An Analytical Model of Memory-Bound Applications Compiled with High Level Synthesis

Maria A. Dávila-Guzmán*, Rubén Gran Tejero†, María Villarroya-Gaudó‡ and Darío Suárez Gracia§
DIIS-I3A, Universidad de Zaragoza — HiPEAC Network of Excellence
Email: *angelicadg@unizar.es, †rgran@unizar.es, ‡mvg@unizar.es, §dario@unizar.es

I. Introduction

HLS tools simplify programming for FPGAs, but generating highly tuned code still remains a challenge because CPU and GPU optimization techniques are not always directly applicable to FPGA. Besides, bitstream generation takes a long time, preventing any "trial-and-error" optimization process. To address this issue, programmers can follow two alternatives. Either they write well-known code patterns from previous explorations, or they rely on pre-synthesis analytical models for estimating performance.

The delay of HLS generated code can be attributed to two main components: compute and memory. While existing analytical models focus more on the compute part [1], [2], or kernel pipeline, this work proposes a memory-focused approach for Intel FPGAs that accurately models the Global Memory Interconnect (GMI), connecting the kernel pipeline with the off-chip DRAM. Our analytical model mainly requires static information and can be easily plugged into existing models to support memory-bound applications, or, even, integrated into HLS tools to guide optimizations.

II. Proposed Model and Results

In HLS source code, each reference to an external variable constitutes a *global access*. Since the main sources of kernel stalls are these global accesses, the GMI implements several strategies to maximize DRAM throughput and kernel pipeline flow. Internally, the GMI architecture has two components: 1) Load/Stores Units (LSUs), tracking in flight memory accesses, and 2) Arbiters, ordering read and write accesses.

Depending on the access pattern, Intel FPGA SDK defines 5 LSU types: two for the Local Memory Interconnect (Constant-Pipelined and Pipelined) and the rest for the GMI: Burst-Coalesced, Prefeching, and Atomic-Pipelined [3]. Each LSU type provides a different maximum bandwidth, being the burst-coalesced with aligned modifier the most efficient type because it maximizes DRAM effective-utilization, which is crucial for memory-bound applications.

The selection of LSU type for each access is defined during RTL generation, which takes a small part of the whole compilation process, and the analysis of the generated RTL provides insights on the DRAM commands performed accessing data. This information merged with a DRAM model [4] enables to accurately estimate the kernel execution time.

Our proposed model stems from a detailed study of the generated RTL, instantiated IPs, and FPGA architecture and condenses the factors causing the memory delay for all possible LSU types, vectorization factors, strides, and DRAM parameters. Thus, for 9 HPC benchmarks, the average estimation error (difference between measured and estimated execution time) of our model is 10%. Please, see [5] for more details.

Table I compares the estimation error of our work with two state-of-the-art models [1], [2] for a Burst Coalesced Aligned μbenchmark (BCA) [3] and a Vector Add application (VA) with 2 DRAM configurations. In all cases, our estimations are more precise, specially for faster memory, because we take into account LSU modifiers, contrary to [1], and do not rely on a predefined memory controller overhead as HLScope+ does [2].

TABLE I
Execution time estimation error for Burst Coalesced Aligned and Vector Add benchmarks on an Stratix 10 GX FPGA

Bench.	DDR4 (MHz)	Wang [%]	HLScope+ [%]	This work [%]
BCA	1866	17.3	12.7	5.6
BCA	2666	69.6	57.8	4.7
VA	1866	19.3	21.0	5.1
VA	2666	67.9	63.3	1.0

To conclude, our model estimates the execution time of memory-bound applications with an average error of 10%. Since projections state a 7% of annual memory-bandwidth increase against a 48% of annual logic resources in new FPGAs, which, in turn, will result in an increase in memory bounded applications, upcoming FPGA systems will require faster and more accurate behaviour models in order to ease programmer's labour.

Acknowledgment

This work was supported by grants TIN2016-76635-C2-1-R, Aragón Government (T58_17R), a Santander-Unizar collaboration grant, and a donation by Intel.

References

[1] Z. Wang, B. He, W. Zhang, and S. Jiang, "A performance analysis framework for optimizing opencl applications on fpgas," in *HPCA*, 2016.
[2] Y. K. Choi, P. Zhang, P. Li, and J. Cong, "HLScope+: Fast and accurate performance estimation for FPGA HLS," in *ICCAD*, 2017.
[3] Intel, "Intel FPGA SDK for OpenCL," 2018.
[4] H. Zheng and Z. Zhu, "Power and performance trade-offs in contemporary DRAM system designs for multicore processors," *IEEE Transactions on Computers*, 2010.
[5] M. Dávila, R. Gran, M. Villarroya, and D. Suárez, "Analytical model of memory-bound applications compiled with high level synthesis," 2020. [Online]. Available: https://arxiv.org/abs/2003.13054

2576-2621/20/$31.00 ©2020 IEEE
DOI 10.1109/FCCM48280.2020.00049