Optimisation of System Throughput Exploiting Tasks Heterogeneity on Space Shared FPGAs

Optimisation of System Throughput Exploiting Tasks Heterogeneity on Space Shared FPGAs

Umar Ibrahim Minhas, Roger Woods and Georgios Karakonstantis
Queens University Belfast

***Abstract*—There are challenges in optimising system throughput in FPGA-based cloud computing due to mapping constraints resulting in suboptimal space sharing of resources, as the number of tasks grow and become more heterogeneous. This work proposes a methodology for exploring and optimising their resource utilisation. By identifying high-level synthesis parameters for each task, machine learning models and intelligent clustering are then employed to define clusters of tasks which will share the FPGA space. Assuming heterogeneity characterisation of tasks and thus static partitioning of the FPGA, it is ensured that each task in a cluster accommodates other tasks' resource requirements resulting in a higher compute density. Using 11 high performance computing tasks, we achieve an average $3.3\times$ higher system throughput at $2.8\times$ better energy efficiency when compared to existing approaches.**

I. Introduction

With inclusion of Field Programmable Gate Arrays in cloud computing and data centres for high performance computing (HPC) [1], there are opportunities for novel mapping and resource allocation strategies to enhance system throughput, a well-researched topic for software-based systems and accelerators offering largely general-purpose and discrete processor cores. This is particularly true for space sharing tasks for which FPGAs do not offer inherent discrete partitioning of resources, similar to software-programmable systems and vendor tools treat the FPGA as a single sea of gates.

For spatial sharing of resources between multiple tasks, frameworks based on partial reconfigurable regions (PRR) [1] support static partitioning of the FPGA space into fixed rectangular regions. PRRs are designed to be largely homogeneous to fit a range of off-line mapped tasks on modern spatially diverse and unsymmetrical tiled-based FPGAs. Mapping of heterogeneous and independent HPC tasks with variable resource requirements (i.e. memory, compute, logic, bandwidth) complex custom designed hardware and I/O on PRR typically results in lower compute density, namely underutilization of FPGA resources by up to 40% [1] and thus lower throughput.

This can be addressed by extending custom partitioning and task-specific resources mapping capabilities of FPGA to space shared tasks. Although static custom mapping has always been supported by tools, to the author's knowledge, its application to space shared HPC tasks has not been explored before. In addition, in the absence of any intelligent optimisations, the space shared execution may perform the same or even worse than single task processing per FPGA.

In this paper, we attempt to overcome the above challenges by developing a methodology for heterogeneous task-specific resource allocation, taking into consideration the space sharing constraints. Design space exploration (DSE) is applied to heterogeneously characterise tasks and the optimum combination selected for space sharing. Complemented with static partitioning and mapping (SPM) of tasks, the methodology enhances compute density in space for a higher system throughput compared to previous approaches.

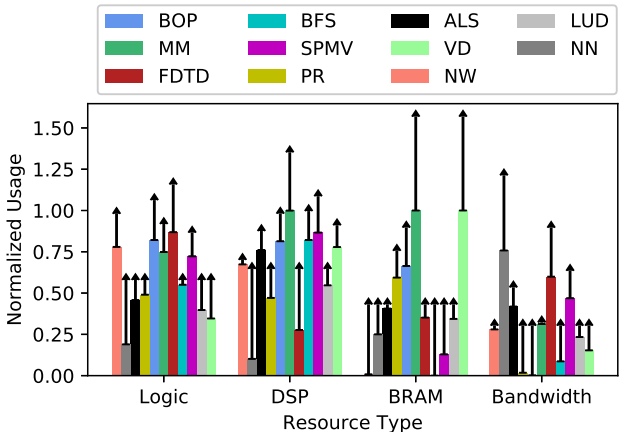

Fig. 1. Normalised resource utilisation by heterogeneous tasks. Error bars show deviation from average

Our contributions can be summarized as follows:

- An innovative methodology that employs DSE, machine learning based characterisation and clustering of heterogeneous tasks to select optimum combination of tasks for space sharing on FPGA. Integrated with SPM of tasks it targets optimisation of compute density in space.
- Validation of the approach using various HPC examples exhibiting varying memory/ratios, including graph analysis, linear algebra, media streaming and data mining - all implemented on a FPGA-based Nallatech 385 platform.
- Evaluation using the *System Throughput (STP)* metric, defined specifically for space-shared multi-task workloads processing [2], showing on average an improvement of $3.3\times$ and $2.8\times$ for throughput and energy efficiency, respectively, compared to PRR.

