A Low-Cost Architecture for Real-Time Processing and Analysis of Binary Images

F.M. Hugen, B. Bultink

Control systems and computer engineering laboratory
Faculty of electrical engineering, University of Twente
P.O. Box 217, 7500 AE Enschede, the Netherlands

Abstract

In this paper an architecture is presented for low-cost and flexible realisation of image processing and analysis algorithms of binary images. The flexibility of the architecture is due to reconfigurable network of simple boolean functions. It is shown that this architecture can readily be implemented by low-cost off-the-shelf components. This is illustrated by some simulations of a hypothetical realisation.

1 Introduction

In industrial applications, binary image processing and analysis is widely used for example in inspection and pick and place problems. Binary image processing and analysis is far less complicated in terms of algorithm complexity and required computational burden compared to grey level image processing and analysis.

However the performance required by more complicated applications of binary image processing and analysis, still demand for high-throughput hardware architectures. Common solutions are processor arrays and dedicated hardware. Processor array solutions as the CLIP4, MPP-2 [1], [2] and more recently, ASP [3] are not suited for implementing a broad class of algorithms efficiently. Dedicated hardware on the contrary couples high-throughput with small silicon area, but lack the flexibility of processor arrays. Examples of dedicated hardware in this context are binary correlators and run-length processors [4]. In this paper we present an architecture that 1) is capable of efficient implementing real-time binary image processing and analysis algorithms, and 2) can be economically realised with low-cost off-the-shelf components.

The organisation of this paper is as follows: In paragraph 2, a short overview of a typical application of binary image processing and analysis is presented while in paragraph 3 the basic architecture is discussed. A more detailed architecture tailored to binary image processing and analysis is presented in paragraph 4 followed by a proposed realisation based upon low-cost off-the-shelf components in paragraph 5. Conclusions are drawn in the last paragraph.

2 Binary image processing

In this paragraph, a brief overview of a typical application of binary image processing and analysis is discussed. In figure 2.1 the chain from imaging process to desired result is sketched. A process is sensed by one or more cameras. The resulting image is processed such that a bi-level or binary image is obtained. After this step, the image is enhanced if necessary. Enhancement of the image can be accomplished by a sequence of local morphology operators (dilation, erosion, outlier pixel removal). After this enhancement, the image is segmented in regions that are supposed to contain the desired information. In most cases such segmentation is based upon isolation of non-connected components (component labeling) but contours (binary edge detection) can be used as well. The segments are used for the analysis of the image. It consists of parameter extraction of segments such as position, orientation (generalised moments) and shape analysis by template matching (binary correlation). With these parameters, it is easy for a normal computer to recognise objects and compute its parameters as position, orientation and shape in real-time.

3 Reconfigurable architecture

Hardware architectures should have the following properties for implementing real-time binary image processing and analysis in applications:

figure 2.1 Binary image processing and analysis chain

- Flexibility. The same hardware should be able to implement many different algorithms.
- High-throughput.
- Scalability. Tailoring performance to a typical application.
- Fault tolerant if necessary.
- Economical realisation: Short time to market, low cost components.

It is obvious that dedicated hardware does not meet the demand of flexibility. Also scalability can be poor due to a fixed size of an implemented operator. Another problem is that each new application that differs in functionality, could require a new realisation, which is very costly for small series.

On the other hand, processor arrays need a lot hardware. Since the datapath of a processor for binary image processing is relatively small compared to the control path, MIMD processor arrays are a waste of silicon area. SIMD processor arrays have a too rigorous topology and a lack of individual processor functionality for implementing algorithms such as connected component labeling. Beside these problems, the control path of the processor arrays can limit the maximum operating frequency of the datapath.

An architecture template that meets the first four properties, is a reconfigurable architecture with generic operators. The concept of a reconfigurable architecture is not new and is used in for example WSI technology [5], [6]. In a reconfigurable architecture, the hardware is divided in some sets each containing many cells with the same functionality. By connecting the cells with a switch network, the functionality of the architecture can be changed. This reconfiguration can be carried out at start up (static reconfiguration), or at run time (dynamic reconfiguration). If the switch network is a multistage switch network, it provides both local and global programmable connections between cells.

figure 3.1 Reconfigurable architecture

Important is the concept of generic operators. This means that the cells of the architecture have a functionality that is not tailored to specific (details of) algorithms. Examples of generic operators are a programmable boolean function and a Cordic ALU. Using generic operators, it is possible to implement algorithms by combining operators through the network in larger and complex structures. The reconfigurable architecture template is depicted in figure 3.1.

Flexibility of a reconfigurable architecture is obtained by (static) reconfigurability, high-throughput by a pipelined and parallel datapath and scalability by the possibility of adding more generic operators. Fault-tolerance can be achieved by triplicating some functional structures in conjunction with majority voting at any level. Even defect tolerance can be achieved by excluding defective cells.

