Invited Paper

Design of the Dataflow Single-Chip Processor EMC-R

SHUICHI SAKAI*, YOSHINORI YAMAGUCHI*, KEI HIRAKI*,
YUETSU KODAMA* and TOSHITSUGU YUBA*

This paper presents the design of the dataflow single-chip processor EMC-R, from the viewpoint of advanced dataflow schemes and their implementations. The EMC-R is a component chip of a highly parallel (with more than a thousand processors) dataflow machine, the EM-4. The distinctive features of the chip are: (1) a refined dataflow model called a strongly connected arc model, (2) two simple and fast synchronization mechanisms, (3) a versatile pipeline design, (4) a RISC-based architecture, (5) a packet-switching unit design with extra facilities and (6) a maintenance architecture for monitoring the system. After these features have been examined, the configuration architecture of the EMC-R that makes them possible is shown. There are six units on the chip: a switching unit, an input buffer unit, a fetch and matching unit, an execution unit, a memory control unit and a maintenance controller. The EMC-R is a CMOS gate array chip containing 45, 788 gates and using 255 signal pins. It is mounted on a PGA ceramic package. The EM-4 prototype system with 80 EMC-Rs is available now. Its hardware system was completed in April 1990. The purposes of the prototype are (1) to evaluate several architectural aspects by measuring dynamic characteristics of practical programs, (2) to confirm the architectural design, and (3) to provide an environment for software development.

1. Introduction

Current VLSI technology will give rise to gigantic computing systems whose component chips contain millions of transistors. Our main concern is how to construct a real computing system that provides maximum performance for certain (or various) applications. There are two main architectural techniques for gaining high performance in such systems: parallelization and pipelining. Fundamentally, the degree of parallelism extracted must be as high as possible and the pipeline pitch must be as fine as possible, provided they are sufficiently filled with operations. Parallel computers have special problems that do not appear in sequential computers: (1) extracting parallelism from a user program, (2) mapping a huge number of tasks among many processing elements (PEs), (3) task synchronization, (4) latency, and (5) software writability/readability.

Dataflow architecture affords one of the most powerful and versatile approaches to these problems. It can naturally extract the maximum available concurrency in a computation, distribute tasks to a large number of processors, and distribute tasks to stages of a circular pipeline [4] in each processor. It can afford a flexible synchronization mechanism, and it can naturally eliminate latency-dominant processing. It can also provide an elegant programming paradigm for high productivity of software. In addition, a dataflow machine can be constructed with a large number of identical processing elements (PEs) and repetitive data networks. This is advantageous for VLSI implementation.

Several architectures based on the dataflow concepts have already been proposed [1, 2, 3, 4, 5, 9, 10, 11]. Among these, architectures based on a tagged token dataflow model [1, 2, 4, 5, 9] utilize a packet flow scheme and a circular pipeline organization. This type of architecture has already been used in some real experimental dataflow machines. One of them, the SIGMA-1 [5, 8], a dataflow supercomputer for numerical applications developed in the authors' laboratory, has 128 PEs and performs at a speed of more than 100 MFLOPS, with 1985 CMOS gate-array technology. The SIGMA-1 illustrates the possibility that a tagged-token dataflow machine can surpass conventional von Neumann computers. It also shows some of the problems of a simple dataflow model and a dataflow architecture.

Current research on the SIGMA-1 has two themes. One is software development and the other is construction of a next-generation dataflow computer. The first includes high-level language design, compiler design, optimization techniques, construction of benchmark programs, and evaluation of dataflow schemes and of the machine itself. A high-level language, DFC-II, was designed and its compiler is now under development

*Electrotechnical Laboratory, 1-1-4 Umezono, Tsukuba, Ibaraki 305, Japan.

[15]. The second theme is to construct a feasible and much more efficient ultra-parallel computer. For this purpose, a dataflow machine called EM-4 [12, 13] is now under development.

The EM-4 is a next-generation dataflow computer whose target structure has more than 1,000 PEs. It will overcome several defects of dataflow machines, which are described later, and can be implemented with state-of-the-art or very-near-future technology. For compact implementation, the PEs of the EM-4 must be packed onto a single chip. The single chip processor is called EMC-R.

This paper describes the design of the EMC-R from the viewpoint of advanced dataflow schemes and their implementations. Section 2 presents the design philosophy and features of the EMC-R. Section 3 describes the architecture of the single-chip processor, focusing especially on its configuration architecture. Section 4 shows the implementation of the EMC-R. It also describes the system organization and current status of the EM-4 prototype. Finally, Section 5 concludes this paper by summing up the distinctive features of the EMC-R and indicating directions for future works on the EM-4.

2. Design Philosophy and Features of the EMC-R

The problem is how to construct an extremely fast, useful, and reliable computer. To do so on the basis of a dataflow concept, four main questions must be answered: how to overcome the defects of the dataflow, how to make a highly parallel computer feasible, how to provide a maintenance structure for all the hardware and software, and how to support high software productivity. The EM-4 and the EMC-R have been designed with close attention to these points. The following subsections will show how the first three questions are answered in the machine design. The last point will be discussed in another paper.

