Invited Paper

A 64-bit RISC Microprocessor for Parallel Computer Systems

YUJI TANIKAWA**, KATSUYUKI KANEKO*, TADASHI OKAMOTO*, MASAITSU NAKAJIMA*, YASUHIRO NAKAKURA*, SATOSHI GOKITA*, JUNJI NISHIKAWA* and HIROSHI KADOTA*

This paper describes a microprocessor designed for a (Processing Element) PE of a scientific parallel computer system. This processor consists of three operational units: an instruction fetch and decode unit, an integer/address operation unit, and a floating-point operation unit. The chip is fabricated by means of a two-Al-layer, 1.2 μm N-well CMOS technology, and contains 440 K transistors in a 14.4×13.5 mm^2 die. The processor, which employs RISC architecture and Harvard-style bus organization, executes most of its 47 instructions, including 64-bit floating-point operations, in one 50-nsec cycle (20 MFLOPS/20 MIPS) with a 5-stage pipeline organization. The performance of the processor and its special functions for parallel computer systems are also discussed.

1. Introduction

Parallel computers are among the most promising machines for high-speed scientific computation. This is mainly owing to the rapid advancement of VLSI technology which has made it possible to develop high-performance components necessary for parallel systems, fast and powerful numerical operation units, and large-scale functional network components [1].

Among various parallel architectures, those with loosely coupled processor networks, processing element where each (PE) has its own distributed memory and is connected through the interconnection network, are thought to be the best. These systems have better opportunities to exploit parallelism and get higher performance in some applications, such as graphics and signal processing, and (Partial Differential Equation) PDE solvers, than those with shared memory architecture [2, 3].

A new microprocessor has been developed that is suitable for parallel computer systems with distributed memory and interconnection networks (see Fig. 1) [4, 5]. The following design targets were considered: (1) high-speed floating-point operations, (2) facilities for composing compact PE, (3) efficient execution of typical scientific programs, (4) compiler support, (5) reliability in data handling, and (6) synchronization and communication functions. As a result, the processor has a full 64-bit floating-point operation unit, an instruction cache, an (Error Checking and Correction) ECC circuit and other hardware crucial for parallel computers.

This paper mainly describes the architecture of the processor. The next section explains the general architecture, and is followed by descriptions of the internal operation and the pipeline structure in Section III, and the instruction set and functions for a parallel system in Section IV. The performance analysis is discussed in Section V. Finally, the conclusions of the paper are stated in Section VI.

2. Processor Architecture

The basic architecture of this processor is a combination of the Harvard architecture, an internal instruction cache, and the RISC structure.

According to statistics on scientific and numerical programs, memory access has remarkable characteristics: non-locality or uniformity of data access, strong locality of instruction access, such as loops or iterative operations, and a 20-30% frequency of memory access instructions. Because of the low latency of memory access caused by the simple structure of the PE, the processor must have a wide data bus to maintain a moderate data transfer rate. From the same reason, bus conflict between instruction fetch and data access should be avoided. To achieve effective data access, we adopted the Harvard architecture with a 64-bit data (and 8-bit ECC) bus, which is isolated from a 24-bit instruction bus to allow double-precision data to be stored in one memory cycle.

An instruction cache of appropriate size is effective for speeding up the internal cycle, because iteration and loop operation often appear in such codes. The size is fixed at 256 words ($\times$ 24 bits), which is large enough to

*Semiconductor Research Center, Matsushita Electric Industrial Co. Ltd.
**Information and Communications Research Center, Matsushita Electric Industrial Co. Ltd.

