IEEE Transactions on Nuclear Science, Vol. NS-30, No. 4, August 1983

A MULTIPROCESSOR BUS ARCHITECTURE FOR THE LEP CONTROL SYSTEM

J. Altaber and R. Rausch,
European Organization for Nuclear Research (CERN),
1211 Geneva 23, Switzerland.

Summary

The architecture and the system aspects of the multi-master bus used for the construction of the process computer assemblies and the message handling assemblies of the LEP control system are presented. This bus architecture provides a fully distributed reservation mechanism and a protected access of the peripherals to prevent processor interferences. To achieve this, a channel concept is proposed, using a system bus for communication between processors and with their peripherals. The architecture is microprocessor independent, provides dynamic bus allocation amongst several microcomputers, offers processor position independence and has a multimaster bus extension to several crates with a homogeneous addressing. A global system concept, called E3S, has been developed including the definition of software primitives. The bus access software is organised in a layered structure matching the module functionality.

Introduction

The control system for the Large Electron Positron accelerator (LEP) will use a fully distributed computer system made up almost entirely of microprocessors[1]. The LEP control system philosophy follows the ideas implemented for the control of the SPS, in operation since 1976, but will take advantage of the technology of the 1980's. Progress made in VLSI, 16/32 bit microprocessors, multiprocessor busses, computer architectures and networks open new possibilities not available in 1972 when the SPS control system was designed. In particular, the minicomputers can be replaced by microcomputers assembled on a suitable multiprocessor bus[2]. As a consequence of this progress, the conventional process computers, the message transfer computers and the input/output preprocessing equipment can be implemented in an homogeneous crate and bus system provided that the system offers the required multiprocessor facilities.

The purpose of this paper is to review the requirements to be satisfied by a multiprocessor bus architecture and to propose an inter-processor communication concept using messages. A communication channel organization with reservation and access protection and a layered software structure for bus access primitives conforming with the proposed E3S[3] standard system specification is described.

Requirements for multiprocessor operation

Without going into details, this paragraph summarises the technical properties we consider important or desirable to achieve a flexible, reliable and efficient multiprocessor operation.

Bus arbitration mechanisms

As several microcomputers have the same back-plane for interleaved input/output transfers a bus arbitration procedure amongst the various masters is necessary. Three categories of arbitration can be identified :

a) Daisy chaining of the masters. This is the conventional way of organising the priority access of temporary masters used in most minicomputer architectures. In a single CPU configuration it requires a wired-OR bus request line going generally to the processor card. The mastership is given to the requester via a GRANT-IN/GRANT-OUT line daisy chaining all potential masters and by-passing non-master cards; the closest to the processor having highest priority. This principle is easy to implement but suffers from three inconveniences. First the priority is position dependent, second insertion/deletion of cards alters the operation and third a fixed priority arbitration algorithm is implemented. This third limitation can be overcome partly by the provision of several pairs of bus request/grant lines and an arbiter module (e.g. the VME[4] bus provides by this means four levels of priority). The crate configuration is even more difficult to trace in this case.

b) Centralized arbitration. In this arrangement separate request lines end up in an arbiter module located at dedicated position of the backplane. The mastership is usually granted to the requester via a bussed line while its request is held by the arbiter (i.e. EUROBUS)[5]. Centralized arbitration is also simple to implement, but it needs a central arbiter module in which any type of arbitration algorithm can be achieved, i.e. priority, round robin, first requested first served or a mixture of these. Centralized arbitration allows full position independence when a circular priority algorithm is implemented.

c) Distributed arbitration. Other buses propose a fully distributed arbitration where the relative priorities are fixed or programmable on each card (i.e. FASTBUS and I/P-896)[6]. A master makes a bus request by putting its binary priority code in parallel onto five or six lines. An on-board self-selection logic resolves simultaneous requests amongst masters. The major advantage offered by a distributed arbitration is that the module is position and priority independent, without the daisy chaining burdens. As no central arbitration module is required system duplication and stand-by capability for back-up can easily be implemented. In addition, the current master is identifiable by all processors during arbitration time. As a counterpart to these advantages an arbitration logic is required on-board of each master and a fair circular priority algorithm is difficult to achieve.

