DMAGIC: A High-level Partitioning Methodology for Discrete Signal Transforms onto Distributed Hardware Architectures

Rafael Arce-Nazario, Manuel Jiménez, Domingo Rodríguez
Physics and Electronics Dept., University of Puerto Rico-Humacao / Electrical and Computer Eng. Dept., UPR-Mayagüez
ra_arce@uprh.edu, {mjimenez, domingo}@ece.uprm.edu

Introduction

Discrete signal transforms (DSTs) represent a major energy, performance, and area component in many applications, which merits the study of specialized methods for optimizing their implementation to modern computing platforms. Large scale applications such as synthetic aperture radar (SAR) imaging and chirp orthogonal frequency division multiplexing (OFDM) require diverse DSTs as their building blocks. Currently, DST implementations to distributed dedicated hardware platforms are of particular interest for two main reasons. First, to attain superior performance, the size and composition of DSTs frequently require implementation to multi-chip architectures, such as multi-FPGA boards. ASIC and FPGA implementations of DSTs have been shown to operate many times faster than general-purpose or DSP processor versions. However this performance gain is achieved at the expense of high resource utilization. Second, driven by concerns about power/performance in ultra deep submicron CMOS technologies, there is a current trend towards multi-core computing architectures. Targeting these types of architectures demands a paradigm shift in algorithm mapping onto hardware structures, as well as tools to assist in the implementation process.

Several high-level partitioning strategies have been proposed for distributed hardware architectures (DHAs) which use generic local optimization techniques [1]. Although these techniques provide good results when applied to common benchmarks, they miss out on alternate considerations which become apparent with knowledge of the algorithm's functionality. DSTs possess algorithmic level properties that have been used to obtain effective formulations for diverse computational architectures. For instance, DSP code-generation techniques, such as SPIRAL [2], explore alternate algorithmic formulations as part of their search for optimized DST implementations on general purpose processors. However, these strategies have not been properly adapted to target distributed hardware architectures.

In this paper, we present DMAGIC, a high-level partitioning methodology for DSTs onto distributed hardware architectures (DHAs) that takes advantage of DST features at two levels of abstraction: the graph and algorithmic levels. At the algorithmic level, an exploration is conducted in search of equivalent transform formulations that are more suitable for the target topology. At the graph level, several DST-specific structural considerations have been integrated into a partitioning heuristic, resulting in faster graph partition solutions with no loss in quality. The developed strategy integrates several processes which allow the exploration of partitioning solutions at both abstraction levels. This interaction allowed DMAGIC to obtain improved partitioning results in terms of implementation latency and runtime, as compared to generic hardware partitioning strategies.

DMAGIC Methodology

Figure 1 shows a conceptual map of the proposed partitioning methodology, called DMAGIC (DST Mapping using Algorithmic and Graph Interaction and Computation). The inputs on the top are a DST specified as a Kronecker Products Algebra (KPA) formulation, parameterized at least by the resolution of its points, and a high-level specification of the target architecture, which includes the number and logic capacity of the devices and their connection topology. The *Kronecker to Graph* process converts the algorithmic formulation into a dataflow graph (DFG) whose nodes denote functional primitives, i.e. small computational components that are common throughout the formulation and have been identified as efficient procedures on the target devices. The dataflow graph is partitioned using a deterministic graph partitioning/placement (P/P) heuristic that has been enhanced to handle typical fast DST structures. An area estimator determines the available processing elements in each architectural device, based on device capacity and the partition scheme. Latency estimation is done using an As-Soon-As-Possible scheduling heuristic, constrained by the resources estimated by the area estimator.

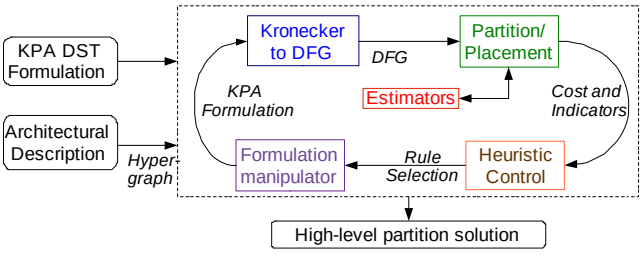

Figure 1: Partitioning methodology.

Based on the P/P results of the current formulation, factorization rules are used to generate a new formulation, which, hopefully, improves the previous results. The conversion/partitioning/reformulation process continues until no considerable improvements are detected, at which point the methodology outputs the best partition/placement scheme encountered throughout the exploration.

Distributed Hardware Architecture

Figure 2 illustrates our target architecture model, referred to as a Distributed Hardware Architecture (DHA). It consists of k dedicated hardware devices (D_0,..,D_{k-1}) with local memory (M_0,..,M_{k-1}), connected in a ring or linear array topology with a crossbar serving as a global communication channel. This architecture is modeled after common multi-FPGA boards produced by vendors such as Annapolis (Wildforce) and Gidel (PROC20KE), as well as high-end academic reconfigurable systems such as the Berkeley Emulation Engine 2 (BEE2). Furthermore, this architecture can be considered scalable due to the number of connections per device and its topological symmetry.

