FR-V Single-Chip Multicore Processor: FR1000

● Atsuhiro Suga ● Satoshi Imai

(Manuscript received September 30, 2005)

To realize the low power consumption and low-cost equipment needed to decode high definition broadcasts, Fujitsu has developed a single-chip multicore processor FR1000 that integrates four 8-way, Very Long Instruction Word (VLIW) FR-V processor cores. This new multicore processor is fabricated using a 90 nm, nine-metal-layer CMOS process and a 900-pin flip-chip package. The processor cores operate at 500 MHz, the memory interfaces at 250 MHz, and system bus at 166 MHz. The use of a single processor core enables MPEG-2 MP@ML video-stream decoding at 190 MHz. Conversely, the use of four processor cores enables the decoding of MPEG-2 MP@HL video streams by just using software. Moreover, this new processor needs only about 3 W to decode MPEG-2 MP@HL video streams. This paper introduces the hardware and software development environment of this new processor, describes the processor's software operation environment, and cites some examples of its application.

1. Introduction

This paper introduces Fujitsu's FR1000 single-chip multicore processor that integrates quadruple Very Long Instruction Word (VLIW) FR-V processor cores and specially developed software. This new processor inherits and builds upon Fujitsu's supercomputing technologies.[1)-3), note)]

There are two major methodologies for increasing the processing performance of a processor: increasing the frequency, and increasing the total number of processing elements in parallel. Processors used for consumer products must satisfy the requirements of high performance, affordable price, and low power consumption. For example, the processor for a personal computer (PC) may achieve high performance, but requires a forced-air cooling device or an expensive sealing package for the chip to operate within a power consumption limit of 100 W. Therefore, employing such expensive devices for a consumer products market under severe price competition is not possible. The price levels for embedded processors are generally set much lower than those of general-purpose processors. Consequently, we must use a less expensive package and cool it without using a cooling fan. To meet these conditions, we must make a low-priced LSI that consumes about 1/50 of the power consumed by a PC microprocessor. Hence, we decided to achieve high performance and low power consumption by retaining the fundamental FR-V concept and increasing the total number of processing elements in parallel. The FR550[1)] uses VLIW and Single Instruction Multiple Data (SIMD) to achieve the paralleled processing of eight instruction levels and four data levels (**Figure 1**). Moreover, without increasing the frequency, it was necessary to execute in parallel

note) VLIW can be referenced in the following document:
Joseph A. Fisher et al, "Embedded Computing — A VLIW Approach to Architecture, Compiler and Tools," Morgan Kaufmann Publishers. FR-V is introduced on p.70 as an example of the VLIW processor.

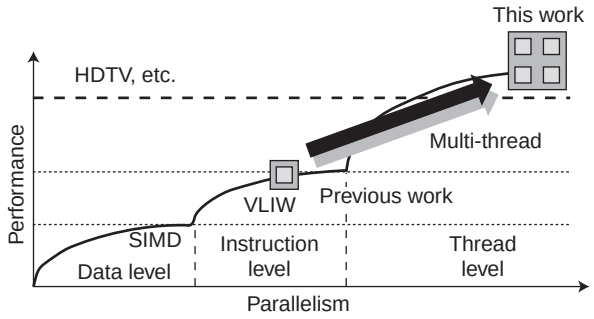


Figure 1
Effects of processor-core parallelism on limits of VLIW architecture.

several processing steps larger than those at the instruction level to boost the level of performance. Since semiconductor process technology has entered the 90 nm age, the FR1000 multicore processor with a VLIW processor that we had initially planned could finally be realized.[3)]

Our target was realizing media processing, such as for images and audio, exclusively by software. Previously, the control CPU was configured with dedicated hard-wired logic (e.g., Application Specific Integrated Circuits [ASICs]) to perform this kind of processing. Until now, a supercomputer or computing server had to divide a main task into multiple parallel tasks, primarily by utilizing the loop parallelism in source code. However, such media processing programs as those for MPEG decoding include embedded parallel tasks like Inverse Discrete Cosine Transform (IDCT) functions and a large portion of slice level tasks. Therefore, we decided to equip the multiple-core VLIW processor with our processor. Thread-level (which is larger than instruction-level) parallel processing was performed on multiple processor cores, with the instruction-level parallel processing performed in each processor core using VLIW architecture as done by existing FR-V processors. As a result, we achieved higher performance at lower power consumption. For example, the decoding of MPEG-2 MP@HL, which represents six times the processing volume of MPEG-2 MP@ML, consumed only about twice as much as power as for the decoding of MP@ML streams by the FR550.