For the LEP multiprocessor bus architecture a distributed arbitration mechanism is implemented in the VME bus in a compatible way with the daisy chain arbitration to allow the use of VME manufacturer's modules. Despite the sligth additional logic, it offers the desired position independency as all modules become temporary master of the bus. The priority of each master module is fixed by on-board switches. For urgent message transactions or interrupt vector generation this priority is temporarily raised by activating the line carrying the most significant priority in addition to the normal module's priority level. This method overcomes the limitation of fixed priority allocation.

0018-9499/83/0800-2281$01.00©1983 IEEE

Addressable interrupts

In a single processor environment all interrupts generated by peripheral modules are sent to the processor. In a multiprocessor system the same peripherals or input/output modules are shared dynamically amongst the various processors. A processor and a slave or two processors co-operate temporarily for execution of a specific task. At completion of a task both modules become available again for other tasks. While for single processor operation a physical link between the module generating the interrupt and the processor receiving it is acceptable, for multiprocessor operation a programmable and addressable interrupt mechanism is required. This can be achieved by a normal input/output cycle generated with high priority.

The generation of an interrupt vector by the source module implies bus mastership capability from that module. For EUROBUS the interrupt vector is an address-only cycle (data-less cycle) while for the VME bus the interrupt vector is a data byte transfer following an interrupt request. For I/P-896 the interrupt vector is a normal write cycle, generated by the source module, in which the address is part of the address space and the data may contain additional information.

The multiprocessor bus architecture for the LEP control system will make use of programmable interrupt vector cycles generated by the source module. The destination processor receives in a FIFO buffer these interrupt vectors accompanied by their source module number. The bus mastership request for interrupt generation is made at high priority to go through the normal bus traffic with minimum time delay.

Resource allocation and protection

In multiprocessor operation several processors may share dynamically resources connected onto the same backplane bus. To become master of a resource a protocol must be defined and be obeyed to avoid inteference between processors. Once allocated to a temporary master, access to this resource must be secured and a signature protection mechanism be implemented to reject any command issued by non authorized masters.

The most popular method to establish communication amongst processors uses the shared memory concept. In this case the access is protected by memory mapping and management unit. This method however does not protect shared memory or peripherals against faulty hardware/software addressing over the common bus. In fact, to guarantee a secure access to a resource the protection must be implemented at the destination.

The processor and slave modules designed for the LEP control system will contain an homogenous register allocation and a signature protection mechanism. For the purpose of assembling "real" systems consisting of modules acquired from multiple sources, some measure of logical compatibility is required. Logical compatibility means that modules performing the same or similar functions do so in a consistent way. Thus a uniform scheme for allocating addresses and interrupt vectors has been adopted. In addition, the format for the Control and Status Register (CSR) of each channel within a module as defined by E3S has been adopted.

Multicrate capability

To assemble large multiprocessor systems and/or numerous input/output and peripheral modules multicrate operation has to be provided. A multicrate highway interconnects up to 7 crates by using bus linker modules. The multicrate operation is a true backplane dataway extension with straight-through operation carrying all bus cycles and having its own arbitration mechanism. An addressing scheme homogeneous for all crates has been adopted in Peripheral Address Space (PAS) and Memory Address Space (MAS).

The crate address is set in the bus linker module within each crate. Any master including the bus linker addresses local modules by using the crate number C=0. When a master needs to communicate with a slave in another crate, it addresses it with the corresponding crate number C=0. Priority access logic to the multicrate highway is provided in the bus linker to solve the contention problem of simultaneous transfers occurring in different crates. This may be done with a daisy-chain or circular priority type. A mandatory requirement from the bus architecture is to provide a bus de-allocate signal activated by the bus linker to suppress mastership of the current master, thus allowing the deadly embrace situation to be overcome.

The channel concept

The bus interface logic of a module is organized into "channels". Each channel is a functional unit organized as a set of consecutive register addresses operating independently of all other channels. Some peripheral modules consist of several identical channels as, for example, a multiple serial interface where these units are usually independent of each other. Each unit has its own Control and Status Register (CSR) and Data Register (DR); each serial line constitutes a channel. Such an organization allows a processor in a multiprocessor system to reserve a channel for its exclusive use for a period of time without interference with other masters. The signature protection mechanism presented above, is for that reason, implemented separately for each channel in a module.

