Selected HPC Solutions Based on the Maxeler DataFlow Approach

Jakob Salom and Hironori A. Fujii

Manuscript received June 30, 2013.

Abstract **-** Two major paradigms of computing are ControlFlow and DataFlow. In the case of ControlFlow, a program is written to control the flow of data through hardware. In the case of DataFlow, a program is written to configure the hardware, so the input data, when it comes, flows through the hardware in the only way that is possible. After a brief comparison of the two paradigms, more light is shed on the DataFlow paradigm and its applications. A novel classification of typical DataFlow applications is presented, in line with the most recent proposals of the European FP7/H2020 initiative. All selected examples are presented in two passes: generally and specifically.

1 Itroduction

The aim of this analysis is to present results of implementations of DataFlow Maxeler engines that are published in publicly available research papers and other sources. After analyzing all of them, their classification is introduced. Selected most representative articles from each available classification group are described. The others are just shortly introduced stating the topic, content, and acquired results (speedups, power reductions, improvements, etc.).

1.1 High Performance Computing and DataFlow Computers

It is hard to avoid starting this chapter without quoting the so often quoted title of an article – „The Free Lunch is Over“[1]. It is over. According to Moore's law, CPU frequencies of somewhere between 10 and 15 GHz should have been reached by now, and they are stuck at below four or five. And for some time. True, in CPU overclocking world contests, where cooling is acquired in any possible way and the power consumption is not important, current world record is in the range of 8 – 9 GHz. But, for the normal production processor usages, with standard cooling, these high figures seem years away.

The biggest problem with raising clock frequency is the mentioned power consumption both for the hardware functioning and for the cooling systems. It should be remembered that the electricity needed for supporting those high frequency CPUs is proportional to their frequency ($P = CfU^2$ - where C is the capacitance scaling constant, f is frequency, and V is voltage). When correct calculations are done in designing supercomputing centers, it becomes clear that one small (and growing) power plant is needed for each such center. On average, the amount spent to invest in high processing super computers equals to electricity bill paid for them during a two years period. There is another by-product of this huge power consumption – heat dissipation. There is always a problem how to get so much heat away from computer chips, boards, and units.

Thus, recent years have brought the end of “free” performance gains through processor-frequency scaling, and also showed that power consumption is now one of the main limiting factors in continuing the free ride in producing more powerful high-performance computing (HPC) systems.

Today's widening science horizons need an ever growing computational power, while this ever growing computational power (always with more available local computer memory) and progress in computer technology bring huge increases in the application of numerical techniques in sciences. The range of scientific problems that can be handled, the level of accuracy of their computer modeling and of how much one can rely on produced scientific prediction, all depend on the mentioned technological levels. So, one is driving the other. Sciences address more and more complex problems using more accurate models that will require powerful computational technology at much higher levels of performance in the years to come. The processing levels that will be needed, nicknamed "exascale", are about 100 times faster than the world's current fastest supercomputer. At the time of writing this paper (June 2013) the world most powerful supercomputer (measured using Linpack 500 Benchmark) was Tianhe-2 (MilkyWay-2) - TH-IVB-FEP Cluster based on Intel Xeon E5-2692 12C 2.2 GHz processors, with 1,024,000 GB of memory and astonishing 3,120,000 cores having peak performance of 54,902.4 TFlops while using amazing 18 MW of electrical power (precisely 17,808kW).

Also, for quite some time it has become obvious that the world is facing ever greater and greater quantities of data that have to be tackled and processed (nicknamed "BigData"). And more data, which has to be handled within the same time span, require ever increasing computer power, too.

So this is the current situation. Large increases of processing power are needed for both technological progress and for processing BigData, and Moor's law has got stuck along the way (if not for transistor count, than for the clock speed, for sure).

