PAPER *Special Issue on Function Integrated Information Systems*

Dynamically Programmable Parallel Processor (DPPP): A Novel Reconfigurable Architecture with Simple Program Interface*

Boon-Keat TAN†, **Ryuji YOSHIMURA**†, ***Nonmembers***, **Toshimasa MATSUOKA**†, ***and*** **Kenji TANIGUCHI**†, ***Regular Members***

SUMMARY This paper describes a new architecture-based microprocessor, a dynamically programmable parallel processor (DPPP), that consists of large numbers of simplified ALUs (sALU) as processing blocks. All sALUs are interconnected via a code division multiple-access bus interface that provides complete routing flexibility by establishing connections virtually through code-matching instead of physical wires. This feature is utilized further to achieve high parallelism and fault tolerance. High fault tolerance is realized without the limitations of conventional fabrication-based techniques nor providing spare elements. Another feature of the DPPP is its simple programmability, as it can be configured by compiling numerical formula input using the provided user auto-program interface. A prototype chip based on the proposed architecture has been implemented on a 4.5 mm × 4.5 mm chip using 0.6 μm CMOS process.

key words: *CDMA bus, parallel processing, interconnection topology, routing flexibility, fault tolerant*

1. Introduction

As VLSI technology has advanced, implementing multiprocessor systems with many processors to achieve parallelism has become popular. Such multiprocessor systems should be implemented with a flexible interconnection topology in order to exploit maximum parallelism and fault tolerance. Several specific interconnection topologies such as mesh, tree, hypercube, and mesh-connected tree have been proposed [1], [2], however many of the proposed architectures are difficult to implement and require large switching circuits. In order to achieve efficient communication, large-scale parallel processing systems such as those reported in [3], [4] utilize special routing techniques and routing chips. As feature sizes shrink, the multiprocessor concept can be incorporated into the design of microprocessors to maximize parallelism. Recently proposed high-performance microprocessors, particularly those with data-path architectures, involve the use of several reconfigurable processing blocks that are interconnected via high-speed switching circuits [5], [6]. All processing blocks in such microprocessors are designed to perform parallel computing tasks, however, the design of the switching circuit becomes the limiting factor in these architectures. In addition, the complexity and difficulty of design increase dramatically as the number of processors increases.

In contrast, the interconnection topology adopted in this work is based on orthogonal sequences as in [7]. We have already successfully implemented a code division multiple-access (CDMA) bus interface for parallel processing systems [8] that utilizes the orthogonal pseudo-noise code sequence. The processor architecture proposed in this paper, a dynamically programmable parallel processor (DPPP), consists of a large number of simplified processing blocks that are interconnected via the CDMA bus interface. Unlike conventional parallel processing systems that depend heavily on the interconnection topology or switching matrix circuitry, the CDMA bus interface provides complete routing flexibility with only a small silicon area. Instead of using multi-function ALUs, each processing block in a DPPP carries out several simple functions. Together with its unique interconnecting topology, the proposed DPPP features high chip utilization, high interconnection flexibility, simple programmability without the need for conventional CAD tools, dynamic reconfigurability, low power consumption, and high fault tolerance. There has been a recent proliferation of yield-enhancing techniques involving the provision of spare elements during fabrication that can be activated to replace faulty primary elements [9]–[11]. However, the actual cost-effectiveness of such schemes is still in doubt. In contrast, a DPPP has high fault tolerance even without allocating spare elements.

2. Architecture

Figure 1 depicts the overall architecture of a DPPP, which consists of simplified ALUs (sALU) as processing elements. The sALUs are interconnected via a multiple-access bus proposed by one of the authors [8]. We named the processing blocks in a DPPP "simplified ALUs" because these blocks have relatively few arithmetic functions compared to conventional ALUs. All

Manuscript received February 13, 2001.
Manuscript revised May 15, 2001.

†The authors are with the Faculty of Engineering, Osaka University, Suita-shi, 565–0871 Japan.

