M.Tech credit seminar report, Electronic Systems Group, EE Dept, IIT Bombay,
submitted November 2002

Hardware Configurations for DSP-based Real Time Simulators

Arup Chakraborty (02307015)
Supervisor: Prof. Mukul Chandorkar

Abstract

This report discusses the hardware interconnection mechanisms of the multiple processors used for real time simulations, Hardware-in-Loop simulation for example. Real-time simulations are required for testing systems under real working conditions. Hardware-in-Loop simulation is an example of a such a simulation, in which the input and output behaviour of a process is simulated in real time and used for testing embedded controllers. Real-time simulations are computationally intensive and often require multiple DSPs. Traditional approaches of tightly coupled and loosely coupled multiprocessing systems are first discussed. Then, a survey of some practical real time multiprocessor systems is presented. The techniques used by these practical systems to improve upon both traditional approaches are also discussed.

1 Introduction

Simulation is the reproduction of reality and all its complexity through the design and execution of a model of the reality. When this simulation is to be done within the time constraints imposed by the reality itself, it is known real time simulation. Let us consider that we have a physical system, and we want to test its behaviour under different input conditions. One approach to do so is to develop a mathematical model of the system and to execute this model on a computer and hence observe the system behaviour. This approach is better than testing the physical system itself in the sense that one can apply all ranges of input to gain an insight to the system without risking any hazard on the system and the user. In many cases it is simply more economical to test using a simulator instead of the actual system.

All the events and processes taking place inside the actual system are bound by certain laws governing it. Let T be a time interval in which a certain process is completed in the actual system. A real time simulator, simulating this process, must accept inputs, perform the computations necessary and keep the output ready within a time less than or equal to T, the simulated time. This time interval between reading the inputs and producing the corresponding outputs is known as time step, which is a critical factor. Real time simulations are useful as interaction of the system under the study, for example functioning of a control and protection system with the outside world can be analyzed critically with respect to time. Also statistical studies requiring thousands of operations can be carried without wear and tear of the system and in much less time.

Since simulation requires computations based on the mathematical model of the system and for real time simulation these have to be done within specific time limits, real time simulators must have large computational power, especially for complex systems. Solving differential equations of a somewhat complex system within a small time step of

50 microseconds with high precision arithmetic requires computational power of range of gigaflops. A normal microprocessor cannot be used for this purpose since they have unacceptably large bus and peripheral device latencies, are unable to perform high precision floating point arithmetic and do not have hard and fast time constraints.

At this point Digital Signal Processors come into picture. DSPs are necessary because

- they provide lot of computational power
- operate on continuous flow of data in real time in a deterministic manner
- can carryout very fast I/O operations and data shifts between various parts of the system

However simulation of some complex systems for example a spacecraft or an aeroplane demand such a great performance that single DSP-based systems are unable to comply with. In those cases one has to go for multi-DSP system.

But apart from the increased computational capability, there are other reasons to go for multiprocessing for real time applications . One of them is that in some cases smaller processors are cheaper per unit power and therefore a multiple processor based system will provide power more cheaply than a single processor of equivalent power.

For systems which are required to respond to various kinds of events in real time, it is better to dedicate different processors for handling different events, than using a single processor. In very sophisticated systems, where these different tasks are very specialized in nature, specialized heterogeneous processors may be used. In such system, each processor behaves as if the rest of the system is a smart peripheral device.

Another important motivation behind the use of multiprocessing systems is that they can be designed to be scalable. This means that the performance of a multi-processor based real-time system can be enhanced simply by adding more processors to it. That way it will be more flexible and cost effective than upgrading to faster single processor. This means that it gives the user the flexibility of trading off between the number of processors required for his application and the cost involved.

In earlier days multiple processors were used so as to ensure the functioning of the system even if one processor fails. But nowadays with the increased reliability of the processors this point is no longer significant.

There is one special kind of real time simulation known as Hardware-in-Loop simulation, explained in the next section. The aim of this work is to make a survey of hardware configurations in which these multiple Digital Signal Processors can be connected for real time simulation, especially Hardware-in-Loop simulation.

1.1 Hardware-in-Loop Simulation

Embedded controllers are being used for the control of various kinds of systems which can be as complex as unmanned aircrafts and automobiles. Such embedded systems operate in safety critical situations. These embedded systems tend to consist of multiple controllers interacting with each other. There may be numerous inputs to such a control system and as also numerous outputs. Hence detailed testing of such embedded control system hardware and software needs to be carried out to ensure fail-proof high performance of the product. The response of the control system to emergency situations and its behaviour in extreme regions of operations must be determined accurately. And such testing contributes to a

major portion of development cycle and cost. It allows the embedded controller to be tested under various real working loads and conditions.

Hardware-in-Loop Simulator (HILS) is tool that is used for testing such embedded systems in real time. The system (for example an aeroplane or an automobile) to be controlled is simulated in real time and interfaced with the controller under test. In other words, HILS accepts outputs from the controller and drive inputs to it, and behaves as the world external to the controller in real time.

