to appear in Proc. Japan Soc. Parallel proc., Kobe Japan 1991, May 14-16 pp 149-156

Architectural Design of a Parallel Supercomputer EM-5

Shuichi SAKAI* Yuetsu KODAMA Yoshinori YAMAGUCHI

Electrotechnical Laboratory †

Abstract

This paper describes an architecture of a parallel supercomputer **EM-5**. The **EM-5** design objective is to construct a feasible parallel supercomputer whose target performance is over 1 TFLOPS, i.e. 10^{12} FLOPS. The design principles of the **EM-5** are: (1) a layered activation model; (2) an advanced direct matching scheme; (3) a highly fused pipeline; (4) a RISC processor **EMC-G** for a highly parallel computer; (5) a functional interconnection network; and (6) a maintenance architecture which can provide real-time monitoring facilities. After examining these features, this paper shows the architectural design of the **EM-5**, whose target structure will have 16,384 processing elements and whose peak performance is about 655 GIPS and 1.3 TFLOPS (double precision).

1 Introduction

A parallel supercomputer for general applications has been one of the most challenging themes in computer research and development. The more VLSI technology advances, the more significant feasibility studies of such a supercomputer becomes.

The authors have been developing a highly parallel computer based on a new data-driven model. As a first step, an **EM-4** system consisting of 1,024 single-chip processing elements (PEs) was designed, and an **EM-4** prototype system with 80 PEs was implemented [1] [2] [3] [4]. The prototype became fully operational in April 1990. It maximally provides 1 GIPS computation performance and 14.63 GB/s communication performance; in executing actual programs, it showed 999 MIPS.

The next research is split into two tasks, which are tightly coupled from/to each other. One is to design and implement a new parallel supercomputer which provides far more performance than the **EM-4**, and the other is to make a language system whose target machine will be both the **EM-4** prototype and the new supercomputer.

In order to achieve the former objective, a parallel supercomputer **EM-5**, whose target structure will have 16,384 PEs, is now under development. This paper presents the **EM-5**.

Before the circuit design of the **EM-5**, new execution model, new architectural schemes and new hardware primitives are proposed and examined. They are: (1) a layered activation model; (2) an advanced direct matching scheme; (3) a highly fused pipeline; (4) a RISC processor **EMC-G** for a highly parallel computer; (5) a functional interconnection network; and (6) a maintenance architecture which can provide real-time monitoring facilities.

The architectural design of the **EM-5**, which will realize all of the above features, is also outlined here. Each PE of the **EM-5** will contain the **EMC-G**, a floating-point unit (FPU), memories and some additional circuits. It will perform 40 MIPS and 80 MFLOPS, and thus the whole system with 16,384 PEs will perform 655 GIPS and 1.31 TFLOPS.

This paper first describes design principles and implementation of the **EM-4** (Section 2). Then it concentrates on the **EM-5** in Section 3, where its features are described in comparison with the **EM-4**. Next, programming paradigms and programming languages are considered to exploit the **EM-5** architecture (Section 4).

2 EM-4

2.1 Objectives and Design Principles

2.1.1 Objectives

The objective of the **EM-4** is to develop a feasible parallel supercomputer including more than 1,000 PEs for general use, e.g., for numerical computa-

*EMAIL: sakai@au-bon-pain.lcs.mit.edu

†1-1-4 Umezono, Tsukuba-shi, Ibaraki 305, JAPAN Phone: +81-298-58-5876. Fax: +81-298-58-5882.

tion, symbolic computation, and large scale simulations. The target performance is more than 12 GIPS. To achieve this objective, an improved dataflow architecture was proposed and adopted.

2.1.2 Strongly Connected Arc Dataflow Model

Although the basic **EM-4** architecture is based on the dataflow model, a new model called a *strongly connected arc model* was introduced to compensate for the pure dataflow architecture [1]. The strongly connected arc model highly improves efficiency and functionality of a dataflow parallel computer.

The model is defined as follows.

