# MARTINUS - MULTIPROCESSOR FOR HIGH CAPACITY REAL-TIME PROCESSING

by

Yngvar Lundh
Norwegian Defence Research Establishment
Kjeller, Norway

## 1 INTRODUCTION

Development of circuit technology in recent years now makes it feasible to build powerful processing machines at quite considerable savings - in comparison to those which have been considered feasible earlier. These savings are such that many real time processes are now practical which were not so before. By utilizing network structures of "computing elements" in which a number of computing elements are connected and so organized that they are able to share the total computing load between them, it is possible to move the limits of the magnitude of processing tasks which can be handled, very far.

Following is a description of a machine, which we shall refer to as a multiprocessor. It is programmable similarly to a "monoprocessor" computer. However, several features of multiprocessor programming are different from, and additional to those of conventional (monoprocessor) programming.

This structure is referred to by the name "Martinus".

## 2 MULTIPROCESSOR STRUCTURE

### 2.1 Process partitioning

First of all, the signal process is broken down into parts which are manageable by single computing elements. Two methods are available for this, alone or in some combination: Division of a process into "processing steps", and division of a processing step into "shares". Figure 1 shows these subdivisions.

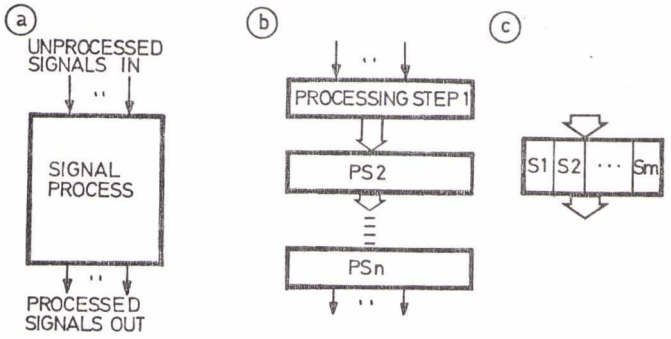


Figure 1 Breaking a process down into manageable parts or steps in a data flow

Figure 1 a) and b) indicate how n processing steps constitute the entire process. Signals are "refined" step by step until they emerge completely processed from the n'th and last step.

A process step may be such that it can be "shared". One computation program or "algorithm" may be executed m times, each time with a different set of input variables, to perform the entire step. This, of course, is much like an ordinary program loop. However, here we are concerned with shares which can be executed independently by separate computing elements if need be, to cope in real time.

## 2.2 Network components

The multiprocessor is constituted of devices which can communicate with each other. All communication is in the form of "packets" of information being sent over buses, see Figure 2. To each device, the bus effectively means a direct channel to each of the other devices connected to the bus. That means, a channel with enough transmission capacity to carry all traffic without need for each device to wait. In actual fact, the bus is a time shared switch. Each device is connected through a bus interface circuit which comprises a buffer capable of storing one incoming and one outgoing packet. The bus circuits are fast, and comprise "transmission arbitration" such that the transmission capacity normally is limited by the devices themselves.

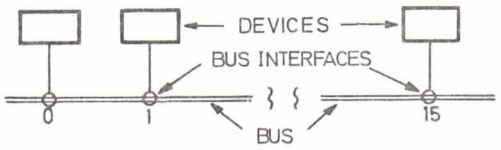


Figure 2 Devices are interconnected by buses

In order to achieve the required bus circuit speed and traffic handling capacity, there is a limit to the number of devices on each bus. The maximum is set at 16 devices.

## 2.3 Network structures

An arbitrary number of devices may be interconnected in a network. Each device has one or more ports for connection to one or more buses. An example network is shown in Figure 3.

Physically, the same network may appear as the equipment outlined in Figure 4.