*This paper was presented at "The 26th European Solid State Circuit Conference (ESSCIRC)" in Stockholm, Sweden, 18-22 September 2000.

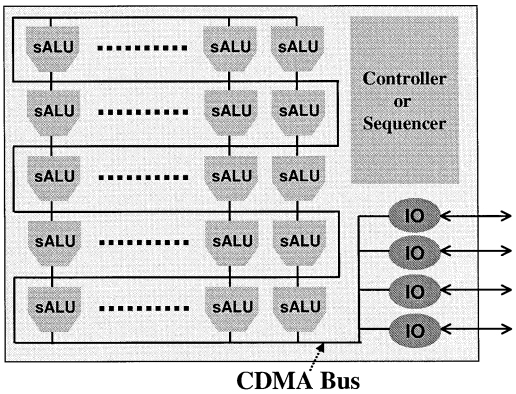


Fig. 1 Architecture of proposed DPPP, comprised of many sALUs interconnected via a CDMA bus.

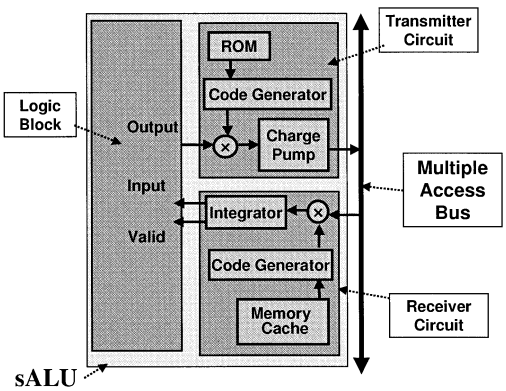


Fig. 2 Structure of sALU, consisting of a transmitter circuit, a receiver circuit and a logic block.

processing blocks in conventional data-path architectures are identical, providing equal processing ability; in the DPPP, some sALUs carry out a different set of tasks and therefore differ from each other. The input and output of the chip are connected to the multiple-access bus via the interface circuit. The multiple-access bus functions as a "virtual wire" that connects all processors regardless of location on the chip. The DPPP is based on the Harvard architecture, in which control signals are separated from data signals. Instructions for all sALU are loaded from an external sequencer via several serial control buses.

2.1 Multiple Access Bus Interface

Figure 2 is a schematic of the structure of a sALU. The sALUs consist of a logic block, a transmitter circuit and a receiver circuit; the transmitter and receiver circuits themselves consist of a pseudo-noise (PN) code generator, charge pumps, an integrator circuit, a mixer, and memories such as ROM and memory cache. Circuit details for the interface are described in [8], [12]. The serial data output of the logic block is modulated by a PN code, which is generated based on an address stored in ROM. Each sALU has a unique address that

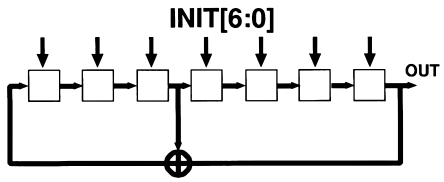


Fig. 3 Linear feedback shift register capable of generating 127 different PN codes.

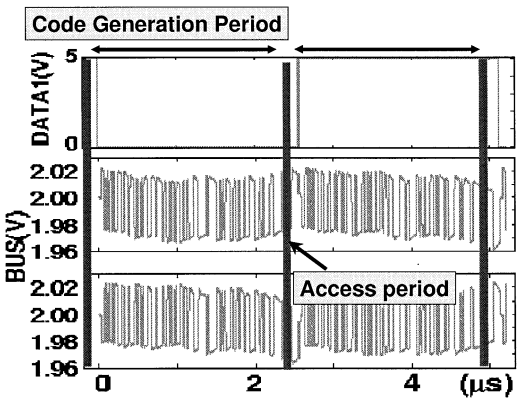


Fig. 4 Simulation of CDMA bus interface operation showing access period and code generation period.

