Efficient implementation of computationally intensive algorithms on parallel computing platforms

Csaba Nemes

Theses of the PhD dissertation

Pázmány Péter Catholic University
Faculty Of Information Technology And Bionics
Multidisciplinary Technical Sciences Doctoral School

Scientific advisor:
Zoltán Nagy, PhD
Consultant:
Péter Szolgay, DSc

Budapest, 2014

1 Introduction and Research Aims

Computationally intensive simulations of physical phenomena are inevitable to solve engineering and scientific problems. Simulations are used to test product designs without fabrication or to predict properties of new physical or chemical systems. Computer engineering has long since been dealing with the acceleration of simulations to decrease the development time of new products or to improve the resulting quality by expanding the design space. Since clock frequency of processors reached the physical limits caused by power dissipation, processor designers are focusing on multi- and many-core architectures to keep up with the predictions of Moore's law. The goal of high-performance computing is to answer how to exploit the computing potential of these novel parallel architectures, such as GPU (*Graphical Processing Uni*t) and FPGA (*Field Programmable Gate Array*), to solve computationally intensive problems.

During my research I investigated the acceleration of two specific problems with the following questions in mind: What is the best architecture for the given application? How can the implementation methodology be improved? What performance can be reached, and what are the implementation tradeoffs in terms of speed, power and area?

The first problem I investigated was the numerical solution of *partial differential equations* (PDEs) on FPGAs. Nagy et al. demonstrated that the FPGA implementation of the emulated digital CNN-UM (*Cellular Neural Network - Universal Machine*) can be generalized to efficiently simulate various types of conservation laws via *finite volume method* (FVM) discretization of the given PDE with the Euler explicit scheme [9]. The mathematical expression (*numerical scheme*) which has to be evaluated for each cell in each iteration can be represented with a synchronous data-flow graph. As the goal is to design a high-performance pipelined *arithmetic unit* (AU), which can operate at high frequency, each mathematical operation, i.e., node of the graph, is implemented with a dedicated *floating-point unit* (FPU). On recent high-end FPGAs, several floating-point units can be realized, which can operate at high frequency, however, the global control signals connected to each floating-point unit slow down the operating frequency of the rest of the circuit.

My research goal was to develop a novel design methodology which constructs high-performance, locally controlled AUs from synchronous data-flow graphs. My questions were the following: How shall I partition the data-flow graph to obtain clusters which can be controlled efficiently? How to control the clusters and how to connect them to avoid synchronization

problems? What is the price of the improved frequency in terms of speed, power and area? Finally, how to automate the generation process of the AUs to drastically decrease the development time of new numerical simulations?

The second problem I investigated was the *Density Matrix Renormalization Group* (DMRG) algorithm [10]. The algorithm is a variational numerical approach, which has become one of the leading algorithms to study the low energy physics of strongly correlated systems exhibiting chain-like entanglement structure [11]. The algorithm was developed to balance the size of the effective Hilbert space and the accuracy of the simulation, and its runtime is dominated by the iterative diagonalization of the Hamilton operator. As the most time-consuming step of the algorithm, which is the projection operation of the diagonalization, can be expressed as a sequence of dense matrix operations, the DMRG is an appealing candidate to fully utilize the computing power residing in novel parallel architectures.

As the algorithm had not been accelerated on parallel architectures (to the best of my knowledge), my research goal was to investigate on which architecture the algorithm can be implemented most efficiently. My objective was to give a high-performance, parallel and flexible implementation on the selected architecture, which can deal with wide range of DMRG configurations.

2 Methods of Investigation

I have implemented a framework in C/C++ to automatize the generation of the AU from a textual or a SystemC [12] description of the numerical scheme. In the framework, the scheme is represented as a graph, which enabled me to design and evaluate different partitioning algorithms. Later, I developed a novel graph representation which can incorporate both partitioning and placement information. For algorithm and representation design, I relied on the literature available on circuit placement [13, 14], graph partitioning [15] and graph visualization [16].

In the framework, the AU is generated in VHDL language, and the results are ready to use with the standard FPGA synthesis tools. The control logic and the mixer units, which supply the AU with data, were also developed in VHDL with Xilinx ISE Design Suite 13.1. Floating-point units used in the AU were implemented via Xilinx IP Cores. The placement constraints were defined in the Xilinx's *user constraint file* (UCF) and submitted at the place-and-route phase. Manual tuning of the placement constraints was carried out with the Xilinx PlanAhead editor.

