An 800-MOPS, 110-mW, 1.5-V, Parallel DSP for Mobile Multimedia Processing

Hiroyuki Igura, *Associate Member, IEEE,* Yukihiro Naito, Kenya Kazama, Ichiro Kuroda, *Member, IEEE,* Masato Motomura, *Member, IEEE,* and Masakazu Yamashina, *Member, IEEE*

***Abstract*— This paper presents a newly developed parallel digital signal processor (DSP) for mobile multimedia processing. The DSP achieves 800 MOPS at 110 mW (1.5 V) through its task-parallel processing on four DSP cores. The parallel architecture, including data sharing and synchronization mechanisms, is carefully designed to be hardware and power efficient for portable multimedia applications. By using the parallel processing architecture, clock gating, and other low-power methods, about 85% power reduction is achieved. The 9.2-mm-square die contains 5.2-M transistors with 0.25-μm CMOS process.**

***Index Terms*—Digital signal processors, multimedia communication, parallel processing.**

I. Introduction

With the progress of multimedia technologies and applications, requests for portable multimedia terminals are increasing rapidly. To answer the increasing requests, very high signal-processing performance and very high programmability are required. However, high-speed signal-processing units in a terminal ordinarily require large power consumption. This limits a battery's life and consequently its portability. In short, the signal-processing units should meet the following three requirements: 1) high performance for the processing of video-class wide-band digital signals, 2) low power for extended battery life, and 3) programmability for the need to cope with various applications with a small chip count. It is a difficult task to meet these contradictory requirements at the same time, however.

Fig. 1 plots the power consumption and the processing performance of conventional digital signal processors (DSP's) [1]–[11]. DSP's recently presented at conferences (ISSCC, CICC) are labeled with their feature size. From this figure, it is clear that high-performance-oriented DSP's have power consumption larger than 1 W, and low-power-oriented DSP's have performance smaller than 200 MOPS. That is to say, conventional DSP's are lacking in performance, while emerging media processors consume too much power.

This paper presents a DSP that exploits the task-level, coarse-grained parallelism inherently existent in multimedia applications. This chip achieves the required high performance in a power-efficient manner while maintaining the programmability of conventional DSP's.

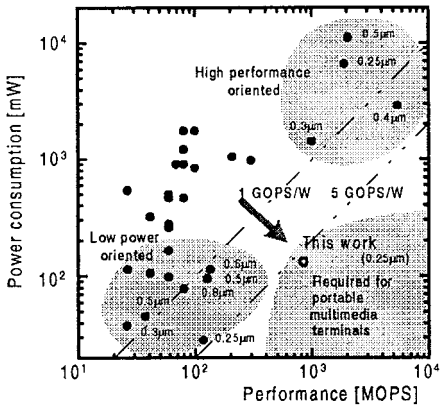


Fig. 1. Performance versus power consumption of conventional DSP's and present work.

We have estimated that for use in portable multimedia terminals, the power consumption of a DSP must be at most 500 mW, and the performance must be at least 400 MOPS. (This performance corresponds to that for MPEG1 video recording.) The development of our DSP has been specifically targeted to this application area.

II. Chip Overview

To reduce the power consumption of a DSP, we employed a parallel processing architecture. By using the parallel processing architecture, we can make the clock frequency and supply voltage low. Also, we can make the transistors for use in the DSP small. Consequently, the power consumption is greatly reduced.

Fig. 2 shows our parallel DSP architecture. This chip contains four sets of autonomous (and symmetric) 16-bit DSP cores. Each DSP core has its own locally connected 64-kb instruction cache memory and 64-kb X/Y data memory. Four sets of data buses connect on-chip shared resources, such as a 128-kb, eight-bank shared memory, peripheral blocks, and an external data memory controller, to each of the DSP cores. The peripheral blocks are composed of interface controllers, timer, status controller, memory controller, resource controller, interrupt controller, and direct memory access (DMA) controller.

The interface controllers (e.g., serial interface, host interface, and parallel ports) are used for exchanging data between the DSP chip and the other chips. The timer is

Manuscript received April 8, 1998; revised October 6, 1998.

H. Igura, M. Motomura, and M. Yamashina are with Silicon Systems Research Laboratories, NEC Corp., Sagamihara, Kanagawa 229 Japan (e-mail: igura@mel.cl.nec.co.jp).

Y. Naito and I. Kuroda are with C&C Media Research Laboratories, NEC Corp., Miyamae-ku, Kawasaki, Kanagawa 216 Japan.

K. Kazama is with the Microcomputer Division, NEC Corp., Sagamihara, Kanagawa 229 Japan.

Publisher Item Identifier S 0018-9200(98)07374-0.

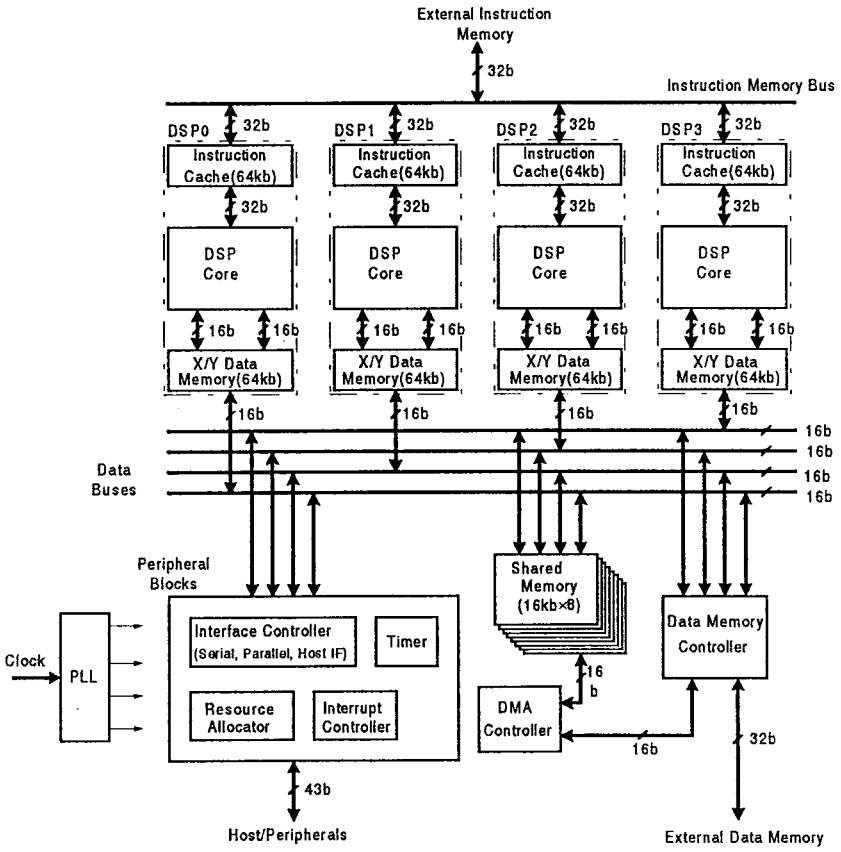


Fig. 2. Parallel DSP block diagram.

