An Architecture and its Performance Evaluation of A Multiprocessor Based Programmable Controller(MBPC)

Jongil Kim, Wookhyun Kwon, and Hongsung Park
Department of Control and Instrumentation Eng
Seoul National University, Korea

ABSTRACT

INFOBUS, which has been designed as a system bus of a multiprocessor system, will be introduced. And the concepts of the multiple transfer and ORed write transfer will be described. These concepts make INFOBUS to be well suited for use as the system bus of the multiprocessor based programmable controller(MBPC). In addition, the mean data transfer time through INFOBUS, which is one of the most significant performance of a bus, will be obtained by analysis and simulation.

Next, MBPC which uses INFOBUS as its system bus will be introduced, and some basic characteristics of MBPC will be described. The construction of exact model for MBPC will be given and simulated using SDL/SIM package. The reference system of our model will be briefly described also. Some results from the simulation will be given and validated.

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 constraints on how problems are executed, on which algorithms and programs are used, on how information can be exchanged.

Moreover, the most important factor limiting computational speed is not availability of appropriate which solves sequence programs concurrently with the updating I/O. With this approach, we can reduce the time required to update I/O.

Scan time, the speed at which a PC can execute its complete control cycle, is critical for fast response operations, therefore limits a PC's processing power and functionality. It is mainly determined not only by the time to update I/O, but by the time to solve its application program. We can reduce the time to update I/O, but how about the time to solve its application program? TI 560/565 can be configured to have two processors for solving the application program, but it does not utilize complete multiprocessing. It only executes the discrete part and the analog part of an application program concurrently. Also, there may be another approach to do this. A special-purpose processor design methodology has been adopted in Modicon 584 PC system[20]. It has been designed using microprogramming method with a bit-slice architecture, 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 many parameters to evaluate the performance of PC's. The processing speed, the capacity, and the reliability are the most significant indexes. In our simulation, however, we will focus on the processing speed; the scan time of PC, and will focus on the performance of INFOBUS.

In Section 2, we will describe some restrictions of PC on implementing multiprocessor architecture. The introduction of ladder diagram, which is widely used as a programming language of PC, will be also given.

In Section 3, we will describe INFOBUS, which has been designed as a dedicated system bus for the multiprocessor based PC. There are many features which make INFOBUS to be well suited for the PC with multiprocessor architecture. Among them, the concepts of multiple transfer and ORed write transfer are described more specifically. 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. Based on some reasonable assumptions, the model of MBPC will be constructed in Section 5, and simulated using SDL/SIM package[17][18]. Some simulation results will be given and validated.

2. Restrictions of PC 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., it has been chosen as the programming language because it is easily understood by the factory engineers. By now, the ladder diagram has been enhanced by the PC manufactures to add program flexibility and sophistication of a general minicomputer[5]. We will simulate the model of MBPC based on the ladder diagram, which is the most general language of PC.

Since each element in the ladder diagram corresponds to the relays, switches, etc., it can be represented by two states; ON and OFF. So we can say that the data operands of PC's are bit addressable. Since, however, most of microprocessors have byte or word type data structures, it is difficult 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.

Next problem arises from the sharing of the ladder diagram among the processors. If we can overcome the data type incompatibility problem, each processor will probably hold the data arranged in the form shown below;

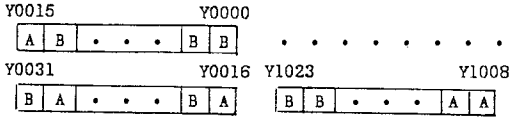


Fig 1. The output data arranged in PC

The ladder diagram has been divided and distributed to the two processors, A and B. In general cases, PC may change the state of output relays according to the ladder diagram given to it. In case of the processor A, it may change the output relays Y0015, Y0016, Y0032, etc., but must not change the output relays Y0000, Y0001, Y0014, etc., since they are included in processor B. In the course of solving, however, each processor may be able to use the data not a part of their own, so the exchange of results is required. But such communication will be performed on the word basis; for example, a 16 bit word data, which includes Y0000 through Y0015, will be exchanged between A and B. These data cannot be used directly, however. This situation can be resolved as follows; each processor clears the bits not a part of their own and sends them to the other. Then the receiver performs bitwise OR operations between received data set and his own data set cleared in the same way. If MBPC contains more than two processors, how to resolve such a problem? Does MBPC have to perform OR operations several times as needed? This problem is one of the most difficult problem to overcome in constructing MBPC.

