RSD-TR-9-82

A Survey of Computer Architectures Used in Image Processing

James F. Poje

Edward J. Delp

April 1982

Center for Robotics and Integrated Manufacturing

Robot System Division

COLLEGE OF ENGINEERING
THE UNIVERSITY OF MICHIGAN
ANN ARBOR, MICHIGAN 48109

TABLE OF CONTENTS

1. Introduction

Digital Image Processing requires a tremendous amount of computational power. Typical techniques can require convolving an entire picture with a small mask or estimating the 2-dimensional Fourier Transform of an image. These images can range in size from a 64 by 64 bit plane to a 1024 by 1024 8-bit picture. Originally, general purpose Von Neuman style computers were used for this processing. These proved inadequate in both their speed and cost. Parallel and pipelined machines are an alternative to bring down cost while improving performance. This report presents an introduction to a few of the current computers proposed and being built especially for this application. It is written with the intent to give an idea of current research topics involved in computer architectures for image processing.

2. General Computer Structure

This is a brief overview of the different classes of computers. Almost any type of computer will fit into one of these categories or some combination of them. A description of image processing algorithms, their complexity, and what class of machine best suits them can be found in [1] as well as any current book in parallel processing.

2.1. Sequential Computers

This is by far the largest in number of machines actually built. Every "home" computer built operates in a sequential fashion. It consists of a controlling unit, an arithmetic unit, and a memory. The standard form of execution is to fetch a single instruction from memory, decode and execute it, and then fetch the next one. Large repetitive operations, such as those found in image processing, are computed by having a program loop through a sequence of operations, performing a function on a single piece of data each time through the loop. Figure 1(a) is an example of a typical sequential machine.

2.2. SIMD

SIMD stands for Single Instruction stream - Multiple Data stream. This type of computer consists of a control unit and a number of processors, each one having some type of local memory or sharing a large global one (see Figure 1(b)). Each processor executes the same instruction in parallel with rest, so that each one produces a result at the same time (see Figure 1(c)). If there are N processors, a SIMD can have at best a speedup of N over a sequential machines, assuming that every processor is being utilized. The problem lies in finding algorithms which will use all N processors continuously.

2.3. MIMD

MIMD stands for Multiple Instruction stream - Multiple Data stream. As in SIMD, there is some type of control unit associated with a number of processors. The difference is that each processor acts independently from the rest.

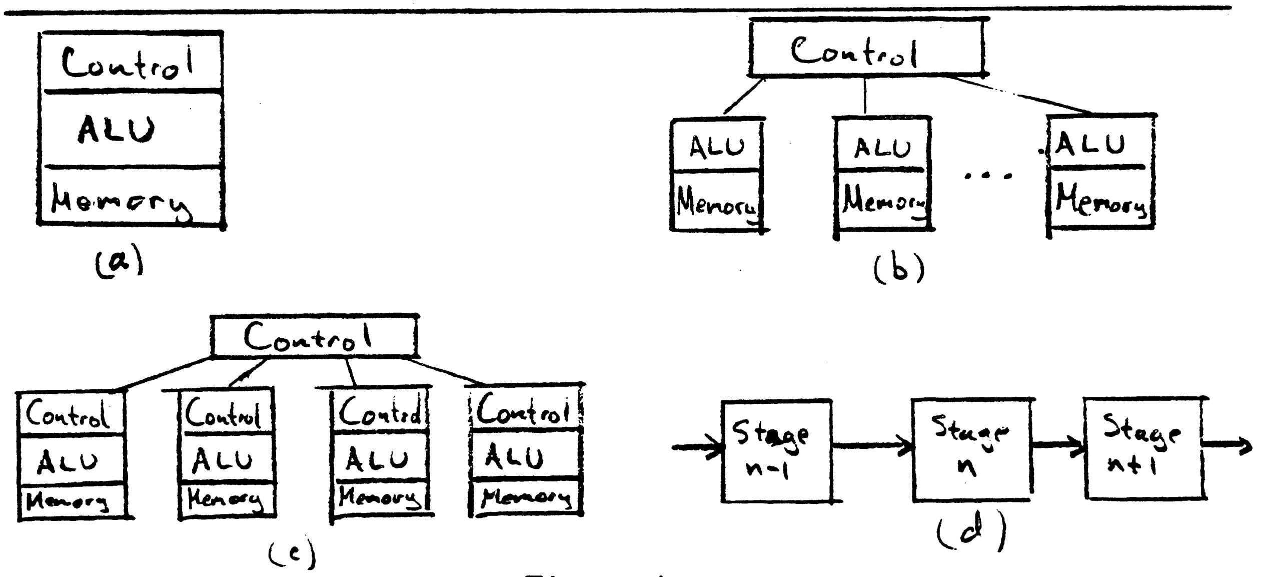

Figure 1
Typical Computer Classes

The use of an array processor results in a significant savings in execution time. Table 1 shows some sample processing times for comparison.

3.2. A Fast Picture Processor with a High Level Language

This is a processor which has been built at the Philips Research Laboratory in Redhill, Surrey, England [3]. It is used for processing binary images and achieves a speedup of about 100 over a typical minicomputer. It is also programmed in a high level language which translates programs directly into microcode.

A block diagram of the processor architecture is shown in Figure 3. The machine is centered around the RAM, which is organized as a two-dimensional array of single bits. Up to 16 picture elements may be accessed at once for reading or writing. All operations the machine performs are executed by means of a lookup table. These are loaded by a program before processing begins, and provide up to a sixteen bit result and sixteen bit next state. The results the tables produce are a function of the state register and picture data. The results are shifted into a FIFO queue, where they are held until deposited back into RAM. The entire system is supervised by the Picture

System	Algorithms	
	Contrast Stretch 512 x 512 2 x 2 Window	Fast Fourier Transform 512 x 512
PDP-11/70	3 Min.	122 Sec.
PDP-11/70 With Array Processor	9.3 Sec.	18 Sec.
CDC-7600	5.1 Sec.	3.3 Sec