Let us take an example of the an autopilot controller[4]. Typical inputs to an autopilot controller are current airspeed, pitch angle (the angle of the nose of the plane with respect to horizontal), pitch rate (the rate at which the nose is diving/rising), gravitational force, etc. These are acquired by the controller using some sensors. The control law implemented in the software calculates the required deflection of the elevator and acts on an actuator. While testing this controller, HIL simulator is designed to take the outputs of the controller and computes the resultant airspeed, pitch angle and pitch rate, according to the physical laws governing the motion of the aeroplane and directly feeds the inputs of the controller. The whole process must be done as fast as the real aeroplane does. This eliminates the requirement of a real plant (in this case, the aeroplane) for testing of the controller. It is to be noted unlike other kinds of simulation here the system under test is not simulated but its external environment.

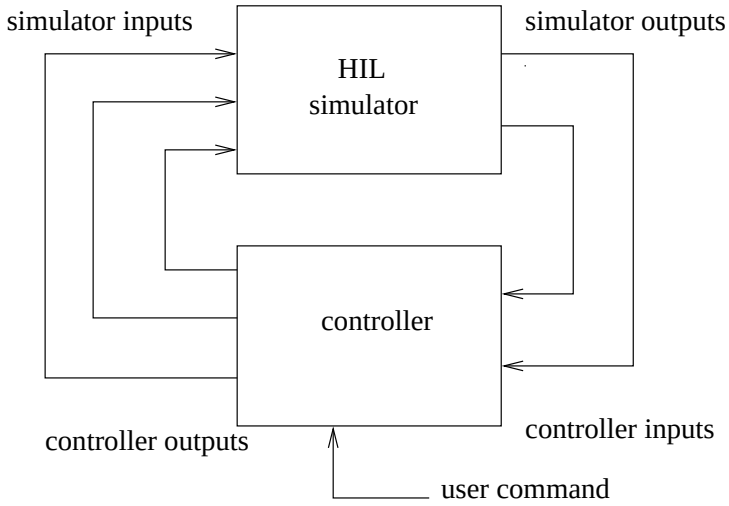

Figure 1: A controller connected with Hardware-in-Loop Simulator (adapted from[4])

The motivations behind using HILS are various[5].

- Any alternative testing procedure may be very difficult and in some cases, impossible. For example, suppose a control system for a satellite is to be tested. A satellite works under weightless condition. So instead of trying to simulate such a condition on earth, it is much easier to simulate the satellite behaviour based on its mathematical model and interface the simulator with the controller.
- Control and maintenance of the testing environment may be difficult and too costly.
- Often safety considerations disallow any kind of direct testing on the actual system.

- In some cases HILS is simply the most economical testing procedure.

Another advantage that HILS offers over workstation/PC-based simulation is that the control software runs on the actual hardware which would be built into the end-product, and not on a workstation.

Earlier, in the absence of HILS, in the development of the product with inbuilt control system (for example a car with a control system to correct skid on icy road condition), a prototype of the product was first constructed , then the control system was built and whole unit was tested. If the performance was not desirable, the design was modified and the whole procedure was repeated. However currently using HILS, the controller hardware and software need to be developed only, and not a full prototype of the end-product and the rest of it is simulated. The result is used to fine tune the controller. Thus development of the controller and the system to be controlled goes on parallelly. When the prototype of the end-product with the controller is ready, it is right the first time. This reduces the development cycle significantly[13].

The rate of iteration of the HILS must be far greater than that of the controller under test to the order of 5 to 10 times[4]. The inputs and outputs of the HILS are respectively outputs and inputs of the the controller. But the resolution of the signals for a typical HILS are about 3 to 4 times that of controller. The range of the HILS signals are slightly higher. These considerations are necessary for ensuring real time high performance of the simulator.

Thus it is evident that HILS is a specialized real time system which is to be specially designed taking into consideration the features and characteristics of the the system to be simulated. However since this is a real time systems, principles of other real time systems will also be applicable here.

1.2 Structure of HILS

A structure of a typical Hardware-in-Loop simulator can be divided into the following general components[4,6].

- The input hardware required for acquisition of the signal generated by the controller.
- The software responsible for handling the input hardware, reading the input signals and keeping them ready for the following stage of the software
- The software part which represents the model of the simulated system. It accepts inputs from input software , computes the outputs according to the mathematical model and characteristics of the system of the system and supplies these output values to last stage of the software.
- The last stage of the software is responsible for driving the output peripheral devices and generation of the output signals in a form acceptable to the controller
- The output hardware consists of these output peripheral devices which are interfaced with the controller.

