

REAL TIME IMAGE PROCESSING-DESIGN ISSUES

Jaspinder Sidhu, Bhupinder Verma, Dr. H. K. Sardana*
Lovely Professional University Phagwara
*Central Scientific Instruments Organisation, Chandigarh
jaspindersingh @lpu.co.in

Abstract: This paper considers the past, present and future of architectures for high performance image processing. After reviewing a number of representative designs of image processing-specific architectures, four current approaches are considered in more detail: standard microprocessor technology, DSP processors, parallel processing and dynamically reprogrammable hardware in the form of Field Programmable Gate Arrays (FPGAs). A case study has been discussed to discuss selection & implementation of a particular approach.

1. INTRODUCTION

Realtime Image processing applications requires handling a large amount of data and it poses a challenge to the machine vision designers. Even a simple situation to examine products of size 64 x 64 pixels moving at rates of 10 to 20 per second along a conveyor amounts to a requirement to process up to 100,000 pixels per second—or typically four times this rate if space between objects is taken into account. The situation can be significantly worse than indicated by these figures. First, even a basic process such as edge detection generally requires a neighborhood of at least 9 pixels to be examined before an output pixel value can be computed. Thus, the number of pixel memory accesses is already 10 times that is given by the basic pixel processing rate. Second, functions such as skeletonization or size filtering require a number of basic processes to be applied in turn. For example, eliminating objects up to 20 pixels wide requires 10 erosion operations, while thinning similar objects using simple "north-south-east-west" algorithms requires at least 40 whole-image operations. Third, typical inspection applications require a number of tasks—edge detection, object location, surface scrutiny, and so on—to be carried out. All these factors mean that the overall processing task may involve anything

between 1 and 100 million pixels or other memory accesses per second. Finally, the above analysis ignores the complex computations required for some types of 3-D modeling, or for certain more abstract processing operations.

These formidable processing requirements imply a need for very carefully thought out algorithm strategies. Accordingly, special hardware will normally be needed for almost all real-time applications. (An exception may be in those tasks where performance rates are governed by the speed of a robot, vehicle, or other slow mechanical device.)

Broadly speaking, there are two main strategies for improving processing rates.

1. The relentless, year on year improvement in the

performance of standard microprocessor technology. Moore's Law describes the phenomenon which has been observed in computer hardware technology, that processing power doubles every 18 months. So merely by doing nothing except waiting, application developers can expect their systems' performance to improve every year.

2. Architectural innovation has been pursued on many fronts, ranging from changes to the instruction set of a processor to be image processing-friendly, to producing totally dedicated ICs for specific image processing problems.

This paper reviews a number of ways in which architectural innovation is attempting to meet the

Computational demands of high performance image processing. The range of relevant architecture and technology approaches which we select are:

- Standard microprocessor technology;
- DSP processors;
- Parallel processing;
- Dynamically reprogrammable hardware.

2. HISTORY OF ARCHITECTURES

2.1 Developments in Microprocessor Technology

Steady improvement in standard microprocessor technology over many years has resulted in a doubling of performance every 18 months (known as Moore's Law). Before rushing into the design of special purpose architectures as in fig.1, care should be taken to make sure the exercise is actually required. If the required speed up is in the region of a factor of two or three, say, then it may be more cost effective merely to wait for future generations of the same processor.

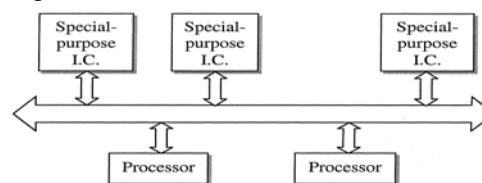


Fig. 1. Bus-based architecture with special purpose ICs.

2.2 DSP processors

Signal processing applications, by their very definition, process real world signals which are generated in real time. Traditionally, much signal processing work has operated on one-dimensional signals, such as speech or audio. To obtain real time performance for these applications, processors with architectures and instruction set specially tailored to signal processing began to emerge [5]. Typical features included Multiply Accumulate instructions, special control logic and instructions for tight loops, pipelining of arithmetic units and memory accessing, and Harvard architecture (with separate data and program memory spaces). More recent designs (such as some in the Texas range of DSP processors) have features explicitly for (two-dimensional) image processing, particularly with image compression in mind.

