An Assessment Of High-Level Partitioning Techniques For Implementing Discrete Signal Transforms On Distributed Hardware Architectures*

Rafael A. Arce-Nazario, Manuel Jimenez and Domingo Rodríguez
Electrical and Computer Engineering Department
University of Puerto Rico, Mayagüez Campus
Mayagüez, Puerto Rico 00681-5000
Email: {rafael.arce, mjimenez, domingo}@ece.uprm.edu

Abstract— **Partitioning is an essential step in the implementation of algorithms to distributed hardware architectures (DHAs) such as multi-FPGA boards. While numerous approaches working at the structural level have been reported, techniques targeted at higher levels are less common. Moreover, when dealing with discrete signal transforms (DSTs), formulation-level partitioners for DHAs have been largely neglected. In this paper, we introduce a first approach towards a functionally-aware methodology that could provide improved results for the high-level partitioning of DSTs to DHAs. Our methodology has been devised through the study of DST partitioning techniques for DHA-similar systems, as well as general DST formulation techniques. An assessment performed on discrete Fourier transforms has achieved as much as 35% in latency reduction when compared to other general, high-level partitioning schemes.**

I. Introduction

Applications for discrete signal transforms (DSTs), such as the discrete Fourier transform (DFT), abound in such diverse fields as communications, biomedical sciences, and astronomy. In these and many other fields, increases in the quantity and resolution of data and the need for faster processing demand novel platforms and methodologies for the implementation of DSTs. Distributed (dedicated) hardware architectures (DHAs), such as multi-FPGA boards, represent a cost effective option for the high performance implementation of DSTs and other performance-demanding algorithms.

The achievement of effective implementations to this type of architectures is highly dependent on the process of partitioning. Numerous approaches have been proposed for manual and automated partitioning of algorithms at the behavioral and structural levels. At the structural-level, techniques incorporate unit/module cloning and communication channel multiplexing in an effort to solve the communication overhead and I/O limitations characteristic of DHAs. Some researchers have introduced higher-level considerations to influence structural partitioning methods [1]. Despite these attempts, incorporating functional information into structural partitioning methods is difficult, since at this level the structures that implement an algorithm have already been decided and high-level information has been fundamentally lost.

* This research was partially supported by NASA Space Grant NGT5-40091 and NSF grant EPS-0223152.

A commonly observed fact in CAD tools is that optimization methodologies that work at higher levels of abstraction usually achieve better performance than their lower level counterparts. This behavior has also been observed in high-level partitioning (HLP) methods [2]. Although several HLP methods have been reported, most of them are designed to solve general partitioning problems, and tend to apply generic local optimization techniques that miss out on alternate formulations that become apparent only with some knowledge of the algorithm's functionality.

The algorithmic formulation of DSTs, specially that of the DFT, has been extensively studied. Automated computational algebra platforms for the algorithmic manipulation of fast transform algorithms have been proposed, as well as automated methods to optimize DST implementation to general purpose processor platforms [3] [4]. Techniques for the reduction of complexity and/or number of operations for DFTs have been known for decades (e.g. FFT, Winograd's Fourier Transform Algorithm), and have been utilized manually to improve the hardware mapping and implementation of this class of algorithms. However, to the best of our knowledge, these methods have yet to be successfully adapted for implementations to dedicated hardware platforms, particularly for dedicated DHAs. By analyzing automated as well as manual partitioning strategies for DST hardware and parallel processor implementations, we have begun to identify techniques that, when integrated into the partitioning process, should lead to a more straightforward exploration of the solution space when compared to other generic, high-level partitioning methods. This article reports on our initial findings regarding the use of algorithmic level characteristics and features of the DSTs to improve their partitioning to DHAs.

