Multiprocessor vision system

Bourbakis, N.G.; Papazoglou, M.; Alexiou, G.A.

Published in:
Microprocessors and Microsystems

Publication date:
1990

Multiprocessor vision system

Nikolaos G Bourbakis, Mike Papazoglou* and George Alexiou† detail the design and operation of a multibit multiprocessor tree network architecture employed as a computer vision system. The three-level hierarchy allows parallelism and pipelining

The paper presents the architecture and functionality of a multiprocessor tree network (MTN) vision system. The design of the MTN is based on 21 R-6502 8-bit microprocessors connected in a quadtree structure. Under this topology the microprocessor modules communicate through a memory-to-memory Multibus configuration. The MTN operates in parallel hierarchical (bottom-up and top-down) manner, where orders go down and abstracted picture information goes up within the MTN hierarchy.

microsystems multiprocessors computer vision

A plethora of vision system architectures has been proposed, designed and/or implemented during recent years[1-5]. They may be divided into two main categories:

- Bit-serial-array-processors (BSAP). BSAPs encompass architectures based on processors that process only a single bit at a time. These architectures also use a host computer for the control, synchronization and I/O operations of the entire system topology[4].
- Multibit multiprocessor networks (MMN). MMNs comprise architectures based on multi-CPU components. This feature provides autonomy and flexibility[5], but this architecture increases complexity and implementation costs.

This paper presents the design of a multibit multiprocessor tree network architecture (MTN) which is used as a computer vision system. The MTN structure is an MMN system based on 21 R-6502 8-bit microprocessors in a quadtree configuration. The MTN processors are configured in three consecutive levels (l_0, l_1, l_2), as shown in Figures 1 and 2.

The processors at the l_0 level receive, in parallel, the picture information from the environment; they process it and forward it to the higher (l_1) level in the MTN system, in an abstracted form. A similar process is repeated at the next (l_1) level and again at the master processor level at the top of the quadtree structure. This level takes the final decisions about the received image. A significant feature of the MTN system is that it is based on the flexibility and independence provided by the quadtree structure[6]. It is not intended to discuss the processing tasks undertaken by the MTN configuration here; rather, discussion will be confined to the description of the architectural design and the overall functionality of the MTN system.

This paper is organized in four sections. The section which follows deals with the MTN design, in particular discussing the general configuration of the MTN structure and focusing on the internal design of the MTN processors. The final sections describe the top-down and bottom-up operations of the MTN system.

MTN STRUCTURE

General configuration

The MTN is a complete quadtree, which consists of $[(4N^2/4^i) - 1]/3$ processor nodes, where $N \times N$ is the size of the image received from the environment, using a two-dimensional photoarray of $N \times N$ cells, and i is the resolution parameter[5]. The MTN consists of $k + 1$ hierarchical structural levels, where $k = \log_2(N)$ and each

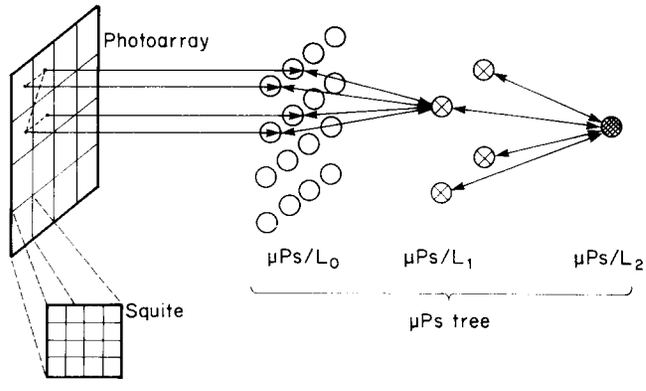


Figure 1. Parallel picture input from the external environment using a photoarray

IBM, 5600 Cottle Road, San Jose, CA, USA
*National Australian University, Department of Computer Science, PO Box 4, ATC 2601, Canberra, Australia
†University of Patras, Department of Computer Engineering, 26500 Patras, Greece
Paper received: 23 June 1989. Final revision: 30 July 1990

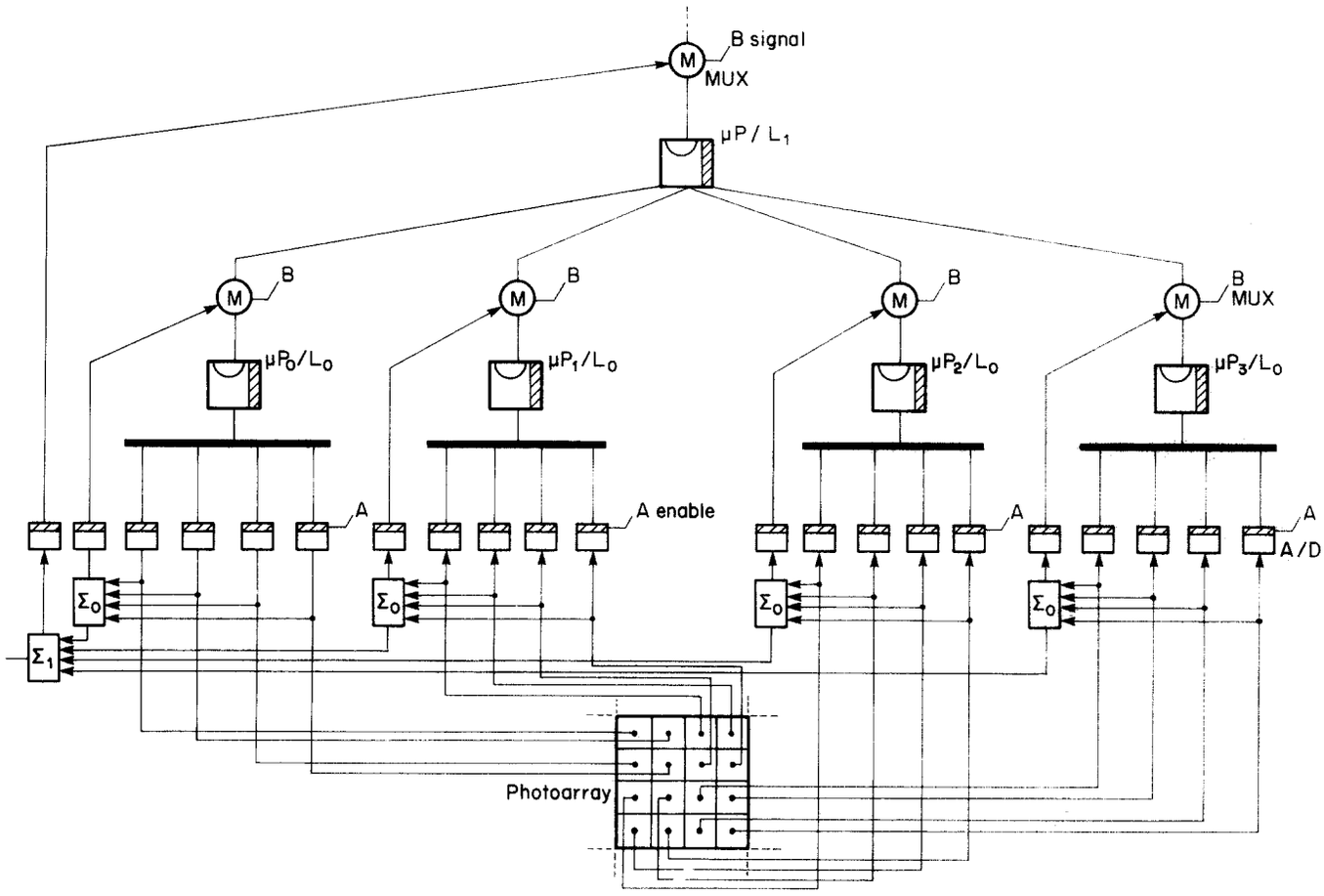


Figure 2. General configuration of the MTN system

level (L_k) contains ($N^2/4^{(k+i)}$) processor nodes (e.g. for $k = 0$ the level includes N^2 processors). Additionally, each processor node at the L_i level ($0 < i < k$) has four successor nodes and only a single parent node. A simple 21-node MTN system is illustrated in Figure 1, where $\mu P s I_i$ denotes the number of processors per level I_i, and squite is a square region of pixels. Each squite receives the image data corresponding to exactly one square region of the photoarray.