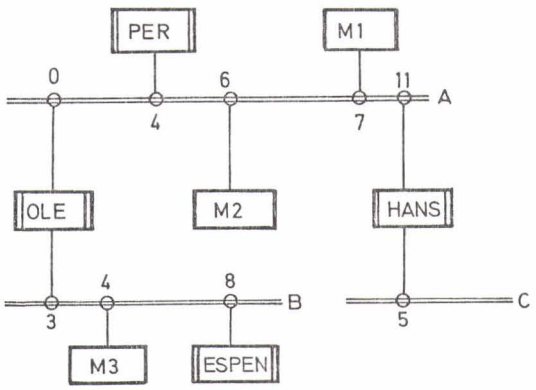


Figure 3 Example network

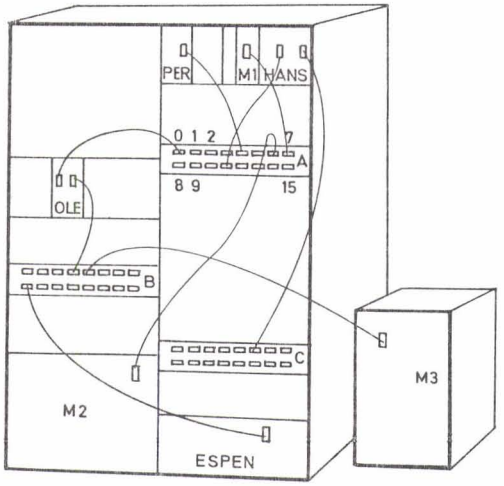


Figure 4 Example physical configuration

A bus appears to the outside as a panel with 16 multipin plugs. A device has a corresponding multipin plug for each "bus port". Connections are "patched" by multiwire cables. Drivers are included in the bus interface and bus port circuits to allow these cables to be up to a few meters in length.

A multiprocessor may be configured to be particularly suitable for a given process, e.g. in accordance with the data flow type architecture indicated in Figure 1. Note however, that we can configure our network any way we want. The structure is limited only by the number (16) of bus interfaces on each bus, and by the number of bus ports on each device.

### 2.4 Types and functions of devices

The typical device is either a computer called "computing device", processor or computing element, or a memory with controller called "storage device" or external memory. More specialized devices are also possible.

In fact, there is no other constraints on the types of devices which may be connected to a bus than the requirement to comply with the formats, control signals etc for data exchange with the bus interface. This format etc is called the "bus interface protocol".

Each device thus may have a special function - or it may also be a general programmable computer. Different types of devices may be used in any configuration to suit the needs of the process. To the buses all devices look alike in that they comply with the bus interface protocol.

### 2.5 Network interface protocol

Devices send information to each other in the form of packets. Each packet consists of a variable number, minimum 1, maximum 16 16-bit words plus a 4 bit destination device address.

Higher levels of protocol are used for the actual use of packets and the formatting of information in them.

One 16-bit word is transferred at a time either to or from a device. 16 data wires and 7 control wires are used for connecting each device.

## 3 MULTIPROCESSOR PROGRAMMING

### 3.1 Stepwise processing and intermediate storage

Each processing step results in a set of intermediate data. Between steps, data are stored in storage devices.

Computing elements typically have memory also - both for program and data. Storage devices are devices capable of just information storage and retrieval. Figure 5 outlines an example of how a process comprising three process steps PS1, PS2 and PS3 is to be performed by one or more processing elements each, and how intermediate data are stored in separate storage devices.

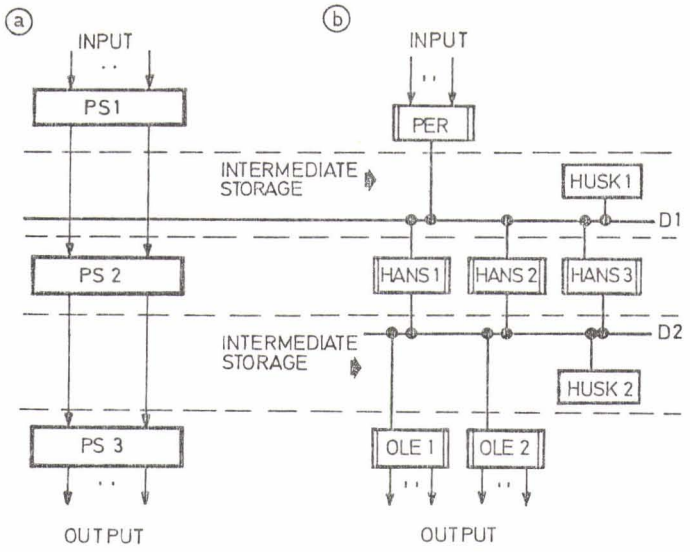


Figure 5 Process steps assigned to devices

In this example, the first step is performed by device PER. The resulting intermediate data from PS1 are stored in storage device HUSK1. The task of performing PS2 is shared by devices HANS1, HANS2 and HANS3, which fetch their input data from HUSK1 and deliver their resulting intermediate data into storage device HUSK2. Devices OLE1 and OLE2 share the task of performing PS3. They fetch their input data from HUSK2.