Table 1
Sample image processing times

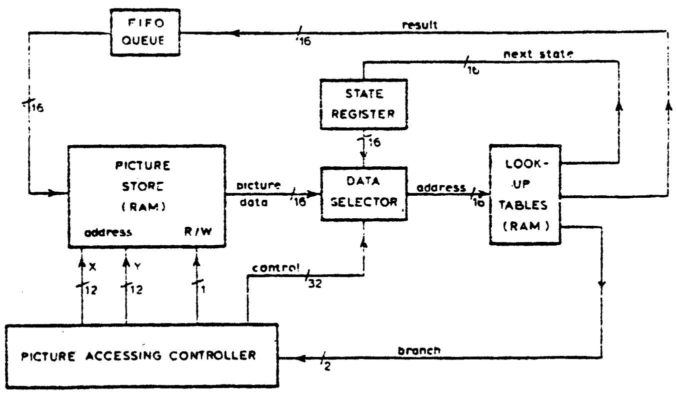

Figure 3
Philips Research Lab Processor

Accessing Controller, which determines the functions to perform.

As an example, suppose that a local 3 by 3 operation were to be performed over the entire picture. During the first cycle, three bits of data would be stored in the state register by proper programming of the lookup table. The second cycle would put three more in there. On the third cycle a result is formed. Since this is really a finite state machine representation of a pipeline, results are produced for each of the next clock cycles until the next row of the picture needs to be scanned.

The programming language uses a hierarchical structure. At the lowest level, individual operators are specified. These can specify logical or arithmetic functions, where to read the data from and where to write the results to. The next level defines what part of the picture to apply that function to. This allows for such things as scanning every other row or working on a small portion of the entire image. The highest level allows organization of different operators, such as first applying an edge detector followed by a line thinner.

The main constraint of the system is that it can only use binary images. This allows the system to be very fast (a 2 by 2 operator over a 100 x 100 picture can execute at 5 ms), but also limits the applications it would be useful for. For binary images it provides a fast general purpose machine, though the main emphasis of present day image processing is with gray scale data.

3.3. A Low Level Architecture for Real Time Computer Vision

This is a processor described by D. Sherdell in [4]. The idea is to divide computations between a number of *different* processors which are able to function independently. This allows each part to be specialized, making its design simpler, faster, and cheaper, while still allowing the entire system to be flexible enough for general applications. A block diagram is shown in Figure 4.

The TV camera and digitizer can provide a 256x256 or 512x512 frame at a rate of up to a frame every 10 milliseconds. Each pixel is an 8-bit gray scale

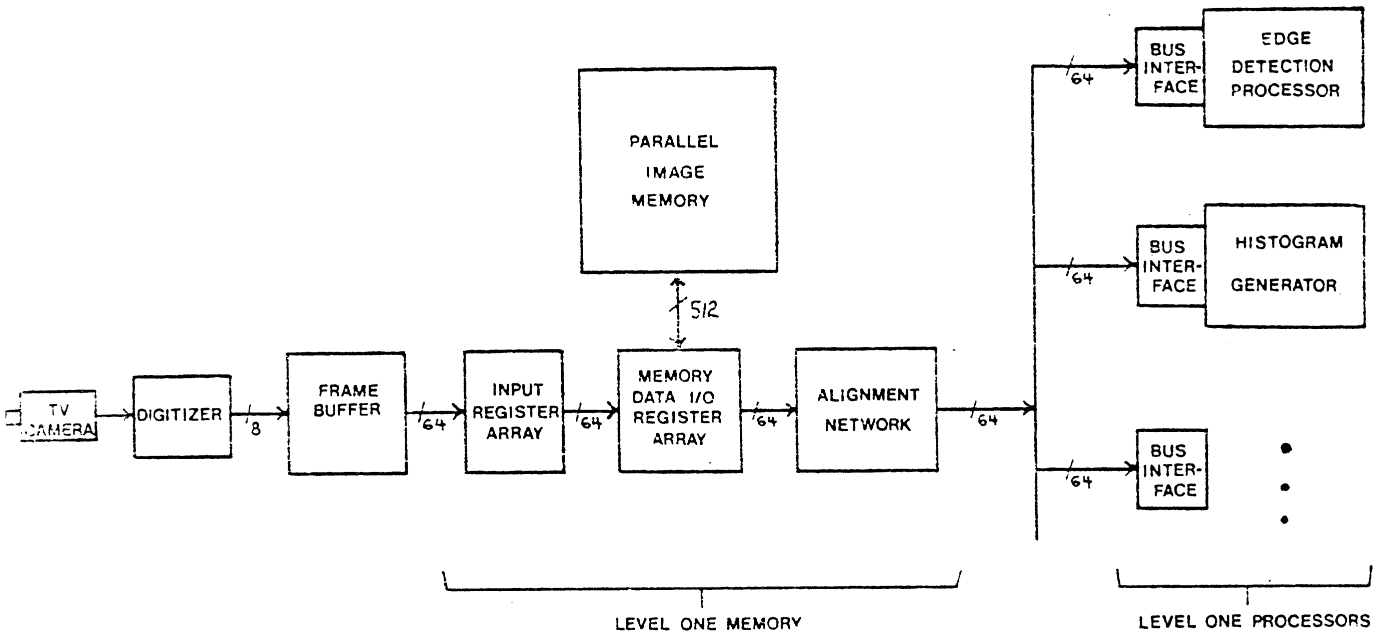

Figure 4
Low Level Architecture for Real Time Vision

value. The parallel image memory is the main storage for a picture. It is divided into 64 independently addressable memory modules, each one organized into 8x8 module arrays. A description of the addressing scheme for the memory can be found in [5]. Each of the level one processors obtains blocks of 64 pixels. These are transferred through the memory data I/O register array and through an alignment network. The blocks are moved in parallel from the image memory to the register array, and then transferred serially as eight 8x8x1 bit planes to the level one processors. This is done to reduce the size of the bus from 512 down to 64. The alignment network is used to insure proper addresssing.

At present, only two of the level one processors have been built. One implements the Sobel edge detector, and the other generates a histogram of the gray scale values. For the Sobel operator, the image is divided into 6x6 blocks of 36 pixels. Each of these blocks is put into a small separate functional unit which performs the Sobel operator on those 36 pixels. The total time to process an entire 256x256 picture is given at 33 milliseconds. The histogram processor is much simpler, consisting mainly of a series of incrementors.