When carefully programmed to exploit the special architectural features, these processors can yield very impressive performance rates. However, there is a cost. The programming model at the machine level is much more complex than for traditional microprocessors. Highly optimising compilers are needed if the processor's potential is to be realised with a high level language; but such compilers are difficult to construct.

When DSP processors first appeared on the scene, they lacked many of the normal control features of standard microprocessors. However, as software development time became recognised as an important aspect in commercial product development, DSP processors have gradually become more and more like standard microprocessors. A second trend which can be observed is that microprocessors have begun to assimilate some of the design features of DSP processors (such as pipelining and, with MMX, movement towards application-specific instructions).

What we are now seeing is thus a process of gradual convergence of microprocessor and DSP technology. As die sizes reduce, thus freeing up chip area for new functionality, we can expect to see new, general purpose processors with hardware to accelerate common tasks such as image and video coding.

2.3 Parallel processing

For several years, research into parallel processing has been partly motivated by the belief that sequential processors must eventually reach their performance limit, at which point significant speed improvements will only be obtained by adding more processors. For this reason, parallel processing techniques have been widely studied for image processing.

Two forms of parallel architecture have generally been applied to image processing: these have been named in Flynn's classification as:

- SIMD (Single Instruction Multiple Data), and

- MIMD (Multiple Instruction Multiple Data)

SIMD processors comprise a closely coupled array of processing elements which obey the same instruction stream in lock-step parallelism. Mesh connected architectures come into this category. In the context of image processing, one rather successful SIMD machine was the ICL DAP, originally developed in the 1970's, and bought over by several companies since then. The DAP yielded very high performance figures, and a large image processing software base has been developed over the years. Its main drawback was its cost, which put it beyond the budget of most high volume applications, and limited its commercial success. It has, however, been used successfully in a number of military applications.

MIMD parallel processing systems comprise a number of loosely coupled, independent processors

which communicate with each other via some form of interconnection network. To avoid the type of communication bottlenecks which occur in a bus-based system, the normal arrangement is to have a distributed memory space, with message passing communication for data sharing. For image processing, the obvious approach to parallelising a program is to distribute an image in segments across the set of processors. With this approach, and with efficient interprocessor communication, good speed ups have been reported (for instance, a factor of 15 on 16 processors).

Remembering Moore's Law, this same speed up could nearly have been achieved in the same timescale merely by waiting for a few years, and doing nothing!

The increasing power of microprocessors and DSP processors has resulted in a number of applications at the lower end of the processing power spectrum being mopped up by these low cost technologies. More recently, other real time, data intensive application areas (including image processing) have become the targets of an emerging technology dynamically reprogrammable hardware.

2.4 Dynamically reprogrammable hardware

Pressure for hardware development to become more rapid and more flexible has resulted in the emergence of various technologies which support programmable hardware. These vary in their reusability and speed of programming. Programmable Logic Devices (PLDs) are not normally reprogrammable at all, since they involve blowing built-in fuses to define their functionality.

EPROMS can be reprogrammed, but it can take several seconds, or even minutes, to carry out the reprogramming. The most promising technology from our standpoint is dynamically reprogrammable Field Programmable Gate Array (FPGA) technology. The power, speed and flexibility of these devices has improved considerably in recent years.

Reprogramming time in some cases is measured in milliseconds or even microseconds.

This has opened up the subject of custom computing using FPGA chips to implement an application-specific ALU (or coprocessor). Image processing is one application area which is proving amenable to acceleration using FPGAs. The usual arrangement for using FPGAs to accelerate image processing operations is to embed the FPGA part in an attached processor (or coprocessor) which has its own local RAM memory and some simple control logic. This attached processor is normally controlled from a host workstation.

Development of low cost, high performance cards is currently a very active area, so anything which is written will almost instantly be out of date. It is possible to design and implement hardware architectures for a range of standard image processing neighbourhood operations. For instance, the outline architecture for one possible form of 3x3 image convolver is as shown in Fig. 3.