II. High Level Partitioning for DHAs

HLP methods targeted to distributed hardware architectures (DHAs) typically operate on a flow graph representation of the algorithm. Their structure includes a high-level cost estimation mechanism coupled with a partitioning engine that relies on probabilistic or heuristic decisions to improve on an initial solution. Besides the selection of appropriate partitioning heuristics and algorithms, researchers in this field have focused their attention to matters such as the integration of HLP onto the high level synthesis process, the granularity of the partitioned flow graph and the input specification. The order in which partitioning is performed, relative to other high-level synthesis tasks such as allocation/scheduling, can impact the speed and success of a design space exploration. Approaches performing partitioning completely separate from allocation and scheduling or as an integrated step have been reported with different levels of success [5][6].

The granularity of the partitioned flow graph has also been a consideration. Fine-grained representations potentially allow for a thorough exploration of the solution space. Coarse-grained representations have the advantage of possessing fewer nodes, which frequently results in a faster exploration. Methods that combine the virtues of both levels have been proposed, relying on the user to manually build the coarse-grain representation [5]. This makes the quality of results dependent on the user's programming style and requires the programmer to be reasonably familiarized with the target system in order to obtain effective results.

III. Discrete Signal Transforms on DHAs

DSTs have received a variety of partitioning treatments when intended for DHA implementation. Discrete Fourier and Cosine Transforms (DFTs and DCTs) are sometimes used as part of the benchmark set for general HLP DHA methodologies [5][6]. Manual implementations have also been proposed for DSTs, both for their matrix representation and fast formulations. The implementation of these fast versions requires a regular but congested communication scheme among the various computational elements, which can dominate performance depending on the target architecture and the utilized partitioning strategy [7]. Some approaches have recurred to exhaustive methods to search for optimal partitions of an FFT, resulting in partitioning schemes similar to data allocation schemes proposed by researchers in the general purpose distributed system area [8]. This variety of treatments underlines the need for automated methods to exploit DST characteristics to improve their partitioning to DHAs.

IV. DSTs and Kronecker products algebra

Our partitioning methodology aims at improving design-space exploration speed and results by incorporating DST characteristics, such as their regularity, redundancy, and the ability to reformulate at the algorithmic level. The effective use of these properties for the mapping and partitioning of DSTs requires a framework for their representation and manipulation in which algorithmic rules can be applied and hardware structures can be deduced in a straightforward manner. We believe Kronecker products algebra to be an advantageous framework for such purposes.

Any linear separable transform of a d-dimensional discrete signal $x[n]$, where each element can be specified by d indexes n_1, …, n_d, can be defined by:

$$X[k_1,..,k_d] = \sum_{n_d} .. \sum_{n_1} x[n_1,..,n_d]\alpha_1(n_1,k_1)..\alpha_d(n_d,k_d) \tag{1}$$

where the α_i's are the transform's kernels. For example, for the d-dimensional DFT $\alpha_i(n_i,k_i) = e^{-j2\pi n_i k_i/N_i}$, and for the d-dimensional DCT $\alpha_i(n_i,k_i) = cos\,[\pi/2N_i(2n_i+k_i)]$. The separable condition of linear transforms induces mathematical formulation of these multidimensional transforms in terms of Cartesian or Kronecker products, in finite dimensional multilinear algebra setting. If the linear transformations are unitary as well, then finite abelian group theory is used on the input/output data indexing sets to describe the transformations in terms of characters or exponential functions of the set of integers modulo the orders of the indexing groups and their Cartesian products. The character representation also induces a Kronecker products representation.

Kronecker products algebra has been used successfully, in a manual manner, to assist in the implementation of fast algorithms for the computation of the unitary transforms. The main idea is as follows. A given mathematical canonical formulation of an algorithm is usually expressed as a composition of what are termed functional primitives, factors which have been identified as efficient procedures on the targeted DHA, establishing in this a one-one correspondence. Variants of this canonical formulation are then sought using properties of Kronecker products algebra and trying to satisfy certain design criteria such as pipelining, parallelism, data flow control, etc., each new variant, in turn, producing a different DHA implementation. The efficiency of the algorithm is usually evaluated through a cost objective function imposed on the design framework.

