circulation of the delay line data a search and operation can be performed on the memory contents.

Since the comparison logic is serial and therefore simple, this organization is well suited to systems using only a few delay lines. It was not used in the NEBULA application because 32 delay lines were required and the 32 35-bit shift registers and the special logic required to resolve multiple matches between the 32 delay lines were considered too costly.

ACKNOWLEDGMENT

The author is indebted to D. L. Amort, R. Smets, F. W. Weingarten, and F. H. Young for suggestions on the system designs, and to the many people on the NEBULA team for assistance in the system construction. Criticisms of the manuscript by B. A. Crane, G. D. Bergland, J. A. Githens, D. E. Wilson, and the reviewers were also appreciated.

The Design of a Highly Parallel Computer Organization

LOUIS J. KOCZELA, MEMBER, IEEE, AND GARY Y. WANG, MEMBER, IEEE

Abstract—The design of a computer organization is presented for general-purpose computing with a very high tolerance of failure while taking advantage of future large-scale integration techniques. A high degree of parallel computations may be executed. The organization distributes processing logic along with memory to form "cells." The architecture and operation of the cells are developed. Design aspects of the cells are presented along with software considerations of the organization.

Index Terms—Computer architecture, digital computer organization, distributed processing, global control, logical design, parallel computation.

INTRODUCTION

The results of a research study of an advanced computer organization concept are presented in this paper [1]. The concept is based on a highly parallel organization and will be specifically referred to as the distributed processor. The advanced computer organization concept evolved from a previous study of advanced multiprocessor organization concepts for future mission applications [2], [3]. A representative future space mission (manned Mars) of the 1980 time frame was selected as a base for computational requirements of this study and large-scale integration (LSI) semiconductor technology was extrapolated to this time frame to form a base for the study. The distributed processor organization was therefore designed to provide general-purpose computing and high tolerance of failures while taking advantage of future LSI techniques. An easily expandable or contractable system was desired to meet a variety of computational requirements. The resulting organizational concept may have a wide variety of future general-purpose computational applications even though it was originally designed for a spaceborne application.

Fig. 1 diagrams the organizational concept which consists of a number of identical interconnected cells. The cells are divided into groups which are connected by an intergroup bus for communication. The cells within each group communicate with each other by an intercell bus and by neighbor communication lines. Each cell consists of a general-purpose processor section and a small amount of memory on a single LSI wafer. One cell in each group is designated as a controller cell and the re-

References

remaining cells may be operated independently of the controller cell.

The organization concept exhibits the capability for computational parallelism since its cells may be operated simultaneously and in parallel. An important distinction between this organization and previously developed highly parallel organizations is that its cells may be dependent upon or operated independently of the controller cell so that there exists the capability for either global or local cell control which can result in efficiently mechanizing parallel computations, as developed in the next section.

High reliability by graceful degradation is achieved, i.e., as cells fail, either spare cells replace the failures or, if no spares are available, only a small percentage of the computational power is lost. Increased flexibility is inherent since the system easily expands and contracts in small increments to meet requirements.

The technology, to be considered as state-of-the-art for the time period of interest, was defined. Although the mission was considered for the 1980's, it was necessary for the hardware designers to have the required technology available considerably before the missions are actually flown; therefore, technology was projected 10 years from the present.

The two leading device-oriented technologies that lead the developments in LSI today are bipolar and metal oxide semiconductor field effect transistors (MOS-FET). Bipolar devices offer a speed advantage over MOS-FETs, while MOS-FET devices offer the advantages of a simpler manufacturing process, less area for the same function, and lower power dissipation compared to bipolar [4], [5]. MOS-FET technology was selected for investigating the organization. One batch fabrication technology being developed which bears important implications on future LSI is the heteroepitaxy (growth) of thin semiconductor films in which active devices are fabricated. This technique is exemplified in the growth of single crystal silicon-on-sapphire (SOS) by chemical vapor deposition. Application of this technology to MOS (MOS-SOS) is presently in the research and development phase. By extrapolation of present results, it was determined that large wafers (approximately 1.5 inches in diameter) with $10^6$ or more devices in the time frame considered may be expected.