2.1 Model-Level Refinement of Dataflow

Conventional dataflow architectures, including the DFM [1], the MIT Dataflow Machine [2], the Manchester Dataflow Machine [4], and the SIGMA-1 [5] involve, several problems in realizing an efficient and feasible computer [12]: (1) a circular pipeline does not provide advanced control and thus does not work well as a "pipeline" for less parallel execution; (2) simple packet-based architecture cannot exploit registers or a register file efficiently; (3) the time complexity and hardware complexity for matching are high, if a colored token style is adopted; (4) the packet flow traffic is too heavy; (5) current dataflow concepts cannot provide flexible and efficient resource management mechanisms; and (6) it takes much time to eliminate garbage tokens. Note that these problems have become clear to the authors through their experience with the SIGMA-1.

To overcome these defects, the EM-4 uses several novel concepts and techniques. One of the most important is the refinement of the dataflow model itself. The improved model is called a *strongly connected arc model.*

An exact definition and detailed consideration of the strongly connected arc model are given in other papers [7, 12, 13]; here, the essence of the model and its implementation in the EM-4 will be described.

In this model, dataflow graph arcs are classified into two categories: normal arcs and strongly connected arcs. A dataflow subgraph whose nodes are connected with strongly connected arcs is called a *strongly connected block* (SCB). In addition to the normal data-driven firing, there is a restriction on firing rules: a strongly connected block must be executed exclusively; that is to say, if one of the nodes in a certain SCB is fired, then the PEs concerned with the SCB can only execute the nodes in the same SCB. This means that each SCB is executed as a critical section in the PEs dedicated to it.

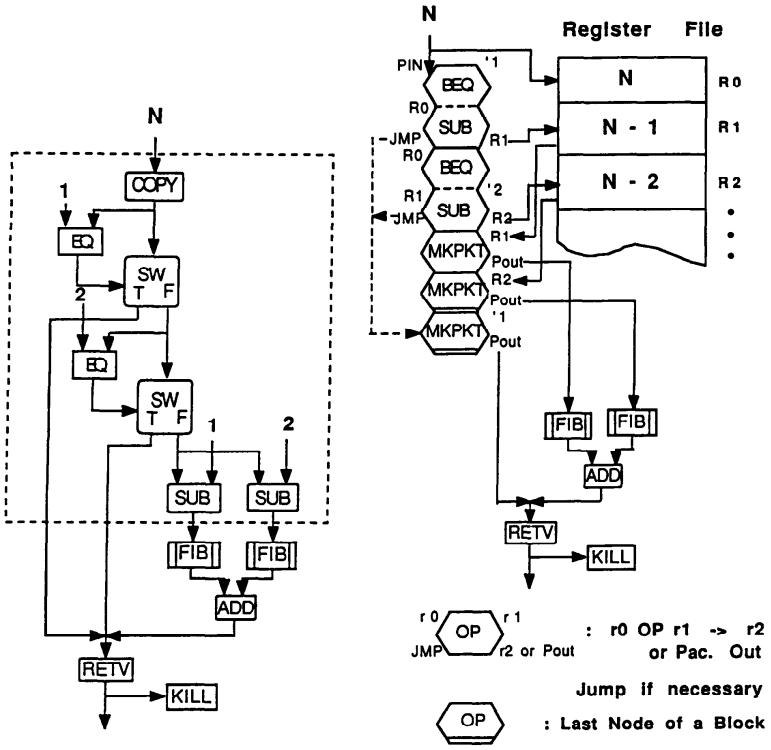
There are many ways of realizing the strongly connected arc model. In the EM-4, all the nodes of one SCB must be allocated to a single PE; the entrance of an SCB is a unique node; there is no child function in the SCB; SCB-tokens are stored in a register file, not transferred as a packet; the firing order of the SCB nodes is determined statically, that is, by a compiler; and every SCB is executed not by a circular pipeline but by an advanced control pipeline.

This implementation gives the EM-4 several advantages over conventional dataflow machines: it allows a fine-pitch advanced control pipeline, the matching overhead is greatly reduced in SCB execution, packet traffic is greatly reduced, flexible resource management can be easily provided by the SCB, and garbage tokens are greatly reduced, since there are no garbage tokens in the SCB.

For example, Figure 1 shows two programs for deriving the Fibonacci number on the EM-4. Figure 1(a) shows a conventional dataflow graph and Figure 1 (b) shows a dataflow graph with the strongly connected arc model. The series of hexagons in Fig. 1 (b) makes an SCB whose execution has the same effect as the dotted rectangle in Fig. 1 (a). In the EM-4, the execution of the rectangular part in Fig. 1 (a) takes twenty-three clocks and the corresponding part in Fig. 1 (b) takes only nine clocks. That is to say, the latter is about two and half times as fast as the former. This difference is caused by the register-based advanced control pipeline (described in Section 2.3) and by the elimination of the packet distribution overhead.

2.2 Simple and Fast Synchronization

To reduce the heavy overhead of data matching, two synchronization (or execution ordering) mechanisms are provided in the EMC-R. One is for static ordering inside the SCB. The other is called a *direct matching scheme,* and is for packet-data matching.