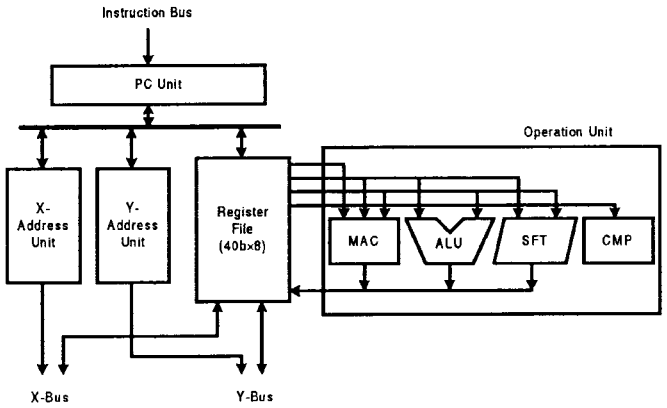


Fig. 3. DSP core block diagram.

used for timing synchronization and periodic generation of interrupt signals. The status controller controls the status of DSP's. The memory controller sets the access modes (DRAM access, wait timing, and so on) for external memories. The resource controller controls the assignment of shared resources such as the interface controllers, a shared memory, and an external memory. The interrupt controller is used for generating interrupt signals. The DMA controller controls the direct memory access between the shared memory and the external data memory.

Fig. 3 shows the DSP core architecture. This DSP core has a program control unit, a register file, an operation unit, and two addressing units. This operation unit includes a 16-b multiplier-accumulator, a 40-b arithmetic logic unit (ALU), a 40-b shifter, and a 40-b comparator. This DSP core can execute one ALU, two load/store, and two addressing operations for each cycle. The performance of each DSP core is 50 million instructions per second (MIPS), which corresponds to 200 MOPS. (Because of operation coupling limitations, the effective number of operations per cycle is lower than the maximum value. The MOPS value means effective megaoperations per second for typical signal-processing applications.) The instruction set of this DSP core is based on that of our conventional DSP [12].

III. Parallel Architecture

To realize effective processing, a DSP must have an architecture that is suited to its applications. Based on several studies of the target applications (e.g., H.263 video codec and G.723 voice codec system), the proposed parallel architecture is optimized so as to fully exploit their coarse-grained parallelism. In this section, several specific architectures required for processing in parallel are illustrated.

A. Memory Organization

Fairly large on-chip memories, i.e., 32-kB instruction and 48-kB data memories, have been integrated to capture most of the memory references under the task-parallel execution, so that slow and power-consuming off-chip memory accesses are minimized.

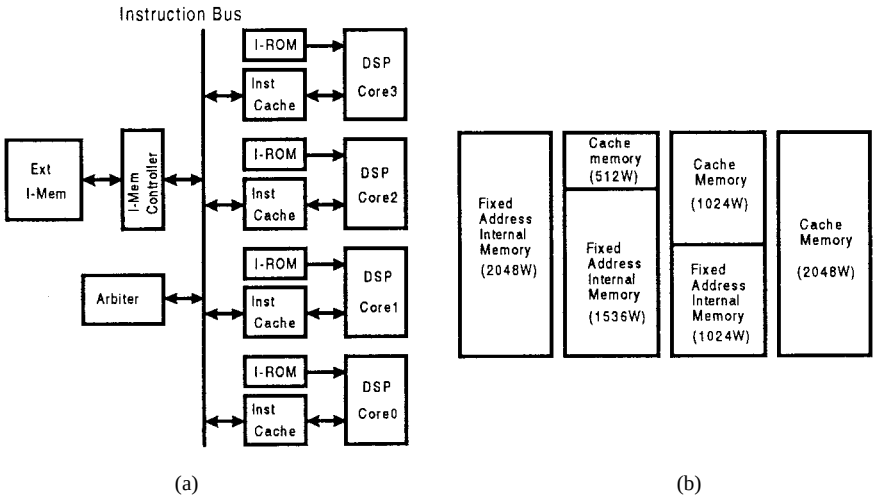


Fig. 4. Instruction cache mechanism. (a) Instruction bus architecture. (b) Instruction memory configuration.

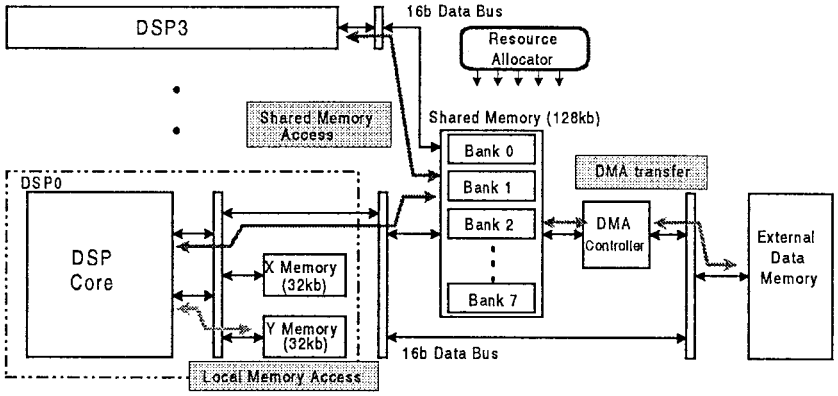


Fig. 5. Hierarchical memory architecture.

