W, namely 9% power reduction, since in the intermediate-load there is only a 0.3 s to wait for dead line in which OSCAR compiler can choose 1/8 frequency (75 MHz) and the lowest voltage (1.0V). On 2 PEs, one AAC encoder parallelized for 2 PEs without power control consumes 2.27 W. One AAC encoder parallelized to 2 PEs with power control consumes 1.18 W that is 40% power reduction from 1.95 W on 1 PE without power control and 34% power reduction from 1.78 W on 1PE with power control since OSCAR compiler applies appropriate DVFS and power gating automatically. Also, two AAC encoders on 2 PEs, in which each AAC encoder is executed on 1 PE with power control, consumed 2.20 W. In the two AAC encoder real-time execution, OSCAR compiler reduces power by 3% from 2.27W to 2.20W. On 4 PEs, one AAC encoder parallelized for 4 PEs without power control consumes 2.85 W. One AAC encoder parallelized to 4 PEs with power control consumes 0.84 W, namely 57% power reduction from 1.95 W of one AAC on 1 PE without power control and 53% power reduction from 2.78W of one AAC on 1 PE with power control. On 8 PEs, one AAC encoders without power control consumes 3.59 W. One AAC encoder parallelized to 8 PEs with power control consumes 0.87 W that is the same as one AAC on 4 PEs. The two AAC encoders on 8 PEs, in which each AAC encoder is executed on 4 PEs with power control, consumed just 1.13 W that is 78% reduction from 2.20 W of two AACs on 2 PEs which is the minimum number of PEs to satisfy the deadline and 26% reduction from 1.53 W of two AACs on 4PEs. The four AAC encoders on 8 PEs, in which each AAC encoder is executed on 2 PEs with power control, consumed 2.21 W that is the same as 2.20W of two AACs on 2PEs. These re-

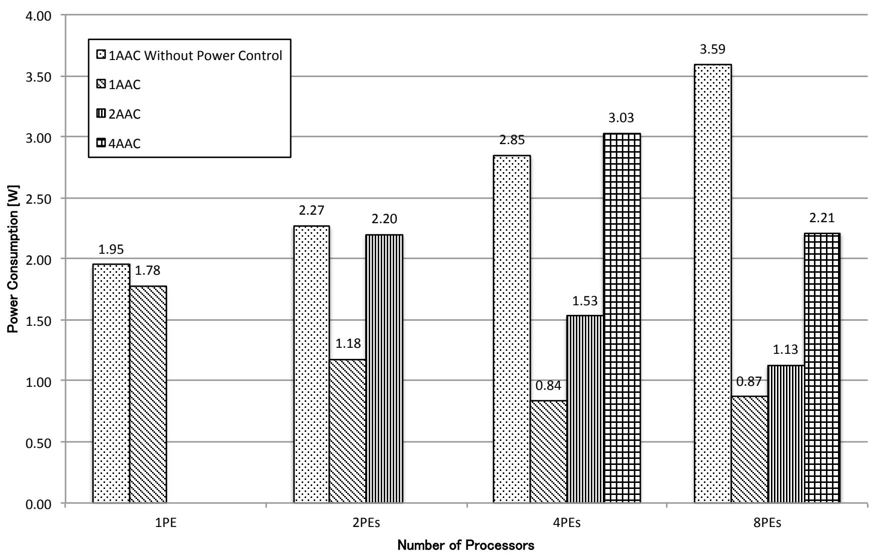


Fig. 7. Power Consumption of Intermediate processing-load applications (AAC encoder)

sults show the simultaneous execution of multiple intermediate-load application programs with OSCAR compiler's power control gives us large reduction by parallel execution of each application since the parallel processing make chances of DVFS and power gating during execution.