As a consequence of probabilistic heuristics used in the Xilinx's place-and-route algorithm, the operating frequency of the final circuit is sensitive to the input seed parameters. During the performance evaluation, the generated circuit was implemented with wide range of seed parameters, and the highest frequency was selected.

The CFD applications which were used to demonstrate the framework were implemented on a Xilinx Virtex-6 SX475T FPGA with speed grade -1. The chip, containing 74400 slices, 2016 dedicated multiplier blocks (DSP48E1), and 38304 Kb on-chip memory (BRAM), is one of the largest Virtex-6 FPGA offering the highest ratio of DSP and memory resources.

The DMRG algorithm was implemented in C/C++ and can be compiled in a CPU-only and a hybrid CPU-GPU mode. In the CPU-only mode, all the basic linear algebra subroutines (BLAS) are accelerated with the Intel MKL library, while in the hybrid mode, some of the operations are executed on GPU using the CUDA 5.0 environment. Matrix-matrix multiplications related to the projection operation of the Davidson algorithm are implemented via the NVidia CuBlas library, while for asymmetric matrix-vector multiplications I designed a new CUDA kernel.

The performance of the hybrid implementation was measured both on a mid-range (Intel Core-i7 2600 3.4 GHz CPU + NVidia GTX 570 GPU) and on a high-end configuration (Intel Xeon E5-2640 2.5 GHz CPU + NVidia K20 GPU) and compared to the CPU-only performance. The main parameters of the GPU cards are summarized in Table 1. The comparison was performed in case of two different models (Heisenberg and Hubbard), and similar speed-ups were observed for both models.

Model name	Number of CUDA cores	Clock frequency	Device memory	Compute capability
NVidia GTX 570	480	1464 MHz	1280 MB	2.0
NVidia K20	2496	706 MHz	5120 MB	3.5

Table 1: Main parameters of the utilized graphical processing units

3 New Scientific Results

The statements of the dissertation are grouped into two categories: the first group deals with the construction of locally controlled arithmetic units from synchronous data-flow graphs, while the second group is focusing on

the first implementation of the DMRG algorithm on modern parallel architectures.

Thesis I I designed a local control to improve the operating frequency of the FPGA implementation of synchronous data-flow graphs and gave a method to determine the number and the topology of locally controlled components in the design space of speed and area.

I.1 I designed and implemented a distributed local control to avoid global control signals and increase the operating frequency of the control unit at the expense of a moderate area increase. [3]

I experimentally showed that global control signals of the arithmetic unit of synchronous data-flow graph based FPGA implementations (e.g. numerical solution of partial differential equations) are the bottlenecks of the operating frequency of the whole circuit if the number of I/Os of the arithmetic unit is large. I designed and implemented a locally distributed control for the arithmetic units of the aforementioned applications to avoid the blocking global signals at the expense of a moderate area increase.

To investigate the trade-off between speed and the number of I/Os, I measured the operating frequency of the proposed control logic without floating-point units in the function of the number of I/Os. On a Virtex-6 FPGA, which is designed for high-performance computations, a control restricted to maximally 10 I/Os can reach 510 MHz frequency, approximately 20% more than a control handling 20 I/Os. Assuming 450 MHz frequency for the rest of the circuit (e.g. floating-point units), the restricted control can be operated without holding back the whole circuit. For the control, it is worth to target a slightly higher theoretical frequency than the minimal 450 MHz because operating frequencies are typically much lower in practical designs, where floating-point units are also implemented.

I designed an optimization procedure to determine the locally controlled components of the arithmetic unit by partitioning the data-flow graph, if the number of I/Os exceeds the threshold required for fast operation. The resulting partition classes can be controlled independently, however, extra synchronizing First-In-First-Out (FIFO) buffers are required between the classes, which increase the number of utilized configurable logic blocks (area requirement of the circuit). I proposed a optimization problem to minimize the number of extra FIFOs when the data-flow graph is partitioned to

meet the I/O constraints required for high performance operation.

I.2 I developed a greedy partitioning algorithm, which outperformed one of the popular state-of-the-art partitioning algorithms in case of the proposed optimization problem. [4]

Regarding a computational fluid dynamics (CFD) application, I experimentally showed that partitioning objectives alone are not sufficient to reach high operating frequency, and placement objectives shall be considered as well. I designed a simple greedy algorithm which takes placement objectives into consideration and supports the manual tuning of the placement phase of high-level synthesis. Using the greedy algorithm with manual placement constraints, the design reached approximately 370 MHz operating frequency in case of a single precision CFD test case outperforming the results of the general-purpose hMetis [17] algorithm by approximately 13%. Without manual placement constraints, the same design reached 328 and 296 MHz frequency in case of single and double precision, respectively.

