Journal of Microcomputer Applications (1988) **11**, 155–167

A RISC-bit-sliced design of the HERMES multilevel vision system

Nikolaos G. Bourbakis, Dimitris Fotakis and Daniel Tabak

George Mason University, School of Information Technology and Engineering, Fairfax, VA 22030, USA

HERMES is a multibit, real time, systolic array architecture consisting of $[N/2^i] \times [N/2^i]$* processor-nodes, $0 \leqslant i \leqslant \log_2 N$, where $N \times N$ is the picture size. It receives the image data directly from the environment, using a 2-D photoarray of $N \times N$ cells and processes them in a parallel-hierarchical (bottom-up and top-down) manner without the use of a host computer for its function. This paper deals with the RISC-type structural design of the HERMES processor-nodes. The HERMES architecture is a heterogeneous one and consists of three structural levels of processors: low (8-bit processors), middle (16-bit processors) and high (32-bit processors). Here, we present the structure and operation of the 8-bit RISC processor which is used as an 8-bit slice for the overall HERMES construction. The use of two 8-bit processors for the structural design of a 16-bit processor is also discussed. The choice of the RISC processors provides HERMES with two significant advantages: (i) high processing speed of 10 MIPS or more per processor-node and (ii) reduced hardware complexity.

1. Introduction

In recent years, multiprocessor vision structures such as pyramids, arrays, cubes and trees claim the top of the computer vision area [1–4, 7–10, 12–15]. These structures can be separated into two main categories [1–3]. The first category, bit-serial processors, includes vision systems architectures with 2-D array configurations. These 2-D arrays of processors consist of a large number of identical processing elements (PE) which operate on a one-bit basis, and they are also called bit-serial array processors [3]. Although, their computational power is considerable, operating in a SIMD manner, these systems cannot by themselves deal with high level image analysis and recognition tasks [1, 3]. There is a need for a powerful host computer not only to do the miscellaneous operations (like I/O), but also to combine the data generated by the array processor [3, 16].

The second category, multibit-multiprocessor, includes vision architectures in various multiprocessor configurations. These systems consist of a number of microprocessor-nodes with identical or non-identical internal structure. Their main characteristic is the large number of bus-lines throughout the system. Thus, they are also called bus-oriented [2, 3, 7, 8]. The multibit-multiprocessor system architectures operate in SIMD and MIMD configurations. Some of them, such as GOP, PIPE, PICAP, FLIP use a host computer or a coprocessor for their function [1, 3, 4], while some others, such as HERMES, DIAS, HOMOGENEOUS, TALOS, etc. do not [1, 7, 8, 14].

*i is a resolution parameter.

0745-7138/88/030155+13 $03.00/0

© 1988 Academic Press Limited

The HERMES vision system discussed here, belongs to the heterogeneous multibit-multiprocessor category, [2]. It has a 2-D array topology of $[N^2/4^i]$, $0 \leqslant i \leqslant \log_2 N$ microprocessors. These microprocessor-nodes vary from simple ones (8-bit) to high level sophisticated processors (32-bit) [17–19]. This means that both primitive operations and high level vision tasks can be performed by HERMES [2, 3, 9, 12]. Another factor that reinforces the HERMES performance over the previously discussed systems is the neighbouring connectivity of $3k+1$ microprocessor-nodes neighbours, where k is the highest level at which the processor-node is able to operate [1, 3]. In addition, it is important to note that none of the systems mentioned previously has the parallel input capability of the image pixels directly from the environment. In contrast, HERMES is able to have direct access to the visual environment via a 2-D photoarray [2, 20]. Moreover, the pseudo-quadtree structure of the HERMES machine claims a number of features that the pyramids and the tree structures already have [2].

The first prototype of the HERMES multiprocessor machine was based on commercial microprocessor chips R-6502 [13]. However, that design did not explore the capabilities of the HERMES system. A RISC-bit-sliced structural design of the HERMES nodes is described in this paper using as a slice an 8-bit processor. The RISC design enhances the performance of the processor-nodes and reduces the size of the occupied area [12, 17–19].

This paper is organized into five sections. Section 2 describes in brief the overall organization of the HERMES machine. Section 3 presents the RISC design of the HERMES 8-bit processor, and the 16-bit sliced processor is discussed in brief. Section 4 presents the hardware features of the HERMES structure described in this paper versus the old one. Last section summarizes the overall presentation and proposes extensions for future work.

