ELSEVIER

Microprocessors and Microsystems 21 (1997) 113–120

MICROPROCESSORS AND MICROSYSTEMS

A transputer-based parallel controller for motor control applications

Ming-Fa Tsai*, Ying-Yu Tzou

Power Electronics and Motion Control Lab., Institute of Control Engineering, National Chiao Tung Univ., Taiwan, R.O.C.

Accepted 27 March 1997

Abstract

This paper presents the design and implementation of a parallel controller based on the transputers IMS T800-20 for high-performance motor control applications. A unified controller architecture comprising transputer-based parallel computing boards and input/output boards suitable for the real-time control of various types of motor drives is proposed. The system can increase its computing and input/output processing capability by paralleling these boards. A host server based on a personal computer for user interface is also developed. To demonstrate the system's capability, we have applied the system to the fully digital adaptive control of an AC induction servo motor. The control functions can be easily implemented in parallel by using the high-level programming language Occam. The comparison with two existing parallel controllers also shows the performance and architectural features of the system. © 1997 Elsevier Science B.V.

Keywords: Transputers; Parallel controller architecture; Motor control

1. Introduction

Recently, interest in parallel-processing system architecture for real-time control applications has been growing rapidly due to increasing demands on system performances and the availability of advanced microprocessors. Many studies concerning the development of the parallel controller architecture for the application of advanced motor control are available. The most widely used parallel controller design is based on the tightly coupled multiprocessor system, in which the communication among processors is through the use of shared memory [1,2]. This type of system architecture is effective when only a certain number of processors are employed. If the number of processors in the system is increased, however, a communication bottleneck will occur, and hence the efficiency will be degraded. Further, the system is usually complex to design and requires the writing of a complex operating system to manipulate the programming. Therefore, a loosely coupled system design based on multicomputers, such as INMOS transputers which use a message passing method for interprocessors communication, has been suggested as an alternative to implement high-performance servo motor drives [3,4] and other real-time control systems [5–7].

In order to build up high-performance parallel controller using the transputers, it is important that the transputer's I/O (input/output) processing should be performed with a minimal delay. According to INMOS [10], two different methods can be used for the interface of the transputer to the real-world environment. One is using link adaptors and the other one is using a memory mapping bus. Harley et al. [4] reported that the memory-mapping method is about 18 times faster than using a link adaptor for the same I/O data transfer. Although much work about using the transputers for real-time control has been done, the implementation of necessary I/O interfaces to controlled plants for fast real-time control by use of the memory-mapping scheme is still lacking [8,9]. Most of the work uses link adaptors for I/O handling [3,5–7], and hence this may influence the system performance.

The purpose of this work is to design and implement a configurable and programmable transputer-based parallel controller system, where many types of peripheral hardware suitable for various types of motor control applications have been designed by means of the memory-mapping scheme so as to enhance the performance and flexibility of the system. The main techniques for the hardware and software designs of the system will be described. The system may provide a flexible and easy-to-use experimental setup for the implementation of sophisticated control algorithms, such as adaptive and learning control, for

* Corresponding author

PII S0141-9331(97)00025-2

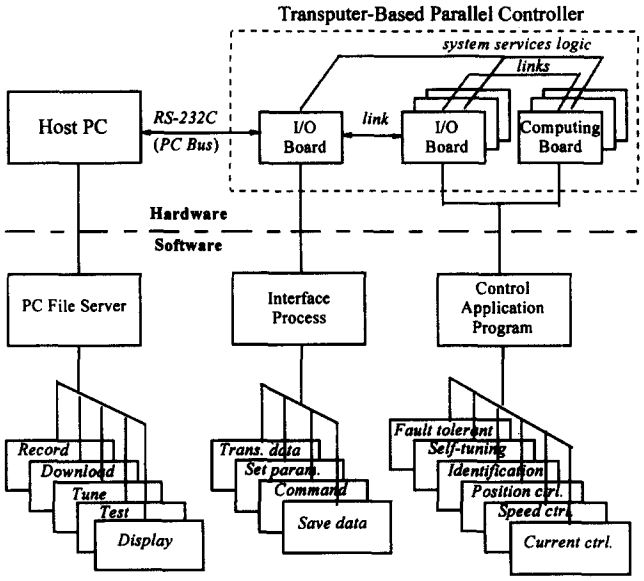


Fig. 1. Transputer-based parallel control system architecture.

high-performance motor control and power electronics applications.

