MODULE 5

Dataflow Computer Architectures

DATAFLOW ARCHITECTURES

The static dataflow model was proposed by Dennis and his research group at MIT [16]. The general organization of the Static Dataflow Machine is depicted in Figure 3.

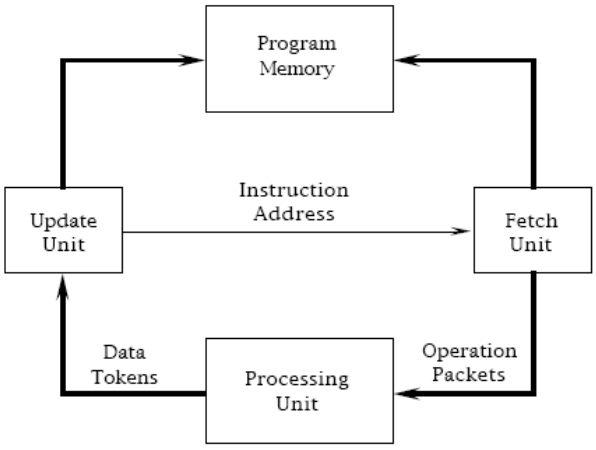


Figure 3: The basic organization of the static dataflow model.

The Program Memory contains instruction templates which represent the nodes in a dataflow graph. Each instruction template contains an operation code, slots for the operands, and

destination addresses (Figure 4).

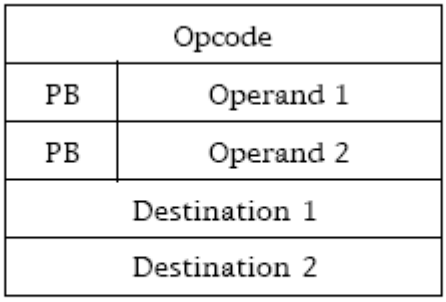


Figure 4: An instruction template for the static dataflow model.

To determine the availability of the operands, slots contain presence bits (PBs). The Update Unit is responsible for detecting the executability of instructions. When this condition is verified, the Update Unit sends the address of the enabled instruction to the Fetch Unit

The Fetch Unit fetches and sends a complete operation packet containing the corresponding opcode, data, and destination list to the Processing Unit and also clears the presence bits. The Processing Unit performs the operation, forms a result packets,and sends them to the Update Unit. The Update Unit stores each result in the appropriate operand slot and checks the presence bits to determine whether the activity is enabled.

The dynamic dataflow model was proposed by Arvind at MIT [5] and by Gurd and Watson at the University of Manchester [24]. The basic organization of the dynamic dataflow model is shown

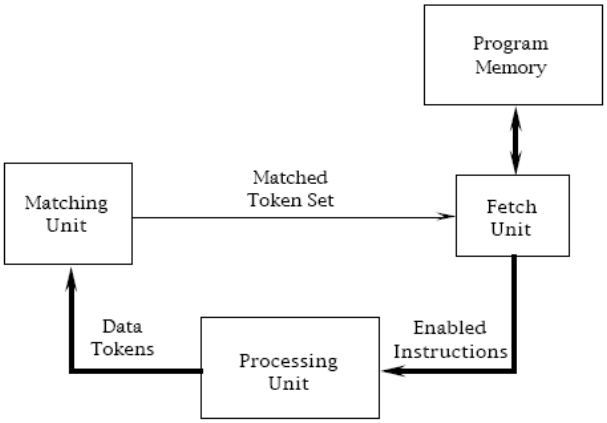


Figure 5: The basic organization of the dynamic dataflow model.

in Figure 5. Tokens are received by the Matching Unit, which is a memory containing a pool of waiting tokens. The basic operation of the Matching Unit is to bring together tokens with identical tags. If a match exists, the corresponding token is extracted from the Matching Unit and the matched token set is passed on to the Fetch Unit. If no match is found, the token is stored in the Matching Unit to await a partner. In the Fetch Unit, the tags of the token pair uniquely identify an instruction to be fetched from the Program Memory. A typical instruction format for the dynamic dataflow model is

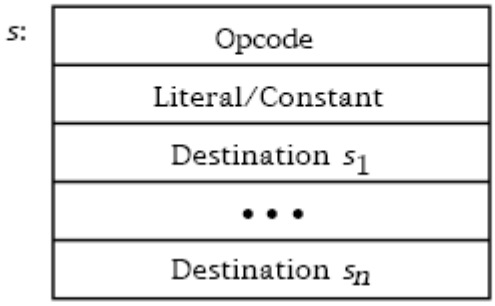


Figure 6: An instruction format for the dataflow model.

shown in Figure 6.

