HIGH-LEVEL PARTITIONING OF DSP ALGORITHMS FOR MULTI-FPGA SYSTEMS - A FIRST APPROACH

Rafael A. Arce Nazario and Manuel Jimenez (Advisor)

Electrical and Computer Engineering Department
University of Puerto Rico, Mayagüez Campus
Mayagüez, Puerto Rico 00681-5000
{rafael.arce, mjimenez}@ece.uprm.edu

ABSTRACT

High-level synthesis and partitioning techniques have been successfully applied towards the automated design of faster, energy-aware and area-efficient digital systems. The algorithms that implement these techniques have traditionally been driven by experimental or random-decisions. This is due to the high complexity of considering the impact of each possible performance-driven transformation on the system as a whole. We believe that DSP transforms, due to their highly structured nature can be high-level partitioned by more deterministic methods. In this article we present some important issues related to these topics and begin to set the ground for future research into the high-level partitioning of DSP algorithms for multi-FPGA systems.

1. INTRODUCTION

High-level synthesis is the task of taking a behavioral (algorithmic) specification of a system and producing a description of the structures needed to implement it at a register-transfer level (RTL). At later design stages, the RTL will be transformed into a netlist for a specific hardware platform such as Application Specific Integrated Circuits (ASICs) or Field Programmable Gate Arrays (FPGAs). Figure 1 illustrates the steps in a typical CAD design flow, highlighting the steps important to our discussion. Based on functional information obtained from the behavioral specification and considerations of scheduling and allocation, *high-level partitioning* aims at dividing the RTL structures into groups such that keeping the grouped structures together will result in the optimization of some parameter. Taking partitioning decisions at this high level of abstraction helps to improve speed and results of lower level (structural) partitioning processes. High-level synthesis and partitioning have been shown to improve speed performance, reduce energy utilization and to allow the viability of implementations otherwise unattainable by other automated techniques [2], [3], [13].

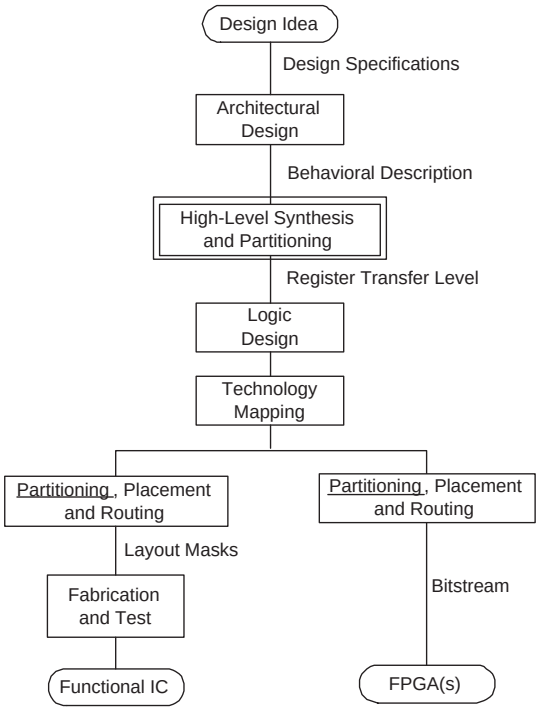

Fig. 1. VLSI CAD design flow

High-level synthesis and partitioning techniques usually start by representing the algorithmic specification in the form of a graph structure, such as a control data flow graph (CDFG). Then, a series of transformations are applied to the graph that should translate to improvement to the systems' performance. Most of the transformations deal with the time assigned to each of the operations in the algorithm (a process commonly called *scheduling*) and/or the functional unit assigned to each operation (called *allocation*).

In previously proposed methodologies for HLS and HLP we observe rather experimental and/or pseudo-random sche-

mes of applying the mentioned transformations to arrive at optimized partitions [11], [13], [3]. For example, Chen utilizes a simulated annealing algorithm to guide the low-power HLS for several data-driven applications [3]. The non-deterministic manner in which these transformations have been applied is explained by the fact that, in the general case, each transformation may have an impact on many other criteria, which cannot be easily determined a priori. Therefore, it is less computationally intensive (and hence time-efficient) to simply iterate towards a solution by making pseudo-random choices than evaluating the impact that each possible move may have on the system.