3. INFOBUS - The Information Bus

INFOBUS has been designed as a dedicated system bus for the multiprocessor based programmable controller system (MBPC). It is intended to resolve the problems stated in Section 2.

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

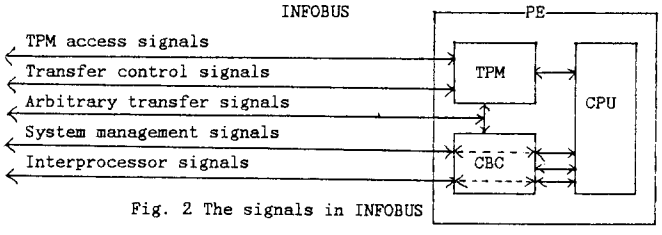


Fig. 2 The signals in INFOBUS

INFOBUS has the following 4 major characteristics :

- Time-Slot shared bus
- Exchange of information via specially designed two-port-memory(TPM)
- 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 only and only one of the PE's can use each time slot as owner. We will call the owner as a master and the others as slaves. The slaves are prevented from accessing the bus, so the master can use it alone. At least one system controller should be necessary to schedule the order of master.

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. To resolve the data type incompatibility, a specially structured hardware can be designed as done in Modicon 584[20]. Such hardware can be designed based on the relatively simple concept or designed to resemble a CPU in functional structure. In our working model[16], which will be described briefly later, a special hardware called a Hardware Logic Solver (HLS) has been designed. But still there are some problems in constructing a multiprocessor based PC. In multiprocessor environment, a task should be divided and distributed to PE's. There are many methods to do this[7][8]. It is almost impossible to divide the task(the ladder diagram) to include all the linear part of the data; i.e, in Fig 1, processor A contains Y0000 through Y0511 while processor B contains Y0512 through Y1023. This is so because these data operands are uniformly distributed to the whole ladder diagram. The disadvantages of such a problem has been discussed in Section 2. Therefore, it is too difficult to divide the task not deteriorating the system performance and to minimize the data type incompatibility problem.

Again refer to Fig. 1. Processor A produces N bits of results 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. 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. And ORed write operation enables us to perform bitwise logical OR operations between the source and destination data at one memory write cycle. This is a kind of read-modify-write operation. Both operations can be easily implemented by adding some logic into the TPM. These two concepts are the features that make INFOBUS well suited to MBPC. Combining them we can enhance the speed of exchange results among PE's, hence some restrictions described in Section 2 are eliminated.

Also INFOBUS has some general features as system bus. INFOBUS has been designed to be a system bus for PC, real-time processing capabilities are necessary. Thus, INFOBUS has arbitary transfer operation, by which the PE's can transfer information at any time. And INFOBUS contains some system management signals which make the system to transfer real time information and 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 microprocessor based system is to be modeled. In this paper, the construction of model is based on the working system[16]. We assummed the PE's are designed using 68000 microprocessor, and TPM's are designed using MOS static RAM with 120 nsec access time which are commercially available now. And INFOBUS interface circuits are constructed mainly using TTL logic devices. Based on these assumptions we can get more concrete data.

Each CPU is running at the frequency of 8 MHz. We designed TPM using 22.12 MHz clock as its basic reference timing, and calculated the propagation delay time and port switching time. As a result the value 89.6 nsec has been obtained as the time of switching from port to port in TPM.

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 assummed 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 reading data memory from their own TPM and transfer them to their own local

The simple and probably the most effective block transfer program written in 68000 assembly language is shown below:

[program 1]. A block transfer program

```
set: move   #(word_size/2-1),d7 ;how many 32 bit words?
     move.l #source_addr,a0     ;master = local memory
                                ;slave  = TPM
     move.l #destination_addr,a1;master = TPM
                                ;slave  = local memory
loop:move.l (a0)+,(a1)+         ;block transfer here
     dbf    d7,transfer
next:  ...                      ;next operation
```

We have counted up the mean clock cycles needed to execute the [program 1]; 15 clocks to transfer one word or 15/16 clocks to transfer one bit in case of no contention.

In real situations, however, the contention increases as the number of PE's increases. It is possible to predict it using some equations, but the statistical property of the contention makes it inaccurate. The simulation is the best way to obtain more accurate and realistic results[9][10][11].

