'88 韓國自動制御學術會議論文集 1988. 10. 21~22

다중 프로세서 방식의 프로그램형 제어기의 구조와 시스템 버스

°김 종일, 권 욱현

서울 대학교 제어계측 공학과

The Architecture of A Multiprocessor Based Programmable Controller With Emphasis On its System Bus

°Jong-il Kim and Wook Hyun Kwon

Dept. of Control and Instrumentation Engr.

Seoul National University

ABSTRACT The architecture of a multiprocessor based programmable controller(MBPC) is presented. It consists of a host processor, processing elements, and Input/Output processors. Some problems in implementing such architecture are also described. To resolve them, we proposed and presented INFOBUS, a system bus for MBPC. The performances of INFOBUS and MBPC are analysed using both analytic models and simulations. Some results from the analysis will be given and validated. In case of 50% of BTI(Block Type Instruction) and 4 processors, the scanning time is shown to be 0.194msec/Kstep with some reasonable assumptions.

1. Introduction

Parallel processing have received wide attention to overcome the limitation of computing systems, especially in processing speed. But we should keep in mind the sequential nature of most conventional computing systems. It imposes certain problems on how tasks are executed, on how the system is constructed, and on how information can be exchanged.

Moreover, the most important factor limiting computational speed is not availability of appropriate hardware, but rather the difficulty of managing the complexity of algorithm formulation and programming and of finding parallelism[1].

In real time applications, such as process control and sequence control, however, the process itself is highly parallel[2]. The concepts of distributed control or the decentralized control are being researched and implemented in many process control systems[3][4].

In sequence control applications, the development of new and more sophisticated programmable controllers(PC's) will have a major and significant impact on that field. But the movement of PC's into larger and distributed systems, has brought new challenges, and requires new architecture, such as a special hardware structure and parallel processing.

Fortunately, the PC has some parallelism in nature. Updating input and output(I/O), solving user supplied application program, and communicating with external worlds are the examples of concurrently executable tasks. Such parallelisms has been exploited and already adopted in many available PC's. TI 560 /565 system is a typical example[19], which solves sequence programs concurrently with the updating of I/O. With this approach, we can reduce the time required to update I/O.

The time required to solve application program, however, is not so simple. TI 560/565 can be configured to have two processors for solving the application program, but it does not utilize complete multiprocessing. It only separates the discrete part and the analog part of an application program and executes them concurrently. There may be another approach to do this. A special-purpose processor was adopted in Modicon 584 PC system[20]. It uses a bit-slice architecture with microprogramming method, mainly to reduce the time to solve an application program and to meet the bit data structure of the sequence programs used in PC. But there are no parallel structures.

There are some restrictions on implementing MBPC. First of all, we should overcome the difference in the data structure of PC and general purpose microprocessors. Such incompatibility has brought another problem; improper structure on implementing with microprocessors. And, since PC's have to handle large amount of data, the PC's reveal other basic problems in multiprocessor environment; distributing tasks, exchanging information, and scheduling.

For these reasons, MBPC will require special structures. In this paper, the special hardware, which can be a candidate for the structure, will be described briefly[21]. Such hardware was designed and now is available in our laboratory. Using it, it is expected that we can overcome the incompatibility problem mentioned above. In addition, INFOBUS, a specially designed system bus, will be described too. It will be shown to be well suited to MBPC by simulating with exact data taken from existing hardware.

There are many parameters to evaluate the performance of PC's. The processing speed, the capacity, and the reliability are the most significant indexes. In this paper, however, we will emphasis on the scan time of PC and the performance of INFOBUS.

In Section 2, we will describe some restrictions of PC on implementing multiprocessor architecture. In Section 3, we will describe INFOBUS, which was designed as a dedicated system bus for the multiprocessor based PC. And the mean data transfer time through INFOBUS will be analyzed and simulated based on the exact data taken from the existing hardware. In Section 4, MBPC will be introduced, and some of its basic characteristics will be described. In Section 5, the model of MBPC will be constructed based on some reasonable assumptions, and simulated using

SDL/SIM package[17][18]. Some simulation results will be given and validated.

2. Some Restrictions on implementing MBPC

The PC was originally developed as a sequential control device to replace electromechanical relays in the factory. From the early stage of PC, it was used by the factory engineers, who design, operate, and maintain the control systems. Since ladder diagram is a method of representing a system of relays, switches, solenoids, lamps, etc. and it is easily understood by the factory engineers, it has been chosen as the programming language. By now, the ladder diagram has been enhanced by the PC manufactures to add program flexibility and sophistication of a general minicomputer[5]. Thus it is assumed that the programming language of our MBPC is the ladder diagram.