I.3 I developed a new graph partitioning algorithm which incorporates both partitioning and placement objectives to improve operating frequency even without manual placement constraints. [1, 5–7]

I proposed a new high-level synthesis approach, which, contrary to the traditional step-by-step strategy, incorporates placement information already at the partitioning step, and designed a new partitioning algorithm implementing the approach using simulated annealing. I evaluated the algorithm in two complex CFD test cases by measuring the operating frequency of the generated circuit in the function of the maximal I/O connection of the clusters. Maximal speed-up (15-25%) compared to the unpartitioned case was reached, when the maximal number of I/Os was set to 9 or 10. Both CFD arithmetic units reached approximately 320-325 MHz in case of double precision.

Thesis II **To improve the performance of the first hybrid CPU-GPU implementation of the Density Matrix Renormalization Group (DMRG) algorithm, I designed a scheduling algorithm for the matrix-matrix multiplications of the most time-consuming step, and developed a new algorithm for asymmetric matrix-vector multiplication in GPU.**

I analyzed the runtime of the algorithm and found the projection operation of the iterative diagonalization method (Davidson) to be the most time-consuming step, which can be rephrased as a sequence of dense matrix-matrix multiplications. I investigated the performance of GPU and FPGA in case of matrix-matrix multiplication, and found that the operation can be implemented on both architectures with high utilization, however, assuming full utilization of both architectures, the GPU is approximately 5 times faster than the FPGA. I created a high-performance hybrid GPU-CPU acceleration of the algorithm in CUDA environment, which is the first kilo-processor implementation of the algorithm and is approximately 3.5 times faster than the high-end, CPU-only version.

II.1 I designed a new scheduling algorithm for the matrix multiplications of projection operation, which is the most time-consuming part of the DMRG algorithm, to maintain high utilization of the GPU in case of different matrix sizes. [8, 2]

I investigated the size of the matrices participating in the matrix-matrix multiplications in case of the Heisenberg and the Hubbard models, which utilized different number of symmetries. I found that the size of the matrices varies widely inside and across the iterations of the algorithm, and the average matrix size is affected by the model and the number of symmetries applied.

I measured the performance of CPU and GPU in case of matrix-matrix multiplication in the function of matrix size using the MKL and the CuBLAS libraries. I experimentally showed that the utilization of GPU can be improved by parallel execution of multiplications, which is supported by the CUDA environment and also possible in the DMRG application.

To accelerate the projection operation, I proposed a hybrid implementation where multiplications are distributed between the available computing architectures. To improve the utilization of GPU during matrix multi-

plications, I designed a new scheduling algorithm supporting two different strategies. I created a single-threaded scheduling strategy for large matrices, in which case one multiplication can utilize all the GPU cores, and a multi-threaded strategy for small matrices, in which case relatively more memory is available and the parallel execution is advantageous. In the single-threaded strategy, multiplications are only scheduled to overlap the communications and the computations, however, for the multi-threaded strategy, I designed a complex algorithm, which also takes the limitations of CUDA kernel scheduling into consideration.

On the high-end K20 GPU, the parallel kernel execution significantly (44%) accelerated the multiplication of smaller matrices (range of 400-600), however, the total runtime of the algorithm slightly decreased (5%) as the average matrix size was larger in the investigated models. In practice, more complex models are investigated containing several symmetries which decrease the average matrix size and anticipate a higher speed-up.

II.2 I developed a new algorithm for GPU to significantly increase the performance of the computation of the extremely asymmetric matrix-vector multiplications used in the DMRG algorithm. [2]

To accelerate the extremely asymmetric matrix-vector operations composing the second most time-consuming part of the DMRG algorithm, I designed a hybrid acceleration, in which the workload is distributed according to the available GPU memory and the performance capabilities of the two architectures. To improve the performance of the GPU part, I proposed a new algorithm to compute the transposed matrix-vector multiplication in CUDA environment, which outperformed the NVidia CuBLAS library by 4-5 times in the DMRG use-cases, where the number of rows of the matrix was in the range of 1-24. Overall acceleration of the matrix-vector operations including the data transfer as well reached approximately 2.4 times speed-up.

4 Application of the Results

Two different types of computationally intensive problems have been researched to investigate the design methodology of the acceleration and to give a high-performance implementation on parallel architectures. Each problem was accelerated via a different architecture, and the results of the investigation were summarized in different thesis groups.

