Facilitating Easier Access to FPGAs in the Heterogeneous Cloud Ecosystems

Umar Ibrahim Minhas, Roger Woods and Georgios Karakonstantis

***Abstract*—With FPGAs being increasingly integrated into existing software-based heterogeneous cloud environments, novel evaluation mechanisms are required to reveal the energy-performance trade-offs of accelerators (FPGAs, GPUs, etc) using high-level heterogeneous programming environments. For FPGAs, this also requires reconsideration of scheduling policies and reconfiguration methods with an aim to integrate software-based approaches as well as optimizations for broader workload sizes. Proposed considerations are evaluated using various configuration techniques for a number of applications.**

I. Introduction

Cloud computing offers users ubiquitous access to shared pool of resources, through centralized data centres managed by high level software based ecosystems. In recent times, there has been an increased interest to integrate Field Programmable Gate Arrays (FPGAs) [1] but these differ in design effort, reconfiguration and scheduling policies when compared to Graphic Processing Units (GPUs). This paper re-evaluates accelerators in the context of a heterogeneous computing environment of cloud ecosystems.

The first challenge concerns evaluation of achievable gains using a state-of-the-art heterogeneous programming model - OpenCL. Although the traditional approach of programming accelerators using platform specific languages (e.g VHDL, CUDA, etc) [2] is performance-efficient, it is not suited to heterogeneous environments due to lack of portability.

The second challenge investigates reconfiguration overhead on FPGAs [3] incurred due to multi-task execution in cloud systems. Initial work has targeted this via intelligent scheduling targeting techniques such as module reuse [4] and faster reconfiguration memories [5]. However, as per our knowledge, no work has looked into bringing more reconfiguration methods into scheduling model, particularly the more recent software based approaches. This allows easier access of the FPGA as a heterogeneous resource and allows trade-off of reconfiguration overhead with performance and can help achieve better overall execution time.

To summarize, we identify the following contributions for FPGAs' integration in data centres:

- Evaluation of accelerators (FPGA and GPU) using a modern uniform programming model (OpenCL) and similar optimization efforts used for GPUs.
- Inclusion of multiple reconfiguration methods into scheduling model for task size aware selection of the reconfiguration method.

II. Methodology

We address the aforementioned challenges as follows.

Evaluation of Accelerators: We evaluate FPGAs against GPUs using the unified parallel programming model, OpenCL as follows:

- *Step-1:* Identify a set of micro-architectural optimization and design space exploration based on platform-independent parameters of OpenCL that are applicable to both FPGAs and GPUs.
- *Step-2:* In addition to *Step-1*, apply application-specific optimizations on three accelerated computing tasks including matrix-matrix multiply (SGEMM), binomial option pricing (BOP) and finite difference time domain (FDTD) while analyzing the architectural and algorithmic challenges using OpenCL.

We then compare the implementations on state-of-art platforms which use the same technology namely, Altera FPGA (Nallatech 385 with Stratix V A7 chip) and NVIDIA GPU (GTX-980). We consider application specific metrics for throughput and energy efficiency while keeping the design effort constant. We also compare the achieved performance in terms of floating point operations per second against theoretical peak performance on each platform.

Although the GPU outperforms FPGA in terms of throughput (due to larger size of device), the results indicate that Altera FPGA performs better in terms of achieved percentage of device's peak performance (68%) compared to NVIDIA GPU (20%) and achieves better energy efficiency (up to $1.4\times$) for some of the examples without the need for detailed hardware optimisation.

Scheduling Model: The most common reconfiguration methods for FPGAs involve reconfigurion of the whole FPGA with a singe task (SBST) (reconfiguration time in the order of few seconds) and dynamic partial reconfiguration (DPR), in which multiple tasks are loaded into set regions where each region is a subset of whole FPGA (reconfiguration time in the order of few 100ms). We explore two additional reconfiguration methods:

- *Multi-core reconfiguration (MCR)* architecture comprising 64 light-weight programmable cores. The cores allow for rapid run-time reprogramming, remotely over ethernet, in less than 20ms [6].
- *Single bitstream multi-task configuration (SBMT)*, which is similar to DPR. However, instead of dividing FPGA into fixed size regions, the OpenCL synthesizer maps combined source code of all tasks to a single bitstream, allowing for system optimization. The intelligent resources allocation per task workload size (W_S) is such

that all tasks have a similar execution time.

In terms of the highest to lowest reconfiguration time per task, T_r, and throughput, T, the order of the above mentioned methods is SBST, SBMT/DPR and MCR. DPR in our case has the same performance as SBMT. SBMT may have lower T_r than DPR because for SBMT, T_r is proportional to logic used per task while for DPR it is proportional to area of region onto which the task is loaded which will always be greater than the logic used by the task.

The net execution time in a multi-task environment depends on the *reconfiguration time - throughput - task size* trade-off. Using the aforementioned information, our task size-aware scheduler tries to minimize the following sum for *n* tasks:

$$\sum_{i=0}^{n} T_r(i) + T(i) \times W_S(i) \tag{1}$$

III. Results

We consider two real world tasks from financial computation and graph analytics, namely BOP and sparse matrix vector multiplication (SpMV). W_S for both tasks can vary largely in real-time environment with number of options for BOP depending on the stocks being monitored and matrix sizes in SpMV depending on the size of network being evaluated.