II. Background and Motivation

Modern cloud computing and data centre workload comprises range of heterogeneous tasks with varying resource requirements [3]. We implemented 11 real HPC tasks including binomial option pricing (BOP), breadth first search (BFS),

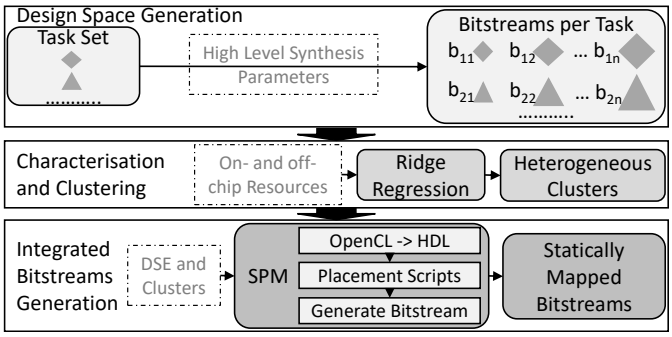

Fig. 2. Off-line model to generate integrated bitstreams

alternative least square (ALS), lower upper decomposition (LUD), matrix-matrix multiply (MM), sparse matrix vector multiplication (SpMV), video downscaling (VD), k-nearest neighbour search (NN), finite difference time domain (FDTD), Page Rank (PR) and Needleman-Wunsch (NW) on Nallatech 385 board comprising Intel Stratix V chip. The study of heterogeneity in Figure 1 for optimised implementations shows that the standard deviation for on-chip heterogeneous resources i.e. logic cells, digital signal processing (DSP) blocks, block random access memory (BRAM) blocks and off-chip bandwidth utilisation is 21%, 26%, 33% and 23% respectively. This highlights a need to consider heterogeneity of tasks while mapping on an FPGA.

Common ways to use FPGAs in data centres include single task configuration per FPGA (STR) [4] and space sharing of FPGA by multiple tasks using PRR [1]. Amongst these, STR provides higher throughput while PRR allows simultaneous multi-task execution and resource allocation variation per task as per workload sizes, particularly for large sized devices.

However, the system performance achieved via PRR is suboptimal as mapping via PRRs is challenging. This because the unsymmetrical spatial distribution of on-chip heterogeneous resources on modern tiled FPGAs and clock regions distribution limit runtime bitstream relocation to homogeneous regions with a step size equal to height of clock regions [5]. Together with placement constraints of static logic for I/O etc, and suboptimal mapping of heterogeneous tasks to homogeneous PRRs, the resource utilisation may be very low [1]. This work looks to achieve higher system performance than STR while allowing space sharing by multiple tasks.

Furthermore, SPM facilitates creation of bitstreams from a high-level, OpenCL, easy adaptation by programmers and flexible integration in software based heterogeneous data centres. It can be complemented with data centre workload characterisation [3] to gain further from off-line modelling.

III. Proposed Methodology

The proposed methodology optimises performance by achieving a higher utilisation of the FPGA resources. Provided with a *task set*, it (Figure 2) first performs DSE to generate multiple hardware *bitstreams per task* that provide a speedup corresponding to the resources and resulting area-throughput curves. The *Characterisation and Clustering* module then uses the *DSE* along with machine learning based regression models to evaluate the relationship between each *on- and off-chip heterogeneous resource* and throughput on FPGA. This characterisation is used to divide all tasks into smaller clusters, such that tasks in a cluster are meant to share the FPGA space at a single instance of time while maximising overall resource usage. The *clusters* are *statically mapped* in the *Integrated Bitstreams Generation* module, using OpenCL front end and placement scripts, to FPGA to achieve high compute density.

We explain various modules of the model below.

A. Design Space Exploration

To achieve this, OpenCL parameters along with general *high level synthesis parameters*, are used to scale the underlying hardware. A task's *kernel* can be scaled over multiple compute units *(CU)* where multiple pipelines can be defined via Single Instruction Multiple Data *(SIMD)* parameter while the key compute intensive loops can be unrolled via an UNROLL *(U)* parameter.