differentiates it from others. The modulated signal is then charge-pumped to the multiple access bus. In order to receive the signal transmitted from a sALU, the address of the sALU should be used to demodulate the signal. The demodulated signal is retrieved via an integrator circuit and is then output to a logic block via the INPUT node, as shown in Fig. 2. The control signal VALID switches HIGH only when a signal is received. The dynamic programmability of DPPP can be easily understood by looking at how the PN codes are generated. Figure 3 shows the configuration of the PN code generator; a generator that consists of linear feedback shift registers (LFSRs) requires an initial value (INIT[6:0]) to create the PN sequence codes [13]. In the DPPP, the initial value for the PN code generator is the address of each sALU, stored in the memory cache or ROM. The address is accessed in the first 1/127 of each clock cycle (access period); the remaining time is the code generation period. During this period, all data is transmitted between processing elements simultaneously through the multiple-access bus using PN code modulation. The memory cache can be reloaded dynamically in the code generation period even when sALUs are functioning because the address is not being accessed.

Figure 4 shows a simulation of multiple-access bus operation with one sALU for simplicity. Note that the voltage swing of the signal is small; i.e. the peak-to-peak voltage of the signal is only 40 mV. Consequently, power consumption by the bus is much lower compared to conventional Sonic type (time-division) inter-

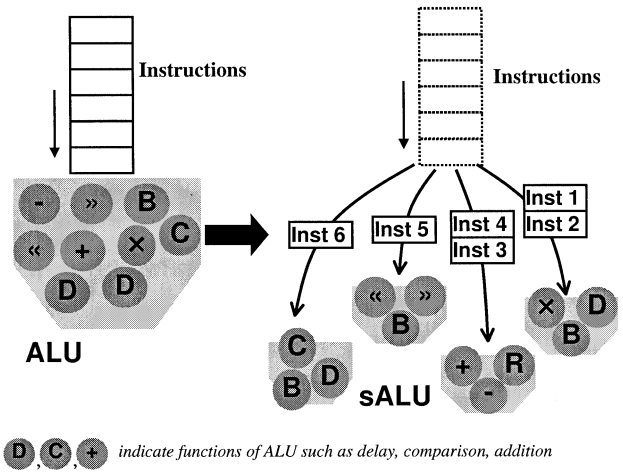


Fig. 5 Distributing functions of ALUs into simplified arithmetic blocks, sALUs, that support several simple arithmetic functions is advantageous for parallelism.

face. Transmissions using the multiple-access bus has been verified using earlier circuits consisting of 10 pairs of bus interface circuits. The bit error rate of the bus interface is appreciably low.

2.2 Logic Block

An ALU in conventional processors can perform many arithmetic functions. In contrast, in order to provide high interconnection flexibility, all functions in the DPPP are distributed in different processing blocks, as shown in Fig. 5. The sALUs incorporate several arithmetic functions such as addition, subtraction, multiplication, shift operation, delay operation, comparison, and bit-wise operations such as XOR and NOR. Distributing the functions into many different processing blocks results in high chip utilization and facilitates parallel processing. We have shown in previous work that assigning individual arithmetic functions to single processing blocks provides high chip area efficiency for specialized digital signal processing [14]. However, the above approach is not efficient for the design of general-purpose processors such as the DPPP because some functions are used with significantly less frequency. In the DPPP, arithmetic functions that give good area efficiency are grouped into the same sALU. For example, adders and subtractors share almost 90% of the hardware, and thus combining the two functions into a single sALU is desirable. Another criteria for grouping is that frequently used functions should be grouped with less frequently used functions.

