CHAPTER II

ARCHITECTURE OF THE SHARED-MEMORY MULTIPLE-MICROPROCESSOR SYSTEMS

This chapter presents a design technique for implementing the shared-memory multiple-microprocessor system. The proposed system uses the common bus interconnection network. This requires arbitration to ensure that only one microprocessor at any time can put information into the bus and shared memory.

Arbiters may be realized in either software or hardware. Software arbiters normally require many clock cycles to resolve an access request whereas a hardware arbiter can be designed to operate in a few clock cycles. Therefore , a flexible arbiter proposed in [22] which employs simple hardware is presented.

In addition to the arbiter design, this chapter also presents two examples of shared-memory multiple-microprocessor systems implemented from the proposed system.

2.1 Arbiter System Configuration [22].

The proposed arbiter can coordinate the multiple-microprocessor system by using 4 Z-80 microprocessors, or any other microprocessor that has a WAIT input and a FETCH state indicator output. Shared random access memory (RAM) is used as a data mailbox. The priority scheme being used is on a Round Robin scheme but can be modified to any sophisticated priority scheme. The arbiter is divided into two main parts,the scanner and the controller,as shown in Fig.2.1.

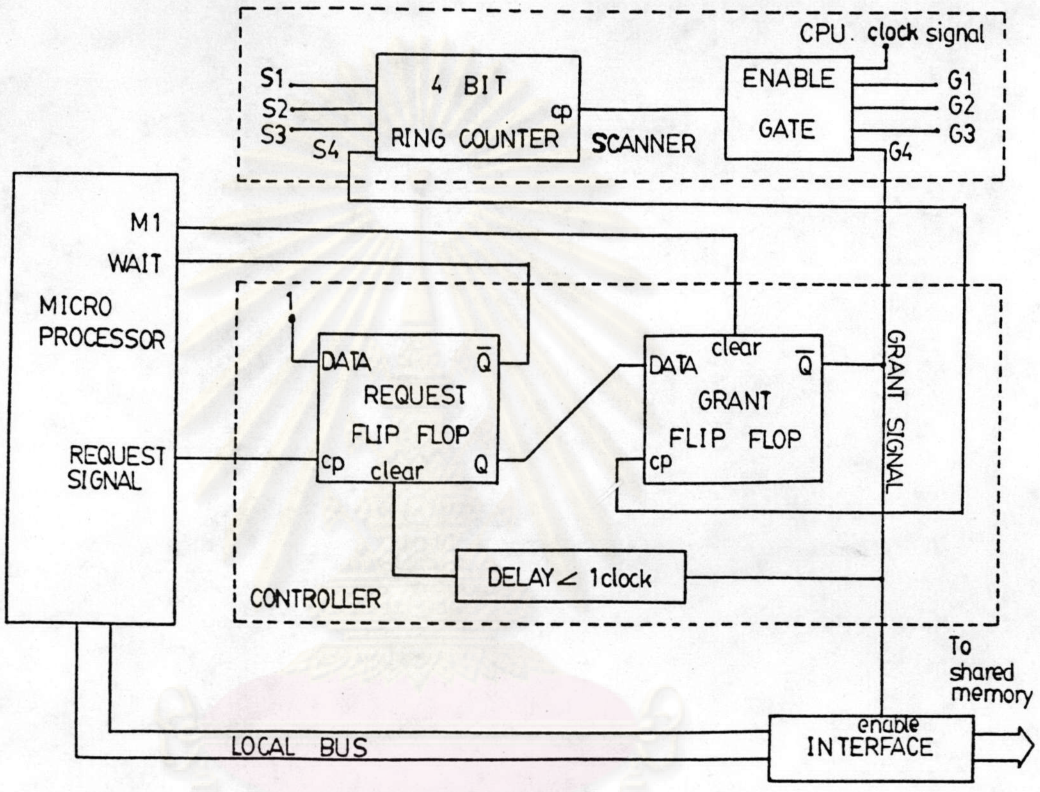

Fig 2.1 Scanner and controller block diagram.

The scanner schedules the shared memory accessing for each microprocessor by issuing only one active scanning signal (Si) at a given time. The microprocessor scanned will be allowed to access shared memory.