For some tasks, task-specific parameters such as block size or number of rows are used, where these define parallel processing of defined parameter size. For tasks where only a part of code is *unrolled*, dynamic profiling based on an always active counter is used to profile and identify compute intensive segments of the kernels. The counter module is written in VHDL and is passed to OpenCL kernel as a software library via Intel OpenCL Library feature. This is complemented by manual exploration to find the parameters that provide the highest throughput variation per unit area. The intermediate source files generated via OpenCL front end are then further processed using the vendor place and route tools for customised mapping.

B. Characterisation and Clustering

Along with clustering, resource variation per task in a cluster can be required to suit variable workload requirements and hence generic characterisation of DSE of each task, rather than any single bitstream, is required. We perform regression modelling to project the weighted contribution of each type of resource towards throughput. This allows a benefit-based approach where a task which profits best from a higher allocation of a certain resource type is clustered with a task which profits the least. We consider 4 different resources, including *on-chip BRAM, DSPs, logic* and *off-chip bandwidth*.

Amongst low complexity linear regression models, *Ridge Regression* avoids random errors when there is correlation between different variables (such as DSP and BRAM, etc.) and hence is adopted. The normalised values of on- and off-chip heterogeneous resources against maximum available for all bitstreams per task form the independent variables for regression, while the achieved throughput, measured on actual hardware for each bitstream and normalised to maximum achievable for each task, becomes the dependent variable.

Regression provides the significance scores of each type of resource for each task to scale throughput. These, in addition

TABLE I
USE CASES CHARACTERISTICS. WHERE UNSPECIFIED, STEP SIZE IS 2×.

Use Case	Bitstreams Scaling	Speedup
PR	(CU: 1,2,4) × (U: 1, 2, 4)	6×
ALS	(CU: 1, 4) × (U: 1, 4)	2×
BOP	CU1 × (U: 1 - 16); (CU: 3, 4, 5) × U8	21×
BFS	U: 1 - 16	5×
SpMV	U: 1 - 32	190×
FDTD	Block Size: 1 - 16	13×
LUD	CU1 × (U: 1 - 16); (CU: 2, 3)× U16	18×
VD	Parallel Rows: 1 - 32	8×
SGEMM	SIMD1 × (U: 1 - 64); SIMD4 × (U: 32 - 64)	204×
NW	Block Size: 2 - 128	33×
NN	U: 1 - 8	5×

to the normalised bandwidth utilisation for each task's largest bitstream, are then used to *cluster* tasks for space sharing at a single time. We have only used bandwidth, as unlike other on-chip resources, it may become bottlenecked early in DSE for extremely memory intensive tasks and a consistent usage may hide the fact that the task has high bandwidth dependence, during regression modelling. In order to define clusters, each task is first represented in multi-dimensional space where each dimension either represents a regression score of a type of resource or the normalised bandwidth.

We then use a custom designed optimisation function that uses a set number of iterations to find the best solution. Each iteration randomly selects first task in the cluster for each new cluster and then searches for other tasks such that all tasks in cluster have maximum distance, and thus heterogeneity between themselves in the multi-dimensional space. The number of tasks in a cluster is defined manually by the system designer based on the size of the device. The sum of mutual distances between tasks is used as a score for the cluster while the sum of all cluster scores defines iteration's score. The iteration with the highest score is used as solution.

C. Integrated Bistreams Generation

As mentioned in Section II, a PRR system is limited to homogeneous regions, thus limiting the optimisation via clustering to only off-chip memory bandwidth. To maximise on-chip resource utilisation, we make use of *static partitioning* of FPGA resources and *mapping* of heterogeneous tasks to generate a single multi-task bitstream.

SPM uses *DSE and clustering* to provide an optimised mapping such that tasks are only allocated resources as per their specific heterogeneous requirements rather than a homogeneous and fixed size rectangular region. This is achieved by using OpenCL front end to create HDL modules. Scripts are then used to map task modules to the corresponding PRR while for SPM, area constraints are set to *all available area for task logic* and an integrated bitstream is generated using place and route tools. We do not implement any custom mapping optimisations per task for SPM on top of those inherently provided by the vendor tools for multiple modules. Both PRR and SPM modules can be partially reconfigured independent of the static logic.