(a) Normal Dataflow Fibonacci Program

(b) Strongly Connected Fibonacci Program

Fig. 1 Program Example of a Strongly Connected Block (FIBONACCI).

The direct matching scheme offers data matching with no associative mechanisms. In this scheme, a matching location is indicated by an absolute address. This is possible since a full matching area for an active function is allocated at the function invocation time, that is, if there are N packet-data matchings in a certain function, N words are allocated to correspond to the instruction at each node. There is no special hardware for the matching itself or for the matching address generation. Note that this is a *dynamic* data matching scheme applicable to any type of tagged-token dataflow machine.

Another paper [13] gives a detailed description of the direct matching scheme. Its hardware is fairly small and there are no mis-hits of associative access at the cost of memory efficiency. In the EM-4 prototype, each PE can provide 1,000 active function instances at the same time, so memory hang-up seldom occurs.

2.3 Versatile Pipeline Design

As mentioned in Section 2.1, a circular pipeline cannot stand by itself, since it does not match the register architecture and cannot perform advanced control. On the other hand, introduction of the strongly connected arc model brings a register-based advanced control pipeline. The EMC-R thus adopts two integrated pipelines (Fig. 2). One is a circular pipeline with a direct matching scheme, and the other is a register-based advanced control pipeline for executing SCBs. The latter is called a *strongly connected pipeline.*

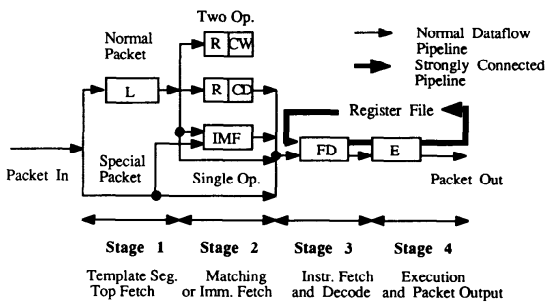


Fig. 2 Pipeline Organization of the EMC-R.

There are four stages in a circular pipeline: (1) *Template Number Fetch*, for linkage between a matching location and an instruction (L), (2) *Matching* (R/CW, R/CD) or *Immediate Fetch* (IMF), (3) *Instruc-*

tion Fetch and Decode (FD), and (4) Operation Execution and Packet Output (E). Each stage is performed in a single clock (80 ns). This pipeline has several bypass routes, that is to say, special packets such as a remote read/write of a memory word bypass the first stage, and packets for single-operand instructions bypass the second stage.

In the second stage, the matching operation contains the following actions: read the matching word from memory (R), check whether the partner is present or not, and eliminate it if it is present (CD) or write the packet data in the same memory place (CW) if it is absent. The direct matching scheme allows all these actions to be executed in a single clock. In the fourth stage, instruction execution and packet transfer are realized in parallel, that is, the header part of the result packet is transferred at the same time as the instruction execution.

The latter two stages are shared with a strongly connected pipeline. In an SCB execution mode, the instruction is executed in parallel with the fetch and decode of the next instruction. Simultaneously, the fetch of the next instruction operands is carried out. Normally the results are stored in a register file, but they can also be sent as a packet. The operands are usually fetched from a register file in parallel with the previous instruction execution.

In this way, two kinds of pipeline are integrated naturally.

2.4 RISC-Based Single-Chip Architecture for a Highly Parallel Computer

So as to shorten the execution cycle and reduce the development period, the EMC-R adopts a RISC architecture. We consider the features of a RISC chip for a parallel processor PE to be as follows:

(1) Small instruction set, (2) A few instruction formats, (3) A few addressing modes, (4) Register file architecture, (5) No microprograms, (6) A few packet formats, (7) Simple synchronization mechanisms, (8) Communication unit is included and performs independently of the processing.

Of these, (1)-(5) are the same as in a sequential RISC chip. A parallel RISC chip needs (6)-(8) for efficiency and simplicity.

The EMC-R has only (1) twenty-six instructions, (2) four instruction formats, and (3) two addressing modes. In addition, (4) it adopts a register file architecture, (5) it has no microprograms, and (6) the packet format is fixed. In the EMC-R, (7) the synchronization mechanisms are static ordering of the SCB nodes and direct matching, which are both fairly simple and fast. As will be mentioned in the next section, (8) it contains a communication unit that performs independently of the processing sections. The EMC-R thus meets all of the above conditions.

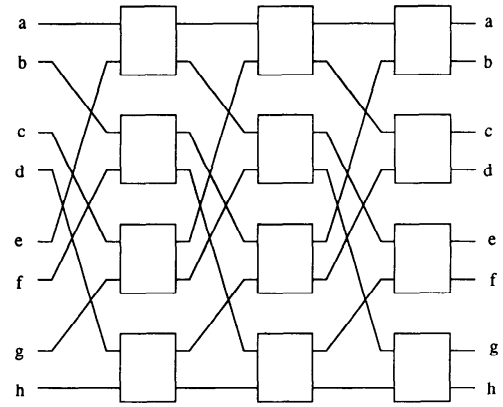


Fig. 3 Processor-Connected Omega Network.

