MVA'98 IAPR Workshop on Machine Vision Applications, Nov. 17-19, 1998, Makuhari, Chiba, Japan

13—1

A Comparison of Linear Processor Arrays for Image Processing

Matthijs van der Molen
matthijs@ph.tn.tudelft.nl

Pieter Jonker
pieter@ph.tn.tudelft.nl
Pattern Recognition Group
Delft University of Technology *

Abstract

This paper describes a comparison of the performance and usability of three Linear Processor Arrays for image processing purposes. The study covered the NEC IMAP-VISION card [1], the ASI CNAPS card [2] and the SYMPHONIE of the CEA-LETI [3]. The manufacturers of these card have made different design decisions regarding the number of processing elements (PEs) and their complexity. It was found that for image processing many simple PEs are more useful than few powerful processors.

1 Introduction

The invention of ever faster and more complex computers makes more and more applications of digital signal processing possible. And in turn, new application areas require increasingly more computational power. One of the most computationally intensive areas is that of (real-time) image processing, due to the amount of data involved and the operations needed to extract information from complex real-world images.

To overcome the limitations imposed on the power of a single processor by waver size, parallel computer systems have been designed and built. By the nature of the data (many pixels which undergo the same treatment) Single Instruction, Multiple Data (SIMD) processors are the most natural choice for image processing applications. It has been shown by Komen [4] and later by Jonker [5] that a Linear Processor Array (LPA) is the best interconnection model for an SIMD image processor. In this study three LPAs which are, or will soon be, commercially available are compared with respect to performance and possibilities/limitations.

2 The Cards

Each LPA is an add-on card, which is connected to a host computer. The processing algorithms are coded and compiled on the host, using a language and software proper to the card. The host will then upload the binary program to the cards program memory and launch it. Communication between the host and the card happens over a data bus and through interrupts. The cards consist of three principle components:

The Control Processor (CP) The CP is a single microprocessor or DSP which handles the communication with the host. Furthermore it controls the program flow, once a program has been launched by the host. Finally it controls data I/O over the data bus.

The Data bus In order to do real-time image processing, not only fast calculations are required, but also a high data throughput. A dedicated data bus has been put in place in the cards in order to assure this. In the IMAP-VISION, this bus is connected to Video Digital-to-Analog (DAC) and Analog-to-Digital (ADC) Converters, allowing the card to be connected directly to a camera and a monitor/TV.

The Processing Elements (PEs) Probably the most important element is the array of PEs. It is a set of microprocessors, each connected to two other processors, called its left and right neighbors. It are these processors which do the data processing. Probably the most important decision when designing an LPA is the complexity and number of PEs to be put in place. Due to limitations on chip surface and on the number of transistors per unit chip surface, more making the PEs more complex (= more powerful) means less PEs can be placed on the card. A lack of experimental knowledge about the effect of the decision on the performance of the card has led the three manufacturers of the cards used in this study to make different choices.

The main characteristics of each of the cards are:

*Address: Lorentzweg 1, 2628 CJ Delft, the Netherlands

2.1 The IMAP-VISION

NEC's Pattern Recognition Laboratory, now called Incubation Center, has developed the IMAP-VISION. The IMAP-VISION is available in a PCI and a VME bus version. The card comprises the aforementioned video I/O interface. It is controlled by a 16-bit microprocessor, containing an adder, a shifter and a multiplier. This CP executes its instructions for data I/O, program flow control and host communication in parallel with the data processing by the PEs. The PE array consists of 256 processors, containing an 8-bit adder and shifter each, plus some hardware for accelerated multiplication (in 8 clock-cycles) and 1 kilobyte (kB) of internal memory. Finally they have an interface to an additional 64 kB of external memory. In the programming model the memory is regarded as a common, 256-byte wide, two-dimensional plane of memory, of which each column is treated by one PE.

