A Run-Time Reconfigurable Datapath Architecture for Image Processing Applications

Marcos R. Boschetti, Ivan S. Silva*, Sergio Bampi
Informatics Institute, Federal University of Rio Grande do Sul, Porto Alegre, Brazil
*Informatics and Applied Mathematics Department, Federal University of Rio Grande do Norte, Natal, Brazil
{marcosrb,bampi}@inf.ufrgs.br
ivan@dimap.ufrn.br

Abstract

This paper describes a run-time reconfigurable architecture targeted to flexible low-level image processing functions. The purpose is to present the evolution of the DRIP (Dynamically Reconfigurable Image Processor) architecture from a statically configurable datapath design to a dynamically reconfigurable approach. The methodology used to redefine the datapath basic building blocks and the hardware units developed to provide an efficient and flexible image processing system are also discussed. An important issue is the granularity of the basic processing elements of the datapath, in view of the combination of programmable function by hardware control - the classical datapath paradigm - and the dynamic reconfiguration. DRIP can perform a large set of digital image processing algorithms with real-time performance to fulfill the requirements of contemporary complex applications.

1. Introduction

In recent years the reconfigurable computing machines introduced a new set of alternatives for hardware and systems designers. Fueled by the increasing FPGA densities and the combination of software targeted tasks with run-time reconfiguration of the system, the configurable SoC (system-on-chip) [1] offers great room for system architecture innovations. Due to the ability to customize hardware modules, it is possible to optimize control, datapath and interconnections according to specific algorithm requirements. Run-Time Reconfigurable (RTR) systems in particular can achieve outstanding benefits from this paradigm, adapting to the instantaneous needs of an application.

Reconfigurable architectures bring the possibility to combine the post-fabrication programmability of processors with the performance, area savings and power consumption of specific circuits (ASICs, ASIPs – application specific instruction-set processors) [2]. As a mid-range solution between generic (software) and specific (hardware) approaches, reconfigurable architectures are a valuable tool for design space exploration.

Contemporary multimedia and telecommunication applications bring new challenges to designers. Besides the high performance demanded by complex algorithms, two increasingly important design aspects must be considered: power consumption per function and hardware flexibility to reconfigure for specific tasks. The success of many embedded systems is directly related to a low-power design that maximizes the device autonomy. In [3,4] reconfigurable computing is presented as a mechanism to build energy-efficient architectures. The flexibility of reconfigurable systems is also a key factor. It allows the modification of deployed products through the addition of new services or the redefinition of algorithms functionality resulting in important commercial advantages that are not found in ASICs. Multimedia operations include algorithms that require heavy real-time processing. Image analysis and machine vision solutions are important in many industrial applications such as robotics, security, medical imaging, and scene inspection. These applications have regular characteristics and an inherent level of parallelism that can be exploited by reconfigurable systems. Complete infrastructure for relocating tasks in hardware in a run-time environment for a complete SoC is presented in [5], specifically targeting image processing applications as a first demonstrator.

In this paper we present a new implementation of the reconfigurable neighborhood processor DRIP [6]. The DRIP structure is inspired in the functional programming

1530-1591/04 $20.00 (c) 2004 IEEE

(FP) paradigm [7]. A great advantage of this approach is the ability to express complex problems with simple basic functions, allowing the efficient implementation of a large number of image processing algorithms. In the FP style a program can be naturally expressed by an equivalent data flow graph [8].

This work is organized as follows. Section 2 introduces the architecture, presenting the general structure and the processing elements, discusses enhancements made in the datapath processing elements and the dynamic reconfiguration mechanism. Section 3 presents performance results obtained with the reconfigurable datapath for important image processing algorithms. Conclusions are drawn in Section 4.