Note that we have shown computing devices and storage devices by slightly different symbols in the schematic diagram. To the bus they are all equal, i.e. all devices use the same bus interface protocol. The differences in their task, are noted by the programmer in this way.

Nets of many devices may require more than one bus, because of the limitation on the number of devices on each bus. Figure 5 shows how multiple buses can be used.

In addition to limitation of the number of interfaces on a bus, there is also a limit to the amount of traffic which the bus can carry. To avoid traffic congestion as a limiting factor, under-utilization of the bus connectivity may sometimes be required.

## 3.2 Supervision

Management of the multiprocessor is performed by a "supervisor"-computing device. It normally uses a separate bus for "direct lines" to all the other computing devices. Figure 6 shows how a supervisor and a supervisor bus are added to the net shown in Figure 5. Depending on the number of devices to be connected, more than one "supervisor bus" may be required.

Depending on the size of the processing load which the real time supervisory tasks represent on the supervisor processor, it is also possible for the supervisor to leave some of its task to "lieutenant supervisor processors", thus forming a hierarchy of supervisor processors.

The management tasks performed by the supervisor comprise: Programming, sequencing and self checking.

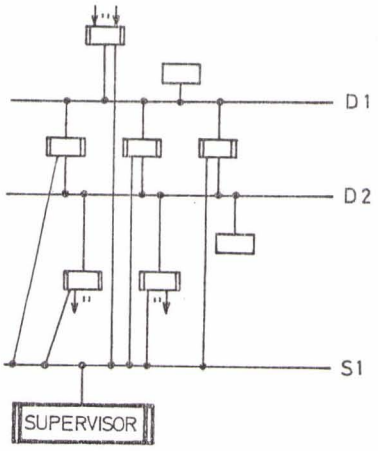


Figure 6 Net comprising supervisor

"Programming" means accepting program statements in symbolic form (source code), converting them into binary code, loading binary programs into the computing devices, and starting execution. Further, it comprises debugging facilities.

The sequencing or synchronization of tasks in computing devices comprises start and stop of program execution on an individual basis as required in continuous real time operation. Part of the sequencing task is to keep track, at all times of the validity of all intermediate data in storage devices.

Self checking is done by a separate set of "passive" programs running (in "background") in the supervisor and all other processors. They are called passive because they are executed in the spare time of the processors, that is the time when active programs, i.e. the signal process itself, can not be run. All processors are arranged to have some spare time in which to run these self checking programs. The purpose of self checking is to detect unexpected or abnormal conditions and report them to the programmers or operators of the signal processing machine.

"Programming" typically takes place prior to the machine starting real time operation, while "sequencing" and self checking are part of the real time operation itself.

The supervisor includes secondary storage (disk memory), and operating system, with multiterminal timesharing, file system etc, and a set of terminals for the programmers and real-time operational personnel. Included with the system programs in the supervisor, are a text editor, cross-translation programs (assemblers and compilers) for each type of computing device, and a "book-keeping system" which facilitates handling of such information as variable names and addresses of intermediate data, timing requirements etc. Also included is a command system which permits operators to select options, change parameter settings, etc during real time operation. Thus, the supervisor is to the multiprocessor what the operating system is to the monoprocessor.

## 4 COMMUNICATION FEATURES

Intra-communication in the multiprocessor, that is communication between devices, is carried by the buses between devices. The bus traffic comprises data and supervision traffic. The latter, to and from the supervisor, concerns the management functions, including, programming, sequencing and self checking. The former concerns the data representing the actual signals being processed, and which move between computing and storage devices as they are being processed step by step.