In a dataflow graph, arcs are categorized into two types: *normal arcs* and *strongly connected arcs*. A dataflow subgraph whose nodes are connected by strongly connected arcs is called a *strongly connected block* (SCB).

There are two firing rules. One is that a node on a dataflow graph is firable when all the input arcs have their own tokens (a normal data-driven rule). The other is that after each SCB fires, all the PEs which will execute a node in the block should execute nodes in the block exclusively. That is to say, if a node **A** whose output arc is a strongly connected arc is executed, then the nodes which are executable in the strongly connected block including the node **A** can only be executed by the concerned PEs.

If all the tokens from the outer world to the concerned block have already arrived and there are no firable nodes, then the strongly connected state is released and normal dataflow processing resumes on the concerned PEs.

In the **EM-4**, each SCB is executed in a single PE and tokens do not flow but are stored in a local register file. This property enables fast-register execution of a dataflow graph, realizes an advanced-control pipeline, and offers flexible resource management facilities.

2.1.3 Direct Matching Scheme

In order to reduce heavy overhead of packet matching, a new matching scheme, *a direct matching scheme* was introduced in the **EM-4**[2].

The direct matching scheme offers data matching with ordinary memory. Since the logic for realizing the scheme is fairly simple, it can easily be implemented using a small-size wired logic.

When a function is invoked, an instance of storage is allocated to a group of PEs. This instance

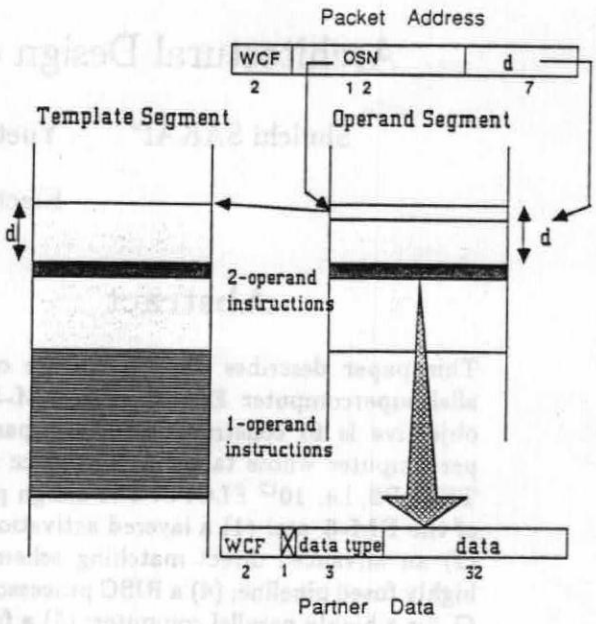

Figure 1: Direct Matching in the EM-4.

is called an *operand segment*. It is used for waiting and matching of operands. An operand segment has memory words whose number is equal to or larger than the number of two-operand instructions in the function. The compiled code for the function is stored in another area, which is called a *template segment*. The address of the instruction in the template segment has a one-to-one simple correspondence with that of the matching: each operand segment is linked with a template segment by storing the template segment number into the top word of the operand segment at the function invocation time; and, in the **EM-4**, an instruction displacement is set to the same one as that of the matching. Each packet thus only holds the matching address as an absolute memory address.

Matching is executed by checking the stored data in an operand segment and by storing the data if its partner is absent. If partner is present, then the instruction address is generated by combining a displacement and the template segment number stored at the top word of the operand segment. Then instruction fetch occurs while two operand data are forwarded to the execution unit (Figure 1).

This scheme eliminates the associative access overhead involving the hash mis-hitting overhead

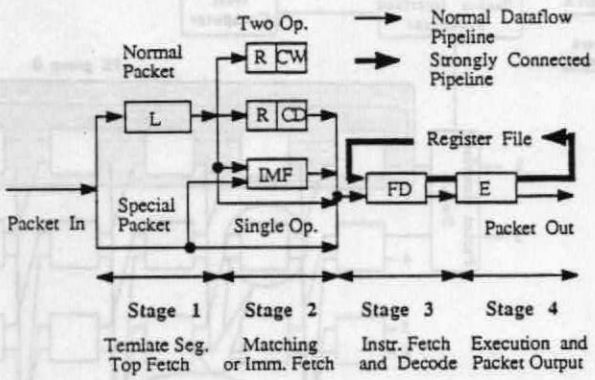

Figure 2: **EM-4** Pipeline Fusion.

and considerably reduces the matching hardware cost, at the expense of memory utilization.