2.5 Interconnection Network with Extra Facilities

The essential conditions that should be satisfied in the interconnection network of a huge computing system are as follows [14]. It needs only O(N) hardware, where N is the number of PEs. The distance between any two nodes is less than or equal to O(logN). The structure is uniform, that is to say, any PE can see the network structure in the same way. Self-routing is possible. Store and forward deadlock (SFD) does not occur or can be easily prevented.

As a network topology satisfying all these conditions, a *processor-connected omega network* is selected. Very few other networks have all of the above features. For instance, a completely connected network, a star, a hypercube, a crossbar switch, a benes network, a bitonic network, and a logN network such as multistage omega do not satisfy the first condition. A mesh does not meet the second condition. A tree does not meet the third one.

Figure 3 shows the topology of the processor-connected omega network, where the number of nodes is twelve. Each box indicates a switching unit connected to a processor. As previously mentioned, the EMC-R contains the switching unit. One reason for this is to reduce the packet transfer time between a switch and a PE. Other reasons are the low hardware cost and design simplicity. This unit and a processor can work independently and concurrently.

Features of the EM-4 network are grouping mechanisms with circular paths, self-routing algorithms, an SFD prevention mechanism with spiral buffer methods, and automatic load balancing facilities, all of which are reported by Sakai et al.[14].

2.6 Maintenance Architecture

A maintenance architecture means an architecture for initializing, hardware/software debugging, and dynamically monitoring a whole system. The maintenance architecture of the EM-4 has three layers:

a maintenance host, a maintenance for a group of PEs, and maintenance circuits on the EMC-R chip. Features of the third layer will be presented here.

The fundamental functions of the on-chip maintenance logic are reading and writing registers on a chip and memory words independently of the system clock and system data path. Almost all the registers have an auxiliary address dedicated to maintenance. In addition, there are several special registers on the chip indicating the PE conditions, such as the number of instructions executed, the load status of the PE, the status of a packet buffer, the error conditions of the PE, and the usage of structure memories. The maintenance processor can read and write the PE registers and an outer memory even while the dataflow programs are running. A dynamic monitor can thus be provided.

The maintenance architecture also supports fault execution in a PE. Each PE can send an interrupt signal to the maintenance processor when an error occurs. If this happens, the system clock of the group to which the PE belongs can be halted immediately. The host can investigate the cause of the error by reading the status of the PE, and can resume the execution from the point at which it was halted after necessary treatment.

2.7 Other Features

Besides the described features above, the EMC-R also provides facilities for fast branching operations, efficient structure handling, lazy (demand-driven) operations, and fast function control. They are described and closely examined in other papers [12, 13].

3. Architecture of the EMC-R

Figure 4 shows a block diagram of the EMC-R, which realizes all the features described in the previous section. The chip consists of six units: a Switching Unit (SU), an Input Buffer Unit (IBU), a Fetch and Matching Unit (FMU), an Execution Unit (EXU), a Memory Control Unit (MCU), and a Maintenance Controller.

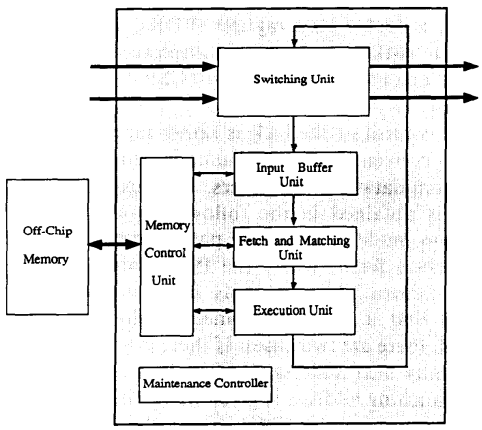


Fig. 4 Block Diagram of the EMC-R.

Table 1 EMC-R Hardware Profile of EMC-R.

UNIT	Gates	Pins
Switching Unit	9,179	176
Input Buffer Unit	9,295	—
Fetch and Matching Unit	3,610	—
Execution Unit	20,620	—
Memory Control Unit	1,664	67
Maintenance Controller	1,420	12
Total	45,788	255

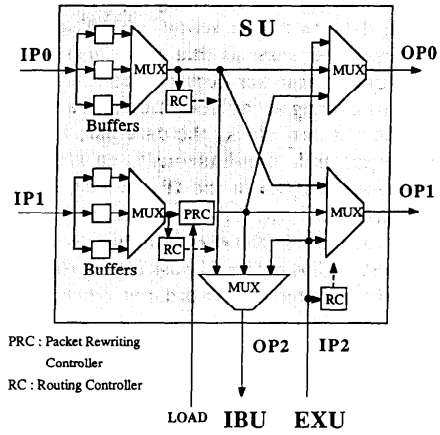


Fig. 5 Switching Unit Organization.

This paper focuses on showing configuration architecture of the EMC-R. The following subsections describe the roles and features of each unit. The EMC-R's instruction, set is listed in the description of the EXU; details of each instruction, the packet format, and instruction formats are presented in another paper [12].

Table 1 shows the gate usage and pin usage of each unit and of the EMC-R itself. The EMC-R consists of 45,788 CMOS gates and 255 signal pins. Each entry in this table will be referred to in the following subsections.