As shown in Fig. 6, the logic block consists of a serial 8-bit ALU with only a few arithmetic functions, a control unit, a timing adjustment block, an overflow processing block and a latency controller. Depending on the functions of the sALU, the overflow processing block may not be needed. The timing adjustment block

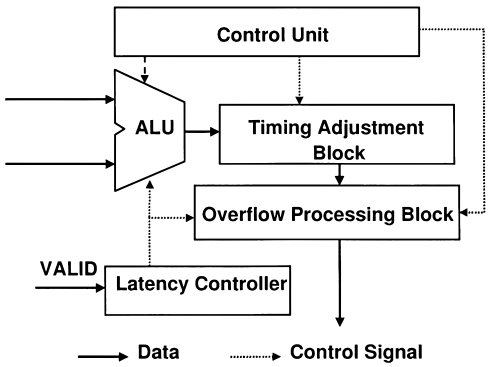


Fig. 6 Structure of logic block.

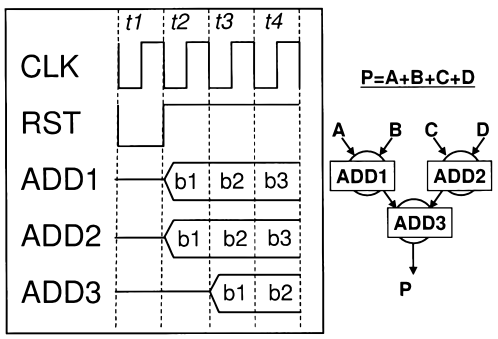


Fig. 7 Simple example of timing adjustment is done.

is required to synchronize arithmetic calculations because some functions require more than one cycle to process data. For example, serial addition requires one clock cycle for a given input whereas a compare operation requires 8 clock cycles. In addition, the starting bit for each processor may arrive at different timings. Figure 7 illustrates an example in which the sum of A, B, C, D is generated using 3 adders (ADD1, ADD2, ADD3). ADD1 and ADD2 operate immediately after reset, whereas ADD3 commences functioning one clock cycle later. This is achieved by using a latency controller, which detects the commencement of operation via a VALID signal.

The control unit stores the information required for the timing adjustment and overflow processing blocks. Dynamic reloading of setup information in the control unit is not possible due to the continuous access by the logic block and timing adjustment block. However, dynamic function changes can be achieved by switching the function to another sALU. In other words, once configured, all sALUs carry out a fixed function, but the connections between all functional processors can be reconfigured dynamically to achieve dynamic programmability.

3. How DPPP Works

As the DPPP is based on the Harvard architecture, the control signals and data signals are separated, as shown

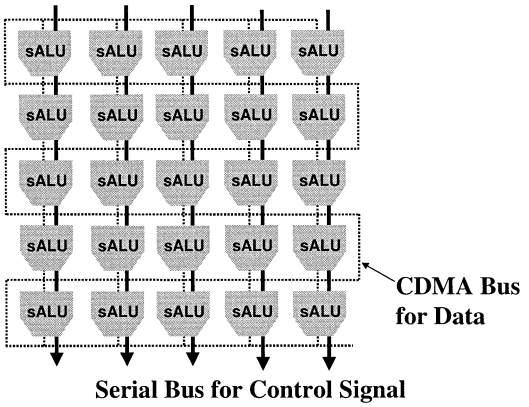


Fig. 8 Instruction sets are loaded via several serial buses while data signals transmitted via CDMA bus.

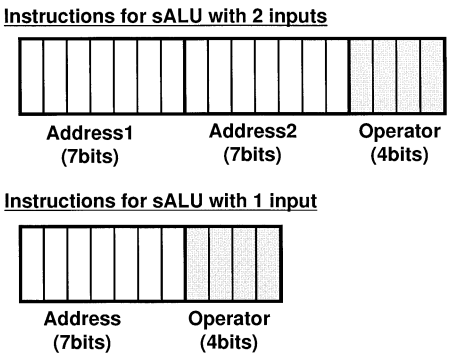


Fig. 9 Instruction format for each sALU.