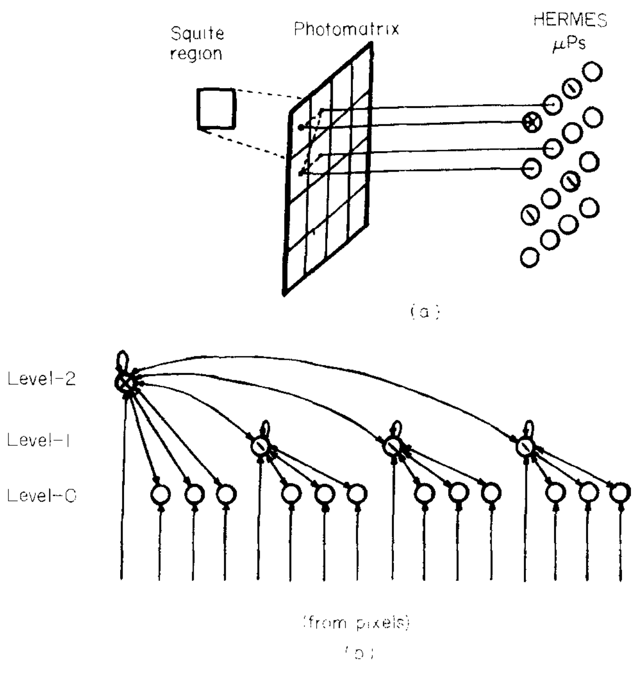

Figure 1. (a) HERMES 2-D array structure of 16 nodes. (b) Hierarchical operation of the HERMES nodes.

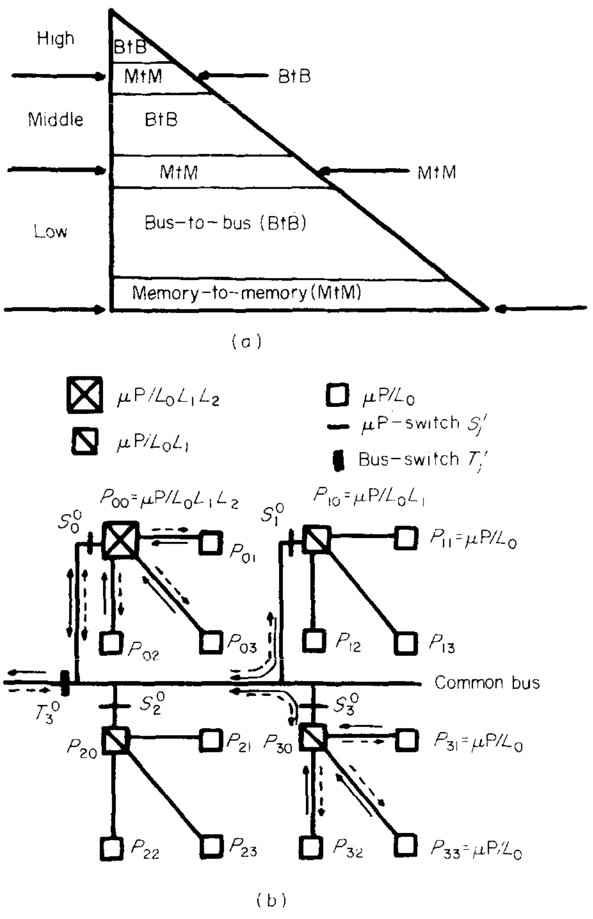

Figure 2. (a) Graphical representation of the three structural levels (low, middle, high) of the HERMES system and their MtM and BtB organization. (b) Bus-to-bus and Memory-to-memory organization of 16 microprocessor-nodes. The dashed lines indicate 'orders' and the solid lines indicate 'abstracted-code' picture information which flow along the HERMES hierarchy. Note that, $\mu P/L_0 \ldots L_k$ indicates a processor-node active at $k+1$ consecutive operational levels on the HERMES structure.

2. HERMES Multiprocessor organization

Definition: 'Abstracted picture information' is defined as the reduced amount of information without loss of its meaning by using coding or recognition methods producing a new description of it [21].

Definition: 'Orders' are defined as the decoded form of the user commands received by the master processor in order to determine the data processing tasks [21].

The overall structure of the HERMES vision machine is illustrated in figures 1–3. In particular, the horizontal 'bus-to-bus' and 'memory-to-memory' organization of the HERMES architecture [2, 13], for 16 nodes is given in Figure 2, while its vertical organization using analogue components, such as automatic gain control (AGC) devices, analogue to digital converters (A/D) etc., is provided in Figure 3.

