Convex Models for Accelerating Applications on FPGA-Based Clusters

Qiang Liu, Tim Todman, Kuen Hung Tsoi and Wayne Luk

Computing Department, Imperial College
London, SW7 2AZ, UK
{qiang.liu2, timothy.todman, khtsoi, w.luk}@imperial.ac.uk

***Abstract*—We propose a new approach, based on a set of convex models, to accelerate an application using a computing cluster which contains field-programmable gate arrays (FPGAs). The computationally-intensive tasks of the application are mapped onto multiple acceleration nodes, and the datapaths on the nodes are customized around the tasks during compilation. We propose models for computation and communication on the FPGA-based cluster, and formulate the design problem as a convex non-linear optimization problem allowing design exploration. We evaluate our approach on a cluster with 16 nodes for Monte Carlo simulation, resulting in a design 690 times faster than a software implementation.**

I. INTRODUCTION

As the complexity and scale of current application algorithms increase, a single-node computing engine may not provide sufficient computational power. A popular way to get more computing power from existing machines is to build them into a cluster, where computing nodes are connected by a network, which leaves the problem: how do we map an application onto the cluster?

In this paper, we propose a compilation approach for accelerating applications on a cluster with multiple computing nodes. Specifically, the computationally-intensive tasks of the application are mapped onto multiple acceleration nodes, such that the customized datapaths on the nodes are generated around the tasks, on a cluster based on field-programmable gate array (FPGA) technology.

The aim of this work is to accelerate computations by efficiently exploiting currently available parallel computational hardware, *e.g.* FPGAs. We especially wish to provide Computer-Aided Design (CAD) optimization techniques to improve productivity of system designers. We propose models for computation and communication on FPGA-based clusters, to allow design exploration and to predict system performance at an early design stage. Given a target FPGA-based cluster with a fixed number of nodes and limited resources on each node, the approach determines design parameters, including the number of segments of a task executing in parallel, how to map these segments onto the nodes, and the datapath and resource usage on each node, for efficient execution of different applications.

Research on task mapping on different platforms has been a hot topic. In previous work, each task may be a statement, an instruction or a function, and heuristics [1] or integer linear programming (ILP) [2] are used to solve the mapping problem. The main differences of our approach from previous work in this area are that: a) the target task contains loop nests and is computationally-intensive; b) a task can be further partitioned into parallel segments to improve parallelism, c) the computation involved in each task and the communication between tasks are modeled; and d) task mapping and datapath customization are determined at the same time by solving a convex non-linear optimization problem, leading to the globally optimal design within the design space.

Our approach could be integrated into a design toolchain [3] to automate designs on a cluster with heterogeneous computation nodes. Designers may also use the models proposed in this paper on their own for design-space exploration, avoiding the need to do this manually whilst allowing a hand-optimized implementation.

The contributions of this paper are thus:

- an approach for concurrently mapping and customizing computation tasks onto an FPGA-based cluster;
- models for computation and communication, and convexity transformations for a convex nonlinear optimization problem, leading to optimized designs; and
- evaluation of the proposed approach for Monte Carlo simulation on a cluster with 16 nodes, each node consisting of a central processing unit (CPU) and an FPGA, with results showing that our method can find optimized solutions within the design space.

The rest of the paper is organized as follows. Section II surveys related work. Section III shows our approach, how we develop the models for computation and communication, and how we apply convexity transformations to enable formulating design space exploration as a convex optimization problem. Section IV shows results and evaluation, while section V concludes.

II. RELATED WORK

Several FPGA-based computing clusters have been developed, including the commercial PICO [4] and research efforts such as Maxwell [5] and Axel [6]. Our work targets different tasks to different FPGAs, but could be adapted to other clusters with a single configuration model, such as Cube [7].

Several researchers have studied the problem of mapping task graphs to multiple FPGAs, either manually or automatically. These efforts vary in several ways: size of task, whether tasks can be subpartitioned, and optimization method.