It consists of an operational code, a literal/constant field, and destination fields. The instruction together with the token pair forms the enabled instruction and is sent to the Processing Unit. The

Processing Unit executes the enabled instructions and produces result tokens to be sent to the Matching Unit.

Note that dataflow architectures can also be classified as centralized or distributed systems based on the organization of their instruction memories. In a centralized organization, the communication cost of conveying one token from one actor to another is independent of the actual allocation instruction in the program memory. In a distributed organization, the instructions are distributed among the processing elements (PEs).

Therefore, inter-PE communication costs are higher than intra-PE communication costs.

The major advantage of the dynamic over the static dataflow model of computation is the higher performance that can be obtained by allowing multiple tokens to exist on an arc.

Dataflow Computer Architectures

Data flow computer A large, very powerful computer that has a number of processors all physically wired together with a large amount of memory and backing storage. Such computers are highly parallel in that they can carry out a large number of tasks at the same time.. Data flow computers are used to execute processor intensive applications such as those associated with areas like molecular biology and simulation.

Static Data Flow Computers

The main components of a static data flow computer are

Processing section: This part does operations on data tokens

Memory section: It has the instruction cell that holds all the instructions and the operands.

Control network delivers a control token from processing section to memory section.

Distribution network deliver data token from processing section to memory section.

The working of a static data flow computer is simple, it almost like a loop. Here is the working in simple terms

Memory section has the all instructions in them and these instructions are enabled for execution by the arrival operands in data token from the distribution network and the arrival of control tokens from control network. The enabled instructions along with the operands are sending to the processing section. There it is processed, the result tokens produced are sent to the memory section through the distribution network and control tokens are sending to the memory section through the control section. These then act operands of other instructions in the memory section.

