Available online at www.sciencedirect.com

ScienceDirect

Microprocessors and Microsystems 31 (2007) 546–563

MICROPROCESSORS AND MICROSYSTEMS

www.elsevier.com/locate/micpro

A reconfigurable computing framework for multi-scale cellular image processing

Reid Porter [a,*], Jan Frigo [a], Al Conti [a], Neal Harvey [b], Garrett Kenyon [c], Maya Gokhale [d]

[a] Space Data Systems Group, International, Space & Response Technologies Division, Los Alamos National Laboratory, Mail Stop D440, Los Alamos, NM 87545, United States

[b] Space and Remote Sensing Sciences Group, International, Space & Response Technologies Division, Los Alamos National Laboratory, United States

[c] Biophysics Group, Physics Division, Los Alamos National Laboratory, United States

[d] Advanced Computing Group, Computer and Computational Sciences, Los Alamos National Laboratory, United States

Abstract

Cellular computing architectures represent an important class of computation that are characterized by simple processing elements, local interconnect and massive parallelism. These architectures are a good match for many image and video processing applications and can be substantially accelerated with Reconfigurable Computers. We present a flexible software/hardware framework for design, implementation and automatic synthesis of cellular image processing algorithms. The system provides an extremely flexible set of parallel, pipelined and time-multiplexed components which can be tailored through reconfigurable hardware for particular applications. The most novel aspects of our framework include a highly pipelined architecture for multi-scale cellular image processing as well as support for several different pattern recognition applications. In this paper, we will describe the system in detail and present our performance assessments. The system achieved speed-up of at least 100× for computationally expensive sub-problems and 10× for end-to-end applications compared to software implementations.

Keywords: Field programmable gate array; Reconfigurable computing; Image processing; Cellular automata; Cellular nonlinear network

1. Introduction

Cellular computing architectures, like Cellular Automata, are characterized by simple processing elements, local interconnect, and massive parallelism [1]. These architectures can be significantly accelerated using hardware platforms which can exploit fine-grain parallelism. In the 1980s these platforms were custom chip or multi-processor-based systems [2,3]. In the 1990s these platforms were Reconfigurable Computers (RCC) [4,5]. RCC leverages commercial-off-the-shelf (COTS) devices and promises to provide increased design longevity compared to custom hardware solutions. Cellular automata (CA) are perhaps the simplest cellular architecture and are typically characterized by a regular array of Boolean processing elements with local interconnect. With increases in reconfigurable device capacity, several RCC-based frameworks for Cellular Automata implementation have been proposed [6–8].

Cellular architectures are an excellent match for many image and video processing applications [9]. For example, a two-dimensional array of linear processing elements with local interconnect implements a convolution. One of the first RCC implementations of convolution was on the Splash-2 [10]. Cellular Nonlinear Networks (CNN) [11] define a more general class of cellular architecture for image processing where the processing element contains both linear functions and nonlinear functions similar to a neural network. CNN have been implemented with custom analog devices [12], and with RCC [13]. The basic two-dimensional cellular architecture can be extended to a three-dimensional array, where

* Corresponding author. Tel.: +1 505 665 7508; fax: +1 505 665 4197. *E-mail address:* rporter@lanl.gov (R. Porter).

0141-9331/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2006.02.016

each layer (a two-dimensional array) interacts with other layers, to perform more complex tasks. This multi-layered architecture is a good match for image processing where complex algorithms are often decomposed into a sequence of primitive operations. Several RCC-based frameworks have been proposed for this type of architecture [14–17]. A multi-layered CNN for retina modeling was implemented with RCC by Nagy and Szolgay [18]. In previous work we implemented a generalized multi-layered convolution neural network for multi-spectral image classification [19].

To solve complex image processing tasks most traditional image processing algorithms depart from the cellular architecture paradigm. For example, many algorithms require multiple passes through an image and communication is not local, e.g., histogram techniques and connected component labeling. One way to extend a cellular architecture to solve more complex image processing tasks is to implement a multi-scale hierarchy. This is a multi-layered cellular architecture in which the number of processing elements is incrementally reduced from one layer to the next. Multi-scale hierarchies can provide many of the benefits of global communication, while maintaining many of the advantages of local, low latency, communication. The Neocognitron was one of the first multi-scale cellular architectures proposed for image processing and was inspired by biological models of cat retina [20]. Multi-scale cellular architectures are now an active research area and many researchers are working to apply them to increasingly complex visual tasks [21,22].

This paper describes a RCC-based framework for multi-scale cellular architectures called HARPO: Hierarchical Architectures for Rapid Processing of Objects. As far as we are aware, this framework provides a novel implementation of multi-scale architectures in which multiple scales are processed in parallel. Existing custom hardware [23] and RCC [24] multi-scale frameworks process different image scales in different execution passes. The image is sub-sampled, by modifying the memory address generator, and the image processing pipeline is applied to a reduced data volume. Since data volume typically reduces by a factor of four at each scale the total execution time with this approach is 1.33 times greater than for a single image pass. In our framework, we process multiple scales within a single processing pipeline. This means all scales can be processed in the time taken for a single image pass (with some increase in latency). Pipelining has been suggested for existing multi-scale implementations, but the performance improvement is negated by the fact that processing units executing at reduced scales are not operating at full capacity [23]. In an RCC-based system we can customize the implementation at run-time and obtain close to 100% utilization. Pipelined multi-scale processing is particularly useful for cellular image processing for two reasons:

1. Many cellular image processing solutions have strong local dependencies between different scales [25]. It is possible that without parallel execution of multiple-scales it will be difficult to exploit the parallelism within a single scale.
2. In most multi-scale cellular architectures the number of processing layers is increased as the scale is reduced [20]. This means that the total data volume that must be processed does not decrease and hence 100% utilization is essential.

In Section 2, we introduce the fundamental building block for cellular image processing algorithms: local neighborhood functions. We describe the most common operations, provide a brief overview of implementation strategies, and motivate the HARPO design choices. Due to the local communication constraints, algorithm development for cellular image processing is typically more difficult than for traditional image processing. One solution is to use unsupervised and supervised learning algorithms as part of algorithm development. In Section 3, we describe the HARPO system and describe how top-level software and hardware components interact as a practical image processing tool via supervised learning. The details of the RCC API and implementation are provided in Section 4. HARPO aims to automatically build top-level hardware pipelines in VHDL from high level specifications. This requires highly parameterized hardware modules with accurate timing and resource utilization estimators which are described in Section 5. The system performance is assessed for large multi-layered, multi-scale applications in Section 6. We conclude in Section 7 with a discussion of future work.

2. Background

Local neighborhood functions, or sliding window functions, are the fundamental building blocks for cellular image processing. These functions are applied at a particular pixel location and their output depends on a finite spatial, temporal, and spectral neighborhood. Typically the same function is applied to all pixel locations in parallel. Neighborhood functions include a large number of traditional image processing algorithms. Image functions such as spectral averaging, clipping, thresholding and pixel scaling can be considered a subclass of local neighborhood functions without spatial extent. Local neighborhood inputs can come from multiple input images such as color channels or spectral dimensions. Neighborhood functions can also receive multiple images in time (as in Fig. 1) e.g., for finite impulse response filters, the neighborhood window has a finite temporal extent and slides through time as the function is applied at each step. For infinite impulse response filters the neighborhood window includes a finite number of state variables. When a neighborhood function is applied at the edge of the image, some inputs will be undefined. In our system we temporarily increase