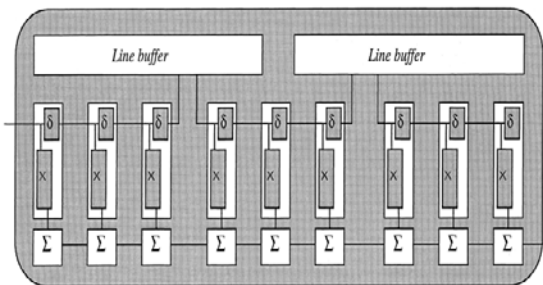


Fig. 3. Outline architecture of a 3x3 convolver

The main advantage of FPGA-based processors is that they can offer near supercomputer performance at relatively low cost. While their performance does not yet match that of special purpose VLSI or ASIC designs, they offer increasingly competitive performance with the huge benefit of dynamic reprogrammability and software control. Another advantage claimed by FPGA designers is that each improvement in VLSI technology has a twofold benefit for FPGAs: not only does the clock speed increase, but the number of cells (and hence the functionality) of the chip increases too. With microprocessors, the argument goes, usually only the clock speed increases. This reasoning is only partly valid.

Possibly the biggest problem for FPGA technology in accelerating image processing applications (or any other application for that matter), is the extremely low level programming model which it supports. Normally FPGAs are 'programmed' in a hardware description language such as VHDL, which is hardware oriented rather than algorithm oriented. It is likely-indeed, essential-that this problem will become a major area of research in the future. One algorithm oriented approach is based on a library of tailorable

FPGA configurations, one for each high level neighbourhood operator of Image Algebra. Developments of this library based approach offer the advantage of providing a programmer-friendly interface which hides the intricacies of the FPGA hardware.

3. CASE STUDY

Towards FPGA based image processing, a distance transform named Euclidean Distance Transform (EDT) was tried in our VLSI Design Lab.

A distance transformation converts a binary image consisting of foreground and background pixels into one where each pixel has a value equal to its distance to the nearest background pixel (alternatively, distances could be to the nearest foreground pixel). Applications of distance transform are numerous. These include shape analysis of objects, machine vision and image matching [1]. A pipelined array architecture is presented for the computation of EDT as in fig 6. The architecture comprises of a two-dimensional array of locally

	1	2	3	4	5	6
1	(2,2) 4	(2,1) 7	(2,0) 8	(2,0) 8	(2,1) 7	(2,2) 4
2	(1,2) 7	(1,1) 10	(1,0) 11	(1,0) 11	(1,1) 10	(1,2) 7
3	(0,2) 8	(0,1) 11	(0,0) 12	(0,0) 12	(0,1) 11	(0,2) 8
4	(1,2) 7	(1,1) 10	(1,0) 11	(1,0) 11	(1,1) 10	(1,2) 7
5	(2,2) 4	(2,1) 7	(2,0) 8	(2,0) 8	(2,1) 7	(2,2) 4

At $k = 3$

Fig.4 Iterative computation of $(\Delta r, \Delta c)$ and δ of pixels.

interconnected identical processing elements where each element is a sequential logic and all elements are operated synchronously. The computations within each element are with integers. Such a digital design with no floating point arithmetic is quite suitable for FPGA implementation.

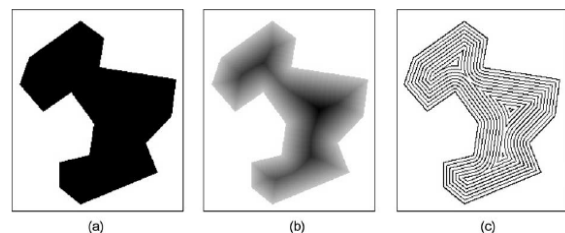


Fig 5. Simulation results. (a) Test image. (b) and (c) Intensity plot and Contour plot of integer approximation to Euclidean distance values