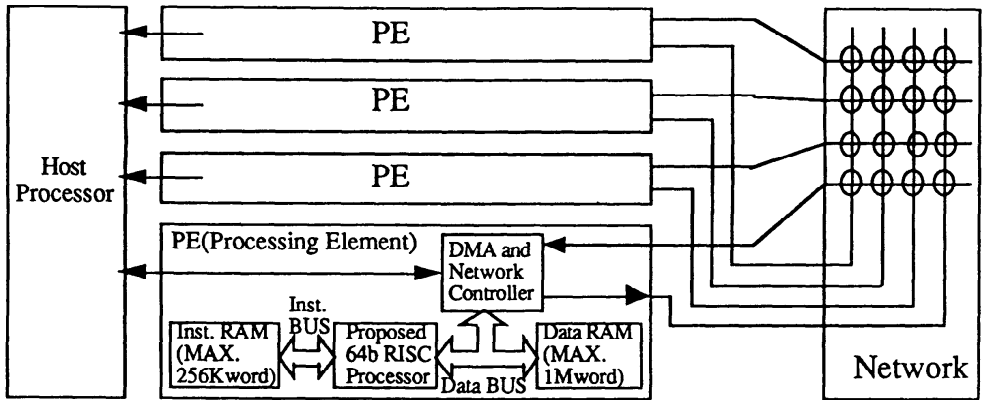

Fig. 1 Organization of the parallel processor System.

store the codes corresponding to the innermost loop of a typical scientific program in FORTRAN.

One of the most important targets in designing a processor for scientific or engineering applications is to provide powerful floating-point operation units that execute floating-point instructions as fast as other instructions, such as integer add or branch. If RISC architecture is employed, the amount of hardware for the control path can be decreased, and a faster and more powerful floating-point unit can be placed on the same silicon area. A computer translates most FORTRAN-like numerical programs in to machine codes with limited instructions, using a few kinds of addressing modes. RISC architecture is again advantageous in this respect. The processor provides 47 instruction, most of which are executed in one internal cycle. Only divide instruction takes seven cycles. This problem will be mentioned again in section IV.

The programming models of the processor have thirty two 64-bit data registers, thirty two 23-bit address registers, a program counter, and a processor control register. These address registers include three index registers, twenty-three general address registers, and six special registers, such as a (Processor Status Word) PSW, a loop counter, or stack pointer. In R-R operation, all of these registers are symmetric and can be used as general registers.

A PE consists of a processor chip, an instruction memory, a data memory, and a peripheral controller chip, which controls communication with both the processor and the network [6]. The processor has an instruction port (18-bit address and 24-bit data) and a data port (22-bit address, 64-bit data, and 8-bit ECC), which can be connected directly to (static RAM) SRAM and (dynamic RAM) DRAM, respectively, without any extra chips to interface with. The data bus interface includes a page-mode access controller and a refresh control circuit. An ECC circuit using 8-bit check code, which corrects 1-bit errors and detects 2-bit errors, is also provided in this interface to recover from external bit errors, such as software errors in DRAMs.

3. Internal Operation and Pipeline Structure

The goal of the microprocessor for scientific use is to obtain high FLOPS in both peak and average performance. The size of the 64-bit hardware is roughly four times larger than that of the 32-bit floating-point hardware. Therefore, not only the speed but also the silicon area, power consumption, and design cost for the floating-point unit have been considered in the circuit and layout design [7].

As shown in Fig. 2, the processor consists of three operational units: an instruction fetch-and-decode unit, which generates instruction addresses and also controls the operation flow; an integer/address operation unit, which generates data addresses and computes ALU instructions; and a floating-point operation unit, which executes 64-bit arithmetic and data I/O operations. The instruction fetch-and-decode unit includes a 256-entry direct map instruction cache, a 32-word stack, an instruction address generator, an instruction register, and a decode unit. The cache can supply instructions every 50 nsec. If there is a cache miss, instructions are fetched within 100 nsec from external memory. The loop counter and the stack are used to accelerate loop and subroutine operations. This is especially effective in the execution of loop operations such as FORTRAN's DO statement. As the loop counter is one of the registers in the integer/address operation unit, the value of counters can be referred to as a operand. The integer/address operation unit consists of an ALU, a shifter/swapper, 32 address registers, a multiplex PE assignment operation unit, and a processor control register. The Shifter/swapper is for address generation in the FFT application, which is useful for signal processing or very large PDE solvers. The floating-point execution unit consists of a floating-point multiplier (FMUL) a, floating-point arithmetic unit (FAU), a 32-word 64-bit register file, an ECC circuit, and coefficient