Each scanning signal is derived from the output of a ring counter for the Round Robin priority rule. If another priority rule is required, the ring counter should be replaced by some other pattern generator. The ring counter is driven by the CPU clock through an ENABLE gate. The ENABLE gate will disable the clock drive when any of the microprocessors is granted access to the shared memory and reenable the clock drive immediately after the microprocessor completes the shared memory operation. Only one scanner is required.

Each microprocessor is assigned one controller. Each controller will receive a REQUEST signal (Ri) and issue a GRANT signal (Gi) or a WAIT signal back to the microprocessor according to a command from the scanner.

Whenever a microprocessor requires the use of the shared memory, it sends a request to the Request flip-flop (Fig.2.2). The Request flip-flop is set by the REQUEST signal (Ri) and then, returns a WAIT signal to the microprocessor. The microprocessor, after having detected the WAIT signal, enters the WAIT state. The GRANT signal will be generated when the Grant flip-flop is clocked by the scanning signal. The interface between the local microprocessor bus and the shared memory bus is enabled by a GRANT signal. At the same time, the GRANT signal is sent to the ENABLE gate in the scanner, which disables the ring counter. This will prevent other microprocessors from accessing the shared memory. After one clock delay to allow memory address setting, the microprocessor is enabled and starts its access to the shared memory.