Dynamic reconfiguration can be used in cases where the processing throughput is much higher than the pixel rate. Algorithms can then be subsequently executed, time-sharing hardware between two input-frames.

4 An architecture based upon reconfiguration for binary image processing and analysis

The reconfigurable architecture of the previous paragraph is not sufficient to implement the processing part of the chain in figure 2.1. Beside processing hardware, also data storage and configuration control is necessary. Data storage is required if two algorithms to be executed subsequently, do not match with in and output sequence of their operands, or are time-shared. Also, some algorithms require storage of local neighbourhood data or other data.

Therefore datastorage is specified in our architecture by databuffers. These databuffers are enhanced by address generators which can be tailored to the accesses required for the execution of algorithms, an approach that is also used in MoM [7].

figure 4.1 Architecture overview

When the system is initialised, the configuration data must be downloaded. If dynamic reconfiguration is used, the conditions for reconfiguring the network locally, must be signalled followed by downloading of the reconfiguration data.

The control of the reconfigurable datapath is implemented by the event manager. When an event is generated by the datapath (i.e. an algorithm is been processed) the event manager determines according to the event type and the reconfiguration table which configuration data has to be downloaded.

5 Realisation of the reconfigurable architecture

An architecture is only successful in applications if it meets the demand of economical realisation. Therefore an example of a realisation of the presented architecture is considered. As a starting point the use of commodities (off-the-shelf low-cost components) is taken. Realisation of the datapath is possible with programmable logic devices. The most promising programmable logic devices are FPGA's (Field Programmable Gate Arrays) such as the Xilinx 3000 and 4000 series and Plessey ERA series [8], [9]. The cells of these FPGA's are quite generic and the network topology seems to be flexible enough.

figure 5.1 Realisation of a static reconfigurable architecture

The availability of design tools for the Xilinx 3000 series to us was the reason to choose this device. The data storage in the datapath are realised with serial access-VRAMS. One or two devices are enough to store 8 512x512 images. Some line buffers for morphology and binary correlators are realised by FIFOs. Xilinx 4000 and Plessey ERA devices contain enough memory-cells to implement these line buffers internally. The address generators can also be implemented in the FPGA's.

Unfortunately, the time needed to reconfigure Xilinx devices is too slow for dynamic reconfiguration in real-

algorithm	area(% of device)	throughput [10 ⁶ pixels/sec]
boolean (and, or, not, exor)	0.6	50
dilate, erode, outlier pixel removal	2.8 / 3.4	50
run length coding	9.0	25
area	6.9	50
binary correlation (11 * 11) template	85	10

table 5.1 Results

time. However it seems to be possible to realise dynamic reconfiguration since Plessey ERA devices are capable of very fast reconfiguration during run-time [9].

Some algorithms are mapped on the implemented architecture and simulated. The number of used cells (as a percentage of the available cells) and throughput (pixels/sec.) are calculated for each algorithm. The obtained results are listed in table 5.1.

6 Conclusions

In this paper a reconfigurable hardware architecture is presented that is suited for real-time binary image processing and analysis. Demands as flexibility, scalability high-throughput can be met. Demonstrated is the possibility to realise such an architecture economically with low-cost, off-the-shelf components with static reconfiguration. Further it is expected that this architecture offers dynamic reconfiguration by choosing the right components in a realisation.

Currently we are investigating how a toolbox, for implementing algorithms on this architecture, should be constructed.

References

- [1] V. Cantoni, S. Levialdi, "Multiprocessor computing for images", Proc. IEEE, vol. 76, No.8, pp.959-968, August 1988.
- [2] M. Maresca, M. A. Lavin, "Parallel architectures for vision", Proc. IEEE, vol. 76, No.8, pp.970-981, August 1988.
- [3] R.M. Lea, "ASP: a parallel computing technology", Proc. SPIE Visual Communications and Image Processing 1990, pp. 78-92, Lausanne 1990.
- [4] Logic devices Inc, "Fast CMOS data Book", July 1990
- [5] M. J. Shute, "Fifth generation wafer architecture", Prentice Hall, 1988
- [6] I. Koren, A. D. Singh, "Fault Tolerance in VLSI Circuits", IEEE Computer, pp. 73-83, July 1990.
- [7] R.W. Hartenstein, M. Riedmuller, K. smidt, M. Weber, "A novel ASIC design approach based on a new machine paradigm", IEEE J. on Solid state circuits, vol. 26 No.7 pp.975-989, July 1991.
- [8] Xilinx Inc. "PGA Data Book", 1988.
- [9] Plessey Semiconductor, "ERA product overview", 1989.