All traffic moves in the form of packets as discussed earlier. Inside each packet is information which permits the receiving device to distinguish between different types of incoming packets. First of all there is a need to separate supervision from data traffic. Further there are many types of packets which may occur within each of these types.

Information coding takes place according to a hierarchy of "protocols". The lowest protocol level of interest to the programmer is the bus interface protocol. Inside each packet, data may be identified as types of "data items". Examples of standard data item types or "data formats" are "16 bit word" or "complex number with 32 bit, fixed point precision". Data transfer takes place either as single packet exchanges or as "block transfer" in which a larger amount of data (many packets) is handled more efficiently. For the programmer, there is a standardized set of statements which he can include in symbolic processor programs, and which effect appropriate "packing and unpackaging".

## 5 IMPLEMENTATION

The general structure of such a multiprocessor, and the ways in which it is programmed have been designed to allow a considerable growth potential.

That means, many types of processing and storage devices can be used together in one multiprocessor. In other words, if as an example, circuit technology permits or a special application demands a new type of processing element, there is a straightforward method in which such a new type of device can be introduced. It does not have to effect the other devices and the way they are programmed and operated. All it means is that new device types may be readily accepted. Similarly, new buses with greater speed may be introduced, when circuit technology becomes available, which look like the old ones to the devices, but which may carry much more traffic - and thus may be commensurate with more powerful devices having greater throughput. None of these changes will require changes in the system programs.

Figures 7-9 show some examples of actual equipment, part of a machine under construction capable of some 120 million instructions per second.

This implementation permits restructuring of the interconnection of devices. This is interesting for large tasks where optimization of the structure, in view of the particular real time signal processing task at hand, is important for economical reasons.

This, of course, does not prevent a fixed structure to be built, which will never be changed. Such a permanent structure could be optimized for a class of problems, and convenient utility software could then be made more efficient and easy to use for such a multi-use type of machine.

## 6 SUMMARY AND CONCLUSION

An overview has been given of the main principles of a multiprocessor which has been particularly designed for the performance of real-time signal processing algorithms.

The principal objective is to achieve arbitrarily high processing capacity by combining the computing power of many computing and storage devices of limited capacity.

To design and implement programs for such a multiprocessor-machine imply many considerations which are not part of "monoprocessors". Part of the description here concerns methods for handling the additional complexity of a multiprocessor.

Due to the availability of more powerful circuit techniques, the implementation of such multiprocessor systems are now much simpler than before. Such complex structures thereby have become feasible, and make many high capacity signal processing tasks practical to implement.

REFERENCES

1. LUNDH, Y. Multiprocessor system for high-capacity signal processing, Proc. International Conference on Computer Communication, Stockholm (August 1974).

DISCUSSION

D. Nairn What speed is the bus?

Y. Lundh One 16-bit word in ~100 nsec.

Y.S. Wu Have you simulated the supervisor load? What is the capacity?

Y. Lundh No. The capacity of NORD 10 is 1 million instructions per second.

R. Seynaeve Could you compare this structure to CRAY-1 type.

Y. Lundh I am not prepared to discuss or compare other people's machines. It may, however, be of interest to note the fact that this reconfigurable system greatly facilitates the possibility of achieving high efficiency in the utilization of the available processor cycles (per second). The total available number of processor cycles, of course, is determined by the number of processors — a number which can be chosen to meet the needs of a given task.

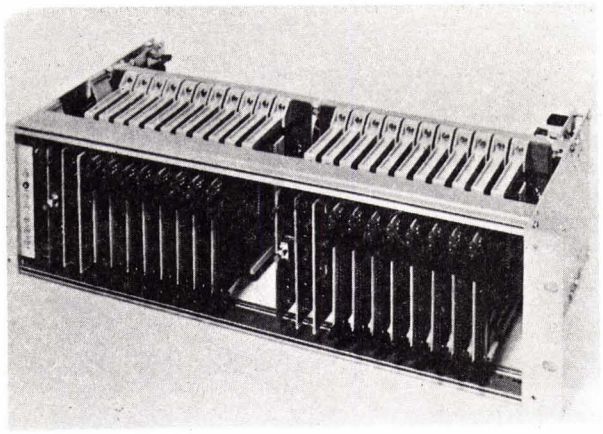

*FIG. 7*

*ONE TRAY HOUSING TWO BUSES, EACH HAVING 8-BUS INTERFACES. LAMPS INDICATING "OUTGOING PACKET" AND "INCOMING PACKET" ASSIST ERROR TRACING.*

