計算機アーキテクチャ59−11
オペレーティング・システム29−11
(1985・10・25)

A Distributed-Configuration for Commanding Executing Systems

分散型コマンド実行システム

Issam A. HAMID* Tadao NAKAMURA* Yoshiharu SHIGEI*

イサム A. ハミド* 中村維男* 重井芳治*

*Department of Information Science,
Faculty of Engineering, Tohoku University

*東北大学 工学部 情報工学科

1- Introduction

Parallelism had to be introduced in varying degrees to the sequential von Neumann machine[1] to have highly specified systems, capable of multi-tasking operations. There are two types of parallelism applied and natural. Applied parallelism is the property of a set of computation that enables a number of groups of identical operations within the set to be processed simultaneously on different or the same data bases. Natural parallelism is the property of a set of computation that enables a number of groups of operations within the set to be processed simultaneously and independently on distinct or the same data bases.

There are two particular organizations properly put the distributed processor organization in perspective. The Solomon machine[2] that consists of a number of processing units under control of a centeral control unit(CCU). The parallel processing operates under control of a CCU. This organization exhibits a high efficiency in the execution of the applied parallel operation, because of parallel application of single instruction to different local data. The significant disadvantage of a global control (GC) structure is that it cannot efficiently; be applied to most general purpose computers[2]. This is primarily because of the inefficiency in executing naturally parallel computations. The major constructive problem is in fanning out the control signals to the many processors which may be distributed over a relatively large area but must still be synchronized. Holland machine [3],[4] consists of an array of cells each with some measure of local control(LC). The efficiency of such a structure on applied parallel operations is not so high. The primary problem with a LC structure with Holland machine is its spatial orientation resulting from the neighbor communication system[4]. This results in serious programing problems such as path building. Global communication is one of the major problem in computer design in the case where the parallelism is used for increasing its operation [5]. This organization has been considered an array of semiconductor wafer with a large part of each wafer devoted to memory and a small part of each wafer devoted to processing. By using the VLSI technology[9], we integrates processing and memory onto the same wafer. We have combined both the LC and GC into one structure, so that the communication occurs in two types of modes local and global combining both the LC and GC features into one structure.

Paper outline; In this paper, section-2 is devoted to the view of parallelism that we have motivated in our system architecture and the way of its application. Section-3 will show the definition of the Distributed Cells Processors(DCP), and its architecture is in section-4. While section-5 gives an analytical approach for the performance evaluation analysis for the organization; section-6 for conclusions.

2a- A View to the Parallelism

An example of the applied parallelism is shown roughly in Fig. 1, in which the number in Fig. 1a indicates the time that might be required

to compute each term on a conventional sequential computers (Seq.). Fig.1b, shows the use of applied parallelism, while the capability for taking advantage of natural parallelism is introduced then the computation may take the form shown in Fig. 1c. We can see the utilization of total parallelism (Tot) results in the reduction ratio in (Tot/Seq).

In an earlier papers[6],[7], we had shown an analytical approach for solving the assignment and sequencing problem; using Petri net. If the first restriction (equal length of time) can be reckoned with, it does offer a lower bound for any computational graph. It could be shown that there is no general solution to the optimum sequencing problem applicable to the study of parallelism.

The tasks are decomposed to the instruction level and then the graph links one instruction to the next instruction. If the computational facility is given the capability for the execution of applied parallelism can be known, the computational graph assumes a new form if computations are executed in parallel when possible.Solution of the computation on a conventional sequential connection results in a computational graph that consists of a time sequential connection of one task for another task. For the graph using applied parallelism, the tasks in parallel must consist of identical operations. The problem is to find the minimum computation time using the minimum degree of parallelism to achieve it. This requires assigning and sequencing the computations in an optimum manner[7].

3- The distributed-Cells processor (DCP)

The organization of this configuration is shown in Fig. 2, consists of cells. These cells are interconnected to form a group and a number of groups are interconnected to form the organization[8], which can also, be called functionally organized system since each processor is alloted to a well-defined function either permanently or dynamically. The allocation of cells dictated dynamically by sharing the load among the available cells. A cell may be considered to be a small conventional computer that has a relatively small memory (512-16 bits words of memory per cell). Parallel operations exist between cells in a group and between the groups in the system. Therefore the organization DCP appears to have a highly parallel computational facility.

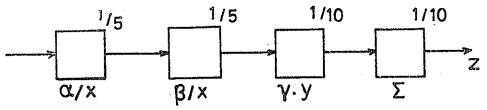


Fig. 1a, Sequential Computation.

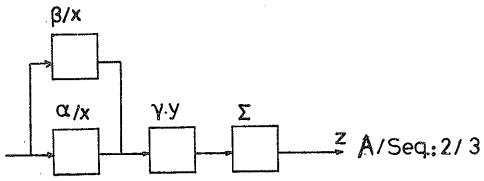


Fig. 1b, The Applied Parallelism used in Computation.

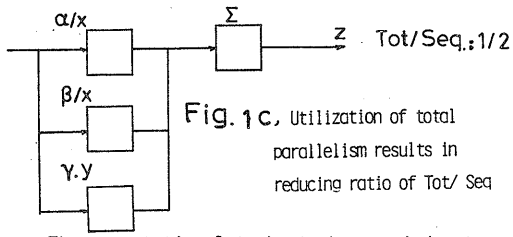


Fig. 1c, Utilization of total parallelism results in reducing ratio of Tot/ Seq

The computational tasks to be carried out are divided or assigned to various groups and finally to various cells in the groups. A group of cells may carry out a complete task, a number of small tasks. Some of the more important considerations in assigning tasks are minimizing communication between cells and groups, and utilizing applied and natural parallelism where possible.