2. Architecture

A neighborhood processor is similar to an array processor [9]. It processes an input image, generating a new image, where each output pixel is a function of its correspondent in the input image and the nearest neighbors. Using a standard neighborhood (e.g.: 3x3, 5x5 or 7x7 pixels), it scans the image line by line. The general DRIP architecture (a block diagram representation is shown in figure 1) can be adapted to any of these neighborhood ranges. The implementation discussed here uses 3x3 pixel windows.

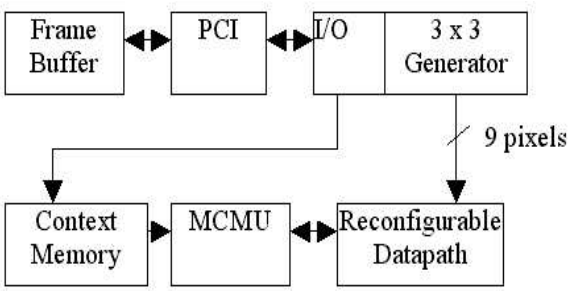

Figure 1. General architecture.

The I/O processor is responsible for the communication with the frame buffer. It comprises the neighborhood generator that provides 3x3 pixel windows to the datapath. In our experiments the PCI interface and I/O processor are the speed bottleneck. The I/O processor additionally receives new configurations to be stored in the context memory area. Each configuration for the datapath is specified by a very compact (405 bits only) representation of the whole datapath configuration bits.

2.1 Reconfigurable Datapath

The datapath is composed of a bidimensional (9x9) matrix of processing elements (PEs). The structure of the pipeline follows a data flow graph represented by a class of non-linear filters widely used in digital image processing. [8]. The hardware implementation of these filters is based on a parallel sorting network, more specifically on the odd-even transposition sort algorithm [10], which achieves a good trade-off between parallelism, regularity and execution time. Figure 2 presents a high level view of the 9-stage pipelined datapath.

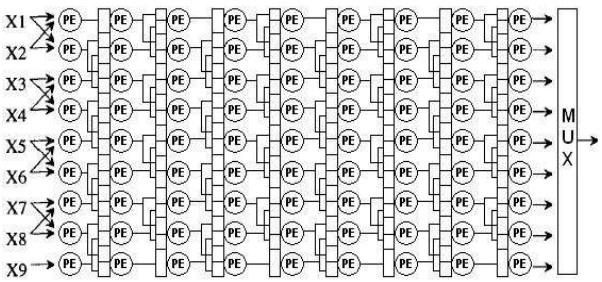

Figure 2. Reconfigurable datapath.

2.2 Processor Elements

The processor element (Figure 3) is the basic programming block of the processor. In spite of being a simple cell, its connection in the DRIP architecture provides all the flexibility needed to implement a wide number of digital image processing algorithms. The processor elements execute only two basic operations: MAX, representing the class of non-linear operations and ADD representing the class of linear algorithms. Each PE receives two input pixels X1 and X2, which are 9-bit wide in our experiments. To increase PE's logical capabilities an integer weight (-1, 0 or 1) is associated to each input.

The simplified view and the corresponding hardware structure of the PE are shown in Figure 3. The 2C block represents the circuit for negative numbers generation. As an example, suppose that for some PE the MAX function is programmed and the multiplexors have, respectively, the first and second inputs selected, in this case the output S will correspond to MAX(-X1,X2).

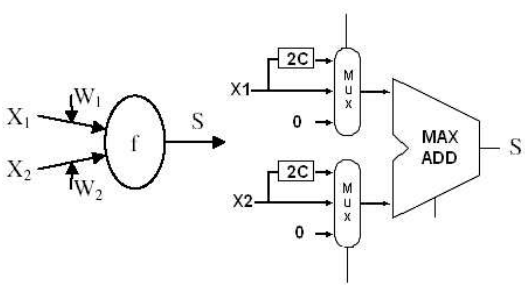

Figure 3. Processor element.