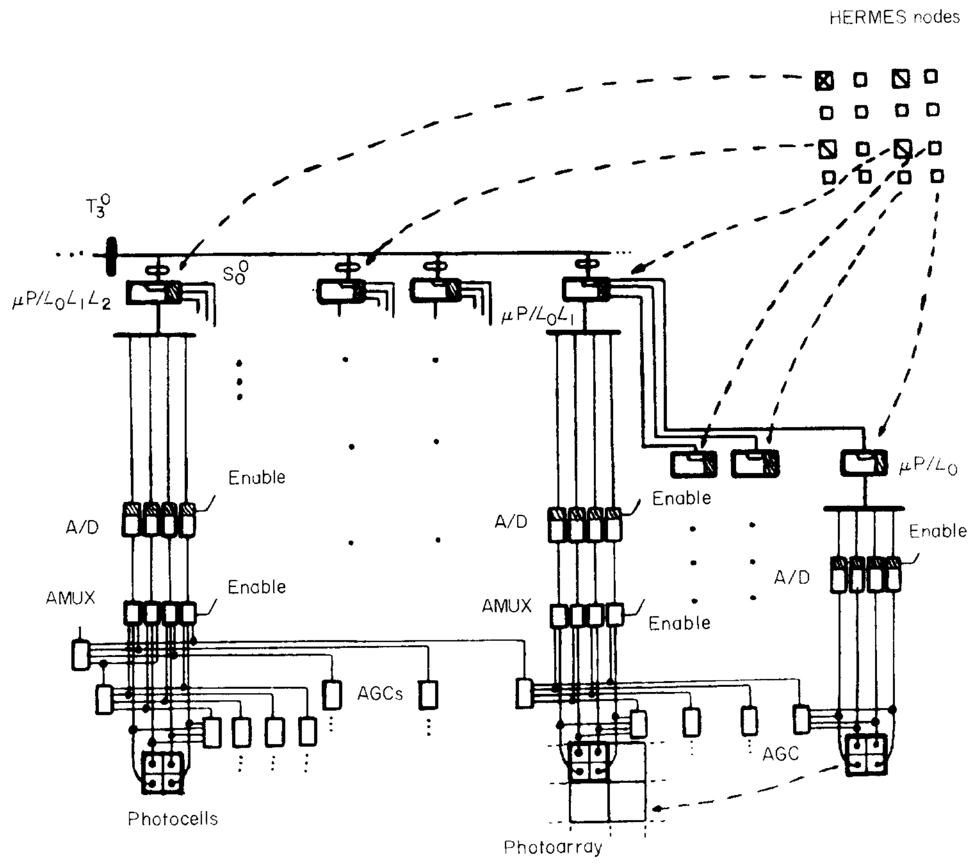

Figure 3. The vertical hardware organization of the HERMES processor-nodes, for 16 nodes.

The HERMES machine receives image data in parallel from the environment, by using a 2-D photoarray of $N \times N$ cells, and processes them in a parallel-hierarchical (bottom-up and top-down) manner, by opening and/or closing of its switches [1, 2]. The microprocessor-master-nodes undertake the opening and closing of their slave-nodes switches [2, 13], which allow the information to flow through the HERMES hierarchy. In particular, during a bottom-up procedure 'abstracted-coded' picture information goes up along the HERMES hierarchy, while during a top-down procedure 'orders' go down from the father-nodes to their son-nodes [2, 13]. In particular, all of the microprocessor-nodes process the available picture information in parallel, at the L_0 level of the HERMES hierarchy. At each of the following levels of processing, along the HERMES hierarchy, a designated node ('the upper left' in each quartet of adjacent nodes) accumulates, correlates, synthesizes and attempts to recognize the available picture information, feeding the results upwards, as shown in Figures 1 and 2. The full-master designated node of the HERMES architecture receives various commands from its users. It then makes decisions based on its 'decision making algorithms' and built up experience, and if necessary, it sends down 'orders' to its successors, thus determining the processing tasks that they have to be executed.

3. RISC design of the HERMES bit-sliced processor-nodes

The HERMES operation is based on three structural levels of RISC-type processors (low 8-bit, middle 16-bit, and high 32-bit) as illustrated in Figure 2(a) and in Table 1 (for $N=256$ and $i=1$).

The hardware realization of the RISC-type processors will be done in a bit-sliced manner of 8-bit chips. Thus, at the lower, 8-bit structural level, a single chip per processor will be used. At the middle structural level [Figure 2(a)] of 16-bits: two chips per processor, and at the high, 32-bit structural level—four chips per processor-node, noting that there are only four processor-nodes (32-bit) at the high structural level of the HERMES organization. The use of a single type of chip (8-bit slice of the RISC-type processor) would simplify the construction of the HERMES system and reduce the overall cost of design, testing and manufacturing.

Each structural level includes processor-nodes with equivalent capabilities and same internal structural design. Note that, at each of these structural levels different processing tasks are applied on the 'picture information' accumulated on them. In particular, at the low structural level, low level vision tasks such as convolution (partial), edge detection (partial), region analysis, Freeman coding, etc. are applied. At the middle and high structural levels appropriate vision tasks are also used, such as statistical and syntactical recognition tasks and relational knowledge-based operations [21]. In this section the structural design of the 8-bit processor is described, which is the basic slice for the design of the middle and high level processors.

Table 1.

Structural levels	Operational levels	No. of processors per operational level	Type of communication	Total number of processors per structural level	Type of processors
High	8	1			
			BtB	4 (∼0·02%)	32-bit
	7	3			
			MtM		
	6	12			
			BtB		
Middle	5	48		252 (∼0·38%)	16-bit
			BtB		
	4	192			
			MtM		
	3	768			
			BtB		
	2	3072			
Low			BtB	65280 (∼99·6%)	8-bit
	1	12,288			
			MtM		
	0	49,152			

$N=256$, $i=1$.