Signal processing transforms tend to have symmetric structures that can be exploited to obtain fast algorithms for their implementations. An automated framework for the derivation and implementation of such transforms was proposed by Egner [5]. The transform is decomposed into a series of sparse matrix operations that capture its redundancy. Kumhom investigated the implementation of the decomposition of the Discrete Fourier Transform to a specific shared memory architecture and showed the mappings that can be found between the matrix formulation and the scheduling and memory access schemes for such architecture [10].

We hypothesize that, given the deterministic nature implied in the formulations that can be automatically found for the DFT and other transforms, the process of high-level partitioning those algorithms can be handled in a more deterministic manner. This is, it should be feasible to guide an optimization process for HLP having in advance at least a partial knowledge of how each transformation or grouping will impact the performance of the system. An immediate result of such implications would be partitioning methods with a higher degree of predictability and that arrive at better solutions in much less time, an important contribution in the field of rapid system prototyping. We have chosen to target energy-efficiency-driven high-level partitioning specifically for multiple FPGA systems for several reasons:

- designers will continue to come up with designs that don't fit into the densest FPGAs
- to be able to exploit the inherent parallelism in some algorithms (e.g. signal processing transforms) requires a significant amount of logic: more than a single FPGA can provide
- it is a topic that has not been widely studied and therefore few solutions are available to the problem

In this article we examine some important issues related to our proposed topic of investigation, in hopes that the reader will get a sense of what factors have influenced our hypothesis and how our proposed work fits into the current streams of research of HLS and HLP.

2. CIRCUIT PARTITIONING

Circuit partitioning is a technique used to divide a circuit or system into a collection of smaller parts while trying to optimize some related objective (e.g., number of interconnections between partitions, energy-consumption) and subject to certain constraints (e.g., number of elements per partition). It is an essential stage in the design of integrated circuits for two main reasons:

1. It divides a circuit into smaller pieces such that further processing tasks can be handled more efficiently (divide and conquer strategies)
2. It breaks a large system into pieces, which can be implemented into separate interconnected components.

The general case of the circuit partitioning problem is the k-way partitioning problem, which can be formulated as a graph partitioning problem. A circuit can be represented by a graph whose vertices represent circuit elements and whose edges stand for the connections between them. The k-way graph partition problem can be stated as follows: Given a graph $G(V, E)$ where each vertex $v \in V$ has a size $s(v)$, and each edge $e \in E$ has a weight $w(e)$, divide the set V into k subsets V_1, V_2, ...,V_k, such that an objective function is optimized, subject to certain constraints.

Solving the k-way partitioning problem in an exact manner is NP-hard, i.e. the complexity grows exponentially with the number of graph vertices. Thus, automatic solutions to this problem rely on heuristic techniques. Many such techniques have been proposed, most of them being variations of two popular and accepted algorithms: the Kernighan-Lin and the Fiduccia-Mattheyses. Working from initial partitions, both of these iterative improvement algorithms arrive at satisfactory solutions by swapping nodes from one partition to another in an effort to improve a cost function. Nodes which when swapped among partitions contribute the most to the cost function are considered first. The process continues until no more benefit can be gained by swapping any pair of nodes (without violating the imposed constraints).

Proposed partitioning methods can be classified according to the information that they use to make partitioning decisions:

- Structural partitioning: These methods rely solely on flattened netlist information. The great majority of published work on circuit partitioning uses this approach [9]. The large number of nodes in a typical VLSI circuit makes even simple heuristic methods time consuming. Several techniques, clustering among them, have been introduced to make these algorithms faster, while retaining the integrity of their solutions.

- Functional partitioning: Along with netlist information, these methods use some other higher-level insight to help guide the partitioning process. The highest level that is considered for most published works is information about block hierarchy, usually derived from a design′s HDL code. For example, in multilevel partitioning, hierarchical information is used to influence clustering and de-clustering decisions. High-level partitioning divides a high-level graph structure (such as a CDFG) into groups whose members should be kept together to optimize such things as the communication trafic between chips and the number of interconnections between them. It can serve as a clustering method to reduce the number of nodes to be considered at the structural partitioning level.