One major problem to resolve for realizing a multicore processor system was providing an environment to enable easy program debugging. We planned to apply inherited programs and the same software development environment developed for single processors to the multicore processor environment. Therefore, even though the processor was actually a shared memory system, we treated it as a distributed memory system to enable use of our inherited assets for a multicore software development environment. This allowed us to use the existing software development environment in the same way as for a single processor core. Moreover, the debugging facility for the multicore processor was realized by adding functions to existing hardware and software functions.

The multicore architecture employed by the FR1000 is based on a single processor with additional architecture to control communication between the multicores. This architecture makes it possible to construct the software development environment for a multiprocessor that inherits the software development environment for single processors.

2. FR1000 hardware architecture

The FR1000 multicore processor integrates quadruple VLIW processors on a single chip. Each VLIW processor core can execute up to eight parallel instructions at a time.

Moreover, each processor core of the FR1000 is a FR550-compatible processor. The instructions executed on the FR550 processor core consist of integer arithmetic instructions, floating point arithmetic instructions, and media instructions with 16-bit, fixed-point arithmetic operations. A single media instruction or SIMD can process four or eight operations in parallel. A processor core capable of executing eight parallel instructions simultaneously can execute 28 operations simul-

taneously in one cycle. Consequently, the FR1000 processor can execute 112 operations simultaneously per cycle.

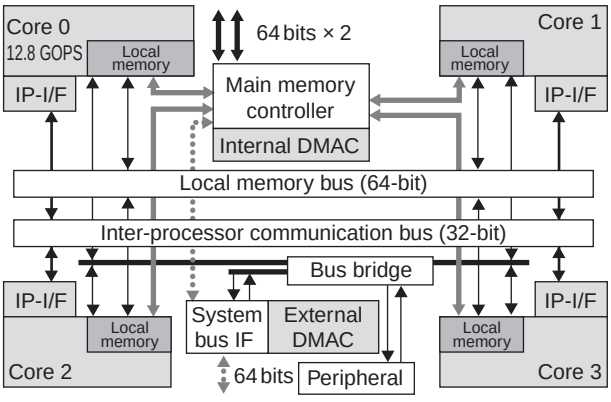
Figure 2 shows the block diagram of this processor. From an aggregated view, the chip consists of four processor cores, a main memory controller with two channels, a direct memory access controller (DMAC) for transfer between local memory units, a DMAC for external transfer, and a 64-bit system bus interface. When configuring the chip for a system that requires the processing of a huge volume of image data such as High Definition TeleVision (HDTV), a high-speed/high-precision printing system, or a graphics system, it is essential to provide high-speed data transfer capability including that for I/O access and between memory units. High-performance arithmetic processing capability is also very important. For example, in an arithmetic instruction, if it takes 90 cycles to transfer data from memory to the arithmetic unit and 10 cycles for the processor core to execute the operation, then data transfer controls 90% of total processor capability. Although the memory bandwidth is not necessarily a cause of concern regarding a single processor, it becomes a major factor for a multicore processor.

Consequently, we adopted the configuration described below to prevent data transfer from posing a bottleneck in terms of performance capability.

1) Four processor cores interconnected by a crossbar bus are employed, with two 64-bit channels of the main memory interface operating at 266 MHz and one system bus interface operating at 178 MHz.
2) To reduce external memory access, each processor core is equipped with 128 KB SRAM (local memory unit) as local storage.
3) The DMA controller is functionally divided into a DMAC for internal data transfer (internal DMAC) and a DMAC for external data transfer (external DMAC), and each DMAC runs independently at the same time. The internal DMAC is used to control data transfer between processor cores, each processor core and external memory, and different memory units. The external DMAC is used to control data transfer between memory and the system bus.
4) In addition to the crossbar bus used for data transfer as described above, there is a built-in, inter-processor communication control feature for transferring commands between processors.

A dedicated crossbar bus connects the local memory unit built into each processor core. Thus, all cores can access all local memory units. The internal DMAC and external DMAC described above are both equipped with 16 channels. Given this bus architecture, the FR1000 processor can simultaneously process data transfer between the built-in local memory units, data transfer between areas in memory, and data transfer between memory and an external device. As shown in **Figure 3**, the data transfer speed between memory and an external device reaches up to 1 GB/s.