2. System overview

Modern digital control systems typically perform real-time control and identification functions together with a number of other activities related to I/O handling, data-base management, and user interface processing. Some of the operations such as current control in a servo drive should be performed within a short sampling time interval (less than 100 μs) for high-performance control requirements. The system should also provide sufficient numerical computation capability for advanced control algorithms. These considerations led to the development of the proposed control system architecture, as shown in Fig. 1, which consists of a transputer-based parallel controller and a host PC (personal computer). The hardware of the

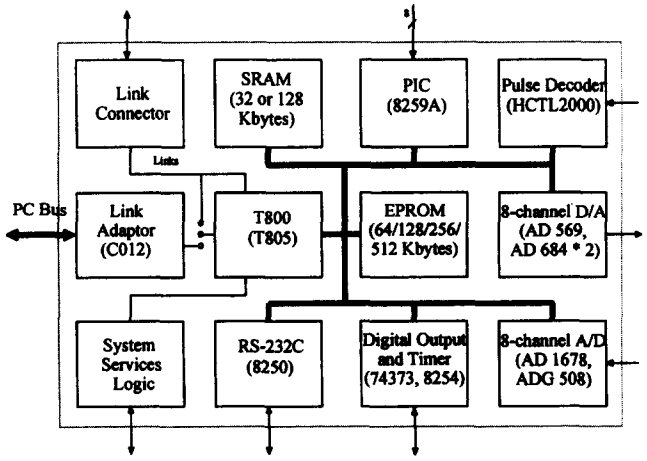


Fig. 2. Block diagram of the transputer-based input/output board.

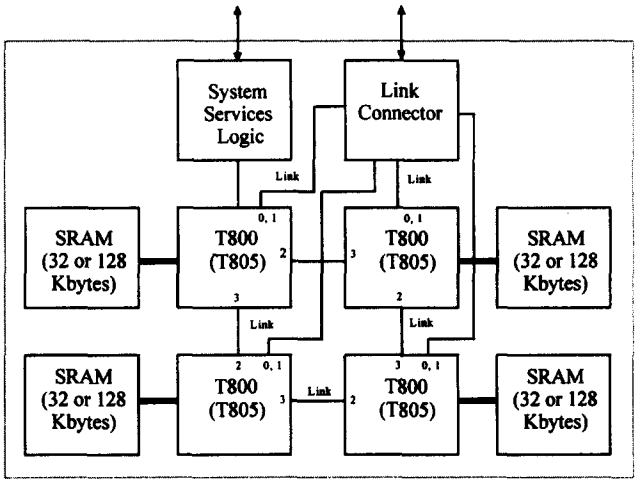


Fig. 3. Block diagram of the transputer-based parallel computing board.

parallel controller consists of one or more transputer-based input/output boards to provide fast I/O processing capability and one or more transputer-based parallel computing boards to provide computational power, as required by advanced control applications. The transputer is a family of high-performance microcomputers specifically designed for parallel processing. Its name was taken from the prefix of transistor, *trans*, and the suffix of computer, *puter*, since it is both a computer on a chip and a silicon component like a transistor. The transputer IMS T800-20 provides a processing power up to 10 MIPS (million instructions per second) and 1.5 MFLOPS (million floating-point operations per second), it also contains a multitasking kernel, which controls the scheduling of multiple tasks within the transputer [10,13]. Other important features of the chip are its four high-speed serial communication links (up to 20 Mbits/s), as well as system services, which include reset, analyse, and error handling logic to initialize and sustain operation of the chip. Using the interconnection of communication links and the system services logic among the transputers, the designed two transputer boards can operate concurrently to perform control functions in parallel. One of the I/O boards is connected to the host PC through an RS-232C serial interface or the PC bus for user interface processing. The host PC can also be used as a software development workstation when the Transputer Development System (TDS) software is implemented [12].

The controller can be interfaced to a motor (or multiple-motor) drive system. Control of the motor is achieved through the accesses to the peripheral devices, which are memory-mapped to the transputer in the I/O board shown in Fig. 2. This board contains a T800-20 transputer, a system services control logic block, up to 512 Kbytes of eraseable programmable read-only memory (EPROM), up to 128 Kbytes of static random-access memory (SRAM), and various I/O interfaces including an eight-channel multiplexed digital-to-analog converter (DAC), an eight-channel multiplexed analog-to-digital converter (ADC), a pulse

decoder/counter (HCTL 2000), and a programmable counter etc. to get feedback signals from the motor's sensors and provide drive signals to the motor. Furthermore, because the transputer is rather limited in its capability for handling multiple external interrupts, a programmable interrupt controller using Intel 8259A is also included for supporting the mechanism. The number of these I/O devices required to fulfil the sensor and drive functions has been reduced considerably by standardizing the input and output signals so that many types of motors can easily be interfaced to the system.

However, to provide sufficient computational power to the controller, the parallel computing board contains four transputers configured as an expandable structure, as shown in Fig. 3. Each transputer in this board is implemented with up to 128 Kbytes of external SRAM memory. Thus the board would have a processing power of 40 MIPS, 6 MFLOPS, and a memory of 1/2 Mbyte. When computational requirements increase, additional power can be obtained by adding more of the parallel computing boards to the system and connecting the appropriate system service logic and communication links to the boards. Therefore, various types of controller architecture topology, such as arrays, meshes, hypercubes, pyramids, and other distributed networks, can easily be built up.