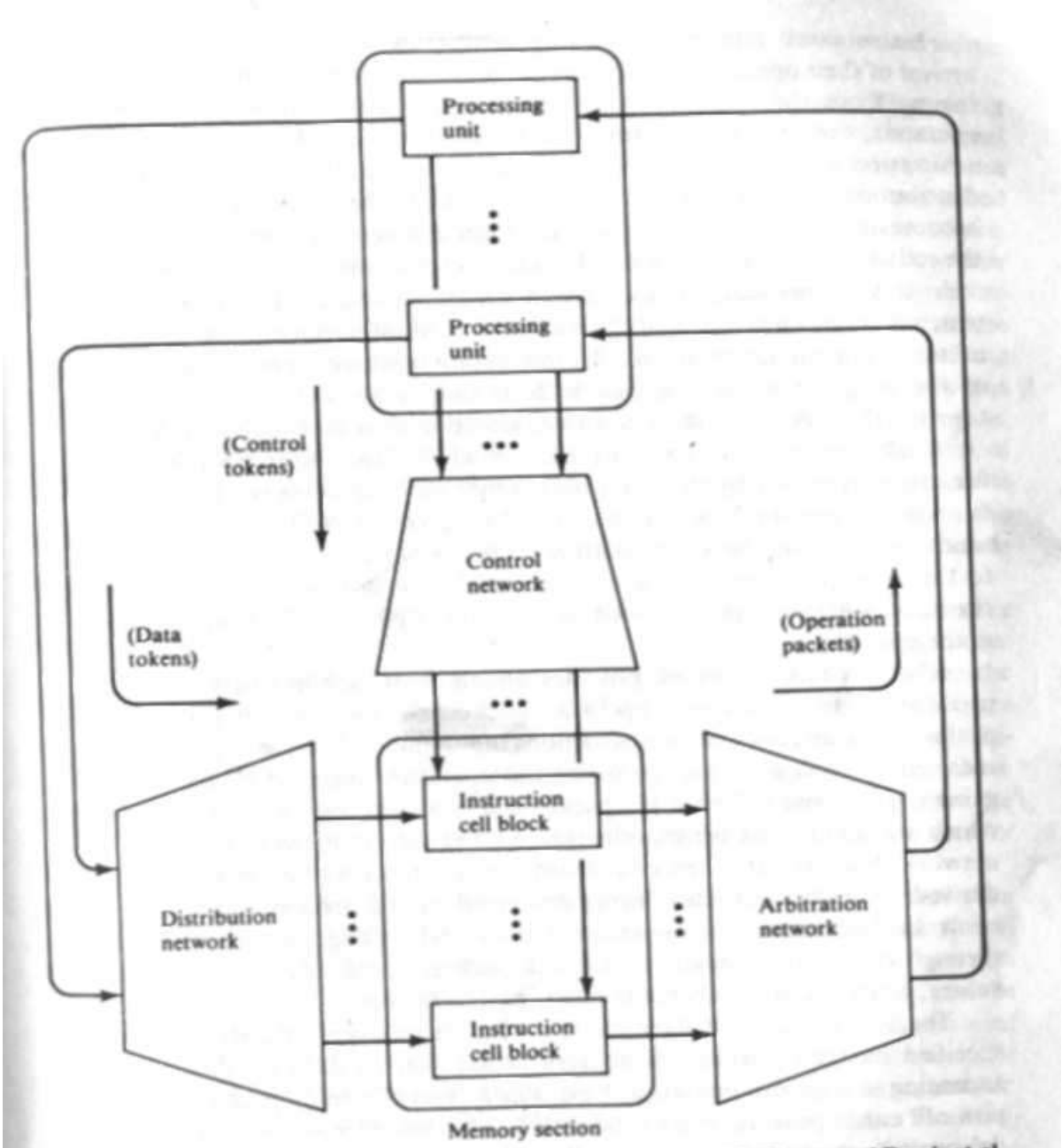


Figure 10.10 The static dataflow computer architecture proposed at MIT. (Courtesy of Dennis et al., 1979.)

Memory section has instruction cells. An instruction cell that is occupied contains an instruction consisting of operation code and several destinations. The destinations contain destinations address which is a cell identifier. It also contains additional control information which is used by

the processing units to generate result tokens. The cell identifier for each cell is unique. Each instruction cell has receivers in them which wait for the token values for use as operands by the instruction. When the cell has received appropriate acknowledge signals and operand token the cell becomes enabled and sends an operation packet consisting of the instruction and the operand values to the arbitration network.

The arbitration network provides connection from each instruction cell to each processing unit. Arbitration network also sorts the operation packets among its output port according to the operation code of the instruction they contain. Decoding of operation codes is partially done here.

The result tokens have result value and destination address, the destination address is a cell identifier. The control tokens have boolean values. These two values in-turn act as operands for another instruction cell.

The communication between all the sections is using packet transmission over channels.