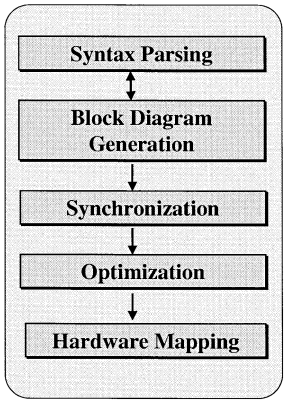


Fig. 10 Program algorithm for user interface.

```
Input:
p11, p12, p21, p22, q11, q12, q21, q22;

Operation:
o11= p11*q11 + p12*q21;
o12= p11*q12 + p12*q22;
o21= p21*q11 + p22*q21;
o22= p21*q12 + p22*q22;
if( o11 ≥ o12) m1= o11 : o12;
// computing maximum component

output:
o11, o12, o21, o22, m1;
```

Fig. 11 Program input for matrix calculation.

in Fig. 8. The data signals utilize the multiple-access bus while the control signal is transmitted using a number of serial buses. Initially, instructions for each sALU are loaded from the external sequencer. However, the instruction set can be reloaded during data processing through the dynamic programmability of the architecture. Figure 9 shows the instruction formats for representative sALUs. The instruction sets for sALUs are simple, consisting of only the addresses of the operand and the operator. The address is used to generate PN codes for the demodulation of data at the multiple-access bus interface. The operator selects the functions in each sALU and provides synchronization information for the sALU. Note that the address required in the instruction set corresponds to the output of each sALU. Therefore, data transfer between sALUs is carried out by selecting the corresponding address. Broadcasting of data to several sALU can be done by setting the same address.

4. Program Interface

A CAD auto-program generation interface using numerical formula input has been developed for the proposed DPPP. As shown in Fig. 10, the program first examines the syntax of the input text and checks for floating nodes. The program then computes the latency of each operation, and allocates each function to processing blocks. Lastly, optimization is performed to eliminate excess delay.

The design flow using the proposed program involves merely compiling the proposed CAD for the numerical formulas. The targeted algorithm can often be expressed in the form of numerical formula. Simple conditional expressions such as 'IF' are also supported. A sample of the input used in the auto-program generation interface is shown in Fig. 11. An IF expression is shown as an example. We evaluated the proposed DPPP and the auto-program generation interface using several conventional applications, as shown in Table 1. As shown by the results, the design and programming time using the DPPP is relatively short The proposed interface is much simpler than hardware designs such as FPGA, which involves the use of hardware description language (HDL). The number of sALUs utilized represents the number of parallel tasks performed in each application.

5. Fault Tolerant

The proposed architecture has high fault tolerance even without using any specific fault-rectifying technique. Fault tolerant is achieved by reallocating the task of

Table 1 Evaluation of program user interface.

Applications	Design Time	Program Time	No. of 'sALU's
CMA adaptive array (4 tap)	50 min	9 ms	34
FIR filter (5 tap)	15 min	8 ms	14
Counter (32 bits)	20 min	5 ms	7
IDCT (2 × 2 Matrix)	35 min	4 ms	12

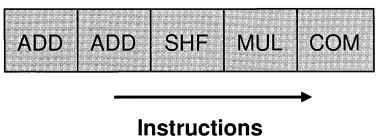


Fig. 12 Instruction sequence to be executed.

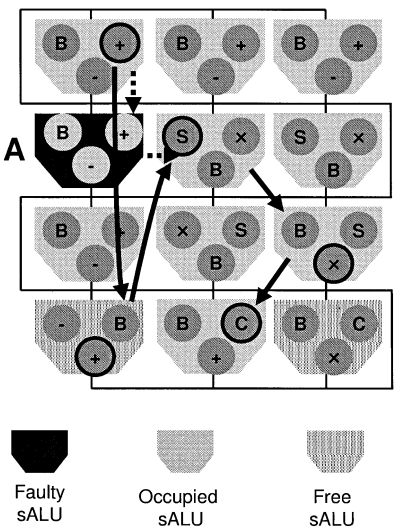


Fig. 13 If all sALUs surrounding the faulty block are occupied, the task can be reconfigured to another sALU much due to the absence of location limitations.

