Structure and operation of the HERMES multiprocessor kernel

by N.G. BOURBAKIS and D.K. FOTAKIS
George Mason University
Fairfax, Virginia

ABSTRACT

In this paper, we present the internal structural design and operation of the HERMES multiprocessor Kernel. HERMES is a heterogeneous, realtime, multiprocessor vision machine based on a two-dimensional array structure of \((N^2/4r)\) microprocessor-nodes. \(N \times N\) is the size of the picture and \(r\) is a resolution parameter. The HERMES kernel consists of four \(Z\)-adjacent processors in the whole HERMES array configuration. The internal design of the Kernel processors and the efficient way of their intercommunication are also discussed.
INTRODUCTION

The performance of the multiprocessor vision system architectures is based on the efficient internal design of their processors and the flexible connectivity of their kernels. It is well known that there is a great variety of multiprocessor vision architectures. However, the internal structural design of their processors and the processors connectivity seem to be similar for many of those multiprocessor architectures. In particular, many of the homogeneous multiprocessor architectures are based on a simple processor internal structural design such as adder or ALU, which processes on one bit operands at a time. The communication among these processors in the homogeneous structures is based strongly on the neighboring connectivity.

On the other hand, many of the heterogeneous multiprocessor architectures use a CPU as a processor. The connectivity among the processors of those architectures has many common features with the homogeneous ones, such as neighboring. Moreover, it presents some features appearing in the local area computer network configurations.

The goal of this paper is to deal with the internal structural design of the HERMES multiprocessor kernel and the connectivity of the processors included in the kernel. Note that, the HERMES multiprocessor kernel consists of four Z-adjacent processors, where the upper left of them in the array is their master-processor and the other three are the slave-processors.

This paper is organized into four sections. The second section discusses the Kernel internal structure and operation of two communication schemes (common-bus, parallel-buses). The third section compares these two communication schemes in a number of factors such as hardware complexity, and the last section summarizes the overall presentation.

KERNEL STRUCTURAL DESIGN AND OPERATION

HERMES Vision Machine

A brief introduction of the HERMES vision machine will assist the reader to understand the overall structure and operation of the kernel configuration. HERMES is a hierarchical, heterogeneous, real-time multiprocessor vision architecture, that has been designed and, its local and global operation has also been simulated by using Petri-net formal models. The horizontal organization of the HERMES machine is illustrated in Figure 1. HERMES consists of \((N^2/4)\) microprocessor-nodes in a two-dimensional array structure, where \(r\) is a resolution parameter.

The HERMES vision machine receives image data in parallel from the environment by using a two-dimensional photo-array of \(N \times N\) cells and processes them in a parallel-hierarchical (top-down and bottom-up) manner. Orders go down and abstracted picture information goes up along the HERMES hierarchy. In particular, the microprocessor-nodes process the available picture information in parallel, at the first level \((L0)\) of the HERMES hierarchy. At each of the following levels of the hierarchical processing a designated node ("the upper left" in each quartet of four Z-adjacent nodes) accumulates, correlates, synthesizes, and attempts to recognize the available picture information, feeding the results upwards, as shown in Figure 2. The designated full-master node of the HERMES architecture receives various commands from its users. It then makes decisions based on both its decision making algorithms and the built in "experience." If necessary, it also sends orders down to its successors in the hierarchy, thus determining the processing tasks that they have to execute.

Kernel Intercommunication Schemes

In this section, the structural design of the HERMES kernel will be described. Two variations on the initial HERMES
design of the busses, interconnecting the processors of the kernel, will be presented:

1. Common bus configuration
2. Parallel bus configuration

The selection between these two different implementations is dependent on some factors such as: total processing speed of the HERMES machine, total cost of the full design of the HERMES system, “real estate” considerations for a future VLSI realization, and architectural schemes of the “off-the-shelf” components (i.e., microprocessors) that are going to be used for the implementation of the HERMES system. In the following, the description of the two policies is given, while their comparison is presented in a third section.

Common bus structure

The basic idea of this scheme (see Figure 3) is the 8-line data bus which is coming out of each processor. In addition, these buses are connected together and to the main bus of the HERMES system. The selective transmission/reception of the master of a kernel is occurred using the individual switches of the processors. The direction of the flow of the information on the data bus is selected by the enable signals of the tri-state individuals switches. The manner in which the “opening” and the closing of switches work, is described in the architectural design of HERMES.

The data-bus is 8-lines wide because, not only does it represent one byte of information but, it also improves the communication procedure between the processors in the best case of the HERMES operation (i.e., when a region of a picture is of one gray level only). The data that are going to be transmitted or received in each processor (four processors per kernel) are latched on an 8-bit register (IN-OUT). The communication packages are packets of 8 bits. Once a communication session is initialized between two processors (master-slave), it has to be terminated before another session occurs.