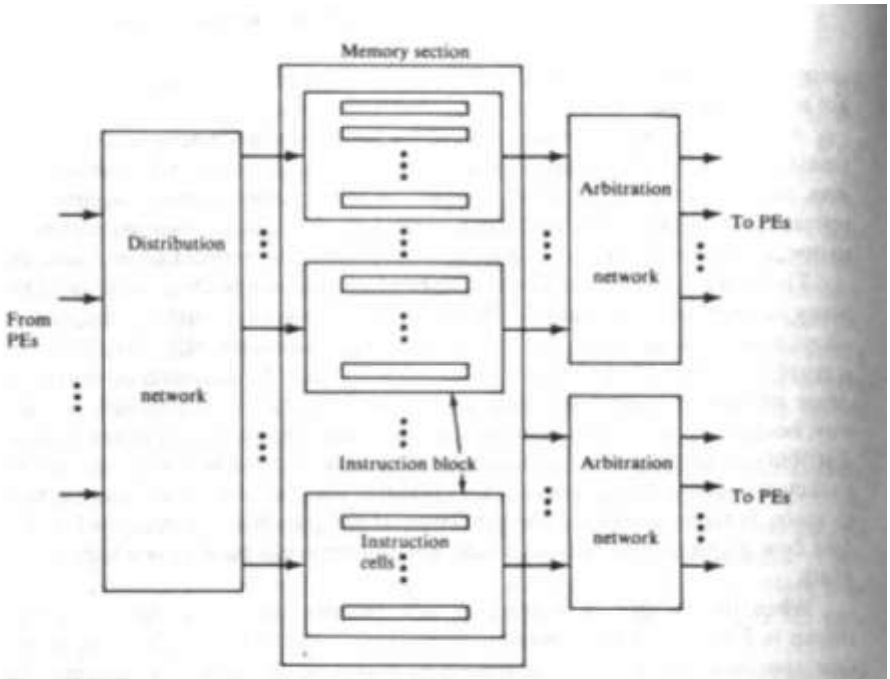


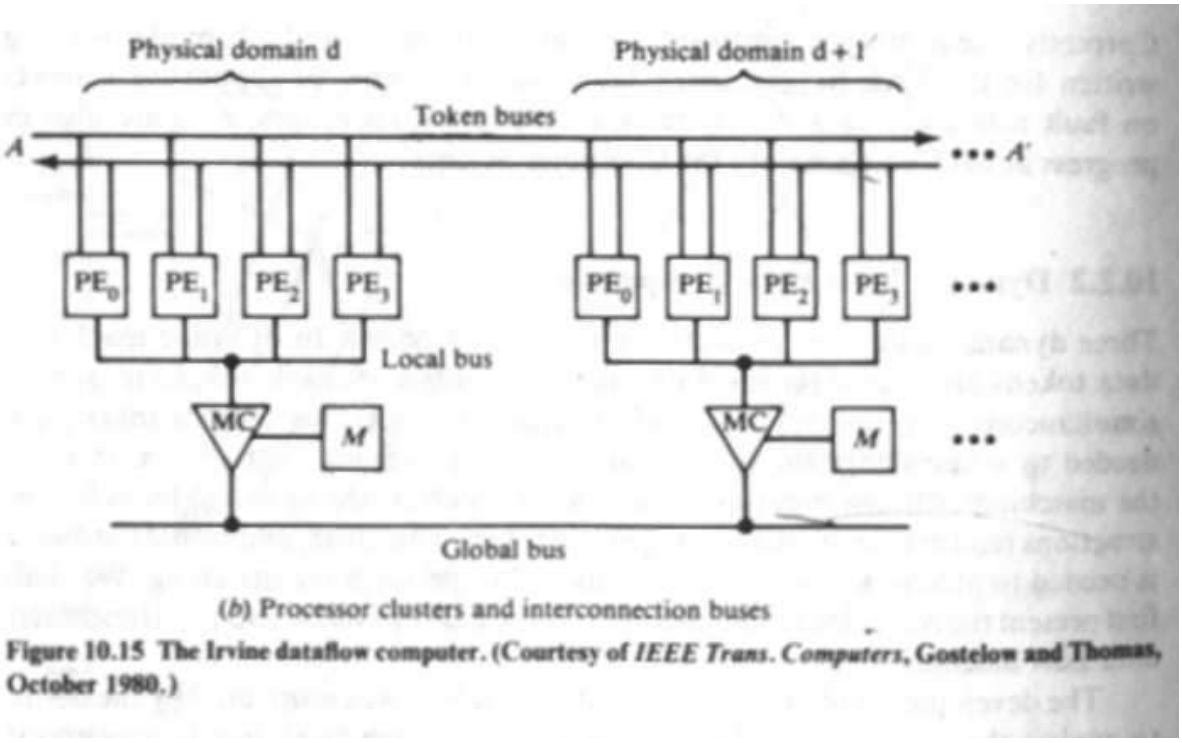
Figure 10.11 The concept of grouping instruction cells into cell blocks.

Dynamic Data Flow Computer

In a dynamic data flow computer the data tokens are tagged or colored. Because of this property multiple tokens can arrive on any input of a operator node. Matching of token tags are done to merge them for instructions requiring more than one operand token. So unlike static data flow computer no control tokens are needed here. Dynamic data flow computer also requires additional hardware for attaching the tags to data tokens and to perform tag matching.