TABLE 1; shows the instructions set predicaments. The hardware of cells in the entire system are all identical with each other. Each group has one cell called a controller cell(CC). While other cells designated working cells. CC is responsible for controlling the inter-cell communication bus and providing the executive control of the group's operation by a specified software routines; (TABLE 2). All of the cells in a group are interconnected to the inter-cell communication bus, that is a one-half word parallel . The one inter-cell bus is used for communication data between cells and for the transmission of global instructions and commands to other cells in the group. The cell that has the unique function of the CC controls the use of the intercell bus, and provides all the global instructions and commands. In a group the working cells either accept and execute the global

instructions (TABLE 3) that are sent on the intercell bus or fetch and execute instructions from their own memory. Cells operated in a GC mode are highly dependent on the CC since they receive instructions from it and are responsive to its global commands. The cell operated in a LC mode is considerably more independent of the CC, as it fetches its instructions from its own memory and must be talked to or commanded individually.

Fig. 3a, 2b shows a block diagram of a cell and the details of its structure. The cell has a logic for intercell bus communication and for its identification. All storage within a cell is directly addressable, and is divided into control registers and storage registers. Instructions whether they come from the cell's own memory or from the intercell bus, are decoded and executed in a manner similar to conventional computers.

Each cell contains a single serial connection to each of its four adjacent neighbors; with wrap around connections at the programmable edges of the array of cells. This communication means is primarily included to facilitate the global programming of a group, where small amounts of data are required to be communicated between cells, e.g., a matrix multiply utilizing applied parallelism.

One of the groups contains additional functions in its CC to operate as the system executive group. To coordinate and control the communications that take place on the intergroup bus. The system executive can be located in or more than one group(s).

There are factors determining the number of cells in a group and the number of groups to be used. From our parallelism studies, it was determined to use four groups of 20 cells each (512 words/cell) for our specified application as a commands controller to a multi-task systems.

4- Architecture

A cell always exists functionally in one of the states below:

1) permanently failed--- power off state;
2) independent state;
3a) dependent under GC, (global state);
3b) dependent under LC, (local state);
3c) dependent in wait state;
4) CC state;

Independent cells are functionally similar to a conventional computer. The cell in the independent state, stays in this state until the CC sends a command addressed to this cell on the inter-cell bus to change states. The independent cells (state-2) can process problems or subdivision of task without needs to global instructions from the other cells.

Dependent cells respond to global instructions and global level commands sent out from the CC (Table 2). A dependent cell exists in

Table 1;

Instruction Predicaments Set;

1) LR-- Load Register from a memory location
2) STR-- SToRe Register into a memory location
3) OPR-- an OPeRation is performed between a register and a memory location contents, the results store in the register.
4) RR-- an operation is performed between one Register and another Register
5) R-- single Register operation such as shift
6) EXEC-- EXECute an instruction in a memory location
7) COMP-- COMPare the contents of a memory location (or register) with a register. The results of the comparison are saved in the COMParison flip-flops.
8) SKIP-- Test the contents of a memory location (or register) with a register or implied value. The result is true or false.
9) JUMP-- a new sequence of instructions is begun. The JUMP may be combined with a test to make a conditional jump.
10) CC-- Controller Cell instructions
11) GC-- Global Control instructions. These instructions control the states and levels of all cells and dependent cell execution of global instructions.
12) IO-- Input/Output instructions. These instructions initiate and control I/O operations.

Table-2

Controller Cell instructions:

1) Control intercell bus I/O logic
2) Generate GC and bus I/O commands to be sent over the intercell bus
3a) Transmit mode, single
3b) Transmit mode, all
4a) Execute mode, single
4b) Execute mode, all
5) Do not change mode
   5.1-- no operation.
   5.2-- Format (F)
     5.2.1-- A --given Address.
     5.2.2-- D16 --16 bit Data word.
     5.2.3-- D32 --32 bit Data word.
     5.2.4-- A,D16 --given Address and 16 bit Data.
     5.2.5-- A,D32 --given address and 32 bit Data.
     5.2.6-- I --Immediate.
     5.2.7-- DS --given address and Data Stream.
   5.3-- State and level control (SIL)
     5.3.1-- Controller cell state ----set level.
     5.3.2-- Independent state --set level.
     5.3.3-- Independent state --no level change.
     5.3.4-- Dependent wait state --Set level.
     5.3.5-- Dependent wait state --no level change.

one of the states 3(a,b,c) depending upon the level of instructions being sent from the CC and the cell's level register contents. A dependent cell in the global state; 3a (also called active state) receives instructions from the CC. The level of the instructions and the cell's level register are the same in this state. A dependent

Table 3

Global control instructions GCI

1) Format (F)
- A --Given Address follows the next instruction.
- D16 --16 bit Data word follows the next instruction.
- D32 --32 bit Data word follows the next instruction.
- A,D16 --Both a 16-bit Data word and given address follow the next instruction, the address comes first.
- A,D32 --Same as A, D16 only the data word is 32 bits long.
- I --the displacement field of the instruction is the data.
- DS --a number of 16 bits words preceded by a given address follow the next instruction.
- END DS --Indicates the end of DS data words.

2) state control of dependent cells on the basis of levels (SL)
- Level, G --all dependent cells at this level go to global state; instructions follow.
- Level, L --all dependent cells at this level go to local control.
- Level, W --all dependent cells at this level go to wait state.
- Level, R --all dependent cells at this level reply on intercell bus with constant.
- Level, R, DG --all dependent global cells at this level reply on intercell bus with a constant.
- Level, IND --all dependent cells at this level go to the independent state

3) state and level control of individual cells.(SIL)
- IND, Level --the cell is made independent and level register set to the value specified.
- IND --the cell is made independent and with no change in the level register.
- DG, Level --the cell is set to the dependent global state and the level register set to the value specified.
- DG --the cell is set to the dependent global state with no change in the level register.
- DW, Level --the cell is set to the dependent wait state and the level register set to the value specified.
- DW --The cell is set to the dependent wait state with no change in level register.
- DL, Level --the cell is set to the dependent under local control state and the level register set to the value specified.
- DL --the cell is set to the dependent under local state with no change in the level register.
- CC the cell is made the controller cell.