Development of high-density LSI arrays for use in near term and future computation systems will make the processor or arithmetic and control sections of computers increasingly simple in comparison to the memories. This trend is evidenced by the fact that semiconductor memories are being investigated for the main memory in future computation systems [6]. These memories may use many semiconductor chips, whereas a processor may only use one or two such chips. As the processing of semiconductor chips matures, the reliability of the one- or two-chip processors will also be significantly greater than that of the magnetic or semiconductor memory. It is clear that there is a need for new computer organizations capable of taking advantage of LSI technology in order to enhance computation systems. One such organization is the distributed processor discussed here. This organization may be considered an array of identical semiconductor wafers or chips with a large part of each wafer devoted to memory and a small part of each wafer (less than 1/10) devoted to processing. The organization thereby integrates processing and memory onto the same wafer or chip and uses a number of these chips to form the organization. Such a structure uses a small hardware increase over a conventional organization in order to offer increased reliability, flexibility and execution speed for certain types of computations. These advantages are explained later. The organization is considered somewhat tech-

Fig. 1. Distributed processor organization.
nology dependent as it is desirable to have each cell manufactured on a single wafer, thereby resulting in one identical wafer throughout the entire organization. However, this is not a rigid restriction since more than one wafer could be used to construct a cell.

**Parallelism**

*Introduction and Definitions*

Parallelism within computations was investigated in order to provide a basis for evaluating various parallel or distributed computer organizations. Two types of parallelism—applied and natural—are defined as follows: applied parallelism is the property of a set of computations which enables a number of groups of identical operations within the set to be processed simultaneously on distinct or the same data bases; natural parallelism is the property of a set of computations that enables a number of groups of operations within the set to be processed simultaneously and independently on distinct or the same data bases.

Applied parallelism is a special case of natural parallelism, since the naturally parallel operations could be groups of identical operations. The distinction between the two types is introduced since it has important implications with regard to computer organization. The two types of parallelism are illustrated with the simple example in Fig. 2. The example is the computation of the expression \( a/x + b/x + cy = Z \). Fig. 2 illustrates how the expression is computed in a sequential manner. The numbers above each circle indicate the time that might be required to compute each term on a conventional sequential computer \((S)\). If applied parallelism were capable of being taken advantage of, the computation on such a machine would take place as illustrated in Fig. 3. The term \( A/S \) in this figure is the ratio of the time required on the machine with the capability for executing applied parallelism to the time required on the sequential machine. It should also be noted that a degree of applied parallelism of 2 is utilized in Fig. 3 (this occurred during the parallel computation of \( a/x \) and \( b/x \)).

If the capability for taking advantage of natural parallelism is now introduced, then the computation may take the form illustrated in Fig. 4. It should be noted that the term \( cy \) may now be computed in parallel with \( a/x \) and \( b/x \). The total computation illustrated by Fig. 4 may be classified as utilizing natural parallelism; however, it should be noted that a subset of this may actually be classified as applied parallelism as indicated by the dashed lines. Therefore, the computation may be said to consist of applied and natural parallelism. This combination may be referred to as total parallelism. Utilization of total parallelism results in the reduction ratio \( T/S \) in Fig. 4.

**Assignment and Sequencing of Parallel Operations**

It may be seen from the above simple example that an important part of any parallelism analysis is the formation of computation graphs to study the degree of parallelism utilized and the inherent reductions in computation time or "effectiveness of parallelism." Unfortunately, the graphs are quite complex to construct for problems of practical interest.

It was found that, in general, the approaches to determining this effectiveness of parallelism or obtaining the "parallelism" curve (degree of parallelism utilized versus computation time reduction ratio) are analogous to problems proposed and studied in the area of operations research under the title of job shop scheduling or assembly line balancing. Considerable research has been carried out in this field of operations research and several references [7]–[13] were reviewed to determine if any of it could be extended to this investigation of parallelism. Evaluation of these references and others showed that there are no general solutions to the optimum sequencing problem applicable to the study of parallelism here. The only solutions to the problem are generally very lengthy computational programs considering all permutations of the solution space to find a minimum or computational algorithms to arrive at a suboptimum solution by restricting the solution space. Analytical procedures are nonexistent for general solutions.