1) Flexible Instruction Cache: For parallel processing systems, the mechanism for serving the instruction code is important because every DSP core needs an individual and continuous instruction stream for task distribution. Fig. 4(a) shows the architecture of instruction memories. We adopted a separated cache mechanism for instruction memories.

An instruction cache memory is connected to each DSP core. The cache memory has a direct mapping algorithm and 128-b line size. Every cache memory is connected to an instruction bus. And the instruction data in the external memory are served through the instruction bus.

Fig. 4(b) shows the configurations of the cache memory. A part or all of the cache memory can be transformed to a fixed-address internal instruction memory. There are four configurations of the cache memory: one entirely cache memory, one entirely fixed address internal instruction memory, and two types of mixed configurations. By storing the repeatedly used instruction codes into the fixed-address internal instruction memory, cache miss can be reduced.

2) Hierarchical Data Memory: The data memory space of the DSP core is separated into three levels: local, shared, and external data memory. These data memories are designed with a hierarchical architecture as illustrated in Fig. 5. Each memory has individual characteristics. For example, the local memory is small in size but low in power consumption. On the other hand, the external memory has high power consumption but a large memory size. By storing repeatedly used data to the local memory, the total power consumption will be reduced.

Because each DSP core has two data buses and the local data memory is separated into two memories (the X-memory and the Y-memory), a DSP core can simultaneously access any two of the four memories, i.e., the X-memory, the Y-memory, the shared memory, and the external memory.

Shared memory is used for inter-DSP core data transfer and DMA transfer. By using DMA transfer and by prefetching the external data, the access latency caused by the external memory access will be reduced. Because the shared memory is separated into eight banks, a DSP core can access a bank that the other DSP cores are not using.

Because the DMA controller used for DMA access has a four-level DMA queue, DMA queue-control mechanisms (stop, start, swap, and so on), and status-informing mechanisms (interrupt at end of DMA transfer, DMA status register, and so on), the DMA transfer is executed effectively without DSP core performance overhead.

By using this hierarchical memory architecture, high access and power efficiency are achieved. This architecture also enables flexible data sharing among DSP cores and the DMA

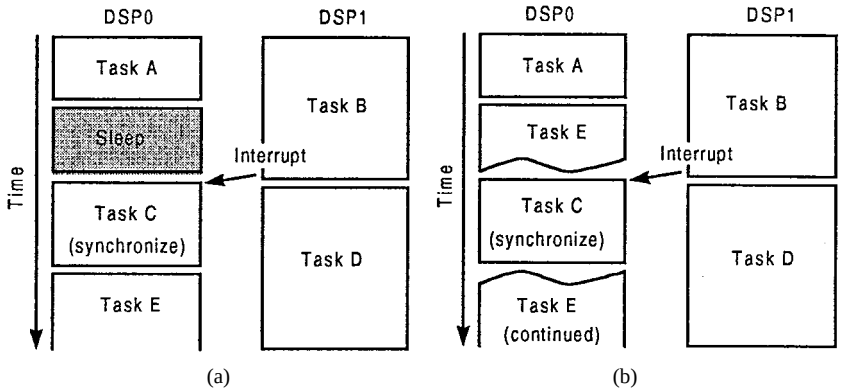


Fig. 6. Synchronization by interruption.

controller without inducing too much hardware overhead, as compared to a flat memory architecture with an extensive cross-bar switch [1].

B. Inter-DSP Synchronization by Interruption Mechanism

Fig. 6 explains the inter-DSP synchronization method. Because the synchronization method strongly affects the performance (including number of transistors, power consumption, programmability, and so on), the type of synchronization mechanism used between the processing cores is very important for parallel processing systems. We used a simple mutual-interruption mechanism for the inter-DSP core synchronization.

When a task (task B in Fig. 6) is finished, the DSP core 1 initiates an inter-DSP core interrupt instruction, and this results in the generation of an interrupt signal for DSP core 0. A synchronization routine initiated by the interrupt signal handles a flag that is used for checking the end of the task B. The tasks in the DSP core 0 use this flag for synchronization with the DSP core 1.

Considering that synchronization, e.g., handshaking before changing an owner of a shared memory bank, is a rare event in the proposed task-parallel execution, this simple mechanism is both sufficient and hardware efficient.

C. Resource Allocation

Assignments of the on-chip shared resources (peripheral blocks, each bank of shared memory, external memory controller, and so on) are controlled by a resource allocator. Every on-chip shared resource can be used in two types of resource allocations: a dynamic allocation and a static allocation. Both types are controlled by the resource allocator. In a static allocation, a DSP core requests the resource allocator to occupy the shared resource, and the resource is allocated to the DSP core. Once a resource is allocated statically, it cannot be accessed by the other DSP cores until the resource is released. Resources that are not allocated specifically are allocated and released dynamically by the accesses from DSP cores. Resources that are used continuously and exclusively (e.g., interface controllers) are suitable for use in static allocation. The static allocation is also used for locking.

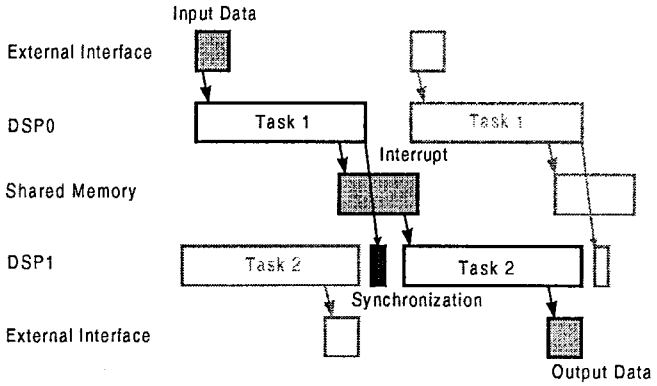


Fig. 7. Parallel processing dispatch method.

IV. PARALLEL PROCESSING

To make the DSP cores execute a program in parallel, the program for the DSP must have extra codes for mutual interaction between the DSP cores. In coding such a parallel program, careful consideration must be made for task assignments to each DSP core and data-transfer rates. In this section, we will introduce the parallel processing dispatch methods.