2.1.4 Pipeline Fusion

In the **EM-4**, nodes in each SCB are executed on a register-based advanced-control pipeline. On the other hand, normal dataflow nodes are executed on a circular pipeline with a direct matching scheme. These two pipelines are fused naturally into a single pipeline, as shown in Figure 2. There are four stages in the circular pipeline: a linking stage, a matching stage, a fetch-and-decode stage and an execution stage. The latter two stages are shared with the register-based advanced-control pipeline. Activities belonging to more than one threads can simultaneously exist in the former pipeline. On the other hand, activities belonging to the same single thread can only exist in the latter pipeline.

There is no context switching overhead, since circular pipeline can immediately fill the pipeline stages when the current SCB is finished.

2.1.5 Multiple-RISC Concept

To accomplish a feasibility and high performance, one PE of the **EM-4** is realized in a single chip, called the **EMC-R**[1]. A Multiple-RISC concept is proposed and utilized in the **EMC-R**. This concept uses a RISC not only for serial instruction execution but also for parallel program execution. A Multiple-RISC processor should therefore have the following features: a small instruction set; few addressing modes; a register file architecture; no microprograms; few packet formats; simple embedded synchronization mechanisms. The **EM-4** satisfies all these conditions [3].

2.1.6 Circular Omega Network

Construction of an interconnection network with both a hardware cost of $O(N)$ (N : number of PEs) and a distance between any two processors of O(logN) was completed in order to realize fast communication with feasible hardware. The **EM-4** utilizes a "circular omega" network topology to satisfy these conditions [9]. The **EMC-R** contains both a data switching and a data processing facility which work concurrently and independently. In addition, the network in the **EM-4** contains store-and-forward deadlock prevention facilities and automatic load-balancing facilities.

All of the features described in this subsection distinguish the **EM-4** from the other advanced architectures, e.g. Iannucci's hybrid architecture [5], Gao's non-dataflow architecture [6], Nikhil's PRISC [7], and Amamiya's Datarol [8].

2.2 Architectural Design and Implementation

2.2.1 Single Chip Processor EMC-R

The **EM-4** consists of single chip processors, **EMC-Rs**. Figure 3 shows the organization of the **EMC-R**, consisting of five units and a maintenance controller. The Switching Unit (**SU**), an element of the circular omega network, is a three-by-three packet switch. The **SU** performs routing, arbitration, and packet transfer as its basic functions. Additionally, the **SU** has store-and-forward deadlock prevention facilities and function level dynamic load balancing facilities [9].

An Input Buffer Unit (**IBU**) is a buffer for packets waiting for matching and execution. A 32-word FIFO buffer is implemented on the **EMC-R** chip by a dual port RAM. There is also an 8K-word secondary buffer located in an off-chip memory.

A Fetch and Matching Unit (**FMU**) provides all the synchronizations and sequencing in the **EMC-R**. It contains data matching, instruction fetch, and pipeline control mechanisms.

An Execution Unit (**EXU**) is an instruction executor which contains an instruction register, instruction decoding circuits, an **ALU**, a multiplier, a barrel shifter, a comparator, a register file with sixteen registers, packet generation circuits, several multiplexors, additional registers and control logic circuits. All instructions except multiple