978-1-4244-8983-1/10/$26.00 ©2010 IEEE

Hashemi and Ghiasi [8] assign task graphs to soft dual-processor platforms using heuristics. Lam et al. [1] use tabu search to map and schedule task graphs to heterogenous systems. These approaches should yield solutions faster than our approach, but our GP models should find solutions that their heuristic approaches do not consider. However, overall implementation time will be dominated by place and route for all methods.

Much work has been done on scheduling and mapping tasks for non-reconfigurable systems. For example, Bednarski and Kessler [2] give an ILP formulation combining several steps including instruction scheduling for VLIW systems. Reconfigurable systems, however, have a much larger and complex design space, which sometimes cannot be formulated in an ILP. This paper proposes a convex non-linear formulation for the problem.

III. Proposed Approach

In our approach, each computation task contains a loop nest, so potentially both loop-level parallelism and instruction-level parallelism can be exploited. If the loop nest in a task is parallelizable, task execution follows a SPMD (Single Program Multiple Data) model [9], where the same program executes on different data sets in parallel without communication, and a barrier point ensures all processes complete so that intermediate results from different processes can be collected and the final result can be generated. Our approach exploits loop-level pipelining for each process to customize the corresponding datapath, and also allocates resources available on each node between parallel processes.

The execution time of a task on a cluster includes two parts: computation time t_{comp} and communication time t_{comm}. Different cluster structures may need different computation and communication models. Our target cluster uses an FPGA on each node, so we derive an execution time model for FPGA-based computation. Our approach is modular: it can apply to other computing engines (*e.g.* GPGPUs) by replacing the execution model. Previous work derives a similar execution time model [10] for mapping an affine loop nest onto an FPGA. However, mapping loops onto a cluster must consider different design aspects.

A. Formulation

For a cluster, the intra-node and inter-node communication costs have to be considered. For simplicity, we only present the model for mapping a single-level loop; the multi-level case can be similarly formulated.

$$t_{comp} = u \times ii + D_{mem} + \sum_{i=1}^{I} d_i \times D_i + MR \times D_r \quad (1)$$

$$t_{comm} = Comm_{intra} + Comm_{inter} \quad (2)$$

$$Comm_{intra} = L/n \times ID/C_{intra} + OD/C_{intra} \quad (3)$$

$$Comm_{inter} = \begin{cases} 0 & \text{if } n = 1 \\ n \times OD/C_{inter} & \text{if } n > 1 \end{cases} \quad (4)$$

In the computation time model (1), a loop with L iterations is mapped onto n cluster nodes and each node processes m segments in parallel. Each segment has $u = \lceil \frac{L}{n \times m} \rceil$ loop iterations which are pipelined with the initiation interval ii. D_{mem} is the cost for reading/writing a datum from/to the accelerator local memories. The computations inside the loop body are scheduled in I levels following the as-late-as-possible scheduling scheme, and each computation level could take $d_i (d_i \geq 1)$ stages, each taking D_i cycles. MR is the number of computation stages for merging (reducing) results computed in parallel on one node. Each stage has a delay of D_r cycles.

The communication time (2) is composed of two parts: the intra-node and the inter-node communication costs. The intra-node communication (3) includes inputting/outputing data from/to controller to/from accelerator. ID is the number of input data required in one loop iteration on the accelerator. OD is the number of results generated on the accelerator. C_{intra} is the intra-node communication bandwidth. The intra-node communication can overlap with computations using FIFOs if data in the computations are accessed in a constant stride. Eq. (4) is the inter-node communication model. There is no inter-node communication if the task is only mapped onto one node; otherwise the results generated on each node must be transferred to the other nodes to form final results. C_{inter} is the inter-node communication bandwidth.

There are two main cases for MR and OD in many algorithms as follows. Analyzing the input algorithms can identify these two cases. We distinguish the two cases to simplify the execution time models and allow the convexity transformations described afterwards.

Case 1: the final result is generated by accumulation in the original algorithms, *e.g.* using '+=', '-=', 'min' and 'max'. In this case, each node produces an intermediate value for the output object and OD equals to the number of output objects OB. If the computations on each node are partitioned into m parallel segments, the intermediate result for each output object can be generated by tree reduction, and thus $MR = \lceil \log_2 m \rceil$.