A. Dispatch Method

Fig. 7 illustrates one type of parallel processing dispatch method. Suppose there is a program composed of tasks 1 and 2, and we assign the tasks to DSP cores 1 and 2, respectively. Because task 2 processes the data that have been processed by task 1, task 2 must wait for task 1 to finish. To inform task 0 of the end of task 1, DSP core 1 initiates an interrupt signal to DSP core 0. Upon receiving the interrupt signal, DSP core 0 calls a synchronization routine that manages task flags and so on, and uses these flags to control the task execution.

Tasks 1 and 2 are processed in the same way as a pipeline. A shared memory is used for transmitting the processed data between the DSP cores, and the flags inform the DSP of the validity of the data on the shared memory. To add codes that handle task queuing to the synchronization routine, task queuing functions are provided.

One of the easiest ways to implement a program on parallel DSP's is to divide a sequential program and attach synchronization codes at the boundary of the tasks. Fig. 8 shows an example of this implementation sequence.

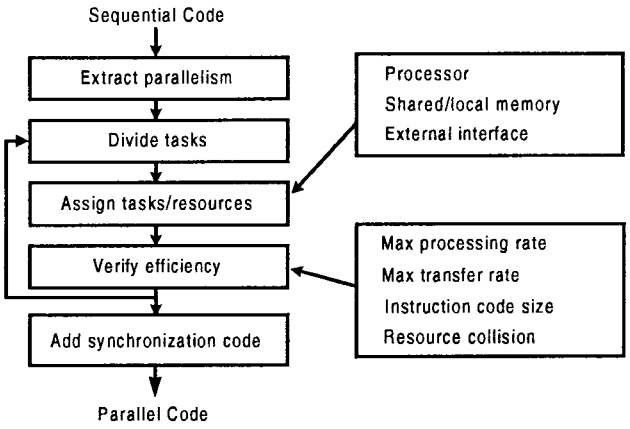


Fig. 8. Parallel program implementation sequence.

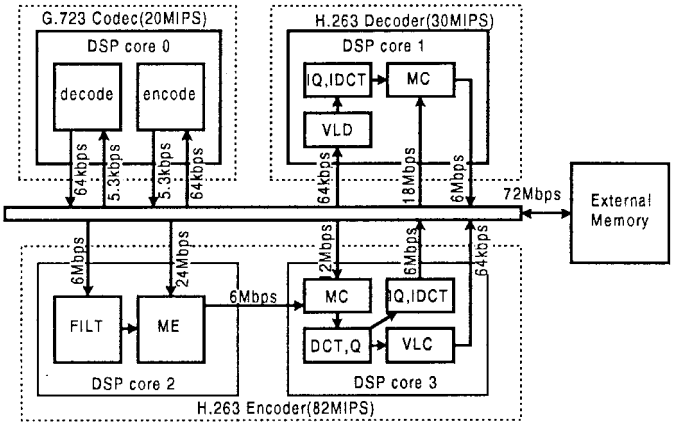


Fig. 9. Task assignment example (H.263 + G.723 system).

First, extract inherently existent parallelism in the sequential program for a single DSP. Next, divide the program so that the sets of tasks are assigned to the DSP cores equally. Assign the shared resources to the tasks and evaluate the processing efficiency and reasonableness of the assignment. Last, add the synchronization and transmit codes.

In this example, the tasks are statically assigned. Because the cache miss and reassignment of shared resources are reduced, power consumption is lower in static task assignment than in dynamic assignment, as long as the tasks are divided equally.

B. Task Assignment Example

Fig. 9 shows the task assignment and the data-transfer rate of an H.263 + G.723 system on the parallel DSP. The external memory access traffic amounts to 72 Mbps. Since the external memory interface of the chip is 32-b wide and 50 MHz, the peak external memory bandwidth reaches 1.6 Gbps with zero-wait SRAM's, 800 Mbps with one-wait SRAM's, and 200 Mbps with seven-wait DRAM's. This analysis shows that the external memory bandwidth of the parallel DSP is sufficient for this particular application.

Fig. 10 explains how one of the target applications, a set of an H.263 video codec (QCIF, 15 fps, 64 kbps) and a G.723 voice codec, is conducted on the parallel DSP. The DSP0 handles all the voice-codec and system-management

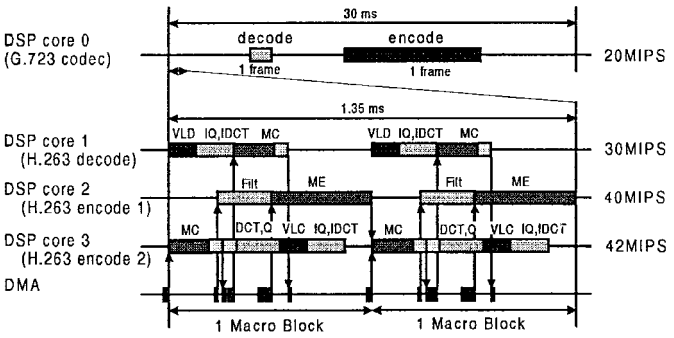


Fig. 10. Execution diagram of H.263 codec (QCIF, 15 fps, 64 kbps) and G.723 codec.

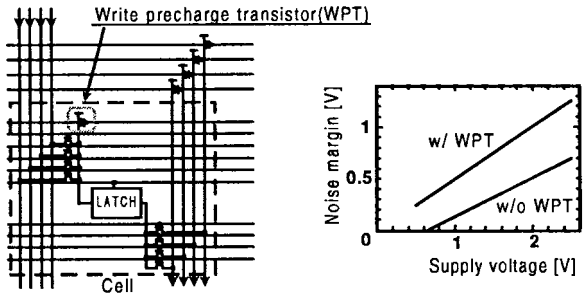


Fig. 11. Register file for low-voltage operation. (a) Multiport memory circuit diagram. (b) Noise margin (simulated).

tasks, which amount to 20 MIPS in total. The DSP1 takes care of video decoding (30 MIPS), while the DSP2 and DSP3 cooperatively work on video encoding (40 MIPS and 42 MIPS, respectively). Voice-codec tasks are in the millisecond range because of the protocol's frame-oriented processing, while macroblock-oriented video-codec tasks are around 100-ms long. The vertical arrows in the figure represent synchronization associated with data sharings among DSP cores and the DMA controller. (The motion-estimation search algorithm used in our H.263 encoder is a two-step search algorithm, and the search range is $-15\sim+15$ for horizontal and $-11\sim+11$ for vertical.)