3. Hardware design

In this section, we describe two major techniques for the controller hardware design: memory-mapping I/O interface and multiple-interrupt control mechanism. It should be noted that all signals followed by '/' indicate that the signals are active low.

3.1. Memory-mapping I/O interface design

The transputer architecture provides a set of configurable control timing cycle settling for easy interface to external memory and for communication with peripherals [10]. The external memory read/write cycle is divided into six Tstates and each Tstate may be configured to be from one to four Tm periods long. Each Tm period is 25 ns for the transputer T800-20 (20 MHz) used. Using this facility, we can implement the external memory and I/O interface for the transputer with minimum access time. Based on the timing characteristics of the external memory and all the I/O devices used in the input/output board, we can divide these devices into two groups: the faster-timing devices (CY7C199, 27C64, AD 569, HCTL2000, 8254) and the slower-timing devices (AD1678, 8250, 8259A), for access of the devices in each group with minimal delay. For both groups, the read/write cycle is configured by connecting the MemConfig to the MemAD9 of the transputer, as shown in Fig. 4. In this way, occupying eight Tm periods, the six-Tstate sequence for the faster-timing devices is T1–T2–T3–T3–T4–T5–T5–T6, and hence the read/write cycle time is 200 ns. For the slower-timing devices, several memory-wait states are inserted to delay the cycle time, allowing all the devices in this group to be accessed in one cycle. This can be achieved by connecting the output of the two-input AND gate, the inputs of which are the chip select signal

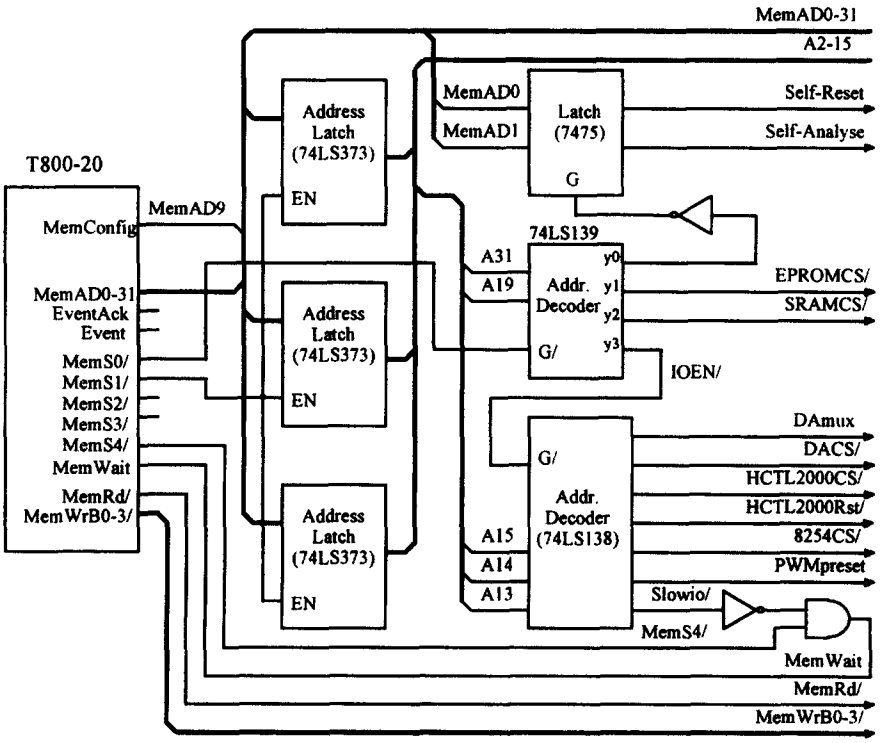


Fig. 4. Schematic diagram of the memory-mapping I/O interface.

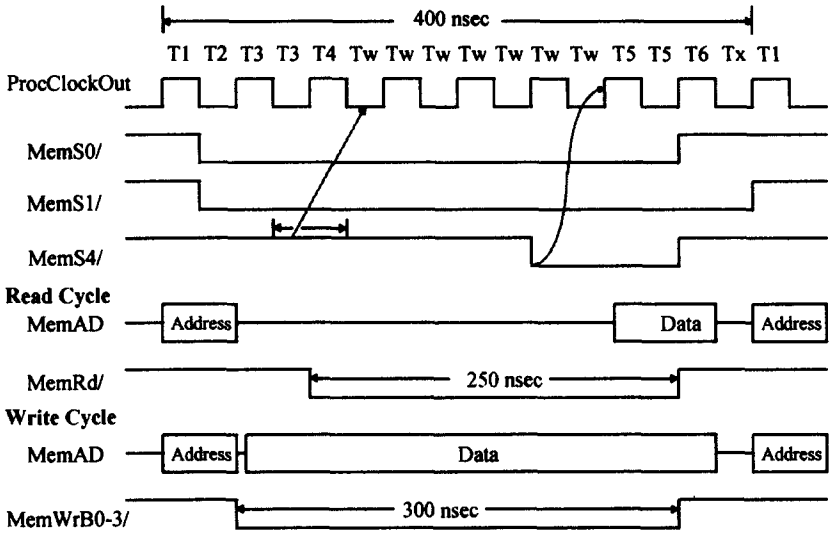


Fig. 5. Read/write configuration control timing for the slower I/O devices.