In our model of simulations, the execution cycle of each instruction in program 1 corresponds to one state in the simulation, and each state wastes the execution time of its original cycles. Except for the memory cycles accessing INFOBUS or TPM, it is assummed that there are no contentions. When contention occurs in accessing INFOBUS or TPM, however, the waiting time is added according to the exact model of 68000 and INFOBUS.

The result is shown in Figure 3. The number in X-axis means the number of processors. And in all cases there exits only one master. Hence, for example, the 3 means that there are one master and two slaves exist. We can observe that in all cases INFOBUS and TPM access time take almost the same value. From the graph, it is observed that the waiting time are negligibly small until there exist four processors (one master and two slaves). And there occurs a sudden increase in the waiting time at 5 and 7. The waiting time increases rapidly and continuously after 7. This

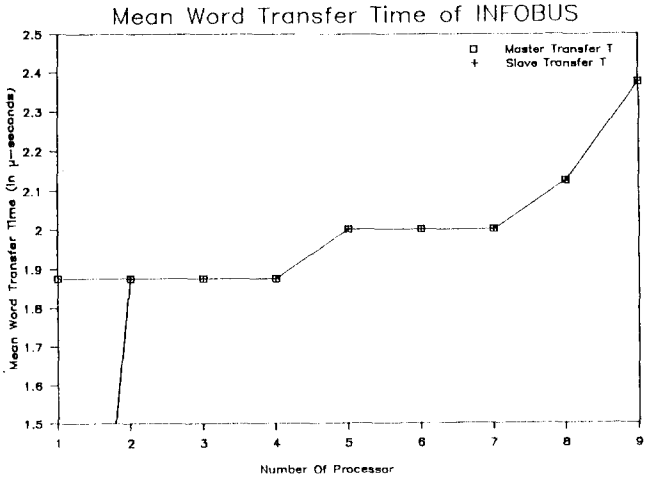


Fig.3 The mean access time of INFOBUS and TPM

result seems to be very extraordinary. From the fact that all PE's execute the same instructions, however, we can state that the bus utilization can be synchronized and stablized after some contentions in the case of less than 8. But after 8 this is no longer true.

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, the most efficient bus utilization can be obtained.

4. MBPC - The Multiprocessor Based PC

MBPC is a programmable controller based on the multiprocessor architecture, which has INFOBUS as its system bus. It consists of a system controller, discrete processors, I/O processors, and analog processors. The system controller supervises the entire system and maintains INFOBUS. The discrete processors solve the ladder instructions, and I/O processors handle the input and output data which have been acquired from or will be sent to the external system. The analog processors perform the analog control functions such as PID. A typical configuration of MBPC is shown:

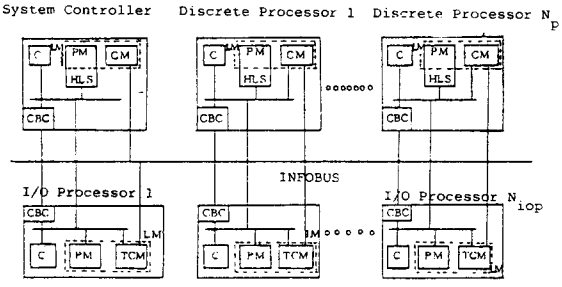


Fig 4. A typical configuration of MBPC

Based on the above configuration we will investigate the performance of MBPC; especially, the most significant performance index, the scanning time, and the utilization of INFOBUS.

Prior to the simulatation of our model, the scheduling principle should be determined. There are many scheduling techiniques available on multiprocessor environments[12][13][14]. In case of MBPC, the number of processors competing to use TPM should be maintained 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 selects the master of time slot in chronological order of arrival [15]. We don't care of the optimality of scheduling, but this is so simple that the software overhead for scheduling can be minimized. The Figure 5 shows the scheduling principle used in MBPC.

5. Simulation of MBPC

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

- We will use the exact data taken from the result of Section 2.
- We neglect the analog processors. This assumption is acceptible because they operate independently with other PE's.
- The system controller is selected among the PEs, and it is considered to work just like the other PE's.
- The master of time slot performs multiple ORed transfer using the program 1 shown above, so the data produced from the previous simulation are useful.
- Each PE runs and accesses its own local memory with no waits.
- Each IOP can update all inputs and outputs sufficiently fast.
- Each 68000 CPU runs at the frequency of 8 MHz.
- MBPC is assummed to solve total of 64000 steps of ladder instructions with 8000 inputs, 8000 outputs, 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:

$$Nstep = Ntotal/Np \qquad \text{——— 1}$$

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

- With appropriate software, we can distribute the control relays identically to the discrete processors, but the output relays to be solved by each discrete processor cannot be divided so. 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 ouput data with equal length. Refer to Figure 5.
- We also assummed 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 have been investigated and the clock cycles have been calculated. This yields the service time of each; 144.608, 194.912, and 245.216 clocks.