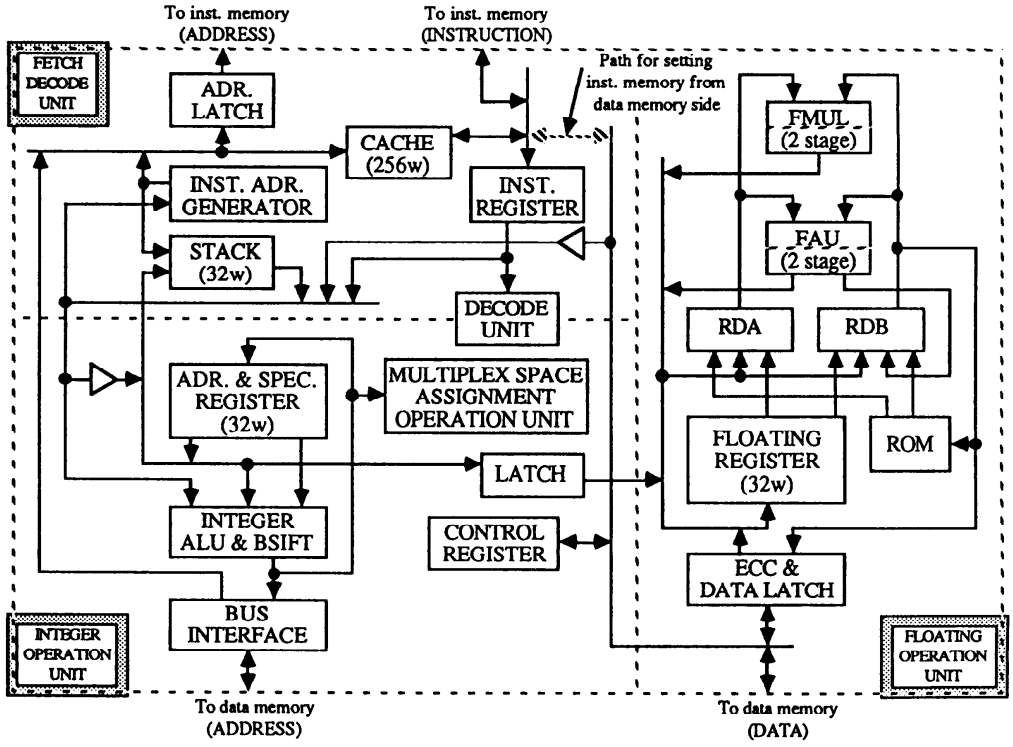

Fig. 2 Block diagram of processor.

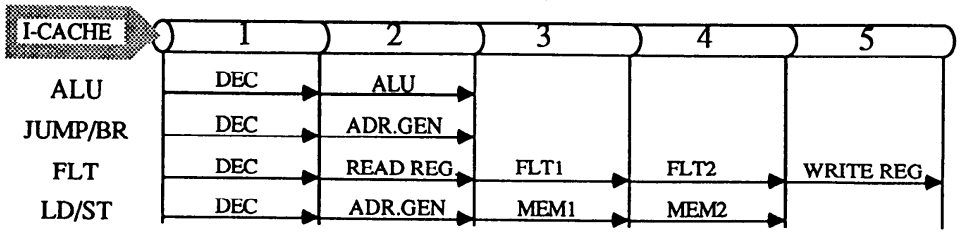

Fig. 3 Pipeline Structure of processor operations.

ROMs. The FMUL employs the second-order Booth algorithm and modified balanced delay tree. A new divison method has been implemented that is based on the convergence algorithm, and is carried out by the FMUL, FAU, and coefficient ROMs every 7 cycles and controlled by internally generated microinstructions.

The processor employs a two-stage floating-point pipe to balance the floating-point operations with other operations. The more the pipe is partitioned, the faster the pipeline cycle is, but at the same time waste cycles caused by register interference or condition reference increase. In terms of performance and complexity of pipeline control, a two-stage pipe seems to be a reasonable choice. Another choice is whether to use a floating-point unit or other handware resources to achieve parallelism. Since floating-point operations takes only two machine cycles in latency or one cycle in the pipeline, if the concurrent operations on floating-point numbers and integers were introduced, the rate of supply of instructions would be incerased or the word width of each instruction would be enlarged. Cooperation and interaction through the bus interface would be another problem, because both units need memory access and also need to cooperate when the floating-point unit uses the data bus. This increases the design complexity. To escape from these problems, we employed a simple single-stream pipeline scheme. This also gives large number of bus slots to peripheral devices that share the data bus with the processor.