3.1 EDT Algorithm

Inputs: binary image
 Outputs: $(\Delta r, \Delta c)$ of pixels
 Initialization: $\Delta r = \Delta c = \delta = 0$ & $done = 1$ for background pixels
 Iterative process:
 repeat for $k=1,2,..$
 for all pixels p do in parallel
 Step 1: Compute $\Delta r_i, \Delta c_i$ and $\delta_i = 1$ to 8
 Step 2: Consider δ_i values corresponding to neighbors p_i whose $done(p_i)=1$. Find the maximum, say δ_m .
 Step 3: If $done(p)=0$ and $\delta_m > 0$, then $\Delta r(p) = \Delta r_m, \Delta c(p) = \Delta c_m, \delta(p) = \delta_m$ and $done(p)=1$.
 If $done(p)=1$, then $\delta(p) = \delta(p) + 2k$.
 end for
 until $done=1$ for all pixels

4. RESULTS

Some results and comments for the five-stage processing element designed for image size 512x512 are as follows.

For this case, n is 512, nc is 8 and tc is obtained from

implementation as 27.02 ns. Hence, $TEDT < 0.156$ ms and $NI > 6390$. NI is much greater than the video rate, which is about 30 frames per second and hence

Table 1
 Results of implementation of different pipelined stages of the center processing element in a 3x3 array of elements

Stage	1	2	3	4	5
Signal propagation delay (ns)	27.02	12.143	12.143	12.143	13.908

Equivalent gate count for processing element: 929.

the computation of EDT on an array-type hardware architecture is quite suitable for real-time applications.

In order to know the amount of reduction in hardware space through pipelining, the non-pipelined version of processing element has also been implemented on the same Xilinx device. The equivalent gate count in this case is 383. For the entire array architecture designed for a 512x512 image, the non-pipelined one consumes 383x5122 gates while the pipelined one consumes 929x5122 /4 gates. This amounts to about 40% reduction in hardware area.

5. CONCLUSIONS

Reconfigurable architecture seems to be a better choice for basic image processing. The design of a pipelined architecture for EDT is presented.

Results of FPGA implementation of the architecture indicate that more than 6000 images of size 512x512 can be processed in a second. This is much greater than the video rate. Further, about 40% reduction in gate count is achieved through pipelining. The ideas presented for the case of 4 pixels per processing element readily extend to the case of more than 4 pixels per processing element (such as 9, 16 and so on).

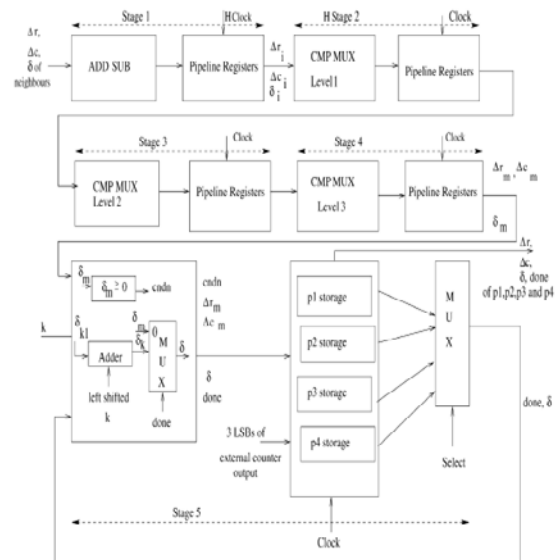


Fig. 6. Pipelined processing element

6. ACKNOWLEDGEMENT

The authors wish to thank Lovely Professional University to use CADENCE EDA tools at VLSI DESIGN Laboratories.

7. REFERENCES

- [1] Davies. E.R “Machine Vision” Morgan Kaufman Publishers, 3rd Edition.
- [2] Crookes. D “Architectures for High Performance Image Processing-the future” Journal of System Arch. 45 (1999) pp. 739-748).
- [3] Ghazawi. T.E. et.al. “The promise of High Performance Reconfigurable Computing” Research Feature published by IEEE Computer Society 2008.
- [4] Pan. Y, Hamdi. M and Li Keqin “Euclidean Distance Transform for Binary Images on Reconfigurable Mesh-Connected Computers” IEEE Transactions on Systems, Man and Cybernetics, Vol.30(1), February 2000.
- [5] S. Pavel, S.G. Akl, Efficient algorithms for the Euclidean distance transform, Parallel Processing Letters 5 (1995) 205–212.