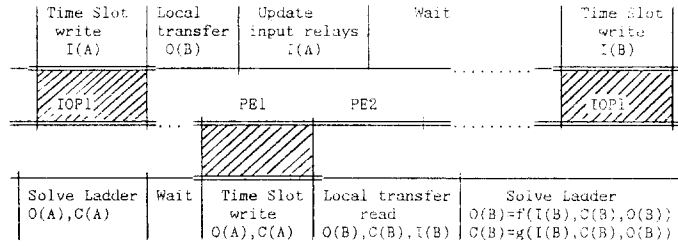


Figure 5. Scheduling of MBPC

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

$$E[Cl] = Nc/Np \qquad \text{——— 2a}$$

$$E[Ol] = No \qquad \text{——— 2b}$$

$$E[TSdsp] = (E[Cl] + E[Ol]) \cdot Tbx$$

$$= (No + Nc/Np) \cdot Tbx \qquad \text{——— 2c}$$

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 has been written by the other PE's just previous scanning period into his own local memory. Thus we can write:

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

where E[Tlocal] : the mean time of local transfer period of a discrate processor

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 time 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.

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

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].

$$\begin{aligned} E[Tbus] &= \sum E[TSdsp] + \sum E[TSiop] \\ &= Np\ E[TSdsp] + Ni\ Tbx \end{aligned} \quad \text{---------- 6}$$

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

$$E[T] = \max \{ E[Tbus], E[Tdsp], E[Tiop] \} \quad \text{---- 7a}$$

where E[Tiop] : the scanning time of the IOP

E[Tiop] has been assummed to be sufficiently small. Thus, eq.7a can be reduced to the following eq. 7b:

$$\begin{aligned} E[T] &= \max \{E[Tbus], E[Tdsp]\} \\ &= \max \{Np\ E[Tdsp], 2E[TSdsp]+(1-1/Np) \cdot Nc \cdot Tbx \\ &\quad + E[Tsol]+\alpha\} + Ni \cdot Tbx \end{aligned} \quad \text{---------- 7b}$$

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. 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:

$$\begin{aligned} E[Tsol] &= E[Tbool] \cdot Nstep + (Tbbi \cdot Rbbi \\ &\quad + Tsbi \cdot Rsbi) \cdot Rbti \cdot Nstep \end{aligned} \quad \text{---------- 8}$$

where E[Tbool] : mean execution time per one pure 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 gain such data without existing system. Fortunately, we can acquire these data from the project accomplished in our lab to develop the large capacity PC[16]. The system developed by the project, have been configured to include a sytem controller, I/O processor and a hardware-logic-solver. The system controller of the project is very similiar to the discrete processor in our MBPC, but there are no INFOBUS interface and TPM. But if we consider the solving of ladder instructions only, that system is very realistic model for our discrete processor.

In both cases, HLS are included which can solve the pure boolean instructions, but the block type instructions cannot be solved by HLS. So the block type interface logic has been designed to pass the block type instruction to discrete processor. Using this logic, HLS solves BTI with the aid of discrete processor.

The HLS can solve the pure boolean instructions at the rate of 0.1msec/Kstep. We can enhance it further, but 0.1msec/Kstep will be used as E[Tbool] in our simulation.

We have designed softwares which implement the block type instructions using 68000 assembly languages. We have studied many routines to implement the block type instructions, counted the execution cycles needed, and gotten their mean cycle times in the form of hypoexponential distribution of order two:

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

To use these data in analytic equations, we evaluated the mean of them. This yields 256.7 clks/step for Tbbi, and 210.9 clks/step for Tsbi. Using them, we can rewrite the eqation 8 as follows:

$$\begin{aligned} E[Tsol] &= \{ 0.1 + (256.7 \cdot Rbbi \\ &\quad + 210.9 \cdot Rsbi) \cdot Rbti \cdot Tclk) \cdot Nstep \end{aligned} \quad \text{---- 10a}$$

since, Rbbi + Rsbi = 1,

$$\begin{aligned} E[Tsol] &= \{ 0.1 + (210.9 \\ &\quad + 45.8 \cdot Rbbi) \cdot Rbti \cdot Tclk) \cdot Nstep \end{aligned} \quad \text{---- 11b}$$