The output and input signals may be analog in which case the input and output blocks require A/D and D/A converters. The signals can also be in digital form and serial and parallel communication can be used. In some cases HIL simulator are designed with the input, output stages as reconfigurable. This is required when the same design is to be used

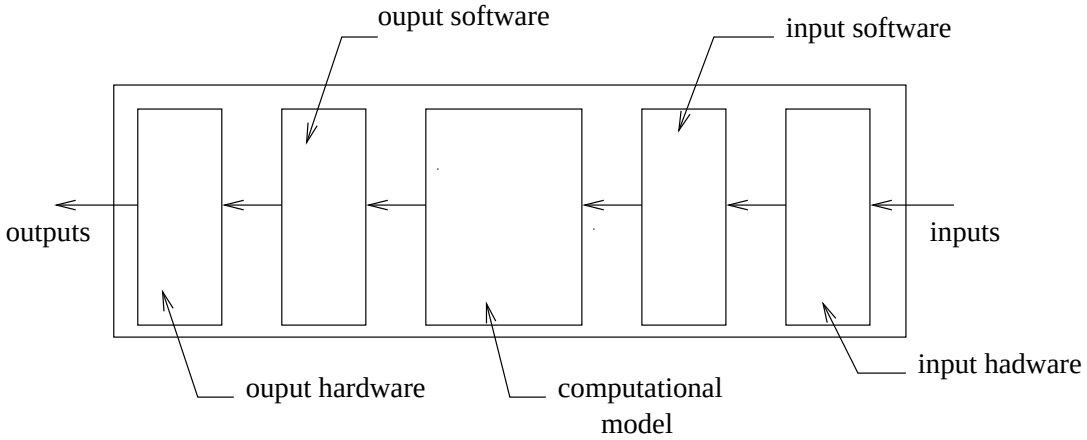

Figure 2: Components of a Hardware-in-Loop Simulator[4]

to test different kinds of controllers with different kinds of inputs and outputs. Thus it is worth money and time to be able to reconfigure the inputs and output blocks accordingly. This is usually done by using FPGAs in input/output stages and table driven software.

Another aspect of HILS design is that, sometimes instead of simulating a sensor or a actuator, it is physically put between the controller and simulator. This is because the model for sensors/actuators may be too complex to be simulated.

2 Multiprocessing Systems

As mentioned earlier, multiprocessing systems are absolutely necessary for complex real time systems and hence for most HIL simulators. To understand the hardware architecture that can be used for HIL simulator application, a look into general multiprocessing system is necessary.

Computing systems can be classified as[2]

1. Single Instruction Single Data(SISD) system, in which case at a time a single instruction stream operates over a single data stream. This does not require multiprocessors.

2. Single Instruction Multiple Data(SIMD) system. A single instruction stream operates on a multiple data stream. Individual processors do not have large local memory. There is a master processor which feeds the other processors with the program instructions and data and execute them in lock step.

3. Multiple Instruction Multiple Data(MIMD) system. Many instruction streams operate over multiple data streams simultaneously. Each processor executes different program and act on different data set.

MIMD systems can further be classified into Tightly Coupled Systems and Loosely Coupled Systems.

Tightly Coupled Systems consist of several multiprocessors having access to a common memory space. This common memory can be used for communication between the processors and also accessing the program instructions and data. Tightly coupled systems are of following types[1]:

1. shared bus based system
2. multiported memory based system
3. bus window based system
4. crossbar switch based system

In shared bus system the processors are connected to the same bus which is also connected to the memory and I/O. The bus is used by the individual processors to access the memory and the input/output devices. The bus may include lines for interrupt processing. The devices connected to a bus can be classified as master or slave. Master devices can request the bus for communication with other devices. Slaves are those which cannot communicate on their own and are accessed by the masters. A processor is always a master while a memory is always a slave. In a shared memory architecture, at a time, there can be only one master. Hence comes the requirement of resolving bus requests coming from more than one devices

A counterpart of the shared bus system is multiported memory in exchange of the common bus. This eliminates the need for bus arbitration but calls for the design of multiported memory. Dual-ported memory is commonly used for multiprocessing systems. Here two processors can access the same memory space at the same time, timing conflict, if any, being resolved by the device itself. This provides a very fast way of transferring data between the processors and also sharing common code. Multiported memory can be used with memory segmentation and some segments may be kept with read/write protection. An individual processor can use such a segment to keep data, which is not to be modified by other processors.

In a bus window multiprocessing system, a part of the memory of one processor is mapped to that of the other. For a specified address range, memory access request of one processor is transferred to memory device connected to other processor. This means each processor can read/write into a part of the memory of the other processors. Data can be transferred between processors very fast. This mechanism can be extended for accessing I/O devices as well. However this leads to a loss in local memory space for each processor.

Crossbar Switch connects the processors with other system resources like memory and I/O devices using multiple switches in the matrix form. The multiple processors can still share the memory modules provided no two processors try to access the same device at the same time. The switching establishes temporary links between the memory or I/O and processors.

Before proceeding further in discussing the advantages and disadvantages of the above system it is necessary to consider certain issues relating the performance of the multiprocessor systems. They are

- Overhead. It is a measure of the the time and processor cycles wasted to establish communication between different system components before processing the user application.
- Latency. It is the time elapsed between issue of a control command and initiation of the appropriate response to it.
- Skew. It is a measure of the time interval between the occurrence of the events in the different processors when they are intended to occur concurrently.

- Determinism is the ability of the system to respond with a *consistent* and *predictable* delay to the input.

The real time systems are intended to have low overhead, latency, skew and highly deterministic.