IV. TEST ENVIRONMENT

Next, we describe the test environment which has been used to provide a comprehensive evaluation of the proposed approach.

A. Use Cases

We have considered 11 HPC tasks belonging to various computing and application domains. The tasks represent a range of computing characteristics including memory, compute and logic bound. The scale as well as parameters used in DSE for each task are summarised in Table I.

B. FPGA and Host Platform

The high level DSE is performed via Intel OpenCL SDK for FPGAs v16.1 while constrained placement is achieved using Quartus Prime v16.1. The hardware profiling is performed on Nallatech 385 with a Stratix V FPGA chip. The power is measured using on-board power sensors accessible via the memory mapped device layer while the bandwidth is measured using Intel FPGA Dynamic Profiler for OpenCL GUI. We use OpenCL runtime and independent *command queues* for each task, allowing parallel execution.

C. System Throughput Metric

Assessing the system performance of a multi-task workload running in parallel on a single processing unit is challenging as the system total execution time may be influenced by workload sizes of individual tasks and corresponding speedups. To evaluate the performance of various approaches in a multi-task environment, we use the STP metric [2] as defined by:

$$STP = \sum_{i=1}^{n} NP_i = \sum_{i=1}^{n} \frac{C_i^{SP}}{C_i^{MP}} \quad (1)$$

where NP is each task's normalised progress defined by the number of clock cycles, C_i^{SP}, it takes when using all resources in STR mode compared to multi-task mode, C_i^{MP}, when it is sharing the space with other tasks. Here, n defines the number of tasks sharing the FPGA. The metric encompasses various system design parameters and effectively provides throughput relative to baseline of single task processing per FPGA, which has STP value of 1.

V. RESULTS AND ANALYSIS

We first describe the scope of DSE before moving to clustering and comparison of performance.

A. Design Space Exploration

The DSE, as summarised in Table I provides real area numbers as well as variation in throughput against resource utilisation for fair evaluation and comparison of spatial mapping and partitioning schemes. To measure speedup, the baseline T_{exec}, corresponding to lowest area bitstream, is defined by the serial pipelined benchmark implementation of task. The maximum throughput is defined by the largest bitstream, limited by the FPGA resources. We have generated $4-9$ bitstreams per task where each bitstream represents a point on area-throughput

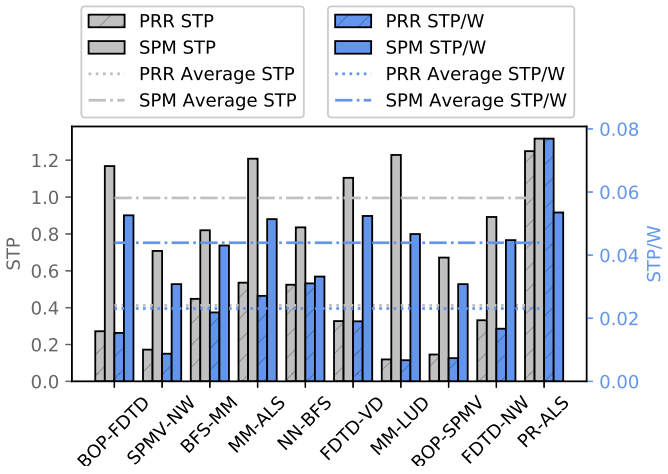

Fig. 3. STP and STP/W for SPM and PRR using unoptimised clustering

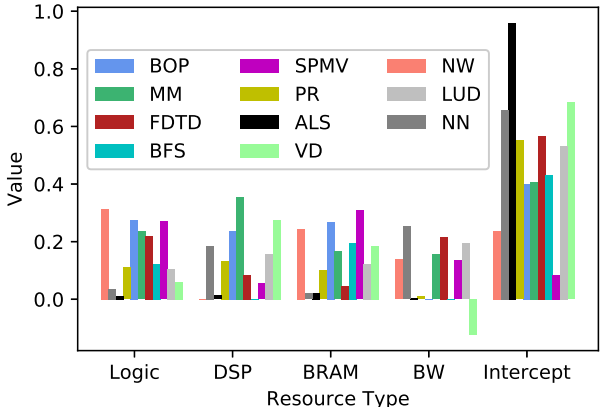

Fig. 4. Ridge Regression models for HPC tasks

curve and maximum speedup ranges from $2\times$ to $204\times$ for tasks. Tasks with higher speedup per unit resource suffer more from suboptimal resource utilisation.