3.1 Switching Unit

The Switching Unit (SU) is an element of a processor-connected omega network. It switches packet data independently of and concurrently with other units. In the EM-4, all packets have the same size of two words. The first part is called the *address part,* and contains a destination address, a packet type, and flags. The second part is called the *data part*, and includes data and a data type.

Figure 5 shows the SU organization. The SU has two input ports, IP0 and IP1, from other PEs, and one input port, IP2, from the EXU. It also has two output ports to other PEs, OP0 and OP1, and one output port,

OP2 to the IBU. IP0 and IP1 have their own buffers, which are divided into three banks. These buffers are called spiral buffers, and provide facilities for SFD prevention [14]. Each IP has a routing controller (RC) that performs self-routing and each OP has an arbitration controller that controls the selection of a multiplexer connected to the outer world PE.

An automatic load balancing mechanism is implemented in a *packet rewriting controller* (PRC). The PRC rewrites a special packet named MLPE (minimum load PE), which always contains the minimum load PE address and its load. The rewriting is perfectly embedded into the data switching action, so the speed of the MLPE packet is as same as that of a normal packet.

Normal packet transfer occurs as follows. A packet address part arriving at the SU is stored in a buffer, if it is empty. In the next clock, the data part of the same packet is input and, simultaneously, an arbitration at the multiplexer (MUX) in the IP, routing in the RC, and an arbitration at the OP are carried out. After that, still in the same clock, the packet address part is stored in the next SU spiral buffer, if possible. In this way, the SU perfectly transfers one word per clock, including routing and arbitration.

According to Table 1, the SU contains 9,179 logical gates. These include three bank buffers, each of which has two words, and each word has thirty-nine bit data. Each network port (IP0, IP1, OP0, OP1) has thirty-nine data lines, two request lines, and three ready lines, that is to say, forty-four lines per port, or a total of 176 I/O lines.

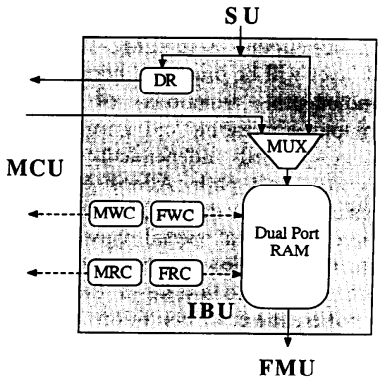
3.2 Input Buffer Unit

The Input Buffer Unit (IBU) is a buffer for packets waiting for matching and execution. A 32-word FIFO-type buffer is implemented on the chip. If an overflow occurs, a part of the off-chip memory is used as a secondary buffer, with a capacity of 8 K words. If the secondary buffer overflows, then the trap occurs and the maintenance controller informs the host of it.

Figure 6 shows the configuration of the IBU. It consists of a *dual port RAM* realizing a FIFO buffer a *data multiplexer,* a *data register* (DR) for a temporal packet storage, a *FIFO write counter* (FWC), a *FIFO read counter* (FRC), a *memory write counter* (MWC), a *memory read counter* (MRC) and control circuits. They are implemented with 9,295 CMOS gates, as seen in Table 1.

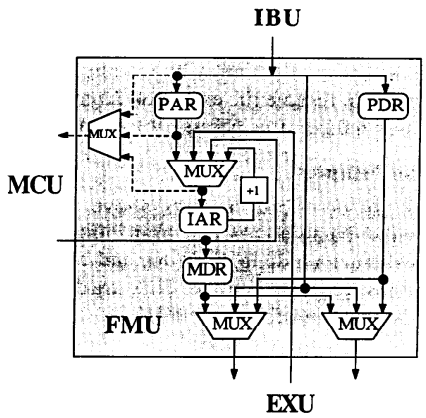
3.3 Fetch and Matching Unit

The Fetch and Matching Unit (FMU) performs all the synchronizations and sequencing in the EMC-R. It provides facilities for data matching, instruction fetch, and pipeline control, including the switching of two pipelines described in Section 2.3. The first three stages of the pipeline illustrated in Fig. 2 are all carried out in the FMU, except the decoding. Figure 7 shows the organization of the FMU. It consists of a *packet address register* (PAR), an *instruction address register* (IAR), a *packet data register* (PDR), a *matching data register* (MDR), several multiplexers, and control logics. The FMU contains 3,610 CMOS gates, as shown in Table 1.



DR : Data Register MWC: Memory Write Counter
MRC : Memory Read Counter FWC : FIFO Write Counter
FRC : FIFO Read Counter

Fig. 6 Input Buffer Unit Organization.



PAR : Packet Address Register PDR : Packet Data Register
IAR : Instruction Address Register
MDR : Matching Data Register

Fig. 7 Fetch and Matching Unit Organization.

The control of the IAR is rather more complex than that of conventional von Neumann computers and conventional dataflow computers. The next instruction address is obtained in the following way. In a normal dataflow mode, a successful packet matching causes an instruction fetch, where the PAR data are used with minor reform. In a strongly connected mode, which means that a strongly connected block is being executed, there are two cases. If there is no branching, the physically next address ((IAR)+1) is used. Otherwise, the branching address is given by the instruction word.