Programs to be run on a multicore processor generally require a means of inter-processor communication. Due to data transfer, the time needed for communication between processors is



DMAC: Direct Memory Access Controller

Figure 2
Block diagram of FR1000.

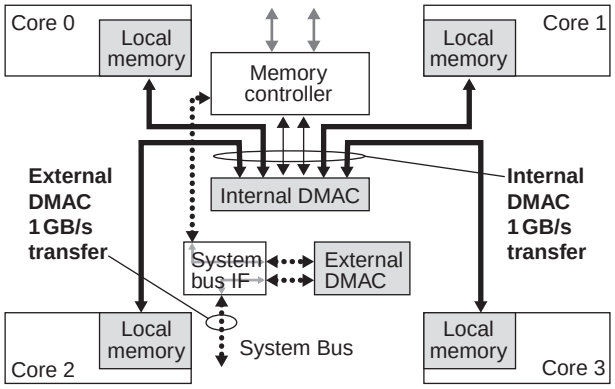


Figure 3
Data transfer performance.

approximately 10 times that required for arithmetic instructions, and thus may become a bottleneck. Therefore, the FR1000 processor is provided with a mechanism dedicated to communication between processors for reducing communication overhead and enabling inter-processor communication independently of data transfer. Moreover, each core has a message buffer dedicated to inter-processor communication that can send 4-byte message data to another core within nine core cycles.

In addition, for program optimization, each core has a hardware Performance Analyzer counter (PA-COUNT) dedicated to measuring performance for analyzing the bottleneck factor while the processor is running. This counter is configured with five 40-bit counters, and can measure the availability ratio of a processor and the stall factor ratios according to the counter's settings while an application program is actually running.

3. FR1000 software environment

The software development environment for the FR1000 processor is realized by enhancing SOFTUNE for the multicore processor.[4] SOFTUNE is a software development environment developed by Fujitsu for the FR-V processor. This new environment supports the use of existing software development methodology for a single processor. **Figure 4** shows the SOFTUNE software development environment for the FR1000 processor. As shown in the figure, several parts have been enhanced for a multicore processor.

Objects can be generated in each core by using the environment for existing single processors. The following two functions have been enhanced for use with a multicore processor.

1) Function that makes a program (commonly used in multiple cores) resident in a shared library.
2) Function that first generates a program object using the program development environment for single processors, and then allows four objects for individual cores to be integrated into one object by using the object integration tool.

Figure 5 shows an image of program debugging by SOFTUNE that has been enhanced for a multicore processor. The In Circuit Emulator (ICE) server has been newly developed to control communication between multiple single-processor debuggers and processor cores for a multicore processor debugger. By combining the use of the ICE server and SOFTUNE debugger in each core, a program running on each core can be debugged using the existing method. Moreover, the activity status of each core is displayed in the core status window on the monitor screen shown in Figure 5.

The operating system for the FR1000 processor is based on REALOS and enhanced for a multicore processor.[5),6)] REALOS is a real-time OS for the FR-V processor that conforms to µITRON specifications, in which the run unit of software is defined as a task. Communication between tasks on a single processor uses Service Call that is already provided for single processors. We have enhanced the existing µITRON standard to provide Service Call for a multicore processor in our own MP-library (**Figure 6**), because the µITRON standard does not currently define a multicore processor. REALOS and Service Call library (MP-library) for a multicore processor

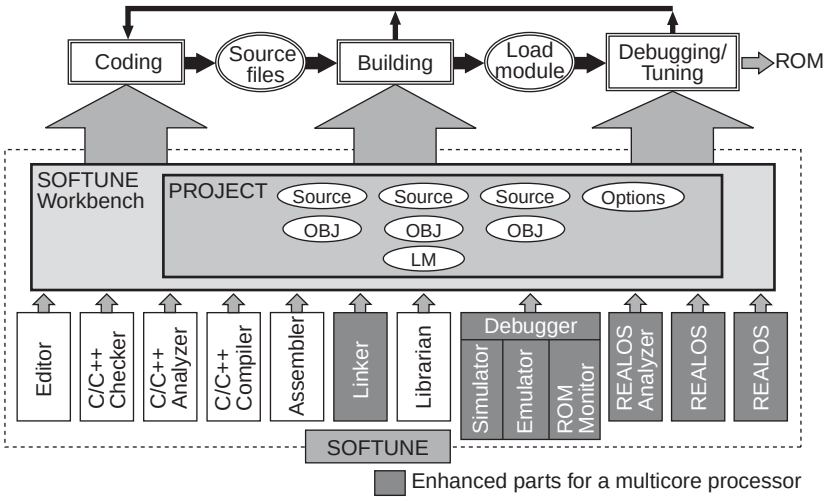


Figure 4
FR1000 software environment: SOFTUNE.