Since each data processed by PC corresponds to the relay, switch, lamp, etc., it can be represented by two states; OFF and ON or "0" and "1". In other words, the data operands of PC are bit addressable. Since, however, most of microprocessors have byte or word type data structures, it is not easy to construct an efficient PC system by using microprocessors without special structure. We will call this kind of problem as data type incompatibility problem throughout this paper.

Suppose that we transformed the bit data structure of PC into byte or word type of most microprocessors, then the amount of data needed to hold the state of input and output is unnecessarilly large. For example, in case of byte type, it requires 8 times than needed in bit type. In addition, it is expected that the time needed to exchange the results among the processors in MBPC also requires the same amount.

Next problem arises from the sharing of the ladder diagram among the processors. The output points in a ladder program are distributed randomly. For example, "storing into Y0000" locates at the 1234-th instruction, while "storing into Y0001" locates at the 7-th instruction, etc. Thus if we assume that a PC system can process bit data structure directly, then each processor probably may hold the data arranged in the form shown below;

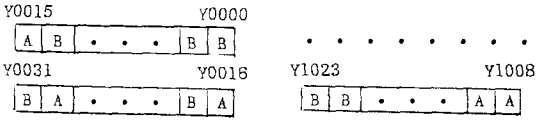


Fig 1. The output data arranged in PC

The ladder diagram has been divided and distributed to the two processors, A and B. The final results of the ladder diagram allocated to the processor A are Y0015, Y0016, Y0032, etc., so processor A may change them. Likewise, the processor B may change Y0000, Y0001, Y0014, etc. And in the course of solving, each processor may require the output data not a part of their own. This fact implies that the whole output data should be shared among all the processors in the system. For this reason, the exchange of the output results of each processor is necessary. It is, however, no so easy to do that. Such communication will be performed on the 16 bit basis; for example, a 16 bit data, which includes Y0000 through Y0015, will be transfered. These data, however, cannot be used directly. The following processes should be done to get proper results. Each processor clears the bits in its result which are not parts of its own, and transfers them to the others. Then the receiver performs bitwise OR operations between received data blocks and its own data block. But this process is time consumming and inefficient. In addition, if we use byte or word type structure, that problem is more fatal.

3. INFOBUS - The Information Bus

To construct a multiprocessor system, a system bus is required through which data communication is performed. So we proposed INFOBUS which is a dedicated system bus for MBPC. Main objectives of INFOBUS are to resolve the restrictions stated in Section 2.

The Figure 2 shows the signals and the basic structure of processing element(PE) on INFOBUS.

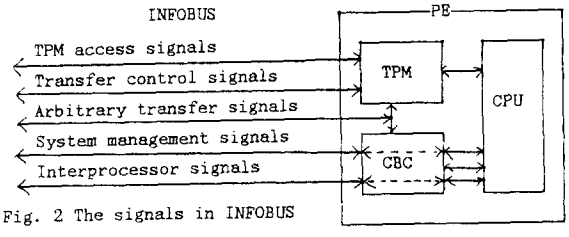


Fig. 2 The signals in INFOBUS

INFOBUS has the following 4 major characteristics

- Time-Slot shared bus
- Two-port-memory communication
- Multiple transfer operation
- ORed write transfer operation

There are many methods by which a system share a common system bus[6]. In case of INFOBUS, the time is divided into many slots and each PE gets the time slot sequentially. We will call the owner as a master and the others as slaves. The slaves are prevented from accessing the bus, while the master use it exclusively. The order of the master is scheduled by the system controller in MBPC system.

Each PE on INFOBUS should have specially designed two-port-memory(TPM). Through TPM, the PE's can communicate and exchange information with each other. In multiprocessor environment, a task should be divided and allocated to all the PE's. There are many methods to do this[7][8]. As stated in Section 2, however, it is almost impossible to divide the task(the ladder diagram) into some blocks which contains sequential addressed output points. Therefore, it is too difficult to divide the task not deteriorating the system performance.

Again refer to Fig. 1 and Section 2. Processor A produces N bits of output points which are randomly distributed, and transfers them to the other PE's. Then next processors do the same thing sequentially. After the end of communication, each PE should do logical OR operations among these data sets prior to use them as mentioned in Section 2. Because such data are too large, this procedure seems to be time-consuming and inefficient. To overcome this situation, the concepts of multiple transfer and ORed write transfer have been introduced. Multiple transfer can be stated that a master processor transfers a word of data to many other processors' TPM at one bus cycle simultaneously. And ORed write operation is a kind of read-modify-write operation which enables us to perform bitwise logical OR operations between the source and destination data at one

memory write cycle. Both operations can be implemented by adding some logic circuits into the TPM. These two concepts are the features that make INFOBUS to be well suited to MBPC. Combining them, we can enhance the speed of exchanging results among PE's, hence some restrictions described in Section 2 are eliminated.