At the I_0 level of the MTN hierarchy, the $N^2/4^i$ processor nodes receive the picture data in parallel from the environment via a photoarray, as shown in Figure 2. This means that each processor node, at the I_0 level, is dedicated to a specific picture region of the $N^2/4^k$ pixels, as shown in Figure 1. The partitioning of the picture in square regions with size $N^2/4^k$ is required by the form of the quadtree structure, whose prime purpose is to retain the neighbouring properties, such as the shortest distance between neighbouring image areas. The quadtree structure contains more integrated information about the image than any other picture partitioning schemata, such as line or rectangular schemata[5].

Structural processor design

Here, the detailed design of a specific MTN structure of 21 R-6502 processors is presented.

The 16 processor leaf nodes (in level I_0) receive the image data directly from their corresponding square regions of ($N^2/4^2$) pixels. The MTN has only one master/parent node, which resides at the root of the quadtree and assumes responsibility for the final decisions concerning obtained image data and the required output operations (see Figure 1).

The MTN is a heterogeneous multiprocessor architecture, which implies that its processor nodes may have different internal structures depending on the particular functions they perform. The MTN tree structure contains three different types of processor nodes:

- full master node (FM),
- master-slave nodes (MS),
- slave leaf nodes (S).

These processor nodes are organized in a hierarchical manner, as shown in Figure 3.

Full master (FM) processor node

The full master (FM) processor node consists of an R-6502 CPU, 8k memory, three target selecting devices (TSD1, TSD2 and TSD3) and four 2 × 4 decoders. Its overall design is illustrated in Figure 4.

The TSD1 is a selection device which receives a selection signal from the R-6502 processor and subsequently directs its data channel either to its corresponding memory module or to the TSD1 device of the FM node. This device can connect its data channel to the slave memories through their corresponding TSD1 modules (see Table 1 and Figure 5).

The TSD2 is a selection device to TSD1, which receives a selection signal from the R-6502 processor and directs its address channel either to its memory, or to the TSD2 of the FM node. The TSD2 can connect its address channel

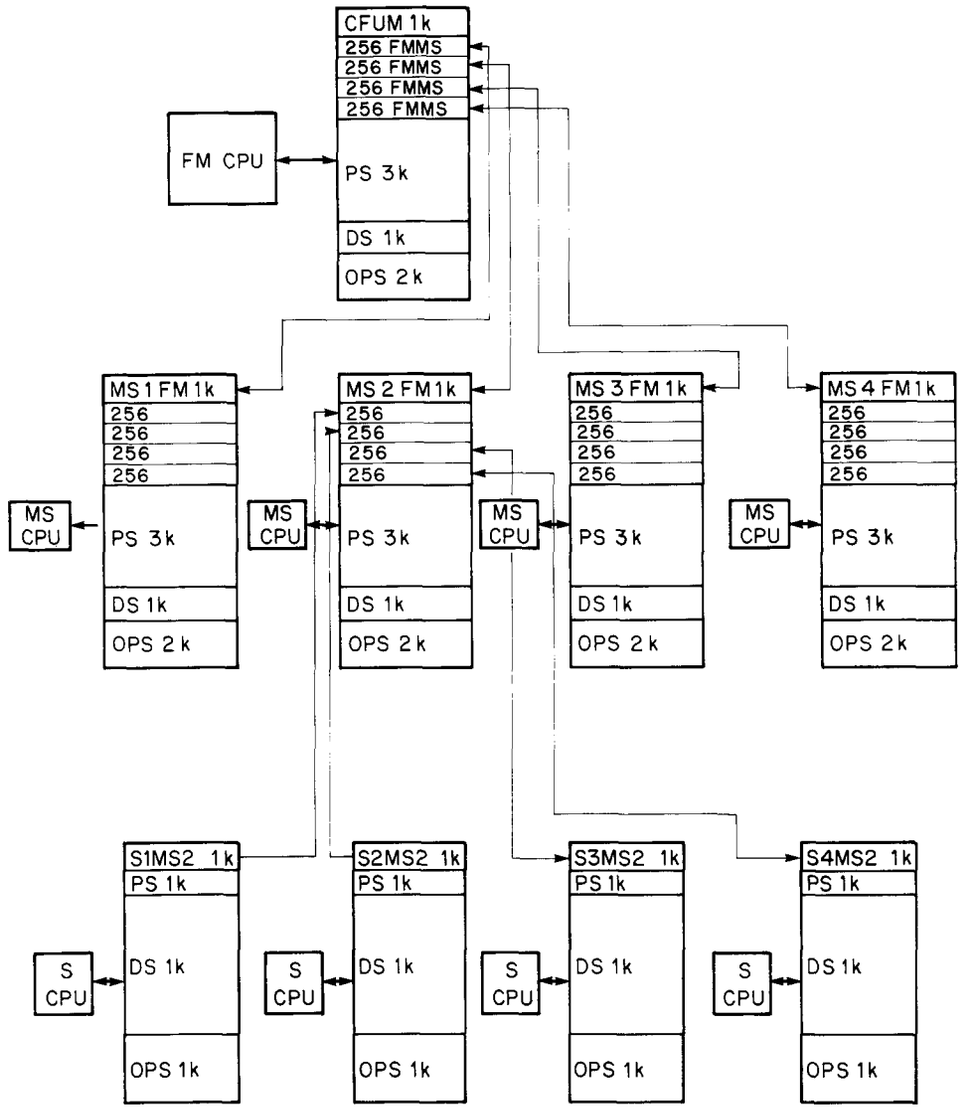


Figure 3. Hierarchical organization of the MTN configuration

to the slave memories through their corresponding TSD1 modules (see Table 1 and Figure 6). In addition, the TSD2 device directs the address channel signals as an input to TSD3.

The TSD3 device receives signals from the TSD2 and outputs operation signals or special commands which are directed either to its corresponding CPU, or the CPUs of the FM node or its slave nodes (see Figure 7). Details of the internal connection of the processor nodes, using the above devices, are given in Reference 6. The R-6502 is a well known 8-bit memory-mapped I/O CPU[7].

The 8k memory is divided into five segments, as shown in Figures 3 and 8 and detailed below.