V. Paritioning Methodology

The main components of our partitioning methodology are illustrated in Figure 1. We begin by accepting a specification of the DST using Kronecker algebra, along with implementation details such as the resolution of the transform points. High level partitioning is conducted in two main stages: (1) the algorithmic formulation exploration (AFE) stage followed by (2) a refinement stage.

In the AFE stage, algorithmic formulations for the specified DST are generated and evaluated in search for one whose corresponding partitioning/placement scheme exhibits minimal communication and area costs for the targeted architecture. To do this, each algorithmic formulation is represented as a coarse-grained flow graph whose nodes denote functional primitives. Each of these functional primitives has been identified and characterized as a procedure or hardware block that can be efficiently implemented to one of the architectural devices. Then, a deterministic partitioning algorithm is run on the graph, assisted by high-level area and communication estimators. The algorithmic formulation that minimizes the area/communication objective function and its corresponding partitioning/placement scheme are set for the next stage.

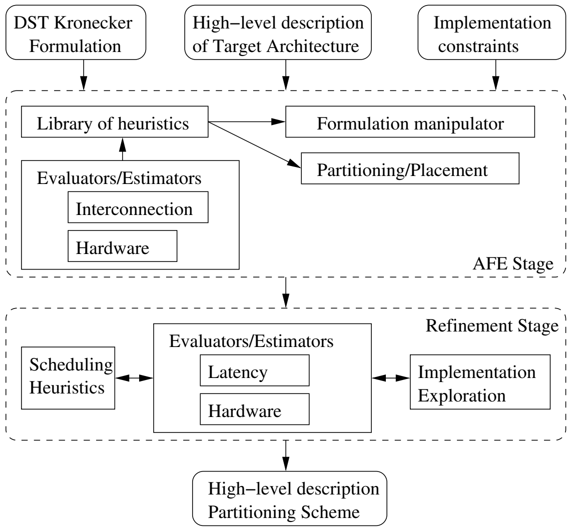

Fig. 1. Conceptual map of methodology.

In the refinement stage, the actual functional units that will be implemented to each of the assigned tasks are determined. For this task, we first conduct a functionally-aware allocation process in which the number and type of functional units is determined for the group of tasks assigned to a device. Following this, scheduling opportunities are explored in an effort to further improve the latency of the implementation by moving specific components to underutilized devices.

We foresee several advantages of our methodology over previous schemes. Granularity of the partitioned graph is coarse, resulting in a fast partition space exploration, yet its nodes have been deduced from a formulation that can be transformed to result in improved implementation given the partitioning heuristic and the underlying architecture, avoiding dependence on programmer style. The regularity of DST algorithms allows us to handle their allocation and scheduling aside from the partitioning loop, contributing to the speed of the HLP solution space exploration.

VI. Assessment

An assessment of the proposed partitioning methodology has been done for several DFTs targeting a multi-FPGA architecture modeled after the Wildforce family from Annapolis Micro Systems. The target architecture, illustrated in Figure 2, consists of 4 FPGAs connected in a linear array topology with a crossbar serving as a global communication channel. Similar assumptions and architecture have been used before to report fine and coarse-grained high-level multi-FPGA partitioning heuristics [5]. We shall refer to these heuristics as *SBPH* henceforth. The cost of communications and memory I/O are as follows: 2 cycles for exchanging data through the inter-device lines (c_{01}, c_{12}, c_{23}), 4 cycles for communicating through the crossbar (c_{XB}), 3 cycles for a memory read and 1 cycle for each memory write. In addition, 4 cycles are assumed for synchronization during communication between devices. Both the direct interconnections and the crossbar are 36 bits wide. Each FPGA is a Xilinx XC4013 with 576 available configurable logic blocks (CLB). We illustrate the process of partitioning a 16-bit fixed-point 4×4 FFT to this architecture using the proposed methodology, with latency as our minimization objective. Our main intention is to show the impact of formulation exploration on the partitioning results.

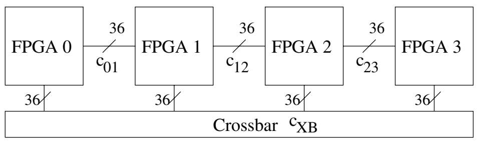

Fig. 2. Target architecture for assessment.

