Designing Domain Specific Computing Systems

Anthony M. Cabrera
Roger D. Chamberlain

Dept. of Computer Science and Engineering
McKelvey School of Engineering
Washington University in St. Louis

Designing Domain Specific Computing Systems

Anthony M. Cabrera and Roger D. Chamberlain

Department of Computer Science and Engineering, Washington University in St. Louis, MO, USA

{acabrera, roger}@wustl.edu

Domain specific computing is an idea that has been proposed as a path forward given the slowing of Moore's Law and the breakdown of Dennard scaling [3]. Two fundamental questions include: (1) how does one define a domain; and (2) how does one go about architecting hardware that performs well for that domain? We present our preliminary work towards answering these questions.

Regarding domain definition, we use multi-spectral reuse distance [1] to quantify variations in spatial and temporal locality to identify sub-domains within a previously described domain of applications, using the Data Integration Benchmarking Suite (DIBS) [2] as a case study. Figure 1 shows the result of using k-means clustering, where $k = 2$, of the DIBS applications. The Earth Mover's Distance (EMD) comparisons of the 64 KiB, 4 MiB, and 2 MiB granularities of reuse distance are used as the features to the clustering algorithm.

Fig. 1. k-means clustering of the DIBS applications.

We posit that these clusters might reasonably represent sub-domains of the initial domain, which we use to inform domain specific hardware design targeting the Intel HARPv2 CPU+FPGA platform with the Intel FPGA SDK for OpenCL. Specifically, the cluster that a given application is in will allow us to determine whether it will benefit from a widely vectorized or deeply pipelined implementation. These two qualities reflect the two design paradigm choices, multiple work-item (MWI) and single-work item (SWI) respectively, available when authoring FPGA designs using OpenCL.

To validate this claim, we select the `ebcdic_txt` and `idx_tiff` applications, build SWI and MWI versions for each design paradigm, and perform a design space search using the coarse-grained design knobs for each paradigm. Figures 2 and 3 show the results for the best versions of the

Fig. 2. Design space search for the MWI version of `ebcdic_txt`.

Fig. 3. Design space search for the SWI version of `idx_tiff`.

two applications, and substantiate the result from Figure 1.

The configuration of the best `ebcdic_txt` implementation was setting work group size to 512, number of replicated compute units to 8, and SIMD factor to 16. Its resulting data rate was 3.186 GB/s. For `idx_tiff`, the unroll factor, was set to 64 and achieved a data rate of 0.337 GB/s. While the high level of spatial locality exhibited by `ebcdic_txt` benefited greatly from a widely vectorized implementation, `idx_tiff` drew more benefit from the parallelism extracted from loop unrolling. The fact that this is not immediately obvious just by looking at the respective OpenCL kernel implementations validates this approach.

Acknowledgment

References

[1] A. M. Cabrera, R. D. Chamberlain, and J. C. Beard, "Multi-spectral reuse distance: Divining spatial information from temporal data," in *Proc. of High Performance Extreme Computing Conference (HPEC)*. IEEE, 2019.

[2] A. M. Cabrera *et al.*, "DIBS: A data integration benchmark suite," in *Proc. of ACM/SPEC Int'l Conf. on Performance Engineering Companion*, Apr. 2018, pp. 25–28.

[3] J. Cong, V. Sarkar, G. Reinman, and A. Bui, "Customizable domain-specific computing," *IEEE Design & Test of Computers*, vol. 28, no. 2, pp. 6–15, 2010.

IEEE computer society