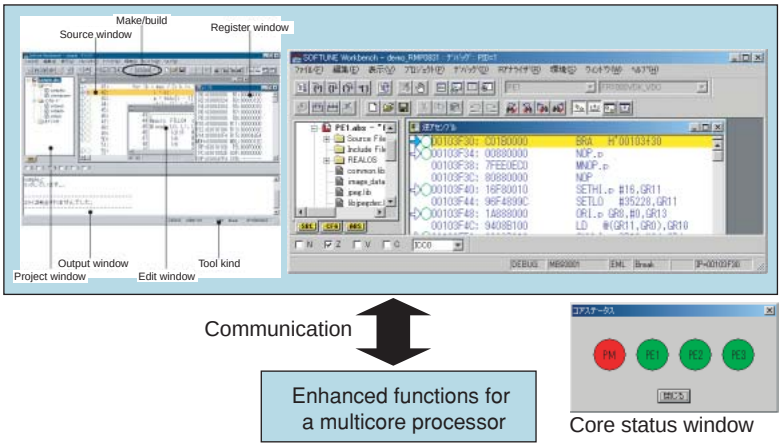


Figure 5
Program debugging environment.

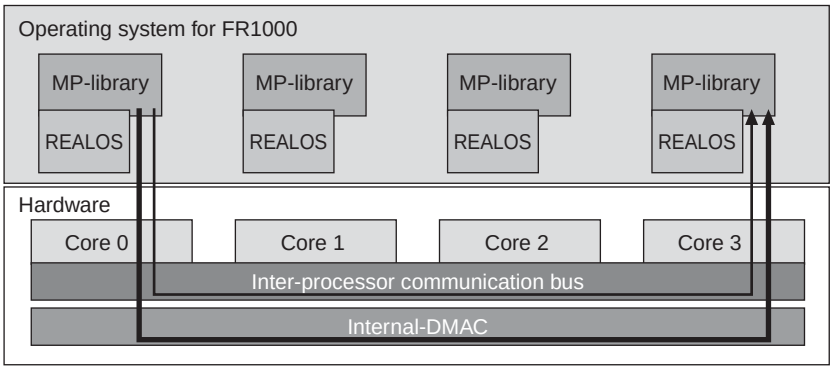


Figure 6
Operating system for FR1000.

coexist and work together on each processor core. Thus, this arrangement allows the user to do the following:

1) Use the existing REALOS Service Call to communicate between tasks within the processor, and
2) Use the new enhanced Service Call to communicate between tasks on different processors.

As a result, a user can now select the existing or new Service Call for communication between processors.

4. Tuning of application programs

Application programs are generally optimized to make full use of a processor's capabilities. This is because it may not be easy to increase the level of performance unless the programs written for a multicore processor are optimized. **Figure 7** shows a flowchart for optimizing the capability of an application program running on a multicore processor. The left side of Figure 7 shows the flow of optimization for a single processor; the right side shows that for a multicore processor.

Three tools are used for program optimization with the FR-V processor: REALOS Analyzer, Sampler, and PA. REALOS Analyzer measures the task availability ratio. Sampler measures the frequency of functions that are executed. PA is used for optimization at the command level. The following describes how these tools are used to optimize the system.

The first step of tuning operation to increase the capability of application is optimization at the algorithm level for a single processor. For example, tasks performed at the C language level, such as the amalgamation of different functions, reduce the number of unnecessary processes. The next step involves finding functions whose execution imposes a heavy processing load by measuring the frequency ratio of each function. A sampler is provided for the FR-V processor to find such functions. The sampler checks the function being executed at the specified time interval, and then displays the result as a count based on the data collected. As shown in **Figure 8**, the output window displays the ratio of each function to the total execution of functions. From this information, it is easy to identify those functions that impose a heavy processing on the processor. These