From the simulation, the transition activity diagram of MBPC has been obtained and shown in Figure 6. The

diagram was taken from the data showing the state

transition of each PE between 1540 msec and 1650 msec. We can choose any point in the diagram as the reference. If we choose 1552.5ms, the starting point of time slot of IOP1, as the reference time, the start of next time slot is at 1605.4ms, so the scanning time in this case becomes 53ms to complete 64Kstep, or 0.83 msec/Kstep. From the diagram, we can conclude that most of the activities of discrete processors are lie on the state of solving ladder instructions, and IOP's are in the waiting state. Also the discrete processors are observed to be in wait state so rarely, but at 1550.4 msec the discrete processor 1 completes its solving state and enters waiting state, because discrete 3 does not complete its local transfer yet.

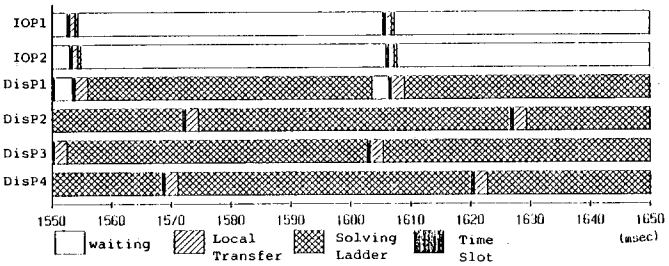


Fig.6 The transition activity diagram of MBPC

Figure 7 and 8 show the mean INFOBUS transfer time per word and the mean TPM access time as the number of processors increases. It is observed that these parameters are gradually increasing as the number of processor increases. We can state that as the number of processor increases the contentions to access the INFOBUS and TPM also increase.

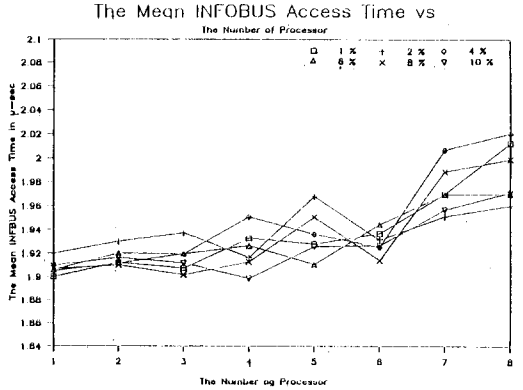


Fig.7 The mean INFOBUS access time (1)

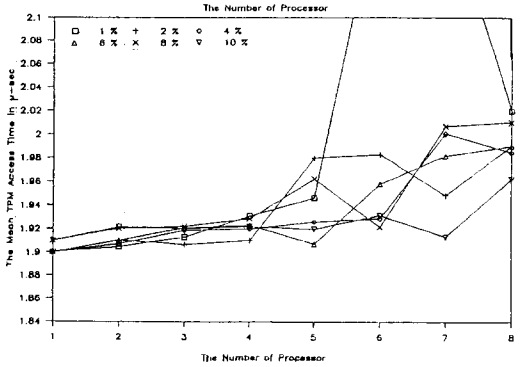


Fig.8 The mean TPM access time (1)

Figure 9 and 10 show the same parameters with changing the ratio of BTI. From these graphs, it is observed that the change of the rate of BTI doesn't have no effects on the parameters relating to the contention.

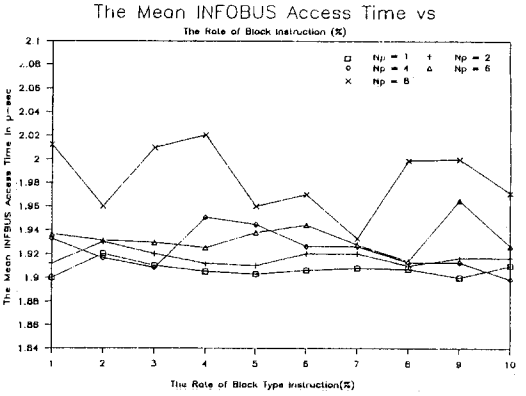


Fig.9 The mean INFOBUS access time (2)