3.1 *Low level processor-node* (*LLPN*)

The low level processor-node, at the L_i level, $0 \leqslant i \leqslant p$, $p \in \mathbf{Z}^+$, $p \geqslant 3$, is considered to be a microprocessor with simple hardware structure and high processing speed. The hardware simplicity is essential since a large number of these LLPN processors (more than 98% of the entire structure) will be needed for the complete fabrication of the HERMES vision machine, even more for its future VLSI realization [2, 12]. The high speed requirement is also basic for the whole system design, since quick responses from the master-nodes to slave-nodes allow a real-time processing of the picture taken through photoarray [20, 22].

In this design, a RISC-type architectural scheme was chosen to provide the simplicity and the speed to the 8-bit slice processor-nodes [12, 17–19, 23]. Note that, each LLPN processor is less powerful than the existing RISC machines [19], since it is only an 8-bit processor mostly appropriate for the HERMES vision machine [12]. On the other hand, more powerful RISC-type processors of 16-bit and 32-bit will be used at the middle and high structural levels of the HERMES structure.

All the LLPN instructions are 16-bit long, and each instruction is primitive enough to be executed in a single cycle [12]. If the current bipolar technology is used [23], a clock of 10 MHz will be used, thus, a performance of 10 MIPS can be guaranteed. The block diagram of the low-level processor-node is shown in Figure 4. A 1 k × 16 ROM is used to store the most used programs for the operational tasks on the pixels and later 'abstracted-picture' information [1, 2, 9]. That is why there is no RAM memory inside the LLPN node to store programs since all the algorithms are written in ROM. However, there are exceptions, for the master-nodes at the higher operational levels of the HERMES hierarchy. In particular, at each level L_i, $i > 0$, of the HERMES hierarchy a RAM memory can be used in order to store, either intermediate picture information accumulated at the master-nodes, or special algorithms when bit-sliced components are used, and/or consecutive images for multi-image processing. This RAM memory can be outside of the processors structure.

In the LLPN diagram, there is also a 64×8 bits register-file to store the intermediate results and 'orders' from the master-nodes. This is enough space, keeping in mind that the slave-nodes (L_0) do not perform complicated tasks and they deal only with a particular picture region ('squite') of $2^i \times 2^i$ pixels, $0 \leqslant i \leqslant k$, $k \geqslant 3$. The control unit of the processor-nodes is microcoded. This means that each instruction of the LLPN corresponds to a 48-bit control word. All the control words are stored in a 16×48 bit control store PROM (look-up table [12]). The field of four most significant bits of each machine instruction is the op. code and at the same time a direct address to the 16×48 look-up table. No decoding is needed. The short access control store produces practically immediately (within even 30 ns) a 48-bit, horizontal word, capable of activating up to 48 control lines simultaneously. In this way a streamlined microcoded-based instruction-execution is achieved. The control lines are connected directly to all parts of the LLPN, thus, the appropriate combination of 0's and 1's will guarantee the execution of a certain instruction. Among the control lines, there is the interrupt line which is used to set up communication with the master-node of each quartet of processors. Besides, the streamlined execution, the design benefits from the usual microcode advantages of greater flexibility for modification and error recovery, while maintaining a high speed of instruction handling [12, 17, 18].