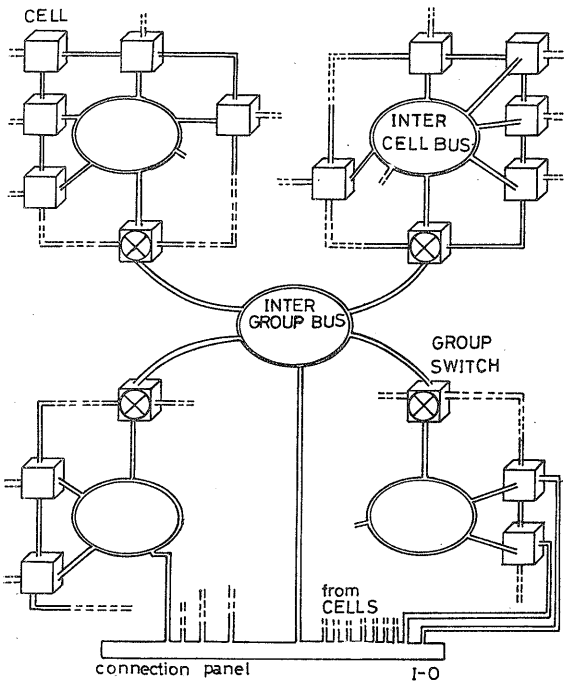


FIG. 2. Distributed Processor Organization

cell that; is not at the proper level to receive global instructions can idle and not execute instructions(wait state).

The capability of dependent cells to use local control means that; the cell bus is not wasted on sending instructions when the instructions could be better stored in the cell's own memory. So that, the cells can efficiently use local programs to correct for bad data, handle exceptional conditions, and later inform the CC of the situation. Even though the cell may use local control in state 3b, it is still a dependent cell. It is basically under control of the CC and responds to certain global commands.

The Fourth state of a cell is the CC state. In this state; a cell controls the intercell bus and issues global instructions. The CC functions may be switched among several cells. Thus there is no requirement that all the executive and controller programs fit in one cell.

4-1 Cell Identification

We have designed two methods for identifying the individual cells. One is by levels and the other is by giving each cell a unique identifier known as a cell address. Thus a cell has a common first name(level) and a unique last name

(identifier). This concept 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, though present in a level register. Dependent cells use both names. The CC can send out information using a first name(level number) to all the dependent cells. If a last name (cell address) is sent, only the cell with this name responds since each cell has a unique last name. However, the CC may communicate with dependent cell responded to levels.

4-2 Instructions Sources

Our system's instructions depend on the state of the cell(s). The independent cells receive all of its instructions from the cell's memory, under the control of the program counter.

In the global state, the dependent cell gets the instructions from the CC, through the inter cell bus. The CC precedes the instructions with a name The name (level number) is contained in a control word (CW) sent on the intercell bus. This CW is a 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. A group of global instructions which is variable in length, can be defined as the information contained between control words on the intercell bus. Every dependent cell compares the level prefix sent by the CC to the its level register contents, if it are different, the cell ignores all the information sent by the CC, until a new level prefix (or other control word CW) is placed on the bus by it.

The following example can show the source of the instructions and its operations for the state of dependent global cell. The example is given in Fig. 5. Note that every cell is required to examine every CW but it will ignore it if the cell is at a different level. When segment 1 in the example occurs, all the cells in the group examine the control byte(CB). Assume the CB is a type that specifies a level. All the dependent cells at this level are ready to receive the CW (segment 2) and will placed in the dependent active(global) state. These cells in the global state receive the CW (segment 2) instruction and/or data following (segment 3). No other cells receive any instructions or data (segment 3) from the bus until the

Segment 1,4, and 6 represent control byte. Slanted lines(2) represent control word, segments 3,5, and 7represent data.

FIG. 5 BUS OPERATION EXAMPLE

next CB occurs (segment 4 in the example). When segment 4 comes on the bus, all the cells again examine the CB. If we assumed that; the CB specifies a different levels, the following operation can be happened;

--Dependent cells that were active (global state) are set to the dependent wait state by the CB at a new level.

--Dependent cells not active and at the new level indicated in the new CB (segment 4); become active and receive the data (instructions) following (segment 4). Other cells are left unchanged.

In any case; many sequences of global instructions may be sent to many sets of cells at very low overhead cost to switch between sets. The low overhead is advantageous when many cells are at each level and short sequences of instructions are to be transmitted to each.

4.3 Sources of Addresses

The cells in the DPC have several ways of specifying an address. One way is by adding an instruction displacement and a bank register (also known as a base register). The bank register is 9 bits long since a cell will have 512 words of storage. The bank and index register used to form a calculated address are always located in the cell itself. Independent cells obtain the displacement from the instruction that was sent on the intercell bus.

In addition to the calculated address, a new concept of a given address is used, which is an address used instead of the calculated address. It is specified by a special instruction preceding a regular instruction. A dependent global cell recognizes a given address by a special control instruction received on the intercell bus, called a global control (GC) instruction. The GC instruction is sent from the CC to signal the dependent global cells that a given address, in addition to an instruction, is to be sent on the intercell bus. This particular GC instruction is called a format-given address. The sequence on the bus, which is changed by the format instruction, is as shown in the below tabulation.

TIME	CONTENTS OF INTERCELL BUS
↓	GC Format (Given address)
	Instruction
	Given address
	Subsequent instruction

The dependent global cell executes the instruction using the given address instead of the calculated address. Thus the CC send an address to all the dependent global cells instead of having the cells calculate the address. The independent cell (and the dependent cell under local control) may also use a format instruction, but here the format instruction is located in the cell's own memory. After the format construction is executed, the cell knows the type of data contained in its following memory locations. an example is given below.

Location	Contents
START	Format instruction (address given)
+1	Instruction
+2	Given address
	Subsequent instruction

Here, the instruction at (START + 1) is executed using the given address; instead of a calculated address. The use of the format instruction is called instruction modification. The modification is usually not a change in the operation of the instruction but rather a respecification of the address.

4-4 Source of Data

Cells receive data from outside the group via the cell's I/O line. Cells may also receive data from outside the group or from other cells in the group via the intercell bus.

Another means know as "Data Follows" is introduced here(Table-3). This is quite similar to the "given address" concept presented above. In fact, data follows is specified in the same way. Dependent global cells recognize this by a special GC format instruction received over the inter-cell bus, the other cells recognize it by the execution of a special format instruction in the same way described above. The use of the data follows concept is interpreted as meaning the word following the next instruction is to be used as the data with which to execute the instruction. The address displacement from the instruction is not used in this case. The data concept is particularly useful in dependent global cells. Having data sent by the CC means individual cells do not have to store constants.