In a tightly coupled system it is possible for very high speed data transfer between processors. Large blocks of data can be transferred with almost no software overhead. This allows for a tight control of the system resources. On the other hand, at a time, only one processor can get the control of the bus or memory. Memory is required by the processor for both fetching the program instructions and inter-processor communication. As the number of processors go on increasing, bus requests at any given time will also increase correspondingly. This limits the improvement in system performance gained from using multiple processors. Thus this system works well with smaller multiprocessor systems but is not very scalable.

A workaround can be achieved by arranging for each processor a local memory to store private data and code and using the shared memory only for inter-processor communication. This attempts to improve system performance by decreasing the overall bus requests. Tightly coupled systems permit only limited physical separation between the processors.

In a Loosely Coupled System each processor node represents an autonomous system with its own memory and I/O subsystem. They do not share any common address space. Hence there is no need for memory or I/O access conflict resolution. The processors are connected by means of some input/output devices. These connections can be through parallel or serial links and data is exchanged using some communication protocol. Processors are interconnected using several schemes as loop, cubes etc.

The communication is very slow with respect to tightly coupled systems due to software overhead involved. The CPU cycles are lost due to these overhead. Maintaining tight control of the system resources is very difficult. As a result such systems lack determinism. Hence they are not suitable for real time application. Another disadvantage of this kind of subsystems is that each processor is connected to expensive system resources as memory and I/O, which remain idle most of the time. Thus commercially made systems are generally tightly coupled. However, a large number of processors can be connected with a large physical separation in loosely coupled system.

3 Real Time Multiprocessor Systems in Use

It is evident that both tightly coupled system and loosely coupled systems has their advantages and disadvantages. Actual implemented systems are between these two extreme ends of the spectrum. While designing, attempt is made to keep the best of both and avoid their disadvantages. There are some systems which are designed and implemented in the light of this philosophy. A survey of some such systems is given below.

3.1 Hardware Coordinated Multiprocessor System

One approach that is used to get around problems of both kinds of systems is to make the control and communication between the processors hardware-coordinated. Ixthos,Inc has developed such a system[7]. Hardware devices are used for providing a dedicated path for all coordination activities. This allows deterministic behaviour of the system with minimum latency and skew.

The hardware control circuit known as MPR (MultiProcessor Resources) is implemented in each circuit board . They consist of set of state registers known as Coherent State Registers(CSRs) which store the state bits. These bits as well other MPR signals like clock signal, are connected from one processor board to another through what is known as MPR Connection (MPC). MPC is a dedicated bus for command and control that provides the hardware signals but does not act as a general purpose data bus. MPR distributes copies of CSRs to different processor board through MPC. This makes a deterministic update of any local change of content of these registers throughout the whole system within a very short time (within 500 ns). There are certain monitor circuits in each board that watches these constantly updated CSRs and may interrupt the processor when some action is necessary.

It is to be noted that apart from the MPC, there may be a general purpose data bus with shared memory. This approach improves the system performance by off loading the control activities from the general data bus.

Software overhead in communication, latency, and skew are very low in hardware coordinated multiprocessor systems. Adding more processors to the system, i.e., scaling does not degrade system performance. However the disadvantage is that extra hardware is required which makes the system expensive.

3.2 The Concurrent Computer Corporation's Approach

In recent times, there has been a tendency towards implementing large multiprocessor systems by connecting several single board systems instead of having a single board for all processors. This is because single board system is less expensive and this approach is flexible regarding the number of processors required for a particular application[12].

It is to be noted that each of these single board systems may itself consist of more than one processor in a tightly coupled manner. Each board have memory and I/O devices and is a autonomous system by itself. It can run a user's task by itself using single or multiple processors available on the board. The data transfers within a board are very fast because of its tightly coupled nature. On-board local bus contention is not significant since number of processors on board is limited to two or three.

However for the overall functioning of the multiprocessor systems, the tasks running on each board must properly communicate with each other and must be synchronized. For this, the boards are connected on a common bus. The VMEbus is a popular standard bus used for this purpose. The features of VMEbus are explained later on.

A range of communication facilities are provided for inter-board communication. The simplest of them is shared memory between the boards. A part of memory space of one board can be accessed by another by means of programmed I/O through the VMEbus. This allows the task executing on one board to communicate with another task running on a second board by reading/writing in the physical memory of the second board and vice versa. The size of memory can be pre-determined or varied between two extreme limits. Large amount of data can be transferred very fast between boards though direct memory access.

VMEbus is also used as a networking medium. Standard networking protocols can be used for data communication, however with much less efficiency.

For real time performance and determinism, it is essential to provide real time synchronization between the processes running on different boards. For example a task running on one board is required to send a interrupt to a task running on another board signaling

that some data is ready in the shared memory region or to begin a predefined action.

A process on one board can interrupt a process on another board by means of what is known as mailbox interrupts. These interrupts are generated locally when specific on-board registers are accessed by a remote board.

The VMEbus itself has several interrupt lines, which may be used for control and communication.

A VME clock board can be connected to the system, which can be directly read by different processes to coordinate occurrence of events through out the system.