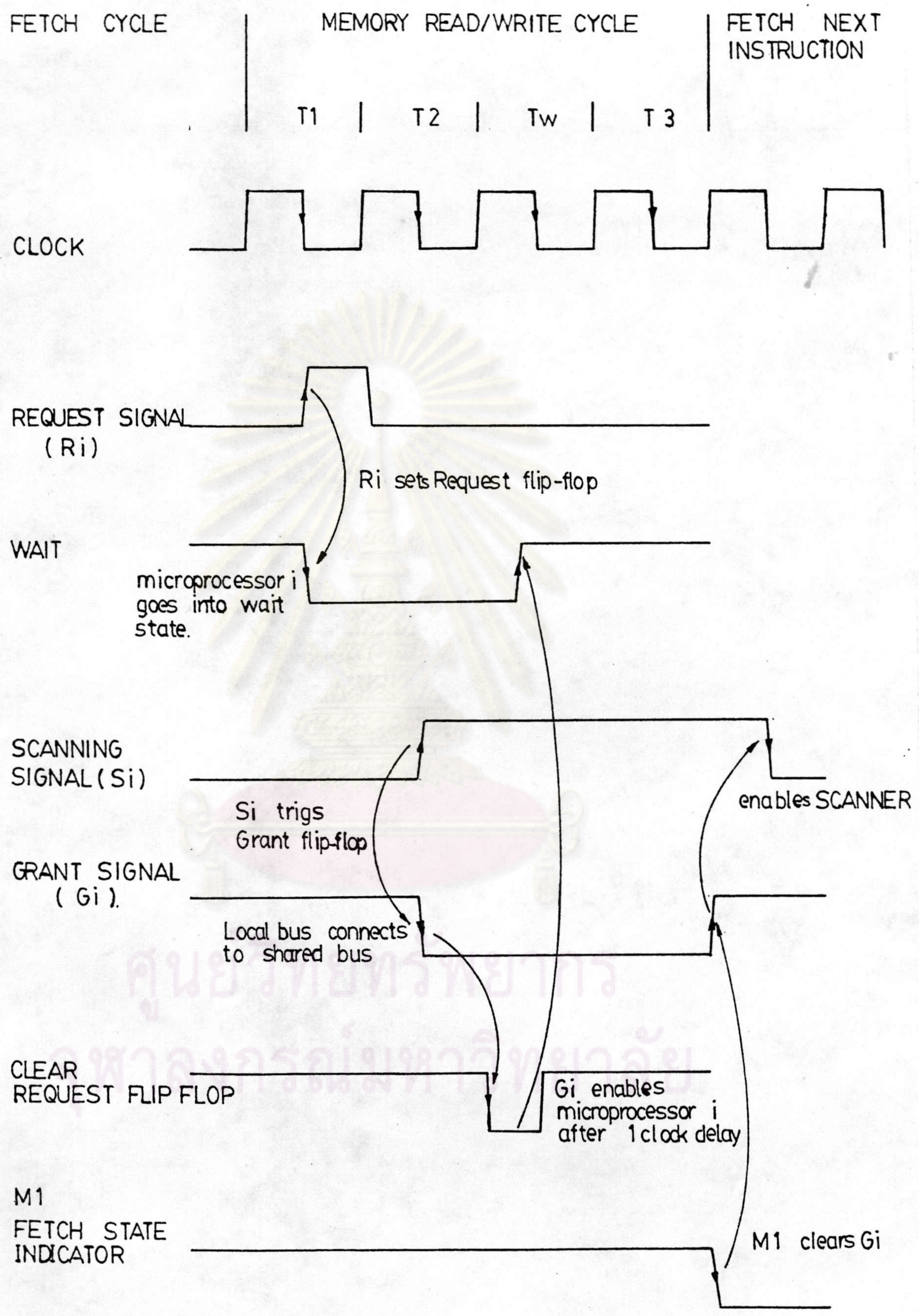

Fig 2.2 Arbiter timing diagram.

After the microprocessor has completed its access to the shared memory and begins to fetch the next instruction, it will issue a FETCH state indicator signal (Ml). The Ml signal will terminate the GRANT signal by clearing the Grant flip-flop and returning to the initial state. The ENABLE gate then becomes active and emits the clock drive to the ring counter. The next microprocessor will be allowed to access the shared memory.

The proposed technique is compared with other previous techniques in Fig. 2.3. Although the data transfer rate is lower than some other techniques, the proposed technique still has its own advantageous features as follows.

1) It can be used with a higher number of microprocessors.

2) It is more flexible since it can be used with almost all other microprocessors in the market.

3) System implementation is easy, requiring only a simple hardware.

4) Data transfer rate is moderate, e.g., in a system of 4 microprocessors working at a 2-MHz clock signal, the average data transfer rate is 87 kbytes/s.

5) Since all conflicts are solved by hardware and each microprocessor sees the shared memory as logical extensions of its private memory,communication software for interprocessor communication is virtually eliminated.

A block diagram of the shared-memory multiple-microprocessor system implemented from the proposed arbiter is shown in Fig. 2.4 .

2.2 Examples of Multiple-Microprocessor System with Shared Memory.

Two multiple-microprocessor systems implemented from the proposed arbiter are presented. The first system uses the

Method	No of μP using shared memory	Average data transfer rate (Bytes/sec)	No of additional device	Remark
DMA	Unlimit	more than 1 MBytes	1-2 LSI	requires specific DMA controller for each uP.
TWO-PORT RAM	2	142.8 KBytes	1-2 LSI	used for small amount of data transfer and expensive.
PETRIU	4	86 KBytes	5 TTL	very difficult to implement and cannot be used with sophisticated priority.
POLCZYNSKI	8	125 KBytes	4 TTL	can be used only with specific microprocessor that FETCH state equal to EXECUTE state.
LOEWER	2	91 KBytes	9 TTL	can be used only with two Z-80.
HOJBERG	6	90 KBytes	7 TTL	requires 2 external high speed clocks.
The proposed method	6	87 KBytes	4 TTL	can be used with any uP that has FETCH state indicator and WAIT state.

Fig 2.3 Comparison between various shared memory methods.
(The average data transfer rate is calculated from normal operation condition while the calculation in [22] is based on heavy traffic condition.)