Datapath (re)configuration consists in the customization of the PE network. According to the mentioned parameters (2 possible functions, 2 inputs and 3 possible integer weights) we are allowed to apply 18 different configurations to a single PE. However, many of them are symmetrical, for example MAX(X1*1,X2*0) is the same as MAX(X1*0,X2*1). This property defines a set of 11 really unique functions, which, in turn, are capable to support that large class of image filtering applications. These 11 functions are shown in Table 1.

Table 1. Configurations.

Configuration	Function
Add(0,0); Max(0,0)	0
Add(0,X_2); Add(X_1,0)	$X_{1(2)}$
Add(-X_1,0); Add(0,-X_2)	-$X_{1(2)}$
Add(X_1,X_2)	addition
Add(-X_1,X_2); Add(X_1,-X_2)	subtraction
Add(-X_1,-X_2)	-X_1 – X_2
Max(0,X_2); Max(X_1,0)	If $X_{1(2)} > 0$ then $X_{1(2)}$ else 0
Max(0,-X_2); Max(-X_1,0)	If $X_{2(1)} < 0$ then $X_{2(1)}$ else 0
Max(X_1,X_2)	Max(X_1,X_2)
Max(-X_1,X_2);Max(X_1,-X_2)	If $X_{1(2)} > X_{2(1)}$ then $X_{1(2)}$
Max(-X_1,-X_2)	-Min(X_1,X_2)

2.3 Processor Elements Granularity

The term granularity, in this work, is being used to define the complexity of a cell. Thus, an increase in the granularity of a PE means that it has more hardware resources than before, however, it still processes 9-bit pixels.

The DRIP architecture evolved from a straightforward pipelined neighborhood processor NP9 [11] with an array of 9x9 processor elements arranged in a 9-stage pipeline, towards a fully reconfigurable datapath. In the previous DRIP version [6] the PE was synthesized for very specific image processing algorithms – hence a low granularity, super-specialized cell to the bit-level processing. Each PE was optimized and mapped to only one of the functions of Table 1 and could not be reprogrammed (no programmable hardware structures such as the multiplexors in figure 3 were used in low-grain PEs). The reconfiguration strategy demanded that the image processing task ended (for a complete image buffer, for instance), in order to fully reconfigure the entire processor datapath elements, down to the bit-level. The loading of the new configuration introduces considerable reconfiguration overhead, in particular in view of the limited support present in the FPGAs for partial reconfiguration. Our experiments with the FLEX10k Altera device have shown that one spends considerable time (hundreds of milliseconds) and energy to fully reconfigure the entire device. By direct measurements on pin-currents, we noticed a large circuit activity, hence power dissipation during reconfiguration.

A higher granularity level was added with the inclusion of extra hardware, increasing both the PE grain size and the program-control support for RTR. Besides conventional datapath control by programming, larger grain size can bring other benefits such as less routing complexity [12] for the reconfigurable datapath.

Nevertheless, our approach is still radically distinct from other much more coarse-grained architectures. For instance, Morphosys [16] system-on-chip, a SIMD multiprocessor architecture composed by 64 reconfigurable cells, also targeting image and DSP applications.

2.4 Environment for Algorithm Prototyping

Considering successive algorithms mapped on the datapath it seems natural that the less modifications needed from one configuration to another the less reconfiguration overhead will be produced. Besides, all possible algorithms are composed of only 11 different functions, which results in a considerable PE reuse.

The RRANN (Run-time Reconfigurable Array Neural Network) [13] project uses partially reconfigurable FPGAs and great part of the design effort is concentrated in reducing the amount of hardware to be reconfigured. In this context, three different types of blocks can be defined: static, dynamic and semi-static. Static blocks remain unchanged between configurations. Dynamic cells are the ones that change completely or almost completely, while semi-static cells present very few structural differences.

Analyzing the different digital image processing algorithms mapped on DRIP datapath it is readily seen that a significant similarity level exists, which reflects in a huge number of static and semi-static PEs. Therefore, we can map only the really needed elements to the reconfigurable fabric reducing resources usage significantly and, at the same time, minimizing the logic depth of the cells that leads to higher operational frequencies.