Additional hardware module can be plugged into each board and the modules of each board are connected by another shared bus. Such modules as also the whole multiprocessor systems using them have been developed by Concurrent Computer Corporation[8]. Such modules are called Real- Time Clock and Interrupt Module(RCIM). Block diagram of such a system is shown in figure 3. The name summarizes its functions. RCIMs are connected to each other by a standard bus, namely PCI bus. Each RCIM consists of a clock that is synchronized with the clocks on all other RCIMs, four programmable real-time clocks that can also generate interrupts, and four input and four output interrupts, which are edge-triggered . These interrupts can also be configured as distributed which means that an interrupt occurring on one board can cause interrupts to occur simultaneously on all other boards. All these facilities can be used to provide a deterministic event synchronization, like a tightly coupled system.

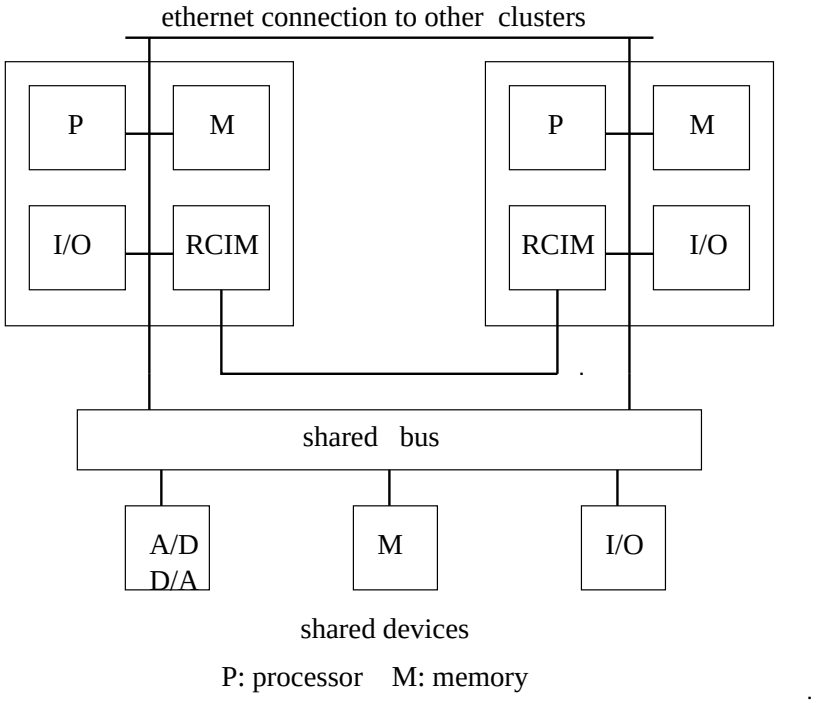

Figure 3: Concurrent's solution(adapted form [8])

3.3 Replicated Shared Memory Network

The Replicated Shared Memory Network is yet another attempt to retain the advantages of both tightly and loosely coupled systems and keeping out their weaknesses[9]. This

system is a very suitable alternative for a multiprocessing system having large number of processors, a situation in which the tightly coupled systems will not work.

Physically, it is a serial ring network with replicated memory at each node. But to the software it appears as a shared memory architecture.

Each processor has a network card. These network cards are connected through a serial ring network, forming a ordinary LAN between the processors. But no kind of complicated message passing or network protocol that eats up CPU time is used by the system. No packets with lengthy protocol information circulates through the ring. This minimizes software overhead for data communication and does not permit non-determinism to creep into the system.

Memory for each processor is divided into the local memory for the processor's own use and shared memory. When a processor writes at a location within it's shared memory region, the value of the corresponding memory locations at all other processor nodes are also changed. For example one processor changes the memory location 0036H to some value A. This value A and the location address are transmitted in the network through the network card. Every other computer, on receiving these data, updates the content of the same relative shared memory location to 0036H. Thus the system functions as a shared memory network although it is physically not so.

Similarly interrupts generated at one processor node can be transmitted to all other using the network in order to achieve control and coordination of the events occurring in the system. The transmissions in the the network is short and of fixed length and no non-deterministic software routines are necessary for this purpose. Data communication is possible within time interval in the order of microseconds and tight control over the system resources are maintained by passing interrupts, all like a tightly coupled system. On the other hand, the processors can be separated over a large physical distance and number of processors that can be connected to the system can be large.

Shared Common Random Access Memory Network (SCRAMnet)[9] is an implementation of the replicated shared memory network and in addition to real time simulation, it has find its application in other real time systems in data acquisition, telemetry, etc.

3.4 Switched Network Interconnect

RACEway is another very high speed interconnection between multiple processors using crossbar switches. It is developed by Mercury Computer Systems[10]. RACEway is switched network of processors. It uses 32-bit parallel data path. It consists of 6-port crossbar switches with processors being connected at the ports. Thus upto 6 processors can be connected to each other using one crossbar switch. Each switch is capable of maintaining 3 parallel data communication paths, each at the rate of 160 MB/s, giving a total throughput of 480 MB/s. On demand, a point to point communication channel is established between any two processors, provided none of them is already in communication with some other processor. End points need not be a processor only, but can also be an I/O node. Systems of higher complexity requiring more than 6 number of processors requires cascading of more than one crossbar switch. Latency in the data path per crossbar switch is also very low.