Note that the 8-line data bus carries pure data without including control bits. Thus, the opening and the closing of a communication session is supervised by two control lines, namely: “one-way interrupt” line from a slave to the master and a “one-way acknowledgement-interrupt” line in the opposite direction. When the master wants to transmit to the slaves, there is the capability to transmit in parallel to all of the processors by opening the individual switches and at the same time interrupting them through the interrupt-acknowledgement line. This interrupt has the highest priority in the internal structure of the processors. As to what the structure of the overall HERMES architecture concerns, the same information is sent to each processor at the same time. However, there is the possibility for the master not to send information to a number of processors so that they remain idle. When a slave-processor wants to transmit information to the master, it enables its interrupt line and lets it be enabled until the last packet of information is sent to the master. In addition, the processor does not send the first packet until the master enables the interrupt-acknowledgement line. After the first packet is sent, the rest of the packets are transmitted in a synchronous manner.

On the other hand, the master has a service policy for the interrupts. The first interrupt that reaches the master is going to be serviced, and there is a fixed priority on the processors (i.e., Z-manner) in the case of the simultaneous existence of two or three interrupts. It is important to note, that any interrupt from a processor, higher in priority than the master, is of the highest priority among the three processors. The individual switch of the master is controlled by itself and by its master in a wider operational kernel (processor of higher priority). It goes without saying that the kernel structure described is compatible with the global structure of the HERMES machine.

From the collection of the Computer History Museum (www.computerhistory.org)
Parallel bus structure

The basic idea of this scheme (see Figure 4) is the existence of separate data buses connecting each slave to the master processor in the kernel. There is also a separate data bus for the communication of the master and the processor of the higher hierarchy. Moreover, there is no need of individual switches for each processor (including the master), since the master is capable of transmitting or receiving in a parallel manner.

The initialization and the termination of the transmission of the packets, from the standpoint of master, is set up by the interrupt and interrupt-acknowledgement lines, as discussed in the previous section. The only difference is the lack of a priority scheme in the master since transmission/reception to/from the slaves is done in a parallel manner. In addition, the transmission of packets is done in a synchronous manner for each slave separately. It goes without saying that there is a data IN-OUT register in each edge of each bus (8 bits) to latch or buffer the communication packets. Note that the slight difference of organization of the buses from the original HERMES machine design does not contradict the compatibility of this structure with the HERMES design.

Slave-Processor Design of the Kernel

The architectural scheme of each of the three processors of the kernel is depicted in Figure 5. As was mentioned in "HERMES—A Heterogeneous Multi-processor Machine Vision System," the horizontally microprogrammed approach was utilized. In this section the basic components and the organizational role of the slave-processors will be presented:

1. Four registers of 8-bit size will accept the pixels from the photoarray in the form of 8 bits.
2. A two to four decoder will select one of the above registers.
3. A fast PROM memory will keep the micro-routines of the microprocessor operation.
4. A microprogram sequencer will always provide the control memory with the address of the next memory word to be forwarded to the control register and the address of the next instruction located in the PROM to be executed. In addition, it will accept the interrupt-acknowledgement signal and will activate the appropriate service routine. It will also arrange the communication sessions with the master.
5. The control memory (look-up-table) will contain the control words required for the execution of the micro-instructions written in PROM.
6. A control register will keep track of the control words coming out of control memory. Each bit represents a control signal connected to a part or parts of the processor, as well as, the interrupt line and the control lines required for the passing of pixel values from the photoarray to the pixel registers.
7. An 8-bit data register, MDR, will keep one of the operands to ALU.
8. An 8-bit IN-OUT register will keep the byte of information that is going to be transmitted or received.

9. The X-register will keep the result of an ALU operation.

10. The ALU will operate on 8-bit operands and is supposed to be a simple and fast hardware component since it is going to perform only a few basic operations (such as add, subtract, compare, change-bit, and, or, xor, and move).

Some fast off-the-shelf logic components are going to be used to implement the above design. The microprogrammed design was preferred in order to have the flexibility to change the basic operation algorithm (stored in PROM) of the processor, according to the final selection of the off-the-shelf microprocessor type used to implement the master of the kernel.

COMPARISON BETWEEN THE KERNEL COMMUNICATION SCHEMES

In this section a comparison between the structural design of the two communication schemes for the HERMES multiprocessor kernel is realized. This comparison is based on a number of factors such as number of data communication lines per kernel-processor, number of control lines per kernel-processor, number of switches, kernel processing speed, total number of components in the kernel area, and type of operational mode.

Table I provides these comparative items giving a global idea about the kernel communication schemes.

Choosing one of the two schemes discussed in this paper requires consideration of several trade-offs, as shown in Table I.

CONCLUSIONS

The structural design of the HERMES multiprocessor kernel and its operation of the interprocessor communication have been presented. Two communications schemes were described as well as the internal architectural design of the kernel slave processors was discussed. Both the communication schemes of the kernel processors include advantages and disadvantages as to what hardware complexity and processing speed of the kernel concerns. The implementation of the HERMES vision machine, using the above two communication schemes, is in progress at George Mason University at the department of Electrical and Computer Engineering.

REFERENCES


<table>
<thead>
<tr>
<th>TABLE I</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparison items</td>
</tr>
<tr>
<td># of data lines per kernel</td>
</tr>
<tr>
<td>processor</td>
</tr>
<tr>
<td># of control lines per kernel processor</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td># of switches per kernel processor</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>kernel processing speed</td>
</tr>
<tr>
<td>total # of components in the kernel area</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>type of operational mode</td>
</tr>
</tbody>
</table>