4.3 Performance of simultaneous execution of multiple Intermediate-processing-load applications (MPEG2 decoder)

Figure 8 shows power consumption of MPEG2 decoder multiple executions. Horizontal axis indicates sum of processor PEs used by all applications. Vertical axis indicates power consumption of entire chip. Each bar indicates number of MPEG2 decoders executed multiple. 1MPEG2 means power consumption of one MPEG2 decoder execution. 2MPEG2 means power consumption of two MPEG2 decoders, which is using half number of PEs indicated by X-axis. Power consumption reduce from 1.99W (one PE) to 1.0W (four PEs) by applying parallel processing. Power consumption of 2 MPEG2 decoders (four PEs for each) is 1.46W. This is almost same power consumption of 1 MPEG2 decoder (one PE, 1.49W). Middle computational load applications can be reduce power consumption by applying parallel processing and executing multiple applications. Power consumption of each MPEG2 decoder is 0.73W, which reduce power consumption at 51% against 1 MPEG2 decoder (one PE). MPEG2 decoder has high bus

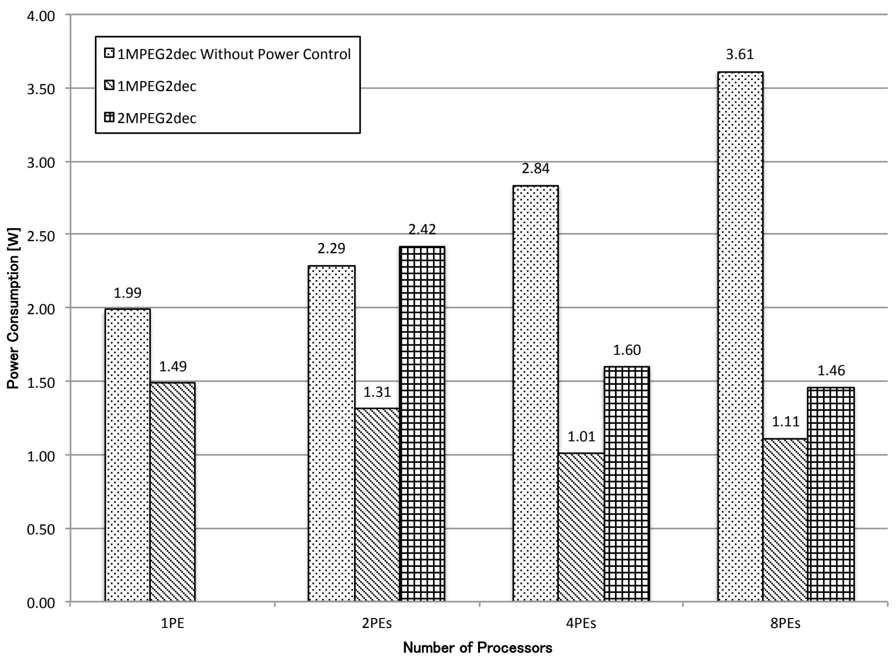


Fig. 8. Power Consumption of Intermediate processing-load applications (MPEG2 decoder)

pressure by Prescanning. MPEG2 decoder should be executed to shift I/O timing. This is because power consumption reduction ratio of MPEG2 decoder is relatively low against AAC encoder. Processors voltage control domain should be divided to control power effectively when middle computational load applications are executed.

4.4 Performance of simultaneous execution of multiple High-processing-load applications (MPEG2 decoder)

Figure 9 shows power consumption of MPEG2 decoder multiple executions. horizontal axis indicates sum of processor PEs used by 352x128 resolution MPEG2 decoder and 352x240 resolution MPEG2 decoder. Vertical axis indicates power consumption of entire chip. When different resolution MPEG2 decoders are executed multiple, timing of FV control are differ. This is because execution time of each decode stage are different. Thus, supply voltage (controlled by entire chip) is hard to down. In addition, power status of each application is different. For example, power status is FULL+MID by 2PE+2PE execution. At this time supply voltage is 1.40V (FULL), so power consumption is 2.28W. This reduces only 3% against 2PE+1PE execution. However, power status is MID+LOW by