An improved version of RACEway known as RACE++ has also been developed. It consists of a 8-port switch allowing 4 simultaneous data paths each at the rate of 267 MB/s between the end point. RACE++ also supports priority-based connection.

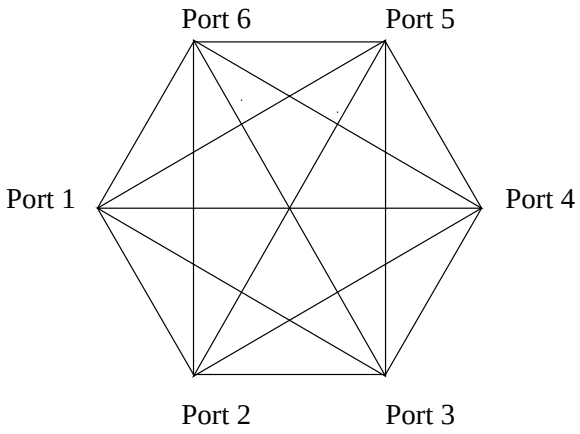

Figure 4: RACEway Crossbar Switch(adapted form [14])

3.5 Interprocessor Communication using multiple dual-ported RAMs

An efficient method of interprocessor communication using dual-ported RAM for three or more processor elements has been developed by Jagadish, Mohan Kumar and Patnaik[11]. The efficiency of the data communication in this method is due to that a memory module is shared by no more than two processors. Hence the conflict in accessing the memory is minimum.

A dual-ported RAM (DPR) has two independent left and right ports. It has separate address bus and data bus and busy signal for each port. The memory space can be independently accessed from each port. On chip arbitration logic handles the address contention to allow maximum speed of operation. In case, the processors connected at two port tries to access the same memory location at the same time, one of the processor has to wait until the other's access is over and a busy signal is generated to indicate this.

In this scheme, there is a DPR between any two neighbouring processors. Apart from this there is processor node acting as network controller. Every other processor shares a DPR with the network controller. Communication between nodes which are not directly connected, takes place through the network controller.

Suppose there are 4 processors PE0, PE1, PE2, PE3 connected in an ring. A DPR is connected between PE0 and PE1 and another DPR between PE0 and PE3. So PE0 can communicate with PE1 and PE3 without any contention at all. There is a network controller node and each processor shares a DPR along with the network controller. If PE0 wants to communicate with PE2, with which it is not connected directly, it can write the data on DPR it shares with the controller. The controller then block transfers the data into the shared memory of PE2. Thus , without having a common memory between all processors, this approach provides multiple interprocessor connection to be established eliminating the need for conflict resolution. Another advantage of this approach is the low cost of this solution.

3.6 Application Specific Topology

The systems which have been described so far are general systems, which can be used for any real time system, in general. However, if the nature of an application is known

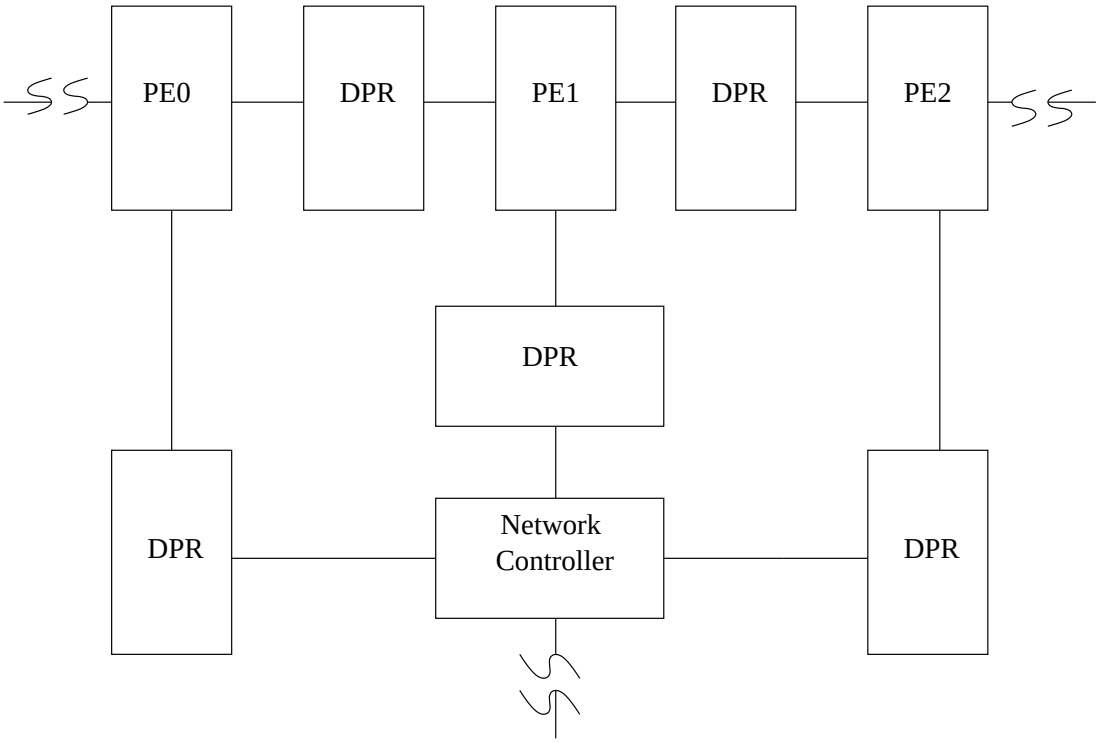

Figure 5: Dual-ported RAM based Interprocessor Communication(adapted from [11])

beforehand, a custom interconnection mechanism can be designed to achieve optimum performance for that particular application[1].