- Common full master user memory (CFUM) has a size of 1k and is used for communication between users and the FM node. User-initiated commands and new algorithms are stored in the CFUM module and FM node results are output to users through CFUM.
- Common full master master-slave (FMMS) memory has a size of 1k and is used for communication between the FM node and its successor MS nodes. FMMS memory space is divided into four subsegments of 256 byte each, each dedicated to an MS node. Through FMMS memory the FM node sends data, commands, and algorithms to its successor MS nodes, and the four MS nodes send data to the FM node.
- Program space (PS) has a size of 3k and is used by the FM node to store its programs for data processing.
- Data space (DS), with size 1k, is dedicated to storing data or intermediate results.
- Operating system space (OPS) is the area where the operating system or the monitor program resides.

Master-slave (MS) processor node

The MS node consists of an R-6502 CPU, 8k memory and the same additional TSD devices as the FM node (see

Table 1. Data/address channel connections of the current processor

A15	A14	Connections
0	0	AB and DB to current processor memory
0	1	AB and DB to master processor memory
1	0	AB and DB to slave processor memory
1	1	AB to TSD3

AB: address bus
DB: data bus

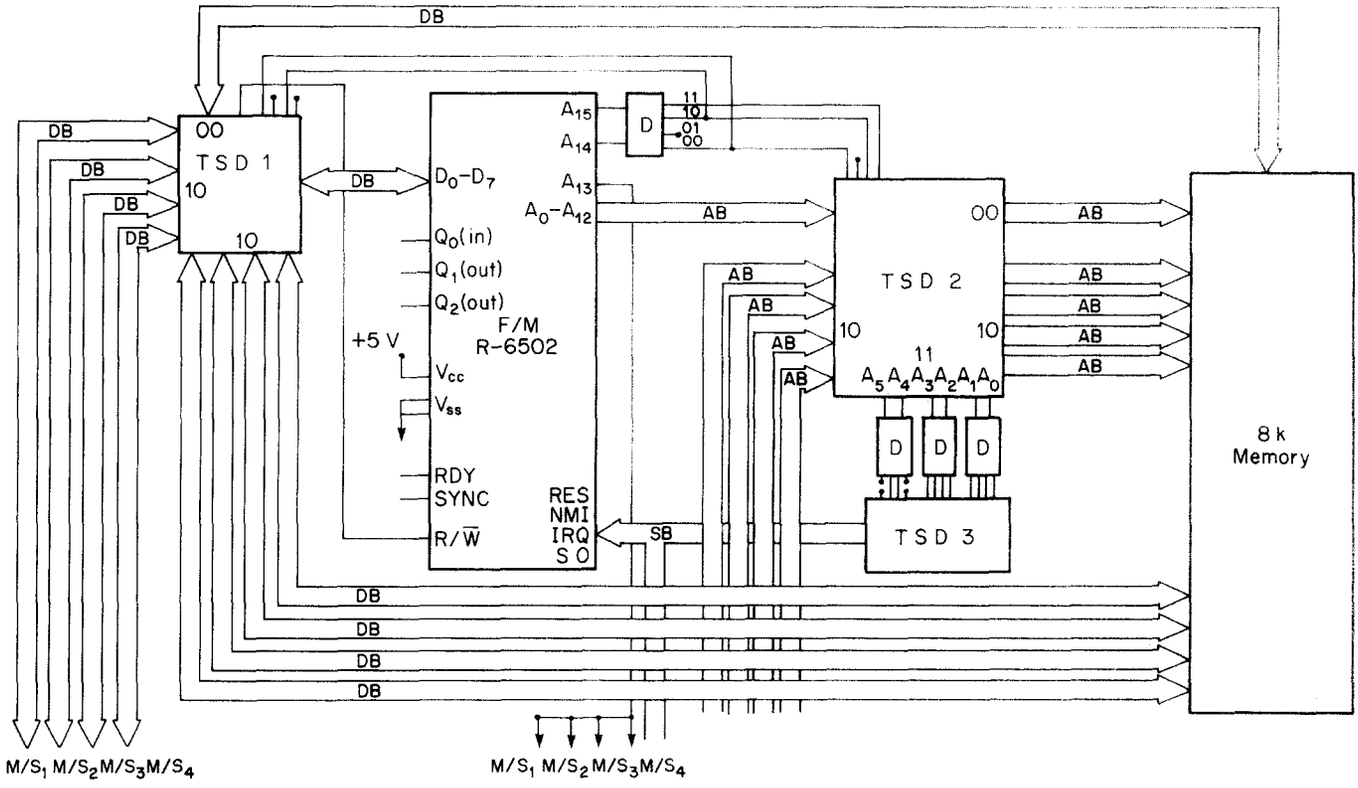


Figure 4. Layout of the internal structure of the full/master (FM) processor

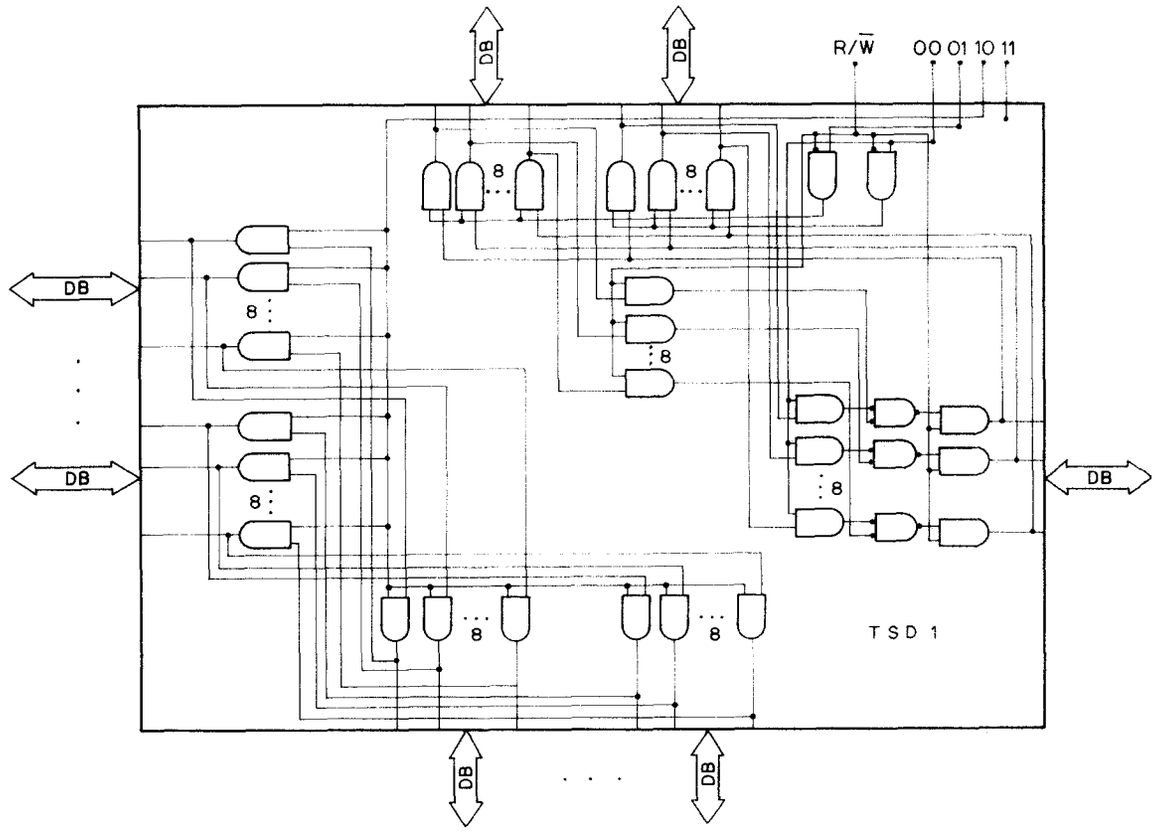


Figure 5. Internal structure of the target selecting device 1 (TSD1)

Figure 3). The internal structure of the MS node is illustrated in Figure 9. The internal organization of the MS node memory is the same as that of the FM node memory, the only difference being that common memory is shared between MS and S nodes instead of the FM and MS nodes, and the memory of its successor node is shared between a given MS and its successor S nodes (see Figure 3).

