A Novel Multiprocessor Architecture for Low-Level Image Processing Applied to Road-Traffic Data Capture and Analysis

M. Atiquzzaman
Dept of Electrical & Computer Systems Engg.
Monash University, Clayton
Melbourne 3168, Australia.
atiq@eng.monash.edu.au

M.G. Hartley
Dept of E.E. & E
UMIST
Manchester M60 1QD
United Kingdom.

M. Y. Siyal
School of E.E.E
Nanyang Technological Univ.
Nanyang Avenue
Singapore 2263.

Abstract

A highly parallel multiprocessor architecture for low-level processing of digital images is described in this paper. It is based on the ideas of distributed memory, data replication to reduce the need for interprocessor communications, and dedicated links between processors to eliminate the need for a common bus for interprocessor communications.

1 Introduction

There is a close relationship between image-processing algorithms and the type of architecture adopted for their execution. To prove effective an architecture must reflect the type of algorithm to be implemented. This calls for the design of algorithm-structured architectures for efficient real-time processing of pictures. This paper describes the novel features of a proposed multiprocessor architecture for the efficient parallel execution of low-level image processing algorithms in real-time. The application under consideration relates to a road-traffic situation for which vehicles must be counted and velocities determined in order to permit on-line urban traffic control strategies and motorway incident detection. However, the ideas propounded may be applied to a wide range of image-processing situations where real-time operation is essential and high-definition pictures are less important.

Vision is an area where parallelism occurs naturally and where parallel algorithms can be used to advantage [1]. Multiprocessor systems executing parallel algorithms can be used to exploit the parallelism inherent in image-processing algorithms. Moreover, the two-dimensional nature of images makes them very suitable for mapping onto a multiprocessor system.

Image-processing algorithms can be broadly classified into two main groups; Pixel-level operations and Region-level operations. Pixel-level operations take an input image into an output image, i.e. a different image is produced after processing an input image. Pixel-level operations are again subdivided into point and local operations, transforms, geometric operations and measurement of properties. Region-level operations determine the properties of regions in an image from the representation of the regions of an image, for example, specification of a region by border code.

Low-level image-processing algorithms are usually limited in scope and repetitive in nature and hence are good candidates for parallel processing. Point and local operations are well-suited for execution in Single Instruction Multiple-Data (SIMD) stream machines [2]. On the other hand, region-level operations are well-suited to ring-type architectures [3]. The algorithms for road-traffic scene analysis are mostly a combination of point and local operations and region-level operations. Examples of point and local operations are frame subtraction, spatial averaging, local edge-detection and thresholding. Finding parameters like the area, centroid and perimeter of an object are examples of region-level operations.

Over recent years much success has been achieved with image-processing systems designed to measure vehicle counts and velocities. The serial nature of such architectures has limited speed of operation to several times slower than real-time but much useful experience has been gained. This paper describes a tentative Multiple-Instruction Multiple-Data (MIMD) stream architecture, geared to the efficient parallel execution of point and local operations as well as region-level operations with emphasis on an economical design and real-time operation.

2 Concurrent loading and processing

An enormous amount of data is produced from a vision sensor, typically a camera. Loading and unloading of data into and from a subsequent processor or processors takes a considerable amount of time and is comparable to the actual computation time [4], especially in the case of low-level image-processing systems, as explained below. The total time taken to handle a picture may be represented by:

$$t_h = t_l + t_p$$

where

t_h = total time taken to handle a picture,
t_l = time taken to load a picture,
t_p = time taken to process a picture.

The time taken to load a picture (t_l) into a machine from a visual sensor depends on the architecture of the machine and is independent of whether the processing is low or high-level. Processing time (t_p) is much smaller in the case of low-level image-processing operations than in high-level operations. Therefore, the ratio t_l/t_p is higher in the case of low-level image-processing than in high-level processing. Consequently, special attention should be given to reduce the data loading time in low-level image-processing machines.

In most of the multiprocessor systems for image-processing, processing of a picture starts after a full picture has been loaded into the machine. In the CLIP system [5], for example, the pixels of a picture are shifted serially through the processor elements until the whole picture is loaded and this operation is then followed by processing. In the ZMOB [3], pixels of a picture are circulated in a rotating loop from whence the pixels are picked up by the processors connected to the loop. This means that the machines cannot start processing until the whole picture has been output from the camera.