Case 2: each node produces a set of elements of an output object, *e.g.* $A[i] = B[i] \times C[i]$. In this case, each node generates R/n values for the output object, where R is the total number of output data. Therefore, $OD = OB \times R/n$ and $MR = 0$.

With this execution time model, we formulate the cluster resource constraints. Eqs (5)–(8) give constraints on the loop pipelining initiation interval variable ii. Eqs. (5) and (6) give the computation resource constraints, where W_f is the number of resources f required in one loop iteration, x_f is number of resources f allocated and R_{if} is the number of resources f required in computation level i. It is the variable x_f that determines resource allocation between the parallel processes. Eq. (7) is the data dependence constraint, where $RecII$ is the dependence distance. Eq. (8) is the memory bandwidth constraint, where BW is the bandwidth requirement in one loop iteration and M_b is the available memory bandwidth. Eqs (9) and (10) are the constraints on the total available computational resources and the number of cluster nodes.

Eq. (11) shows the lower and upper bounds of the number of parallel partitions of a loop with L iterations.

$$W_f \times x_f^{-1} \times ii^{-1} \leq 1 \tag{5}$$

$$R_{if} \times x_f^{-1} \times d_i^{-1} \leq 1, 1 \leq i \leq I \tag{6}$$

$$RecII \times ii^{-1} \leq 1 \tag{7}$$

$$BW \times m \times M_b^{-1} \times ii^{-1} \leq 1 \tag{8}$$

$$m \times x_f \leq Res_f \tag{9}$$

$$1 \leq n \leq N \tag{10}$$

$$1 \leq n \times m \leq L \tag{11}$$

We now have models for mapping a task with a loop nest onto an FPGA-based cluster. The optimization problem is to minimize $t_{exe} = t_{comp} + t_{comm}$, subject to Eqs.(5)–(11), where integer variables m, n and ii are unknown. Once we obtain the solution to the optimization problem, we will know how many parallel segments a task can be partitioned, how these segments are mapped onto the cluster, and how each segment executes. However, the current formulation is not a convex optimization problem, due to conditional branches in the inter-node communication expression (4) and the logarithm in the MR expression. This would make the optimization problem hard to solve globally.

B. Convexity transform

We therefore perform *convexity transformations* on the two expressions. We observe that the optimization problem described above is similar to the geometric programming problem [11], which is an optimization problem where the objective function and the inequality constraints are posynomials with positive coefficients and all variables are positive. The geometric programming problem can be easily transformed into a convex form.

The logarithm in the $MR = \lceil \log 2m \rceil$ expression can be approximated by a posynomial, $\sum_i m^{a_i}$ where $0 < a_i < 1$. The number of terms of the posynomial and the values of a_i are determined by the range of m.

We reformulate the inter-node communication model. The difference is that when $n = 1$ the communication cost is 1 cycle rather than 0. This small introduced error can be ignored in practice.

$$Case1: \ Comm_{inter} = \begin{cases} 1 & \text{if } n = 1 \\ n \times OB/C_{inter} & \text{if } n > 1 \end{cases} \tag{12}$$

$$Case2: \ Comm_{inter} = \begin{cases} 1 & \text{if } n = 1 \\ R \times OB/C_{inter} & \text{if } n > 1 \end{cases} \tag{13}$$

Then, we introduce a new binary variable n' and let

$$n' \geq \frac{a \times n^2 + b \times n + c}{(a + b + c) \times n}, 1 \leq n \leq N \tag{14}$$

```
for (i = 1; i <= numTraj; i++)
  for (j=1; j <= numSum-1; j++)
  {
        x1 = C2 * randNum;
        x2 = C1 + x1;
        x3 = exp(x2);
        priceTraj[i] = priceTraj[i] * x3;
        priceSum[i] = priceSum[i] + priceTraj[i];
  }
```

Fig. 1. MCS code segment.

where parameters (a, b, c) are a solution of the following inequality system given N.