The status of the FMU means the synchronization status or sequence control status of the EMC-R. In the

EMC-R, the synchronization is performed only in the FMU and the execution is done only in the EXU. The status register of the synchnonization are clearly separated from those of the execution. This is called *synchronization-execution separation,* which is one of the major features differentiating the EM-4 from other advanced architectures [10, 11].

3.4 Execution Unit

The Execution Unit (EXU) executes an instruction, that is to say, it mainly performs the last pipeline-phase operation illustrated in Fig. 2. Table 2 shows the instruction set of the EMC-R. There are twenty-six instructions, which are classified into four categories: *arithmetic and logic, branch, memory or register read or write,* and *communication.* Arithmetic and logic instructions and communication instructions can make a packet and send it to an outer PE. Memory read or write instructions are the only instructions that can communicate between the off-chip memory chips and the EMC-R. Since there is no space on the chip for floating-point circuits, floating-point calculations will be performed by an off-chip commercial VLSI, if necessary. This is why Table 2 has no entries for floating-point instructions. These instructions will be realized by L (memory load) and S (memory store) instructions with special tag values.

Figure 8 shows the structure of the EXU. The EXU contains an *instruction register* (IR), an *instruction decoder* (DCD), operand registers (OP0 and OP1), an ALU, a *barrel shifter* (BSHF), a *multiplier* (MUL), a *versatile comparator* (CMP), packet generation circuits, and control circuits.

In an execution cycle, the contents of operand registers are sent to the ALU, and so on. Then the operation is carried out according to the instruction in the IR. The result is sent in a packet or stored in a register file. All of these actions are executed in a single system clock, and, in the same clock, the fetch and decode of the next instruction and data load from the FMU or a register file are performed.

Table 1 indicates that the EXU contains 20,620 gates, including register file handware.

3.5 Memory Control Unit

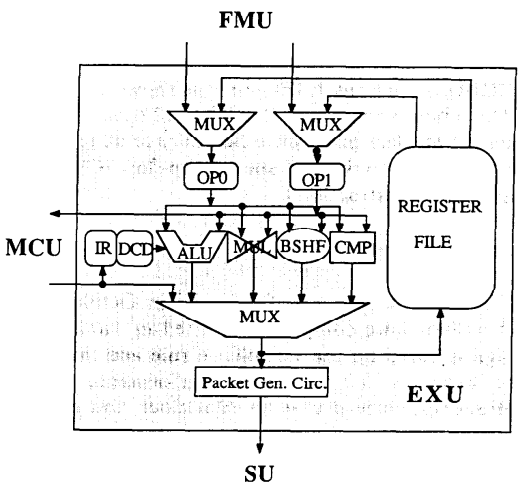
The Memory Control Unit (MCU) arbitrates memory access requests from the IBU, the FMU, and the EXU, and sends data between the off-chip memory and the EMC-R chip. The MCU consists of an *arbitration controller,* an *address multiplexer,* and a *data multiplexer.*

The off-chip memory size can be up to five megabytes. It is used for a secondary packet buffer, a matching store, an instruction store, a structure store, and a working area.

The amount of MCU hardware is 1,664 gates, as shown in Table 1.

Table 2 EMC-R Instruction Set.

CATEGORY	INSTRUCTION	ACTION
Arithmetic and Logic	ADD	integer add
	SUB	integer subtract
	MUL	integer multiply
	DIV0	preparation of division
	DIV1	element of division
	DIV2	correction of division
	DIVR	remainder of division
	DIVQ	quotient of division
	SHF	shift
	AND	bitwise AND
	OR	bitwise OR
	EOR	bitwise exclusive OR
	NOT	bitwise NOT
	ALUTST	ALU test
Branch	BEQ	branch by equality
	BGT	branch by greaterness
	BGE	branch by greaterness or equality
	BTYPE	branch by data type
	BTYPE2	branch by two data types
	BOVF	branch by overflow
Memory or Register Read or Write	L	load from memory
	S	store to memory
	LS	load and store from/to memory
	LDR	load from register
Communication	GET	send packet for remote operation
	MKPKT	make packet by two operands

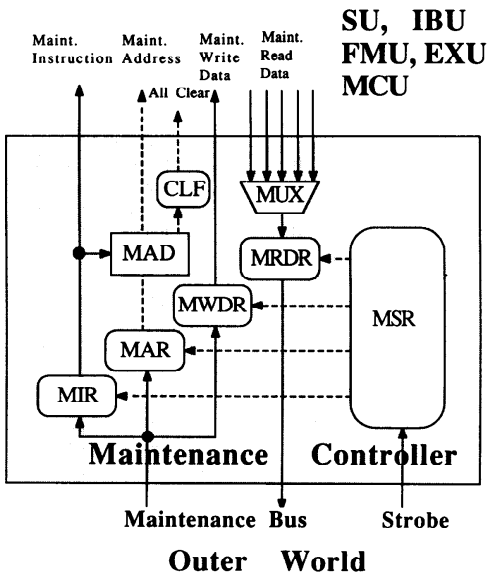


IR : Instruction Register OPi : Operand Register i

Fig. 8 Execution Unit Organization.

3.6 Maintenance Controller

The Maintenance Controller plays the role described in Section 2.6. It uses a clock independent of the system clock.