Thus, as shown in Fig. 3, the processor employs to tally a pipeline, with five stages pipeline, excluding instruction fetch (IF): instruction decode (DEC), integer

operation (ALU), first floating-point operation (FLT1), second floating-point operation (FLT2) and write-back to the floating-point register (WR). The ALU stage includes address caluculation, read out of floating-point registers, and operation flow control, such as branch and/or internal stack manipulation. A memory operation takes two cycles (MEM1, MEM2) which are assigned to the FLT1 and FLT2 slots.

4. Instruction Set and Functions for a Paralel System

In its instruction set architecture, the processor provides 47 instructions, most of which are executed in one machine cycle. Twelve instructions are for floating-point operations, nine for integer operations, six for data movements, five for I/O operations, ten for flow control operations and five for system-and multiprocessor-related instructions respectively. Fig. 4 shows typical instruction formats. A 9-bit field of a 24-bit instruction word is served to opecode. The Computation operations of both the ALU and the floating-point unit are based on two operands and have the same format. With regard to memory access for matrix data, the memory operation has three addressing modes: 16-bit absolute, indirect with 5-bit displacement, and indirect with index and 5-bit displacement. The processor supports maskable and non-maskable interrupts. When an interrupt occurs, the program execution starts at the address 0, after the completion of the instruction on the decoding stage. The release of a reset signal causes execution to start at the address 0. This simplified exception handling allowes flexible monitoring of software.

The following functions are specially supported for parallel system construction: (1) synchronization and communication between PEs or PE and host computer, (2) facilities for composing compact PE, (3) reliability in data handling, and (4) multiplex PE assignment operation for a large-space 2- or 3-dimensional problem.

The processor provides signal lines for data communication between PEs. In the target parallel computer system under development, shown in Fig. 1, each processor shares (local data memory) LDM with a dedicated DMA controller, which communicates with other PEs as shown in Fig. 5. In order to realize concurrent operation between the processor and DMA chip, in other words, concurrence between computation and communication, the processor provides two kinds of store instructions, (normal store) ST and (store with notification signal) STF. When the processor generates the data to be transferred in each iterative loop, the processor stores them by means of the STF instruction. After it detects that signal, the DMA chip can 'snoop' or fetch the data and send them to the network in order.

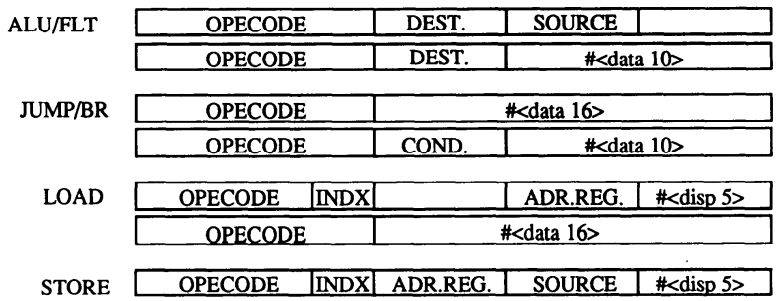

Fig. 4 Instruction formats for the processor.

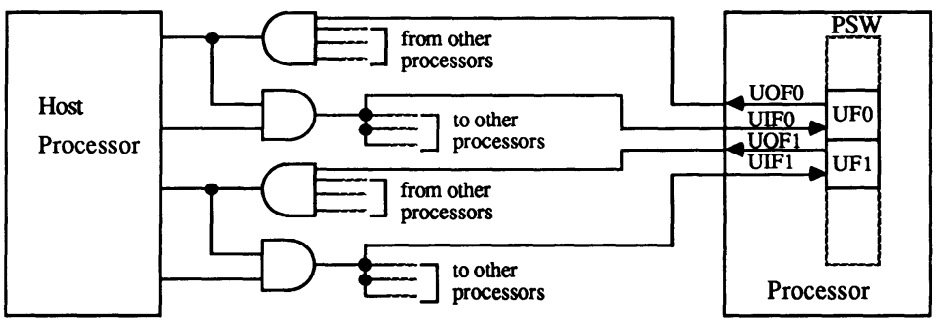

Fig. 5 Synchronization and communication.