It is required for INFOBUS to have real-time processing capabilities, because PC is a real-time processing system. To do this, INFOBUS has arbitrary transfer operation, by which the PE's can transfer information at any time, and some system management signals by which the system can transfer real time control information.

The mean transfer time of INFOBUS is an important parameter to the system behavior. Because memory cycle depends on the type of processor, a processor model has to be modeled. In this paper, we assumed that MBPC model is based on the working system[16]. We designed the PE using 68000 microprocessor, and the TPM using MOS static RAM with 120 nsec access time. And INFOBUS interface circuits are constructed mainly using TTL logic devices. Based on these assumptions we can get the transfer rate of INFOBUS in case of some conflicts.

The clock frequency of each PE is 8 MHz and we designed TPM using 22.12 MHz clock as its basic reference timing. The propagation delay time or port switching time in TPM is calculated from the circuits: the typical time for switching from port to port is 89.6 nsec .

Using this and the parameters taken from the 68000 data sheets, we could simulate the behavior of exchanging information through INFOBUS. The SDL/SIM has been used in this work, which can describe the discrete time events on the state transition basis[17][18]. We assumed that all the competing PE's are trying to access the TPM infinitely. One and only one master is performing multiple ORed write transfer to many TPM's on INFOBUS, while the other slaves are transfering data from their own TPM to local memory.

A typical block transfer routine written in 68000 assembly language is shown below:

```
    [ program 1 ]. A block transfer program
set:  move   #(word_size/2-1),d7  ; how many words?
      move.l #source_addr,a0      ; master = local
                                  ; slave  = TPM
      move.l #destination_addr,a1 ; master = TPM
                                  ; slave  = local
loop: move.l (a0)+,(a1)+          ; start transfer
      dbf    d7,loop              ; transfer
next: ..........                  ; next operations
```

We counted the mean clock cycles required to execute the [program 1], and it was shown that in case of no contention 15 clocks to transfer one word or 15/16 clocks to transfer one bit were required.

In real situations, however, the contention increases as the number of competing PE's increases. In our model of simulations, it is assumed that there are no contentions in local operations. And to access INFOBUS or TPM, the waiting time is added according to the exact behavior model of 68000 and INFOBUS.

The result is shown in Figure 3. The number in X-axis means the number of competing processors. In all cases there exits only one master. Hence, for example, 3 in X-axis means that one master and two slaves are competing for INFOBUS or TPM. From the graph, it is observed that the waiting time are negligibly small until there are four processors(one master and three slaves). And there are sudden increases in the mean transfer time at 5 and 8. The mean transfer time increases rapidly after 8. From the result we can state that the access cycle of the master and slave(s) are synchronized and stabilized after some contentions in case of less than 8, because all PE's execute the same routines with the same cycle time. But after 8, the contentions are increased very rapidly.

From the results, we can conclude that if we can schedule the number of PE's competing to access INFOBUS less than or equal to 4, there are no waiting time in the average.

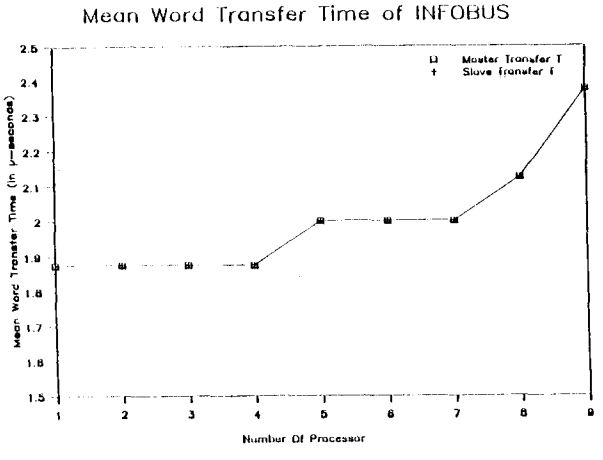


Fig.3 The mean access time of INFOBUS and TPM

4. The architecture of MBPC

MBPC is a programmable controller system based on the multiprocessor architecture. It consists of a system controller, discrete processors, I/O processors, and analog processors. INFOBUS is used as the system bus for MBPC. The system controller supervises the entire system and maintains INFOBUS. The discrete processors solve the ladder instructions, and I/O processors processes the input and output processing with the external. The analog processors perform the analog control functions such as PID. A typical configuration of MBPC is shown in fig.4.