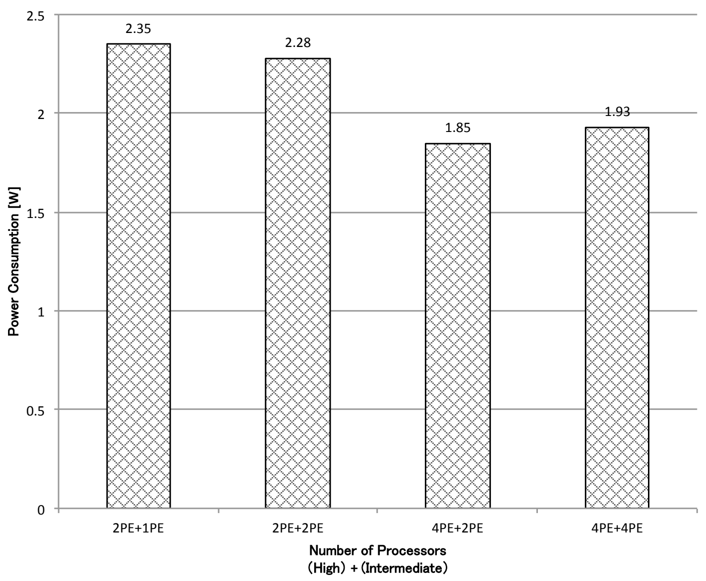


Fig. 9. Power Consumption of High processing-load application and Intermediate processing-load application

4PE+2PE execution. At this time supply voltage is 1.20V and power consumption is 1.85W. This reduces 21% against 2PE+1PE execution (2.35W). When middle computational load application and high computational load application are executed multiple, hardware co-operation to divide supply voltage domain is effective to reduce power consumption.

5 Conclusions

This paper has evaluated the power reduction scheme of OSCAR automatic parallelizing compiler when multiple media application programs parallelized and po-wer-controlled by OSCAR compiler are executed simultaneously on the Renesas / Hitachi / Waseda RP2 eight core homogeneous multicore processor. Execution performances are almost no differences from a single program execution when multiple parallelized programs are executed simultaneously since OSCAR compiler's cache memory optimization function minimizes main memory, or off-chip shared memory, accesses and prevents main memory contentions. Power consumption of multiple applications with relatively light computational load was reduces by 68% against non-power controlled applications. When multiple applications are executed simultaneously, total power consumption was 1.01W with 8 AAC encoders, each of which was executed on one core. At this time, an average power consumption of each AAC encoder was 0.13W and this value was 22% of power compared with one application executed on 8 cores. This result shows that when we execute 8 AAC encoder (1 core for each), power consumption reduce at 78% against 1 AAC encoder (8 cores). Light computational load applications should be executed as much as possible to reduce power consumption of one application. Average power consumption of multiple applications with middle computational load reduces at 30% by parallel executions. In addition, when 4 AAC encoders (2 cores for each) are executed, power consumption is 0.55W by 1 AAC encoder. This power consumption reduces at 37% against 1 AAC encoder (8 cores), and reduces at 69% against 1 AAC encoder (1 cores). When multiple meddle computational load applications are executed, parallel processing can reduce power consumption. Average power consumption of 2 MPEG2 decoders (4 cores for each) is lower than average power consumption of 1 MPEG2 decoder (1 core). At this time, power consumption of each MPEG2 decoder (0.73W) reduces at 51% against 1 MPEG2 decoder (1.49W). However, if voltage control domain is divided by hardware, there is more room to reduce power consumption. This tendency is more notable at executing both high computational load application and middle computational load application multiple. This paper confirmed automatic parallelization and automatic power control scheme of OSCAR compiler is effective to reduce power consumption of multiple applications.

Acknowledgements

I would like to thank Junji Sakai (NEC Corporation) whose comments and suggestions were of inestimable value for our study. I am also indebt to Kazuhisa Ishizaka (NEC Corporation) whose comments made enormous contribution to our work.

A part of this research has been supported by NEDO "Advanced Heterogeneous Multiprocessor", NEDO "Multi core processors for realtime consumer electronics" and STARC "Automatic Parallelizing Compiler Cooperative Single Chip Multiprocessor".