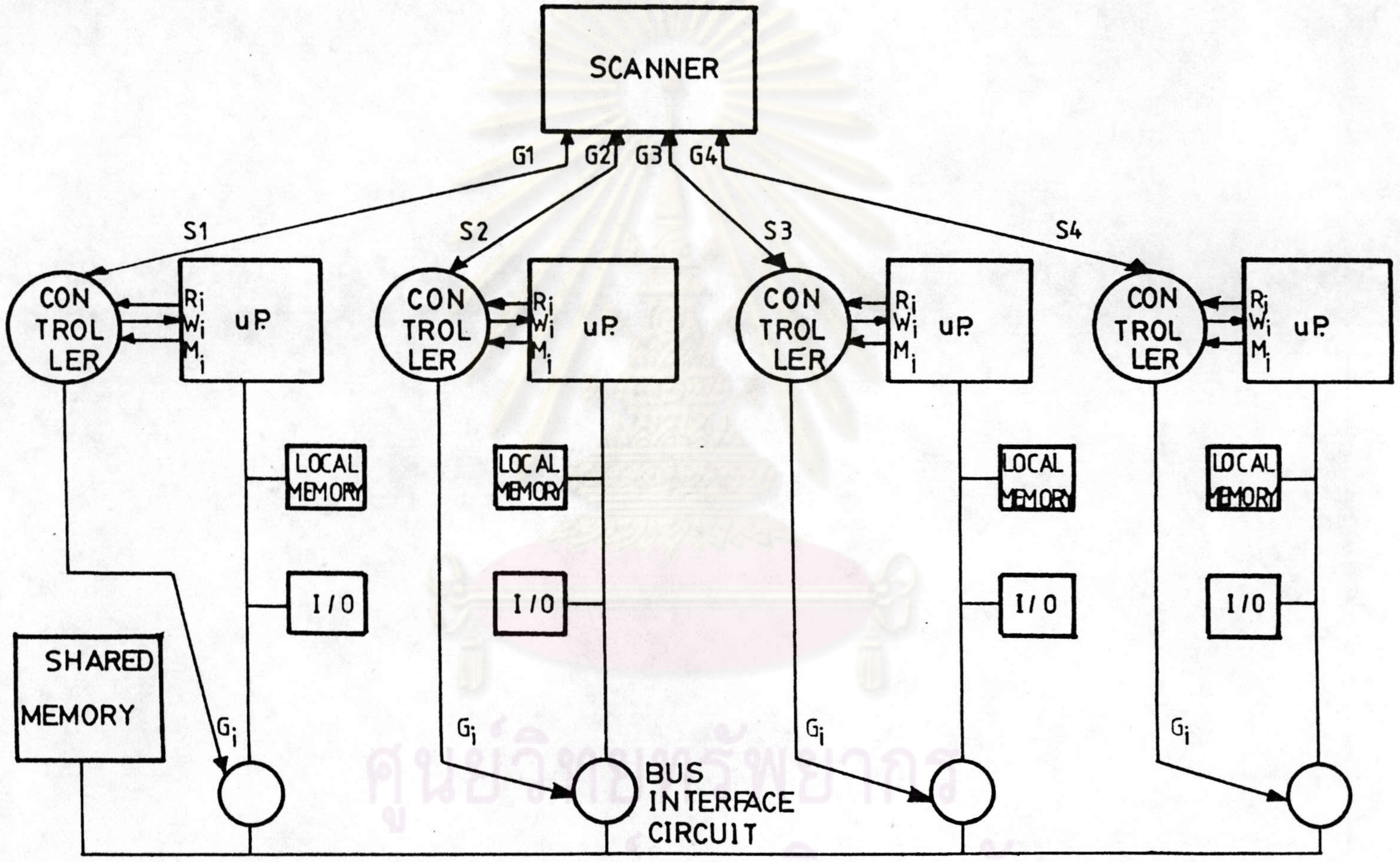

Fig 2.4 The shared memory multiple-microprocessor system block diagram.

multiple-microprocessor system as front-end communication subsystem for host computer. The Tri-modular redundant multiprocessor for real-time application is presented as the second example.

2.2.1 The Multiple-Microprocessor Based Front-End Communication Subsystem [23].

To exemplify an application of the system, the shared memory multiple-microprocessor system is used as the front-end communication subsystem for the host computer. The front end subsystem consists of 3 microprocessors. Two microprocessors are assigned as the REMOTE LINK UNIT (RLU) managing data communication between the front end subsystem and 32 remote data acquisition terminals. Another microprocessor is assigned as the HOST INTERFACE UNIT (HIU) managing communication with the host computer (Fig. 2.5).

The throughput of the single microprocessor is small when compared to the mainframe host computer. The communication task must, therefore, be distributed to various microprocessors. The interprocessor communication is done through the shared memory by using the proposed technique. This eliminates communication software between each microprocessor and simplifies programming as each microprocessor sees the shared memory as logical extensions of its local memory.

Each remote terminal is assigned to collect analog data from 64 sensors installed at each site. Each analog data is converted to 8 bits digital data by high speed analog to digital converter. The remote terminals send the data collected to the RLU through multiple serial data transmission lines via SDLC protocol. One RLU manages data communication with 16 remote terminals, i.e., RLU-1 collects data from terminal 1-16 and RLU-2 collects data from the terminal 17-32. The RLU sequentially polls each terminal, checkes the