Irvine data flow machine

The structure of Irvine data flow machine is shown below.



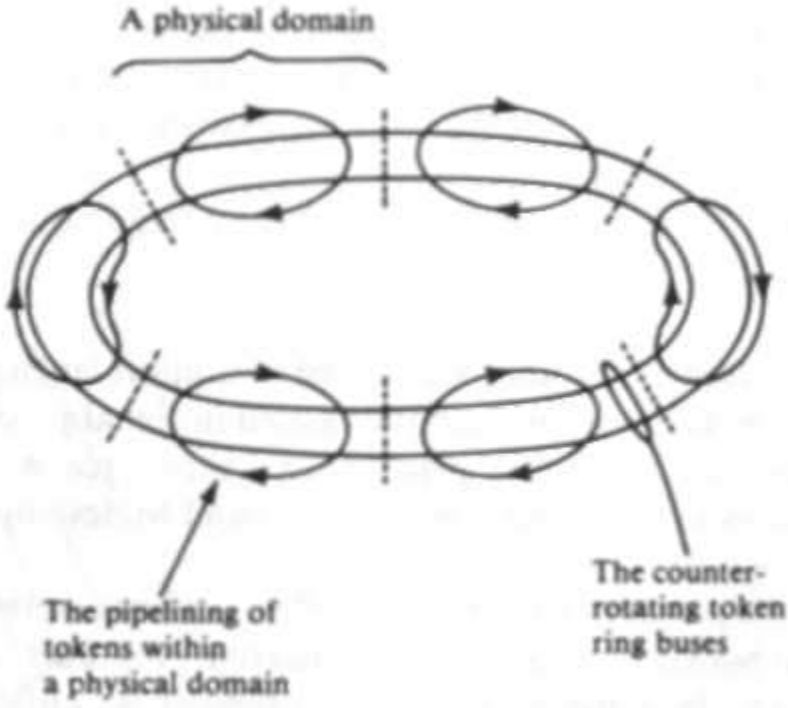
(*b*) Processor clusters and interconnection buses

Figure 10.15 The Irvine dataflow computer. (Courtesy of *IEEE Trans. Computers*, Gostelow and Thomas, October 1980.)

Irvine data flow machine has multiple PE clusters and all these PE clusters can work concurrently. The PE clusters are connected by two system buses. Tagged tokens are transferred among the PE using token rings. The token bus is a pair of bidirectional shift-register rings. Each ring has a number of slots and the number of slots is equal to the number of PE. Each slot may or may not have data token. A local bus and memory controller is used to share local memory

among cluster of a particular domain. A global bus is used to transfer data among different physical domains.