References

1. Dac Pham et al. The design and implementation of a first-generation cell processor. In *In Proceeding of the IEEE International Solid-State Circuits Conference*, 2005.
2. K. Hayase S. Shibahara O. Nishii T. Hattori A. Hasegawa M. Takada N. Irie K. Uchiyama T. Odaka K. Takada K. Kimura H. Kasahara Y. Yoshida, T. Kamei. A 4320mips four-processor core smp/amp with individually managed clock frequency for low power consumption. In *2007 IEEE International Solid-State Circuits Conference(ISSCC2007)*, Feb. 2007.
3. Yutaka Yoshida Kiyoshi Hayase Tomoichi Hayashi Osamu Nishii Yoshihiko Yasu Atsushi Hasegawa Masashi Takada Masaki Ito Hiroyuki Mizuno Kunio Uchiyama Toshihiko Odaka Jun Shirako Masayoshi Mase Keiji Kimura Hironori Kasahara Masayuki Ito, Toshihiro Hattori. An 8640 mips soc with independent power-off

control of 8 cpu and 8 rams by an automatic parallelizing compiler. In *Proc. of IEEE International Solid State Circuits Conference (ISSCC2008)*, Feb. 2008.

4. Y. Kiyoshige Y. Nitta S. Matsui O. Nishii A.Hasegawa M. Ishikawa T. Yamada J. Miyakoshi K. Terada T. Nojiri M. Satoh H. Mizuno K. Uchiyama Y. Wada K. Kimura H. Kasahara H.Maejima Y. Yuyama, M. Ito. A 45nm 37.3gops/w heterogeneous multi-core soc. In *IEEE INTERNATIONAL SOLID-STATE CIRCUITS CONFERENCE (ISSCC 2010)*, Feb. 2010.
5. J Cornish. Balanced energy optimization. In *International Symposium on Low Power Electronics and Design*, 2004.
6. Hiroki Mikami Takamichi Miyamoto Jun Shirako Keiji Kimura, Masayoshi Mase and Hironori Kasahara. Oscar api for real-time low-power multicores and its performance on multicores and smp servers. In *Proc. of The 22nd International Workshop on Languages and Compilers for Parallel Computing (LCPC2009)*, Oct. 2009.
7. H. Honda, M. Iwata, and H. Kasahara. Coarse grain parallelism detection scheme of a fortran program. *Trans. of IEICE*, J73-D-1(12):951–960, Dec. 1990.
8. H.Kasahara and et al. A multi-grain parallelizing compilation scheme on oscar. *Proc. 4th Workshop on Language and Compilers for Parallel Computing*, 1991.
9. Hironori Kasahara. Advanced automatic parallelizing compiler technology. *IPSJ MAGANIE*, Apr 2003.
10. K. Ishizaka, T. Miyamoto, M. obata J. Shirako, K. kimura, and H. Kasahara. Performance of oscar multigrain parallelizing compiler on smp servers. In *Proc. of 17th International Workshop on Languages and Compilers for Parallel Computing*, Sep. 2004.
11. M. Obata, J. Shirako, H. Kaminaga, K. Ishizaka, and H. Kasahara. Hierarchical parallelism control for multigrain parallel processing. In *Proc. of 15th International Workshop on Languages and Compilers for Parallel Computing*, Aug. 2002.
12. H. Kasahara, H. Honda, M. Iwata, and M. Hirota. A compilation scheme for macro-dataflow computation on hierarchical multiprocessor system. *Proc. Int Conf. on Parallel Processing*, 1990.
13. H. Kasahara, H. Honda, and S. Narita. Parallel processing of near fine grain tasks using static scheduling on oscar. *Proceedings of Supercomputing '90*, Nov. 1990.
14. 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 18th International Workshop on Languages and Compilers for Parallel Computing(LCPC2005)*, Oct. 2005.
15. C. Lee et al. Mediabench: A tool for evaluating and synthesizing multimedia and communications systems. In *In 30th International Symposium on Microarchitecture (MICRO30)*, Nov. 1997.
16. E. Iwata et al. Exploiting coarse-grain parallelism in the mpeg-2 algorithm. In *Technical Report CSL-TR-98-771*, Sep. 1998.