The main purpose of the entire system is very fast low level processing of images. The idea is that the output from this system may then eventually be used as the input to a higher level machine which would do some of the initial analysis of the pictures.

3.4. PUMPS

PUMPS is a general purpose image system developed at Purdue University [6]. As in the previous system, it allows for attached processors to be added to perform image processing routines. Each of these may then be built as special purpose machines to increase their speed performance. A diagram of the entire system is shown in Figure 5.

The TPU's are Task Processing Units. They are connected to the attached processors (PPVU's) through the Special Resource Arbitration Network (SPAN), which performs like a crossbar switch. Each TPU has its own memory and cache for execution of its local kernel program. Image and other data

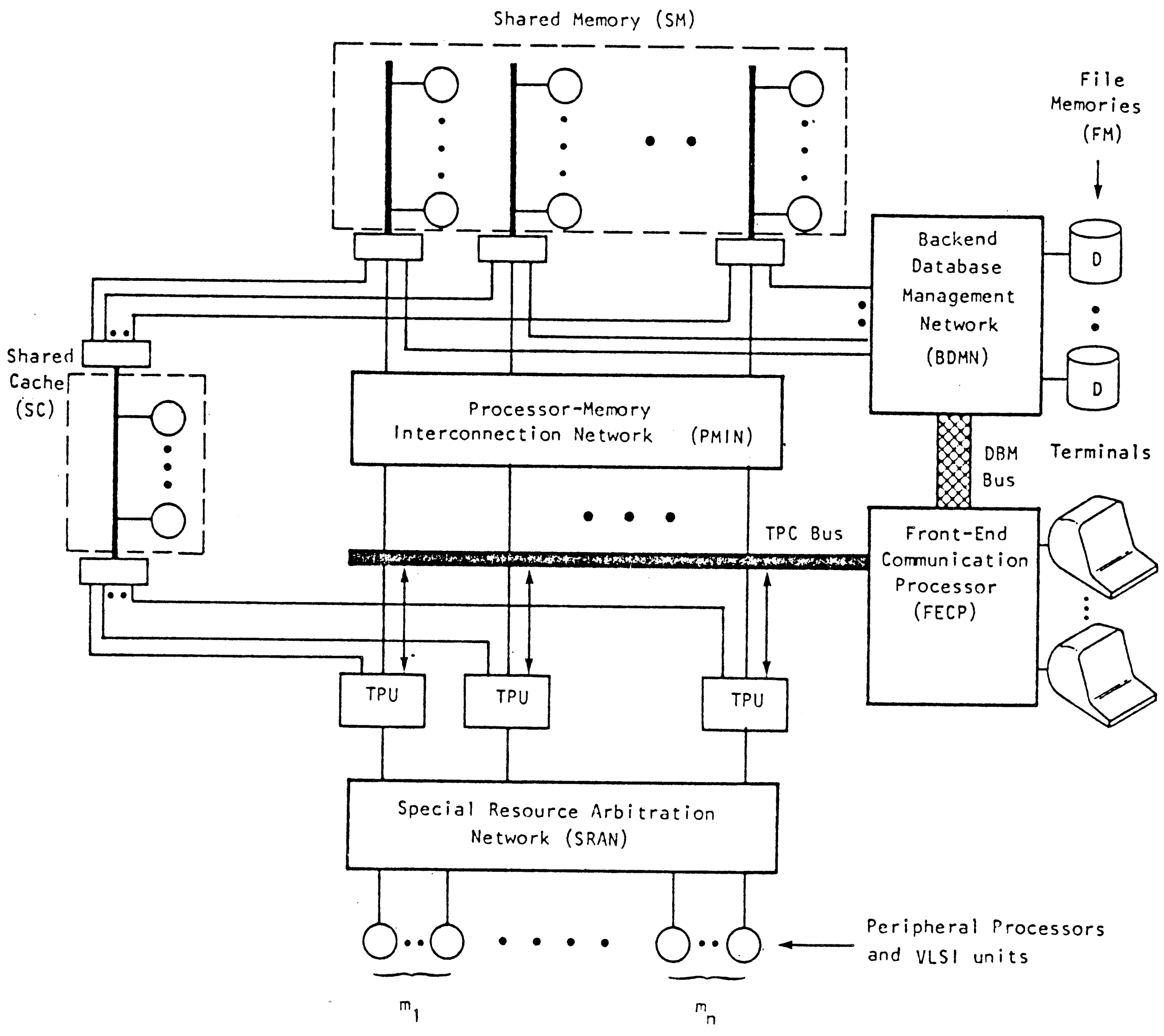
Shared Memory (SM)
File Memories (FM)
Backend Database Management Network (BDMN)
D
D
Shared Cache (SC)
Processor-Memory Interconnection Network (PMIN)
DBM Bus
Terminals
TPC Bus
Front-End Communication Processor (FECP)
TPU
TPU
TPU
Special Resource Arbitration Network (SRAN)
Peripheral Processors and VLSI units
m_1
m_n

Figure 5
PUMPS

used by it is in the shared cache and shared memory. The user interface is through the Front-End Communication Processor, which enables users to access pictures with the help of a database management network. It is also interfaced to processor-memory interconnection network which provides access to the whole multiprocessor system.

The backend database management network is operated on a query scheme, where the user can specify a set of attributes. Both images and image programs are stored in the database, allowing one to call up a number of processing functions. The classification of these functions may be found in [6].

For real time processing and analysis of pictures, PUMPS may be organized into a macro-pipeline structure, as in Figure 6. In this case, each of the S_i is either a TPU or an attached processor. Each stage consists of some global operation, such as edge detection of FFT. The B_{ij} are buffers which are necessary to hold part of the images until the next stage is ready to perform on it. These buffers may range from a few words to entire pictures, depending on the dissimilarity of execution times between stages.