(Slowio/) for this group of devices and the MemS4/, to the MemWait pin of the transputer, as shown in Fig. 4. The MemWait pin is sampled during two Tm periods before the end of the T4 or the wait period, and then a wait period can be inserted at the end of T4 if the MemWait is high when sampled. The resulting read/write control timing for the slower-timing devices is shown in Fig. 5. It should be noted that an extra period, Tx, is inserted at the end of the T6 period. This is because the T5 period must always begin on a rising edge of the ProcClockOut signal when wait states are inserted and the external memory interface cycle should consist of an even number of Tm periods. The read/write cycle time for the slower-timing devices is then 400 ns long.

Because the external SRAM memory (CY7C199) implemented in the parallel computing board has high speed access time, the read/write cycle for each transputer on this board is configured by connecting the Memconfig to the MemAD8. The six-Tstate sequence is then T1–T2–T3–T4–T5–T6 and the read/write cycle time is 150 ns. No wait states are inserted in the read/write cycle so as to minimize the access time to the external memory in this board.

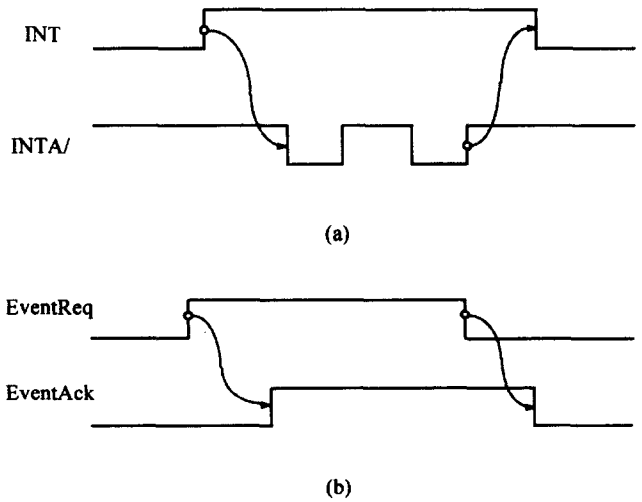


Fig. 6. Interrupt request and acknowledgment timing for (a) 8259A and (b) T800.

3.2. Multiple-interrupt control mechanism

In a practical real-time controller multiple interrupts often occur, but the transputer provides only one event (interrupt) pin. In the literature, the design of a multiple-interrupt handler for a transputer using a five-input OR gate and an event register has been proposed [4]. In our system the Intel 8259A programmable interrupt controller (PIC) device is adopted. This device is specifically designed for use with an Intel 80xx microprocessor-based system. It can handle up to eight-priority interrupts for the microprocessor by placing an interrupt-vector byte on the data bus during an interrupt acknowledgment pulse interval [14]. In order to make the 8259A work together with the transputer (T800), they should have compatible timing requirement for the interrupt handling. The interrupt request and acknowledgment timing characteristics for the 8259A working in the Intel 8086 or 8088 mode is shown in Fig. 6a. As can be seen from this figure, the INTA/ signal for the 8259A contains two negative pulses. But the EventAck signal issued from the transputer is a positive pulse shown in Fig. 6b. Therefore, it becomes necessary to design an interface circuit to transform the EventAck signal into two negative pulses.

Figure 7 shows the schematic diagram to implement this scheme. The circuit uses the EventAck as the triggering signal for the D-type flip flop (74LS74). Initially, the $\bar{Q}$ output of the flip-flop is logic high. As soon as an interrupt request occurs, the EventAck signal becomes high, and the $\bar{Q}$ output becomes low and remains low until the $\bar{Q}$ output

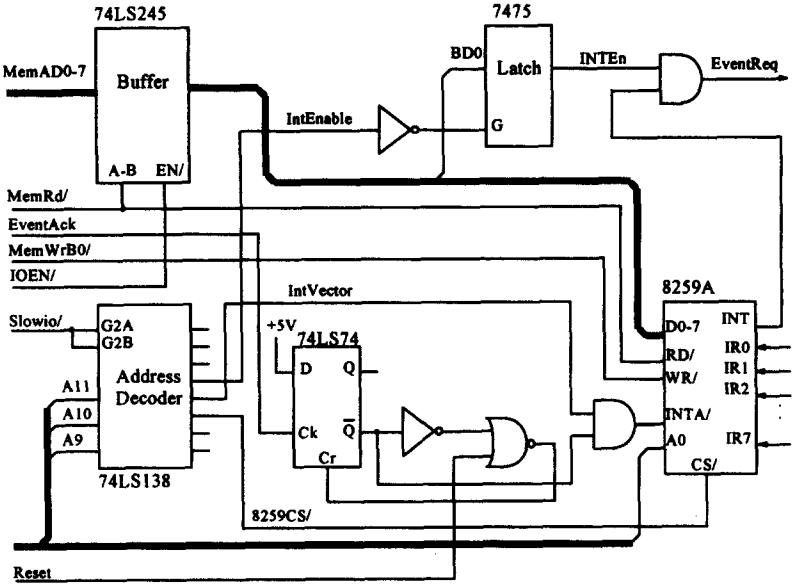


Fig. 7. Schematic diagram for the multiple-interrupt control mechanism.