**Application of Parallelism Studies**

The spaceborne computational problem was analyzed to determine the inherent parallelism [1]. Fig. 5 contains the applied parallelism speed curve. It shows the computation reduction ratio versus the degree of applied parallelism available in the computation system. The 100-percent utilization curve in the figure is the 1:1 curve, i.e., for a degree of parallelism of 2, the computation reduction ratio is 2, for 5 it is 5, etc. The actual curve at first deviates slowly from the 1:1 curve, and then reaches an asymptotic reduction ratio value of 13.66 for higher degrees of parallelism. The knee of the curve occurs at approximately a degree of 15; beyond this degree the curve deviates sharply from the 1:1 curve. Fig. 6 contains the natural or total parallelism speed curve. A higher reduction ratio is achieved, as compared to Fig. 5, for any degree of parallelism. The curve does not have as sharp a knee as in Fig. 5. However, somewhere in the range of 40 to 60 in degree of parallelism, the curve starts to deviate rapidly from the 1:1 curve. These results indicate the amount of parallelism that may be efficiently utilized in a computer system for the particular spaceborne application considered. This provides some basis for the number of cells in the organization for the particular application considered.

**Development of the Organization**

Applied and natural parallelism were discussed above. The distinction between these two types of parallelism is important with regard to the organization of a highly
Fig. 2. Sequential steps in computation.

Fig. 3. Applied parallelism in the computation.

Fig. 4. Applied and natural parallelism in the computation.

Fig. 5. Applied parallelism speed curve.

Fig. 6. Natural parallelism speed curve.
parallel computer. Global control in a computer organization implies common control of a number of processing sections by a central control unit, while local control implies control of a processing section by its own control unit. From the definitions of parallelism, global control is used to efficiently handle applied parallelism since a central control unit provides common instruction and data storage. Local control is required to handle naturally parallel (exclusive of any applied parallel) computations since, by definition, the parallel operations may be different.

The Solomon machine [14], [15] is a good example of an organization primarily designed for global control, and the Holland machine [16], [17] of one primarily designed for local control. Both of these organizations have limitations, depending upon the type of application they are to handle. The problem is that they are not adaptable to general-purpose applications. Each of these types of organizations exhibit very explicit advantages for certain types of applications.

If both local and global control features are efficiently combined into one structure, the advantages of a machine of one type with the advantages of the other type are realized. The disadvantages associated with each type are reduced or eliminated. Such a structure is the distributed processor and is explained below.

The organization of the distributed processor is shown in Fig. 1. The distributed processor structure consists of cells. These cells are interconnected to form a group, and a number of groups are interconnected to form the organization. A cell may be thought of as a small conventional computer that has a relatively small memory. The structure may also be thought of as a number of memory modules, each with its individual processing section. These modules are known as cells and contain 512 16-bit words of memory per cell. This amount of storage is derived primarily from technology considerations in that it is desirable to have each cell mechanized on a single LSI wafer. However, the amount of storage per cell may be increased by using more than one LSI wafer to mechanize a cell. Parallel operation may exist between cells in a group and between the groups in the system. Therefore, the organization appears as a highly parallel computational facility.

The cells in a group and in the entire system are all identical in terms of hardware. However, at any particular time, some cells are functionally operated differently from other cells. Each group has one cell designated as a controller cell. The remaining cells in the group are designated as working cells. The controller cell is responsible for controlling the intercell communication bus and providing the executive control of the group’s operation. All of the cells in a group are connected to the intercell communication bus. This bus serves as the primary means of communication among the cells in a group. The intercell bus is used for communicating data between cells and for the transmission of global instructions and commands to cells. The cell that has the unique function of the controller cell controls the use of the intercell bus and provides all the global instructions and commands.

The working cells in a group either accept and execute the global instructions that are sent on the intercell bus or fetch and execute instructions from their own memory. In fact, some cells may use the global instructions from the intercell bus, while others use instructions from their own individual memory. Since all of the cells are physically identical, they are functionally capable of existing in different states. Thus, both local and global control may be carried out simultaneously within a group. As a result, both the natural and applied parallelism inherent in a group’s tasks may be efficiently carried out.

Only cells operating under global control use global instructions and commands from the intercell bus. However, all cells in the group respond to commands sent to individual cells via the intercell bus, whether the cell operates under local or global control (these commands are addressed to one particular individual cell). Consequently, the controller cell can command one or more cells operating under global control simultaneously, while cells operating under local control must be commanded or talked to on an individual basis. Global control implies that one or more cells are using this type of control. The controller cell provides all control information sent on the intercell bus. Therefore, cells operated in a global control mode are highly dependent upon the controller cell since they receive instructions from it and are responsive to its global commands. However, the cell operated in a local control mode is considerably more independent of the controller cell, since it fetches its instructions from its own memory and must be individually talked to or commanded. In general, the controller cell has the overall control of the cells in a group. The degree of this control can be varied, depending upon the functional use of each cell.