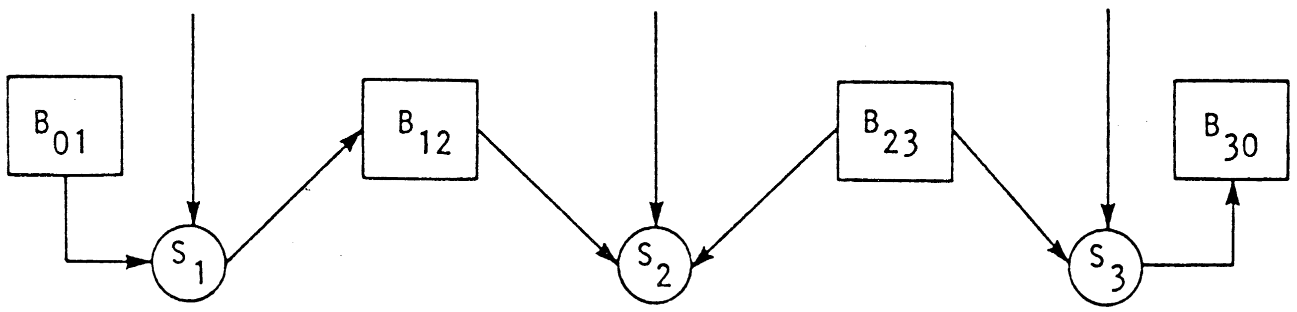

Figure 6
Marco-pipelining

3.5. PASM

PASM is another computer being developed at Purdue University [7]. It stands for PArtitionable SIMD/MIMD system. It uses a set of microprocessors which may be configured in either SIMD or MIMD mode. It will also contain a fairly sophisticated operating system and compiler which are planned to help unload the burden of specifying configurations from the user. A simple diagram of PASM is given in Figure 7.

The heart of the system is the Parallel Computation Unit (PCU). It consists of N ($N = 2^n$) microprocessors which perform the actual computations. Each one contains two banks of memory, so one bank can be loaded while the other one is operated on. The control storage and microcontrollers are microprocessors used as control units for the PCU when it is in SIMD mode. They are also used to orchestrate the activities of the PCU during MIMD mode. The memory storage and memory management provide a means of loading and unloading each of the modules inside the PCU. An interconnection net-

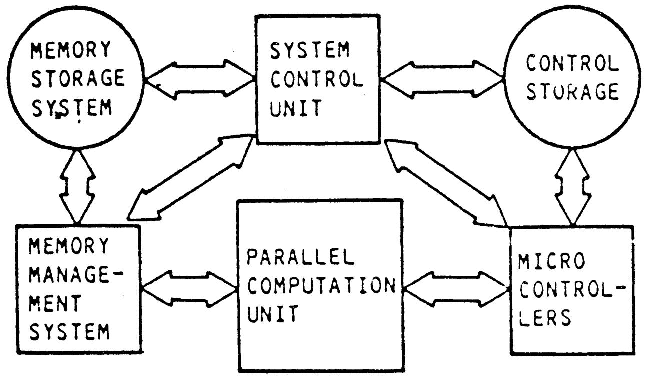

Figure 7
PASM Block Diagram

work resides inside the PCU which allows any of the processors inside to be connected to any other one through the use of an augmented data manipulator. A PDP-11 is to be used for a system control unit to oversee the entire operation.

An operating system called PASMOS is being designed. Its purpose is to provide some of the following features:

- file management to and from the PCU
- editing, compiling, and assembling facilities
- transfer instructions to the microcontrollers
- initiate PCU execution
- data formatting
- error handling

One of the more difficult aspects is the task scheduling and job control. This involves servicing all parts of the PCU to keep them busy, and to keep an up to date idea of the configuration.

The language planned is a block structured high level language. It is functionally oriented, which means that the programs written usually consist of simple functions which need to be applied over the entire image. The compiler is responsible for breaking the code down so it can be optimized over the processors, freeing the user from what would generally be an extremely difficult task. An example of the type of code to generate a histogram of the image is as follows:

```
for i=1 to row_size
        for j=1 to col_size
                hist[pixel[i][j]] = hist[pixel[i][j]] + 1;
```

One way to have PASM execute this would be to partition the image so that each processor is given a small portion to compute the histogram of. Then, using a binary merge it would combine all of the histograms together.

3.6. FLIP

FLIP stands for a FLexible multiprocessor system for Image Processing which has been developed at the Research Institute for Information Processing and Pattern Recognition in West Germany [8]. Its main application is to perform homogeneous operations which are mainly found in the field of picture preprocessing and feature extraction. A system view is given in Figure 8.

FLIP operates as an attached processor. Its central processing unit is called FIP (Flexible Individual Processor), which contains 16 IP (Individual Processors) . To handle data manipulation there is the PEP (Peripheral data Exchange Processor) which acts as a fast buffer. It contains three internal processors. In normal operation, the data flows through PEP to FIP, where it is operated on, and then directly back to PEP for storage. All data transfer is accomplished through DMA.

Each IP has two input buses as well as its own output bus. This allows the system to be configured in a number of different structure. The speed of the

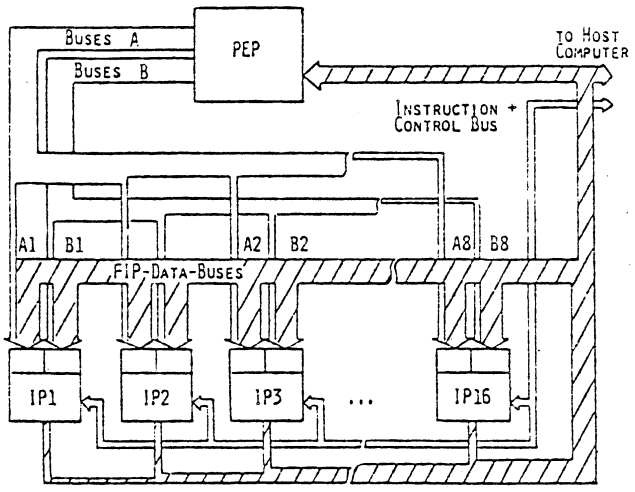

Figure 8
System view of FLIP

system is given at 64 million instructions per second

4. Cellular Architectures

This class of machine attempts to operate on images using only very simple logical functions. This results in extremely high speed, low cost per functional unit, and very simple circuitry. It allows for a tremendous number of these units to be included within a small area. The main disadvantage is that, while complex functions such as Fourier Transforms could be done, it would require an enormous amount of time both in clock cycles and programming because of the simple nature of the processors.