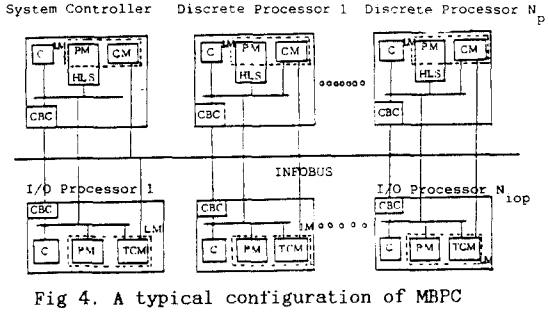


Fig 4. A typical configuration of MBPC

Based on the above configuration we will analyze the performance of MBPC. The most significant performance indices are the scanning time and the utilization of INFOBUS. Prior to the simulation of our model, the scheduling principle should be chosen. There are many scheduling techniques in multiprocessor environments[12][13][14]. In case of MBPC, the number of processors competing to use TPM may be

restricted to be less than 4 from the result of Section 3. But in this paper, we adopted the simplest form of scheduling, the first-come-first-served discipline, which schedules the master of each time slot in chronological order of arrival[15]. The optimility of such scheduling may be no good, but it is so simple that the software overhead for scheduling can be minimized and it is easily implemented in real time applications. The Figure 5 shows the scheduling principle used in our model of MBPC.

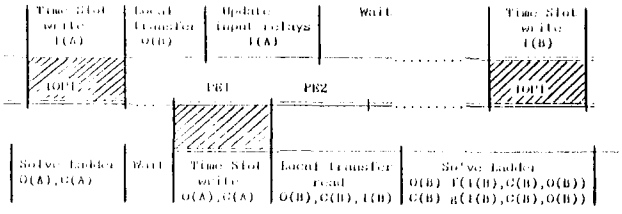


Figure 5. Scheduling of MBPC

The model of processing element for MBPC can be chosen with any structure. In our model, we chose a Logic Solving Processor(LSP)[21] to resolve the data type incompatibility problem. It will be described briefly later.

5. Simulation of MBPC

To construct the simulation model of MBPC, we assumed the followings.

- The mean transfer time through INFOBUS is assumed to be the result of Section 4.
- It is assumed that the analog processors are not included in the model. This assumption is acceptable because they operate independently with other PE's.
- It is assumed that one of the PE's acts as the system controller, and the overhead time as a system controller is negligbly small.
- The master of time slot performs multiple ORed transfer using the program 1 shown in Section 3, so the data produced from the previous simulation are assumed.
- It is assumed that each PE runs and accesses its own local memory with no waits.
- It is assumed that each IOP can update inputs and outputs at the rate of 1Mbps.
- MBPC is assumed to solve total of 64000 steps of ladder instructions with 8000 input, 8000 output, and 8000 control relays.
- Ladder diagram is identically distributed among the discrete processors. Nstep shown below is the number of steps which a discrete processor should solve:

$$Nstep = Ntotal/Np \quad \text{---------- 1}$$

where Ntotal : total steps of ladder instructions
Np : the number of discrete processor

- With appropriate software, we can allocate the control relays to the discrete processors in sequential order, but the output relays to be solved by each discrete processor cannot be allocated so, as stated in Section 2. Thus they should transfer all the data which contain the outputs that are not part of their own, clearing them as stated in Section 2 and 3. Using ORed transfer, only the unmasked data are effectively transferred. The IOP's can share the input and output data with equal length.
- It is also assumed that the discrete processors should service 3 kinds of real time clocks, which occurs every 1 ms, 10 ms, and 100 ms. The corresponding service routines was analyzed and the clock cycles was calculated. This yields the service time of each; 8, 12, 16 clocks respectively.

We can represent E[TSdsp], the mean time of time slot of each discrete processor, as follows:

$$E[Cl] = Nc/Np \quad \text{---------- 2a}$$
$$E[Ol] = No \quad \text{---------- 2b}$$
$$E[TSdsp] = (E[Cl] + E[Ol])\cdot Tbx$$
$$= (Nc/Np + No)\cdot Tbx \quad \text{---------- 2c}$$

where Nc, No : the total number of control relays and outputs
E[Cl],E[Ol] : the mean number of control relays and outputs a discrete processor should transfer
Tbx : the mean transfer time of one bit data through INFOBUS

At local transfer state, a discrete processor has to transfer all the data in the TPM which was written by the other PE's the previous scanning period into its own local memory. Thus we can write:

$$E[Tlocal] = (Ni + No + Nc)\cdot Tbx \quad \text{---------- 3a}$$
$$E[Tlocall] = No / Niop\cdot Tbx \quad \text{---------- 3b}$$

where E[Tlocal], E[Tlocall]
: the mean time of local transfer period of a discrete processor and IOP respectively