We developed a specific CAD tool [14] for targeting different image processing algorithms into the DRIP datapath. The tool performs, at pre-synthesis time, a similarity analysis through a topological comparison between target algorithms. As a result an optimized VHDL model for the application is generated. Hence, the synthesized design is optimized for a large range of image processing algorithms. Figure 4 shows, as an example, a PE that supports MAX(X1,X2) and MAX(X1,0). In this case, the CAD tool identified that the multiplexor of the first

input and 2C negative number generators were not needed, besides the second multiplexor could be a 2:1 multiplexor.

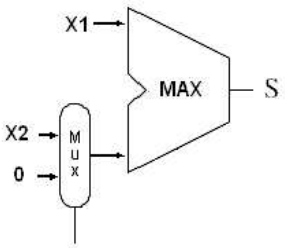

Figure 4. Custom processor element.

2.5 I/O Processor and Reconfiguration Control

The I/O processor communicates with the frame buffer through a PCI bus. It is responsible for providing the pixels to the datapath and to send new configurations to the multicontext memory area. It also comprises the neighborhood generator with its data buffer. The data buffer stores the pixels needed for the algorithm processing. We identified that 2n+3 pixels is the maximum buffer size necessary, where n is the image line width. The 3x3 neighborhood generator is a FIFO-like structure as can be seen in figure 5.

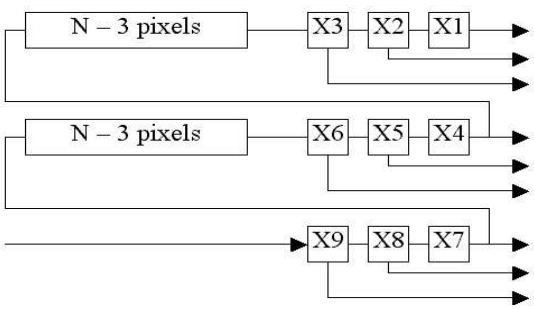

Figure 5. Neighborhood generator.

The multicontext control management unit (MCMU) is responsible for the reconfiguration process. It communicates with the context buffers in the context memory area, where complete datapath configurations are stored. The context memory area comprises 16 configuration buffers. An entire configuration for the datapath requires only 405 bits that are divided in 9 slices of 45 bits. Each slice contains the configuration of an individual column, which is a DRIP pipeline stage.

The replacement of a configuration can be performed in parallel or in a pipelined fashion. In parallel mode the configuration data is transferred from a context buffer to the datapath in one clock cycle. In pipelined mode, the columns from 1 to 9 are reconfigured one at each clock cycle. The MCMU is able to manage new configurations received from the I/O processor. This allows the inclusion of new algorithms extending the processor functionalities. In this situation the context buffers can be seen as a background configuration plane [15] that can store new algorithms while the current active program is still running.

3. Results

As explained previously, DRIP can perform a large number of image processing operations. Some examples include: linear (convolution), non-linear and hybrid filters, binary and gray-level morphological operations (dilation, erosion, thinning and thickening algorithms, morphological edge detectors, gradient), binary and gray-level geodesic operations, etc. Table 2 shows the maximum datapath frequency achieved for five of these algorithms using two different Altera FPGA devices. Apex 20K is roughly twice as fast as Flex10K for these circuits.

Table 2. DRIP datapath performance for FLEX10k and APEX20k devices.

Algorithm	FLEX10k (MHz)	APEX20k (MHz)
Median Filter	32.89	77.05
Morph. Edge Detector	48.78	110.45
Erosion	39.84	78.74
Dilation	44.12	84.18
Separable Median Filter	46.30	105.9

These numbers assure real-time performance even for significant image resolutions. The parallelism and regularity of the datapath and the characteristics of image processing applications make DRIP well suited for low-level multimedia tasks.