4.1. CLIP4

The CLIP4 architecture [9] puts a processor at each pixel location. It consists of an array of 96x96 of these processors, each one capable of performing simple Boolean operations. Figure 9 shows a diagram of one of these processors. In the actual implementation, images are digitized to 64 gray levels, and only the innermost 96 row and column pixels are used. This is done due to economical constraints.

As can be seen in the figure, the Boolean operation is a function of the **A** and **P** lines. The **A** line is, in normal operation, used to input the current data state. The **P** line may be set to either the **D** register, which is used to define the state of the processor, the **C** register, which is used to hold the carry out on arithmetic operations, or it may be one of the eight neighbors of that pixel. The outputs may be fed back into the **D** register, the **C** register, or out to the external connections which feed results to the surrounding pixels.

Two types of arithmetic operations are possible. The first type is so-called bit plane arithmetic where sums or differences between bit planes are formed. This means that the output of an arbitrary pixel O(i,j) is a function of

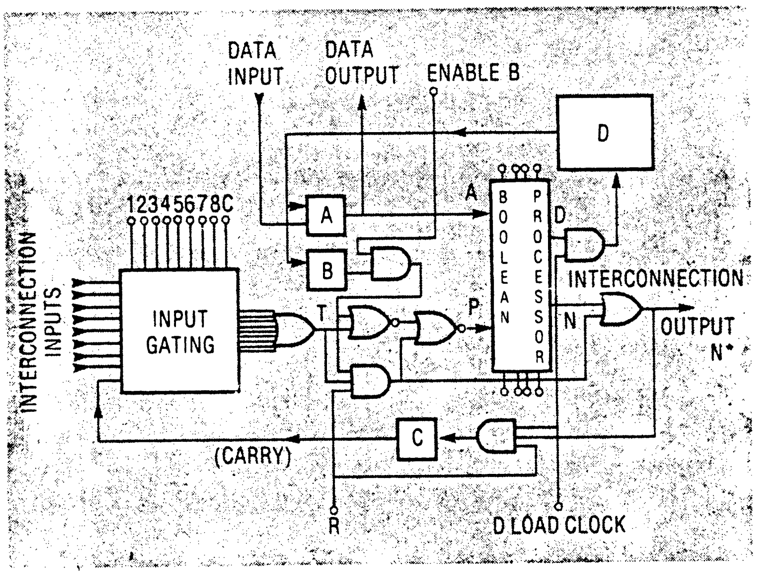

Figure 9
CLIP4 Processor

pixels at an identical location, say A(i,j) and B(i,j). The other mode is to perform operations between adjoining cells. This features allows the propagation of sums and carrys along rows, column, or diagonals.

The operating speed of CLIP4 is application dependent. Its precursor, the CLIP3, had been shown to operate as much as 10^6 times faster than an IBM 360/65 for some operation, though a lot of this time is due strictly to I/O. Typical speedups would normally be in the range from $10-10^6$.

4.2. Systolic Cellular Logic

This approach [10] is to combine the fast speeds obtained through parallelism with the inexpensive cost of pipelining. Bit plane cellular logic is used in order to reduce the complexity of the circuitry.

The system is to be built out of VLSI chips (see Chapter 8 of [11]). Each chip, called a tile, is used to process an 8x8 array of pixels. Each of these tiles is connected to their four neighbors. Likewise, inside the tile each pixel processor is connected to the nearest four. This reduces much of the circuitry normally used to connect a pixel with eight other ones. To access one on the diagonal, though, requires more processing time since the data must be shifted on the horizontal and vertical first.

Each pixel on chip contains three bits which are used to perform the logic operations. The registers have the following significance:

P : "Propagation" register - this value is accessible to neighbors
T : " Target" register - results of the logical operations
C : "Conditional" register - used for choice of operation

4.3. BASE 8

The BASE 8 binary array processor designed at Purdue University [12] has the appearance of being a fully parallel binary array system. It also operates on bit planes of 8x8 blocks.

Each bit in the 8x8 array has a Processing Element (PE) which is connected to all 8 adjacent PE's. The organization of a single PE is shown in Figure 10. The NNF is used to select one of the 256 functions of the 8 nearest neighbors. This is fed in to the Process Function (PF), along with the current state of the PE which is held in the Memory (MR). The output from this may be put back into the MR, or it may be sent as a result back to main memory.

The organization of the BASE 8 system is shown in Figure 11. There are three identical Data Acquisition Units (DAU), which are responsible for acquiring data for one operand. The APU generates a result which is fed to the Data Storage Unit (DSU) which, in most cases, sends this result back to main memory. All data units share a common 16 bit data bus, which is connected to the PDP 11's UNIBUS. The edge storage memory is used to hold data which resides on the boundaries of the DAU's. This information is used when a PE