Most image-processing machines use some sort of intermediate storage, either high-speed buffers or hard disk, before the pictures are loaded into the machine for processing. The storing of pictures in intermediate storage, and subsequent loading of the pictures into the different processors of the machine for processing takes a considerable amount of time, as the process has to be done serially. Inevitably this method of loading data into the machine tends to slow down the system.

The multiprocessor system described in this paper consists of a linear array of N processor modules

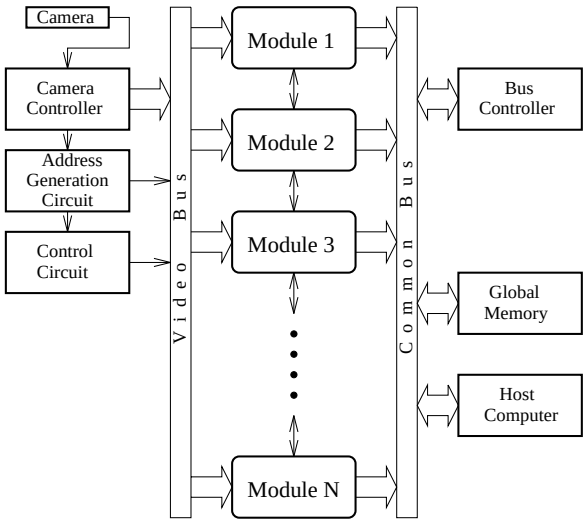

Figure 1: The proposed multiprocessor system for low-level image processing

connected to their neighbours through dedicated interprocessor communication links (Figure 1). Pictures are loaded straight from the camera system into the processor modules instead of the pictures being stored in intermediate storage after being output from the camera. Copying of pixels at a later stage from intermediate storage into the Local Memories of the processors through a common bus or ring is thus avoided.

It will be seen in Section 3 that the design advocated requires a processor for a fixed number of lines of a picture. It will also be seen that the pixels of each line of a picture are required by m (typically three or five) adjacent processors in order to facilitate neighbourhood operations. Each line of a picture is therefore loaded into three consecutive processor modules. When a processor module is being loaded with pixels, the processor is halted by an external HALT signal, thereby preventing the processor from premature access to it's Local Memory which is being used by the camera to load pixels. Therefore, at any instant, m consecutive processors are halted and the rest of the processors are processing pixels.

As soon as the m lines of a picture have been written into a processor's Local Memory, the processor is restarted by an external signal. The processor, after being restarted, starts processing while the rest of the lines of the picture are being output from the camera and loaded into the other processor modules of the system. Thus the processors need not wait

till the full picture has been output from the camera. The simultaneous loading and processing of a picture saves time spent for loading a picture, in contrast to machines like CLIP and ZMOB, where processing of a picture starts after the full picture has been output from the camera and loaded into the machine.

3 Data Replication

In a multiprocessing environment, a task is broken up into several sub-tasks, each of which is executed by a separate processor. This normally gives rise to the need for the processors to communicate among themselves. Inter-processor communication has always been a problem in the design of multiprocessor systems.

Inter-processor communication using a common bus slows down a system and is a major bottleneck in most multiprocessor systems. Using a ring instead of a common bus alleviates the problem, but does not remove it completely. In the case of communication through a ring, a processor communicates with another processor using the ring. The time required for two processors to communicate depends on the direction of data flow in the ring relative to the two processors involved in the communication. Depending on the direction of data flow in the ring, communication between two near-neighbour processors may require a substantial time.

To minimise the need for interprocessor communications, a novel technique has been implemented in the multiprocessor system described in this paper. Data are replicated among different processors as soon as the data are generated. No inter-processor communication is necessary for the first pass over a picture. The inter-processor communication required for subsequent passes is performed through point-to-point dedicated links between processors instead of using a common bus or a ring.

In the system described in this paper, one processor is assigned to one line of an $N \times N$ picture. In the interests of high-speed operation it is important to ensure that the image-processing algorithms involving pixel and point operations do not require any inter-processor communications during the first pass. How ever neighbourhood operations on any pixel require the values of adjacent pixels on the same line and nearby lines. For example, a typical 3×3 pixel operation such as averaging or filtering results in the replacement of every pixel in the picture following a convolution involving contributions from the original value and its eight immediate neighbours. Of the nine pixels required, each processor