B. PRR vs SPM

Due to the size of the device under test (DUT) and area available for PRRs, the space sharing is limited to a cluster of 2 tasks. For SPM, up to 3 tasks can be fit at one time. However, the STP normally decreases with more than 2 tasks per cluster because of small size of the DUT and to keep the comparison fair, we limited the cluster size to 2 tasks for SPM as well. Using the DSE, the largest bitstreams per task are selected within the area constraints of the PRR and SPM.

We generate 10 random clusters of tasks for baseline analysis of PRR and SPM without any optimisations. For evaluation of STP, the data sizes for tasks in a cluster are chosen such that both tasks have similar T_{exec}. The results in Figure 3, show that SPM can provide $2.4\times$ higher STP on average on the basis of higher compute density in space but consuming higher power. Figure 3 shows that although the gain is lesser, SPM provides $1.9\times$ better energy efficiency (STP/W) on average compared to PRR.

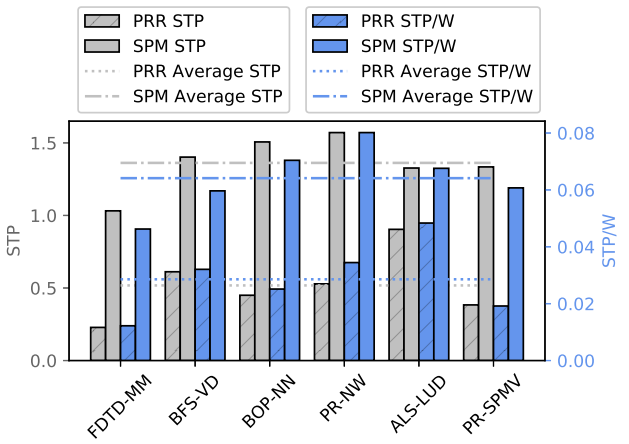

Fig. 5. STP and STP/W for SPM and PRR using optimised clustering

C. Clustering

Next we present the gains made by intelligent clustering of tasks. Figure 4 shows the contribution of resources towards tasks' throughput using the DSE and Ridge Regression. As can be seen, the weighted contribution of each heterogeneous resource is different from the resource utilisation numbers provided in Figure 1. The intercept relates to the baseline throughput. Using the Regression models and the intelligent clustering, we created a set of 6 clusters.

Figure 5 shows gains for both PRR and STP, using similar T_{exec} for tasks in cluster, compared to Figure 3. For PRR, the STP increase of $1.2\times$ is mostly due to optimisation of off-chip memory bandwidth utilisation. For SPM, the STP increase of $1.4\times$ correspond to both on-chip and off-chip resource optimisation. The gain of $3.3\times$ and $2.8\times$ for throughput (STP) and energy efficiency (STP/W), respectively, between SPM in Figure 5 and PRR in Figure 3 is the maximum achievable via proposed optimisations compared to PRR while providing $1.3\times$ higher throughput compared to STR.

VI. Conclusion

This work proposes an innovative methodology to achieve higher system throughput for space sharing FPGAs. The approach proposes characterisation and clustering of tasks based on heterogeneities in resource usage, complemented by SPM of tasks to maximise resource utilisation. The achieved performance is higher than existing STR and PRR approaches while allowing resource allocation variation per task.

References

[1] A. Vaishnav, et al., "Resource elastic virtualization for FPGAs using OpenCL" In *FPL*, 2018.
[2] S. Eyerman and L. Eeckhout, "System-level performance metrics for multiprogram workloads," *IEEE Micro*, vol. 28, 2008.
[3] X. Chang, et al., "Effective modeling approach for IAAS data center performance analysis under heterogeneous workload," *IEEE TCC*, 2016.
[4] U. I. Minhas, et al., "Exploring functional acceleration of OpenCL on FPGAs and GPUs through platform-independent optimizations." in *ARC*, Springer, 2018.
[5] K. D. Pham, et al., "Bitman: A tool and api for fpga bitstream manipulations," in *DATE*, IEEE, 2017, pp. 894–897.