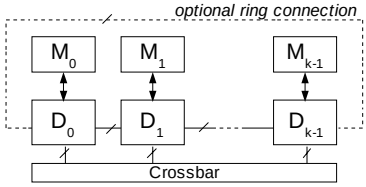

Figure 2: Distributed Hardware Architecture

Graph partitioning tools

The software tools and heuristics developed for DMAGIC have been influenced by our intention of using DST properties throughout the partitioning process. For instance, the graph partitioning/placement heuristic takes advantage of DST regularity by constructing initial solutions that are aware of the butterfly structure [3]. The resource estimator achieves accurate area estimates by targeting an intra-device architecture typical of folded DST structures.

Moreover, several of these individual tools could be used to further understand and research the properties of DSTs and their hardware mapping. For example, the Kronecker to DFG tool can be used to graphically visualize parallelisms and partitioning opportunities in DST formulations.

Algorithmic exploration

A great number of algorithmic-level rules exist to reformulate DSTs into functionally-equivalent / computationally-distinct formulations. This is augmented by the fact that the space of equivalent DST formulations grows exponentially with DST size. Thus, an effective formulation-space exploration heuristic is essential to our proposed methodology. DMAGICs heuristic was established after experimentally assessing the effect of algorithmic reformulations, such as factorizations and permutations, on partition quality

The initial assessment was performed on FFTs, due to their high regularity and the availability of alternate formulations [3]. Exhaustive formulations were generated for small-sized FFTs using a small set of regularity-conserving rules and partitioned using the DMAGIC graph partitioning components. Analysis of the partitioning results helped define a greedy polynomial-time heuristic that utilizes the Cooley-Tukey factorization rule to deterministically explore the space of equivalent FFT formulations. This technique was later extended to DCTs by deriving a Cooley-Tukey-like regular DCT algorithm [4].

Results and Discussion

Figure 3 shows the results of partitioning a range of FFT and DCT sizes with DMAGIC vs. an implementation of a previously published high-level partitioning methodology [1]. The target architecture consisted of four FPGAs connected in ring topology and crossbar. Neighbor and crossbar communication, as well as arithmetic operations latencies are as in [1]. Latency reduction of up to 34.1% can be attributed to the exploration of alternative formulations and to the DST considerations taken at the graph level. Furthermore, significant runtime savings are due to the fact that the formulation exploration algorithm takes advantage of coarse DST representations, whereas [1] relies on fully expanded dataflow graphs.

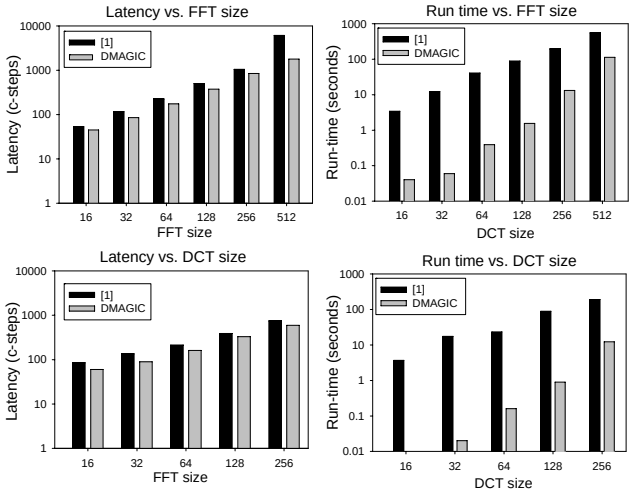

Figure 3: Results for FFT and DCT partitioning.

Conclusions and Future Work

Algorithmic-level reformulations affect the implementation performance of DSTs to distributed hardware platforms. Graph-level awareness of common DST structures helps accelerate graph partitioning for these transforms. DMAGIC uses DST features at both of these abstraction levels to produce improved partitioning solutions in a time-effective manner. Future work includes DMAGIC's extension to additional transforms, targeting current SoC topologies, and multi-core processors, e.g. CELL BE.

References

[1] V. Srinivasan, et al., "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*., vol 9, n 1, 2001.

[2] M. Püschel, et al., SPIRAL: Code Generation for DSP Transforms, Proceedings of the IEEE, vol. 93, no. 2, 2005.

[3] R. Arce-Nazario, et al., "Functionally-aware Partitioning of Discrete Signal Transforms for Distributed Hardware Architectures". *Proceedings of the 49th MWSCAS*. 2006.

[4] R. Arce-Nazario, et al., "Mapping of DCTs onto Distributed Hardware Architectures". Submitted to *Journal of VLSI Signal Processing*. April 2007.