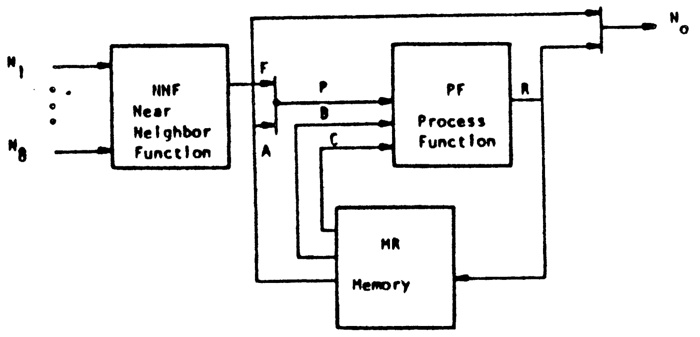

Figure 10
BASE 8 Processing Element

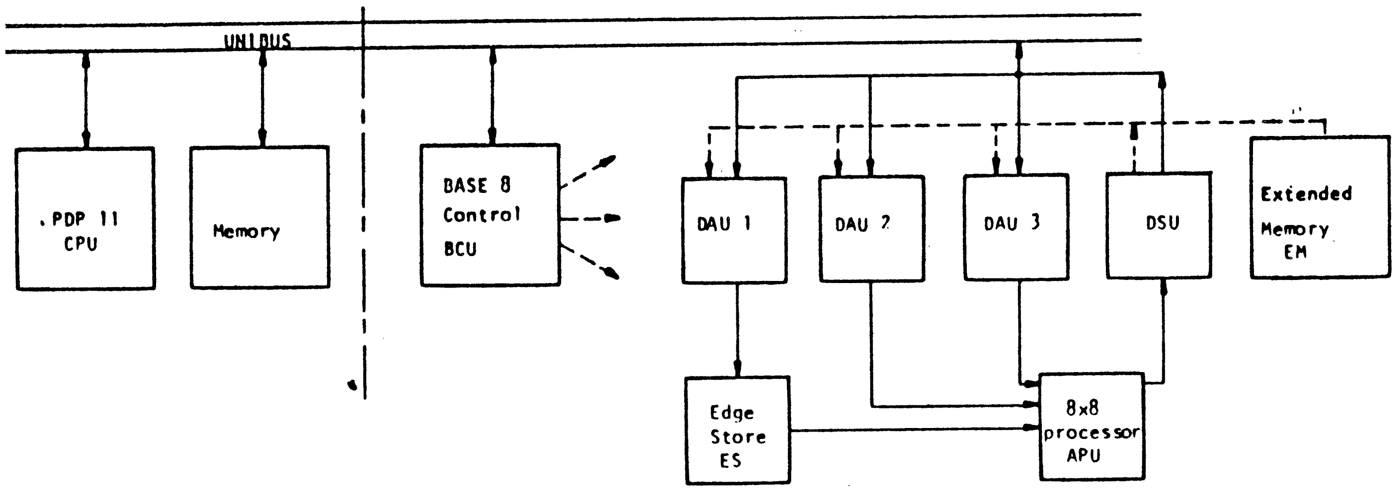

Figure 11
BASE 8 System Organization

needs neighbor values for an edge pixel.

Operations that the BASE 8 system has been used for include the Robert's cross gradient, computation of areas of objects in binary images, gray level histograms, nucleated cell detection, and algorithms to fill missing parts in blob images.

4.4. Cytocomputer

The Cytocomputer developed at ERIM [13], [14] consists of a pipeline of programmable stages, each capable of performing a simple Boolean function. A diagram is given in Figure 12.

Each clock cycle a stage will produce an output (after the system is initialized) which is piped as input to the next stage. The main advantages are the fact that each stage is programmable, any number of stages may be strung together (though this may require intermediatary storage of the image), and the extremely high speed. A more complex type of stage has also been implemented. It is called a 3-D stage and uses 8 bit gray level data instead of a bit plane map. The function produced by a 3-D stage is generated through a lookup table. The operations it is capable of performing are motivated by a set algebra developed by Matheron and Serra.

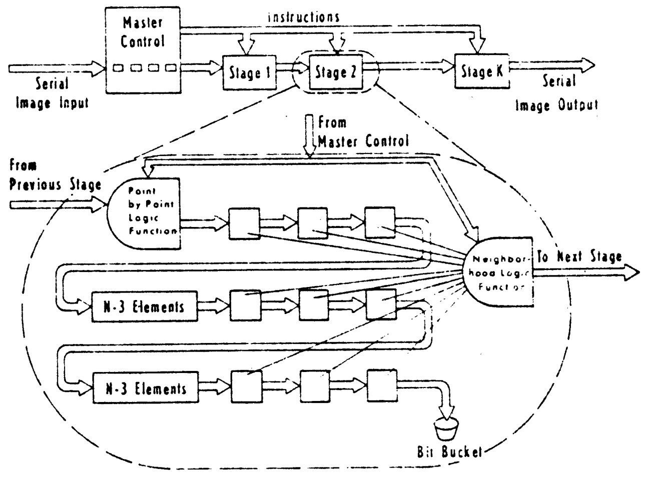

Figure 12
Cytocomputer

5. Special Purpose Machines

These machines consist of special high speed processors which operate in conjunction with a general computer system. Most are used for low level "smoothing" operations, which require a simple function to be applied to an entire image. Their inflexibility allows them to be built to achieve very high speeds.

5.1. Hardware Convolver

The Universal Picture Computer (UPIC) [15] developed at the University of Leuven, Belgium, is another machine which decentralizes computer power, delegating computations to a number of modules which perform a single image operation. One of these modules is a hardware convolver which calculates the convolution of a 3x3 window of pixels with a programmable mask. It boasts a speed of 40 msec for execution over a 512x512x8 bit image. Figure 13 is a diagram of this module.

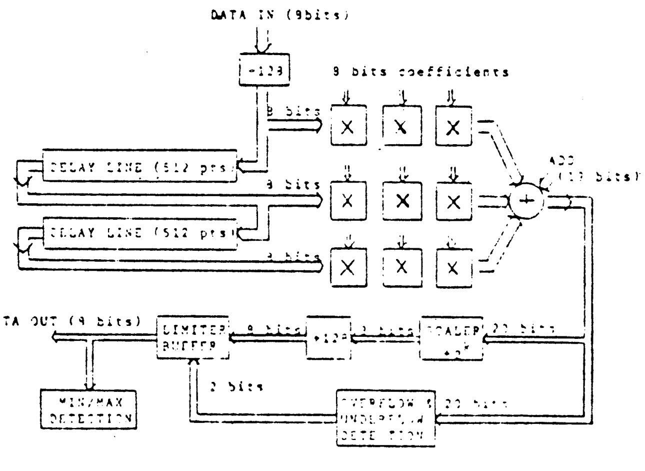

Figure 13
UPIC Hardware Convolver

As the data is input, 128 is subtracted in order to preserve a dynamic range of 256 gray levels. The data is put into the delay lines in order to have the processor operate on three rows of the picture. The data values are then brought into 9 TRW multiplier-accumulators. This produces a 20 bit result, which is divided by 2^k (where k is selectable) for scaling. This value is then output as a final result. Circuitry is also available to detect under and overflow when the value of 2^k is too small or too large. On an underflow, the output is forced to 00_{16}, and on overflow clamped to FF_{16}. Logic also determines the the maximum and minimum values output., which can be read by the controlling computer in order to recalculate the scale factor 2^k. The main weakness of the system is its handling of boundary values, where it assumes that the picture is circular. This results in a degradation of the border of an image.