There are several other types of format instructions used, one of which is "Immediate" instruction(Table-3). This format instruction specifies that the address displacement from the next instruction is used as the data to execute the instruction. It was designed to rapidly move data from the CC to a number of cells or a cell. The format instructions introduced here are called, Modifiers, as they are used to modify the address or data specification (or both) of an instruction.

4-5 Instruction Execution

Some of the functions of the CC instructions are; 1) Control the intercell bus communication section of its cell, 2) generate the GC instructions, 3) generate intercell bus communication control signals, 4) control the transmission/execution of instructions in the CC, 5) specify a format modification, and 6) control the state or level identification of the cell.

Some of the function of the GC instructions are; 1)specify a format modification, 2)control the state of dependent cells on the basis of level identification, and 3) control the state and/or level of individual cells upon the basis of address identification.

Below Tabulation shows the comparison between GC and CC instructions execution.

Global control--CC instruction execution

State	Fetch from own memory (CC inst)[1]	Received from bus (GC inst)[1]	Sent over bus GC(inst)[1]
CC	All[2]	X	F,SIL,SIL,C
INP	F,SIL[3]	SIL,C	X
DL	F,SIL[4]	SIL,SL,C	X
DG	X	F,SIL,SL,C	X
DW	X	SIL,SL,C	X

[1] Where X stands for not applicable and C for communication bus I/O commands.
[2] Except 5.3.2 through 5.3.5 (Table 2).
[3] For independent cells only a level control is allowed from its own memory (5.3.2 table 2).
[4] For dependent local cells only state control to the dependent wait or independent state is allowed from its own memory, 5.3.2 to 5.3.5 (table 2).

The difference in the instructions executions are indicated by the GC format modifier byte preceding the instruction.

5-1 Design Description of the Cell

We designed our cells using the VLSI

Fig 3b

Explanation of the block diagram of the cell;

L :
Lower accumulator is used primarily in multiply, divide and double precision operations to hold the lower half of data word. This accumulator and U have a 1-bit extension onto their 16 bits in order to hold the overflow carries which may be generated in the multiply operation.

U :
Hardware Upper accumulator is used to hold one of the memory up.

U_1, U_2, U_3, U_4 : Memory Upper accumulators.

P : Program counter

MAR :
Memory Address Register grasps the memory address for operand memory cycles. It is loaded with the address displacement from the instruction words; B_1 or B_2 is added to it and, if indicated, one of the index (T) registers is also added to it. This register is necessary since the B and T registers are in memory and enter the processor through MB. As a result, MB cannot be used to hold the operand addresses.

B_1, B_2 : B memory index-Bank registers hold both index and bank values for address calculation and looping control. One of these two registers , indicated by the B bit, is added to the address calculations. (this is necessary since the displacement from the instructions is only 7 bits long and 9 bits are needed to address memory.)

T_1 to T_3 : M memory index registers have the same functions as the B registers. However, operand addresses can be generated without adding any T register to B plus the address displacement.

ALTU :
Adder, Logical and Transfer Unit operations carrys out arithmetic and logical operations, including comparisons. It provides for transfers among all the hardware registers and detection of overflows.

IR :
Instruction Register holds the 6-bit op code throughout the instruction execution.

TR :
Tag Register holds the T bits of the instructions. It is necessary so that B plus the address displacement can be generated, stored in MAR, and then added to T prior to an operand cycle.

SCR:
Shift Control Register holds the shift count for shift commands, and for setting up bits in byte manipulation operations. It is counted down to zero by one count for each shift. It provides a temporary storage area for certain control information during several intercell bus communication operations.

LENR:
LENgth Register is used for byte instructions to specify the length of the byte that is being used.

HA :
Hardware Accumulator identifies which of the four accumulators is presently in hardware. Every instruction that specifies an accumulator must compare these 2 bits to that in the instruction.

CFF:
Comparison Flip-Flops hold the results of a comparison; greater than, less than, or equal. They may be tested by an instruction for a conditional jump, skip, etc.

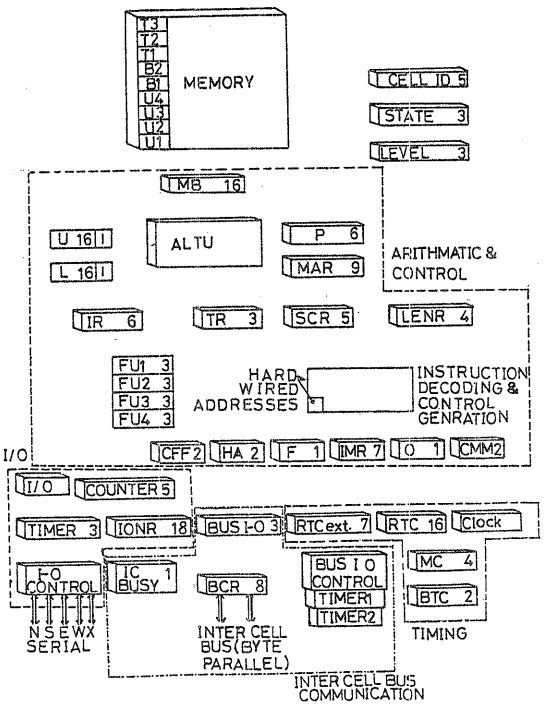


FIG. 3a, Block Digram of the Cell

continue Fig. 3b

BCR:
Bus communication register receives the present byte on the intercell bus. It is used by the bus I/O control to decode commands and control the bus communications.

I-O :
This flip flop is set to either the input or output state during an I-O instruction execution. It controls the type of interrupt generated by the counter BCR.

$FU_{1,2,3}$: Accumulator flags are set to one of five states; N,E,S,W, or X. The first four indicate the accumulator holds data for a a neighboring cell and the last one indicates no neighbor data. There is one flag register for each accumulator.

RTC ext:
Real Time Clock Extension is a simple counter driven by the clock.

RTC:
Real Time Clock is a simple counter driven by the RTC ext. It is set and read by instruction and counts down to zero.