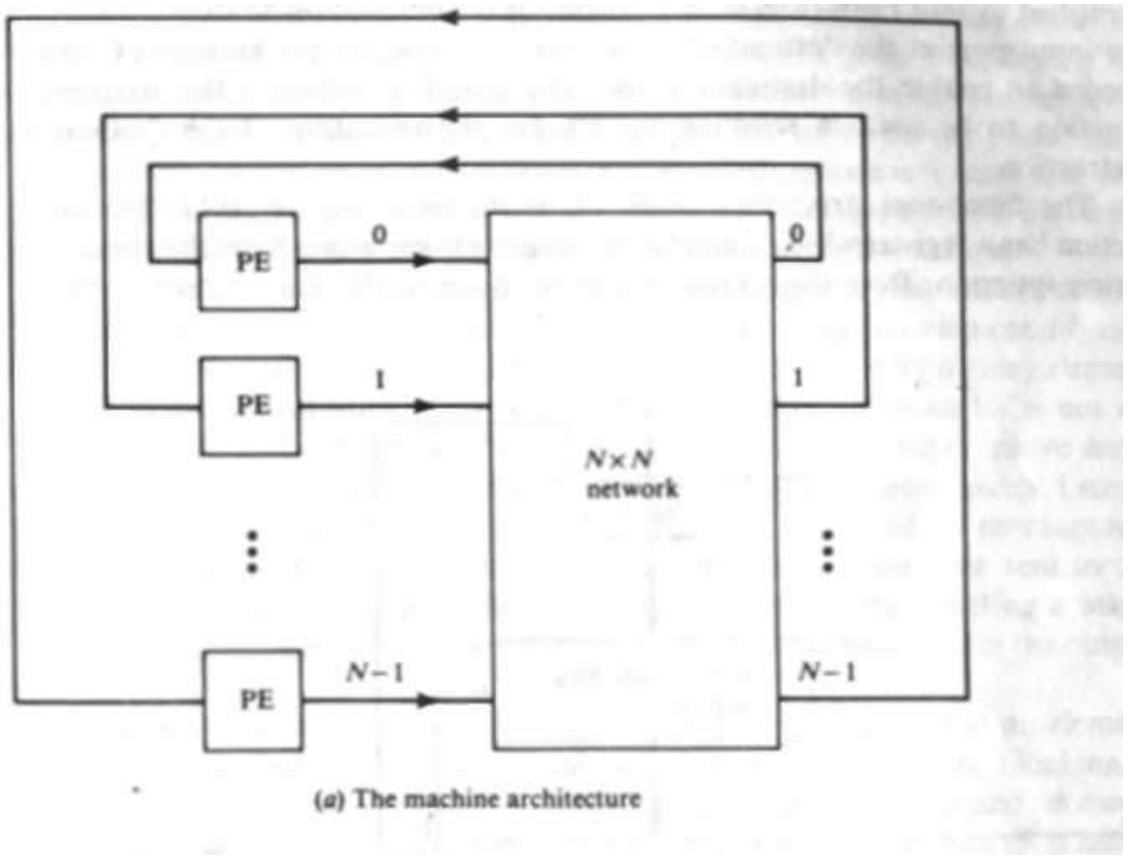
A PE must accept all tokens sent to it. After receiving a token the PE sorts all the tokens in it according to the activity name. If all input tokens for a particular activity has arrived (checking is by token matching) the PE will mutate that activity.



(a) Token rings and local pipelining

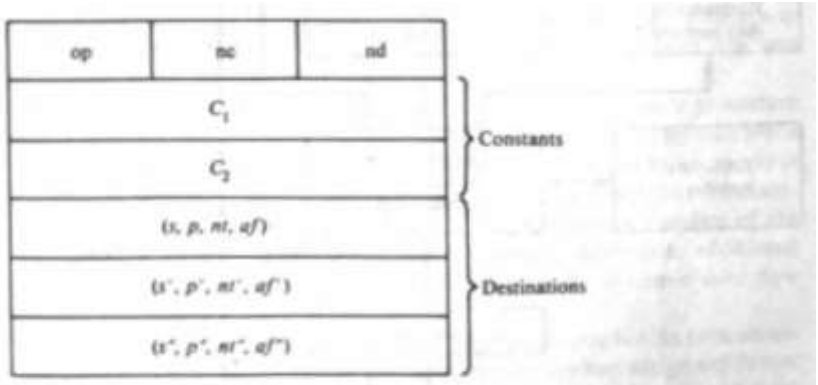
Arvind machine

It is a modification if Irvin machine. In this for inter PE communication we use N x N packet switch network instead of token. Each PE in the system is a complete computer. The activities are divided among PE according to the mapping from tags to PE numbers. A statistically chosen assignment function is used by each PE to determine the destination PE number.



(a) The machine architecture

The general format of an instruction is given above



(b) A typical instruction

Figure 10.16 Arvind's dataflow machine organization and instruction format. (Courtesy of Arvind et al., Sept. 1980.)