A simplified block diagram of a cell is shown in Fig. 7. The cell consists of a memory and arithmetic and control section in a manner similar to conventional computers. In addition, the cell contains logic for intercell bus communications and for its own identification. This is the part of the cell that differs significantly from conventional computers.

The organization consists of a number of groups interconnected by an intergroup communication bus, as shown in Fig. 1. Each of the groups can operate independently of, and simultaneously with, the other groups. Therefore, another level of parallelism is introduced into the organization because it is possible to simultaneously have more than one group in operation where each group is utilizing local and global control to carry out its tasks. Each group is connected to the intergroup
bus by a group switch, as shown in Fig. 1. The group switch is controlled by the controller cell of the group. One of the groups contains additional functions in its controller cell to operate as the system executive group. This primarily involves coordinating and controlling the communications that take place on the intergroup bus. Since all of the cells are identical, the system executive can be located in any one or more of the groups.

Many factors determine the number of cells in a group and the number of groups to be used. From the requirements and the parallelism studies, it was determined to use four groups of 20 cells each (512 words per cell) for the spaceborne application considered. [With more than one group it is possible to simultaneously have in execution more than one computation using applied parallelism (global control) since the groups can each have one global program being executed.]

**Architecture**

The distributed processor system consists of groups, which are made up of cells. Because the cells in a group are connected by neighbor communication lines and the controller cell sends global instructions to cells, the group is the fundamental unit of the computing system. This section describes the features desirable to unify the cells. The topics covered in this section are detailed in [1].

**Cell States**

Although all cells are identical in hardware, a cell always exists functionally in one of seven different and mutually exclusive states. These states are listed in Table I.

Independent cells are functionally similar to a conventional computer and fetch all instructions and operands from their own memories. The cell in the independent state stays in this state until the controller cell sends it a command on the intercell bus to change states. The independent cells can process problems that are not amenable to global processing, i.e., by local control.

Dependent cells respond to global instructions and global level commands sent out from the controller cell. A dependent cell exists in state 4), 5), or 6), depending upon the level of instructions being sent from the controller cell and the cell’s level register contents. (The concept of levels is described below under cell identification.)

A dependent cell in the global state (also called the active state) receives instructions from the controller cell via the intercell bus. The level of the instructions and the cell’s level register are the same in this state. As mentioned above, the instructions being sent from the controller cell to dependent cells are identified as being at a certain level. A dependent cell that is not at the proper level to receive global instructions can idle and not execute instructions. This is the wait state. Therefore, if the controller cell is servicing certain dependent cells, other dependent cells can wait their turn for service.

Instead of waiting for the controller cell to send in-

---

**TABLE I**

<table>
<thead>
<tr>
<th>Cell States</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Permanently failed—power off</td>
</tr>
<tr>
<td>2) Shut down—power saving state</td>
</tr>
<tr>
<td>3) Independent</td>
</tr>
<tr>
<td>4) Dependent under global control (global state)</td>
</tr>
<tr>
<td>5) Dependent under local control</td>
</tr>
<tr>
<td>6) Dependent in wait state</td>
</tr>
<tr>
<td>7) Controller cell</td>
</tr>
</tbody>
</table>
instructions for its level, a dependent cell can fetch and execute instructions from its own memory. This is the dependent under local control state. This state appears similar to the independent state in that both cells operate in a local control mode. The basic difference between them is that the dependent local control state implies that the cell is dependent, and dependent cells respond to certain global commands on the intercell bus, whereas independent cells do not. The dependent local cells are under more control of the controller cell than the independent cells. This state has advantages when using a cell partly for global programs (global control) and partly for local programs (local control).

The seventh state of a cell is the controller state. In this state, a cell controls the intercell bus and can issue global instructions. The ground rule for making all the cells of the same hardware allows any cell to become controller cell, providing the advantage that the controller cell functions may be switched among several cells, thus, there is no requirement that all the executive and controller programs fit in one cell.