Figure 6 shows the frame throughput (in frames/sec) for 3 different situations. The morphological edge detector and the median filter represent, respectively, the fastest and the slowest configurations when considering only one algorithm mapped to the datapath at a given time, as in the previous DRIP approach. The inferior line in figure 6 (triangle style line) represents DRIP-RTR, in a configuration that supports faster reconfiguration, in order to run up to 5 different image processing algorithms. All implementations have considered the DRIP hardware processing in VGA (640 x 480), SVGA (800 x 600 and 1024 x 768), and SXVGA (2048 x 1536) image resolutions.

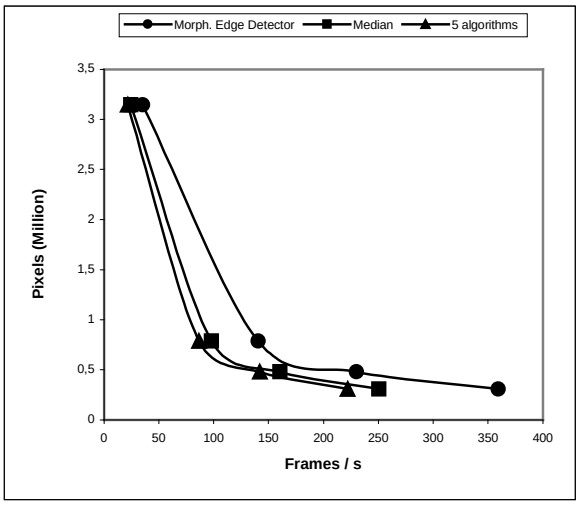

Fig. 6. Pixels vs Frames/s for some image resolutions.

As said before, the values in table 2 were achieved considering a datapath implementation that can support only the synthesized algorithm, it is a maximum performance table. Due to the overheads naturally added when allowing run-time reconfiguration, a datapath implementation where it's possible to perform RTR reconfiguration between the five algorithms of table 2 needs 1830 logic cells (LCs) running at 68,2 MHz.

As can be seen in Figure 6, DRIP-RTR is slower than the datapath implementations that can support only one algorithm. However, thanks to the new approach, DRIP-RTR can be reconfigured in one clock cycle what is faster, in the order of millions of times, than the statically reconfigurable version.

Table 3 shows the number of Altera logic cells used in each version of the neighborhood processor configuration. NP9 is statically configured to be capable of performing all image processing algorithms. DRIP-RTR uses the medium granularity elements, and in this example it also supports the five algorithms of table 2. The values for the reconfigurable DRIP are based on the worst-case application.

Table 3. Logic cells usage.

(Re)configurable Processor	# Altera LCs
NP9	6526 LCs Flex10K100 + Flex10K70 [6]
Reconfigurable DRIP	1113 LCs Flex10K30 [6]
DRIP - RTR	1830 LCs APEX20k

DRIP-RTR has a data flow oriented architecture with medium granularity, allowing reconfiguration for any bit-width operations. In all results shown in Tables 2 and 3, 9-bit gray scale pixels are processed. Hence, the comparisons are valid for this resolution; however the synthesis method can be easily extended for n-bits pixel scales.

4. Conclusions

Reconfigurable architectures are extending products life cycles through real time in-system debugging, field-maintenance and remote upgrades. This flexibility comes together with characteristics of special purpose designs such as high performance and low power consumption.

In this context, we presented the evolution of the DRIP image processor. From a statically reconfigurable design a new flexible run-time reconfigurable system was developed. Increasing the processor basic building block grain size, we added the extra hardware to support a new level of programmability. Our CAD tool is used to determine the right amount of hardware needed by each PE in the reconfigurable datapath.

DRIP-RTR achieves a significant performance with very little reconfiguration latency. The MCMU can efficiently handle the reconfiguration process and is also suited to support the extension to new algorithms. This can be done for data-flow oriented tasks that allow for highly parallel and highly reconfigurable datapaths.