Since the synchronization take place only at coarse-grained task boundaries, synchronization overhead is negligible. The total DSP utilization for this application is more than 70%. The synchronization overhead for the H.263 and G.723 system is actually very small, less than 2%.

C. Instruction ROM

Each DSP core includes an instruction ROM that has codes for booting and system services. The system service routines are used for providing services that involve inter-DSP core communications, e.g., storing instruction codes in the other cores, setting up all sorts of modes, acquiring statuses, resetting for rebooting, and transferring data. These system service routines are initiated by an interrupt signal.

V. Circuit Techniques

The effectiveness of the parallel DSP architecture enables the chip to achieve targeted performance at reduced supply voltage—namely, 200 MIPS (800 MOPS) peak performance at

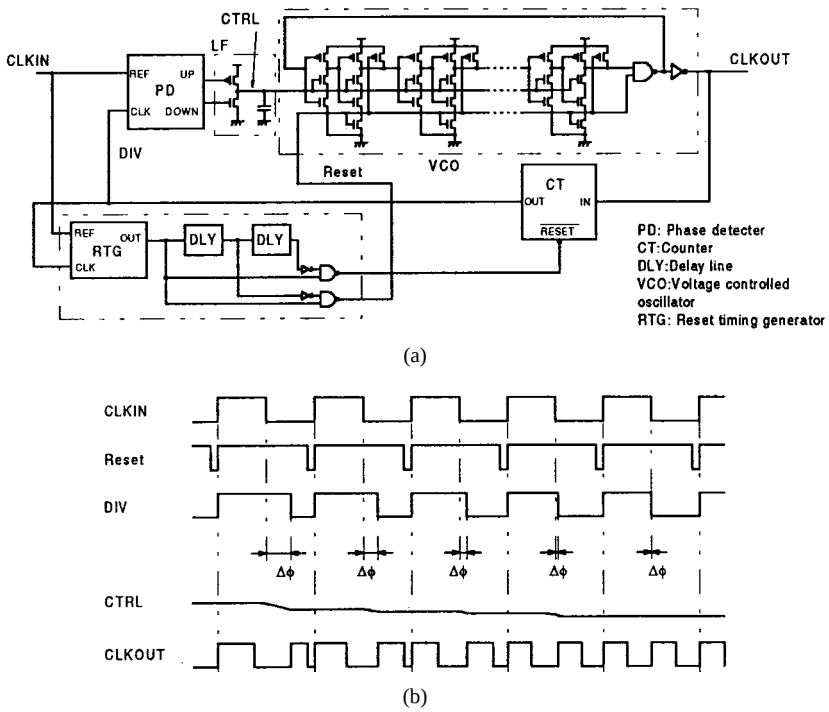


Fig. 12. Phase-correction-type PLL circuit. (a) Block diagram. (b) Phase-correction mechanism.

1.5 V—with the use of 0.25-μm CMOS technology. However, the reduced supply voltage requires circuit design techniques to increase the noise margin of key circuits such as memory, register file, and phase-locked loop. In this section, we will introduce several circuit techniques we have developed for reducing its power-supply voltage. In addition, several other circuit techniques for reducing power consumption will be introduced.

A. Supply Voltage Reduction

Since a multiport register file, in general, uses single-ended data lines for area efficiency, it suffers from reduced write noise margin due to the V_t drop caused by access nMOS transistors. As illustrated in Fig. 11(a), our register file includes an additional pMOS transistor in each memory cell. By precharging the cell through this transistor before write operations, the write noise margin is improved as shown in Fig. 11(b).

Fig. 12(a) shows the phase-correction-type PLL circuit we developed. This circuit has high stability against external parameter fluctuation because the circuit can cancel the integral component of the phase error accumulated into the voltage controlled oscillator (VCO) by the internal reset signal created periodically. Fig. 12(b) illustrates the phase-correction mechanism. Because the positive edge of the reset signal forces the VCO to reoscillate, further accumulation of the phase error is avoided.

B. Signal Transition Ratio Reduction

The power effectiveness of the parallel DSP can be further improved by eliminating unnecessary signal transitions on capacitive nodes.

Fig. 13(a) shows our newly developed clock control architecture. Since clock distribution accounts for a significant portion of total power dissipation, we have employed a three-level conditional clock distribution scheme: chip level, core level, and block level. The chip-level and core-level schemes use dedicated instructions and external/internal interruptions to explicitly trigger clock-off and clock-on, respectively. The block-level clock control, on the other hand, is implicitly initiated when an instruction decoder finds functional blocks that are unused by a current instruction.

Fig. 13(b) shows the block-level clock control mechanism. The clock signal for pipeline registers is controlled by the control signals generated from an instruction decoder. When a data path following a particular flip-flop (FF) is not activated by the instruction, the clock signal for that FF is suspended. The control signals are latched at the 3/4 cycle of the decoding stage by using a delayed clock (DCLK) to avoid glitches on the clock.

In addition, we have also designed the data buses hierarchically, so that frequently accessed shared resources, such as the shared memory, have smaller bus capacitance than the infrequently accessed ones. Arrival timings of control signals to turn off/on the clock and the shared bus are carefully designed so as to avoid glitches.

Fig. 14 shows the hierarchical data-bus structure. To reduce the power dissipation in data buses, we place frequently accessed shared resources, such as the shared memory, at capacitance nodes smaller than those for infrequently accessed resources. The control signals to cut off the data stream are generated from the predecoder in the previous pipeline stage. Thus, the control signals can open or close the bus latches before the data stream reaches them so that no data-stream leaks occur.