MIR : Maint. Inst. Reg. MAR : Maint. Address Reg.
MAD : Maint. Addr. Decoder MWDR : Maint. Write Data Reg.
MRDR : Maint. Read Data Reg. MSR : Maint. Status Reg.
CLF : System Clear Flip Flop

Fig. 9 Mainterance Controller Organization.

Figure 9 shows its organization. The Maintenance Controller consists of a *maintenance status register* (MSR), a *maintenance instruction register* (MIR), a *maintenance address register* (MAR), a *maintenance address decoder* (MAD), a *data multiplexer for reading register data,* a *maintenance write data register* (MWDR), and a *maintenance read data register* (MRDR).It contains 1,420 gates, as shown in Table 1.

The table also indicates that it has 12 signal pins. But there are no clear lines, since clearance of all registers is carried out by writing a special flip-flop (CLF) by a maintenance instruction.

4. Implementation

The EMC-R chip was fabricated in October 1989, and its hardware complexity is listed in Table 1. The design is based on the 1.5 micron rule and the typical gate delay is 0.7 ns. Peak performance is 12.5 MIPS/chip, since it uses an 80ns-clock and every instruction needs only one clock for its execution. This peak is obtained at the SBC execution. On the other hand, in a normal dataflow mode, the peak performance is 6.3 MIPS/chip. The chip has 299 pins, of which 255 are used for signals and 43 are used for power supply. The EMC-R chip is mounted on a ceramic PGA package.

The EM-4 testbed system is now operational, and is used for testing the EMC-R and the PE group board, and for developing basic software. Each PE group board has five EMC-Rs, network connection lines in-

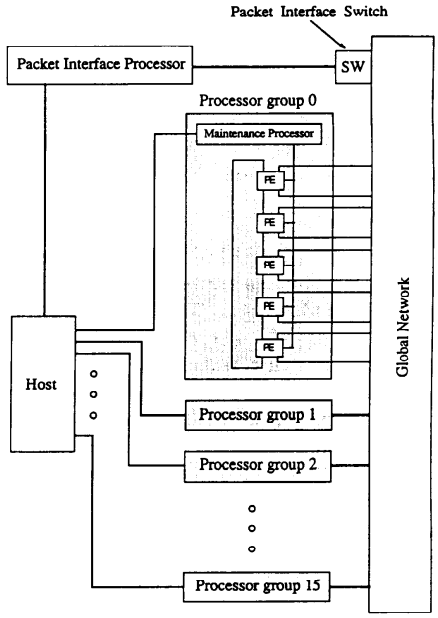


Fig. 10 The EM-4 Prototype.

side the group, a maintenance processor, timing coordination circuits, and some other circuits. An assembler, an initialization software, a basic monitor, and some other software are already usable. In the testbed, each PE has about 1.2 MB of off-chip SRAM.

The first version of the EM-4 prototype, which has 80 PEs, was constructed in April 1990. The aims of developing the prototype were to confirm the architectural design of the EM-4, to provide an environment for developing software, and to evaluate several aspects of the architecture by measuring dynamic characteristics of practical programs. Figure 10 shows the structure of the prototype. It consists of sixteen *PE group boards,* each group of which is the same as that on the testbed system, a *global network*, a *packet interface switch board,* a *packet interface processor,* and a *host computer.* A global network is implemented on two mother boards and with thirty-two cables connected between PE group boards. A packet interface switch offers a packet interface between packet interface processor and the prototype system. One of the roles of a packet interface processor is gathering information from the prototype, such as trace information and program results. It also provides system information, such as function skeletons and program-starting information as a packet. The host computer is a workstation whose role is the top-level control of the whole prototype system. It is connected by VME buses to maintenance processors, each of which is dedicated to the PE group maintenance. It is also connected to the packet interface processor to pass on machine information. Initialization, debugging, monitoring, display of results, and software development are ordered by the host com-

puter.

The peak performance of the prototype is now 1 GIPS. Our experience with the EM-4 prototype will soon show whether the EMC-R chip design is adequate or not, whether the network performance is sufficient or not, and whether the maintenance architecture is powerful enough or not.

5. Conclusion

This paper has described the basic design of the EMC-R from the viewpoint of advanced dataflow schemes and their implementations. The main features are model-level refinement with a strongly connected arc dataflow model, two simple and fast synchronization schemes, versatile pipeline design, RISC-based design, and interconnection networks with extra facilities. In addition, the paper has described the chip architecture that makes these features possible, and its implementation.

Several other advanced dataflow architectures are now being developed; for example, Iannucci's hybrid architecture [10], Nikhil's PRISC architecture [11], Gao's non-data flow architecture [16], and Amamiya's Datarol architecture [17]. The EM-4's distinctive features are as follows:

(1) A strongly connected arc model for versatile resource management and fast local execution, as described in Section 2.1

(2) Integration of two kinds of pipelines, as described in Section 2.3

(3) Separation of synchronization and execution, as described in Section 3.3

(4) Single-chip architecture with a data switching element.

In addition, the EMC-R is the only advanced dataflow processor that has actually been fabricated and is now fully functional. The EM-4 prototype with 80 EMC-R's has been implemented and the overall debugging phase finished. The performance evaluation will be presented in a forthcoming paper.