2.2 The SYMPHONIE

At the moment the SYMPHONIE does not yet exist as a card. Its characteristics have been studied using a software simulator, furnished by the LETI. The card will have a 32-bit CP containing an adder and shifter. It has not yet been decided if the data bus will be connected to the PCI-bus host interface, or to a dedicated data interface. A single card will contain 64 PEs, but multiple cards can be connected to form a single system of up to 1024 PEs. The PEs are 32-bit super-scalar processors, meaning they consist of an adder and a shifter/multiplier which can operate in parallel. Unlike the IMAP-VISION the CP and PEs can not operate in parallel. The PEs are connected by a complex interconnection network which can operate in different modes: data on all PEs can be shifted 1 PE to the right or left, data on 1 PE can be broadcast to all other PEs and 1 PE can write directly in the memory of 1 other PE (DMA mode). The PEs will have access to 16 kB internal memory and 64 kB external memory, but the latter is not yet documented or implemented in the simulator. A peculiarity of the SYMPHONIE is the memory distribution. The memory is viewed as a 16-bit 1024 by 512 byte 2-dimensional plane of which the elements are distributed over the PEs helicoïdally: instead of each PE having one column or one line of memory, each has a diagonal. Thus there is parallelism for both column and line operations. A disadvantage however is that the programmer has to do a lot more thinking, about where a specific pixel resides, and for many algorithms precious clock-cycles are lost calculating at which PE a certain pixel is located. This is especially true for so called bucket-processing [6] or stack-processing [7][8] algorithms, which do runtime region-of-interest (ROI) calculation.

2.3 The CNAPS

The CNAPS holds the middle as far as PEs are concerned: a CNAPS board can contain from 16 to 512 PEs in VME-bus version and 64 or 128 PEs in PCI-bus version. The card we used contains 128 PEs. Each PE is a 16-bit microprocessor, containing an adder and a shifter/multiplier. They can not work in parallel independently, but it is possible to perform a multiply-and-accumulate in one clock-cycle. The CP is a 32-bit DSP, which can not be programmed explicitly. Only some predefined control operations exist for data I/O. Global data manipulation has to be done on one of the PEs, leaving the others inactive. Contrary to the other cards there is no unified memory view. The card is considered to have N independent memories. This has the advantage that data can be distributed according to the type of data and operations. The disadvantage however is that all data I/O has to be done sequentially via the CP. Data can be transferred using the host interface or a dedicated I/O path, called Direct-I/O. A third possibility is a set of data paths each connected to a subset of PEs, called Quick-I/O, allowing a certain extend of parallel data I/O. A major disadvantage is that the PEs are only connected through a 4-bit bus (2 in, 2 out) to each of its neighbors. Thus data would have to be sent to a neighbor two bits at a time, which is only possible in the card's assembly language. In the high-level language the only solution would be to have the PEs output their value(s) to the CP sequentially, and have them read it in a different order. This makes the CNAPS not very suitable for image processing, because that requires almost always neighborhood operations.

3 The comparison

The cards were compared using a number of more or less standard image processing algorithms, known from literature. They were chosen so as to represent most of the categories described by Kyo and Sato [7], covering different levels of mathematical complexity and different amounts of communication. They all operate on 8-bit gray-level images. Unfortunately no timings are included for the CNAPS, since the card could not be made to work in our laboratory environment. The algorithms used are:

Binarization (Point Operation) This very simple algorithm, also called thresholding, compares the value of each pixel to a threshold. If the value is greater or equal, the corresponding pixel in the output image is set to 255, else

it is set to 0. This 8-bit operation, which requires no (neighborhood-) communication tests the "raw" speed of the cards. The IMAP-VISION clearly comes out fastest (see Table 1).

Sobel filter (Local Neighborhood Operation) This operation applies both a horizontal and a vertical Sobel edge detection filter, and sums and thresholds the absolute values of both results, in order to find strong edges in the image. It uses the values of all 8 neighbor pixels to do some very simple calculations, so it mainly measures neighbor-PE communication speed. For this algorithm, the IMAP-VISION is still about twice as fast as the SYMPHONIE, but the difference is less then for thresholding, because 16-bit operations are used, which take two clock-cycles on the IMAP-VISION. Also the SYMPHONIE can profit some more of the internal parallelism of the PEs.