3. HIGH LEVEL SYNTHESIS AND PARTITIONING

As we briefly mentioned in the Introduction, usually the first task of HLS or HLP is to obtain a graph representation that expresses the relationship among the operations, data and control of a specified algorithm, which was originally expressed in a hardware description language such as VHDL. A commonly used data structure is the Control Data Flow Graph. Some researchers have tried to apply HLS or HLP techniques directly to the resulting graph, however, this may be too dependent on the algorithm programmers style [12], [13]. Thus, a series of transformations are applied to the graph in which the functionality of the algorithm is preserved but the chances of obtaining partitions with better performance are improved. Vahid and Chen discuss some of these transformations [3],[13] . For example, a *called* procedure may be inlined into its *calling* procedure if this represents a saving in the amount of data exchanged between hardware components. Furthermore, a lengthy procedure can be split into two or more smaller ones if this guarantees a more efficient hardware utilization in the final implementation.

In Vahid's approach the graph transformations are aimed at Hardware/Software partitioning and their effectiveness is only attested experimentally, i.e. there seems to be no measure of how any given transformation might affect final partitioning results [13]. Also, only one type of transformation is applied per experiment (the iteration and combination of transformations is left as future work). In Chen's work the transformations are conducted by a simulated annealing process which "randomly" chooses each move and then evaluates the resulting systems power consumption [3]. We propose to investigate how information from the functional structure of DSP transforms can be used to guide the application of the transformations in a more deterministic manner. We believe that a global cost function can be derived from the DSP matrix formulation that quantifies transformation impact on the overall performance of the system. This function can be used to guide the partitioning process by explicitly indicating which moves (and in what order) will result in better partitions.

4. ENERGY EFFICIENCY

The majority of energy spent on CMOS digital circuits is due to dynamic power, expressed by the following equation:

$$\sum_{all\ nodes} \frac{1}{2} C_y V^2 D(y) f_{clk} \ , \qquad (1)$$

where C_y is the load capacitance of a gate y, V is the voltage swing and $D(y)$ is the transition density of gate y. FPGAs typically consume more energy than ASIC solutions since they need to use a higher number of transistors (in interconnections and logic) to implement a given functionality [3]. Most of their consumed energy is attributed to the interconnect, which results in higher load capacitances at each gate [7]. Although no reports have been found on the energy utilization of multi-FPGA systems, we would expect that the inter-FPGA connections, being even longer than the intra-chip ones, have an even greater impact on the energy utilization of such systems. This represents yet another constraint to partitioning in multi-FPGA systems, along with I/O pin and area limitations.

Multi-FPGA systems have been traditionally used as prototyping tools for logic emulation. However, we believe that the need for reconfigurability and cost-effectiveness for low-volume applications will increase their attractiveness especially in niche and remote applications (e.g. space exploration, remote sensing). In these situations, power consumption becomes a crucial issue and thus the pertinence of our research to the digital systems design area.

5. MULTI-FPGA SYSTEMS

A multi-FPGA system (MFS) includes several FPGA's connected in a constrained routing topology [8]. This differentiates them from custom-made single or multiple-chip systems where the designer can specify the physical connections between chips before the system is fabricated. The task of mapping an application to a multi-FPGA system is complex, specially when the application is not well-structured, so automated mapping and partitioning methods have been researched [8].

Multi-FPGA system vendors provide off-the-shelf cards with options as to the density/speed of each chip as well as the number of FPGAs (some systems have more than 200 FPGAs). Synplify's Certify, the only commercial tool we have found aimed specifically at multi-FPGA partitioning, doesn't perform performance-oriented partitioning. This is not surprising since MFSs are not commonly thought of as

final implementation platforms but rather as logic emulation tools for prototyping. As MFSs become final platforms for niche applications, issues such as power and speed will become essential goals in the partitioning process. Our research will be directed towards the power-efficiency goal from a high-level synthesis perspective.