Slave leaf (S) processor node

The S node has the same hardware features as the previous two processor nodes (see Figure 10), the only

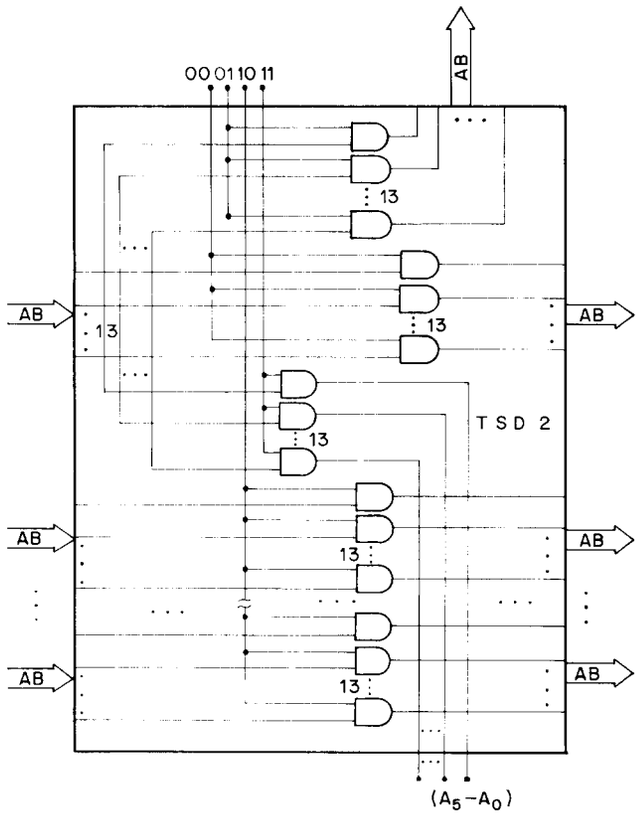


Figure 6. Internal structure of the target selecting device 2 (TSD2)

substantial differences lie in its memory organization. S node memory is shared in four segments. The first, common memory, with size 1k, is used for communication between the MS and S nodes (the MS node sends programs, commands and possibly data, while the S node sends data, in the form of abstracted picture information, to the MS node). The second segment, the PS memory, is 1k in size and is used by the S node for storing new program statements required for the current processing task. These statements do not exist in the slave processor's ROM and are consequently downloaded from its parent node. The third segment, the DS memory, of size 4k, is the storage area for the pixels received from the external photoarray environment. Finally, the OPS is the operating system memory space.

Interconnecting the processor nodes

The interconnection of the processor nodes in the MTN system presents some problems. The most serious problem is that the R-6502 CPU is not able to send selection signals and special commands, since the only available pins which can send signals of any kind are the address bus (AB) pins. This problem can be circumvented by the following method. The A15 and A14 pins should be freed from addressing operations and used for forwarding the selection signals to the TSD1 and TSD2 devices. Additionally, the A13 pin should be connected to the SO pin of the slave node whose function is explained below. The remaining address bus pins are directed to the TSD2 and consequently have a double task to perform, depending on the selection signals which the TSD2 device receives from the CPU's A15 and A14 pins (see Table 1). Accordingly, the appropriate number of control signals and special commands may be sent out from the AB channel. However, this technique reduces memory accessibility from 64k to 8k by removing the A15, A14 and A13 address lines.

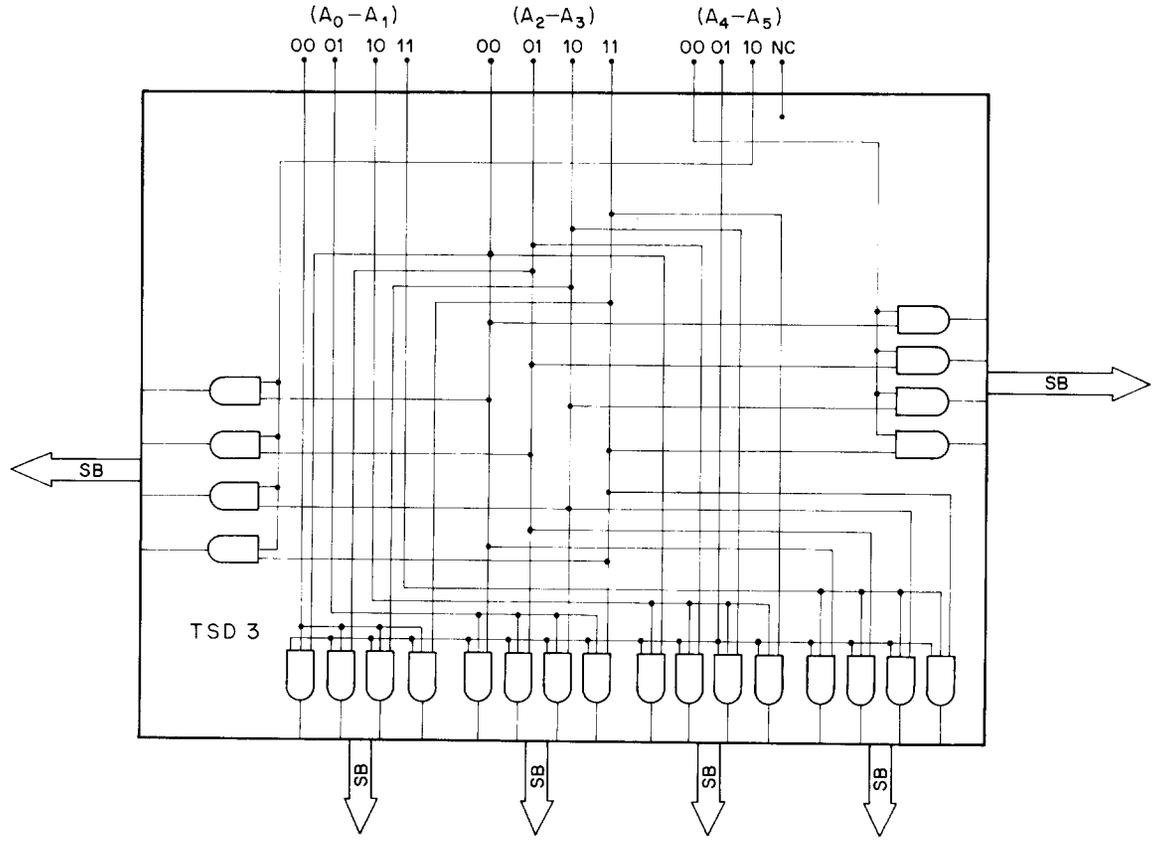


Figure 7. Internal structure of the target selecting device 3 (TSD3)