BTC :
Bit Time Counter is driven by the clock and generates two pulses (count to two). It is used to control the instruction execution and drive instruction decoding and control generation logic.

MC :
Model counter is driven by the BTC and generates eight control signals.

BUS I-O :
Holds the identification of the present operation being carried out on the intercell bus (input, output, etc.).

IC BUSY :
InterCell Bus Busy. This flip flop is used to determine if the bus is presently being used.

O:

Continue Fig. 3b

Overflow flip-flop is set if an overflow results in the ALTU during an arithmetic operation. Like the CFF, it may be tested by an instruction.

F:

Failure is set by the hardware failure detection circuitry. It can be sampled by an instruction during software self-test.

IMR :

Interrupt Mask Register is set by the programmer to mask off any interrupts that are not to be allowed.

CCM:

Controller Cell Mode holds the mode of a cell in the controller cell state. It has no meaning in cells in any other state. It will be set to one of the four transmit/execute modes and will control the execution of instruction in the controller cell.

Cell ID:

Holds the Cell address or IDentification. It is used in the bus I/O control section to decode commands over the bus.

STATE:

This register holds the current state of a cell. It will be in one of 5 states;

000	independent
001	dependent global
010	dependent local
011	dependent wait
100	controller cell.

LEVEL:

specifies one of eight levels that the cell may use as a means of identification during dependent state operation.

Timer 1,2 :

Used to time certain operations over the intercell bus. Expiration of the timers during these operations results in an interrupt being issued.

IONR :

I-O and Neighbor Register. This is a serial/parallel in and out buffer register used for communication between I/O devices and also one of the four neighbor cells. It is 18 bits long because of the need for synchronous and control bits in addition to the data bits(16 bits)..

TIMER 3:

Is set during certain I/O operations such as requesting data from a neighboring cell. Its expiration will cause an interrupt.

Counter :

Counts the serial in/out of the IONR. It will issue interrupts during I/O operations.

technology. The block diagram of the cell is shown in Fig.3. Also the description functions for the cell's blocks were given in Fig. 3a. The cell's hardware consists of six general areas: Memory; Identification; Arithmetic control; Intercell bus communication; I/O; and Timing. (Logical gating and some control flip-flop have been omitted from the diagram for clarity.)

5-2- Intercell Communication Bus Structure

A total of 10 lines are used for the intercell bus(see Fig. 6), one is the control line to denote control or data, the other is used for parity, the remaining eight lines(half word) are used for control or data. The bus is a half-word

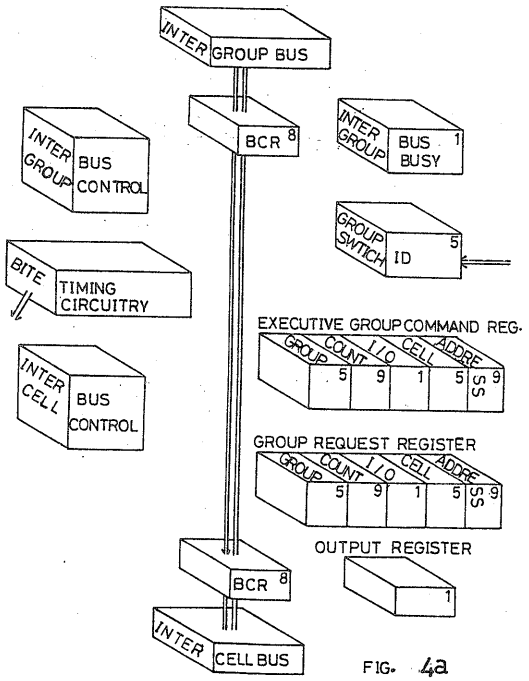


FIG. 4a
Block Digram of Group Switch

Fig.4b

Explanation of the Block diagram of the group switch

BCR :

Buffer communication Register-- Two half word long registers that receive the present byte over the bus they are connected to.

Inter Group Busy:

Inter group Bus busy flip-flop. This F-F is set when the executive group command register is loaded in a group it is reset when an "End of Transmission" command is send over the intergroup bus. It is sampled by command over the inter cell bus.

Group Switch:

A 5-bit register to hold the identification or address of the group switch. This register is used to decide which commands are addressed to the group switch. It has a serial load line for initialization purpose. Note that this provides the capability for having up to 32 group switches or 16 groups(two group switches used for group) in the system.

Group Request Register:

Holds a request received over the intercell bus by a command addressed to the group switch to load this register. It is sampled by command over the inter group bus and holds the requested I-O over the intergroup bus.

Executive Group Command Register:

Holds a command received over the intergroup bus that was addressed to this group switch. It is sampled by command for this group identifying the all addresses, and number of words to be input or output over the intergroup bus. This register is automatically reset when it is sampled over the intercell bus.

Output Register: continue Fig. 4b
Set by a command received over the intergroup bus. It is sampled by a command over the intercell bus.
Bite timing circuitry:
This circuitry consists of a flip-flop that is alternately set/reset by a command from the intercell bus. The rate at which it is set determines the voltage generated across a circuit, two out of tolerance level detectors across this circuit, will check for a high and low value of voltage, for failure signal detection.
Bus Control for intercell & intergroup communications :
Contain all the control circuitry to decode information received over the buses and execute commands addressed to the group switch. This control circuitry generates inter group bus commands from certain intercell bus commands. It is responsible for opening up the communication path between two buses.

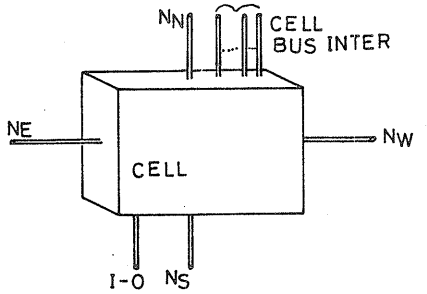


Fig. 6, Communication Lines For the Cell

(8-bits) parallel. One extra line is used and designated a control line, used for defining the control words or commands on the bus. The commands or control words utilize levels or addresses (first or last names) to identify which cell or cells; intended for that command or control word. The I/O instruction uses a long instruction format (a 32-bit instruction) as the below tabulation;

6	1	9	7	9
op code	I/O	Address	Device	word count

The I/O cells are to curry out their communication with the conditioners over a single line connected between the I/O cell and all conditioners that are to communicate with the cell.
The communication functions to be carried out over the inter group bus are:-
1) Executive group(EG)to scan groups for requests.
2) EG to request to send/receive words to/from another group directly.
3) EG to command another group.
The EG that is in charge of controlling the use of the inter group bus (in the group request register) dose not have the groups(GS) under its immediate control.