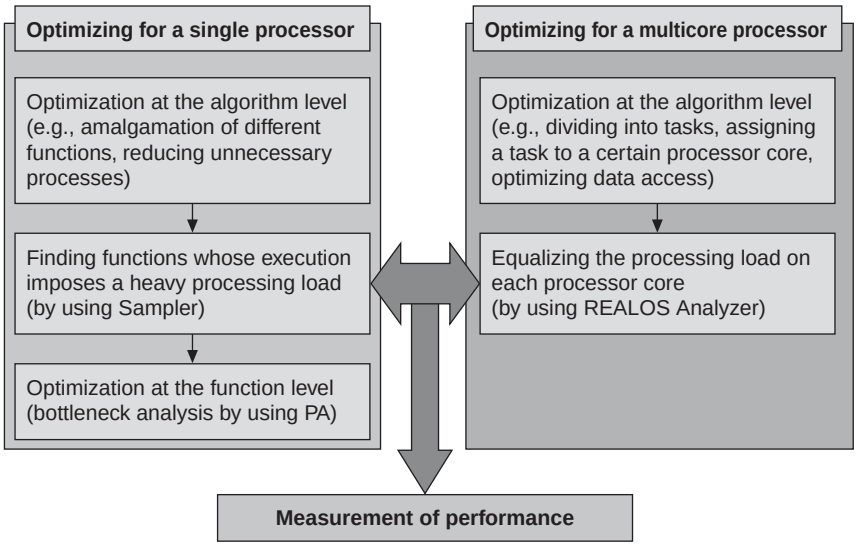


Figure 7
Tuning flow on FR1000.

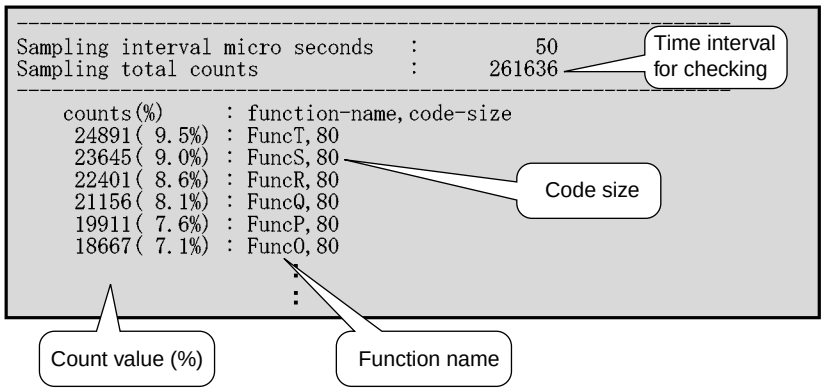


Figure 8
Output of function processing load: program tuning environment (Sampler).

are the functions selected for optimization.

Although it was understood that the system could be further optimized to increase availability of the processor cores by measuring the availability ratio of each processor core and the interlock factors, the existing embedded processor could not perform such optimization because, unlike the CPU in a supercomputer, the embedded processor lacks the hardware necessary to measure performance while running an actual application program. Fortunately, the FR-V processor is provided with such hardware to measure performance. The PA tool was developed for this purpose. When PA is used to evaluate performance, the hardware activates the measuring device by providing a dedicated library. The following items are checked:

- Instruction Level Parallelism (ILP)
  (Total number of valid instructions in VLIW)
- VLIW Per Cycle-VPC
  (Ratio of VLIW instructions executed without a stall)
- Operation Per Cycle-OPC
  (Ratio of instructions executed without a stall)
- Interlock factors
- Hit rate of cache
- Frequency distribution of input instructions

Figure 9 shows sample measurement results obtained by PA. The contents of measured data can be displayed in the form of a graph and also as diagnostic results of the bottleneck factor(s). Thus, these indicators are used to optimize an application.

Optimization at the multicore processor level must start during that within the functions after single processor optimization is under way, as shown in Figure 7. At this stage, however, the fact that the scope of optimization is not focused on one processor alone must be kept in mind. In the FR1000 environment, a program is divided into tasks based on the µITRON standard as described above. Specifically, a program must be divided into tasks, with each task then assigned to a certain processor core for execution. At such time, it is crucial to make modifications and adjustments at the algorithm level, such as equalizing the processing executed by each processor and minimizing the communication overhead between processors.