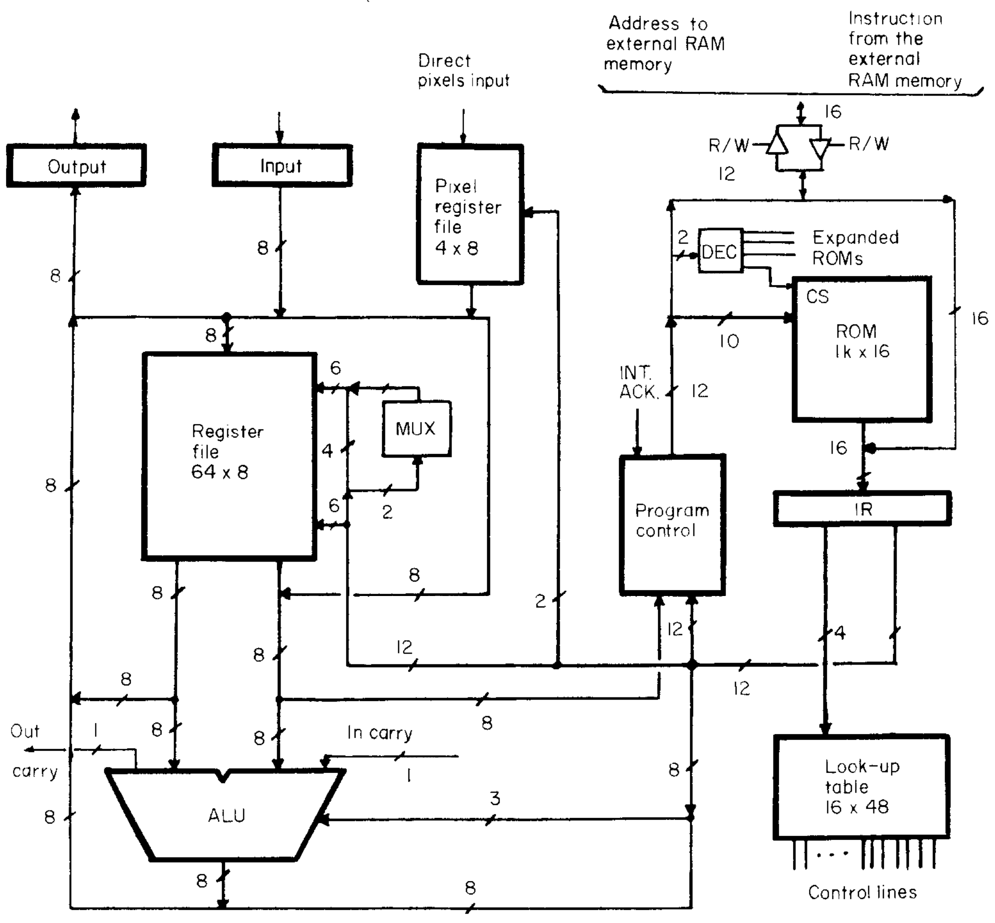

Figure 4. Internal structural design of the LLPN.

There is a two-stage instruction pipeline. In particular, while an instruction $(i-1)$ is being executed by the ALU, the next (i) instruction is being fetched and decoded. Note that, the only data format recognized by the LLPN processors is the 8-bit (byte). The instruction set includes 16 standard instructions according to the 4-bit op. code field of the instruction (see Table 2). However, there is an extra field in the SHF instruction for different types of shift operations, that increases the instruction set to 23 without increasing the size of the look-up table. The shifting operations were taken in consideration because they are necessary for any possible advanced computation (i.e. multiplication).

The ROM which is considered to be the program memory and the external RAM memory for the master-nodes, are accessed by three addressing modes: (i) Immediate, (ii) 12-bit direct, and (iii) Register indirect. In the register indirect addressing mode, the 8 most significant bits of the 12-bit addresses are provided by one of the registers, while the 4 least significant bits are zeroes. In this case, not all the memory locations are accessed

Table 2. *Instruction set*

Assembly notation			Operation	Description
1.	ADD	Rd, Rs	$Rd \leftarrow Rd + Rs$	Integer Add
2.	MOVIN		(INPUT)←RAM	Move data from RAM to INPUT register
3.	NAND	Rd, Rs	$Rd \leftarrow \overline{Rd \wedge Rs}$	Logical NAND
4.	ADDP	Rd, Ps	$Rd \leftarrow Rd + Ps$	Integer Add with Pixel value
5.	MOVOUT		RAM←(OUTPUT)	Move the contents of OUTPUT register to RAM memory
6.	NADP	Rd, Ps	$Rd \leftarrow \overline{Rd \wedge Ps}$	Logical NAND with Pixel value
7.	MOV	Rd, Rs	$Rd \leftarrow Rs$	Move from register
8.	JMP	M	$PC \leftarrow M$	Jump unconditionally
9.	MOVI	Ri, #A	$Ri \leftarrow A$	Move immediate
10.	IN	Rd	Rd←(INPUT)	Move from the input register
11.	OUT	Rs	(OUTPUT)←Rs	Move to the output register
12.	JMZ	M	$PC \leftarrow M$	Jump on zero result
13.	JMN	M	$PC \leftarrow M$	Jump on negative result
14.	JMP	(Rm)	PC←(Rm)	Jump indirect unconditionally using the contents of Rm with 4 zeroes in the LSB position
15.	NOP			No operation
16.	SHF	Rd, #S	Rd←Rd shifted	Shift operations according to S where,
			S *shift operation*	
			000 : shift left	
			001 : shift left through carry	
			010 : Shift right	
			011 : shift right through carry	
			100 : Rotate left	
			101 : Rotate right	
			110 :	
			111 :	

Comments
Rd, Rs, Rm – registers from the register file.
Ps—register from the pixel register file.
Ri—register from the 16 lower-address registers of the register file.
A—immediate 8-bit constant.
S—3-bit constant.
M—Memory address.

using this addressing mode. However, the memory space is divided into banks of 256×16 which could be used to store algorithms run by the LLPN processors. Due to this addressing mode, the master-node of each quartet of processors could send an 8-bit number to its slave-nodes which could be used directly as a starting address of a procedure in the ROM memory of the chosen LLPN. Moreover, there is the possibility for an 8-bit immediate value to be moved from the register-file to the 16 lower address registers.

