A mapping tool for configurable pipeline co-processors

Elias Barbudo, Eva Dokladalova, Thierry Grandpierre, Laurent George

A mapping tool for configurable pipeline co-processors

Elias Barbudo, Eva Dokladalova, Thierry Grandpierre, Laurent George
Université Paris-Est, ESIEE Paris
Laboratoire d'Informatique Gaspard Monge
UMR CNRS 8049
{name.surname}@esiee.fr

Abstract

The increasing real-time processing requirements have lead to the significant use of heterogeneous computing architectures. In this context, the time-critical tasks are frequently processed by a coarsely configurable pipelined FPGA-based hardware. Obviously, the manual application mapping on this architectures leads to a tedious work. In this paper, we resume the first results of our approach to the automated mapping of a real-life application on a data stream pipeline-based architecture.

1 Introduction

The need for real-time processing has lead to the significant use of heterogeneous computing architectures in order to face the increasing complexity of current applications and their implementations.

Pipeline-based systolic architectures, such as the Morphological Co-processing Unit (MCPU) [1] or the Programmable Pipeline Image processor (P^2IP) [5], are examples of a part of such complex systems designed specifically for applications where the latency is critical.

The manual application mapping in this architectures leads to a tedious work, considering the number of parameters one has to tune finely with a minimum knowledge, on the part of the end user, of the internal structure and the functions of every allocatable module.

A solution to this problem is through task modeling by a Directed Acyclic Graph (DAG). The DAG model is exploited in several mapping algorithms such as List Scheduling Heuristics[6], Clustering Heuristics [3] or Guided Random Search Techniques [7]. Most of the DAG mapping algorithms direct their efforts on a processor and thread level, trying to map and schedule the tasks on a processor or on an array of processors. Our approach aims to extend this work to the mapping of an application on data stream pipeline based hardware architectures equipped with coarsely programmable processing elements. Moreover, be suitable for hardware architectures such as a Convolutional Neural Network[2].

The rest of this paper is organized as follows. Section 2 defines the framework of the case study: the characteristics of the architecture under study and a real-life application example. Section 3 describes the principles of the proposed methodology and the steps required to obtain the final mapping. It includes the preliminary results summary. Section 4 recalls the study objectives and outlines shortly future work.

2 Problem statement

To illustrate the problem on a concrete example, we consider the MCPU as a candidate for the study of the mapping approach.

2.1 Morphological Co-processing Unit

The MCPU is a programmable FPGA-based system able to implement applications based on basic morphological operators such as erosion and dilation, and their deep concatenation and combinations. The architecture of the MCPU is depicted in Figure 1.

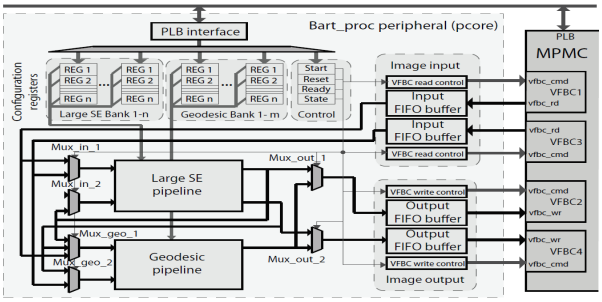

Figure 1: Architecture of the Morphological CO-Processor Unit

The system contains several processing pipelines. And the pipelines are composed by a concatenation of several stages. A stage consists of main processing elements called Mathematical Morphological Block (MMB), arithmetic logic units (ALU), data-path multiplexers (MUX), image properties measurement module (MEA) able to perform, i.e. the intensity integral computation.

The MCPU is controlled by two sets of parameters: i) hardware configuration parameters (image resolution, MMB pipeline depth, number of pipelines; ii) run-time parameters: MMBs operations, kernel size, shape, data-path inside MMBs.

The configuration of the system is done through an embedded server able to communicate with a distant client connected by an Ethernet Link. The client enables the blocks needed to implement his algorithm and select the type of operation that the MMB blocks will perform.

The objective of the proposed approach is to automatically map the application onto the MCPU. In this paper, we focus on the mapping problem on only the large SE[1] (kernel) pipeline (Fig. 2).