Usually in real time applications, the different parallel tasks can be identified. These parallel tasks can be assigned to different processors. Processors executing tasks which need to communicate with each other may be provided with a direct link. For example, suppose, in a particular application, there may be 4 tasks assigned to four different processors– P1,P2,P3,P4. The nature of the application demands communication between P1 and P4, P2 and P3, P1 and P2. In that case, direct links can be provided between these pairs. This kind of system works well with small systems. But for complex systems where there are many processors , each need to communicate with several of others, this topology can turn out to be a mesh, very difficult to manage.

3.7 The Ordered Memory Access Architecture

This architecture is based on the fact that for real time applications, often, the sequence in which different processors need to communicate with each other can be pre-determined. A controller is then used to grant access to the shared bus or shared memory or I/O devices according to the pre-determined order.

Based on this concept of ordered interprocessor transactions, the Ordered Memory Access(OMA) Architecture has been developed[3]. During the compilation of the code, the sequence of the accesses of shared resources by different processors is prepared and this access ordered list is downloaded to the memory of a "central transaction controller", which acts a bus access controller. The transaction controller then steps through this list and grants the bus access to the processor first in the list, by asserting a bus grant signal

to the concerned processor. The processor takes the control of the bus, performs the read or write operation on a shared memory location and then releases bus. The transaction controller waits for the bus release signal, and then grants the bus access to the processor next in the list. When it reaches the end of the list, it loops back to the beginning of the list.

Again, in real time applications, inputs are accepted and outputs are required after fixed periodic intervals. The I/O operations are managed by including the bus accesses for shared I/O devices in the access order list and granting those bus accesses accordingly during execution.

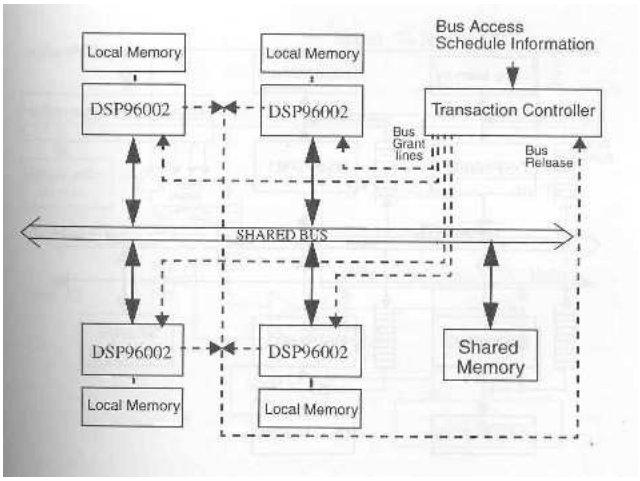

Figure 6: Block Diagram of the OMA prototype([3])

3.8 The VMEbus

Real time multiprocessor systems generally use a standard bus, so as to make systems and modules made by different companies compatible. In this respect the VME(Versa Module Europa)bus is a very popular bus widely used in many real time applications. In most of the multiprocessor systems described above, namely the Ixthos solution, the Concurrent Computer Corporation solution, the RACEway all uses VMEbus. An improved version of SCRAMnet, named SCRAMnet+ supports VMEbus.

VMEbus is an industrial open standard system. The VMEbus boards can be plugged into a backplane having 21 different slots. The VMEbus specification includes the physical dimensions of boards and backplane as well as electrical specifications of the bus and various communication protocol. Physically VMEbus board can be of two sizes– smaller one is known as 3U and the larger is known as 6U. Each of them has a 96-pin connector known as P1, arranged in three rows. The board is plugged to the backplane using P1 connector. 6U boards may also have an optional 96-pin connector, named P2.

8-bit, 16-bit, 32-bit data transfers are possible. Address bus can be 16-bit, 24-bit, 32-bit wide. The boards connected to the VME backplane can act as controller or master or slave. The VMEbus can also be divided into (a)arbitration bus required the bus arbitration logic, (b)data transfer and address bus, (c)interrupt bus, (d)utility bus to supply power to the boards, system clock, system reset etc.

The controller always sits in the first slot in the backplane. After receiving a bus request

signal from a master, if the controller finds the bus free, it sends the bus grant signal to the requesting master. While the bus is busy it indicates the bus busy condition by driving the bus busy line low. The controller also manages the interrupts on the bus. There can be only one controller in the VMEbus

The master reads and writes data to or from a slave. When the master gets the bus grant signal, it uses the address and the data busses for accessing a slave. There can be any number of master and slave in the bus, but at a time only master have the control of the bus.