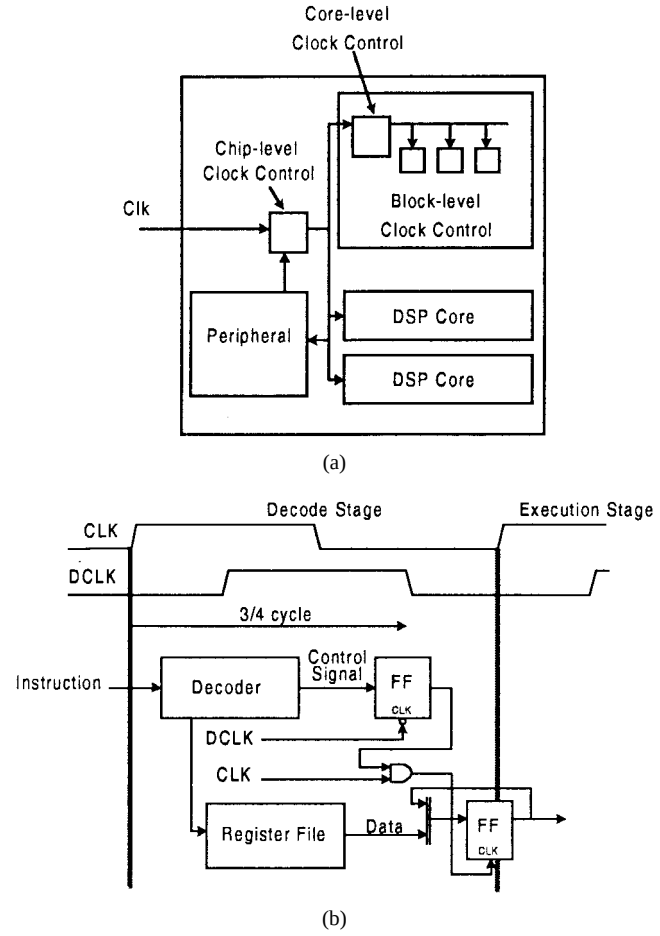


Fig. 13. Clock control mechanism. (a) Three-level clock mechanism. (b) Block-level clock control.

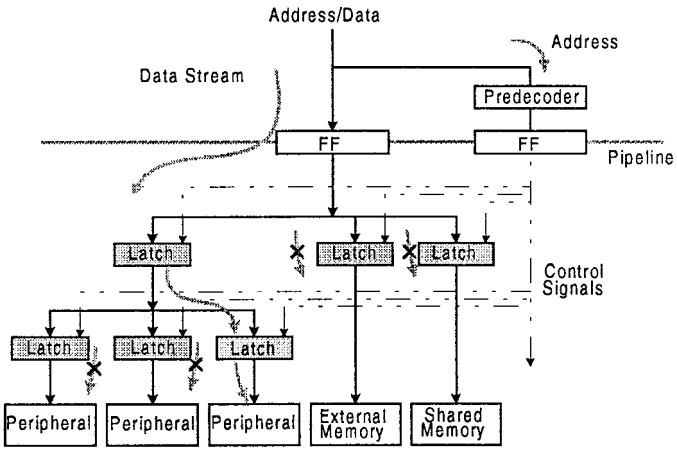


Fig. 14. Data-stream control mechanism.

C. Transistor Size Reduction

As mentioned before, we also used small-sized transistors to reduce the power consumption. The average gate width of nMOS and pMOS transistors for the logic gates is 1.76 and 2.20 μm, respectively, which we believe is less than half the width of conventional normal LSI's.

VI. Discussions

Fig. 15 analyzes power reductions made available with the present parallel DSP for the H.263 + G.723 system

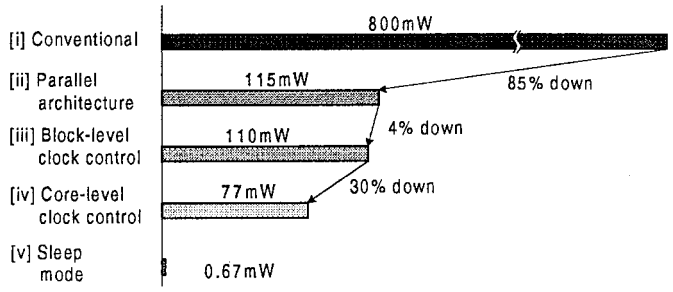


Fig. 15. Power reductions achieved by developed schemes.

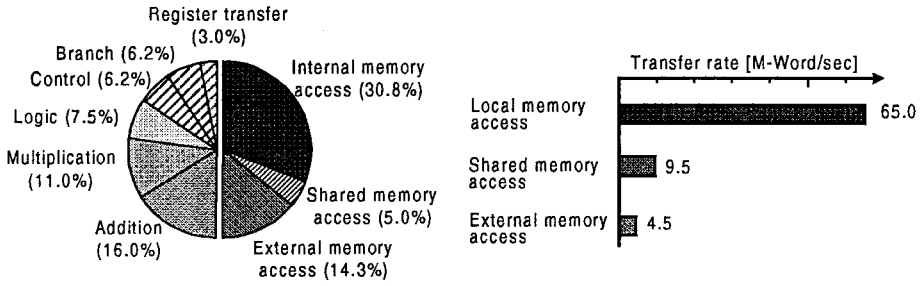
mentioned before. Since a conventional single DSP solution (case [i]) would have required four times the clock frequency with increased power supply voltage, the parallel architecture alone (case [ii]) achieves 85% power reduction. (Because the effectiveness of the parallel execution depends on the applications, the actual power reduction may be lower than this.) The block-level and core-level clock controls reduce the power consumption further by 4% (case [iii]) and 30% (case [iv]), respectively. Note that 110 mW (case [iii]) and 77 mW (case [iv]) are the maximum and average power-dissipation values of the chip for this application. Last, the chip-level clock control puts the chip into a sleep mode with 0.67-mW dissipation (case [v]).

Fig. 16(a) shows power mix by operation types (for case [iv] and including a 1-Mb, 3.3-V SRAM as the external data memory) and the distribution of memory access traffic. (This graph includes the power consumption by external SRAM.) Fig. 16(b) shows the transfer rates of different types of memory accesses. The memory accesses dominate (about 50%) the total power consumption because of their large instruction ratio. The external memory access has a particularly large power-consumption ratio (about 14%) in spite of its low traffic rate, which illustrates the largeness of the power consumption with an external memory access and the importance of reducing the number of external memory accesses.