5- Performance Evaluation;

In this section we have motivated an analytical strategy for predicating the performance for our system on iterative algorithms, where each iteration consists of some amounts of access to global data(instructions)and some amount of local processing. We have used synchronous and asynchronous application cycle. The amount of processing time and global data accesses incurred by the parallel processes depends upon characteristics of the algorithm and its decomposition.

In an earlier paper[7] we presented a model using Petri net with which we have modeled the system, and combined the net which have hardware constraints with the net represent the sequencing of instruction execution and evaluation of the merging net we can reach to the optimal design for the system hardware. Here, we have used the fact that most algorithm are iterative in nature, and the iterations tend to perform approximtely the same amount of processing and data access.

Let us assume that a given algorithm in the CC consumes t_p units of processing time units for accessing global data to or from other cells. The ratio of the above two times depends upon that algorithm. The Characteristics of an algorithm and its decomposition depends on the amount of processing time and the global data access that are incurred by the parallel process. As most algorithms are iterative in nature, moreover; the iterations perform nearly the same amount of processing and data access.

In our analytical approach; we have decomposed an algorithm into a set of processes that have high processor utilization and low speedup. We wanted to minimize execution time with such decomposition. We have assumed if an algorithm needs multiple access to a shared variables. We have to select on of two conditions; either or not each process/(cell) should have a local copy. We think it is rather better presents the both approaches for better performance analysis.

When local copies are substituted for global data, processing time per process/cell increases because; first, it takes processing time to copy the data; and second; the time for accessing the data/cell, (for subprocess it will be changed to, the processing time/cell.) The second cost has

been neglected under our assumption; as well as; processing access periods are interleaved within an iteration.

For simplicity in understanding, we will give below the definations and functions which we have used in our approach.

τ_p= Time units that a given algorithm consumes for each iteration in a group.

τ_a= Time for accessing a global data/group of cells. t_p = processing time /cell. t_a=access time /cell. ($Ra=\tau_p/\tau_a$) , represents the ratio.

Then; the decomposition function ($Df_p=\tau_p/t_p$; $Df_a=\tau_a/t_a$), which consists of two important parameters Df_a; Df_p.

$\tau_c = \tau_p + \tau_a$; represents cycle time for single iteration/group; but as well as there is waiting time t_w for a cell to get access for a global data from the CC; then the cycle time for a cell $t_c=t_p+t_a+t_w$.

Now let us define the speedup (υ) in terms of the above parameters; $\upsilon(n)=\tau_c/t_c(n)$; and also a function of number of cells in a group. The speedup with respect to (w.r.t) the average processor utilization(§); is; $\upsilon= §(\tau_p+\tau_a)/(t_p+t_a)$; Usually processing and access periods are interleaved within iteration for one group. We will search the case when decomposition function' s parameters are equal; $Df_a=Df_p=n$; i.e., decomposed to number of cells; then we have a basic solution which is a means for a measurement and comparison with other decomposition approaches.

In case synchronization is required after each processing and accessing[10]; the worst case performance is;

$$\upsilon = Df_a\ Df_b(1+Ra)/(nDf_p+RaDf_a) \qquad \text{-----(1)}$$

If in case no synchronization is required, the best performance can be obtained;

$$\upsilon=\min[Df_aDf_p(1+Ra)/(Df_a+Df_p),Df_a(1+Ra)/n], \quad \text{----(2)}$$

As we stated above that, the basic solution is used for comparison; which the case $Df_a=Df_p$, then the speedup which is the normalized speedup(υ_{normal}) is; $\upsilon_{normal}= \upsilon\tau_{c\text{-}basis}/\tau_c$. ---(3)

Example;

In order to have an overview outline for the analysis, and without loss in generality, we have used here an example of matrix multiplication algorithm and apply the methodology to the objective system.

Suppose our matrix is of dimension AxA, so that; the matrices;|X|=|Y|×|Z|; can be decomposed into n processes(cells)[11], each cell has a submatrix of dimension $A/\sqrt{n}xA/\sqrt{n}$. Suppose; $a_1= A/\sqrt{n}$, and; Π_{rs}; is the s-th submatrix with r-th row; then; $(\Pi rs)_{ij} = {}^{1)}\ \Pi_{[(r-1)a1+i,(s-1)a1+j]}$; assume $a_2=\sqrt{n}$; and the ζ , γ; present the global variables which are referencing to the submatrices of |Y| and |Z|, respectively, that should be shared by the working cell through the CC; then; the statement (3); is,

$$I_{i=1,a1}I_{j=1,a1}\{\Pi rs)ij \overset{2)}{\leftarrow} I_{q=1,a2}\Sigma(\zeta rq)ik\times(\gamma qs)kj\} \quad \text{--3}$$

The global variables access in the above statement will be replaced with local ones; so that we will have the statement (4) as follows;

$$I_{q=1,a2}I_{i=1,a1}I_{j=1,a1}\{(y_{rq})_{ij} \overset{2)}{\leftarrow} (Y_{rq})_{ij};(z_{qr})_{ji} \overset{2)}{\leftarrow} (Z_{qr})_{ji}. \quad \text{-----4}$$

Suppose; τ_1=processing time, for one iteration without local copies(i.e., time for one iteration in statement (3); also τ_2=time for one global access, (exclusive of waiting time); and τ_3=time for copying an element of a single global matrix to or from the controller cell.