signal gets a delay, which is approximately equal to the pulse width, through the NOT and NOR gates and clears the flip-flop. The first negative pulse is then generated. Afterwards, a read instruction for getting the interrupt vector from the data bus is then issued from the transputer. This in turn generates the second negative pulse.

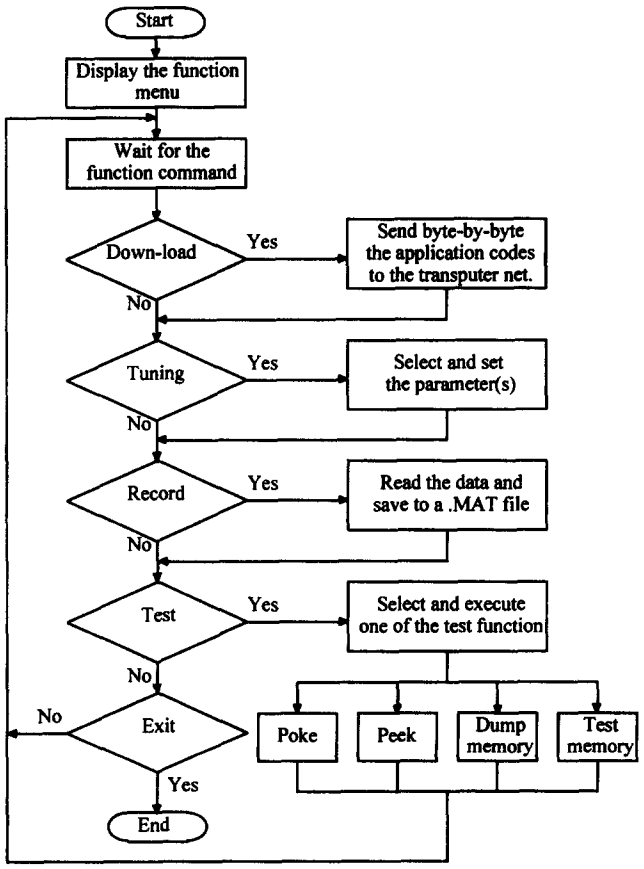


Fig. 8. Flow chart of the host server.

4. Software design

The system software can be separated into two parts: the software on the parallel controller and software on the host PC. As the hardwired kernel in each transputer can handle the process scheduling, communication, and synchronization automatically [10,13], the system software design in the controller turns out to be very simple. It simply consists of an EPROM loader, which can boot from ROM and load a transputer network from an RS-232C port. The EPROM loader in a designed transputer-based I/O board has been created by modifying a monitor program provided in the TDS tools directory [12]. This program contains the Occam language source of the standard ROM loader used in the INMOS evaluation boards, B00x, which include RS-232C serial ports. The main modification is then to place the address of the RS-232C port to the same address as used in the transputer-based I/O board.

To interface the user to the controller, a host server based on the personal computer has been designed in C language with the flow chart shown in Fig. 8. The server contains some menu-driven functions including program down-load, parameter tuning, data storage, display, and board test. The program down-load unit is handshaking with the EPROM loader according to the TDS loading protocol [12]. The design of the board-test program is based on the poke and peek functions given by the transputer.

A typical way that the system works is described as follows. At the beginning the host server down-loads the developed application program from the host PC into the root transputer of the controller. The EPROM loader executed by the root transputer then loads and passes the application program to the appropriate target transputer of the controller. Afterwards, the system starts to execute the application program and checks the input ready status of