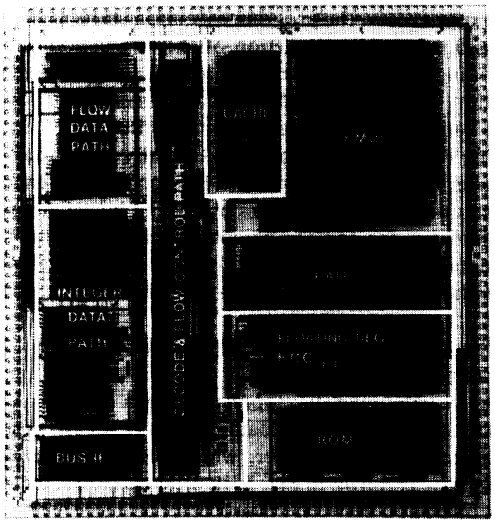

Fig. 6 Microphotograph of the chip.

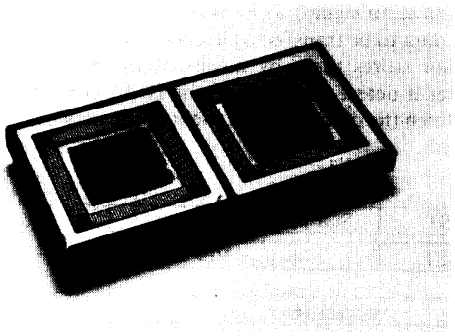

Fig. 7 Photograph of a 242-pin multi-chip package for the processor chip and the peripheral controller chip.

Table 1 Chip characteristics.

Technology	1.2-um 2-metal layer CMOS
No. of transistors	440 KTr
Chip size	13.5 × 14.4 mm²
Total Pads	172 (including power-pads)
Package	172-pin PGA or 242-pin PGA (multi-chip package with DMAC)
Extermal clock rate	40 MHz
Internal machine cycle	50 nsec (25 nsec 2 phase)
Cache (size and structure)	256 words direct-map
Floating point register	64 bit × 32 words
Address register	23 bit × 26 words
Internal stack	23 bit × 32 words
No. of instruction	47
Execution cycles	
Integer/Floating-point operation	1 (excluding divide)
Floating-point device	7
Memory access	2
Branch/Jump	1 (not taken)-3 (taken)
I/O cycles	
Data bus (syncronous)	2 (CAS access)-4(RAS-CAS access)
Instruction bus (asyncronous)	2
Power dissipation	1.3 W (typical)

Table 2 Performance of the processor.

LFK No.	performance [MFLOPS]		item
	with cache	without cache	
1	4.2	2.2	hydro fragment
3	2.9	1.7	inner product
5	2.2	1.4	tridiagonal elimination
7	4.0	2.6	eq. of state fragment
9	4.4	3.0	integrate predictor
10	2.1	1.3	difference predictor
11	1.6	1.1	first sum
12	1.6	1.1	first difference

For the process synchronization, there are two input and output user flags: the user input flag (UIF) and the user output flag (UOF), which are directly connected bits (user flag 0, 1) of the PSW and can be used as branch conditions, as shown in Fig. 5. When each processor has finished one process, it sets the UOF (in this case, the UOF means the local end of the process) in the PSW. In the next cycle, each processor waits for the UIF to be asserted (in this case, the UIF means the global end of the process). All UOF's are gathered and ANDed to generate UIF signal, and the execution of the next process is started.

By using these signal lines, the hardware of a parallel system can be easily designed. The simple structure of a PE makes it possible to design a compact but highly parallel computer system. The processor provides special hardware for multiplex PE assignment operations for problems including more grids than the number of PEs, which is useful for solving PDE problems in a large physical space where each PE has to support multiple grids.

As regards the controllability of the processor in parallel computer environment, the instruction bus space and the control register that defines the processor status and the starting address can be mapped on the data bus space. When the processor is in the reset state, the host processor can read and modify all the resources of a PE:data memory, instruction memory, and processor status-through the peripheral controller chip and the processor itself.

In the new processor being developed, the data width in an integer block will be changed from 24 bits to 32 bits to expand its application.