**Cell Identification**

Two methods exist for identifying individual cells. One method identifies the cells at one of eight levels and the other gives each cell a unique identifier known as the cell address. Thus, a cell has a common first name (level) and a unique last name (identifier). This concept of having two names is important when discussing the dependent and independent cells.

Independent cells use only their identifier, or cell address. The level (or first name) is not used, and although present in a level register, has no meaning unless the cell later assumes a dependent state.

Dependent cells use both names. The controller cell can send out information using a first name (level number) to all the dependent cells. All the dependent cells at this level respond. If a last name (cell address) is sent, only the cell with this name responds since each cell has a unique last name, regardless of state. Consequently, the controller cell communicates with independent cells individually. However, the controller cell may communicate with dependent cells individually or to more than one at the same time, because the dependent cells respond to levels and more than one cell may have the same level.

**Source of Instructions**

Unlike a traditional computer which has instructions stored in its memory (which are always available to the processor), the distributed processor has different sources of instructions depending upon the state of a cell. The independent cell receives all its instructions from the cell's memory, like the traditional computer. The program counter is used to control the fetch of instructions.

In the dependent global state the cell receives its instructions from the controller cell. These cells receive instructions from the intercell bus, then execute them. The controller cell precedes the instructions with a name. The name (level number) is contained in a control word which is sent on the intercell bus and is prefix to a group of instructions. This prefix is the level of all instructions until a new level prefix or another control instruction is sent.

Every dependent cell compares the level prefix sent by the controller cell to the level register contents contained in the cell. If the prefix and the level register contents are different, the cell ignores all the instructions, data, etc., sent by the controller cell until a new level prefix (or other control word) is placed on the bus by the controller cell.

The dependent cell not receiving instructions from the intercell bus can fetch instructions from its own memory. This cell is in the dependent local control state. The fetch of instructions is identical to an independent cell. This cell is constantly examining the bus for a command at its level as noted above.

The controller cell always fetches instructions from its own memory. The instructions destined to be executed by the dependent global cells are not executed by the controller cell. All other instructions are executed by the controller cell and are not sent to the global cells.

**Execution of Instructions**

The instruction repertoire of a cell consists of conventional instructions such as load register, store register, operate on a register, compare, jump, etc. In addition, two unique classes of instructions are included, i.e., controller cell (CC) instructions and global control (GC) instructions. Some of the functions of the CC instructions are 1) control the intercell bus communications section of the cell; 2) generate the GC instructions; 3) generate intercell bus communications control signals; 4) control the transmission/execution of instructions in the controller cell; and 5) control the state or level identification of the cell. The cell identified as the controller cell is capable of executing all the CC instructions. Only a very restricted part of the CC instructions can be executed by independent and dependent local cells.

GC instructions are actually instructions or commands sent over the intercell bus by the controller cell. They are generated by the execution of a CC instruction and are then sent over the intercell bus. Some of the functions of the GC instructions are 1) control the state of dependent cells on the basis of level identification, and 2) control the state and/or level of individual cells upon the basis of address identification.

The execution of the instructions depends on the state of the cell and where the addresses, instructions, and data are located. Some of the more important differences in the execution of the instructions are given below.
Dependent global cells execute instructions received over the intercell bus. A notable difference over conventional instruction execution occurs with the compare and skip instruction categories, the difference being that the program counter is not used in dependent global cells and the results of executing these instructions is to change the level register and change the state to dependent wait. CC instructions are not sent over the intercell bus and are therefore not executed in dependent global cells.

Dependent cells under local control execute instructions similarly to dependent global cells. Of course, the instructions in this case are located in the cell's own memory. Cells in this state may execute a very restricted part of the CC instructions, namely, those to control the state and/or level of the cell. In addition, cells in this state will execute certain GC instructions received over the intercell bus, namely, those to control the state of dependent cells on the basis of level identification and those to control the state and/or level of individual cells upon the basis of address identification.

Independent cell instruction execution is quite similar to that in a traditional computer. Just as in the dependent local state, only a very restricted part of the CC instructions can be executed. The only GC instructions on the intercell bus that will be executed by an independent cell are those that control state and/or level on the basis of cell address identification.