Fig. 17 shows a power mix by element of the DSP without clock control. The power consumption by clocking and flip-flops is dominant (about 53%). This result shows the possibility of effectively reducing power consumption by means of clock control. In our work, the power-consumption reduction by block-level clock control was not large because the number of controlled flip-flops is limited. Increasing the number of objects to be controlled will further reduce the power consumption.

Fig. 18 shows the effect of prefetching with DMA access. In this figure, execution efficiency with and without the prefetching are plotted. As can be seen from this graph, the larger the external memory wait cycle is, the more effective the prefetching is. As an example, the execution efficiency is improved about 20% when the external memory wait cycle is seven.

Fig. 19 shows a micrograph of the chip, and Table I summarizes its key features. Four DSP cores (DSP0–DSP1) are placed vertically. Peripheral blocks and shared memory are placed on the other side and connected to the cores by data buses. The chip contains approximately 5.2-M transistors on a



(a) (b)

Fig. 16. Memory-access power-consumption analysis. (a) Power-consumption mix by operation types. (b) Distribution of memory traffic.

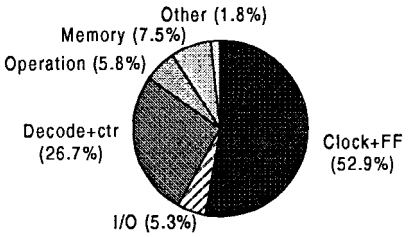


Fig. 17. Power-consumption mix by blocks.

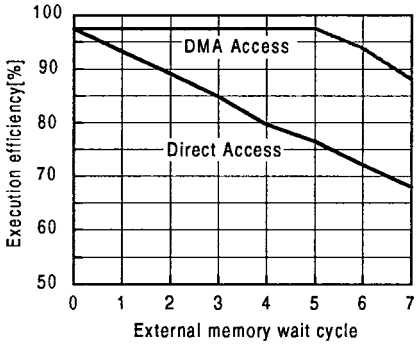


Fig. 18. Improvement by DMA access (simulated).

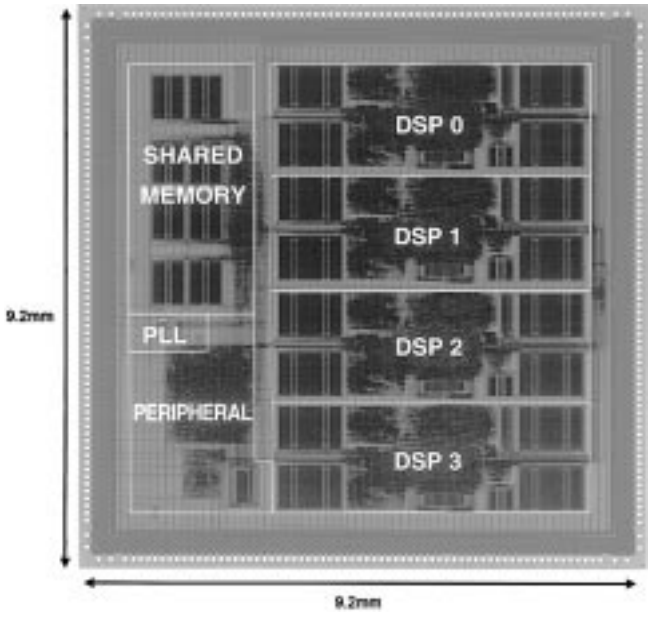


Fig. 19. Chip micrograph.

TABLE I
PARALLEL DSP FEATURES

Clock Frequency	50MHz
Power Consumption	110mW (@1.5V, 50MHz), 20mW (@0.9V, 20MHz)
Process	0.25μm, 3-layer Al, CMOS process
Number of Tr	5.2M
Supply Voltage	1.5V (internal), 2.5V (external)
Performance	800MOPS (@1.5V), 320MOPS (@0.9V)
Area	9.2mm × 9.2mm
Number of Pins	208 pin
Internal Instruction Memory	2048W × 32bit × 4 (256kbit)
Internal Data Memory X/Y memory Shared memory	 2048W × 16bit × 2 × 4 (256kbit) 2048W × 16bit × 4 (128kbit)

9.2-mm—square die and achieves 800 MOPS at 1.5-V power supply (320 MOPS at 0.9 V).

VII. CONCLUSIONS

This paper has presented a parallel DSP that employs a task-level, coarse-grained parallel DSP architecture. The DSP includes four DSP cores and a shared memory for data transfer between the DSP cores and peripheral blocks. Several key architectures (e.g., distributed instruction cache mechanism, hierarchical data-bus structure, inter-DSP core interrupt mechanism for synchronization and DMA transfer) have been developed for effective processing and lower power consumption. Additional circuit techniques (e.g., three-level clock control) for lower signal-transition ratio have also been developed to reduce the power consumption further.

In this paper, it has been shown that the proposed parallel architecture can meet the performance and power-consumption levels required in portable multimedia terminals. Since only minor modifications will be required to implement this architecture into conventional DSP programs, the proposed approach will be able to provide a cost-effective solution to multimedia systems.

ACKNOWLEDGMENT

The authors would like to express their deep gratitude to Dr. H. Abe, Dr. T. Nishitani, and Dr. M. Fukuma for their encouragement and support throughout the course of this work.

REFERENCES

[1] K. Balmer, N. Ing-Simmons, P. Moyse, I. Robertson, J. Keay, M. Hammers, E. Oakland, R. Simpson, G. Barr, and D. Roskell, "A single chip multimedia video processor," in *Proc. IEEE Custom Integrated Circuits Conf. '94*, pp. 91–94.

[2] H. Yamauchi, Y. Tashiro, T. Minami, and Y. Suzuki, "A highly-parallel single-chip DSP architecture for video signal processing," in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing '91*, pp. 1197–1200.

[3] "Buyer's guide to DSP processors," Berkeley Design Technology, Inc., 1994.

[4] H. Igura, M. Izumikawa, K. Furuta, H. Ito, H. Wakabayashi, K. Nakajima, T. Mogami, T. Horiuchi, and M. Yamashima, "A 0.25-μm CMOS 0.9-V 100-MHz DSP core," *IEEE J. Solid-State Circuits*, vol. 32, pp. 52–61, Jan. 1997.