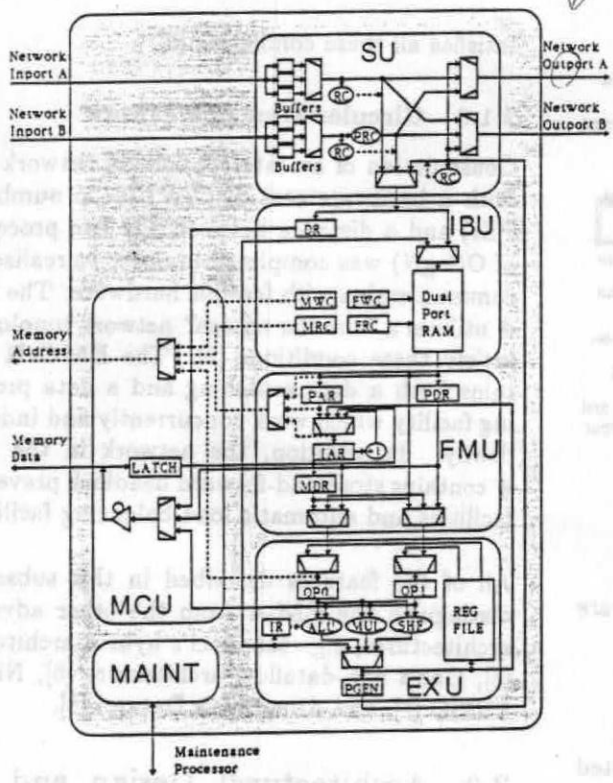

Figure 3: Organization of the **EMC-R**.

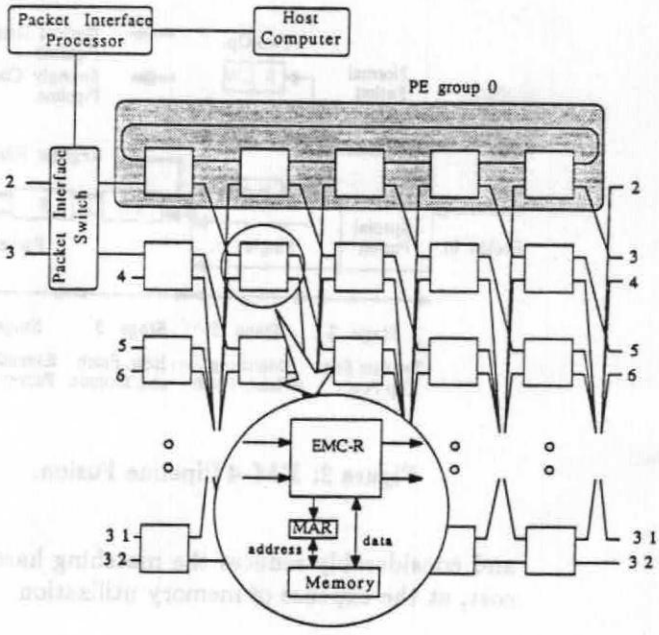

Figure 4: Organization of the **EM-4** Prototype.

memory read/write are performed within a single clock.

A Memory Control Unit (**MCU**) arbitrates memory access requests from the **IBU**, the **FMU**, and the **EXU**, and sets the address and data multiplexors so that data can be transferred to/from the selected unit. Memory requests may conflict among three units, where memory access by the **IBU** breaks the execution pipeline and degrades throughput.

The **EMC-R** is implemented in a CMOS gate array chip with 299 pins contained in it: 43 for power and ground supply, 255 for signals, and 1 unused. The chip contains 45,788 gates involving all the units and maintenance circuits. The design is based on the 1.5 micron rule with an inverter gate delay of 0.7 ns. The **EMC-R** uses an 80ns clock and every instruction needs only one clock for its execution, and thus the peak performance is 12.5 MIPS.

This chip has been produced since October 1989, and testing has proven there were no major bugs.

2.2.2 EM-4 Prototype

The 80 PE **EM-4** prototype was implemented as a first step. [1] Figure 4 illustrates the **EM-4** prototype's organization. Eighty PEs are divided into 16 groups, each of which has 5 PEs. A group is a unit for allocation of a function instance. Each PE consists of the **EMC-R**, a memory address register (**MAR**), and off-chip memories. The **EM-4** prototype is packaged into a $60.0cm \times 92.0cm \times 140.5cm$ single rack. Each group is realized on a four-layered multi-wired board. It is $50.8cm \times 47.0cm$ and is comprised of five **EMC-Rs**, static RAMs, **MARs**, drivers, and maintenance circuits. There are 16 PE group boards.