The controller cell operates in basically two modes as far as instruction execution is concerned, i.e., the transmit mode and the execute mode. The controller cell can either send out instructions over the intercell bus or execute them internally. The transmit mode is entered by executing a particular CC instruction. Essentially, this instruction causes the controller cell to place the subsequent memory words on the intercell bus. The program counter controls the fetch of instructions. The execute mode is also entered by executing a particular CC instruction. In this mode the instruction execution is quite similar to the independent cell. The primary difference occurs in the CC and GC instruction execution. The controller cell can execute all the CC instructions (it is thereby capable of sending out instructions over the intercell bus), but it does not execute any GC instructions since it does not examine the intercell bus for any instructions.

**Input-Output**

The input–output method is shown in Fig. 1. Input–output devices may be connected to the cells directly via a single serial line, to the intercell busses (byte parallel), or to the intergroup bus (byte parallel). All connections are made via the I/O connection panel. It is assumed that the sensors will contain their own conditioners so that the signals will be binary digital in nature.

A number of alternative I/O approaches were considered. Most of the other approaches utilized separate I/O cells in the system. These approaches have the advantage of eliminating one connection from each cell; however, the system is now more prone to failure due to the specialized I/O cells. The selected approach offers the possibility of using any cell as an I/O cell. Failure in I/O cells may require the plugging–unplugging of connections to affect a reconfiguration; however, an entire group or system is not lost due to individual failures with the selected approach.

**Failure Detection and Reconfiguration**

Many approaches to detecting failures and subsequent reconfiguration were considered. The selected detection method consists of a combination of hardware and software self test routines. Reconfiguration procedures depend on the type of failures such as a working cell failure or a controller cell failure. Reconfiguration is carried out under software control.

**Cell Design**

The primary consideration in defining the features of the processor section of a cell was their ability to save memory storage, or total bits of memory. In general, it is desired to minimize the total amount of hardware required. However, it is expected that the memory section of a cell will require approximately 9/10 of the area of the wafer, whereas the processor section only approximately 1/10. Therefore, since most of each cell is devoted to memory, it is necessary to consider features that could be added to the processor section to reduce the amount of memory required. There is also some impetus to minimize the amount of hardware devoted to the processor section. This increases yields of the cell LSI wafers. This may be understood by realizing that the memory section primarily consists of a regular geometric connection structure of gates and registers, whereas the processor section is characterized by an irregular connection structure. Therefore, if discretionary wiring techniques are utilized, yields can be increased on cell wafers by incorporating redundant memory logic. It is relatively difficult to incorporate redundant logic in the processor section to increase yields. In general, it is necessary to trade off the increase in processor complexity to the amount of memory savings for a given feature. These considerations resulted in providing as much of the processor registers as possible in the memory section of the cell (this includes accumulators and index-bank registers).

The processor section contains four accumulators. The basic instruction and data word length is 16 bits and is shown below.

<table>
<thead>
<tr>
<th>Bits</th>
<th>1:6</th>
<th>7:9</th>
<th>10:16</th>
</tr>
</thead>
<tbody>
<tr>
<td>op code</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B/T</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address Displacement</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This format provides for 64 basic op codes for instructions and a 128-work bank from the address displacement. Three bits are provided for the index/bank registers (referred to as B and T registers). The selected use of the three B/T bits is to provide for the specification of one of two B registers in conjunction with one or none of three T registers. This scheme provides five index/bank specifications and the capability of double indexing. (Bit seven may be thought of as the B bit and bits eight and nine as T bits.)

In addition to the accumulators, the cell contains conventional registers and control circuitry such as the memory address register, program counter, compare condition flip-flops, etc. However, some rather unique control registers are required such as a controller cell mode, cell ID, cell state, and cell level.

**Group Switch**

The group switch controls the information transfer between the intercell bus and the intergroup bus. It is relatively simple in terms of hardware complexity and contains two buffer communications registers (one for each of the busses); an address ID register for identification purposes; a command register for holding commands received from the system executive group; a request register for holding requests from the controller cell of the group of which it is a part; and control circuitry for reception and generation of bus commands.

**Communication Busses**

The basic philosophy behind the design of the busses is that each cell is capable of addressing only its own memory directly. Other cells’ memories must be accessed by a request type scheme over a common bus.