Figure 10 shows the algorithm used for MPEG-2 MP@HL decode processing that we actually applied to a multicore processor. In Core 0 a bit stream is entered and the header analyzed. One picture consists of multiple slices. When Core 0 detects the head of a slice, it activates Core 1 through Core 3 and processes the slice. Core 0 is placed in the wait state until all image processing has been completed. Each picture is processed in this manner. Even though performance and

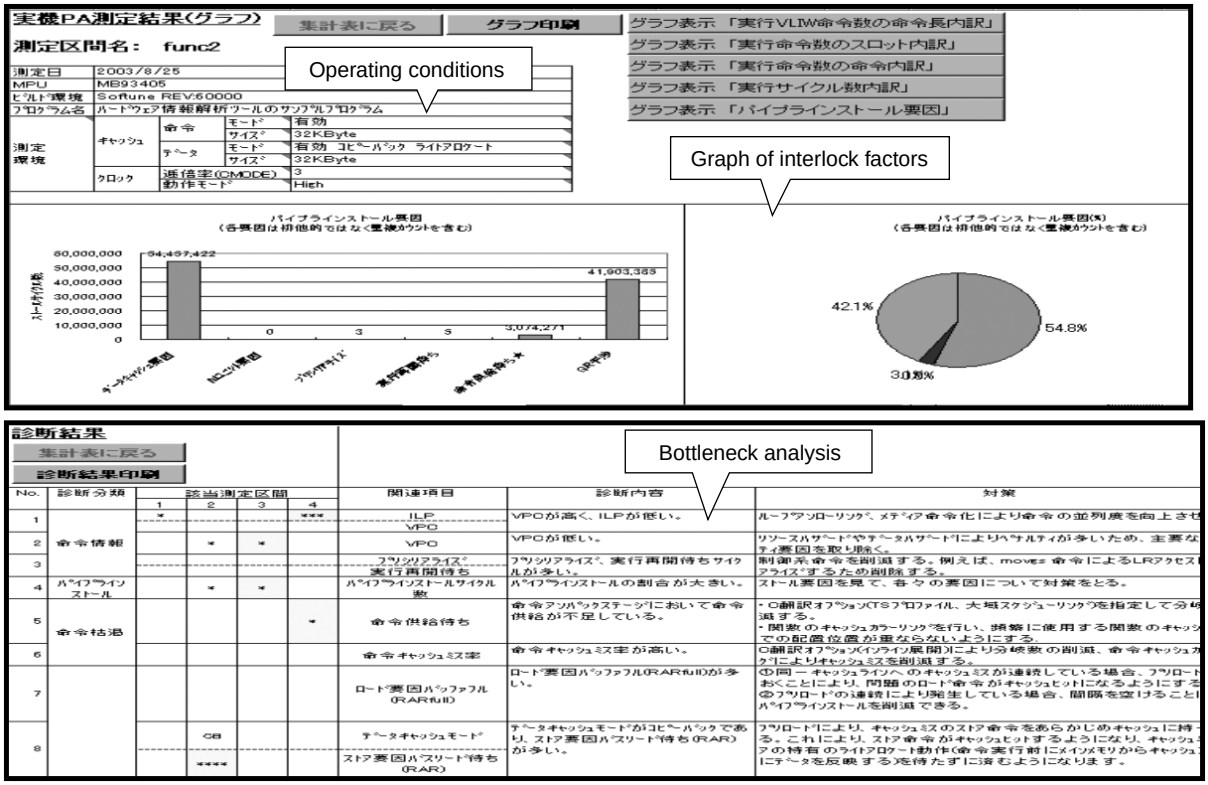


Figure 9
Results of PA analysis tool: program tuning environment.