6. DSP APPLICATIONS

It is not uncommon to encounter hardware implementations of DSP algorithms that require more logic or interconnections than a single FPGA can provide. Thus, they are prototyped to multi-FPGA systems. To make rapid prototyping to these systems a feasible task, efficient automatic partitioning tools are essential. Several authors hint at the need for partitioning methods that use higher-than-hierarchy functional information [1], [4], [6]. Based on these remarks, we hypothesize that incorporating higher-level functional information into the partitioning process will help to further improve performance in multi-FPGA implemented systems. In the case of DSP applications, higher-level information may include insights into the signal operations and mathematical constructs used in the DSP algorithms formulations, and how they can be used to efficiently guide the partitioning process.

Signal processing transforms tend to have symmetric structures that can be exploited to obtain fast algorithms for their implementations. An automated framework for the derivation and implementation of such transforms was proposed by Egner [5]. A signal processing transform is decomposed into a series of sparse matrix operations that capture its redundancy. Later, these decompositions can be compiled into imperative-style numerical programs. Kumhom investigated the implementation of an automated decomposition of the Discrete Fourier Transform to a specific shared memory architecture and showed the mappings that can be found between the matrix formulation and the scheduling and memory access schemes for such architecture [10]. The methodology by Egner is capable of producing different formulations for a given transform. Each formulation dictates an implementing algorithm. We sense that there is a relation between the kind of manipulations involved in the matrix decompositions proposed by Egner and the graph transformations suggested by Vahid and Chen. We shall further investigate these topics in hopes of finding a mapping betweem them.

7. FUTURE WORK

Our work is still in its early stages. Some issues we shall be addressing in the near future are:

- Evaluate the performance of existent tools for partitioning of DSP algorithms into multi-FPGA systems
- Analyze a class of DSP algorithms in search of structures and operators that may be introduced as cost functions to guide high-level partitioning
- Devise a generalized representation or set of rules to apply functional information to the partitioning process.
- All previous steps should be aware of other CAD stages that may be affected by new partitioning methodology (e.g., synthesis, floorplanning, placement)

8. REFERENCES

[1] C. J. Alpert and A. B. Kahng. Recent directions in netlist partitioning: a survey. *Integr. VLSI J.*, 19(1-2):1–81, 1995.

[2] O. Bringmann, C. Menn, and W. Rosenstiel. Target architecture oriented high-level synthesis for multi-FPGA based emulation. In *Proceedings of the European Design and Test Conference 2000*, pages 326–332, 2000.

[3] D. Chen, J. Cong, and Y. Fan. Low-power high-level synthesis for FPGA architectures. In *Proceedings of the 2003 international symposium on Low power electronics and design*, pages 134–139. ACM Press, 2003.

[4] Y. Cheon and D. F. Wong. Design hierarchy guided multilevel circuit partitioning. In *Proceedings of the 2002 international symposium on Physical design*, pages 30–35. ACM Press, 2002.

[5] S. Egner, J. Johnson, D. Padua, M. Pschel, and J. Xiong. Automatic derivation and implementation of signal processing algorithms. *ACM SIGSAM Bulletin Communications in Computer Algebra*, 35(2):1–9, 2001.

[6] W.-J. Fang and A. C.-H. Wu. A hierarchical functional structuring and partitioning approach for multiple-FPGA implementations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(10):1188–1195, 1997.

[7] V. George, H. Zhang, and J. Rabaey. A flexible power model for fpgas. In *Proceedings. 1999 International Symposium on Low Power Electronics and Design*, pages 188–193, 1999.

[8] S. Hauck. *Multi-FPGA Systems*. PhD thesis, University of Washington, 1995.

[9] F. M. Johannes. Partitioning of VLSI circuits and systems. In *Proceedings of the 33rd annual conference on Design automation conference*, pages 83–87. ACM Press, 1996.

[10] P. Kumhom. *Design, Optimization, and Implementation of a Universal FFT Processor*. PhD thesis, Drexel University, 2001.

[11] E. Lagnese and D. Thomas. Automatic derivation and implementation of signal processing algorithms. *ACM SIGSAM Bulletin Communications in Computer Algebra*, 10(7):847–860, 1991.

[12] A. Saha and R. Krishnamurthy. Some design issues in multichip FPGA implementation of DSP algorithms. In *Proceedings of the Fifth International Workshop on Rapid System Prototyping*, pages 131–140, 1994.

[13] F. Vahid. A three-step approach to the functional partitioning of large behavioral processes. In *ISSS*, pages 152–157, 1998.