We can divide the input and output relays in a localized manner, i.e. IOP1 will handle X0-X511, while IOP2 will handle X512-X1023, etc. Thus each IOP will transfer E[Il] words at its time slot period.

$$E[TSiop] = E[Il]\cdot Tbx = Ni/Niop\cdot Tbx \quad \text{---------- 4}$$

where E[TSiop] : the mean of time slot of IOP
Niop : the number of IOP in the system

Referring to the Fig.5, we can establish some equations about the scanning time of discrete processor. For each discrete processor, E[Tdsp], the scanning time of a discrete processor means the time required to perform the sequence shown in the Fig.5.

$$E[Tdsp] = E[TSdsp] + E[Tlocal] + E[Tsol] + \alpha$$
$$= (No+Nc/Np)\cdot Tbx + (Ni+No+Nc)\cdot Tbx + E[Tsol] + \alpha$$
$$= 2E[TSdsp] + (1-1/Np)\cdot Nc\cdot Tbx + Ni\cdot Tbx + E[Tsol] + \alpha \quad \text{---------- 5}$$

where E[Tsol] : the time required to solve ladder
α : some overhead time, such as RTC service time and waiting time

If INFOBUS becomes the bottleneck of the system, then we can write the scanning time to be E[Tbus].

$$E[Tbus] = E[TSdsp] + E[TSiop]$$
$$= Np\cdot E[TSdsp] + Ni\cdot Tbx \quad \text{---------- 6}$$

Now we can write down the mean scanning time E[T], by combining eq.5 and eq.6.

$$E[Tiop] = E[TSiop] + E[Tlocall] + (Ni+No)/Niop\cdot Sbx$$
$$= (Ni+No)/Niop\cdot Tbx + (Ni+No)/Niop\cdot Sbx - 7a$$
$$E[T] = \max \{ E[Tbus], E[Tdsp], E[Tiop] \} \text{ --- } 7b$$

where E[Tiop]: the scanning time of the IOP
Sbx :input/output scanning rates of IOP

How to express the time needed to solve the ladder instructions? There are no standard ways to express it. So the strategy used in this paper will be described and formulated.

It is the main objective of PC to solve the ladder diagram, which includes pure boolean logic and block type instructions such as counter, timer, drum, shift, arithmetic operations etc. Our model of discrete processor executes the two types of instructions with different speed. For this reason, it is meaningful to express E[Tsol] as the function of the ratio of block type instructions to the entire ladder instructions. In addition, the block type instructions can be divided into basic block type instructions(BBI) and special block type instructions(SBI). The counters and timers are included in the BBI which are very frequently used in most PC applications.

By this strategy, the mean time to solve the ladder instructions, E[Tsol], can be expressed as follows:

$$E[Tsol] = E[Tbool]\cdot Nstep + (Tbbi\cdot Rbbi + Tsbi\cdot Rsbi)\cdot Rbtix\cdot Nstep \quad \text{---------- } 8$$

where E[Tbool] : mean execution time per one boolean instruction
Tbbi,Tsbi : mean execution of BBI, SBI
Rbti : the ratio of BTI to the Nstep
Rbbi,Rsbi : the ratio of BBI and SBI to BTI

To know E[Tbool], Tbbi, and Tsbi, it is required more exact data about the hardware and software. It is, however, difficult to get such data without existing system. Fortunately, we can acquire these data from the project to develop the large capacity PC[16] accomplished in our lab and enhanced model of it[21]. The system developed by the project, have been configured to include a system controller, I/O processor and a Hardware-Logic-Solver(HLS). And the enhanced model which is called as a Logic Solving Processor(LSP) contains Boolean Solving Unit(BSU) which is similar to HLS and a microprogrammed Register Solving Unit(RSU) to solve the BTI[21].

From the analysis of LSP, we can get the following data. BSU can solve the pure boolean instructions at the rate of 0.07msec/Kstep with 16Mhz clock. We programmed microprogram routines which implement the block type instructions in RSU of LSP, and analyzed them and got their mean cycle times in the form of hypoexponential distribution of order two:

$$f(t) = p1\cdot u1\cdot \exp(-u1\cdot t)+p2\cdot u2\cdot \exp(-u2\cdot t) \quad ; t \geq 0 \text{ ------- } 9$$

