Available online at www.sciencedirect.com
SCIENCE DIRECT®

Microprocessors and Microsystems 27 (2003) 115–130

MICROPROCESSORS AND MICROSYSTEMS

www.elsevier.com/locate/micpro

Temporal partitioning methodology optimizing FPGA resources for dynamically reconfigurable embedded real-time system

C. Tanougast*, Y. Berviller, P. Brunet, S. Weber, H. Rabah

Laboratoire d'Instrumentation Electronique de Nancy, Université de Nancy 1, BP 239, Vandoeuvre Lès Nancy 54506, France

Received 2 September 2002; revised 22 November 2002; accepted 27 November 2002

Abstract

In this paper we present a new temporal partitioning methodology used for the data-path part of an algorithm for the reconfigurable embedded system design. This temporal partitioning uses an assessing trade-offs in time constraint, design size and field programmable gate arrays device parameters (circuit speed, reconfiguration time). The originality of our method is that we use the dynamic reconfiguration in order to minimize the number of cells needed to implement the data-path of an application under a time constraint. Our method consists, by taking into account the used technology, in evaluating the algorithm area and operators execution time from data flow graph. Thus, we deduce the right number of reconfigurations and the algorithm partitioning for Run-Time Reconfiguration implementation. This method allows avoiding an oversizing of implementation resources needed. This optimizing approach can be useful for the design of an embedded device or system. Our approach is illustrated by various reconfigurable implementations of real time image processing data-path.
© 2003 Elsevier Science B.V. All rights reserved.

Keywords: Field programmable gate arrays; Embedded system; Temporal partitioning; Run-time reconfiguration; Dynamically reconfigurable systems; Image processing; Design implementation; High-level synthesis

1. Introduction

Since the introduction of field programmable gate arrays (FPGA), the process of digital systems design has changed radically [1,2]. Indeed, FPGAs occupy an increasingly significant place in the realization of real time applications and has allowed the appearance of a new paradigm: *hardware as flexible as programming*.

The dynamically reconfigurable computing consists in the successive execution of a sequence of algorithms on the same device. The objective is to swap different algorithms on the same hardware structure, by reconfiguring the FPGA array in hardware several times in a constrained time and with a defined partitioning and scheduling [3,4]. Dynamic reconfiguration offers important benefits for the implementation of designs. Several architectures have been designed and have validated the dynamically reconfigurable computing concept for the real time processing [5–9]. However, the optimal decomposition (partitioning) of an algorithm by exploiting the run-time reconfiguration (RTR) is a domain in which many works are left. The works in the domain of temporal partitioning and logic synthesis exploiting the dynamic reconfiguration generally focus on the application development approach [10]. Thus, firstly we observe that the efficiency obtained is not always optimum with respect to the available spatio-temporal resources. Secondly, the choice of the number of partitions is never specified. Thirdly, this can be improved by a judicious temporal partitioning [11].

We discuss here the partitioning problem for the RTR. In the task of implementing an algorithm on reconfigurable hardware, we can distinguish two approaches [10]. The most common is what we call the application development approach and the other is what we call the system design approach. In the first case, we have to fit an algorithm with an optional time constraint in an existing system made from a host CPU connected to a reconfigurable logic array. In this case, the goal of an optimal implementation is to minimize one or more of the following criteria: processing time, memory bandwidth, number of reconfigurations and power consumption. In the second case, we have to implement an

* Corresponding author. Tel.: +33-383-6841-59; fax: +33-383-6841-53.
E-mail addresses: tanougast@lien.u-nancy.fr (C. Tanougast), berville@lien.u-nancy.fr (Y. Berviller), brunet@lien.u-nancy.fr (P. Brunet), sweber@lien.u-nancy.fr (S. Weber), rabah@lien.u-nancy.fr (H. Rabah).

algorithm with a required time constraint on a system throughout the design exploration phase. The design parameter is the size of the logic array that is used to implement the data-path part of the algorithm. Here an optimal implementation is the one that leads to the minimal area of the reconfigurable array.