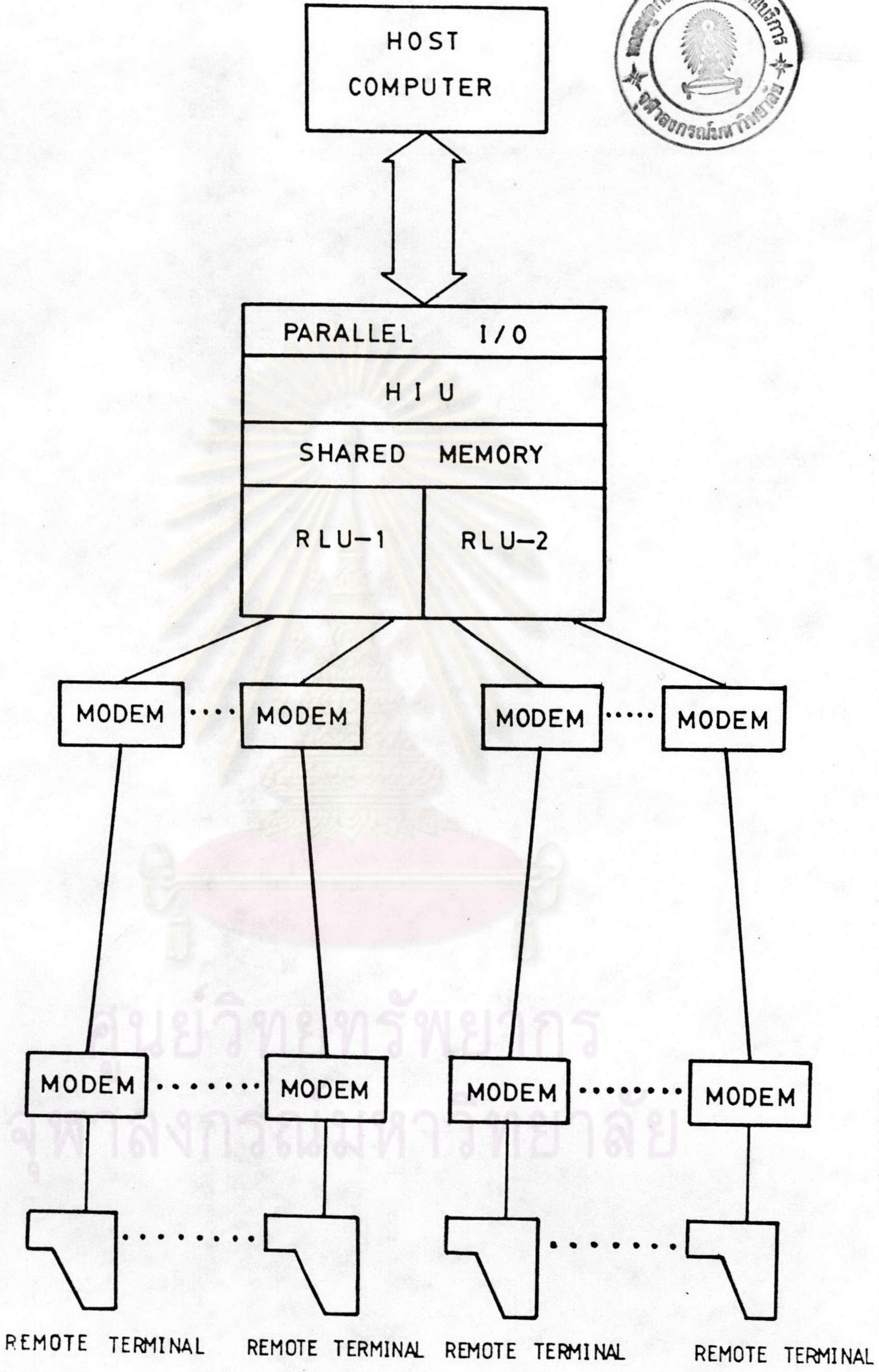

Fig 2.5 The shared memory multiple-microprocessor system as a front-end communication subsystem.

validity of the data received, and then stores the data in the shared memory.

The HIU groups the data from 32 terminals, stores in the shared memory, and sends the data block to the host computer upon requested. Data transmission between HIU and host computer is done through a high-speed parallel bus.

The technique then becomes a possible inexpensive solution to the real-time control problem, especially for a system that requires a faster response time.