A NAND operation is used to implement any other logic function, such as AND, OR, NOT, etc., by simply repeating the NAND instruction a number of times. The LLPN architecture recognizes the following six formats for the operand address field of each instruction [12], as illustrated in Figure 5.

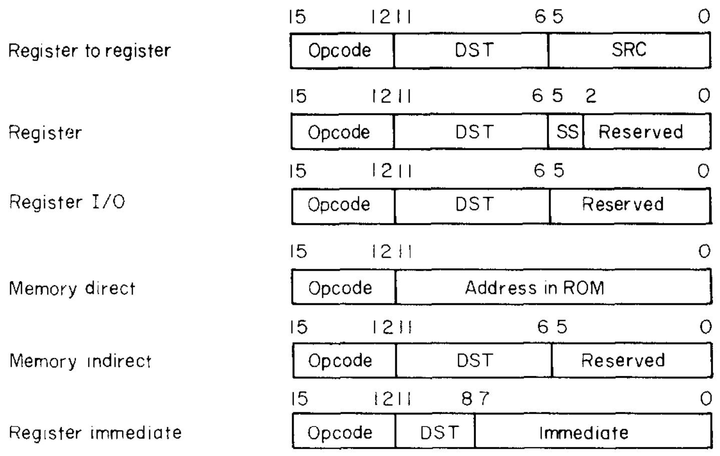

Figure 5. The formats of the LLPN instructions.

The register fields are 6 bits wide. This means that 64 registers can be accessed each time. The register-file is implemented as a dual memory in which the DST (destination) and the SRC (source) can be accessed simultaneously and the result can be stored back into the DST register. Using the current technology [23] the above execution procedure can be done within one CPU cycle (100 ns or less) on custom VLSI implementation.

3.2. *Middle level processor-node* (*MLPN*)

The 16-bit processor which will be used in the middle structural level of the HERMES system hierarchy, is composed of two identical 8-bit processor-slices, as shown in Figure 6. These slices are exactly the same as the 8-bit low level processor node (LLPN). Each

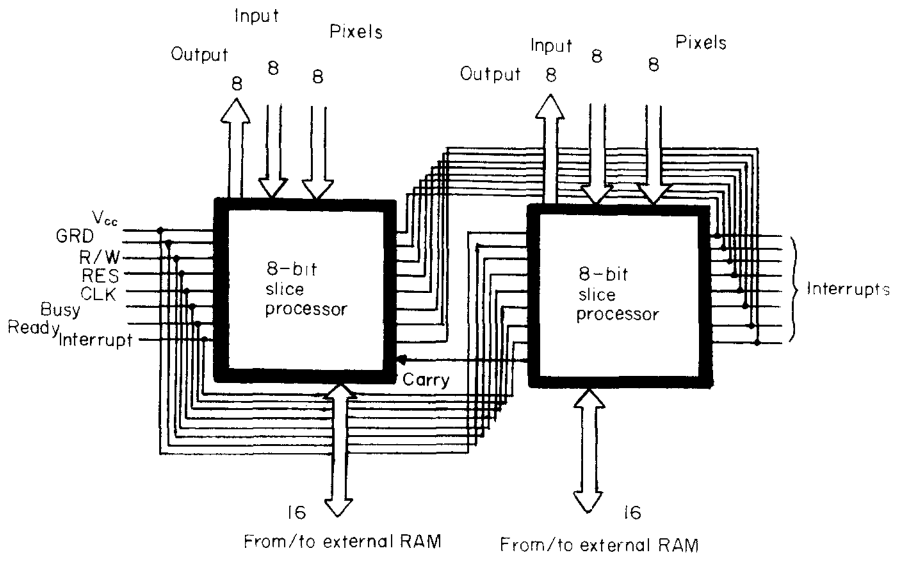

Figure 6. A global configuration of the 16-bit-sliced processor node.

slice can operate on 8-bit data independently of the other slice. Therefore, there is an instruction parallelism within the 16-bit processor, and since there are separate control units in the slices, one could say that there is program parallelism, as well. On the other hand, the processor can operate on 16-bit data, where 8-bit data per slice are used. One reason to have this independent function of the two slices is the fact that a 16-bit processor communicates with a set of 16-bit processors via a bus-to-bus mode, as well as, with a set of 8-bit processors via a memory-to-memory mode in the HERMES hierarchy.

The size of each program memory (ROM) is 1 k × 16/slice, while it can be expanded to a total of 4 k × 16. Also, where there is only a 64 × 8 bit register file as the data memory in each processor-slice, an external RAM memory of up to 4 k × 8/slice can be used.