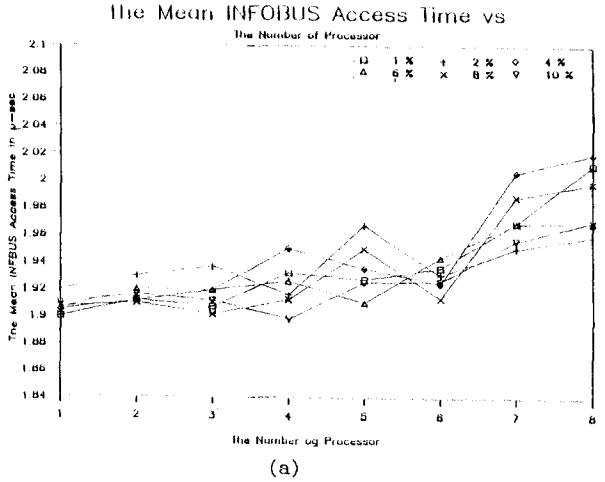
To use these data in analytic equations, we evaluated the mean of them. From this we can estrimate that E[Tbbi] and E[Tsbi] are 6.5clks/step and 4.3clks/step, respectively. Using them, we can rewrite the equation 8 as follows:

$$E[Tsol] = \{0.07 + (6.5\cdot Rbbi + 4.3\cdot Rsbi)\cdot Rbti\cdot Tclk\}\cdot Nstep \quad \text{---------- } 10a$$

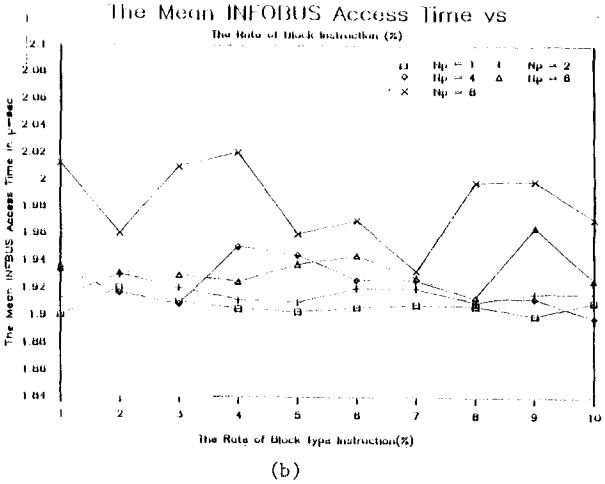
since, Rbbi + Rsbi = 1,

$$E[Tsol] = \{0.07+(4.3+2.2\cdot Rbbi)\cdot Rbti\cdot Tclk\}\cdot Nstep \quad \text{---------- } 10b$$

Now, we performs many simulations based on the above model and assumptions. In all cases, we assumed that there are 2 IOP in MBPC and 70% of BBI. In the graph, the number of processor means that the number of discrete processors.



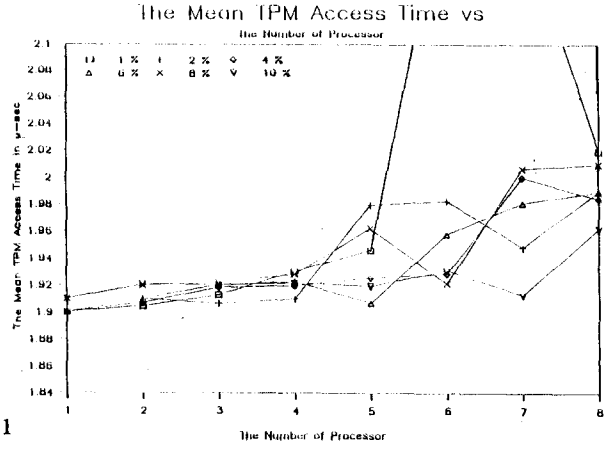
(a)

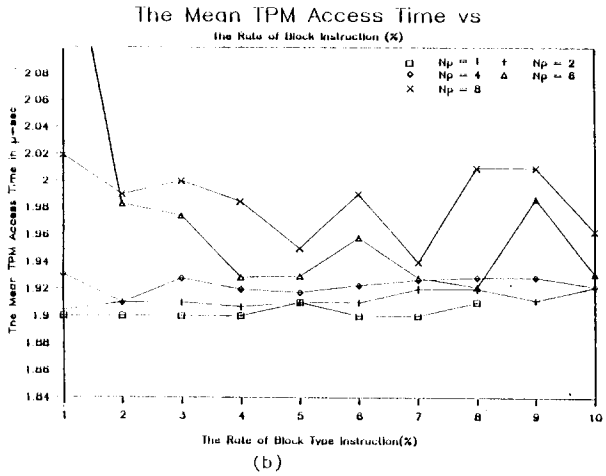


(b)

Fig.6 The mean INFOBUS access time

Figure 6 and 7 show the mean INFOBUS transfer time per word and the mean TPM access time versus the number of processors and the rate of BTI. From the graphs, it is observed that these parameters are gradually increasing as the number of processor increases, because the contentions to access INFOBUS and TPM also increaseas as the number of processor increases. Again it is observed that the change of the rate of BTI doesn't have no effects on the INFOBUS access.