Communication by messages

The concept prevailing for the implementation of the LEP multiprocessor assemblies is based on a Function to Function Architecture (FFA)[7] in which each processor is dedicated to a specific task and where communication is established amongst processors by messages, Fig. 1. A General Processing Unit (GPU) is implemented on a single card and may be coupled to an extension card via a private connection forming thus a single functional module. A set of functional modules plugged into a multimaster bus forms a specific system like a Process Control Assembly (PCA) or a Message Handling Assembly (MHA). The LEP multiprocessor concept distributes functionally the computing power, provides an easy to use and manageable structure, offers simplified operational procedures and off-loads the system bus. The system bus traffic is in this case limited to inter-function traffic and shared data traffic if required.

To allow function to function communication amongst these modules a standardized method of interfacing to the system bus has to be applied and a defined module to module communication protocol to be obeyed.

The standardized method for interfacing modules to the system bus follows the homogeneous register allocation of the channel concept described previously.

The inter-module communication protocol uses a set of conventions for resource reservation with protected access. The basic primitive commands are of the following type : send, receive, respond, wait for response. Communication between a channel and a processor for sending a message is done after the reservation phase with the signature protection presented above.

Interrupts accompanying a message transaction are generated by the resource module and selectively addressed to the temporary master processor receiving them through its own signature protection mechanism.

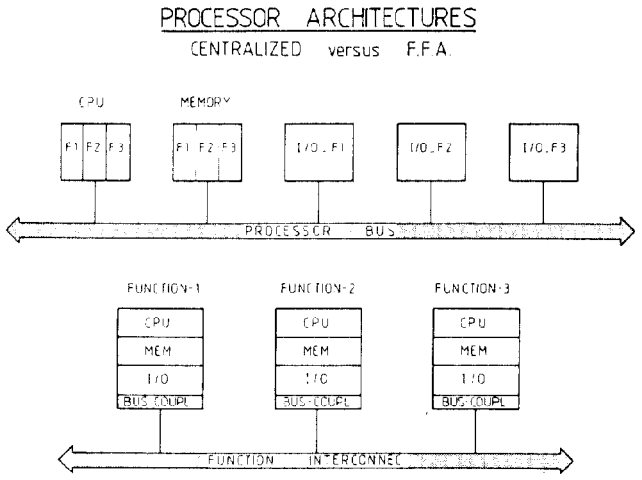

Fig. 1

Software primitives

For accessing the system bus a structure has been defined for the software primitives in a similar way as it has been done for the modules interfacing registers. This structure is organised in three layers :

a) Layer 0 for bus cycles.

These are the basic software sequences generating the various bus transactions : READ cycle, WRITE cycle, READ and RETAIN, WRITE and RETAIN, VECTOR cycle, INIT cycle and STATUS cycle. RETAIN allows non interruptible input/output bus cycles to be executed by a processor.

b) Layer 1 for modules and registers.

Each processor module contains a configuration table giving the addresses of all modules in the system including itself. Layer 1 primitives uses this table to verify and translate module and register addresses into bus addresses. Typical functions are : READ bus, WRITE bus, READ block, WRITE block, BASIC RESERVE, CONNECT to interrupt, DISCONNECT from interrupt, WAIT interrupt and INIT bus.

c) Layer 2 for channels.

Each processor module contains a configuration table giving the CSR and module addresses of all channels in the system including its own and the current state of reservations and connections for each channel reserved by the processor. Layer 2 primitives uses this table to verify and translate channel codes into module and register addresses. For layer 2 typical functions are : RESERVE channel, RELEASE channel, CONNECT to channel, DISCONNECT from channel, WAIT channel, READ channel, WRITE channel, RECEIVE frame, SEND frame and INIT channel.

References

1) M.C. Crowley-Milling, The Control System for LEP, 1983 Particle Acc. Conference., March 21-23, 1983, Santa Fe.

2) J. Altaber, M.C. Crowley-Milling, P.G. Innocenti, R. Rausch, Replacing mini-computers by multi-microprocessors for the LEP control system, 1983 Particle Acc. Conference, March 21-23, 1983, Santa Fe.

3) E3S - ESONE Standard System Specification. Proposal to be submitted to the ESONE General Assembly, May, 1983, Berlin.

4) VME bus specification by Motorola Mostek, Thomson-EFCIS, Signetics.

5) EUROBUS, British Ministry of Defense, BUS-DSWP/7232 and Ferranti Computer Systems Ltd., Bracknell.

6) I/P-896 - P896 US-IEEE Computer Society and I896 European Ewics-TC10 Working Group.

7) M. Conrad, W.D. Hopkins, Texas Instruments, Houston, Functional architecture threatens central CPU's - Electronic Design, September 3, 1981.