The instruction set consists of 23 instructions most of them being memory or I/O oriented. More explicitly, there are 2 arithmetic, 2 logic, 6 move and I/O, 4 memory related, 1 no operation, and 6 bit operations, as shown in Table 2. Although there is a lack of logic and arithmetic instructions, there are several powerful move, I/O, and memory type instructions in order to support efficiently the communication among the same or different processor-nodes at the different functional levels of the HERMES hierarchy. The small number of instructions selected is sufficient to handle efficiently the envisioned tasks of the proposed system. Following the same structural philosophy, the 32-bit processor is composed of four 8-bit slices. The detailed design of the 32-bit processor will be provided in future work. Figure 7 shows a global design of a 32-bit processor.

4. HERMES hardware features

In this section, we present a comparison between the previous HERMES design, based on commercial microprocessor chips R-6502, and the new one based on RISC-bit-sliced processors, 8-bit/slice. The comparison discussed here, deals with the hardware characteristics of these two HERMES configurations, as shown in Table 3, and explains the reasons for the selection of the new RISC-type design of the HERMES processor nodes.

5. Conclusions

The detailed structural design of the HERMES low level processors, based on a RISC-type architecture, has been described in this paper. In particular, the internal structural

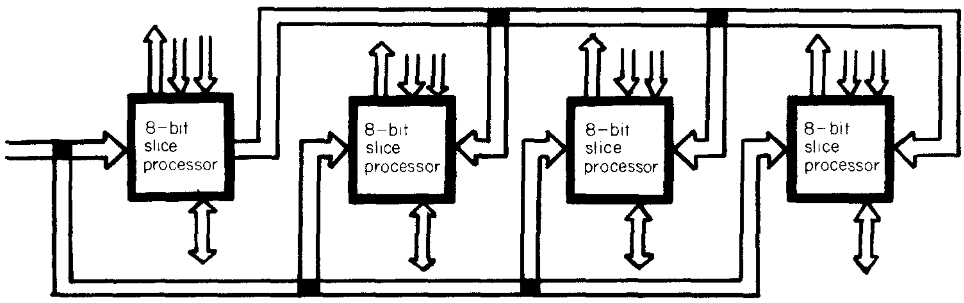

Figure 7. A global configuration of the 32-bit sliced processor node.

Table 3.

Items	Previous HERMES design	New HERMES design
Type of nodes	R-6502 (8-bit)	RISC (8-16-32) bit-sliced
Overall organization	Memory-to-memory [13]	Bus-to-bus with switches and memory-to-memory
Type of buses used	Parallel-buses	Common and parallel-buses
Processor size	Medium [13]	Small [17–19]
Processing speed	Slow	High (10 MIPS)
Memory size per μP	8 kbyte (RAM)	2 kbyte (ROM) and 8 kbyte (RAM)
Type of instructions	Variable (1–3 bytes)	Fixed (16 bits)
No. of instruction of the processor-node	56 (151)	16 (21) low level
Direct processing of pixels	No	Yes
Register-file	No	Yes (64 × 8)
No. of addressing modes	13	3
Logic instructions	AND, BIT, OR, XOR	NAND
Control unit per processor-node	Instruction decoding circuit	Fast, horizontally microcoded look-up table
No. of pins per processor node		
(a) LLPN	40 pins	64 pins
(b) MLPN	40 pins	96 pins
(c) HLPN	40 pins	176 pins

design of the LLPN processor has been presented. The LLPN nodes constitute more than 98% of the HERMES structure and their size is very critical for the entire system construction.

In addition, a comparison between the previous HERMES design and the new one was given in regards with the hardware organization. Extensions of this research effort are: (i) the construction of the HERMES system using RISC-type processors, (ii) fault tolerance of the HERMES architecture, (iii) performance evaluation of the HERMES system, and (iv) development of a multilevel operating system (MYLOS) for the efficient function of HERMES. The above research efforts are in progress.

References

1. N. G. Bourbakis, D. Fotakis & D. Tabak 1987. On data flow based functional model for the multiprocessor HERMES vision machine. *Proceedings IEEE Conference on Supercomputing*, May, S. Clara, CA. Vol. III, pp. 314–323.
2. N. G. Bourbakis 1982. Design of real-time supercomputing vision architectures. *Proceedings IEEE Conference on Supercomputing*, May, S. Clara, CA. Vol. I, pp. 392–398.
3. E. P. Danielson & S. Levialdi 1981. Computers architectures for pictorial information systems, *IEEE Computer*, Nov. 53–67.
4. G. H. Granland 1981. GOP image processor. Picture processing Lab, Linkoeping University, Sweden, TR-1981.
5. G. J. Li & B. W. Wah 1985. The design of optimal systolic arrays. *IEEE Transactions on Computers*, **34** (1), 66–77.