SDGD (Local Neighborhood Operation) For a kernel size of 3x3 the Second Derivative in the Gradient Direction uses only neighbor pixels, but it involves a lot more computation than the Sobel filter. For each pixel 6 integer (16-bit) multiplications and one long (32-bit) division are computed. Therefore the results for this algorithm are more or less an average of computational power and neighbor connection speed. It is found that for this algorithm the SYMPHONIE beats the IMAP-VISION, because of its larger word-size and hardware multiplier. Also quite al lot of internal parallelism is used. It is possible that the CNAPS would have been the fastest card for this algorithm. The lack of neighbor-PE connections can be overcome by loading overlapping stripes of three image columns on each PE. The CNAPS can do all 16-bit operations in one clock-cycle, using its hardware multiplier, and it has twice as many PEs as the SYMPHONIE. On the other hand it has a lower clock frequency.

Rotation over 90 degrees (Point Operation) This algorithm involves no computation, only communication: pixels have to be sent to another processor. On the IMAP-VISION a smart algorithm by Kyo [7] is used. On the SYMPHONIE it uses the DMA mode, which is easy to implement, but not very fast. On the CNAPS an image in its common, so called file memory, can be rotated in 0.0 ms: you simply change the way of distributing it over the PEs. However an image already in PE memory has to be sent to the CP and back to the PEs pixel by pixel. It can be seen that the rotation is done much more efficiently on the IMAP-VISION, even though the SYMPHONIE can use its special interconnection network to send data to another PE in the background. The algorithm used on the IMAP-VISION could not be used on the SYMPHONIE because it only works if there are as many PEs as image columns.

Histogram (Statistical Operation) A histogram is a table in which each element contains the number of pixels that have a gray value equal to the index of the element. It is calculated by counting pixels in parallel and then combining the counting results of each PE, using an algorithm by Kyo [7]. Even though the final result is a 16-bit array, most of the computation is done is 8-bit, so the IMAP-VISION is faster, thanks to its larger number of PEs.

SDT (Recursive Neighborhood Operation) The Strumpford Distance Transform is an algorithm which, for every pixel in an image, calculates an approximation of the Euclidean Distance (in pixels) to the nearest object pixel. It has been implemented both in the form of a recursive neighborhood operation (RNO) and using distributed bucket (/stack) processing. It has has been shown by Olk that the distributed bucket algorithm is faster if four or more RNO-scans are performed over the image. Although the SDT only needs two scans, the distributed bucket algorithm has also been implemented because it allows the speed of the cards for this type of algorithms to be compared. On the IMAP-VISION the distributed bucket-processing algorithm is much slower than the RNO-scans. Remarkably this is not the case for the SYMPHONIE. The overhead involved for the RNO algorithm causes it to be slower, even though only two scans over the image are made.

Hough Transform (Global Operation) The Hough transform is an algorithm for detecting geometrical shapes in an image, in this case lines. For a predefined set of values for the parameters describing the shape, here the slope and the offset of the line, the shape is superimposed on the image. Then the element in parameter space, corresponding to that particular combination of values is incremented for each pixel lying on the shape which has a value higher than a certain threshold. This is the only algorithm implemented for which the SYMPHONIE is clearly much faster than the IMAP-VISION. This is due to the very high amount of multiplications, which are done in 16-bit. On this algorithm the CNAPS may do very well, because the only type of communic-

ation is broadcasting, and it can do the 16-bit multiplications in one clock-cycle.

FFT (Global Operation) The FFT is a fast algorithm for calculating the spectrum of an image. A 2D-FFT can be decomposed in the 1D-FFT over the rows of the result of a 1D-FFT over the columns of the image. Thus, if an image is distributed column-wise over the PEs, each PE performs a 1D-FFT over the column(s) it has in its memory. Then the image is rotated over 90 degrees, the same 1D-FFT algorithm is applied over the rows (which are now columns) and the image is rotated back. On the IMAP-VISION swapping to external memory had to be used during the calculation, because not enough internal memory is available to hold all data. On the SYMPHONIE the memory distribution is a problem. Because the PEs neither have a line nor a column in memory a lot of communication is needed for both 1D-FFTs. It turned out to be faster to redistribute the image such that each column resides on one PE. It turns out that this pre- and post processing, together with the much slower rotation, make the FFT on the SYMPHONIE slower than on the IMAP-VISION, despite the 16-bit x 16-bit multiplications used.

Summary The following table gives the timing results of the described algorithms for the IMAP-VISION and the SYMPHONIE.