The maximum performance of the prototype is 1 GIPS and the maximum data transfer rate is 14.63 GB/s. Moreover, the maximum synchronization performance by dyadic data matching is 200 MSYPS, i.e. 200 Mega SYnchronizations Per Second.

The **EM-4** prototype has been fully operational since April 1990.

Initial performance evaluations of the **EM-4** with sample programs were reported in [4] and more advanced evaluations will appear in [10].

[1] Note that the **EMC-R** is designed for the 1,024 PE system; it thus has addressing facilities for the 1,024 PE network and 1024 PE memories.

3 EM-5

3.1 Objectives and Design Principles

3.1.1 Objectives

The objectives of the **EM-5** is to develop a feasible parallel supercomputer including more than 16,384 PEs for general use, e.g., for numerical computation, symbolic computation, and large scale simulations. The target performance is more than 1.3 TFLOPS, i.e., 1.3×10^{12} FLOPS (double precision), and 655 GIPS.

Unlike the **EM-4**, the **EM-5** is not a dataflow machine in any sense. It exploits side-effects and it treats location-oriented computation. In addition, the **EM-5** is a 64-bit machine while the **EM-4** is a 32-bit machine.

3.1.2 Layered Activation Model

The **EM-5** is based on a *layered activation model* as well as the **EM-4** was based on a strongly connected arc model.

In this model, data-driven arcs connect node blocks called *indivisible node blocks* (INBs). Each INB is fired by the arrival of tokens through the data-driven arcs, and after the INB fires, all the PEs which will execute a node in the block must execute nodes in the block exclusively.

The major differences between the layered activation model and the strongly connected arc model are:

1. The execution order of the INB nodes are determined not only by data dependencies. The **EM-5** is thus not a dataflow machine for execution within an INB.

2. Within an INB, side-effects can be exploited. This breaks the functionality of a dataflow model, but introduces much more flexibility.

3. There are global variables among INBs.

The layered activation model will be more accurately defined and examined in another paper.

3.1.3 Advanced Direct Matching Scheme

The **EM-5** uses a refined direct matching scheme, i.e. a *advanced direct matching scheme* for activating INBs. In this scheme, each packet flowing on a data-driven arc holds an operand segment number, operand segment displacement for matching, template segment number and template segment

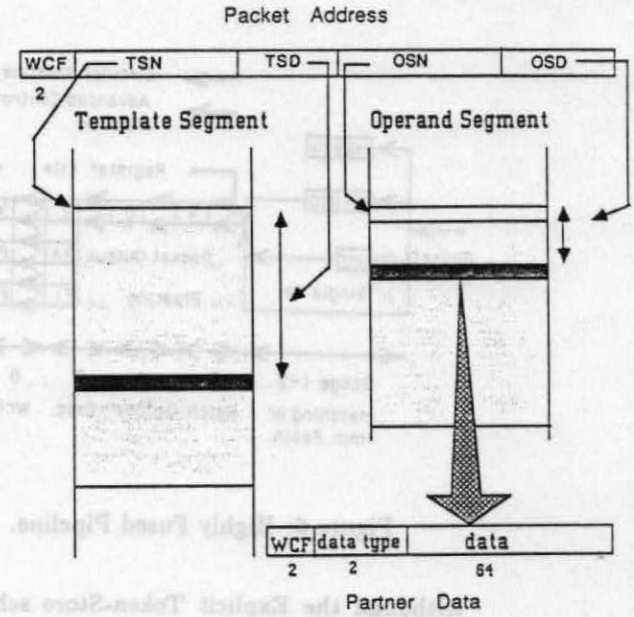

Figure 5: Advanced Direct Matching.

displacement for instruction fetch. Matching is executed by reading memory word whose address is indicated by the packet, checking a flag of the word, and sending a pair of data to the execution unit or writing the packet data into the memory (Figure 5). The differences between the advanced direct matching scheme in the **EM-5** and the direct matching scheme in the **EM-4** are as follows.