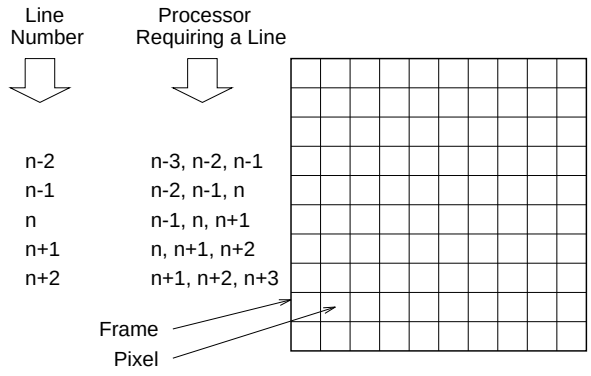

Figure 2: Lines of a picture required by different processors for a 3×3 neighborhood operations.

of the machine has three pixels in its own line. The other six pixels belong to the lines immediately preceding and succeeding it. Thus for a 3×3 neighbourhood operation, each processor requires three consecutive lines of a picture. It would appear that this requirement gives rise to the need for inter-processor communication to permit exchange of pixel information. This is a time-consuming phenomena which would slow down the speed at which the system operates.

The need for inter-processor communication to fetch pixels from neighbouring lines for $m \times m$ neighbourhood operations has been eliminated in the proposed multiprocessor system by replicating each line three times and writing each line of a picture into the three adjacent processor modules which will require it. For example, line number n is required by processors $n-1$, n and $n+1$ (Fig. 2). Therefore, line number n will be simultaneously written to the Local Memories of modules $n-1$, n and $n+1$ as soon as it (line number n) is output from the Camera System.

Replication of each line three times means replication of a picture three times. Three times more Local Memory (RAM) is required than would be necessary for storing a single picture. The increase in cost due to additional memory is insignificant as compared to the gain in speed of the overall system and to the overall cost of the system. For the more general case, $n \times n$ neighbourhood operations will be possible by replicating each line n times, where n is an odd integer. As in the 3×3 operations no interprocessor communication will be necessary.

Note that the method proposed represents a compromise involving serial processing applied to N consecutive pixels along each line of the picture. For

the highest speed, computing power together with limited local RAM storage of relevant pixel data is required at each of the N^2 pixels of the complete picture. Such an arrangement is not yet practical other than by the expenditure of very substantial resource. The proposed scheme represents a workable compromise.

4 A Novel Multiprocessor Architecture

The rest of the paper describes, in rather more detail, the tentative architecture of a multiprocessor system for processing digitised pictures, typically of resolution $N \times N$, in real-time. The multiprocessor system (Fig. 1) is a Multiple-Instruction Multiple-Data (MIMD) stream machine having N processor modules, each module being responsible for processing one line of an $N \times N$ picture. Each of the modules is connected to two buses - the Video Bus and the Common Bus. The Video Bus comprises address, data and control lines.

A Camera System consisting of a $N \times N$ photodiode camera and a camera controller is connected to the Video Bus. Pictures are continuously output from the Camera System onto the Video Bus, from whence the pixels of the pictures are written directly to the different modules.

The Address Generation Circuit derives its input from the Camera System and, at any instant, generates the address of the pixel on the Video Bus at that instant. The address comprises the x and y co-ordinates of the pixel on the Video Bus. The y co-ordinate is fed to the address lines of the Video Bus. The x co-ordinate is fed to the Control Circuit which determines the processor modules which should receive the pixel residing on the Video Bus at that instant. The output of the Control Circuit is connected to the control lines in the Video Bus. The control lines are used for writing pixels into the individual processor modules.

A Host Computer co-ordinates the activities of the whole machine and is responsible for every type of decision. The processor modules pass the results of processing of the pixels of their line to the Host Computer through the Global Memory. The results are written to the Global Memory through the Common Bus. The Host Computer then fetches the results from the Global Memory for the final phase of the processing.

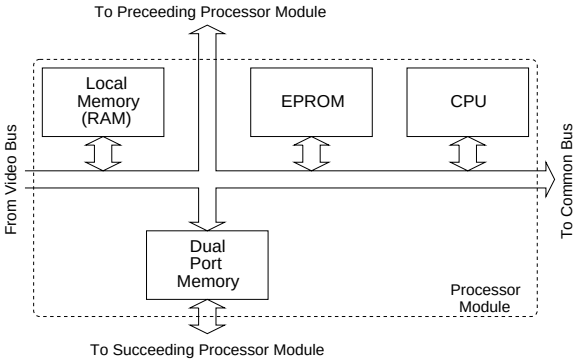

Figure 3: A processor module

5 Processor Modules

Each processor module is a small computer in itself (Fig. 3). Each has-its own CPU and EPROM for storing resident programs, Local Memory (RAM) for storing data and use as a scratch-pad memory, two communication links to permit communication with the two immediate-neighbour modules in any subsequent pass (should this prove necessary) and I/O facilities to communicate with the Video Bus and the Common Bus. The local RAM of a module can be accessed by the CPU as well as by the Camera System. Arbitration circuitry controls the access to the local RAM by the CPU and the Camera System.