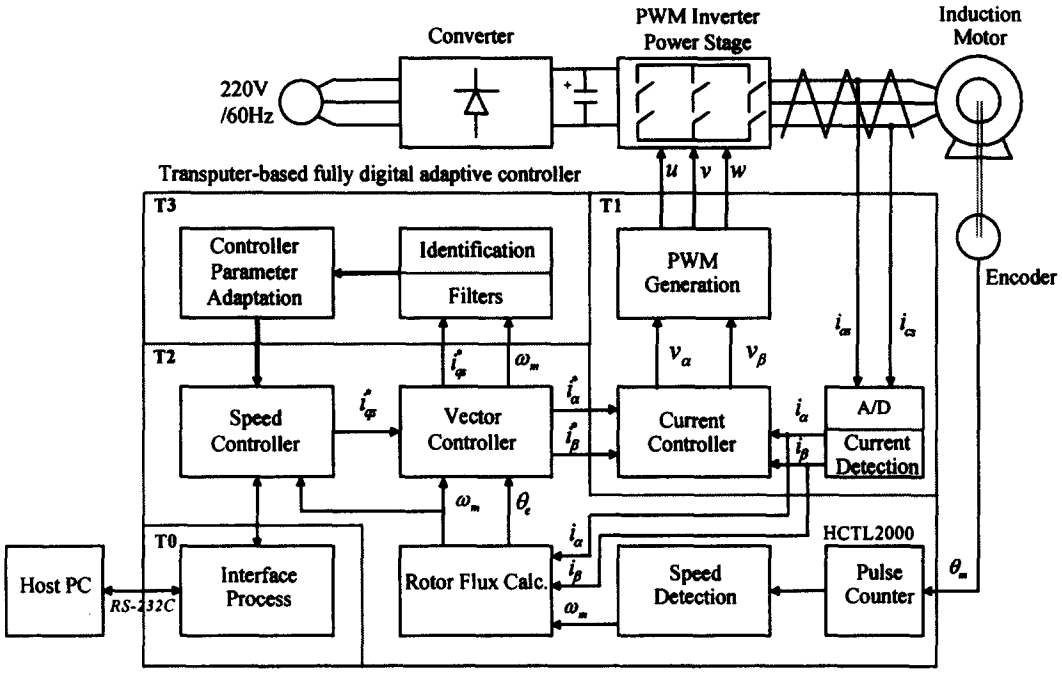


Fig. 9. Block diagram of the transputer-based adaptive control system for an AC induction motor drive.

the RS-232C serial port at each sampling time. If the status is empty then it skips the interface processing and starts another control loop. Otherwise, it signals an acknowledgment to the host PC for the desired interface processing. The user can interactively operate on the host PC to give a desired function.

5. Applications and performance evaluation

The described system has been tested successfully and is currently used in our power electronics and motion control (PEMC) laboratory. An interesting application to verify the performance and demonstrate the configurability of the system has been the implementation of the self-tuning adaptive control of an AC induction motor drive, as shown in Fig. 9, due to its computational complexity and fast I/O requirement [15]. The controller has three nested control loops. The inner loop consists of the current controller task, the middle loop consists of the vector and speed controller tasks, and the outer loop consists of the model identification and controller parameter adaptation tasks. It is important to minimize the execution time of the controller tasks so as to increase the sampling frequency, which in turn enhances the control of the system with larger bandwidth. This could be achieved by parallelizing the control algorithm into smaller tasks and mapping them onto the parallel controller architecture on the

Table 1
A comparison of three parallel controllers in architecture and performance for real-time control

Controller	Architecture types	Building units	Communication types (and rates)	I/O types (and rates)	Matrix multiply
The proposed controller	Loosely coupled multicomputers	Transputer-based parallel-Computing boards and transputer-based I/O boards	Links (20 Mbits/s)	Memory-Mapping (Faster devices: 200 ns or 5 Mbyte/s, slower devices: 400 ns or 2.5 Mbyte/s)	21 μs
SPARTA	Tightly coupled multiprocessors	IBM Hemes processor-based PIE modules and I/O boards	PC bus (PIE to PIE: 12 Mbits/s)	PC bus (PIE to I/O board: 0.4 Mbyte/s)	1.9 μs
CONDOR	Tightly coupled multiprocessors	MC68020-based single board computer, A/Dand D/A boards	VME bus (5K messages/s or 160 Kbits/s)	VME bus	27 μs