$$N \times (N-2) \times a - N \times b - (2N-1) \times c \leq 0$$
$$a \geq c, a \geq 1, b \geq 0, c \geq 0 \tag{15}$$

In this way,

$$n' = \begin{cases} 1 & \text{if } n = 1 \\ 2 & \text{if } n > 1 \end{cases} \tag{16}$$

and the inter-node communication model becomes monomials.

$$Comm_{inter} = \begin{cases} n \times n'^{\log_2 (OB/C_{inter})} & \text{if Case1} \\ n'^{\log_2 (R \times OB/C_{inter})} & \text{if Case2} \end{cases} \tag{17}$$

Using this inter-node communication model and the logarithm approximation, the optimization problem for mapping a task onto the cluster is transformed to the integer geometric programming problem, solvable by existing solvers for a globally optimal solution. Capital letters in the above formulations denote compile-time constant parameters. We show some experimental results from using these models to map a complex task onto the cluster in Section IV.

IV. Results

We validate our proposed approach by applying it to Monte Carlo simulation (MCS) [12]. We apply the optimization models in section III to explore the design space of mapping this application onto a cluster with 16 nodes, and to show how the number of nodes of a cluster impact on performance. Each node has an AMD Phenom CPU and a Xilinx Virtex-5 LX330T FPGA. The CPU has 4GBytes system memory and the FPGA has 1GBytes local memory in four banks. The intra-node communication is a PCI express (PCIe) bus. Through the PCIe bus, the CPU can use DMA to transfer data between the system memory and the FPGA local memory at a speed of $C_{intraw} = 170$MBps for write and $C_{intrar} = 230$MBps for read. Inter-node communication is via Gigabit Ethernet. CPUs on different nodes communicate with each other by broadcast and point-to-point on Ethernet at a maximum speed of $C_{inter} = 100$MBps. The computation resource constraints in each node are the number of DSP blocks and slices on a Virtex-5 LX330T FPGA.

Monte Carlo simulation (MCS) has wide applications in finance. Asian Option prices depend on the average price observed over a number of discrete dates in the future; the

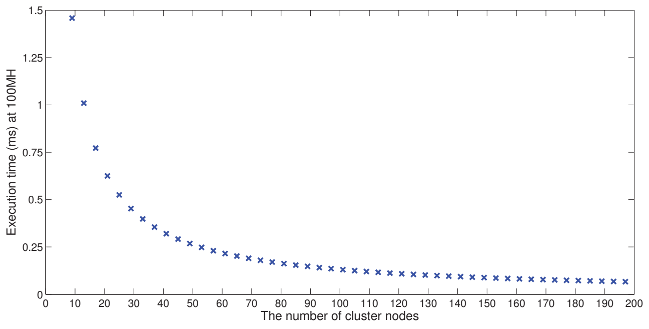

Fig. 2. The number of nodes vs speed in MCS.

mathematical model has no closed solution [13]. Therefore, MCS performs many simulation trajectories to estimate the price. This simulation contains two loops: the outer loop controls the number of simulation trajectories, and the inner loop sets several observation points during a contract period, as shown in Fig. 1. Each simulation trajectory receives random numbers and computes the price independently of other trajectories. Therefore, there is no communication during the MCS. As a result, we see that in Fig. 2 the execution time of MCS keeps decreasing as the number of nodes increases, and the speedup is more obvious when the number of nodes is less than 50 for the size of the input data.

We accelerate MCS on the target cluster following the design proposed by our approach and results are shown in Table I. In this experiment, the simulation simulates 10^6 trajectories and each trajectory sets 12 observation points. The execution time predicted by our model is 0.82 ms, while the real execution time obtained in the experiment is 0.87 ms. The relative error is 5.75%. When compared to the implementation on a dual-core CPU, our design improves the speed by about 690 times. This significant speedup is due to the fact that the MCS of Asian Option pricing involves no intra-node and no inter-node communication. Moreover, given a speed requirement, our approach is able to find the appropriate number of cluster nodes from Fig. 2.