2.2.2 The Tri-Modular Redundant Multiprocessor [24].

The fault tolerant real-time systems involve redundant components so as to tolerate a fault by reconfiguring, for masking the faulty element . A tri-module multiprocessor system provides the necessary features for fault tolerance [25]. Such a system is capable of identifying a faulty element (by software/hardware voting), segregating it by reconfiguration, and operating in a degraded mode with the remaining elements .

This section presents the development of a multiprocessor-based TMR fault tolerant controller used to control a mobile trolley. The fault tolerance is achieved employing identical, modular software running on three identical processors. The system is fully reconfigurable and is degradable to a two-processor state in the event of occurrence of a permanent fault. Transient faults are tolerated employing a retry mechanism. This is possible irrespective of the system being in a three-processor or a two-processor configuration. In a three-processor configuration, the system is fault tolerant and in two-processor configuration it is fault detecting. The process of removal of a faulty processor and its reinstallation after repairs does not need any reinitialization of the system and these changes remain transparent to the application.

The system has been built using three Z-80 microcomputer boards communicating with one another through a shared global memory (GM). A time efficient bus arbiter of the type proposed in this chapter allocates the GM to each microcomputer on a Round Robin basis. The complex hardware-like microcomputer board and the associated circuitry are implemented using commercial components and are triplicated. The hardware of GM, bus arbiter, and multiplexing logic are implemented using components with high reliability, tested as required during operation, and are not triplicated to keep implementation simple for the present work.

The system (Fig. 2.6) is implemented using three Z-80 based microcomputer boards. Each microcomputer board has a programmable peripheral interface (8255), a universal synchronous/asynchronous receiver/transmitter (8251), a programmable interval timer (8253), 16K bytes of RAM, and 8 K bytes of ROM. These microcomputers communicate with one another through a common GM. A time efficient bus arbiter scheme has been utilized for this purpose. A microcomputer generates a request to the bus arbiter whenever it needs to read/write a data in GM. The bus arbiter allocates the GM to one of the requesting microcomputers on a round robin basis. The system has both analog and digital data I/O channels for interfacing.

The system has been used to control the motion of a three-wheeled trolley that incorporates two step motors, one for the drive and the other for steering purposes, and employs a feedback loop with optical sensors (Fig. 2.6). The control commands of driving sequences for these motors are provided by all three microcomputer boards. The control commands issued by one of the healthy microcomputers are chosen using 3:1 multiplexing logic.

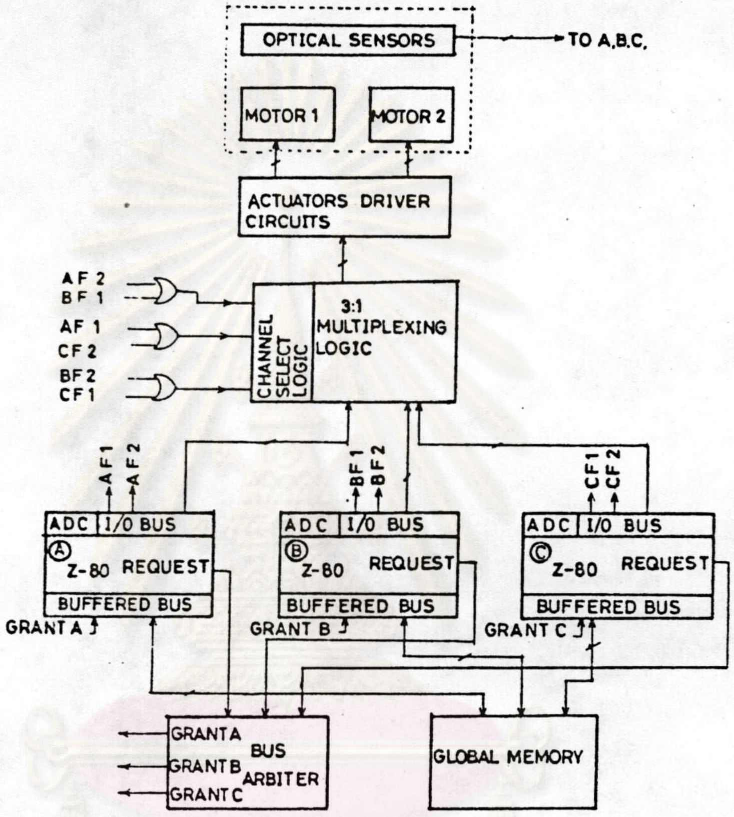

Fig 2.6 The tri-modular redundant multiprocessor [24].