OP- opcode

nc -number of constants (max two) stored in the instruction,

nd - number of destinations for the result token

Fields in the destination

S – Destination address

p -input port at the destination instruction

nt - number of tokens which is needed to enable the destination instruction

af -assignment function used in selecting the PE

The structure of an individual PE is shown below

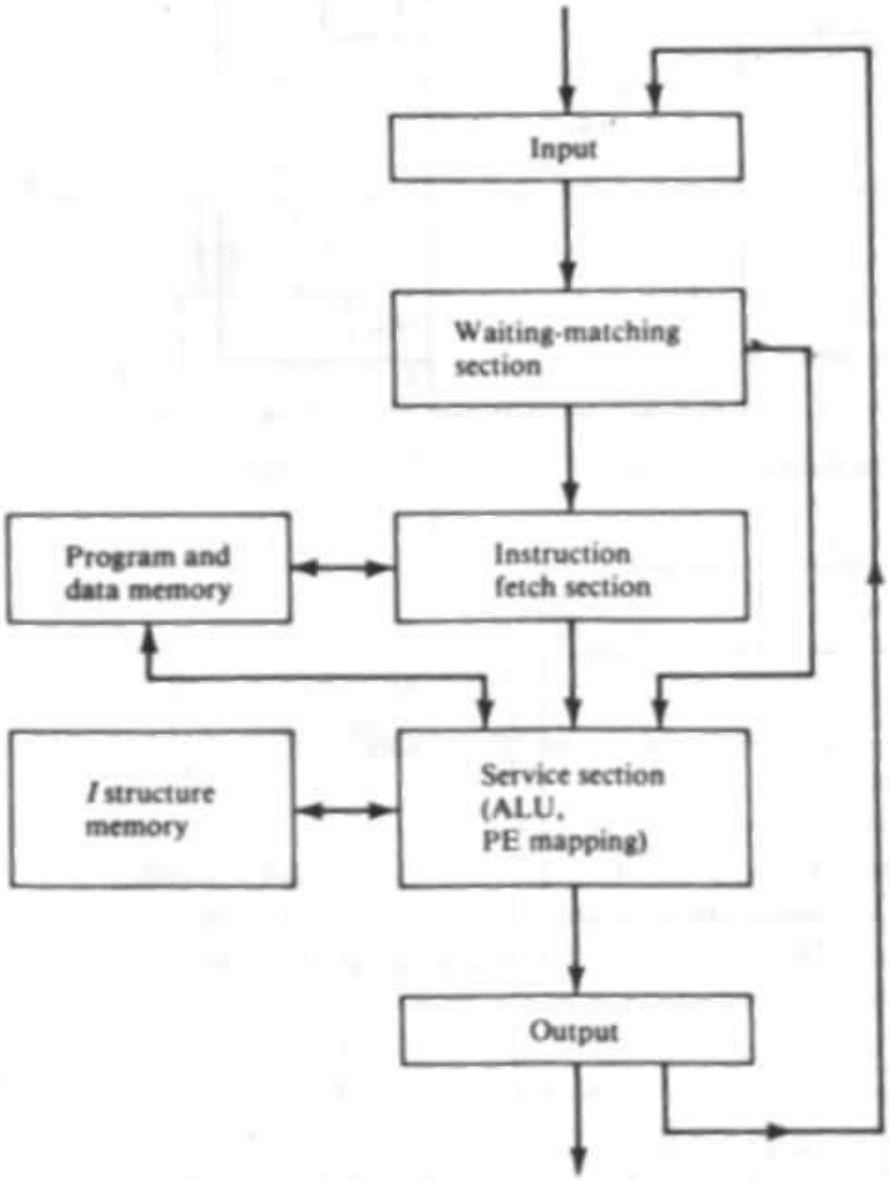


Figure 10.17 The processing element (PE) in Arvind's dataflow machine at MIT.

The working is fairly simple

Token to a PE can come from the output section of the same PE or from communication system. The input section has registers which if empty can store these tokens. Next the tokens are passed to the waiting matching c where it waits until another token with suitable doesn't come.

Whenever the tags of the token match, the tokens are passed to the instruction fetch buffer. But a problem can arise here, suppose the waiting matching section is full and another token comes .If the last arrived token does not match this will lead to a dead lock. So if a buffer-full condition exists the token has to be stored is somewhere else.

When matching tags are found, an instruction from the local program memory is fetched based on the statement number part of the tag. After instruction fetch an operation packet containing operation code, operands and the destinations is formed and is given to service section. The service section has a floating-point ALU and some hardware to calculate the new activity names and destination PE numbers. After ALU functions the output is given to the out section but since we may have some delay in communication network some buffer space is provided in the output section