Then the processing and access times(t_p & t_a);for an individual process(cells); are obtained by multiplying the number of iterations /cells by the time needed for processing and accesses, with an iteration/group; so that; in case without local copying; $t_p=(a_1\ a_1\ a_2\ a_1)\tau_1= A^3\ \tau_1\ /\ n$; then; $\tau_p= A^3\ \tau_1$,

$t_a=(a_1\ a_1\ a_2\ a_1)\ 2\tau_2 = 2A^3\ \tau_2\ /\ n$; $\tau_a = 2A^3\ \tau_2$, $Ra= \tau_1\ /\ 2\ \tau_2$, then $Df_a= n$, $Df_p= n$. ▲

Now; let us search with local copies so that;

$t_p = A^3\tau_1\ /n + A^2\tau_3\ /\sqrt{n}$; $\tau_p= A^3\ \tau_1+ 2\ A^2\ \tau_3$.

$t_a = (\ a_1\ a_1\ \sqrt{n})\ 2\ \tau_2 = 2A^2\ \tau_2/\sqrt{n}$; then; $\tau_a=2A^2\tau_2$.

Then the ratio; $Ra = (A\tau_1 + \tau_3)/\ 2\tau_2$; so that;

$Df_a =\sqrt{n}$; $Df_p = (A\tau_1 + \tau_3)n/\ (A\tau_1 + \tau_3\ \sqrt{n})$.

We can see that; the change in global data allocation and its management made change in the Ra; (the processing to access ratio), and Df_a & Df_b; (the decomposition function).

General investigation:

Here we shall see the tradeoff due to local and global data, because in parallel processing we often use the same global data items more than a time during an iteration; to have better alternative design we make a local copy of the

---

1) $I_{i=j,k}$; represents; the iterations for i from j to k by 1.

2) "$\leftarrow$" ; represents assignment.

data so that the first reference to each item needs to access a common resource. As well we have suggested such a decision it better to study its tradeoff.

Theorem; If the decomposition function is of the form $Df_a = n^i$, $Df_p = n^j$, then the lower bound speedup(υ) will have a maximum point iff not;

i=1 or j=i+1. --------(5)

PROOF; We shall suppose the contrary to the theorem so that; there is a decomposition function with which υ has no maximum point; so that take the derivative of equation-1; w.r.t. n, so that; $(\partial\upsilon/\partial n)=0$; and set its numerator=0, for maximum. so that; $n^{j-i+1}(i-1)+ j\,Ra= 0$. This equation has a solution iff not i=1 or i=j+1. So (υ) has no maximum at these points. ▲

An example of the application of the above theorem is; the decomposition function ($Df_p=n^j$, $Df_a=n$); which has no maximum speedup.

We have selected the decomposition function as $Df_p = Df_a=n$, because of its better characteristics; (as we have seen in the example).

If in case we have maximum (υ) for a number of cells which were assigned according to the decomposition function(n,n), which is our concerning, then n_{ct} is the critical point in which we shall have a degrade in performance due to the increasing number of processor beyond n. Actually the general form of (n_{ct}) is difficult to derive, but let apply it to the example which we have introduced it before.

From the matrix example; let the subscript (wtho) points to the implementation without local copies; and the subscript (wth) points to the copying one. We had from the matrix example the following derived equations; $Ra_{wtho} = \tau_1 /2\tau_2$;

$\tau_{p\text{-}wth} = (A\,\tau_1 + \tau_3)\,A^2 \simeq A^3\,\tau_1$; (assuming the time to iterate through a row is much greater than the cell time to copy one data element,i.e., ($A\,\tau_{wtho} >> \tau_3$). $Ra_{wth}=(A\tau_1+\tau_3)/\,2\tau_2 \simeq A\,Ra_{wtho}$.

As well, we want to find the cells number where the performance of an (n,n) decomposition equals that the local copy implementation. We have to find n_{ct}; where the performance of (n,n) decomposition equals that of local copy implementation; by setting; $\upsilon_{wtho}=\upsilon_{wth(normal)}$.

From equation (1); we can have;

$$\upsilon_{wth}(normal)=(Df_{a\text{-}wth}\,Df_{p\text{-}wth}(1+Ra_{wth})\tau_{c\text{-}basis})/(Df_{p\text{-}wth}n+Ra_{wth}Df_{a\text{-}wth})\tau_c \quad \text{------(6)}$$

$$\upsilon_{wtho}=(1+Ra_{wtho})A^2/\,(A^2+Ra_{wtho}\,A) \quad \text{---------(7)}$$

For local copies version of the matrix example equation(6) becomes;

$$\upsilon_{wth\text{-}normal}=(Df_{a\text{-}wth}Df_{p\text{-}wth}A(1+Ra_{wtho}))/(Ra_{wth}Df_{a\text{-}wth}+nDf_{p\text{-}wth})$$
$$=\Phi nA(1+Ra_{wtho})/((n^{3/2})\Phi+Ra_{wth}\sqrt{n}+A\Phi Ra_{wtho} \quad \text{--(8)}$$

By setting equation(7) and (8) equals; we can have; $n_{ct}+Ra_{wtho}/\Phi-A\sqrt{n}_{ct}=0$. $(\Phi=\tau_1/\tau_3)$.

Assuming $\tau_3<<\tau_2$; we can have $n_{ct}\simeq A^2$.

If we assume τ_3 is insignificant w.r.t. τ_2, it becomes profitable to make local copies as long as the number of processors is at least; less than the number of matrix elements, if τ_3 increases w.r.t. τ_2, n_{ct}; will decrease.

If we set; $(\partial\upsilon_{normal}\,/\partial n)=0$; so that for the decomposition like the local copies version of matrix example; the maximum speedup can be achieved with a less number of processors.

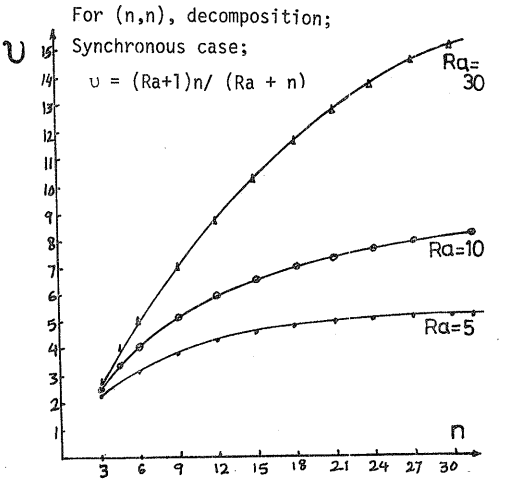
$(\partial\upsilon/\partial n)=0$; so setting the numerator=0; $A(1+Ra_{wtho})\Phi[A\Phi Ra_{wtho}+Ra_{wtho}\sqrt{n}-n\Phi(3/2)\sqrt{n}-(nRa_{wtho})/(2\sqrt{n})+n\sqrt{n}\Phi]=0$.