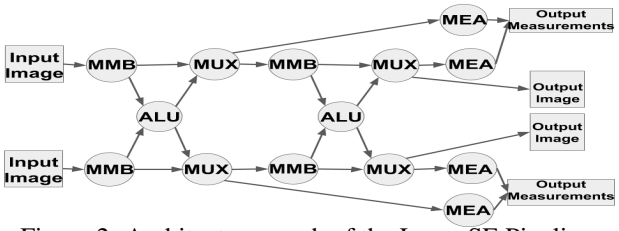

Figure 2: Architecture graph of the Large SE Pipeline.

2.2 Application example

Let's consider a text orientation detection application as an example [4]. The application detects the slope of text by searching the maximum over all orientations of linear morphological openings.

The application consists in a series of independent 1-D openings with an arbitrary angle. Then it computes intensity integrals vector where the index of the maximal element corresponds to the rotation angle(Fig. 3).

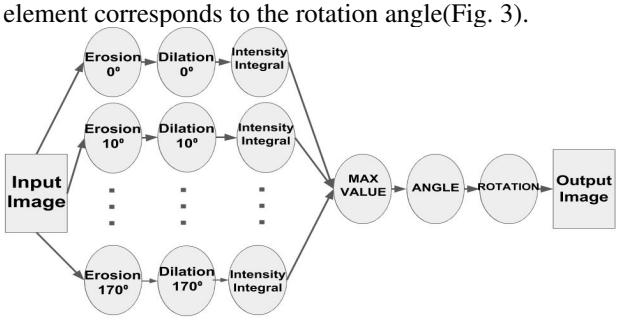

Figure 3: Graph of the text orientation application.

3 Proposed methodology

To automate the mapping of the mentioned application onto targeted architecture, and later generate the programming context, we propose an approach based on three models, each based on a DAG graph formalism:

1. **Architecture graph**: a node represents one computing module with operator types for parameters; also we define the weight of an edge as the computing latency of the tail node.
2. **Application graph**: a node represents a one operator of the application and the edges are oriented according to the application data flow.
3. **Implementation graph**: is obtained by transformations of the two previous graphs. From this graph, we generate the mapping list, which is a list of the resulting configurations and parameter data.

At first, we perform a topological sorting of the architecture graph and the application graph. Then, we apply a decision procedure. Here, we pass through the topological sorting result to define the correspondences between nodes, related to the type of module and the type of operation, on the architecture graph. Progressively, the implementation graph is created, from which we will build the final mapping list to configure the architecture.

3.1 Preliminary results

The text skew detection requires to implement 17 independent processing steps (Figure 3). Considering that the MCPU allows to implement only two steps at the same time, the applied methodology allows to create the 9 required processing instances. Indeed, when the application graph cannot be mapped onto the architecture graph, the mapping process iterative creates the needed instances, by storing the partial results in one or several temporal subgraphs, depending on the needs. When the process finishes, the created subgraphs merge in the final implementation graph.

4 Conclusions

In this paper, we presented a first study of an application mapping on pipeline-based architectures dedicated to the latency critical tasks. The objective is to automate this application process. We explore the possibilities of extension of the graph-based mapping methodology. We developed the mapping algorithm for the MCPU, and we compared it to the empirical results. The future work will focus on the development of a framework that includes the mapping algorithm and the application of a scheduling methodology, by introducing the latency optimization.

References

[1] J. Bartovský, P. Dokladal, M. Faessel, and at al. Morphological co-processing unit for embedded devices. JRTIP, pages pp. 1–12, Jul 2015.

[2] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. LeCun. Neuflow: A runtime reconfigurable dataflow processor for vision. In CVPR 2011 WORKSHOPS, pages 109–116, June 2011.

[3] Y. Ma, B. Gong, and L. Zou. Energy-efficient scheduling algorithm of task dependent graph on dvs-unable cluster system. In 2009 10th IEEE/ACM International Conference on Grid Computing, pages 217–224, Oct 2009.

[4] L. Najman. Using mathematical morphology for document skew estimation. In SPIE Document Recognition and Retrieval IX, pages 182–191, 2004.

[5] P. Possa, N. Harb, and et al. P2IP: A novel low-latency programmable pipeline image processor. Microprocessors and Microsystems, 39(7):529 – 540, 2015.

[6] A. M. Sllame and V. Drabek. An efficient list-based scheduling algorithm for high-level synthesis. In Proceedings Euromicro Symposium on Digital System Design. Architectures, Methods and Tools, pages 316–323, 2002.

[7] Y. Xu, K. Li, and at al. A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. Information Sciences, 270:255 – 287, 2014.

[1] SE - Structuring Element