And now, what to do? Hiperthreading, multicore, more cash memory [1]? Can sufficient speedup on general purpose architecture be easily achieved, avoiding the load/store and instruction fetch bottlenecks of traditional Von Neumann architectures? How easily the algorithms can be changed to correctly parallelize usage of many cores of GPUs or of a few cores of CPUs? Of course, that has to be done now without having the advantage of frequency increases.

As the theory says, the computers used in modern scientific computing can be grouped into general-purpose and problem-specific ones. The basis for the first group was outlined by John von Neumann back in 1945 [2]. Although this is still the mostly used model of computer architecture now, lately, one can witness the broad scientific community accepting problem-specific architectures for solving the most challenging computational problems. Special-purpose computers can be optimized for particular characteristics of an application and, consequently, can be orders of magnitude faster in solving specific tasks. This dramatic increase in productivity not only leads to acceleration of specific applications, but in some cases, the improved performance permits researchers to tackle problems that were previously simply not amenable to numerical solution.

Special-purpose computers, were first made by programming custom solutions into the hardware, creating application-specific integrated circuits (ASICs). By all means this is the best-suited solution leading to fastest performances, but being unchangeable (once having been made), ASICs are mostly used as fixed computing elements of other devices. Also, being totally hardware specific, ASICs are

comparatively most expensive to produce. Over the years, some other alternatives entered the market - digital signal processors (DSPs), pro

grammable logic arrays (PLAs), and complex programmable logic devices (CPLDs). Finally or along with them, Field Programmable Gate Arrays (FPGAs) were created. From a designer's point of view, they can be treated as programmable ASICs. True, they give less speed and are less power efficient, but offer much more flexibility - simply because they are, as the name suggests, programmable. Also, ASICs and FPGA chips both provide a substrate for implementing highly parallel spatial computing.

Since FPGAs are programmable, such computer architecture is called reconfigurable computing architectures. Reconfigurable computing bridges the gap between traditional hardware and software approaches because a programmer or a system designer has an additional flexibility of hardware implementation. One thing that, in some way, decreases the usability of FPGA (compared to ControlFlow architectures) is a much slower clock. Thus, both architectures general-purpose and problem-specific ones have their advantages and disadvantages. As we argue in this article, the best way is to combine both, letting critical time consuming parts of the program be accelerated on FPGAs and allowing other non-critical parts of the program to use the GPPs (general purpose processors). It is important not to implement the complete algorithm as a hardware solution (on FPGA).

Although the dataflow engines can be implemented on any suitable substrate, FPGAs are used, since they offer the flexibility to implement efficient dataflow engines and to be reconfigured for each application, also, they have an existing mass market driving demand and continual improvements. Maxeler FPGA dataflow engines run at a few hundred megahertz, but they can already beat the performance of conventional CPUs, running at several gigahertz, by orders of magnitude. In addition, dataflow engines are easily able to exploit increasing silicon capacity since performance comes directly from parallelism and can scale linearly with silicon area, without depending on clock frequency increases.

When reading many papers during the preparation of this review of Maxeler HPC projects and results, it is hard not to notice that almost all of them try, at the beginning, to justify the fact that an FPGA is used as a supplement to general purpose computers in order to achieve accelerations. As if this shouldn’t have been a normal, standard procedure. The end results are remarkable, so it should be expected that, very soon, this will become a standard procedure.

The average achieved accelerations mentioned in those papers are more than one order of magnitude. Getting used to that, one is a bit disappointed when, in some of them, the end speedup is less than five - how un-modest that is. And, at the same time, when using one single architecture (GPU or CPU), years are needed to get even such a one-digit speedup either through (now small) frequency increases or through CPU/GPU code optimization.

1.2 Dataflow Engine Applications

The topics of this analysis is, DataFlow programming paradigm applied to DFE Maxeler engines. These DFEs were at first only implemented with an aim to accelerate known and already used applications and algorithms. Lately, supported by excellent results of those changes in applications, more and more new algorithms and solutions are being developed that are used on DFEs for the first time i.e. without any history on a ControlFlow architecture.