The abovementioned features are expected to accelerate the system performance significantly and to give system flexibility, which will not be obtained by any other architecture. These effects will be measured in the EM-4 prototype from the Spring of 1990.

The current problems with the EM-4 are to develop software and to show the actual effectiveness of the EM-4 architecture and the EMC-R chip. High-level language design and compiler design are the two of the most significant problems with the software. There will be two main high-level language. One is DFC-II [15], which will soon be available on the SIGMA-1 machine, and the other will be an advanced functional language based on EMLISP, which was developed on the EM-3 machine [9]. Two phases are being considered for a compiler. In the first phase, high-level languages are translated to an intermediate language, which expresses pure dataflow graphs. This is necessary for both high-level languages. The second phase creates the actual machine code. This includes the automatic creation of strongly connected blocks and optimal structure allocation.

Future problems are to construct a machine with 1,000 PEs, to consider a PE chip architecture containing far more transistors, to establish optimization methods for such systems, and to reconsider and expand the strongly connected arc model.

Acknowledgement

We wish to thank Dr. Akio Tojo, Director of the Computer Science Division and Mr. Toshio Shimada, Chief of the Computer Archtecture Section, for supporting this research, and the staff of the Computer Architecture Section for fruiful discussions.

References

1. AMAMIYA, M., TAKASUE, M., HASEGAWA, R. and MIKAMI, H. Implementation and Evaluation of a List-Processing-Oriented Data Flow Machine, *Proc. the 13th Annual Symp. on Computer Architecture* (June 1986), 10-19.
2. ARVIND, DERTOUZOS, M. L. and IANNUCCI, R. A. A Multiprocessor Emulation Facility, MIT-LCS Technical Report 302 (September 1983).
3. DENNIS, J. B., LIM, W. Y. P. and ACKERMAN, W. B. The MIT Dataflow Engincering Model, *Proc. IFIP Congress*83 (1983), 553-560.
4. GURD, J., KIRKHAM, C. C. and WATSON, I. The Manchester Prototype Dataflow Computer, *Comm. ACM*, **21**, 1 (1985), 34-52.
5. HIRAKI, K. SEKIGUTI, S. and SHIMADA, T. System Architecture of a Dataflow Supercomputer, TEN CON87, Seoul (1987), 1044-1049.
6. OTSUKA, R., SAKAI, S. and YUBA, T. Static Load Allocation in Dataflow Machines, *Proc. of Technical Group on Computer Architecture*, IECE Japan, CAS86-136, in Japanese (1986), 75-84.
7. SAKAI, S., YAMAGUCHI, Y., HIRAKI, K. and YUBA, T. Introduction of a Strongly-Connected-Arc Model in a Data-Driven Single-Chip Processor EMC-R, Proc. Dataflow Workshop 1987, IECE Japan, in Japanese (1987), 231-238.
8. SHIMADA, T., HIRAKI, K., NISHIDA, K. and SEKIGUCHI, S. Evaluation of a Prototype Data Flow Processor of the SIGMA-1 for Scientific Computations, Inf. Process. Lett., 16, 3 (1983), 139-143.
9. YAMAGUCHI, Y., TODA, K. and YUBA, T. A Performance Evaluation of a Lisp-Based Data-Driven Machine (EM-3), *Proc. 10th Annual Symp. on Comp. Arch.* (1983), 163-169.
10. IANNUCCI, R. A. Toward a Dataflow/von Neumann Hybrid Architecture, *Proc. 15th Annual Symp. on Comp. Arch.* (1988), 131-140.
11. NIKHIL, R. S. and ARVIND Can Dataflow Subsume von Neumann Computing?, *Proc. 16th Annual Symp. on Comput. Arch.* (1989), 262-272.
12. SAKAI, S., YAMAGUCHI, Y., HIRAKI, K., KODAMA, Y. and YUBA, T. An Architecture of a Dataflow Single-Chip Processor, *Proc. 16th Annual Symp. on Comp. Arch.* (1989), 46-53.
13. YAMAGUCHI, Y., SAKAI, S., HIRAKI, K., KODAMA, Y. and YUBA, T. An Architectural Design of a Highly Parallel Dataflow Machine, *Proc. of IFIP* 89 (1989), 1155-1160.
14. SAKAI, S., KODAMA, Y. and YAMAGUCHI, Y. Parallel Computer Organization with a Processor Connected Omega Network, Technical Report of IECE Japan, CPSY89-31, in Japanese (1989), 1-6.
15. SEKIGUCHI, S., SHIMADA, T. and HIRAKI, K. A Design of an Intermediate Language-SASGA-for Dataflow Computers, Technical Report of IECEJ, CPSY89-36, in Japanese (1989), 31-36.
16. GAO, G. R., TIO, R. and HUM, H. H. J. Design of an Efficient Dataflow Architecture without Dataflow, *Proc. of Intl. Conf. on FGCS* (1988), 861-868.
17. AMAMIYA, M. Datarol. An Optimized Dataflow Machine Architectute for Ultramultiprocessing, ACM/SIGARCH Workshop (1989).

(Recieved September 12, 1989)