The formulation of a size-4 × 4 FFT using Kronecker Algebra could be stated as follows[1]:

$$F_{4\times4} = (F_4 \otimes I_4)(I_4 \otimes F_4)P_4^{16} \tag{2}$$

A coarse-node flow graph consisting of 8 nodes, each representing a size-4 FFT can be deducted from the previous expression, as illustrated in Figure 3a. Each of these nodes can have different implementations, which vary in latency and area. If a maximum area / minimum latency implementation fits in the available area, we use it trusting that it will result in a minimal latency global implementation. Otherwise, we use the minimal area for all nodes during the initial partitioning phase and explore other implementations later. In the $F_{4\times4}$ case, the maximum area implementation is estimated to fit into the compound logic area of the target system. Each F_4 area is estimated at 128 CLBs, resulting in a total functional unit area of 1024 CLBs, notably less than the platforms capacity of 2304. This leaves more than 50% of the available logic area for the control unit and additional registers needed for the final implementation.

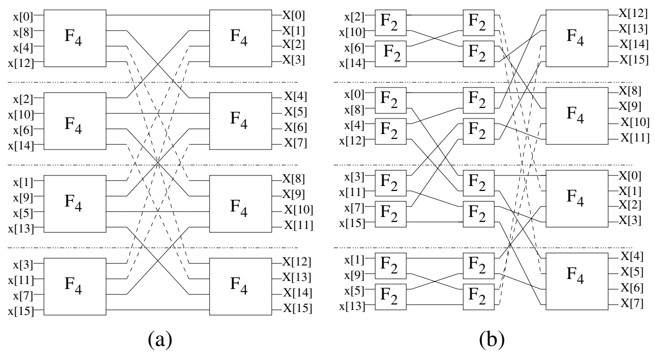

Fig. 3. Comparison of two equivalent formulations for a 4 × 4 FFT. Dashed horizontal lines represent partitions. Dashed interconnections denote communications through the crossbar.

[1]In this notation: $\otimes$ represents the Kronecker product, P_m^n is a stride m permutation, T_m^n is a diagonal matrix of coefficients, also known as the twiddle factors, and F_N is a size-N Fourier transform.

Given that communication channels will probably represent the most constrained resource for highly connected algorithms such as the fast versions of DSTs, we use the cut size as a high-level estimator that should translate to reduced latency on the implementation. The coarse node graph is partitioned with the objective of minimizing an objective function that considers the cost for communicating through the various architectural channels:

$$F_{obj} = \max_{p \in paths} (\sum_{i \in stages} Cost(c_{p,i})) \ , \qquad (3)$$

where *paths* refers to the input to output paths in the graph, and *stages* refers to the various stages of communication between the processing elements. $Cost(c_{p,i})$ is the product of the cost for the communication channel crossed by path p during stage i, if any, and the number of points that are transferred through that channel. For example, the partition scheme shown in Figure 3a has 1 stage of communication and requires that six points communicate through the crossbar, which has a cost of 4. The objective function evaluates to 24.

Our partitioning heuristic guides the reformulation onto an expression such as:

$$F_{4\times4} = (F_4 \otimes I_4)(I_4 \otimes ((F_2 \otimes I_2)T_2^4(I_2 \otimes F_2)))P_4^{16} \ , \quad (4)$$

whose partitioning scheme, shown in Figure 3a would reduce the cost function to 16. A scheduled version of this partition scheme would imply a latency of 116 cycles using an ASAP algorithm. This can be compared to previous works in HLP that have used DSTs as part of their benchmark set, which obtained a latency of 179 cycles for an $F_{4\times4}$ [5]. We believe that our advantage lies in the use of formulation-defined coarse graphs as well as the reliance on neighbor direct device connections.