Intercell communications within a group are carried out over the group’s intercell bus. The intercell bus is under complete control of the cell in the group designated as the controller cell. This cell operates the bus by the execution of certain CC instructions. Communication takes place in basically two types of modes, i.e., local and global. Local use is basically communication between two cells with control set up on the basis of cell address identification and not dealing with the control of levels or states of cells. Global use implies communication of the controller cell with one or more of the other cells, with control set up on the basis of cell address identification, levels, or states for the purpose of global control of the cells. Software routines are set up in the controller cell to control bus operation.

A total of 10 lines are used for the intercell bus. One line is used to denote control, or data, and is designated the control line. Another line is used for parity. The remaining eight lines are used for control, or data words. The control words are decoded by all the cells and only the appropriate cells partake in or accomplish the desired communication on the intercell bus.

Some of the types of intercell communications to be carried out include 1) controller cell sending or receiving data to/from a cell under controller cell’s command; 2) controller cell commanding communication between two cells; 3) controller cell scanning cells for communications requests and establishing the requested communications; and 4) controller cell issuing GC (global control) instructions to one or more cells.

Intergroup communications are accomplished by using the intercell and the intergroup busses. This communication is somewhat different from that given above for intercell communications. The difference is that the cells in a group are under immediate control of the controller cell and immediately respond to commands over the intercell bus. However, the executive group that is in charge of controlling the intergroup bus use does not have the groups (represented by the group switches) under its immediate control. Each group periodically samples its group switches to see if any commands have been placed in it by the executive group. (Any group can do this; however, only the group acting as the executive group will be issuing such commands.) The groups then respond to these commands. There is some delay in getting a response to these commands. The delay is dependent on the rate at which the group switch is sampled by the controller cell of a group for these commands. Once the controller cell picks up these commands, it immediately responds by commanding the proper cells in its group to begin communications.

The groups place requests for intergroup communications in their group switch. These requests are periodically sampled by the executive group. Based upon these requests and any other individual requests of the executive group, commands are placed in the group switches for intergroup bus control.

**Software Analysis**

The distributed processor has many levels of control. This results in many levels of control (executive action) in the software to program the computer. The executive programs may be considered at three levels: 1) the system executive, 2) the group executive, and 3) the cell executive.

The group executive has many functions. Basically, it controls the operation of the cells in the group and the intercell communication bus. It reconfigures the group resources due to changes in processing requirements and due to failures. Finally, it must interface with other groups.

The group executive provides executive services for the operational cells such as getting I/O data over the intercell bus, getting intermediate computational data from other operational cells, setting clocks, etc. These services are provided in two modes—periodic and background. Periodic services are required at fixed predetermined intervals of time. Background services are fitted between the periodic, according to some priority scheme. The services are provided by command or request. Command services are completely controlled by the controller cell; e.g., the controller cell commanding
one cell to send another cell certain data. Request services are requested by the operational cells; e.g., an operational cell requesting data from another cell.

To allocate system resources and schedule the use of these resources, there is made available to an executive scheduler program in the group executive a list of the hardware resources in the group and a list of tasks performed by the group. In addition, a list of the software requirements of each task is made available. The executive scheduler, using these lists, begins to allocate the resources to the tasks. Periodic tasks are first assigned according to priority, then the background tasks are assigned. The first assignment primarily consists of deriving the intercell bus usage schedule. The tasks are then physically assigned to cells.

One of the group executives also contains additional functions, giving it the capability of a system executive. The system executive is designed to coordinate the groups but not to control them. By this means, the system reliability is increased. The system executive contains a complete up-to-date list of the total hardware resources in the computer system and is responsible for assigning the tasks throughout the computer system. The system executive is responsible for control of the intergroup bus in much the same manner as the group executive is responsible for control of the intercell bus.

Each cell in a group requires a cell executive routine. This routine varies in size and complexity from cell to cell depending on the specific tasks to be performed in each cell. In any case, the cell executive must be a small, minimal program. A major part of the cell executive is that of processing interrupts. Some of the types of interrupts are 1) machine errors, 2) intercell bus I/O, 3) illegal data such as attempted division by zero, 4) illegal operation, and 5) real-time clock goes to zero.

ACKNOWLEDGMENT

The authors wish to express their appreciation to those who assisted in the development of the distributed processor organization. Particular thanks are extended to P. N. Bogue for his efforts in group architecture and software analysis, and to G. J. Burnett for his efforts in the evaluation of parallel computer organizations.

REFERENCES