A VMEbus slave watches the address bus, to determine whether it is being addressed. Once a particular slave device is addressed by the master, the slave receives information from or outputs data on the data bus as requested by the master

The VMEbus can provide a maximum data transfer rate of 40 Mbyte/s, and improved versions like VME64x can give upto 160 Mbyte/s, and VME320 upto 500 Mbytes/sec.

Further information regarding VMEbus is available in [15,16].

4 Conclusion

Hardware in Loop is a very effective tool in testing real time controllers. However it has its limitations[4]. It can only read the outputs of the device under test and supply appropriate inputs to it. It does not know and cannot tell what goes on inside the controller. If the hardware or the software is not functioning properly, the information obtained from reading the outputs of the controller may not be sufficient to tell which part of hardware in not working and why, or what are the values of the variables used. Thus it is not a replacement of the instruments like oscilloscope, logic analyzer or software debuggers

Since both the simulator and the controller is running in the real time, it is very difficult to study the process going inside the latter without interfering into both of them simultaneously. This however defeats the purpose of HIL, since, in that case, it is no longer acting in real time. To solve this problem, a synchronization feature can be built between the two concurrently running entities. This allows us to expand the time scale, such that both the controller and its external world run in slow motion, giving us ample time to check critical transactions.

Traditional approaches of tightly coupled and loosely coupled systems are valid, in general, for any multiprocessing systems. To suit the need of real time applications, these concepts are extended or modified and are used to built practical systems. The key is to maximize data transfer rate, provide very precise synchronization with minimum conflict and overhead. Some practical systems do this by going for the shared bus architecture and then modifying it by providing alternative interprocessor communication channels apart from the shared bus. Ixthos' MPR, MPC, Concurrent's RCIM connected by PCI bus are examples of such alternatives. SCRAMnet and the scheme by Jagadish, Mohan Kumar, Patnaik minimize or eliminate contention by arranging for physically separate memory devices. Data are transferred to and from these devices either through a networking medium or by means of a processor dedicated for this function. RACEway crossbar switch establishes multiple communication channels simultaneously and thus reducing conflicts. Application specific topologies solve this problem by providing physical interprocessor links wherever necessary. In the OMA architecture, each processor gets access to the bus, when it's turn comes.

The hardware configurations are discussed in this report in context of general real

time systems. However, as mentioned in the beginning, an HIL simulator is nothing but a specialized real time system, to be designed specially considering the characteristics of the simulated system, these configurations can be used for making an HIL. Real time simulators like HIL, based on the systems such as SCRAMnet, and that of Concurrent Computer Corporation, have been developed and are available in the market. Others like hardware coordinated system developed by Ixthos, RACEway, OMA architecture are useful for any real time systems ,and are also applicable for HIL applications.

References

[1] Y. Paker, "Multi-microprocessor Systems", Academic Press, 1983.

[2] Ben Catanzaro, "Multiprocessor System Architectures", Prentice Hall, 1994

[3] Sundararajan Sriram, Shuvra S. Bhattacharyya, "Embedded Multiprocessors-Scheduling And Synchronization", Marcel Dekker, Inc, 2000

[4] Martin Gomez, "Hardware-in-the-Loop Simulation", Embedded Systems Programming, http://www.embedded.com/story/OEG20011129S0054, November 2002

[5] John Boyd, Roger Theyyunni, "Development Of A Real-Time Simulation System", Embedded Systems Programming, available on Applied Dynamics International HTTP site, http://www.adi.com/pdfs/dev_real_time.pdf, November 2002.

[6] Macro A. A. Sanvido, Walter Schaufelberger, "Design Of A Framework For Hardware-in-the-Loop Simulations And Its Application To A ModeL Helicopter", http://www.aut.ee.ethz.ch/sanvido/HIL/eurosim2001_paper.pdf, November 2002.

[7] Ixthos, Inc., "Overview of Real-Time DSP Multiprocessing", http://www.insyst.fr/Ixthos/mutiprocessing.pdf, November 2002

[8] Concurrent Computer Corporation's HTTP site, http://www.ccur.com/realtime/, November 2002

[9] Systran Corporation, "Shared-Memory Computing Architectures For Real-time Simulation–Simplicity And Elegance" http://www.systran.com/ftp/literature/sc/sande.pdf, November 2002.

[10] Mercury Computer Systems, Inc. HTTP site, http://www.mc.com, November 2002.

[11] N. Jagadish, J. Mohan Kumar, L.M. Patnaik, "An Efficient Interprocessor Communication Using Dual-Ported RAMs", IEEE Micro, October 1989.

[12] Stephen Brosky, "Closely-coupled Single Board Computers", http://www.ccur.com/realtime/closelycoupled.htm, November 2002.

[13] Edward C. Jenning, "Birth of the Virtual Car", International Symposium on Automotive Technology and Automation, 1999, available on, http://www.ccur.com/realtime/Jennings-VirtualCarAmerformat.htm, November 2002.

[14] "Digital Signal Processor Subsystem", http://www.nssl.noaa.gov/orda/dspsys.htm, November 2002.

[15] VMEbus Industry Trade Association's HTTP site, http://www.vita.com/, November 2002.

[16] Leroy Davis, "VME Bus", http://www.interfacebus.com/Design_Connector_VME.html, November 2002.