Table 1 shows the estimated latency for several FFT sizes using our methodology vs. lower bound projections for SBPH. The lower bounds were obtained by scaling the reported SBPH $F_{4\times4}$ results, using the approach presented by Hagen, et al.[9]. This approach essentially states that given a heuristic H, an instance I and the solution cost for the instance $c_H(I)$, if a new instance kI is constructed by scaling the original by k, then $c_H(kI) > k \cdot c_H(I)$. Latency is computed as the sum of the I/O latency, processing latency, and intercommunication latency (i.e., time spent waiting for data from other partitions). I/O latency scales linearly with the size of the transform, while processing latency, being proportional to the number of operations, is scaled accordingly. Intercommunication latency is scaled linearly, following the lower bound cut size for butterfly networks proved by Bornstein, et al. [10]. As is observed in Table 1, estimated latencies obtained using our method compare favorably with the lower bound estimates, achieving up to 35% latency reduction. Furthermore, we estimate that a tool that implements our methodology would avoid excessive run-times as encountered in some reported work. First, our methodology removes the scheduling/allocation tasks out of the partition iterative improvement loop. Secondly, the regularity of the flow graphs allows us to use graph partitioning heuristics which are not purely stochastic and thus time consuming. For instance, we have used METIS [11], a highly effective and fast graph partitioner using the multi-level partitioning paradigm, to obtain competitive partitioning schemes for graphs obtained from Kronecker formulations of the FFT.

TABLE I
ESTIMATED LATENCY FOR SEVERAL FFT SIZES.

FFT size	Lower bound [5] (cycles)	Our approach (cycles)
4×4	179	116
8×8	804	696
16×16	3680	3080
32×32	16800	13520

VII. CONCLUSIONS AND FUTURE WORK

Our initial assessment suggests that the integration of DST characteristics onto the high-level partitioning of these algorithms promises to improve partitioning results as well as lead to faster design space exploration. We have evidenced how transformations at the formulation level can favorably impact partitioning results. Although our assessment only reflects experimentation with Discrete Fourier Transforms, we foresee that our methodology will be effective for other DSTs as well, given that they exhibit similar regularity and formulation features. Experimentation with other DSTs is part of our future work. Our plans also include the development and validation of the various components of the proposed methodology, and the integration of these components into a prototype high-level partitioning tool.

REFERENCES

[1] Wen-Jong Fang and Allen C.-H. Wu. Multiway FPGA partitioning by fully exploiting design hierarchy. *ACM Trans. Des. Autom. Electron. Syst.*, 5(1):34–50, 2000.

[2] N. Kumar, V. Srinivasan, and R. Vemuri. Hierarchical behavioral partitioning for multicomponent synthesis. In *EURO-DAC '96*, pages 212–217, 1996.

[3] Markus Püschel et al. SPIRAL: Code generation for DSP transforms. *Proceedings of the IEEE, special issue on "Program Generation, Optimization, and Adaptation"*, 93(2), 2005.

[4] Matteo Frigo and Steven G. Johnson. The Design and Implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

[5] V. Srinivasan, S. Govindarajan, and R. Vemuri. Fine-grained and coarse-grained behavioral partitioning with effective utilization of memory and design space exploration for multi-FPGA architectures. *IEEE Trans. Very Large Scale Integr. Syst.*, 9(1):140–159, 2001.

[6] 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.

[7] A. Jones, A. Nayak, and P. Banerjee. Parallel Implementation of Matrix and Signal Processing Libraries on FPGAs. In *Intl. Conf. on Parallel and Distributed Computing and Systems (PDCS)*, August 2001.

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

[9] L.W. Hagen, D.J.H. Huang, and A.B. Kahng. Quantified suboptimality of VLSI layout heuristics. In *DAC'95*, pages 216–221, 1995.

[10] C. F. Bornstein, A. Litman, B. M. Maggs, R. K. Sitaraman, and T. Yatzkar. On the bisection width and expansion of butterfly networks. In *Proceedings of the 12th International Parallel Processing Symposium*, pages 144–150, March 1998.

[11] George Karypis. *Multilevel Hypergraph Partitioning*, chapter 1. Kluwer Academic Publishers, 2002.