5.2. GOP

The General Operator Processor (GOP) at the Picture Processing Laboratory at Linkoeping University in Sweden [16] is a more general machine for convolution. It is to be used as an attached processor to a general purpose computer system. Like the previous processor, it also computes 3x3 window functions, but it also allows for a number of parameters to be programmed into the system, which enables insertion of functions with a high degree of non-linearity. The block diagram is in Figure 14.

Part A is a reconfigurable pipelined parallel processor which performs the computation of the product terms. It achieves high speed by having a cache memory inside for neighborhoods of image segments. Data is moved in, one line at a time, where it replaces the oldest line. It allows for up to 16 images to be input at once, and can work with operators of up to 64x64. The terms produced are stored in buffer memory.

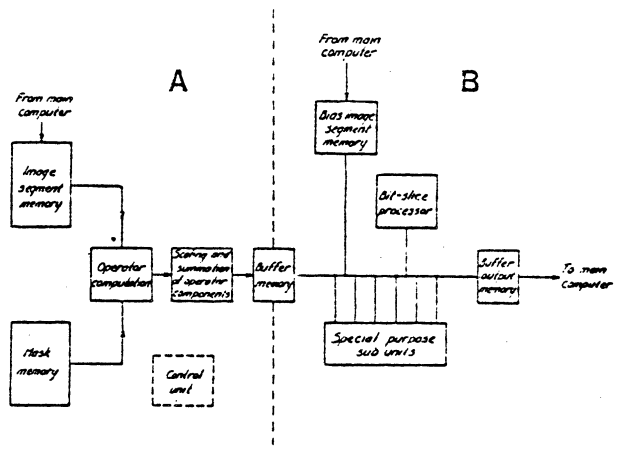

Figure 14
GOP Convolver

Part B has a serial structure organized around a bit-slice processor. The bit-slice part is used to control a few sub-units which perform tasks such as multiplication, rescaling, and accumulation of products. It's micromemory is directly programmable through the main computer. A great deal of memory space is allocated to use for lookup tables in order to speed up complicated procedures. It is this portion of GOP that biasing information may be fed in to affect the processing of the input picture.

Processing a 512x512 image with a 3x3 operator takes about 0.5 seconds on GOP. For larger operators the processing time is approximately 30-40 nsec per pixel processed. One of the unusual aspects of this machine is its ability to process a combination of images. This can be useful for applications such as relaxation processes, where it would be feasible to look at a number of different levels of the image during processing all at once.

5.3. A VLSI Pyramid Machine

The machine proposed in [17] is a study into the feasibility of building a tree-structured parallel architecture in VLSI technology. It would provide an ideal organization for executing many quadtree and pyramid algorithms used in image processing.

Binary tree structures can be implemented very efficiently in VLSI, using what is known as an H structure (see Figure 15). The appealing aspects are that it is a highly regular layout both in processor location and in interconnection design. This greatly reduces the time and effort needed for design and implementation of the chip. Each node in this structure would consists of a simple processing unit. From the tree structure each node would be connected to its four brothers, its four sons, and its father node. It can be shown that, by using current projections for semiconductor technology a pyramid for a 512x512 image could be implemented using only a set of 272 chips by 1990. The one drawback to this system is that all of the data would have to be

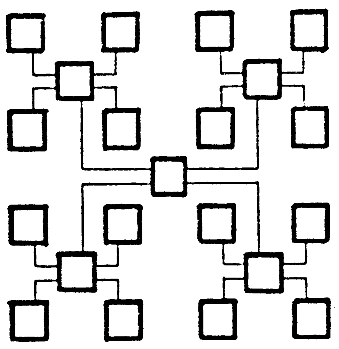

Figure 15
Binary Tree Using an H structure

transmitted serially to each node, adding much to the computation time.

5.4. An Analog Relaxation Processor

One of the more unusual machines proposed has been an analog relaxation processor [18]. Unlike any of the other processors this one would perform computations through analog circuitry (for example, op amps). This is suggested because of the extremely large number of computations required for relaxation processes and the very high speed of analog circuits.

In its very simplest form, a relaxation process assigns a probability with each pixel as an estimate of that pixel residing on a particular label. An iterative technique is applied in order to produce new probability assignments for that pixel based on the current probabilities of a small neighborhood of pixels. An analog processor used for these computations would have the following advantages: (1) the input images are never digitized, (2) a parallel implementation in which calculations for a pixel are piped promises to be much faster than current digital techniques, and (3) parallel analog hardware is likely to be simpler, requiring fewer components than an equivalent digital circuit.

5.5. ZMOB

ZMOB is a "standard" SIMD machine being built at the University of Maryland Computer Science Department [19]. It consists of a MOB (256) of Z80-A microprocessors. These are all attached to one main controlling unit.

All of the processors are connected through a high speed 48 bit wide circular bus. Each one has the capability of sending messages to any other one, to all of them, or to a selected group. A reasonably sophisticated protocol is used for relaying these messages. ZMOB's control unit also uses this bus for broadcasting control instructions to the entire system. This control unit is interfaced externally to a host via a UNIBUS.

At present, a two unit prototype has been built and is fully functional. The next step is to build an 8 or 16 unit machine, and eventually put together the entire system. When fully functional, it is expected to have a 100 million instructions per second throughput, with a 16 million byte memory.

5.6. PICAP II

The PICAP II system at the Electrical Engineering Department at Linkoeping University in Sweden is a general image processing system which is designed around a number of special purpose high speed processors. A large (4 Mbyte) high speed parallel memory communicates over a time-shared bus to these processors. Two of the main features of this system are its time-shared bus and its filter processor (FIP) [20], [21].