The first question faced by a developer considering transfer of a specific application to the DFE, is what are the key features the application must possess in order for the DFE implementation to yield a substantial speedup?

Our analysis of the existing bulk of DFE applications shows that there are four criteria that an application should meet in order to expect a substantial speedup as a result of its transfer:

1. BigData. Prime advantage of Maxeler DFEs is that they are able to accelerate the movement of data. In order to "feel" the acceleration, the application has to handle large quantities of data - it has to be a real BigData application. Also, since there is a short period of initial latency before the first result is delivered from a DataFlow FPGA engine, it is necessary to have a lot of data on its input to compensate that latency and provide high speedups compared to general purpose computers.
2. Multiply used data. The application should handle data more than once – the data ought to be reusable. The data transfer from the supporting CPU portion of the application to the DataFlow portion must be performed and it takes time, so the data ought not to be handled only once.
3. Loop structure. The application should have loops that consume most of processing time. The loops (handling BigData or doing high-end calculations) are the portions of the application that are moved to the DFE. So if those loops take 90% of the processing time, the speedup will be lower than ten, and if they take 99%, the theoretical limit is a 100 times shorter execution time.
4. Initial latency. The algorithm must tolerate initial latency. For the first result to "come out" of the DataFlow engine all input data have to pass through all compiled logic on the FPGA board at the clock speed of the board (right now 200MHz), and that takes time ($t = N * T$, where N is the number of parallel data moves inside the loop and T is the length of the clock cycle). All the following results (providing that the data choreography have been properly done) come out at the rate of the clock speed what is much, much faster than on CPU/GPU systems.

Secondly, what are the phases of an application transfer procedure?

1. Data analysis – Data choreography. Unlike ControlFlow application where authors do not have to worry or even to think about where the data is or how it will arrive into the application, in DataFlow programming Data choreography is an item of capital importance and very high on a To Do list. There is no pre-fetch logic that will try to predict what the next instruction or next data will be. Actually, the next instruction is written in the logic elements that are lying in front of the data item (where the data will be during the next system clock cycle), whereas the next data is the one that the programmer places behind the data currently being processed (that will be processed by the current logical elements during the next system clock cycle). So a lot of time

must be spent analyzing the data in order to have it consolidated in the way best suited for the DataFlow architecture.

2. Application adaptation. As it was pointed out, not the whole application is transferred to DFE, but only the loops that handle huge quantities of data or do high-end calculations. Those are so called "HotLoops" – loops that take too much time on the standard CPU/GPU processors. The selected portions have to be rewritten completely - adapted to the DataFlow engine.
3. Algorithm improvements. If the results of the application adaptation haven't acquired satisfactory speedups, another algorithm should be chosen or created for the same scientific programming area. In order to choose one, theoretical analyses and specification of the best from the known algorithms are done. That procedure can be a trial and error one, but after some testing, the best suited algorithms for DataFlow architectures are chosen.
4. Data analysis – Floating point precision. In order to achieve the best performance in FPGA accelerators, it is necessary to optimize arithmetic units and data types to suit the range/precision at every point in computations, bearing in mind that data accuracy must be maintained. The FPGA architecture is flexible in implementing non-standard arithmetic, and, since data-flow programming model creates a separate unit for each arithmetic operator in the code, a lot of designing freedom is at hand. In ControlFlow architecture the chosen floating point precision influences program time execution. In DataFlow it influences also the size of the created program. The higher the precision, the much higher is the number of logical elements used, thus decreasing the chip area available for other coding. On the DFE architecture the data particle can be of any size (precision). So for the floating point numbers designers are not limited only to single (32 bits) or double (64 bits) precisions. In order to get better speedups, careful consideration should be applied to determining the minimal, but absolutely sufficient, size of the floating point data unit. The decisions related to data and floating point numbers precisions should be made together with the users - BigData users and scientists – the ones who know precisely what might be the lowest limits that will certainly not affect the accuracy of algorithm results.