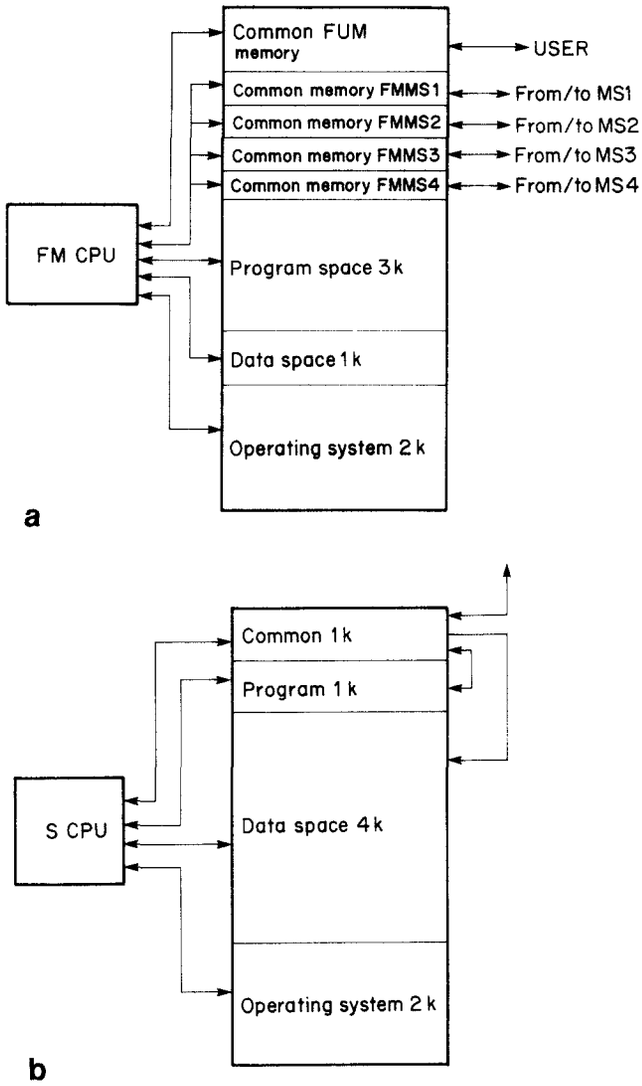


Figure 8. Segmentation of the microprocessor memory

Another significant problem, which impacts the distribution of memory space, concerns the communication between the master node and its slave nodes. This problem can be solved by means of the selection signals from the A15 and A14 pins of the RS-6502 CPU and the use of TSD1 and TSD2 devices. Specifically, the R-6502 CPU can access the entire 64k of available memory by means of the appropriate selection signals. Moreover, the four slave nodes can also access their corresponding portion of that memory simultaneously by utilizing the appropriate selection signals. In this way an appropriate communication path is established between the FM node and the MS nodes as well as between the MS nodes and the S nodes.

There remain some minor points to be discussed, primarily concerning the processor nodes. For example, the FM node has no upper level to communicate with, and therefore has no need of a TSD device to direct its address channel, data channel, or even control signals for communication to an upper level. This can easily be achieved by disconnecting the 01 input of the TSD1 and TSD2 devices, and the 00 input of the TSD3 device (see Table 2). On the other hand, the S nodes have no lower level to communicate with, except the photoarray, from which they receive the image pixels directly.

MTN OPERATION

As shown in Figures 11 and 12, MTN operation encompasses two modes of functionality:

- top-down mode, where commands flow down the MTN hierarchy;
- bottom-up mode, where abstracted picture information flows upwards along the nodes of the MTN levels.

Top-down operation

During top-down operation of the MTN hierarchy, the FM node receives commands from the user, decodes them and sends them to its four successor nodes for further processing and/or transmission. A similar task is executed in each node. The pattern of execution can be analysed as five functional steps:

- *Input.* The node inputs the commands transmitted from its parent-node.
- *Decoding.* The node decodes the received commands which specify the particular processing task executed by each node.
- *Execution.* The node executes the process tasks allocated to it, if any.
- *Encoding.* The node encodes the new possible tasks which its descendant node must execute.
- *Output.* Each non-slave node transmits the command packages to its descendant nodes.

Bottom-up operation

Bottom-up operation takes place at the slave node level, (I_0). Each slave node accepts in parallel the true picture information which corresponds to its respective picture region (squite) of $n \times n$ pixels, where $1 <= n <= \log_2 N$. Subsequently, the computer vision process in MTN follows a hierarchical pattern of execution. Each node, at level I_0 attempts to recognize the information existing in its own region, according to the commands transmitted from its parent node. When the individual perception of a node is completed, each node communicates with its parent node and transmits to it the picture information in an abstract-coded form. The parent node (MS) accumulates the information packages received from its

Table 2. Communication scheme of the current processor

A5	A4	A3	A2	A1	A0	Connections
				0	0	From current to slave processor1
				0	1	From current to slave processor2
				1	0	From current to slave processor3
				1	1	From current to slave processor4
		0	0			RES
		0	1			MNI } Special command signals
		1	0			IRQ } Special command signals
		1	1			SO
0	0					From current to master processor
0	1					From current to slave processors
1	0					Local communication
0	0					Not connected