1. It is not necessary to store the information of linkage between the operand segment and the template segment, when a new function is invoked. This eliminates the overhead of function invocations.
2. The linking stage is removed from the PE pipeline. This shortens the circular pipeline, and thus shortens the turn-around time. Moreover, the matching memory is only accessed at the matching phase, which eliminates the inefficiency caused by a memory bus bottleneck.
3. Each matching word can be reused in the case where nodes are totally ordered. This greatly improves the memory space efficiency.
4. A displacement of a matching place is independent of that of a corresponding instruction. This brings flexibility of code generation and matching place allocation.

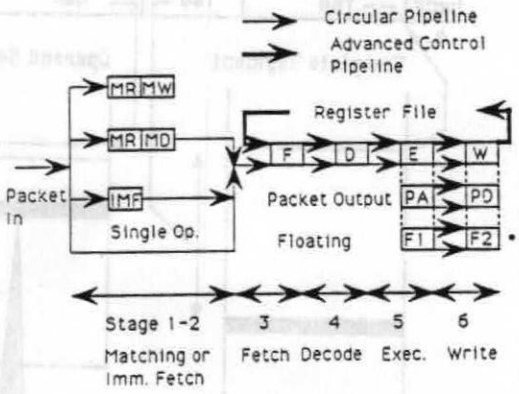

Figure 6: Highly Fused Pipeline.

Although the Explicit Token-Store scheme in the Monsoon Machine [11] also uses the absolute-address matching, the advanced direct matching does not need an extra instruction field nor a matching-address calculation for data synchronization since a displacement of the matching place is contained in each packet. This scheme realizes a much more efficient and flexible synchronization than Monsoon does at the cost of instruction field and packet field.

Besides the advanced direct matching, the **EM-5** contains several flexible synchronization mechanisms which will be closely examined in another paper.

3.1.4 Highly Fused Pipeline

Figure 6 illustrates the **EM-5** pipeline. It is also a fused pipeline of a circular pipeline and a register-based advanced-control pipeline. However, the following features make it considerably different from the **EM-4** pipeline.

1. By introducing the advanced direct matching scheme, the linking stage is eliminated. This brings efficiency and simplification of hardware.

2. Pipeline pitch is logically a half as long as that of the **EM-4** during the execution of an ICB. This doubles the execution throughput. Actually, the register-based advanced-control pipeline is the same one as the normal RISC pipeline, except the parallel operations of a calculation and packet-sending. Pipeline pitch will be 25 ns.

3. The **EM-5** also contains a floating-processor pipeline, which can be cooperatively operated with the **EMC-G** pipeline.

3.1.5 Multiple-RISC Concept

The features of the Multiple-RISC mentioned in 2.1.5 are also held in the **EMC-G**, a single chip processor of the **EM-5**. From the viewpoint of shortening execution time, the **EM-5** exploits the RISC feature more than the **EM-4**, since its pipeline pitch is logically twice as short as that of the **EM-4**.

Unlike the **EM-4**, there is no switching unit within the processor chip. There is no floating-point processor within in it either. This multi-chip design is adopted because of the limitation of pins and spaces in a chip. In other words, the multi-chip construction brings the fast double-sized floating-point calculations and flexibility of network construction at the cost of hardware and physical space.

3.1.6 Functional Interconnection Network

The circular omega network adopted in the **EM-4** will not be utilized in the **EM-5**, since a system with 16,384 PEs must exploit more clustered communication. So far, the topology of the interconnection network has not been determined yet.

It will have automatic load-distribution facilities and store-and-forward deadlock prevention facilities as the **EM-4** network has. Moreover, it will contain advanced facilities which will dynamically perform flow control by watching the network status.