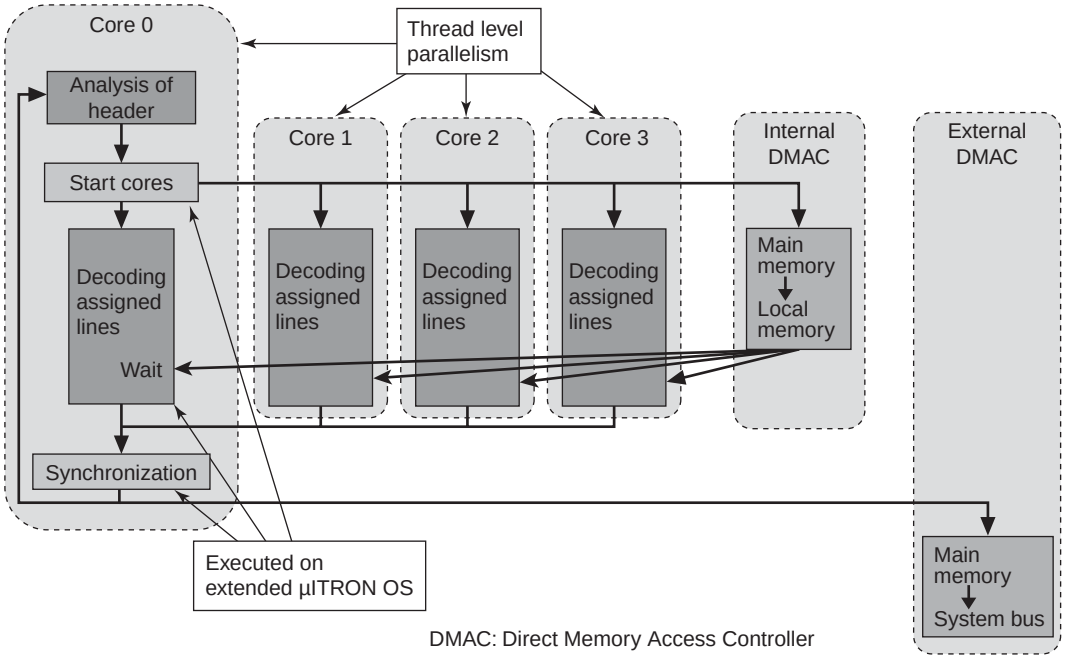
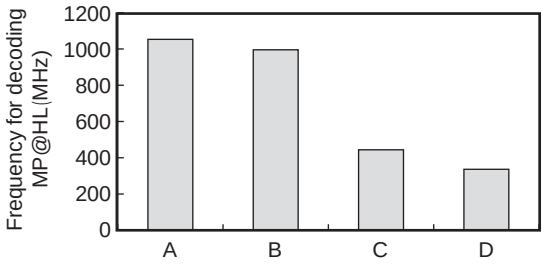


Figure 10
Parallel processing of MPEG-2 MP@HL streaming data on four cores.

its attainment can be recognized by measuring a specific task executed in a specific core, and how much time it takes to do so, all such information can be obtained by a task performance measuring tool called REALOS Analyzer run on the µITRON OS on the FR-V processor. REALOS Analyzer can analyze the dynamic stack usage by each task in addition to the execution time and availability ratio of each task. **Figure 11** shows the sequence diagram of the MPEG-2 decoding. Graphs are shown in the sequence of Core 0, Core 1, Core 2, and Core 3, starting from the top. The black parts indicated for Core 0 denote the execution time of header analysis. The other bar graphs denote the processing times for decoding the slice layer. From Figure 11, we can see that the heaviest processing load is imposed on Core 0, followed by that on Core 1, regardless of our efforts to balance the workload in each core except Core 0. In particular, the availability ratio of Core 2 is about 60% and that of Core 3 about 50%. Therefore, we can understand that the processing performed in Core 0 must be optimized or the workload shifted from Core 0 to Core 2 or Core 3 to increase total capability. It is important, however, to understand that this unbalanced load is caused by the characteristics of image data used for testing. In other words, the balance of processing load varies depending on the image to be processed. Conse quently, more image data should be collected before drawing further conclusions.

Figure 12 shows the REALOS Analyzer performance optimization results for the current MPEG-2 MP@HL decoding program. The vertical axis represents the operating frequency required for decoding MPEG-2 MP@HL. First, we applied MP@HL to the MPEG-2 MP@ML decoding processing for a single core. Upon executing the decoding on a FR1000 core, we found that a processing frequency exceeding 1 GHz was needed (Bar A in Figure 12). We then showed the measurement results of performance when testing the same models that had been modified for use with a multicore processor by using the method shown in Figure 10 (Bar B in Figure 12).



A : Decoding MP@HL by a single core
B : Decoding MP@HL by 4 cores
C : Decoding MP@HL by 4 cores with changing memory map to equalize memory access of each channel
D : Decoding MP@HL by 4 cores with data access processing separated from the core

Figure 12
Performance optimization results for MPEG-2 MP@HL decoding program.

