MARTOS

ISSN 0323 696 X
ZfR – 83.02

ZfR-INFORMATIONEN

5th INTERNATIONAL SYMPOSIUM ON COMPUTER NETWORKS FRANKFURT/ODER

MARCH 1–4, 1983

AKADEMIE DER WISSENSCHAFTEN DER DDR
ZENTRUM FÜR RECHENTECHNIK
BERLIN-ADLERSHOF

Herausgeber:
Akademie der Wissenschaften der DDR
Zentrum für Rechentechnik
1199 Berlin
Rudower Chaussee 5
Ag 521/731/83 III-12-12

MICROPROCESSOR BASED COMMUNICATION CONTROLLERS ARCHITECTURE AND IMPLEMENTATION

A.B.Martos - I.Tétényi

Computer and Automation Institute, Hungarian Academy of Sciences - MTA SZTAKI

Abstract: Two basic architectural model for implementing communication controllers are discussed. The first scheme is a distributed easily reconfigurable multimicrocomputer system consisting of traditional MOS and bipolar bit-slice microprogrammable microprocessors. An EC8421 compatible communication multiplexer was implemented using this shared memory multimicroprocessor system. The second architecture discussed is a centralized system consisting of a central processor, a communication scanner and a host channel interface. On the instruction level it is compatible with the EC8371 front-end processor but it has a microprogrammed CPU and microprogrammed peripherals. A system tradeoff analysis related especially to network applications is provided regarding the different approaches.

1. Introduction

When speaking about communication controllers one can't avoid to deal with the characteristics of the task they have to perform. Considering the structure of network communication software it can be stated that due to international standards most networks apply layered structure (Open Systems Architecture) or a similar well structured model. The common attribute of these models is that they all have a hierarchical structure where each functional entities consists of the same layers. Every layer has well defined boundaries and offers services to the next higher layer. The hierarchical and distributed feature of network architecture involves a tendency towards a hardware structure that is decomposed and distributed in a similar way.

2. Implementation of a Communication Multiplexer

The EC8421 type communication multiplexer is originally

5th Int. Symp. on Computer Networks, Frankfurt/Oder
March 1-4, 1983, ZfR-Informationen 83.02

a nonprogrammable device. It has a well defined hardware-software interface not only to communication lines but to host computer as well. In this way the designer has a fairly high degree of freedom in choosing the internal structure of the system. The growing demand for a simple inexpensive, easily reconfigurable device for network application gave the idea to develop a multimicroprocessor system that may be well suited even to other communication network applications.

A strong argument for a multimicroprocessor based system is its potential cost effectiveness. Multiprocessor systems require, however, a very careful design so that interprocessor communication, synchronization overhead won't result in significant performance degradation. The basic concept was a highly modular architecture with a corresponding moduler software system for reasons of incremental expandability, ease of maintenance and low producing cost.

The general architecture of the multimicroprocessor system is shown in Fig. 1. To overcome the problem of high speed host channel interfacing two types of microprocessors are used: traditional MOS microprocessors as communication line handling units and a bipolar bit-sliced microprogrammable microprocessor as host interface handler. The microprocessors are interconnected via a shared common memory bus. There are two ways of interactions between the microprocessors. First the occasional transfer of data blocks through the shared memory. Secondly the interrupt requests generated to each other.

Every microprocessor has a private bus used to interconnect the central processor, private memory, direct memory access module, common memory interface and some input-output interfaces. The implementation of the processor units varies with the different types of microprocessors (MOS or bipolar) but it doesn't disturb the general conception. None of the processors can access addresses on another processor's private bus. As the system has disjoint local memory address spaces it can be classified as a loosely coupled distributed

system.

The software of the communication multiplexer has the following layers:

- data multiplexing layer,
- data link layer.

The data multiplexing layer performs the logical multiplexing of data. A high independence is achieved between the two layers. The operator communication, system restart, etc. belong to this level as well. One may notice that this level is almost empty. It is due to the simplicity of functions.

The data link layer accomplishes the error free transmission of data, performs terminal emulation, etc. Its functions are supported by an LSI communication chip (USART).

The multiplexer based on multimicroprocessor architecture has some attributes that are realized in a different way from the original hardwired machine. The implemented store-and-forward method releaves the host processor of asking for message repetition in case of erroneus receive. Even the actual configuration can be programmed in a simple manner.

3. Implementation of a Front-End Processor

The development of an EC8371 type compatible front-end processor has fundamentally different requirements from that of the communication multiplexer. It is a programmable device and original system/application programs must be able to run the new design as well. It is of vital importance because the host interface is partly hidden in the system software and because the existing software is hundreds of Kbytes of magnitude. The requirement of compatibility on the instruction level has strictly limited the feasible architectures.