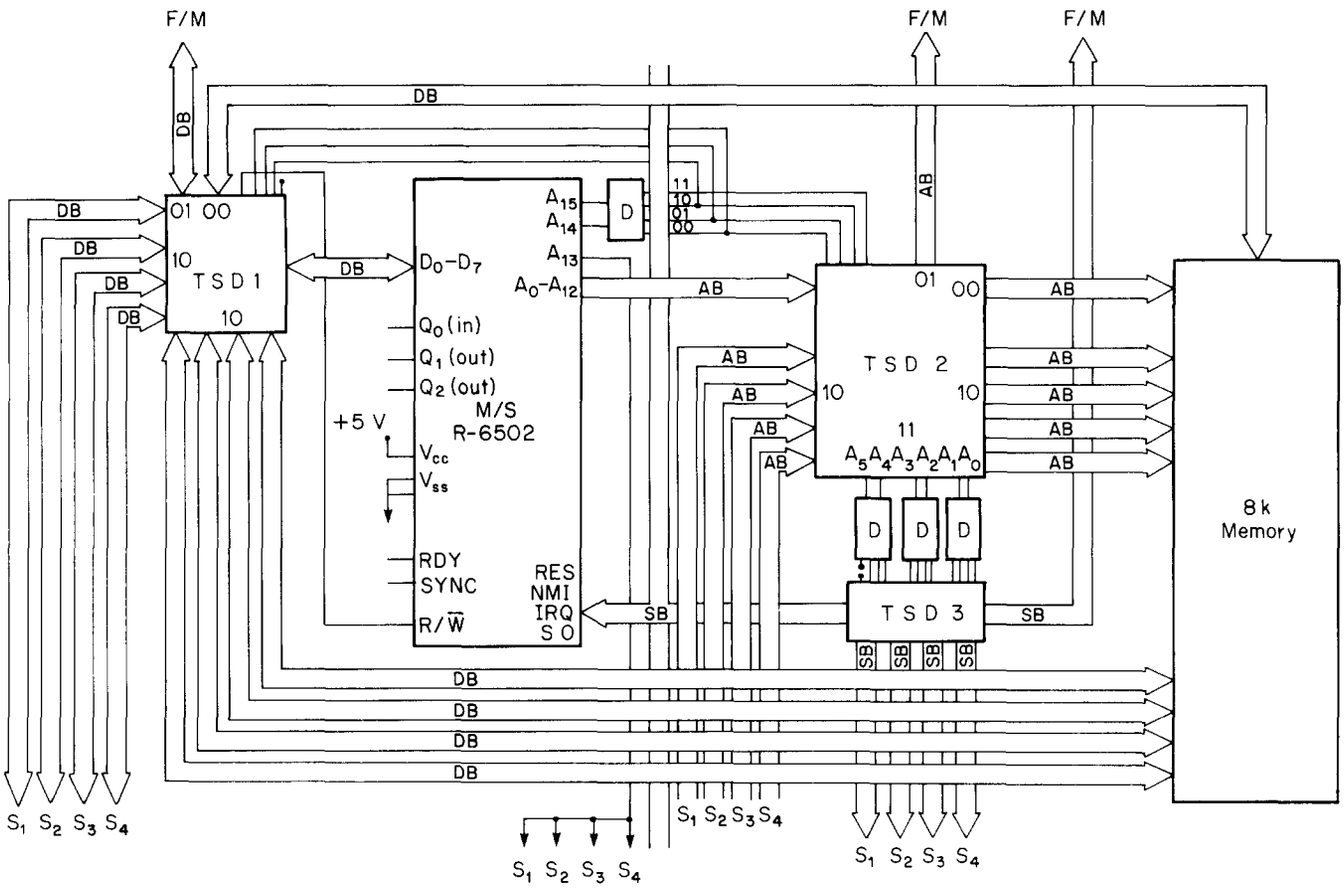


Figure 9. Internal structure of the master/slave (MS) processor

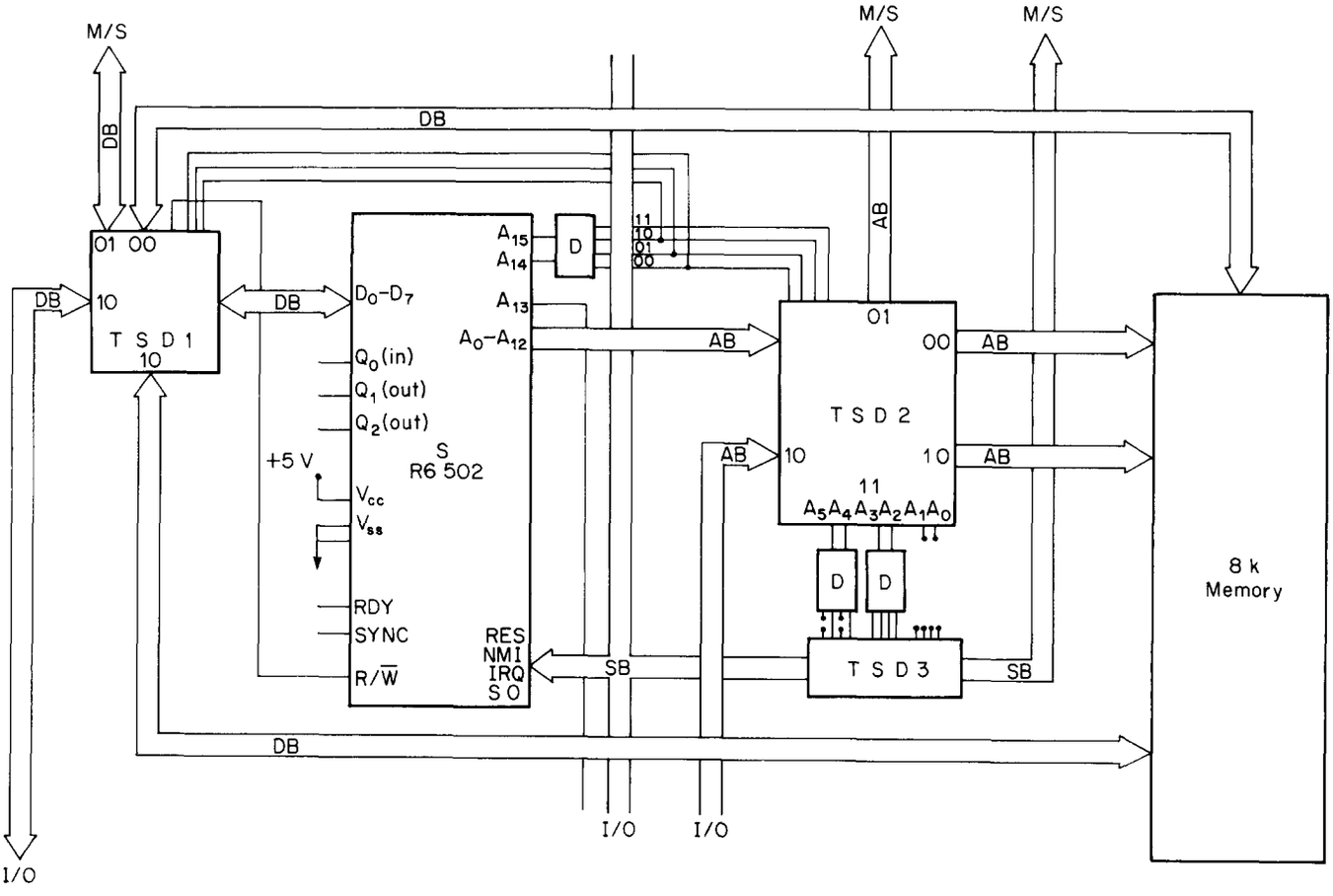


Figure 10. Internal structure of the slave (S) processor

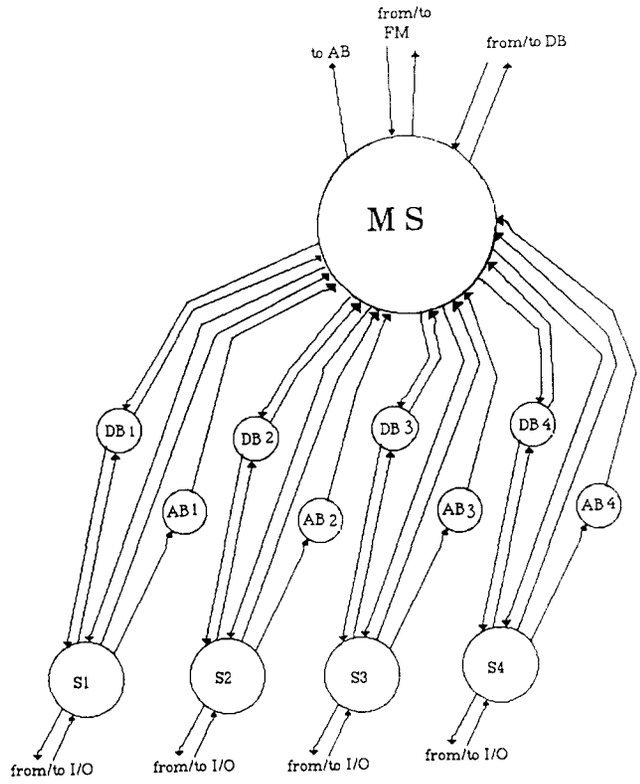


Figure 11. Directed graphical representation of MS and S processor operation

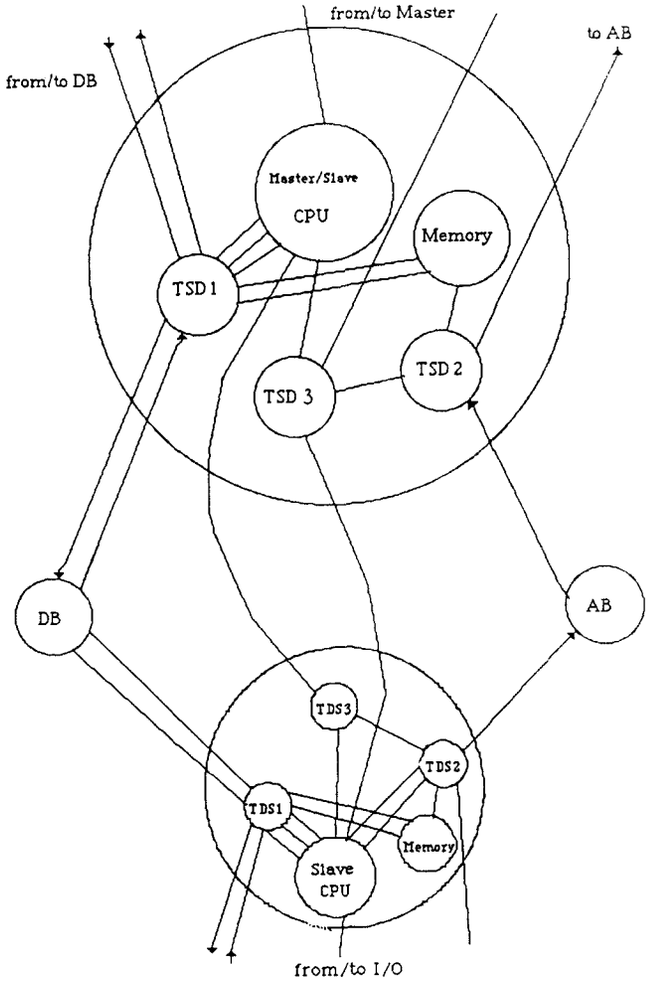


Figure 12. Directed graphical representation of the operation and communication of the MS and S nodes

descendant nodes, synthesizes and normalizes them and attempts to recognize their individual components. When the perception process is complete, the MS node transmits to the FM node the abstracted form of its own pictorial information with its perceived description. The FM node receives all the information packages, synthesizes and normalizes them and makes the final perception and decision on its constituents. Details about the perception and understanding procedures, using a knowledge-based vision system are discussed in Reference 8.

Processor node intercommunication

According to the architectural features discussed above, a descendant node communicates with its parent node by setting the required information into its parent memory, (see Figure 3). A parent node communicates with its descendant node by setting its orders into predefined locations of its memory and informing the descendant node to pick them up. These predefined memory locations are dedicated to specific descendant nodes. When a recognition task takes place at the slave node level, each slave processor is asked to communicate with its parent processor.

Initially, the S node sends a signal to the MS node through the TSD2 and TSD3 devices and sets its S0 pin (an interrupt request communication signal) to request communication. Subsequently, the MS node checks its S0 bit in microprogramming mode and notes when that bit was set. At this point the MS processor carries out its current processing task, disconnects