(b)

Fig.7 The mean TPM access time

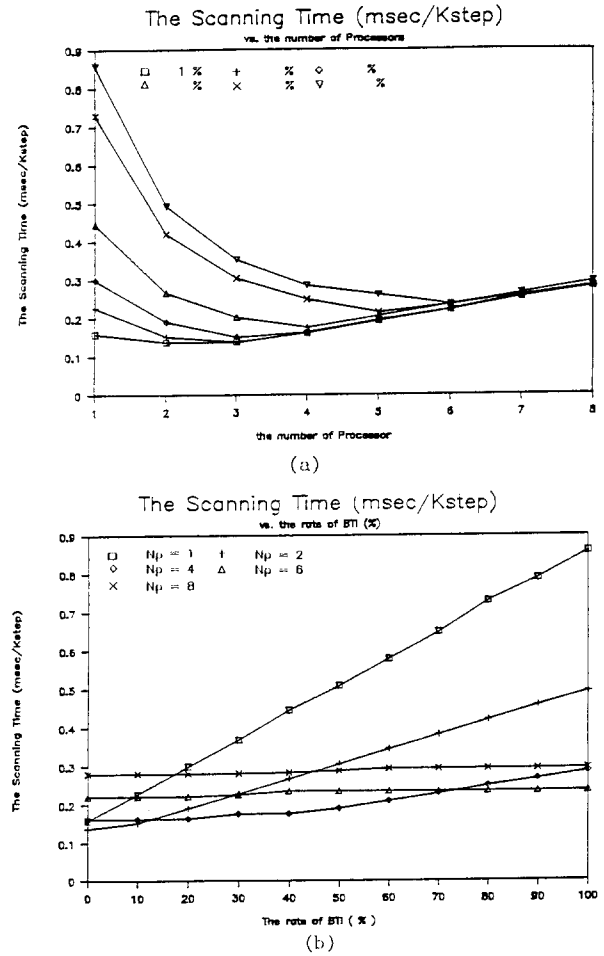


Fig.8 The mean scanning time

From Figure 8-(a), it is observed that there are minimum points in all cases. Those points are the time where E[Tbus] in equation 7b starts to be the bottleneck. These points are the optimal number of processors. For example, if we want to solve an application program which contains 40% of BTI, the optimal number of discrete processors is 4. From Fig 8-(b), it is observed that the scanning time is propotional to the rate of BTI when there are 2 processors. In case of 6 or 8 processors, however, the scanning time is almost constant because of E[Tbus]. From the two graphs, we can conclude that in all cases 4 processors in MBPC reveals the best performance.

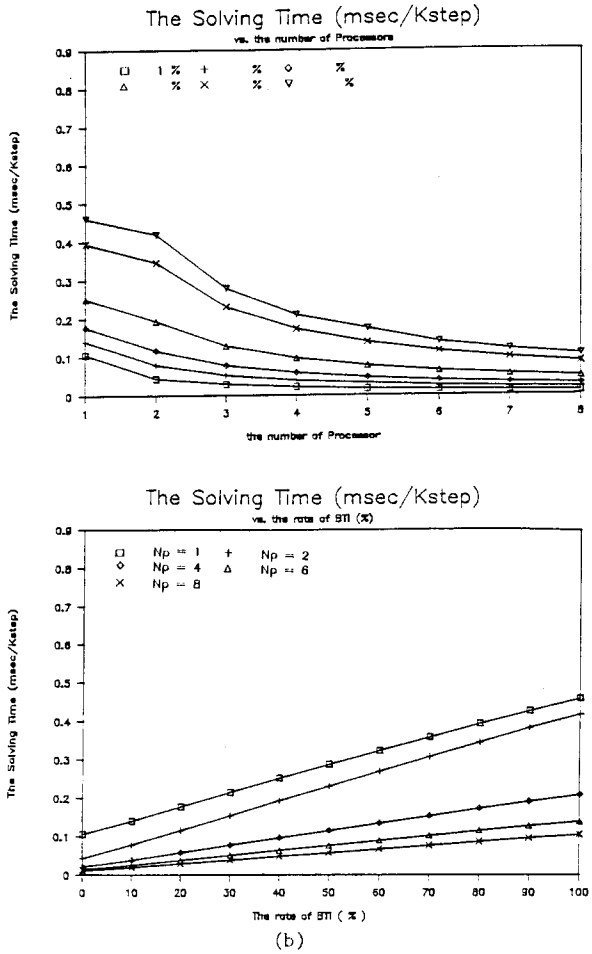


Fig.9 The mean solving time of ladder

From figure 9-(a), it is observed that the solving time of LSP is propotional to the inverse of the number of processors. Thus we can conclude that in the view point of ladder solving time it follows the function 1/n which means the optimal speed up factor in multiprocessing[15]. From Figure 9-(b), we can observe that the scanning time increases proportional to the rate of BTI as predicted in equation 8.