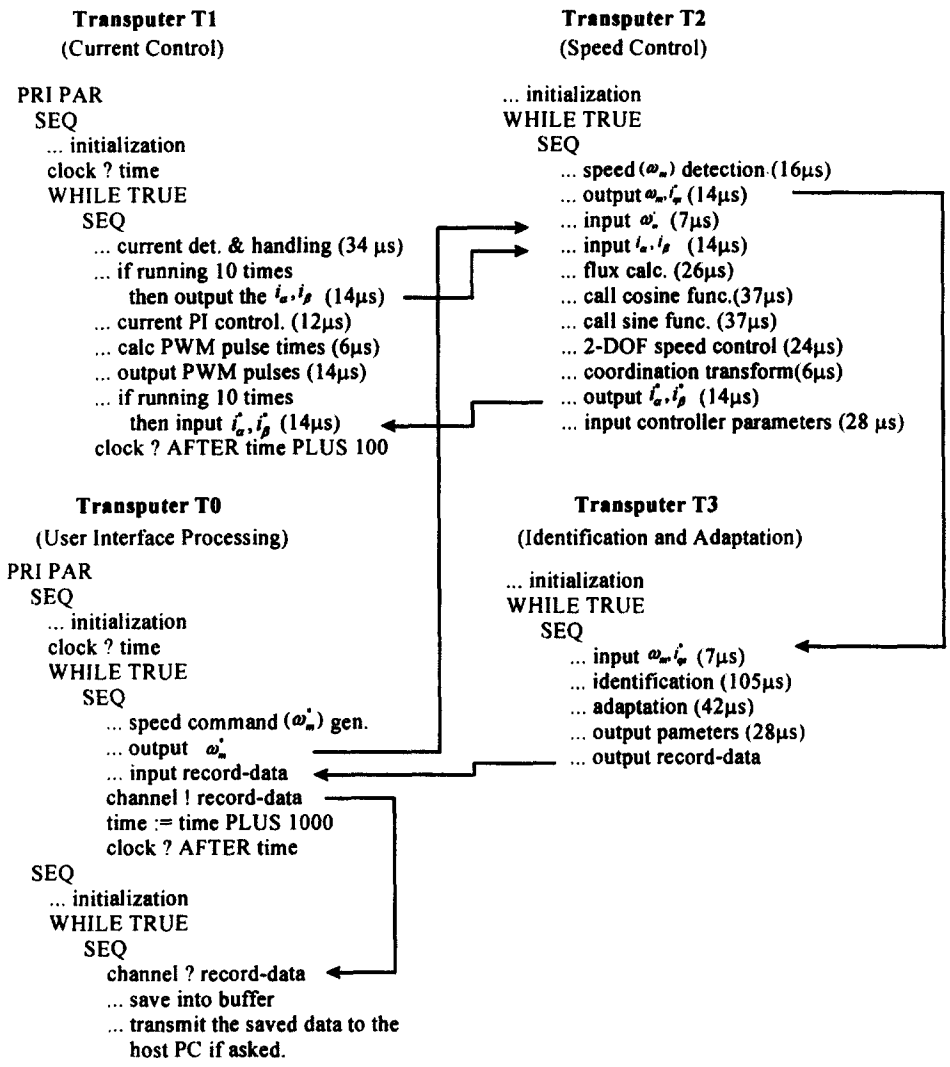


Fig. 10. The Occam programming for the parallel implementation of the adaptive speed control of an induction motor.

assumption that the control algorithm has a sufficient degree of concurrency.

Because the three loops are nested, each loop can be executed concurrently. By direct parallelization of the three nested control loops, the controller tasks can be mapped onto three transputers. One additional transputer is used to implement the user interface process. An Occam language programming for this parallelization is shown in Fig. 10. Using three-transputer mapping for the controller tasks, we have the 'speed-up' of 2.06, which is obtained by dividing the total sequential executing time, 418 μs, using one transputer by the concurrent execution time, 203 μs, using three transputers. In this implementation, the synchronization of the tasks is done by data handshaking, that is, only when the data needed for the task is received from the appropriate communication link can the task proceed. It should be noted that the synchronization frequency between the T1 and T2 processors depends on the sampling times of the current controller task and the speed controller task running on the two processors, respectively. In this application the sampling time on the speed control loop is one ms and that on the current loop is 100 μs. So the synchronization and communication between the two processors takes place once the current controller task has run ten times.

The next demonstrated example is through comparison of the controller in performance and architecture with two existing parallel controllers, SPARTA [16] and CONDOR [17]. The performance evaluation is carried out through the execution time of a fixed-point matrix-vector multiplication, which is given by

$$y = Ax \tag{1}$$

where A is a 2×4 matrix, x is a 4×1 vector, and the result y is a 2×1 vector. An integer data length of 16 bits was used for A, x, and y. The results are illustrated in Table 1. As can be seen from this table, of the three controllers, although the proposed controller is not the best for the arithmetic calculation, it is the fastest in terms of the interprocessor communication and I/O rate. Above all, the

proposed controller is the most expandable in architecture, as previously stated.

6. Conclusions

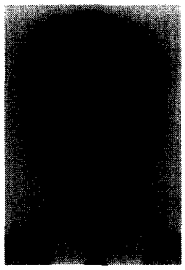
In this paper, the design and implementation of a parallel controller architecture using IMS T800-20 transputer chips for high-performance control applications has been discussed. The system architecture is configurable by using one or more input/output boards, which utilizes a memory mapping bus for various I/O interfaces and one or more parallel computing boards, and hence demonstrates the configurability and modularity. The system can be easily programmed using Occam 2 language [11]. The system's capability has been demonstrated by implementing the parallel adaptive control of an ac induction motor drive. The system also shows its performance and architectural features by comparing the system with two other existing parallel controllers. Although the application work highlights the motor control, this system may be useful for the study and implementation of different complex control algorithms for other high-performance control applications.

Acknowledgements

This research was supported by the National Science Council under Contract NSC79-0404-E-009-028, Taiwan, R.O.C.