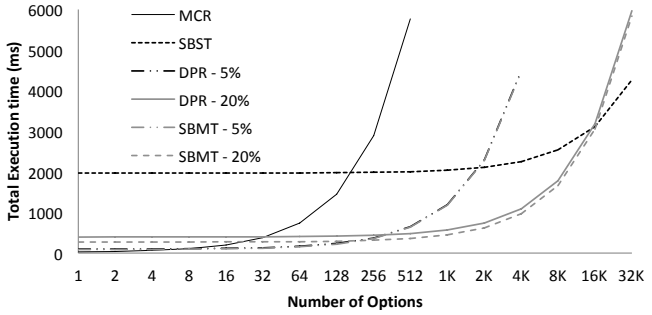

Fig. 1. Total execution time against task size for BOP using various reconfiguration methodologies

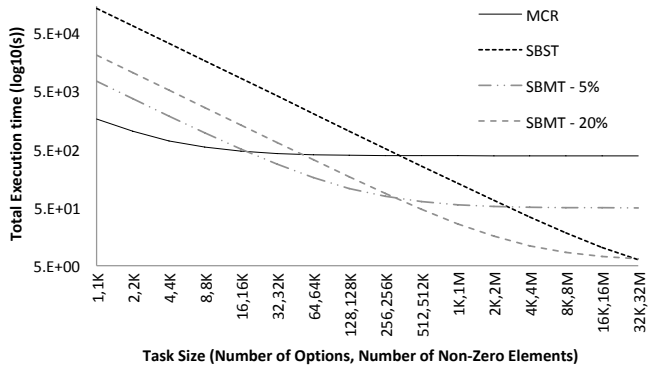

Fig. 2. Multi-task computations using various task sizes

The results for total execution time against workload sizes using various configurations are shown in Fig. 1 for BOP while SpMV followed a similar trend. For DPR and SBMT, we use two different configurations using logic utilization less than or equal to 5% and 20% of total FPGA logic. In an ideal scenario where all tasks have same execution model, the selection of reconfiguration method based on task size can provide up to $86\times$ and $106\times$ improvement over worst case for tasks with a similar execution model as BOP or SpMV, respectively.

In a second analysis, we mimic a multi-task environment by using multiple instances of BOP and SpMV as independent tasks (requiring reconfiguration each time). To evaluate the effect of individual tasks' W_S, we take a total W_S of 32K options and 32M non-zero elements (NNZ) for BOP and SpMV task instances, respectively and vary the W_S per instance. The evaluation of reconfiguration policy against W_S is shown in Fig. 2. It shows that even when computing variable tasks, the intelligent selection of reconfiguration policy can provide up to $79\times$ and $60\times$ improvement over worst case on the lowest and highest end of W_S/instance, respectively.

The results project a case for selection of reconfiguration method based on the dynamic nature of task queues. For a more dynamic queue requiring frequent reconfiguration for tasks with smaller workload sizes, a space sharing model or run-time programmable model is better. However for bigger tasks, whole FPGA reconfiguration may be used.

IV. Future Work

In our evaluation of multiple reconfiguration methods, SBMT performs the best on average because,

- SBMT has lower average T_r than DPR as for SBMT, T_r is proportional to area of task while it is proportional to the area of region onto which the task is loaded in DPR.
- SBMT has the best compute density, i-e total execution time for a set of tasks excluding T_r, as depicted by another set of experiments using four computationally different tasks.

This motivates us to further explore SBMT. In our context of SBMT using OpenCL, we explore the design space using scaled versions by changing OpenCL options such as work-items, loop unrolling, local memory, etc. To speed up this exploration, we propose a static source code analysis framework that can leverage the correlation between multiple versions of same task to achieve accuracy in performance estimates.

References

[1] N. Tarafdar et al. "Designing for FPGAs in the Cloud", in *IEEE Design & Test*, pp. 23-29, 2018.

[2] U. I. Minhas et al., "GPU vs FPGA: A comparative analysis for non-standard precision", in *International Symposium on Applied Reconfigurable Computing*. Springer, pp. 298-305, 2014.

[3] A. Purgato et al., "Resource-efficient scheduling for partially-reconfigurable FPGA-based systems", in *IEEE International Parallel and Distributed Processing Symposium Workshops*, pp. 189-197, 2016.

[4] R. Cattaneo et al., "Para-sched: A reconfiguration-aware scheduler for reconfigurable architectures", in *IEEE International Parallel & Distributed Processing Symposium Workshops*, pp. 243-250, 2014.

[5] H. Liang et al., "Parallelizing hardware tasks on multicontext FPGA with efficient placement and scheduling algorithms", *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, pp. 350-363, 2018.

[6] U. I. Minhas et al, "Nanostreams: A microserver architecture for real-time analytics on fast data streams", *IEEE Trans. on Multi-Scale Computing Systems*, 2017