a fault processing block to any unoccupied sALU. For example, when instruction sets such as those shown in Fig. 12 are being executed by the DPPP with the configuration shown in Fig. 13, the instructions are executed by different processing blocks. The arithmetic functions used in each sALU are highlighted in Fig. 13. When the sALU labeled A becomes faulty, it can be replaced by any other free sALU. Note that this can be done regardless of whether the surrounding processing blocks are occupied. In conventional array processors, faulty processors cannot be replaced under these circumstances unless special interconnection topologies and spare elements are provided. This follows from the location-independence of the connections between sALUs in the DPPP.

In other words, the DPPP will continue to function correctly as long as there remain unoccupied sALUs to

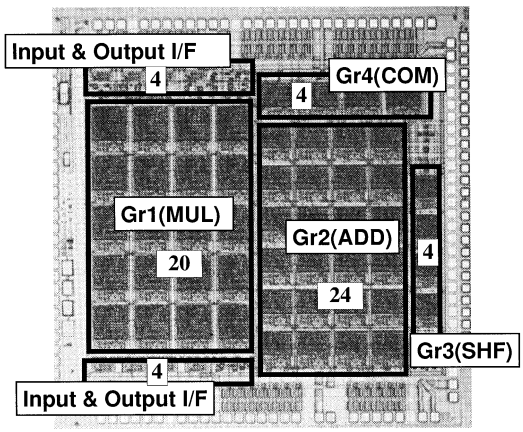


Fig. 14 Micrograph of prototype chip consisting of 60 sALUs fabricated on 4.5 mm × 4.5 mm silicon.

Table 2 Composition of processing blocks

sALUs (numbers)	Primary function	Sub-functions
Gr1(MUL) (20)	Multiplication	Delay Block, AND, OR
Gr2(ADD) (24)	Addition Subtraction	Delay Block, NOT, XOR
Gr3(SHF) (4)	Shift Operation	Shift 1 bit2 bit right Shift 1 bit2 bit left
Gr4(COM) (4)	Comparison	Max, Min, Delay Block

replace faulty sALUs. The occurrence of faulty sALUs will not kill the entire chip, reducing only the flexibility in configuring computing task. Another feature of the DPPP in terms of fault tolerance is that rerouting and reconfiguration of the faulty sALU to its replacement can be done easily and dynamically by changing the receiver codes of the affected sALUs.

6. Chip Implementation

As a feasibility check of the proposed architecture, a DPPP was implemented into a 4.5 mm × 4.5 mm prototype chip using a 0.6 μm triple-metal, double polysilicon CMOS process. A micrograph of the chip is shown in Fig. 14. The DPPP prototype chip has 8 inputs, 8 outputs, and 52 processing blocks that can be divided into four groups as shown in Table 2. Each sALU carries out a primary function and several subfunctions. Grouping was selected so as to realize high chip density. The multiple bus access interface operates at 200 MHz. The performance of the chip is 106MOPs when parallel processing is fully exploited. As all processing blocks is able to function simultaneously, the chip supports 52 parallel tasks. Note that due to high density of the proposed architecture, number of processing elements incorporated in the prototype chip is relatively large compared to conventional architectures.

The number of different sALUs is chosen arbitrary,

as shown in Table 2 taking into consideration that some functions such as addition and multiplication are more frequently used than others. However, the selection of composition should be optimized based on a statistical analysis of how frequently each arithmetic function will be used in typical algorithms or applications. For example, when the prototype chip is configured to execute a 16-tap FIR filter program, the utilization of hardware resources is 90% (16 multipliers, 15 adders, 16 delay blocks). In addition, optimal throughput is guaranteed as all tasks can be executed in parallel. This is realized through the provision on the prototype chip of a sufficient number of adders for a corresponding number of multipliers i.e. the ratio of Gr1(MUL) to Gr1(ADD) approximately equals to one. Bad composition would reduce the performance significantly.