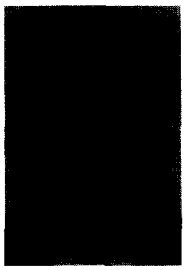
References

[1] F. Harashima, et al., Multimicroprocessor-based control system for quick response induction motor drive, IEEE Trans. Ind. Appl. IA-21 (4) (1985) 602–609.
[2] K. Kubo, et al., A fully digitalized speed regulator using multimicroprocessor system for induction motor drives, IEEE Trans. Ind. Appl. IA-21 (4) (1985) 1001–1008.
[3] G.M. Asher, M. Summer, Parallelism and the transputer for real-time high-performance control of AC induction motors, IEE Proc. 137 (Part D) (No. 4) (1990) 179–188.
[4] R.G. Harley, et al., Real-time issues of transputers in high performance motion control systems, IEEE Trans. Ind. Appl. 29 (2) (1993) 306–312.
[5] H.A. Thompson, P.J. Fleming, Fault-tolerant transputer-based controller configurations for gas-turbine engines, IEE Proc. 137 (Part D) (4) (1990) 253–260.
[6] F. Garcia-Nocetti, et al., Implementation of a transputer-based flight controller, IEE Proc. 137 (Part D) (3) (1990) 130–136.
[7] A.C.J. Stavenuiter, G.T. Reehorst, A.W.P. Bakkers, Transputer control of a flexible robot link, Microprocessors and Microsystems 13 (3) (1989) 227–232.
[8] J.R. Elphick, T. Clarke, S.T. Lawes, A high-performance analogue input/output system for transputer applications, Microprocessors and Microsystems 19 (1) (1995) 3–8.
[9] P. Croll, G. Wilson, Peripheral handling techniques for the transputer, Microprocessors and Microsystems 13 (2) (1989) 124–128.
[10] Transputer Reference Manual, Inmos Prentice-Hall, Hemel Hempstead, UK, 1988.
[11] Occam 2 Reference Manual, Inmos Prentice-Hall, Hemel Hempstead, UK, 1988.
[12] Transputer Development System, Inmos Prentice-Hall, Hemel Hempstead, UK, 1988.
[13] A.M. Tyrrell, J.D. Nicoud, Scheduling and parallel operations on the transputer, Microprocessing and Microprogramming 26 (1989) 175–185.
[14] iAPX 86, 88 User's Manual, Intel Corporation, 1981.
[15] H.J. Wu, M.F. Tsai, Y.Y. Tzou, Transputer-based fully digital adaptive control system for high-performance ac induction motor drives, IEEE Power Electron. Spec. Conf. Rec., (1994) 1250–1256.
[16] J. Ish-Shalom, P. Kazanzides, SPARTA: Multiple signal processors for high-peformance robot control, IEEE Trans. Robot. Autom. 5 (5) (1989) 628–640.
[17] S. Narasimhan, D.M. Siegel, J.M. Hollerbach, CONDOR: An architecture for controlling the Utah-MIT Dexterous Hand, IEEE Trans. Robot. Autom. 5 (5) (1989) 616–627.



Ming-Fa Tsai was born in Taiwan, Republic of China, on April 17, 1958. He received the B.S. and M.S. degrees from the Departmemt of Electronic Engineering and Control Engineering of National Chiao Tung University, Hsinchu, Taiwan, in 1980 and 1990, respectively. In 1980, he joined the Chinese Army for a two-year period service, after which he worked at the Chung Shan Institute of Science and Technology in Lungtan, Taiwan as a research assistant for four years.

From 1986 to 1987, he was a development enginer with the Telestar international company, Taipei, Taiwan, and got a one-year project concerning the manufacture of commercial flight simulator with the Singer Link company, New York, U.S.A. Since 1990, he has been working toward his Ph.D. in the Departmemt of Electronic Engineering at the same university and is engaged in the research of motor control. In 1996, he was a contract engineer with the Sunplus Technology company, Hsinchu, Taiwan, where he was involved in microcontroller ASIC design. His research interests also include the control theory, parallel processing techniques, and power electronic systems.



Ying-Yu Tzou (S'81–M'88) was born in Taiwan, Republic of China, on February 13, 1956. He received the B.S. and M.S. degree in control engineering from the National Chiao Tung University, and the Ph.D. degree in electrical engineering from the Institute of Electronics Engineering of National Chiao Tung University in 1978, 1983, and 1987, respectively.

During 1980–1981 he was with the Electronic Research and Service Organization (ERSO) of the Industry Technology Research Institute (ITRI) as a design engineer in the control system department. During 1983–1986 he was with Microtek Automation, Inc., as a project manager for the development of a Computer Numerical Controller (CNC) for machine tools. He is currently an Associate Professor of the Department of Control Engineering of the National Chiao Tung University, and also serves as an industrial consultant for many local power electronics and automation companies. He was the Director of the Institute of Control Engineering during 1992–1994. His special interests now are sensorless ac drives, intelligent UPS, FPGA based control ICs for motor drives, and DSP applications in power electronics and motion control.