6. Conclusions

The features that make INFOBUS to be suitable for the multiprocessor based PC's are described in this paper. Since the interface circuits of INFOBUS including the control logic for TPM can be implemented using MSI/SSI or custom IC technology. Using the multiple transfer and ORed write transfer, the performance of INFOBUS, especially the time for exchanging data through INFOBUS, is remarkably enhanced.

We chose a specially designed Logic Solving Processor[21] as our fundamental model for processing element. Some useful and important results about MBPC are observed from the simulation of our exact

MBPC model. Using INFOBUS, the architecture of MBPC reveals good parallelism in spite of simple scheduling and simple task allocation. These properties are important properties in implementing actual multiprocessor architecture.

Moreover, the scan time of MBPC is very fast. In case of 50% of BTI and 4 processors, the scanning time is 0.194msec/Kstep. In case of TI 560/565, the scan time is 2.2ms/Kstep, and 1ms/Kstep in case of Modicon 584.

References

[1] Vasilii Zakharov, "Parallelism and Array Processing," IEEE TOC C-33, No.1, Jan. 1984

[2] Hibert D. Kirrmann and Felix Kufmann, "Poolpo- A Pool of Processors for Process Control Applications," IEEE TOC C-33, No.10, Oct, 1984

[3] J.M.Ayache, J.P.Courtiat, and M.Diaz, "REBUS, A Fault-Tolerant Distributed System for Industrial Real-Time Control," IEEE AC,AC-23, No.6, Dec. 1978

[4] Khalil M. Zahr, "Design Optimization of Microprocessor Based Remote Multiplexing Systems," IEEE AC, AC-23, No.6, Dec. 1978

[5] Lymam F. Brown, "A Role for Programmable Controllers in Factory Distributed Control," IEEE Tr. on Industry Applications, IA-21, No.4, 1985

[6] M.A.Marsan, G.Balbo, and G.Gonte, "Comparative Performance Analysis of Single Bus Multiprocessor Architectures," IEEE TOC C-31, No.12, Dec. 1982

[7] Benjamin W. Wah,"A comparative Study of Distributed Resource Sharing on Multiprocessors,"IEEE TOC C-33, No.8, Aug. 1984

[8] Richard S. Brice, J.C.Browne,"Feedback Coupled Resource Allocation Policies in the Multiprogramming -Multiprocessor Computer System," Comm. of the ACM, Aug. 1978

[9] D.P.Bhandakar,"Analysis of Memory Interference in Multiprocessors," IEEE TOC C-24,No.9, Sep. 1975

[10] M.A.Marsan,"Modeling Bus Contention and Memory Interference in a Multiprocessor System," IEEE TOC C-32, No.12, Dec. 1982

[11] T.Lang, M.Valero and I.Alegre,"Bandwidth of Crossbar and Multiple-Bus Connections for multiprocessors," IEEE TOC C-31, No. 12, Dec. 1982

[12] Zvi Rosberg,"Process Scheduling in a Computer System," IEEE TOC C-34, No.7, July 1985

[13] Reinhard Manner,"Hardware Task/Processor Scheduling in a Polyprocessor Environment," IEEE TOC C-33, No.7, July 1984

[14] D.G.Kafuraand V.Y.Shen,"Task Scheduling On a Multiprocessor System with Independent Memories,"SIAM J. of Comput., Vol. 6 No.1, 1977

[15] Domenico Ferrari," Computer Systems Performance Evaluation," Prentice-Hall, 1978

[16] Information System Laboratory," Development of Large Scale Programmable Controller," Final Report

[17] Bengt Stavenow and Jan Karlsson,"SDL applied to Discrete Event Simulation," SDL Newsletter, No.3, June. 1982

[18] Bengt Stavenow and Jan Karlsson, " SDL/SIM: A Simulation System for Discrete Event Simulation," Lund Institute of Technology, Lund, Sweden

[19] TI Inc., Texas Instruments Industrial Control Products, Model 560/565 Product Profile, Texas Instrument.

[20] GOULD Inc., 584. Microcode Machine Spec., Gould, Jan. 1980

[21] Jong-il Kim, Wook Hyun Kwon, and Jaehyun Park, "Architecture of a Logic Solving Processor For Programmable Controllers," Proc. of the 27th SICE, Aug. 2-4, 1988