The total number of possible interconnection is 7888(68 outputs $\times$ 116 inputs). In the DPPP, the entire bus interface required $2.2 \times 10^6\,\mu\mathrm{m}^2$ chip area. The chip area required for the interconnect is almost negligible because all sALUs in the DPPP are interconnected via a differential bus. If implemented with a switch matrix circuit, more than $5.0 \times 10^6\,\mu\mathrm{m}^2$ is required without considering the area required for interconnects, which are expected to a further significant addition to chip area.

7. Conclusions and Future Works

A parallel processor based on new architecture, a DPPP, was proposed. The main feature of the DPPP is the utilization of a code division multiple-access bus instead of a conventional interconnection topology, distinguishing itself from other parallel processors. The use of a CDMA bus is beneficial in that it provides complete routing flexibility and dynamic programmability. In addition, the DPPP consists of many simple processing blocks that operate in parallel, making it possible to achieve high chip density. An auto-program generation interface was also proposed that allows numerical formulas to be compiled and run on the DPPP. The DPPP can be improved in terms of speed and power consumption by further improving the CDMA bus interface. Other future developments include the provision of high-level programming language support for more complicated tasks that cannot be expressed in terms of numerical formulas or block diagrams. However, with its large number of processing elements and complete routing flexibility, the DPPP is useful for carrying out pipelining and for performing concurrent tasks. It is our belief that the DPPP will provide a new design paradigm for parallel computing.

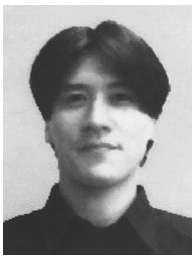
Acknowledgement

The VLSI chip used in this study was fabricated under the chip fabrication program of the VLSI Design and Education Center (VDEC) of the University of Tokyo with the collaboration of the Rohm Corporation and the Toppan Printing Corporation. This work is supported by the Japan Society for the Promotion of Science (JSPS) as part of the Research for the Future Program.