The architecture of the front-end processor can be represented by Fig. 2. Major system elements include the central processor, main memory, communication scanner and

host channel adapter modules. A front panel is attached to the central processor for control and maintenance purposes. The early models of scanner and channel adapter have access only to the system input-output bus and all characters received or transmitted are handled by the central processor. As a large amount of data must be exchanged between the scanner and channel adapter latest models have already direct memory access capability (dotted lines) and some increased intelligency.

Because of the limitation mentioned above a parallel processing approach was hard to realize. In spite of that the decision was made to apply microprogrammable microprocessors as central controllers in the central processor and in the scanner and channel adapter modules as well. Microprogramming results in a more structured organization of hardware logic, a substantial increase in versatility and in a greatly simplified control portion of the system. The pure hardware development changes, however, into a complex hardware-firmware development process. Microprogram developing efforts may be a considerable part of the entire work. A powerful developing aid is needed for testing and debugging the firmware.

To support the operator's functions the original front panel was replaced by an intelligent console display. The console allows the system operator to simulate accesses from the central processor to the main memory and peripherals. Due to its intelligence the console is suitable for supporting the firmware development and maintenance as well.

4. Conclusions

By applying high speed intelligent controllers in the front-end processor it is potentially possible to releave the central processor of some low level tasks (phisical and data link control, etc.). The incorporation of microprogrammable microprocessors into peripheral controllers wasn't made really with the intention of

having distributed intelligence, for it can't be exploited by the current software, but because equipment size could be reduced by a factor of two. Yet, it is obvious that the distributed approach is in the nature of network processing and it may result in a considerable increase in throughput. The overwhelming part of the message handling task requiring sophisticated intelligence is not related to the body of the messages but the header. Error checking, character conversions, character or bit stuffing, synchronization, etc. can be performed without having any knowledge of other communication links. But as to the quantity of work the latter comes to the major part because of the large amount of data.

As for hardware prices it is quite clear that several microcomputers can be used before the hardware cost will equal that of a large unicomputer. The centralized approach 'a priori' requires a considerable investment usually without having accurate conception or knowledge of prospective utilization.

Another important attribute of a multiprocessor system is its potential for reliability. A structure consisting of large numbers of identical processors represents a form of redundancy that is essential in the design of a reliable system. Faulty elements can be detected and isolated, allowing graceful degradation but not complete system breakdown.

Yet another factor that favors the use of multimicroprocessors is the resulting modularity of the system. During system life cycle easy reconfigurations and even incremental increase (or decrease) may be performed when throughput needs or user's requirements change. This is especially important in network applications where quantity of users, load on different communication lines are often subjects to violent fluctuations.

Considering software problems in distributed systems it is clear that new methods and tools are required so that software development cost for distributed multimicroprocessor systems won't cancel out the cost savings of not using

a centralized architecture. Hardware chips can take over even software-supported data link control functions and it is expected that communication controllers will incorporate these control chips as well. The question, therefore, arises: Will communication software development give an adequate answer to the challenge of hardware-supported distributed architectures?

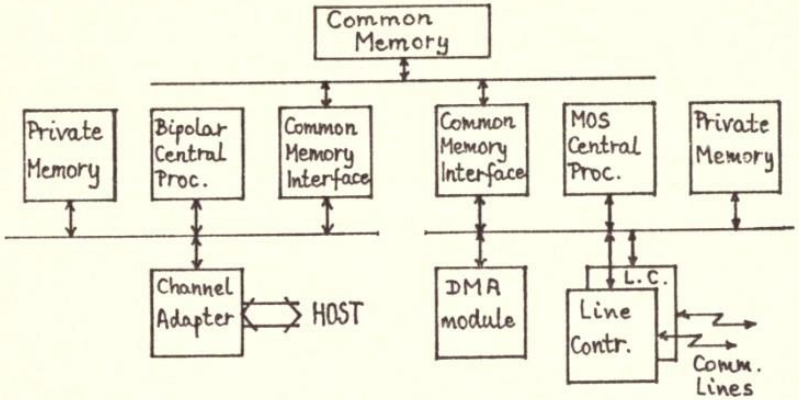

Figure 1.

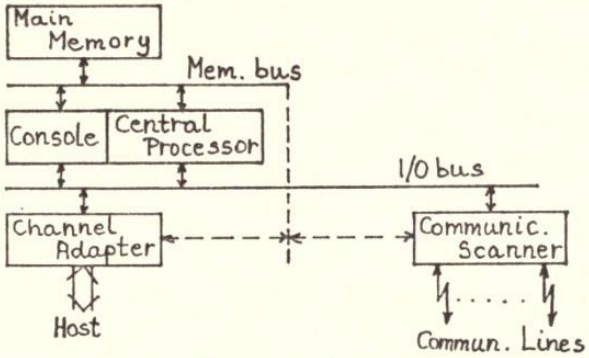

Figure 2.