The Control Circuit sends a signal to a processor module to indicate that pixels of a picture are to be written to the processor's Local Memory. The Camera System has a higher priority than the CPU over the Local Memory, and hence the CPU releases the control of the Local Memory at the end of the machine cycle. When the Camera System is writing pixels to a Local RAM, the corresponding processor enters a WAIT state and cannot access the RAM. The camera system writes m consecutive lines of a picture to each Local RAM. At the end of writing of the pixels into the RAM by the Camera System, the processor quits the WAIT state and processes the data that has been written into the RAM.

The machine can perform a wide variety of image-processing operations without any interprocessor communication. With the proposed configuration the machine would be able to perform point and pixel-level operations as well as 3×3 neighbourhood operations without the need for a processor to communicate with its neighbouring processors. In the case where a second pass over a picture is necessary, the modified values generated during the

first pass may need to be exchanged between processors. Interprocessor communication in such cases is performed through dedicated links between processors. Certain region-level operations like area measurement and centroid calculation of an object can also be carried out without any interprocessor communication. The system can be further extended to execute $n \times n$ neighbourhood operations without the need for any inter-processor communications merely by extending to n the number of lines made available to the RAM associated with each processor module.

Each processor has its own program stored in the corresponding EPROM. The processors work asynchronously; they start processing as soon as data is ready in their Local RAMs. Lines of a picture are output from the Camera System starting from the top to the bottom of a picture. Since lines output from the Camera System are written serially to the Local RAMs from the Video Bus, the data becomes available to the different processors one after another starting from processor 1 (the processor assigned to line 1 of a picture) to processor N (the processor assigned to line 100). The processors, therefore, start processing one after another beginning from processor 1 and continuing to processor N.

For example, in a 3×3 neighbourhood operation on a 100×100 picture, processor 3 starts processing as soon as all the pixels of line numbers 2, 3 and 4 have been output from the Camera System. Processing by processor 3 continues in parallel to the output of the rest of the lines of the picture. Processor 3 finishes processing before the output of line 2 of the next frame from the Camera System. The maximum frame rate at which the camera can be operated for real-time analysis is therefore determined by the time taken by a processor to process the pixels of one line of a 100×100 picture.

The processors pass the results of processing of the pixels to the Global Memory using the Common Bus. The processors execute the same algorithms and because of the synchronous nature of operation, the processors finish processing one after another. Therefore, the requirement of the different processors to use the Common Bus also arises one after another. This avoids the simultaneous requirement for the Common Bus by the different processors; the processors use the Common Bus one after another in their communication of results such as edge-coordinates to the Host Computer.

6 Conclusion

A novel multiprocessor architecture consisting of N processor modules to process $N \times N$ pictures in real-time has been described. Replication of pixels has made it possible for the machine to execute $m \times m$ neighborhood operations without any interprocessor communication in the first pass over a picture. Interprocessor communications required in subsequent passes are accomplished through dedicated links between processors.

References

[1] M. Brady, "Parallelism in vision," *Artificial Intelligence*, vol. 21, pp. 271–283, 1983.

[2] A. Rosenfeld, "Parallel algorithms for image analysis," in *Modern Signal Processing* (S.Y. Kung, H.J. Whitehouse, and J. Kailath, eds.), Prentice Hall, 1985.

[3] T. Kushner, A.Y. Wu, and A. Rosenfeld, "Image processing on the ZMOB," *IEEE Transactions on Computers*, vol. C-31, no. 10, pp. 943–951, October 1982.

[4] M. Atiquzzaman, "Performance of a linear array multiprocessor under different data loading methods," *Microprocessing and Microprogramming*, vol. 36, no. 4, pp. 167–178, September 1993.

[5] T.J. Fountain and V. Goetcherian, "CLIP 4 parallel processing system," *IEE Proceedings Part-E*, vol. 127, no. 5, September 1980.