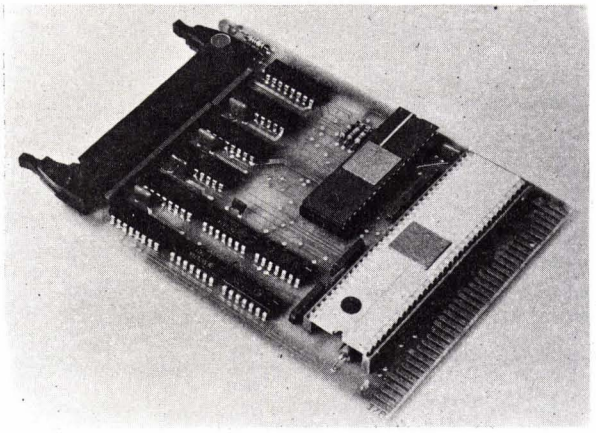

*FIG. 8*

*ONE BUS INTERFACE CARD COMPRISING PACKET BUFFER STORAGE, TRANSMISSION AND CONTROL CIRCUITS.*

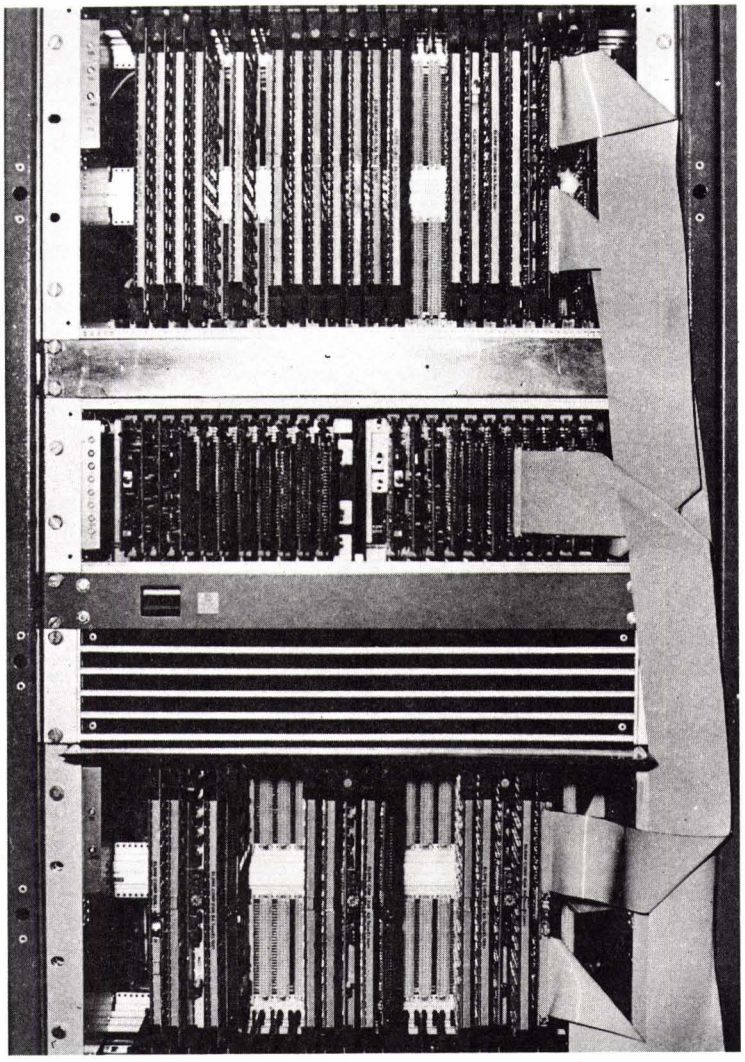

*FIG. 9*

*SECTION OF A MULTIPROCESSOR SHOWING HOW DEVICES AND BUSES ARE INTER-CONNECTED BY MULTIWIRE "FLAT-CABLE" AND PLUGS.*