The design methodology proposed in Thesis 1 can be applied during any type of complex AU design when the AU has a significant number of I/Os and the performance takes priority over the area requirements. In my research,

the AU design was motivated by the numerical solution of different conservation laws via the FVM discretization, however, other applications require complex AU design as well, e.g. Monte Carlo experiments requiring the computation of an expression with a lot of input variables.

Numerical solution of conservation laws was successfully demonstrated on FPGAs in case of simulation of CFD [1], electromagnetics [18] or seismic waves [19]. Areas profiting from the acceleration of these simulations include automotive, aircraft and wind power industries, circuit design and seismology.

The idea to feedback the high-level floorplan information to high-level circuit design can also be generalized. In the proposed methodology, the partitioning of the FPUs can be altered freely to find a favorable floorplan, however, in theory, any free design parameter could be tuned in a similar way. The proposed methodology can be integrated into high-level synthesis tools at the AU generation step or at other parts of the compilation process where a free parameter shall be optimized for speed.

The results of Thesis 2 were primarily applied in the GPU implementation of the DMRG algorithm, however, they can be used in further applications where similar challenges occur. The presented scheduling of matrix-matrix multiplications can be applied in Tensor Network (TN) methods [20], which compose a broader class of algorithms including DMRG as well, while the proposed kernel for asymmetric matrix-vector multiplication can be applied in Davidson implementations frequently used in quantum chemistry (e.g. [21]).

As the DMRG algorithm is one of the leading tools to study the low energy physics of strongly correlated quantum systems exhibiting chain-like entanglement structure, it can be applied to simulate anisotropic materials (e.g. polymers [22]) or to describe accurately the electronic structure of open d shell molecules [23]. Furthermore, the interacting system of atoms trapped in an optical lattice, proposed as physical implementation of quantum computer, is also tractable via DMRG [24].

Acknowledgement

First of all, I would like to thank my scientific advisor Zoltán Nagy and my consultant Prof. Péter Szolgay for guiding and supporting me over the years.

I am equally thankful to Örs Legeza and Gergely Barcza for teaching me quantum physics. Furthermore, I would like to give an extra special thank to

Gergely Barcza for supporting me as a friend and as the godfather of my son.

I am grateful to Prof. Tamás Roska, Prof. Árpád Csurgay and Judit Nyékyné Gaizler, PhD for giving me encouragement and the opportunity to carry out my research at the university.

Conversations and lunches with Balázs Varga and Ádám Balogh are highly appreciated. Realistic thoughts from my ex-colleague and friend Tamás Fülöp always helped me to accurately determine my position even without a GPS. The cooperation of my closest colleagues András Kiss, Miklós Ruszinkó, Árpád Csík, László Füredi, Antal Hiba and Endre László is greatly acknowledged.

Most importantly, I am thankful to my family. I wish to thank my wife Fruzsi and son Zente for their loving care and tolerating my frequent absence. Finally, I am lucky to have my parents, grandparents, and one great grandparent sponsoring my never-ending studies.

List of Publications

Journal Publications of the Author

[1] Z. Nagy, C. Nemes, A. Hiba, Á. Csík, A. Kiss, M. Ruszinkó, and P. Szolgay, "Accelerating unstructured finite volume computations on field-programmable gate arrays", **Concurrency and Computation: Practice and Experience**, vol. 26, no. 3, pp. 615–643, 2014.

[2] C. Nemes, G. Barcza, Z. Nagy, Ö. Legeza, and P. Szolgay, "The density matrix renormalization group algorithm on kilo-processor architectures: implementation and trade-offs", **Computer Physics Communications**, 2014. DOI: 10.1016/j.cpc.2014.02.021.

Conference Publications of the Author

[3] C. Nemes, Z. Nagy, M. Ruszinkó, A. Kiss, and P. Szolgay, "Mapping of high performance data-flow graphs into programmable logic devices", in **Proceedings of the 2010 International Symposium on Nonlinear Theory and its Applications**, 2010, pp. 99–102.

[4] C. Nemes, Z. Nagy, and P. Szolgay, "Efficient mapping of mathematical expressions to fpgas: exploring different design methodologies", in **Circuit Theory and Design (ECCTD), 2011 20th European Conference on**, 2011, pp. 717–720.

[5] C. Nemes, Z. Nagy, and P. Szolgay, "Automatic generation of locally controlled arithmetic unit via floorplan based partitioning", in **Cellular Nanoscale Networks and Their Applications (CNNA), 2012 13th International Workshop on**, 2012, pp. 1–5.