its memory and connects the S node with the appropriate locations in its memory. This procedure is implemented by the proper signals through the TSD2 device. It is important to note that the S node does not need to send its code number to its corresponding MS node. This is because there always exists the possibility of four S-nodes asking simultaneously to communicate with the same MS node S0 bit through an OR gate. Accordingly, the MS node releases its memory to S nodes whether or not all four nodes require communication. This feature provides the MTN system with a parallel communication ability along its hierarchy. After the storage process has taken place each S node resets the S0 bit of the MS node. If the MS node wishes to send orders to its S nodes, it sends the appropriate control signal and allows the S nodes to access the dedicated memory locations of the MS node. Communication between FM node and MS nodes follows the same procedure.

Discussion and an illustrative example

The multiprocessor system proposed here is to be used for computer vision purposes although its structure can support a large variety of different parallel calculations, such as matrices multiplication and orthogonal transformations (FFT, WHT, etc.). A number of low-level vision processes have been implemented on the MTN structure; examples are global convolution, segmentation, thinning, normalization and chain coding[9-11]. In this paper, however, emphasis is placed on the hardware organization and design of the MTN system using R-6502 processors.

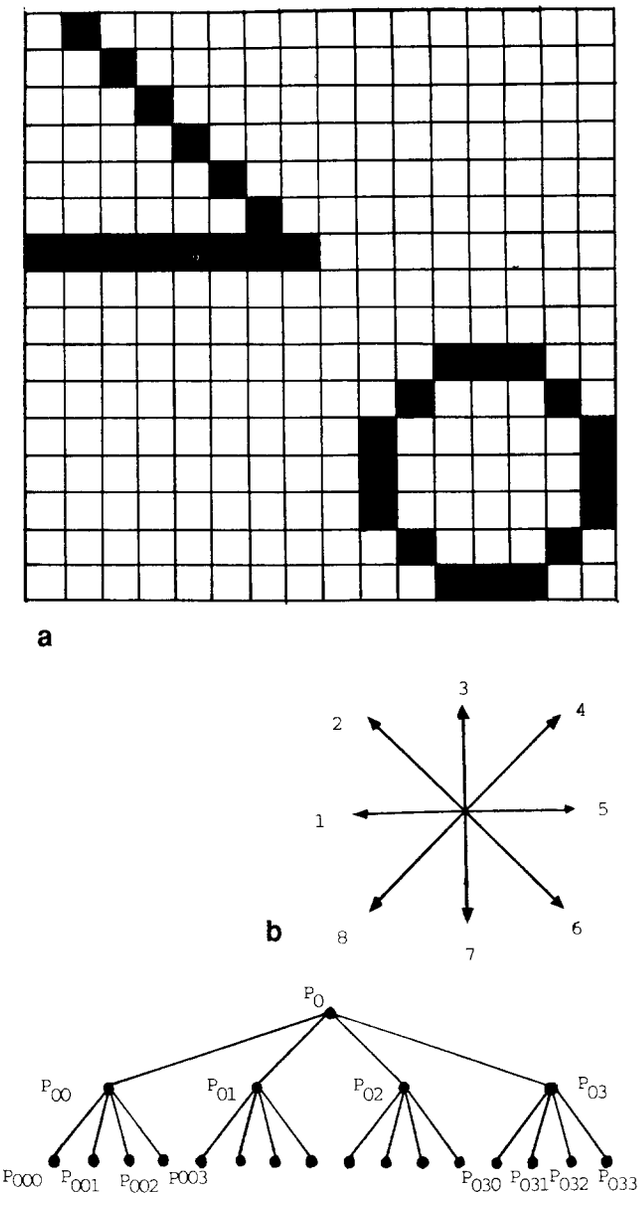


Figure 13. ***a****, digitized image of 16 × 16 pixels;* ***b****, Freeman code;* ***c****, 16 node MTN*

The somewhat outdated R-6502 processors were used because, at the time, they were the only chips available to the authors. However, the quadtree structure of the MTN system could be implemented using any type of processor, transputers being particularly suitable.