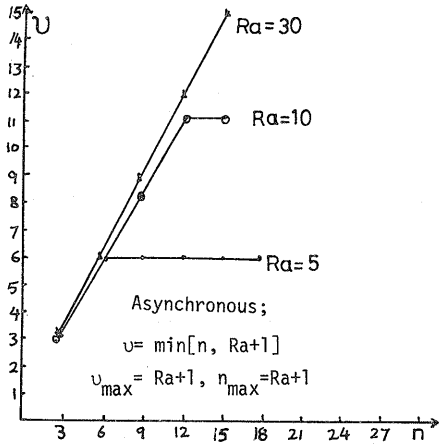
The time to iterate through a row is much greater than the cell time to copy one data element.

$Df_{p\text{-}wth} \simeq A\Phi n/(A\Phi+\sqrt{n})$; (assuming; $A^2>>Ra_{wtho}$.

$n=(2A\tau_{a\text{-}wtho})^{2/3}$.

We could see that the speedup function depends on the way which the algorithm is decomposed. The pre-process times vary as the processors number is increased due to the defined decomposition function for the processing and accessing time for the cells. So algorithms can be divided into decomposition groups based on these functions; as shown in the curves.

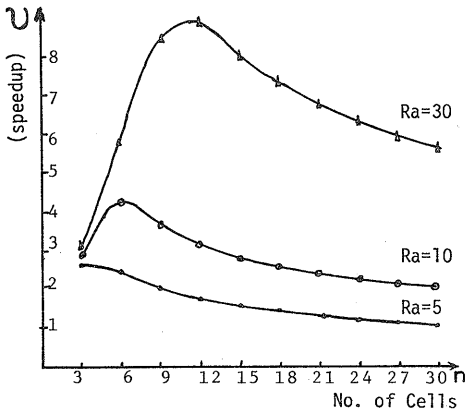




For $(n,\sqrt{n})$; decomposition;
Synchronous;

$\upsilon = (Ra+1)n/(Ra+n^{3/2})$;

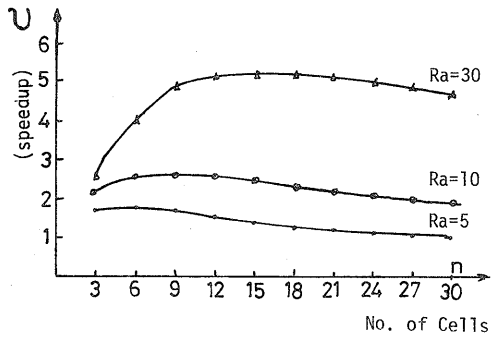
$\upsilon_{max} = 2^{2/3}(1+Ra)/3Ra^{1/3}$; $n_{max} = (2Ra)^{2/3}$.



Asynchronous;

$\upsilon = \min[(1+Ra)n/(Ra+\sqrt{n}), (Ra+1)/\sqrt{n}]$;

$\upsilon_{max} = (Ra+1)Ra^{1/3}/(1+Ra^{2/3})$, $n_{max} = Ra^{2/3}$.



Conclusions

We have presented in this paper the architecture in which we introduce the parallelism in more effective way, by using the applied and natural prallelism. These cells have many levels of instructions, and communicate with each others by format modification instructions. The local and global communications have been combined by global and local commands interchanged through local and global buses. The performance evaluation have been motivated using analytical approach, which can show the effect of using many cells for decomposition of an algorithm or tasks through the cells to achieve the speedup in system operations. We have defined decomposition functions for the processing and access times, which tell how the pre-process times change as the number of cells is increased. Algorithms can be divided into decomposition groups based on these functions. The speed up curves show the effect of such a design in the system for synchronous and asynchronous tasking.
We also had evaluated and modeled the system using Petri net and Timed Petri net.

References

[1] Nakamura, T., "Software of the brain structured computer," Proceedings of COMPSAC'84, Chicago, Illinois, pp.408-414 Nov. 1984.

[2] Hockney R.W., and Jesshope C.R., "Parallel Computers," Published by Adam Hilger Ltd, 1981

[3] Holland J.H., "A Universal Computer Capable of Executing an Arbitrary number of subprograms Simultaneously," Proc. East. Joint Comput. Conf. 16, pp. 108-113, (1959)

[4] Bustoes E., Lavers J.D. and Smith K.C., "A parallel array of microprocessor-an alternative solution to diffusion problems," COMCON'79 (Fall) Digest, pp.380-390, 1979

[5] Dahl O.J. Dijkstra E.W, and Hoare C.A.R., "Structured Programming," (New York: Academic), 1972.

[6] Hamid I.A., " An overview to distributed computer system models, with a concentration on classical graph theory," Info. Proc. Soc. of Japan, (IPS) Conf. Proc. No. 29, 1984.

[7] Hamid I.A., "Optimal Solution for sequencing decision for parallel machines," Info. Proce. Soc. of Japan. (IPS) Conf. Proc. No. 30, 1985.

[8] Hamid, I.A., "Distributed Configuration for commanding executing systems " Info. Proc. Soc. of Japan. Conf. Proc. No. 31, 1985.

[9] Kunii T. L., "VLSI engineering beyond software engineering," Lecture Notes in Computer Science 163, Springer-Verlag, Tokyo, 1984

[10] Versolovic, D. Segall,.Z, "Performance Prediction for multiprocessor systems," Proc. 13th Int. Conf. on Parallel Processing, 1984.

[11] Whiteside R., " A case study in the application of a tightly coupled multiprocessor to scientific computations," in Parallel Compuatations, Rodrigue, G., 1982, pp. 315-364.