[6] Z. Nagy, C. Nemes, A. Hiba, A. Kiss, A. Csik, and P. Szolgay, "Fpga based acceleration of computational fluid flow simulation on unstructured mesh geometry", in **Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on**, 2012, pp. 128–135.

[7] Z. Nagy, C. Nemes, A. Hiba, A. Kiss, Á. Csík, and P. Szolgay, "Accelerating unstructured finite volume solution of 2-d euler equations on fpgas", in **Conference on Modelling Fluid Flow (CMFF'12)**, 2012.

[8] C. Nemes, G. Barcza, Z. Nagy, Ö. Legeza, and P. Szolgay, "Implementation trade-offs of the density matrix renormalization group algorithm on kilo-processor architectures", in **Circuit Theory and Design (ECCTD), 2013 21th European Conference on**, 2013, pp. 100–104.

References

[9] Z. Nagy, Z. Vörösházi, and P. Szolgay, "Emulated digital cnn-um solution of partial differential equations", **International Journal of Circuit Theory and Applications**, vol. 34, no. 4, pp. 445–470, 2006.

[10] S. R. White, "Density matrix formulation for quantum renormalization groups", **Phys. Rev. Lett.**, vol. 69, pp. 2863–2866, 19 1992.

[11] Ö Legeza, R. Noack, J. Sólyom, and L. Tincani, "Applications of quantum information in the density-matrix renormalization group", in **Computational Many-Particle Physics**, ser. Lecture Notes in Physics, vol. 739, Berlin Heidelberg: Springer-Verlag, 2008.

[12] P. R. Panda, "Systemc: a modeling platform supporting multiple design abstractions", in **Proceedings of the 14th international symposium on Systems synthesis**, ser. ISSS '01, Montreal, P.Q., Canada: ACM, 2001, pp. 75–80.

[13] G.-J. Nam and J. Cong, **Modern Circuit Placement: Best Practices and Results**, 1st. Springer Publishing Company, Incorporated, 2007.

[14] C. J. Alpert, D. P. Mehta, and S. S. Sapatnekar, **Handbook of Algorithms for Physical Design Automation**, 1st. Boston, MA, USA: Auerbach Publications, 2008.

[15] A. Kahng, J. Lienig, I. Markov, and J. Hu, **VLSI Physical Design: From Graph Partitioning to Timing Closure**. Springer, 2011, ISBN: 0133016153.

[16] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis, **Graph Drawing: Algorithms for the Visualization of Graphs**, 1st. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.

[17] G. Karypis and V. Kumar, "HMETIS 1.5: A Hypergraph Partitioning Package", Department of Computer Science, Tech. Rep., 1998, http://www-users.cs.umn.edu/~karypis/metis.

[18] J. Durbano and F. Ortiz, "Fpga-based acceleration of the 3d finite-difference time-domain method", in **Field-Programmable Custom Computing Machines, 2004. FCCM 2004. 12th Annual IEEE Symposium on**, 2004, pp. 156–163.

[19] H. Fu, W. Osborne, R. G. Clapp, O. Mencer, and W. Luk, "Accelerating seismic computations using customized number representations on fpgas", **EURASIP J. Embedded Syst.**, vol. 2009, 3:1–3:13, Jan. 2009.

[20] J. I. Cirac and F. Verstraete, "Renormalization and tensor product states in spin chains and lattices", **Journal of Physics A: Mathematical and Theoretical**, vol. 42, no. 50, p. 504 004, 2009.

[21] C. Vömel, S. Z. Tomov, O. A. Marques, A. Canning, L.-W. Wang, and J. J. Dongarra, "State-of-the-art eigensolvers for electronic structure calculations of large scale nano-systems", **J. Comput. Phys.**, vol. 227, no. 15, pp. 7113–7124, Jul. 2008.

[22] W. Barford, **Electronic and Optical Properties of Conjugated Polymers**. Oxford University Press, 2005.

[23] G. Barcza, Ö. Legeza, K. H. Marti, and M. Reiher, "Quantum-information analysis of electronic states of different molecular structures", **Phys. Rev. A**, vol. 83, p. 012 508, 1 2011.

[24] M. Lewenstein, A. Sanpera, V. Ahufinger, B. Damski, A. Sen(De), and U. Sen, "Ultracold atomic gases in optical lattices: mimicking condensed matter physics and beyond", **Advances in Physics**, vol. 56, no. 2, pp. 243–379, 2007.