A diagram of FIP is shown in Figure 16. It consists of a 32K byte local memory with four subprocessors. The local memory contains full horizontal lines of a picture. The number of lines in this memory depends on the size of the image. This also limits the size of the operator that may be applied. For instance, a picture with a line-length of 512 pixels can be processed by an operator with a maximum width of 64. Each of the subprocessors is run in MIMD mode. They are pipelined and are used to perform the following function:

$$AC \ll F_2(AC,F_1(\omega,MAP(p)))$$

P : a pixel
ω : a constant set in the instruction
MAP(.): pixel to pixel mapping
AC : accumulator
F_1,F_2 : linear and non-linear functions

For the case of simple convolution, F_1 would be multiplication and F_2 would be addition. Other configurations allow for non-linear functions as well.

The memory modules are assigned an addressing scheme where each one contains the next 16th address. For example, module 1 contains address 1,17,33,... The bus this interfaces to is 32 bits wide and has time slots of 100 nsec. The memory modules have a cycle time of 400 nsec. The contention for the bus is resolved every clock. This means that there is no reservation system for it, and this has the tendency to keep the data flowing smoothly. The maximum data rate is 40 Mbytes per second, which compares favorably with the fact that a TV camera can deliver data in the range of 10 Mbytes per second.

6. Conclusions

A number of different processors used for image processing have been presented. It has been shown that there exists quite a few different structures to handle the massive computations and I/O problems inherent in this field. If any trend can be found, it has been to throw more CPU and memory at the problem. The continuing decline of these costs will surely add to more massively designed processors in the future. The concerns which will have to be addressed by then will be languages, algorithms, and better communication between the people who build the machines and the people who use them.

REFERENCES

[1] P.H. Swain, H.J. Siegel, and J. El-Achkar, "Multiprocessor Implementation of Image Pattern Recognition: A General Approach," *Proceedings of the 1980 IEEE Conf. Pattern Recognition*, pp. 309-317, 1980.

[2] M.R. Portnoff, R.J. Sherwood, and R.E. Twogood, "Digital Image Processing Via an Array Processor," *Proceedings of the 1981 IEEE Conf. Pattern Recognition and Image Processing*, pp. 326-328, 1981.

[3] P.R. Wavish, "A Fast Picture Processor," *Proceedings of the 1980 IEE Conf. Pattern Recognition* , pp. 282-285, 1980.

[4] D. Sherdell, "A Low Level Architecture for a Real Time Computer Vision System," *Proceedings of the 1980 IEEE Conf. Pattern Recognition*, pp. 290-295, 1980.

[5] D.C. Van Voorhis and T.H. Morrin, "Memory Systems for Image Processing," *IEEE Transactions on Computers*, vol. C-27, no. 2, pp. 113-125, Feb 1978.

[6] F.A. Briggs, K. Hwang, K-S Fu, and M. Dubois, "PUMPS Architecture for Pattern Analysis and Image Database Management," *Proceedings of the 1981 IEEE Conf. Pattern Recognition and Image Processing*, pp. 387-398, 1981.

[7] H.J. Siegel, L.J. Siegel, F.C. Kemmerer, P.T. Muller Jr., H.E. Smalley, and S.D. Smith, "PASM: A Partionable SIMD/MIMD System for Image Processing and Pattern Recognition," *IEEE Transactions on Computers*, vol. C-30, pp. 934-947, Dec. 1981.

[8] Karl Luetjen, Peter Gemmar, and Harmut Ischen, "FLIP: A Flexible Multiprocessor System for Image Processing," *Proceedings of the 1980 IEEE Conf. Pattern Recognition*, pp. 326-328, 1980.

[9] M.Duff, "CLIP4: A Large Scale Integrated Circuit Array Parallel Processor", pp. 728-732, 1976.

[10] S.T. Tanimoto, "Systolic Celluar Logic," *Proceedings of the 1981 IEEE Conf. Pattern Recognition and Image Processing*, pp. 306-309, 1981.

[11] C. Mead and L. Conway, in *Introduction to VLSI Systems*, Reading, Massachusetts, 1980.

[12] A.P. Reeves and R. Rindfuss, "The Base 8 Binary Array Processor," *Proceedings of the 1979 IEEE Conf. Pattern Recognition and Image Processing*, pp. 250-255, 1979.

[13] R.M. Lougheed and D.L. McCubbrey, "The Cytocomputer: A Practical Pipelined Image Processor," *Proc. 7th Int. Symp. on Computer Architecture*, 1980.

[14] S.R. Sternberg, "Architectures for Neighborhood Processing," *Proceedings of the 1981 IEEE Conf. Pattern Recognition and Image Processing*, pp. 374-380, 1981.

[15] P. Wambacq, L. Van Eycken, J. De Roo, A. Oosterlinck, and H. Van den Berghe, "Description of Two Hardware Convolvers as Part of a General Image Computer," *Proceedings of the 1981 IEEE Conf. Pattern Recognition and Image Processing*, pp. 294-296, 1981.

[16] G.H. Granlund, "GOP, A Fast Flexible Processor for Image Analysis," *Proceedings of the 1980 IEEE Conf. Pattern Recognition*, pp. 489-492, 1980.

[17] C.R. Dyer, "A VLSI Pyramid Machine for Hierarchical Parallel Image Processing," *Proceedings of the 1981 IEEE Conf. Pattern Recognition and Image Processing*, pp. 381-386, 1981.

[18] C.R. McLean and C.R. Dyer, "An Analog Relaxation Processor," Proceedings of the 1980 IEE Conf. Pattern Recognition, pp. 58-60, 1980.

[19] C. Reiger, "ZMOB: Hardware from a User's Viewpoint," *Proceedings of the 1981 IEEE Conf. Pattern Recognition and Image Processing*, pp. 399-407, 1981.

[20] B. Kruse, B. Gudmundsson, and D. Antonsson, "FIP - The PICAP II Filter Processor," *Proceedings of the 1980 IEEE Conf. Pattern Recognition*, pp. 484-488, 1980.

[21] Per-Erik Danielsson, "The Time-Shared Bus - A Key to Efficient Image Processing," *Proceedings of the 1980 IEEE Conf. Pattern Recognition*, pp. 296-299, 1980.