3.2 Architectural Design

3.2.1 Single Chip Processor EMC-G and PE Configuration

Figure 7 shows the first version of the organization of the **EM-5** PE, including the **EMC-G**. The PE consists of the **EMC-G**, memories for secondary buffer, memories for instructions and data, and a floating-point unit. The **EMC-G** consists of four units and a maintenance controller.

The meaning of each unit in the **EMC-G** is almost as same as that in the **EMC-R**. The followings describe the features of the PE, examining the major differences between the **EMC-G** and the **EMC-R**.

1. As was previously mentioned, **EMC-G** adopts a 64-bit architecture while the **EMC-**

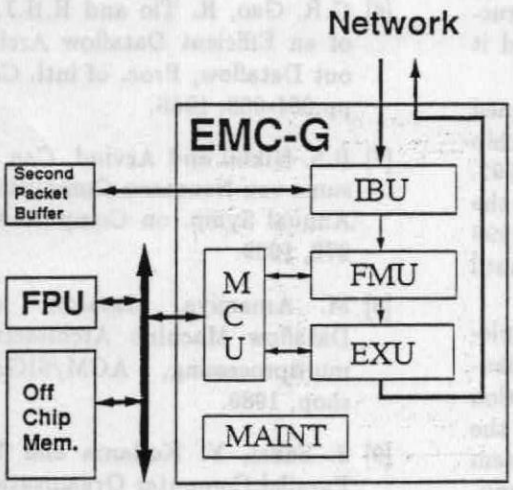

Figure 7: EM-5 PE.

R adopts a 32-bit architecture. Each data path consists of a 64-bit bus and each data register consists of a 64-bit register.

2. The EMC-G has a global addressing mechanism for a 16,384-PE system.

3. There is no embedded network switch within the EMC-G chip.

4. Pipeline pitch is logically half as long as that of the EMC-R.

5. An Input Buffer (IBU) has its dedicated memories whose bus is independent of the instruction/data memory. This prevents the pipeline break which occurs in the EMC-R, i.e. pipeline break caused by memory usage of the IBU.

6. A Fetch and Matching Unit (FMU) controls the synchronization and sequencing control. In addition, it controls the floating-point unit by an interlock.

7. An Execution Unit (EXU) is an instruction executor which contains operand registers, calculation circuits, a $32-word \times 64-bit$ register file. Floating-point operations are carried out by a floating point unit (FPU).

The EMC-G will be implemented in a CMOS standard-cell chip with 391 pins and about 100,000 gates. The design will be based on the 1.0 micron rule with an inverter gate delay of 0.4 ns. The EMC-G uses a 25 ns clock and every instruction needs only one clock for its execution, thus the peak performance is 40 MIPS.

The peak performance of the floating-point unit will be more than 80 MFLOPS. The maximum data transfer rate in each port is 335 MB/s.

3.2.2 EM-5 Organization

The EM-5 consists of 16,384 PEs, an interconnection network, maintenance processors, a secondary memory system and a host computer. The whole system will be constructed in a two-or-three-layered fashion.

The maximum performance will be 655 GIPS and 1.31 TFLOPS. The maximum data transfer rate will be about 5.49 TB/s. Moreover, the maximum synchronization performance by the dyadic data matching will be 131 GSYPS, i.e. 131 Giga SYnchronizations Per Second.

Network construction, maintenance mechanisms, and I/O mechanisms of the EM-5 will be described in another paper.

4 High Level Languages

A language system for DFC-II, data-flow-C version-2, is is now under development for both EM-4 and EM-5. It is a language with sequential description and parallel execution [12], while the first version of DFC was a pure functional language. The DFC-II can break a single-assignment rule and a program can contain global variables.

The compiler of the DFC-II produces the intermediate code, and the post-compiler will translate it into EM-4/EM-5 assembler.

The compiler and the post-compiler of the DFC-II will be fully operational on the EM-4 in 1991.

Besides the DFC-II, several other languages are planned to be implemented on the EM-4/EM-5, such as Id [13] and FORTRAN. In the near future, the language most suitable to a certain application will be selected among these high-level languages. Moreover, an object-oriented programming paradigm is now examined to be implemented on the EM-5[14].