6. J. L. Hennessy 1984. VLSI processor architecture. *IEEE Transactions on Computers*, **33** (12), 1221–1246.
7. N. Dimopoulos 1985. On the structure of the HOMOGENEOUS multiprocessor. *IEEE Transactions on Computers*, **34** (2), 141–150.
8. N. G. Bourbakis & H. Nguyen 1987. TALOS—A distributed image analysis/synthesis multiprocessor system. *Proceedings IEEE Conference on IECON*, Cambridge, MA. Nov.
9. N. G. Bourbakis & P. A. Ligomenides 1986. A real-time, multimicroprocessor vision system. *Proceedings IEEE Conference on CVPR*, June, Miami, FL, pp. 381–387.
10. G. Fritsch *et al.* 1983. EMS-85 The Erlangen multiprocessor system for board spectrum of applications. *Proceedings IEEE Conference on Supercomputers.*
11. C. L. Seitz 1984. Concurrent VLSI architectures. *IEEE Transactions on Computers*, **33** (12), 1247–1265.
12. D. Fotakis & N. Bourbakis 1987. A RISC-type structural design of the HERMES multiprocessor kernel. *Proceedings International Conference on Supercomputing*, June, Athens, Greece, pp. 1011–1030.
13. N. G. Bourbakis & C. Vaitsos 1985. A multimicroprocessor tree network configuration used on robot vision systems. In *Digital Techniques* (S. Tzafestas, ed.) pp. 483–490. Amsterdam: Elsevier.
14. N. G. Bourbakis & P. Ligomenides 1985. High performance architectures for real-time multilevel picture information systems. *Proceedings IEEE Workshop on LFA*, June, Mallorca, Spain, pp. 271–276.
15. M. Duff 1986. *Intermediate Level Image Processing*. New York: Academic Press.
16. M. Duff 1983. *Computing Structures for Image Processing*. New York: Academic Press.
17. D. K. DuBose, D. Fotakis & D. Tabak 1986. A microcode RISC. *Computer Architecture News*, **14** (3), 5–16.
18. D. Tabak 1987. *Reduced Instruction Set Computer RISC-Architecture*. Research. New York: Studies Press and Wiley.
19. D. A. Patterson & C. H. Sequin 1982. A VLSI RISC. *IEEE Computer*, **15** (9), 8–21.
20. D. Panagiotopoulos & N. Bourbakis 1984. The VLSI design of a 2-D image processing array. *International Journal on Microprogramming and Microprocessing*, **14**, 125–132.
21. N. G. Bourbakis *et al.* 1987. KALYPSO—A picture recognition and knowledge creation language. GMU-ECE-TR-1987.
22. N. G. Bourbakis *et al.* 1987. A hardware implementation of a parallel 2-D photoarray, *Proceedings IEEE Conference on IECON*, Cambridge, MA, Nov. Vol. 856, pp. 616–620.
23. Bipolar microprocessor logic and interface. *AM2900 Family 1985 Databook*.

Nikolaos G. Bourbakis received the bachelor degree in mathematics in 1974 from the University of Athens, Greece. During 1978–84 he received a certification in electrical engineering from Patras University, Greece. Also, during 1978–82 he received the PhD degree from the Computer Engineering Department at Patras University. He was an Assistant Professor of computer engineering at Patras University during the period 1983–84. In 1984 he became an assistant professor of electrical and computer engineering at George Mason University.

His major research interests are in: real-time multiprocessor vision system architectures; expert systems with emphasis to text processing/analysis systems; multilevel systems simulation; multi-operating systems; computer languages (UAL); vision languages (SCAN, TOPOTHESIA, KALYPSO); algorithms for picture processing; learning algorithmic schemes and AI; 3-D robot vision; knowledge driven VLSI architectural system design (GEOMETRIA).

He has published more than 60 articles in international conferences and journals and he is author of three books (in Greek). He is reviewer in numerous international journals, books, and research proposals. He was session chairman and member in a number of international conferences and co-editor of the 1st Informatics Conference Proceedings, 1984. He is a Research Associate with the Computer Technology Institute, Patras, Greece.

Dimitris Fotakis received his BS in computer engineering in 1985 from the University of Patras. He also received his Master degree in electrical and computer engineering, in 1986, from the George Mason University, VA. He is a graduate student in the PhD program of the GMU. His research interests are in multiprocessor system architectures, simulation and algorithms.

Daniel Tabak received his PhD in electrical engineering at the University of Illinois, Urbana in 1967. While at the University of Illinois he has been employed as a graduate assistant on the ILLIAC II and ILLIAC IV projects. After completing his PhD he has been employed both in industry, GE Co., Wolf R&D (EG&G), and academia—RPI, Ben Gurion University, University of Texas, University of CA, Berkeley, and Boston University. Since 1985, Dr Tabak has been a professor of electrical and computer engineering at the George Mason University. During his career Dr Tabak has conducted research and published extensively in the areas of computer-based applications, process control, parallel processing, computer architecture and microprocessors. He has just published his second book: *RISC Architecture* (Wiley, 1987). Dr Tabak is now working on an extensive article on multiprocessors, which will be extended into a book later. Dr Tabak is a senior member of the IEEE and a member of the ACM, Eta Kappa nu, Sigma Xi and Euromicro. He is a Director of Euromicro, representing the USA.