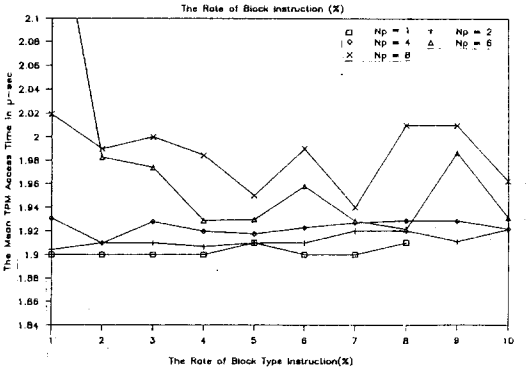
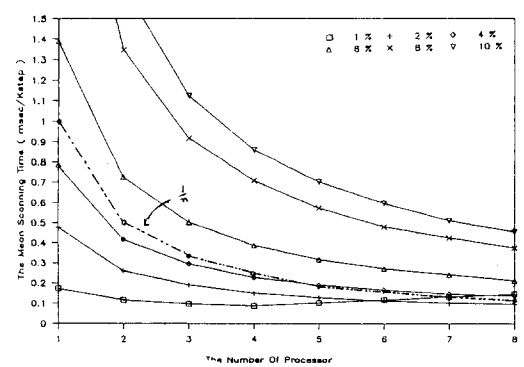


Fig.10 The mean TPM access time (2)

From Figure 11, it is observed that in most cases the graphs, representing the mean scanning time versus the number of processor, follow the function 1/n which means the best case of speed up factor with multiprocessor[15]. Thus we can conclude that MBPC has very high parallism, and MBPC has a cost effective architecture. This can be clarified by the graph shown in Figure 12, which represents the rate of the change of scanning time when the number of discrete processor increases at a given Rbti. As the number of discrete processor increases, the change of scaning time imposed by the change of Rbti decreases.



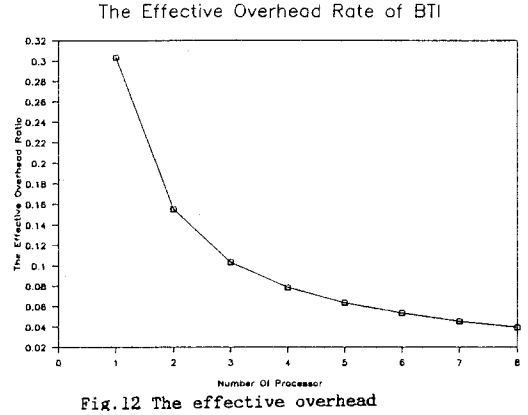


Fig.12 The effective overhead

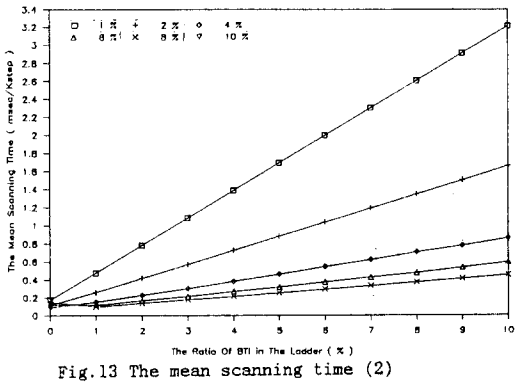


Fig.13 The mean scanning time (2)

From Figure 13, we can see that the scaning time is increasing propotionally to the rate of the block type instructions.

6. Conclusions

The features that make INFOBUS to be suitable for the multiprocessor based PC's are described. Since the interface circuits of INFOBUS including the control logic for TPM can be implemented using custom IC technology, INFOBUS can be realized easily and relatively small efforts. Using the multiple transfer and ORed write transfer, the performance of INFOBUS, especially the mean transfer time through INFOBUS, is remarkably enhanced.

We have chosen a working large-scale programmable controller[16] as our fundamental model. And it have been modified to become MBPC described throughout the paper. Some useful and important results about MBPC have been observed from the graphs which have been obtained from the simulation of our exact MBPC model. These results are based on the exact model. Using INFOBUS, MBPC can have a good parallism inspite of simple scheduling, so the overhead of scheduling can be minimized. This is a good property in implementing multiprocessor architecture.

Moreover, the scan time of MBPC is relatively fast, although it has simple distribution strategy of task and simple schedule strategy. 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 og 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 Indenpendent 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