Algorithm	IMAP-VISION	SYMPHONIE
Binarisation	* 0.03 ms	0.21 ms
Sobel filter	* 0.56 ms	1.13 ms
SDGD	6.56 ms	* 5.48 ms
Rotation		
- $+90°$	* 0.55 ms	1.70 ms
- $-90°$	* 0.58 ms	1.71 ms
Histogram	* 0.19 ms	0.35 ms
SDT		
- RNO scans	* 6.92 ms	22.19 ms
- buckets	19.87 ms	* 19.67 ms
Hough Transform	48.02 ms	* 21.17 ms
FFT		
- 1-Dimensional	* 23.55 ms	47.67 ms
- 2-Dimensional	* 53.57 ms	97.24 ms

[table 1] Timings of the algorithms (* indicates the fastest card).

4 Conclusions and recommendations

When comparing the results for the IMAP-VISION and the SYMPHONIE, it can be seen that the IMAP-VISION is almost always faster, sometimes by far. And for the algorithms for which it is slower, the difference is often small. This can be explained by Amdahl's Law. Imagine an algorithm which contains 20% multiplications. Using a card with hardware multipliers can do those multiplications 8 times faster than one without. This gives an overall speed-up of the algorithm of 100 / (80 + (20/8)) = 1.21 . If, instead of adding a multiplier, twice as many PEs where used, the overall speed-up whould be around 2. This holds especially for the IMAP-VISION, because it executes sequential code on the CP at the same time as the parallel code on the PEs. This means the PEs are active almost all of the time. Thus for image processing applications more PEs are preferable over complex PEs. However this may no longer hold if the number of PEs exceeds the number of columns (rows) in the image. Furthermore it was found that "smart" hardware, like the SYMPHONIE's interconnection network and memory distribution, are often more of a nuicance than of help. In many cases the problems they are intended to solve can be attacked just as well, or even better, by smart programming. More details about the algorithms, the results and a discussion on the validity of the results can be found in [9].

5 Acknowledgments

The authors would like to thank the following persons for their support and hospitality:

Sholin Kyo and Yoshihiro Fujita
Incubation Center
NEC Research Laboratories

Jean François Larue and Patrick Arnoul
Laboratoire d'Electronique, de Technologie
et d'Instrumentation
Commissariat à l'énergie Atomique

References

[1] Y. Fujita, e.a., "A 10 GIPS SIMD Processor for PC-based Real-Time Vision Applications – Architecture, Algorithm Implementation and Language Support –", Computer Architecture for Machine Perception (CAMP), pp.22-32, 1997.

[2] D.W. Hammerstrom, D.P. Lulich, "Image Processing using One-dimensional Processor arrays", Proc. IEEE, Vol. 84, No. 7, pp.1005-1018, 1996.

[3] D. Juvin, e.a., "SYMPHONIE: Calculateur Massivement Parallèle, Modélisation et Réalisation", Journées Adéquation Algorithmes Archtectures, Toulouse, 1996.

[4] E.R. Komen, "Low-level Image Processing Architectures Compared for some Non-linear Recursive Neighborhood Operations", Ph.D. Thesis, Faculty of Applied Physics, Delft University of Technology, Delft 1990.

[5] P.P. Jonker, "Why Linear Processor Arrays are Better Image Processors", International Conference on Pattern Recognition (ICPR), Vol.3, pp. 334-338, 1994.

[6] J.G.E. Olk, P.P. Jonker, "Bucket Processing: a Paradigm for Image Processing", International Conference on Pattern Recognition (ICPR), 1996.

[7] S. Kyo, K. Sato, "Efficient Implementation of Image Processing Algorithms on Linear Processor Arrays using the Data Parallel Language 1DC", Machine Vision and Applications (MVA), pp.160-165, 1996.

[8] S. Kyo, e.a., "A Parallelizing Method for Implementing Image Processing Tasks on SIMD Linear Processor Arrays", Computer Architecture for Machine Perception (CAMP), pp.180-184, 1997.

[9] M.W. van der Molen, "A Comparison of Linear Processor Arrays for Image Processing", M.Sc. Thesis, Faculty of Applied Physics, Delft University of Technology, Delft, 1998.