[5] S. Mutoh, S. Shigematsu, Y. Matsuya, H. Fukada, and J. Yamada, "A 1 V multi-threshold voltage CMOS DSP with an efficient power management technique for mobile phone application," in *IEEE 1996 ISSCC Dig. Tech. Papers*, pp. 168–169.

[6] D. Brinthaupt, J. Knobloch, J. Othmer, B. Petryna, and M. Uyttendaele, "A programmable audio/video processor for H.320, H.324 and MPEG," in *IEEE 1996 ISSCC Dig. Tech. Papers*, pp. 244–245.

[7] B. DeLoore, P. Lippens, P. Eeckhout, H. Huijgen, A. Loning, B. McSweeney, M. Verstraelen, B. Pham, G. de Haan, and J. Kettenis, "A video signal processor for motion-compensated field-rate up conversion in consumer television," in *IEEE 1996 ISSCC Dig. Tech. Papers*, pp. 248–249.

[8] M. Kurokawa, A. Hashiguchi, K. Nakamura, H. Okuda, K. Aoyama, T. Yamazaki, M. Ohki, M. Soneda, K. Seno, I. Kumata, M. Aikawa, H. Hanaki, and S. Iwase, "5.4GOPS linear array architecture DP for video-format conversion," in *IEEE 1996 ISSCC Dig. Tech. Papers*, pp. 254–255.

[9] W. Lee, P. Landman, B. Barton, S. Abiko, H. Takahashi, H. Mizuno, S. Muramatsu, K. Tashiro, M. Fusumada, L. Pham, F. Boutaud, E. Ego, G. Gallo, H. Tran, C. Lemonds, A. Shih, M. Nandakumar, B. Eklund, and I.-C. Chen, "A 1 V DSP for wireless communications," in *IEEE 1997 ISSCC Dig. Tech. Papers*, pp. 92–93.

[10] E. Iwata, K. Seno, M. Aikawa, M. Ohki, H. Yoshikawa, Y. Fukuzawa, H. Hanaki, K. Nishibori, Y. Kondo, H. Takamuki, T. Nagai, K. Hasegawa, H. Okuda, I. Kumata, M. Soneda, S. Iwase, and T. Yamazaki, "A 2.2 GOPS video DSP with 2-RISC MIMD, 6-PE SIMD architecture for real-time MPEG2 video coding/decoding," in *IEEE 1997 ISSCC Dig. Tech. Papers*, pp. 258–259.

[11] T. Yoshida, Y. Shimazu, A. Yamada, E. Holmann, K. Nakakimura, H. Takata, M. Kitao, T. Kishi, H. Kobayashi, M. Sato, A. Mohri, K. Suzuki, Y. Ajioka, and K. Higashitani, "A 2 V 250MHz multimedia processor," in *IEEE 1997 ISSCC Dig. Tech. Papers*, pp. 266–267.

[12] "μPD7701x FAMILY digital signal processor user's manual," NEC Corp., Oct. 1995.

Hiroyuki Igura (M'94–A'95) received the B.S. and M.S. degrees in electronic engineering from Kyushu University, Fukuoka, Japan, in 1990 and 1992, respectively.

He joined the System ULSI Research Laboratory, Microelectronics Research Laboratories, NEC Corp., Sagamihara, Japan, in 1992 and has been working on low-power microprocessor LSI's. His research interests include VLSI architecture, parallel processing, and high-speed circuit technologies. He currently is with the Silicon Systems Research Laboratories of NEC Corp.

Yukihiro Naito received the B.S. and M.S. degrees from the Tokyo Institute of Technology, Tokyo, Japan, in 1992 and 1994, respectively.

He joined NEC Corp., Kanagawa, Japan, in 1994. His research interests include image coding and architectures for programmable digital signal processors.

Kenya Kazama was born on January 28, 1970, in Hiroshima, Japan. He received the B.S. and M.S. degrees in energy engineering from Yokohama National University, Yokohama, Japan, in 1993 and 1995, respectively.

In 1995, he joined NEC Ltd., Kanagawa, Japan, where he has been working on the development of logic device process integration.

Ichiro Kuroda (M'98) received the B.E. degree in mathematical engineering and instrumentation physics from University of Tokyo.

He currently is a Principal Researcher with the Signal Processing Technology Group, C&C Media Research Laboratories, NEC Corp., Kanagawa, Japan. He has been working in the area of VLSI signal processing, especially on programmable DSP chips, tools, and application design.

Mr. Kuroda is a Technical Committee Member of Design and Implementation of Signal Processing Systems of the IEEE Signal Processing Society.

Masato Motomura (M'93) received the B.S. and M.S. degrees in physics and the D.E. degree from Kyoto University, Kyoto, Japan, in 1985, 1987, and 1996, respectively.

In 1987, he joined the Microelectronics Research Laboratories of NEC Corp., Kanagawa, Japan, where he worked on intelligent memory LSI architectures. He currently is with the Silicon Systems Research Laboratories. His research interests include processor architectures, low-power media processing, and memory-logic integration. He was a Visiting Researcher at the Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, from 1992 to 1993.

Dr. Motomura received the 1992 IEEE JOURNAL OF SOLID-STATE CIRCUITS Best Paper Award for his work on dictionary search processor LSI. He is a Member of the IEEE Computer Society and the Institute of Electronics, Information, and Communication Engineers of Japan.

Masakazu Yamashina (M'87) received the B.S., M.S., and Dr.Eng. degrees from the Tokyo Institute of Technology, Tokyo, Japan, in 1982, 1984, and 1993, respectively.

He joined NEC Corp., Kawasaki, Japan, in 1984. He was engaged in the research and development of CMOS video signal processor LSI and BiCMOS DSP LSI's. During a leave from NEC in 1989–1990, he studied improving algorithms for autonomous robots at Stanford University, Stanford, CA. He presently is a Research Manager at the System ULSI Research Laboratory, Silicon Systems Research Laboratories, NEC Corp., Kanagawa, Japan. His current research interests include high-speed microprocessors and high-speed circuit techniques.

Dr. Yamashina is a member of the Institute of Electronics, Information and Communication Engineers of Japan.