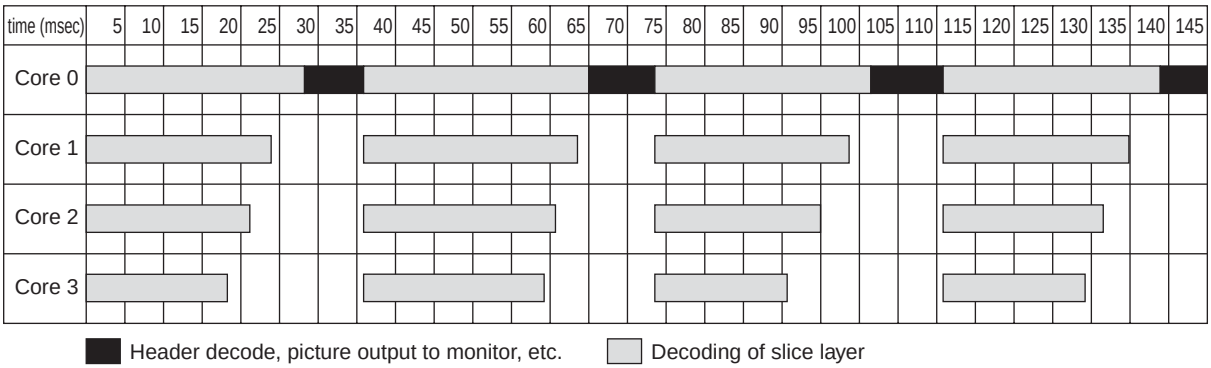


Figure 11
Sequence diagram of MPEG-2 MP@HL decoding.

However, we could not increase performance at all by simply modifying the program for use with a multicore processor. We still needed a processing frequency in excess of 1 GHz. After analyzing the situation, we found that performance did not increase by simply modifying the program because a heavy load was imposed by memory access. This condition could be attributed to many functions simultaneously accessing the same memory unit in the multicore environment. Accordingly, to reduce the memory access load, we changed the memory map for MPEG-2 decode processing and equalized the load imposed by accessing data in memory as well. Thus, we were able to double the performance relative to that before this adjustment, and execute decoding at a processing frequency of approximately 500 MHz (Bar C). Furthermore, we could increase performance even more by separating data access processing from the core by using built-in local memory. We were ultimately able to execute MPEG-2 MP@HL decoding at a frequency of approximately 380 MHz (Bar D).

5. Conclusion

The FR1000 is designed to achieve 51.2 GOPS, 1.0 GB/s-DMA using four processor cores, an internal DMAC, external DMAC, two 64-bit channels of main memory interfaces operating at 266 MHz, and a system bus interface operating at 178 MHz. By using the FR1000, we have demonstrated MPEG-2 MP@HL stream decoding at a power dissipation of 3.0 W without using dedicated circuits.

References

1) H. Okano et al.: An 8-Way VLIW Embedded Multimedia Processor Built in 7-Layer Metal 0.11 µm CMOS Technology. ISSCC Dig. Tech. Papers, 2002, p.374-375.
2) Fujitsu: FR-V Media Solution. (in Japanese). *http://img.jp.fujitsu.com/downloads/jp/jed/brochures/catalog/a07000301j.pdf*
3) Shiota et al.: A 51.2GOPS, 1.0GB/s-DMA Single-Chip Multi-Processor Integrating Quadruple 8-Way VLIW Processor. ISSCC Dig. Tech. Papers, 2005, p.18-19.
4) Fujitsu: SOFTUNE Comprehensive development environment. (in Japanese). *http://edevice.fujitsu.com/jp/catalog/pdf/a07000207j.pdf*
5) Sakamura Laboratory: ITRON Project Archive. *http://www.sakamura-lab.org/TRON/ITRON/home-e.html*
6) Fujitsu: SOFTUNE REALOS Series. (in Japanese). *http://edevice.fujitsu.com/jp/catalog/pdf/a071000912j.pdf*

Atsuhiro Suga, *Fujitsu Laboratories Ltd.*
Mr. Suga received the M.E. degree in Electrical Engineering from Yokohama National University, Yokohama, Japan in 1989. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1989. He is a senior researcher at the System LSI Development Laboratories of Fujitsu Laboratories, Ltd. His research interests include microprocessors and their related architecture for digital consumer products.

E-mail: suga.atsuhiro@jp.fujitsu.com

Satoshi Imai, *Fujitsu Laboratories Ltd.*
Mr. Imai received the B.S. and M.S. degrees in Electrical Engineering from Tokyo Institute of Technology, Tokyo, Japan in 1997 and 1999, respectively. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1999. Since then, he has been engaged in research and development of microprocessors.

E-mail: imai.satoshi-02@jp.fujitsu.com