The entire MTN structure consists of $((4N^2)/4^i) - 1/3$ processor nodes, where $N \times N$ represents the size of the image received by the MTN system, and i is a resolution parameter dependent on the number of processor nodes used at the lowest level of the quadtree structure. Specifically, for an image of 256 × 256 pixels, if the number of processor nodes used, at the lowest level, is 128 × 128 then $i = 1$. For the same image size, however, if the number of nodes used is 16 × 16 then $i = 4$ and each processor leaf node receives 256 pixels. The 21 processor nodes MTN-system have 16 processor leaf nodes, which means that for real pictures of 256 × 256 pixels, each leaf node must receive 4096 pixels. Processing of a small picture of 16 × 16 pixels containing simple objects has already been carried out. Figure 13a shows an experimental image of size 16 × 16, which contains an angle and an octagon. The image in Figure 13 has been processed and the results, illustrating the hierarchical operation of the MTN system, are provided in Appendix 1.

CONCLUSIONS

The architectural design and the global operation of a multiprocessor-multibit tree network (MTN) architecture are presented. The MTN system is a vision system which presents a number of significant features: parallelism, hierarchical (i.e. bottom-up) processing, and pipelining (processing more than one picture at a time). The MTN functional structure includes two modes:

- bottom-up, where abstracted picture information travels up the hierarchy;
- top-down where orders go down along the MTN hierarchy.

The MTN system described is based on the R-6502 microprocessor, although almost any type of microprocessor could be used.

ACKNOWLEDGEMENT

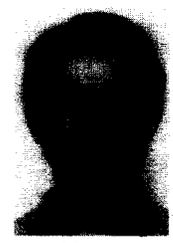
The authors wish to express their thanks to the reviewers for their fruitful suggestions.

REFERENCES

1 **Duff, M** *Computing Structures for Image Processing* Academic Press, New York, NY, USA (1983)
2 **Uhr, L** *Parallel Computer Vision* Academic Press, New York, NY, USA (1987)
3 **Tanimoto, S and Klinger, A** *Structured Computer Vision* Academic Press, New York, NY, USA (1980)
4 **Danielson, P E and Levialdi, S** 'Computer architectures for pictorial information systems' *IEEE Comput.* No 11 (1981) pp 53–67
5 **Bourbakis, N** 'Design of real-time supercomputing vision system architectures' *Proc. IEEE Conf. on Supercomputing* Santa Clara, CA, USA Vol 3 (May 1987) pp 392–398
6 **Bourbakis, N and Vaitsos, C** 'A multi-microprocessor tree network configuration used on robot vision systems' In **Tzafestas, S (Ed.)** *Digital Techniques* Elsevier Science (1985) pp 483–490
7 *AIM 65 Hardware Manual* Rev 3 Rockwell International, CA, USA (1979)
8 **Gowrinshakar, T R and Bourbakis, N** 'Specifications for the development of a knowledge-based image interpretation system' *Int. J. Eng. Appl. of AI* (1990)
9 **Bourbakis, N** 'A parallel-symmetric thinning algorithm' *Int. J. Pattern Recognition* Vol 22 No 4 (1989) pp 387–396
10 **Bourbakis, N and Fotakis, D** 'A heuristic scheme for recognition of progressive digital straight lines with unevenness' *Proc. IEEE Workshop on LFA* (August 1988) pp 176–183
11 **Bourbakis, N and Tabak, D** 'Working mechanisms and their Petri-net modeling for the HERMES multiprocessor vision system' *Int. J. Eng. Appl. of AI* Vol 1 No 2 (1988) pp 102–110

APPENDIX 1: IMAGE PROCESSING EXAMPLE

```
                LEVEL   0: Features Extraction & Coding
=============================================================================

      Extracted & Coded Information          Message to the Upper Level
      -----------------------------
P033: (4,1)7881                        ==>   [033]<417881>
P032: (1,2)766                         ==>   [032]<12766>
P031: (2,1)566                         ==>   [031]<21566>
P030: (2,4)88                          ==>   [030]<2488>

P023: 0                                ==>   [023]0
P022: 0                                ==>   [022]0
P021: 0                                ==>   [021]0
P020: 0                                ==>   [020]0

P013: 0                                ==>   [013]0
P012: 0                                ==>   [012]0
P011: 0                                ==>   [011]0
P010: 0                                ==>   [010]0

P003: (1,2)669(3,3)11                  ==>   [003]<126693311>
P002: (3,1)555                         ==>   [002]<31555>
P001: (4,1)                            ==>   [001]<41>
P000: (1,2)66                          ==>   [000]<1266>

                LEVEL - 1: Synthesis, Encoding & Matching
=============================================================================

           Synthesis & Encoding
           --------------------
P03: /[033]<417881>/[032]<12766>/[031]<21566>/[030]<2488>/    ==>

P03: /(5,8)(6,8)(7,7)(8,6)(8,5)/(5,2)(6,2)(7,3)(8,5)/(2,5)(2,6)(3,7)(4,8)
     /(2,4)(3,3)(4,2)/    ==>

P03: (2,4)556677881122334

           Matching
           --------
P03: P03 HOLDS AN OCTAGON WITH SIZE  8x8.

     Message to the Upper Level
     --------------------------
     [03]<A/OCTAGON 8x8/>


           Synthesis & Encoding
           --------------------
P00: /[003]<126693311>/[002]<31555>/[001]<41>/[000]<1266>/    ==>

P00: /(5,6)(6,7)(7,8)(7,7)(7,6)(7,5)/(7,1)(7,2)(7,3)(7,4)/(4,5)
     /(1,2)(2,3)(3,4)/    ==>

P00: (1,2)6666669(7,7)111111


           Matching
           --------
P00: P00 HOLDS TWO STRAIGHT LINE SEGMENTS CONNECTED AT (7,8)

     Message to the Upper Level
     --------------------------
     [00]<12666666977111111>

                LEVEL - 2: Synthesis & Conclusions
=============================================================================

     Synthesis
     ---------
P0:  THE OCTAGON AND THE STRAIGHT LINE SEGMENTS CANNOT BE COMBINED
     END;
```



Nikolaos Bourbakis received a BS in mathematics from the National University of Athens, Greece in 1974 and a PhD in computer science and engineering from the University of Patras, Greece in 1982. He is currently working with IBM, San Jose, CA, USA. His research interests are in multiprocessor architectures, languages and algorithms for image, vision and text; AI tools for VLSI design; and robotics (navigation symbiosis). He has published more than 80 papers and is the author or coauthor of six books related to computers, image and AI. He is editor-in-chief of the International Journal on Tools for AI, *the editor of a series of AI research books and the founder and general chairman of the IEEE Conference on Tools for AI.*

Mike Papazoglou received a BS in electronics from the University of Dundee, UK in 1978, an MSc in computer systems from the University of Edinburgh, UK in 1979, and a PhD in microcomputer systems from the University of Dundee in 1982. He is a senior lecturer at the National Australian University, Department of Computer Science, Canberra, Australia. His research interests are in distributed information systems, object-oriented database systems, semantic data models, object management, and intelligent information systems. He has published many papers and is the author of a book on relational database management. He was co-programme chairman of the IEEE Conference on Tools for AI-89 and is a member of various scientific committees.

George Alexiou received a BS in physics in 1977 and a PhD in Electronics in 1980, both from the University of Patras, Greece, where he is a senior assistant professor in the Department of Computer Engineering. His research interests are in VLSI design; analogue electronics, digital filters, microcomputer system architectures and digital systems. He has published papers in a number of international journals and conference proceedings and is the coauthor of a book on digital systems design.