TABLE I
MCS OF ASIAN OPTION PRICING.

Design	**Parallel segments**	**DSP blocks**	**Slices**	**Freq (MHz)**	**Time (sec)**
This paper	96	192 (100%)	42124 (81%)	100	0.0008
2-core CPU	8 threads			2400	0.6

V. CONCLUSION

We propose an approach to map computationally-intensive tasks onto an FPGA-based computing cluster. Task mapping and datapath customization are handled concurrently. We derive the models of computation and communication of tasks on an FPGA-based cluster, apply convexity transformations, and formulate design space exploration as a convex optimization problem while respecting the system structure and resources on the target cluster. The problem is investigated in depth and is transformed in a novel way into a geometric programming problem. Results for Monte Carlo simulation on a 16-node cluster show that the method allows the design space to be explored, and the optimized design generated. Future work includes extending the model to other kinds of accelerators and to support the mapping of complex applications.

Acknowledgement. The support of an Imperial College London Research Excellence Award, the FP7 REFELCT (Rendering FPGAs for Multi-Core Embedded Computing) Project, the UK Engineering and Physical Sciences Research Council, HiPEAC, Alpha Data, nVidia, and Xilinx is gratefully acknowledged.

REFERENCES

[1] Y. M. Lam, J. G. F. Coutinho, W. Luk, and P. H. W. Leong, "Mapping and scheduling with task clustering for heterogenous computing systems," in *Proc. Int. Conf. on Field Programmable Logic*. IEEE, 2008.

[2] A. Bednarski and C. W. Kessler, "Optimal integrated VLIW code generation with integer linear programming," in *Proc. Euro-Par 2006*. Springer LNCS 4128, Aug. 2006, pp. 461–472.

[3] W. Luk, J. Coutinho, T. Todman, Y. Lam, W. Osborne, K. Susanto, Q. Liu, and W. Wong, "A high-level compilation toolchain for heterogeneous systems," in *Proc. Int. Conf. on SOC*, 2009, pp. 9–18.

[4] G. Edvenson and M. Hur, "Accelerating bioinformatics searching and dot plotting using a scalable FPGA cluster," *A Pico Computing Life Sciences White Paper*, November 2009.

[5] R. Baxter, S. Booth, M. Bull, G. Cawood, J. Perry, M. Parsons, A. T. Alan Simpson, A. McCormick, G. Smart, R. Smart, A. Cantle, R. Chamberlain, and G. Genest, "Maxwell – a 64 FPGA supercomputer," *Engineering Letters*, vol. 16, no. 3, 2008.

[6] K. H. Tsoi and W. Luk, "Axel: a heterogeneous cluster with FPGAs and GPUs," in *Proc. Int. Conf. on FPGA*, 2010, pp. 115–124.

[7] O. Mencer, K. H. Tsoi, S. Craimer, T. Todman, W. Luk, M. Y. Wong, and P. H. W. Leong, "Cube: A 512-FPGA cluster," in *Proc. IEEE Southern Programmable Logic Conference (SPL 2009)*. IEEE, 2009.

[8] M. Hashemi and S. Ghiasi, "Versatile task assignment for heterogenous soft dual-processor platforms," *IEEE transactions on CAD of ICs and systems*, vol. 29, no. 3, March 2010.

[9] H. Zima and B. Chapman, *Supercompilers for Parallel and Vector computers*. ACM press, 1990.

[10] Q. Liu, T. Todman, W. Luk, and G. Constantinides, "Automatic optimisation of MapReduce designs by geometric programming," in *Proc. Int. Conf. on FPT*, 2009, pp. 215–222.

[11] S. Boyd and L. Vandenberghe, *Convex optimization*. Singapore: Cambridge University Press, 2004.

[12] J. Makino, "The GRAPE project," *Computing in Science and Engineering.*, vol. 8, no. 1, pp. 30–40, 2006.

[13] http://www.interactivesupercomputing.com/success/pdf/caseStudy_financialmodeling.pdf, "Financial Modeling – Monte Carlo Analysis," accessed 2009.