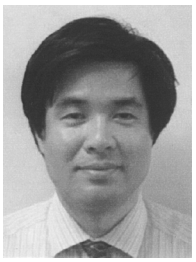
References

[1] K. Efe and A. Fernandez, "Mesh-connected trees: A bridge between grids and meshes of trees," IEEE Trans. Parallel and Distributed Systems, vol.7, no.12, pp.1281–1291, Dec. 1996.
[2] Q.M. Malluhi and M.A. Bayoumi, "The hierarchical hypercube: A new connection topology for massively parallel systems," IEEE Trans. Parallel and Distributed System, vol.5, no.1, pp.17–30, Jan. 1994.
[3] J. Levison, I. Kuroda, and T. Nishitani, "A reconfigurable processor array with routing LSIs and general purpose DSPs," Proc. Int. Conf. Appl. Spec. Array Process. IEEE Computer Society, pp.102–116, Los Alamitos, CA, USA. Aug. 1992.
[4] T. Gross and D.R. O'Hallaron, iWarp: Anatomy of a Parallel Computing System, Massachusetts Institute of Technology Press, 1998.
[5] D.C. Chen and J.M. Rabaey, "A reconfigurable multiprocessor IC for rapid prototyping of algorithmic-specific high-speed DSP data paths," IEEE J. Solid-State Circuits, vol.27, pp.1895–1904, Dec. 1992.
[6] R.W. Hartenstien, R. Kress, and H. Reinig, "A reconfigurable data-driven ALU for Xputers," IEEE Workshop on FPGAs for Custom Computing Machine, FCCM'94, Napa, CA, April 1994.
[7] Y. Yuminaka and Y. Sasaki, "Wave parallel computing systems based on orthogonal sequences," IEICE Trans., vol.J81-D-I, no.2, pp.133–142, Feb. 1998.
[8] R. Yoshimura, B.K. Tan, T. Ogawa, S. Hatanaka, T. Matsuoka, and K. Taniguchi, "DS-CDMA wired bus with simple interconnect topology for parallel processing systems LSIs," ISSCC Digest of Techinical Papers, pp.370–371, Feb. 2000. others.
[9] N. Shigei, H. Miyajima, and S. Murashima, "On efficient spare arrangements and an algorithm with relocating spares for reconfiguring processor arrays," IEICE Trans. Fundamentals, vol.E80-A, no.6, pp.988–995, June 1997.
[10] T. Horita and I. Takanami, "An efficiently reconfigurable architecture for mesh-arrays with PE and link faults," IEICE Trans. Inf. & Syst, vol.E80-D, no.9, pp.879–885, Sept. 1997.
[11] S. Dutt and J.P. Hayes, "Some practical issues in the design of fault-tolerant multiprocessors," IEEE Trans. Comput., vol.41, pp.588–598, May 1992.
[12] B.K. Tan, R. Yoshimura, T. Matsuoka, and K. Taniguchi, "DS-CDMA wired bus for parallel processing system," International Symposium on Advanced Analog CMOS Circuits, pp.39–44, Tokyo, Japan, Dec. 2000.
[13] S. Glisic and B. Vucetic, Spread Spectrum CDMA systems for Wireless Communication, Artech House Publishers, 1997.
[14] B.K. Tan, R. Yoshimura, T. Matsuoka, and K. Taniguchi, "A novel dynamically programmable arithmetic array (DPAA) processor for digital signal processing," The 13th Workshop on Circuits and Systems in Karuizawa, pp.373–378, April 2000.

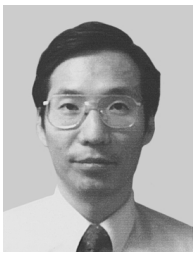
Tan Boon-Keat was born in Penang, Malaysia in 1972. He received his M.S. and Ph.D. degrees from Osaka University, Osaka, Japan in 1999 and 2001 respectively. He is now working as a Research Associate at the same university. His current research interests are reconfigurable processors, high performance digital circuits and analog circuits.



Ryuji Yoshimura was born in Nara, Japan in 1973. He received his B.S. and M.S. degrees from Osaka University, Osaka, Japan in 1996 and 1998 respectively. He is currently working at Texas Instruments Japan Ltd. His current research interests is Analog Circuits.



Toshimasa Matsuoka was born in Osaka, Japan in 1966. He received the B.S., M.S. and Ph.D. degrees in electronic engineering from Osaka University, Osaka, Japan, in 1989, 1991 and 1996, respectively. During 1988–1991, he was involved in the research of heterostructures and superlattices of GaAs and related compounds. During 1991–1998, he worked for the Central Research Laboratories, Sharp Corporation, Nara, Japan, where he was engaged in the research and development of deep submicron CMOS devices and ultra thin gate oxides. Since 1999, he has been worked with Osaka University. His current research includes phase lock loops and CMOS RF circuits. Dr. Matsuoka is a member of the Japan Society of Applied Physics and the IEEE.



Kenji Taniguchi received the B.S., M.S. and Ph.D. degrees from Osaka University, Osaka, Japan, in 1971, 1973 and 1986 respectively. From 1973 to 1986, he worked for Toshiba Research and Development Center, Kawasaki, Japan where he was engaged in process modeling and the design of MOS LSI fabrication technology. He was a Visiting Scientist at Massachusetts Institute of Technology, Cambridge, from July 1982 to November 1983. Presently, he is a Professor of Electronics Engineering at Osaka University. His current research interests are in analog circuits, radio frequency circuits, device physics and process technology. Prof. Taniguchi is a member of the Japan Society of Applied Physics. He is a fellow of the IEEE.