Previous advanced works in the field of temporal partitioning and synthesis for RTR architecture [12–19] focus on application development approach targeting already designed reconfigurable architecture. These methodologies are used in the domain of existing reconfigurable accelerators or reconfigurable processors. All these approaches assume the existence of a resources constraint. The most important thing here is that the number of reconfigurable resources is a predefined constant (implementation constraint). In this strategy, the associated tools capture the algorithm and the characteristics of the target platform on which the algorithm will be implemented. In this case, the goal is to minimize the processing time and/or the memory bandwidth requirement. Moreover, all these approaches are not capable of solving practical DRL (dynamically reconfigurable logic) synthesis problems yet. These techniques use in general simplified models of dynamically reconfigurable systems, which often ignore the impact of routing or reconfiguration resource sharing in order to reduce complexity of a DRL design space search.

Among them, there is the GARP project [12]. The goal of GARP is the hardware acceleration of loops in a C program by the use of the data path synthesis tool GAMA [13] and the GARP reconfigurable processor. GARP is a processor tightly coupled to a custom FPGA-like array and designed specially to speed-up the execution of general case loops. The logic array has a DMA feature and is tailored to implement 32 bits wide arithmetic and logic operations with the control logic, all this allows to minimize the reconfiguration overhead. GAMA is a fast mapping and placement tool for the data-path implementation in FPGAs. It is based on a library of patterns for all possible data-path operators. The SPARCS project [14,15] is a CAD tool suite tailored for applications development on multi-FPGAs reconfigurable computing architectures. Such architectures need both spatial and temporal partitioning, a genetic algorithm is used to solve the spatial partitioning problem. The main cost function used here is the data memory bandwidth. Other works propose a strategy to automate the design process that considers all possible optimizations (partitioning and task scheduling) that can be carried out from a particular reconfigurable system [16,17]. Shirazi et al. and Luk et al. [18,19] proposes both a model and a methodology to take advantages of common operators in successive partitions. A simple model for specifying, visualizing and developing designs that contains reconfigurable elements in run-time has been proposed. This judicious approach allows reducing the configuration time and thus the application execution time. But additional logic resources (area) are required to realize an implementation with this approach. Furthermore this model does not include timing aspects in order to satisfy the real time and it does not specify the partitioning of the implementation. Indeed, the algorithm partitioning must be previously known to determine the elements that do not need to be reconfigured for the next step. However, this concept is interesting to use for DRL designs simulation as *Dynamic circuit switching (DCS)* [20]. This simulation uses virtual multiplexors, demultiplexors and switches, which are implemented to simulate the dynamic configuration design.

These interesting works do not pursue the same goal as we do. In priority, we try to find the minimal area that allows meeting the time-constraint. This is different from searching the minimal memory bandwidth or execution time which allows meeting the resources constraint. Here, we propose a temporal partitioning that uses dynamic reconfiguration of FPGA (also called DRL Scheduling) to minimize the implementation logic area. Each partition corresponds to a temporal floorplanning for DRL embedded systems (Fig. 1) [21]. We search the minimal floorplan area that implements successively a particular algorithm. This approach improves the performance and efficiency of the design implementation. In contrast to previous work, our aim is to obtain, from an algorithm description, a target technology and implementation constraints, the characteristics of the platform to design or to use. This allows avoiding an oversizing of implementation resources. For example, by summarizing the sparse information found in some articles [22–24], we can assume the following. Suppose we have to implement a design requiring P equivalent gates and taking an area S_{FC} of silicon in the case of a full custom ASIC design. Then we will need about $10 \times S_{FC}$ in the case of a standard cell ASIC approach and about $100 \times S_{FC}$ if we decide to use an FPGA. But the large advantage of the FPGA is, of course, its great flexibility and the speed of the associated design flow. This is probably the main reason to include a FPGA array on System on Chip platforms. Suppose that a design is requiring that 10% of the gates must be implemented as full custom, 80% as standard cell ASIC and 10% in FPGA cells. By roughly estimating the areas, we come to the following results: The FPGA array will require more than 55% of the die area, the standard cell part more than 44% and the full custom part less than 1%. In such a case it could make sense to try to reduce the equivalent gate count needed to implement the FPGA part of the application. This is interesting because the regularity of the FPGA part of the mask leads to a quite easy modularity of the platform with respect to this parameter.

Embedded systems design can take several advantages of the use our approach based on RTR FPGAs. The most obvious is the possibility to frequently update the digital hardware functions. But we can also use the dynamic resources allocation feature to instantiate each operator