The results showed that the datapath and control overheads introduced in the development of the run-time reconfigurable strategy is a price worth paying. There is a 11% difference in the performance in favor to the statically reconfigurable approach (datapath support for only one algorithm), however, due to the RTR version programmability and the on-chip context buffers, it can be reconfigured in one clock cycle, what is much faster (millions of times in DRIP) than reprogramming the entire FPGA. Considering the development of configurable systems-on-chip, DRIP-RTR can be an efficient core specialized in general image filtering applications.

Acknowledgments

The authors acknowledge the support from CAPES and CNPq, brazilian research agencies for R&D support.

References

[1] Becker J., "Configurable systems-on-chip". Integrated Circuits and Systems Design, 2002. Proceedings. 15th Symposium on , 2002 pp. 379 -384

[2] DeHon A., Wawrzynek J., "Reconfigurable Computing: What, Why, and Implications for Design Automation". Design

Automation Conference, 1999. Proceedings. 36th , 1999 Page(s): 610 –615

[3] Rabaey J., "Reconfigurable Processing: The Solution to Low-Power Programmable DSP". Proceedings ICASSP 1997, Munich, April 1997.

[4] David R., Chillet D., Pillement S., Sentieys O., "DART: A Dynamically Reconfigurable Architecture Dealing with Future Mobile Telecommunications Constraints". Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002,

[5] Mignolet, J-Y., Nollet V., Coene, P., Verkest D., Vernalde S. and Lauwereins R., "Infrastructure for Design and Management of Relocatable Tasks in a Heterogeneous Reconfigurable System-on-Chip". In: Design, Automation and Test in Europe, 2003, pp. 986-991.

[6] Adário A. M. S, Roehe E. L., Bampi S, "Dynamically Reconfigurable Architecture for Image Processor Applications". In: 36th Design Automation Conference, New Orleans, June 1999. Proceedings, 1999, pp 623-628.

[7] Backus J., "Can Programming Be Liberated from the Von Neumann Style? A Functional Style and Its Algebra of Programs". Comm. Of the ACM, vol 21, n. 8, Aug. 1978, pp.613-641.

[8] Leite N. J, Barros M. A.,"A Highly Reconfigurable Neighborhood Processor Based on Functional programming". In IEEE International Conference on Image Processing, Nov 1994, Proceedings… pp. 659-663

[9] Fountain, T.J. "Processor Arrays: Architecture and Applications". London: Academic Press, 1987.

[10] Knuth, D. E., "The Art of Computer Programming" , Reading, Massachusetts: Addison-Wesley, 1973

[11] Adário A. M. S., "A FPGA Implementation of a Neighborhood Processor for Image Processing Applications", Master Thesis, Unicamp, Brazil, 1997.

[12] Hartenstein H., "A Decade of Research on Reconfigurable Architectures - a Visionary Retrospective". Proc. International Conference on Design Automation and Testing in Europe 2001 (DATE 2001), Exhibit and Congress Center, Munich, Germany, March 2001"

[13] Eldredge, J., Hutchings B. L., "RRANN: The Run-Time Reconfigurable Artificial Neural Network ". In: Custom Integrated Circuits Conference, 1994, San Diego, California. Proceedings pp 77-80.

[14] Boschetti M. R., Adario A. M. S., Silva I. S., Bampi S., "Techniques and Mechanisms for Dynamic Reconfiguration in an Image Processor". Integrated Circuits and Systems Design Symposium, SBCCI2002. Proceedings. 15th Symposium on, 2002

[15] Salefski B., Caglar L.," Re-configurable computing in wireless", Design Automation Conference, 2001. Proceedings, 2001

[16] Singh H., Ming-Hau Lee, Guangming Lu, Kurdahi F.J. ,Bagherzadeh N., Chaves Filho E. M., "MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications". Computers, IEEE Transactions on, Volume: 49 Issue: 5, May 2000 pp 465-481