5 Conclusion

This paper presented the design principles and the architecture of the EM-5 in comparison with the EM-4. It also described the language systems under development and examined the realization of

object oriented computation. The target structure of the **EM-5** will have 16,384 PEs, and it will perform 655 GIPS and 1.3 TFLOPS.

The PE of the **EM-5** will logically be designed in 1991. The whole gate design of the single chip processor **EMC-G** will be completed in 1992. The **EM-5** system with 16,384 PEs, including the interconnection network, will be designed in 1993 and its prototype system will be constructed until March 1994. (full size)

Future problems excluding the hardware implementation are: to design and implement the language systems; to further examine the execution model; to establish the scheduling methods on the EM-5; to run test programs and evaluate system performance; and to write and run actual application programs.

Acknowledgment

We wish to thank Dr. Toshitsugu Yuba, Director of the Computer Science Division, and Mr. Toshio Shimada, Chief of the Computer Architecture Section for supporting this research, and also the staff of the Computer Architecture Section for their fruitful discussions.

References

[1] S. Sakai, Y. Yamaguchi, K. Hiraki, Y. Kodama and T. Yuba, An Architecture of a Dataflow Single Chip Processor, Proc. 16th Annual Symp. on Comp. Arch., pp.46-53, 1989.

[2] Y. Yamaguchi, S. Sakai, K. Hiraki, Y. Kodama and T. Yuba, An Architectural Design of a Highly Parallel Dataflow Machine, Proc. IFIP Congress 89, pp. 1155-1160, 1989.

[3] Y. Kodama, S. Sakai and Y. Yamaguchi, A Prototype of a Highly Parallel Dataflow Machine **EM-4** and Its Preliminary Evaluation, Proc. of InfoJapan90 pp. 291-298, 1990.

[4] S. Sakai, Y. Kodama and Y. Yamaguchi, Prototype Implementation of a Highly Parallel Dataflow Machine **EM-4**, to appear in Proc. of 5th Int'l Parallel Processing Symp., 1991.

[5] R.A. Iannucci, Toward a Dataflow/von Neumann Hybrid Architecture, Proc. 15th Annual Symp. on Comp. Arch., pp. 131-140, 1988.

[6] G.R. Gao, R. Tio and H.H.J. Hum, Design of an Efficient Dataflow Architecture without Dataflow, Proc. of intl. Conf. on FGCS, pp.861-868, 1988.

[7] R.S. Nikhil and Arvind, Can Dataflow Subsume von Neumann Computing?, Proc. 16th Annual Symp. on Comput. Arch., pp. 262-272, 1989.

[8] M. Amamiya, Datarol: an Optimized Dataflow Machine Architecture for Ultra-multiprocessing, ACM/SIGARCH Workshop, 1989.

[9] S. Sakai, Y. Kodama and Y. Yamaguchi, Parallel Computer Organization with a Processor Connected Omega Network, Technical Report of IECEJ, CPSY89-31, in Japanese, pp. 1-6, 1989.

[10] Y. Kodama, S. Sakai and Y. Yamaguchi, **EM-4** Performance Measurements, to appear in Proc. of the 3rd Joint Symp. on Parallel Processing, in Japanese 1991.

[11] G.M. Papadopoulos, D.E. Culler, Monsoon: an Explicit Token-Store Architecture, Proc. 17th Annual Symp. on Comput. Arch., pp. 82-91, 1990.

[12] S. Sekiguchi, T. Shimada and K. Hiraki, A Design of an Intermediate Language - SASGA - for Dataflow Computers, Technical Report of IECEJ, CPSY89-36, in Japanese, pp.31-36, 1989.

[13] R. S. Nikhil and Arvind, Id: a Language with Implicit Parallelism, MIT-LCS Computation Structure Group Memo 305, 1990.

[14] S. Sakai, Y. Kodama and Y. Yamaguchi, Object Oriented View of a Parallel Supercomputer **EM-5**, WOOC91, 1991.