5. Performance

The chip is fabricated by meaus of a two-Al-layer, 1.2 μm N-well CMOS technology, and contains 440 K

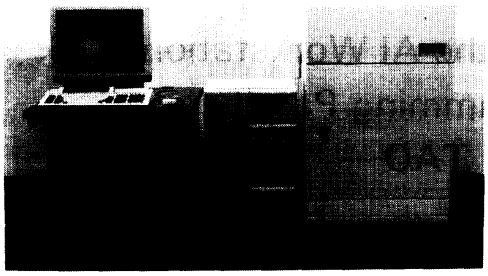

Fig. 8 The parallel computer system ADENA.

transistors in a $14.4 \times 13.5\ mm^2$ die. A microphotograph is shown in Fig. 6. The chip requires a 40 MHz master clock and a 20 MHz slave clock for synchronization with peripheral chips. The average power dissipation including I/O drive is under 1.3 W when measured at 40-MHz operation with a 5 V power supply. The package is a 172-pin plastic PGA or a 242-pin ceramic multi-chip package with a peripheral chip, which shares a common data bus with the processor (Fig. 7). The major chip characteristics are summarized in Table 1.

The peak performance of the processor is 20 MFLOPS or 20 MIPS. The performance for typical problems have been measured 7 MFLOPS for fractal computation and 4 MFLOPS for Gaussian elimination execution, respectively, not counting floating-point storage operations. Table 2 shows the performance result for some as Livermore FOTRTRAN Kernels with codes generated by the compiler. The a average (Cycles Per Instruction) CPI was 1.25. The performance mainly depends on the ratio of floating-point operations to load/store instructions. If the memory location of the program or programming techniques, such as unrolling, delayed jump, and so on, is considered, the performance may become a few MFLOPS higher.

A new parallel computer system called. (Alternating Direction Edition Nexus Array) ADENA with 256 PEs has been developed with this processor as its element [8, 9]. The performance of this computer is 2.56 GFLOPS of peak level and 1.8 GFLOPS for fractal computation. As shown in Fig. 8, this system has a host computer, an engineering work station, and a slave parallel computer, which the main part of ADENA.

6. Conclusion

The processor has several features for an environment of scientific computation and parallel processing. For improved scientific computation, a combination of RISC structure, Harvard-architecture with a 64-bit data bus, and an on-chip instruction cache have been introduced. Dedicated hardware and a pipeline scheme allow FMUL/FAU operations every 50 nsec. Division is executed every 350 nsec. A typical performance of 4–7 MFLOPS has been achieved in practical applications. As a result, GFLOPS-level computation will be possible in a parallel computer system with a few hundred processors.

Acknowledgment

The authors would like to thank Prof. Nogi for discussions and suggestions, and Dr. S. Horiuchi and T. Ishihara for their encouragement.

References

1. KANEKO, K. et al. Network-component chip set for a parallel processor system, 1988 Symposium on VLSI Circuits Dig. Tech. Papers (Aug. 1988), 39–40.
2. HOSHINO, T. An invitation to the world of PAX, *IEEE computer*, **19**, 5 (May 1986), 68–79.
3. HAYES, J. P. et al. A microprocessor-based hypercube supercomputer, *IEEE Micro*, **6** 4 (Oct. 1986) 6–17.
4. KANEKO, K. et al. A 64-bit RISC microprocessor for a parallel computr system, ISSCC Dig. Tech. Papers (Feb. 1989), 78–79.
5. KADOTA, H. et al. VLSI parallel computer with parallel processing: ADENA, *Proc. 1989 International Conference on Parallel Processing* (Aug. 1989), 319–322.
6. NAKAKURA, Y. et al. A versatile data-transfer control unit for a parallel processor system, 1989 Symposium on VLSI Circuit Dig, Tech. Papers (May 1989), 15–16.
7. KANEKO, H. et al. A VLSI RISC with 20-MFLOP Peak, 64-bit Floating-Point Unit, in *IEEE Journal of Solid-State Circuits*, **24**, (Oct. 1989), 1331–1340.
8. NOGI, T. Parallel Computation, Patterns and Waves-Qualitative Analysis of Nonlinear Differential Equations, (1986), 279–318.
9. NOGI, T. The ADENA Computer, in International Symposium on Applied Mathematics and Information Science, Kyoto University towards Multidimensional Flow Models, Mathematics and Computers, Kyoto University (1984), 7/9–16.

(Received August 28, 1989)