1.3 Viewpoint, Goals and Mission

As said – it is "the end of free lunch", there are no more great steps forward in processor clock rates. The latest supercomputers are limited by power consumption and heat dissipation and projecting technology forward it will not be possible to achieve targeted supercomputer performance within reasonable power envelopes without embracing innovative new technologies.

According to the results, which are paraphrased in the following chapters, that "new" is already here. It is only necessary to start to appreciate it. Obviously, if Linpack (or similar) benchmark is continued to be forced as the best grading tool, the looking for the new miracle will continually go on. As Michael Flynn at al. [2] put it – in order to be able to handle "Petadata" one has to forget as the only goal "Petaflops". Petaflops alone cannot crunch BigData and as a byproduct they have too much electrical power wasted. So the TOP500 benchmarks should not take just one and narrow dimension of supercomputing into account, they must have alternative performance measures for ranking, the ones with a much wider scope.

Maybe the solution is using multidimensional parameters like performance per watt, performance per cubic foot, or performance per dollar. Or, in the BigData environment that is becoming so powerful, should not petabytes per Watt be used, or per cubic foot or per dollar?

Obviously a new metering tool is needed. The goal of this paper is to support that by showing how much faster a combination of CPUs and DataFlow machines can be, compared to the ones made of only general-purpose ControlFlow ones.

The application design method that uses ControlFlow units to interface with the world, to prepare data and receive results and that uses DataFlow engines to crunch all that data – that is the solution that will dominate in the near future. In order to be able to compare computers made in that way among them, the benchmarking methods have to be changed. Otherwise, the benchmarks will keep measuring the roads that will not be fully walked on.

To repeat - the mission of this paper is to point out the better programmers'/designers' supercomputing solutions and to stress the need for creating new, more complex benchmarking systems. This will be shown by presenting articles that describe a number of successful transfers of application to Maxeler Data Flow Engines. Maxeler leads in innovations in this field, by developing and delivering High Performance Computing solutions that gain at least an order of magnitude advantage in performance per unit of rack space, or per Flops per Watt, and gives top price-performance considering total cost of ownership for monolithic applications.

2 Classification

The classification methods that will be used in this analysis were developed during the preparation of the PF7 European Union proposal for the ICT-2013.4.2 "Scalable data analytics" objective, called DAFNE.

1.1. Application and Sub-application Criteria

The classification solely depends on criteria used. Here two application criteria and one sub-application criterion will be used to present two classifications and one sub-classification.

1. The first and main criterion is what science or industry the application is targeted for. There are three groups:

a. **Exascale Fundamental Drivers** - targeting Formal sciences (Decision theory, Logic, Mathematics, and Statistics), Finances, Transportation, and algorithms pertaining to FPGA improvements;
b. **Exascale Science and Technology Drivers** - targeting Natural sciences (Life sciences, Physical sciences, and Earth sciences), Social sciences, and Medicine;
c. **Exascale Engineering and Innovation Drivers** - targeting Engineering and different industries.

This will be the main classification used in the paper. It consists of the group name (as a tree's branch) and the exact name of the science or industry (as a branch's leaf).

The other two, classification and sub-classification, will be just mentioned.

2. The second criterion is what type of the task the algorithms in the application perform. The algorithm, according to its role, is placed in one of five groups of tools:

a. Optimization toolbox

b. Complex networks analysis toolbox

c. Image, video, text processing, and analysis toolbox

d. Numerical analysis, modeling, and simulation toolbox

e. Machine learning and data mining toolbox

3. The sub-application criterion: According to what type of adaptation was performed during the application transfer procedure **i) a particular single algorithm** or **ii) a set of algorithms (framework)** or **iii) a set of related algorithms composed as a complete standalone solution**. And, additionally, according to what each algorithm targets **a) a specific application** or **b) a targeted set of applications**.

1.2. Examples for Each Leaf within the Main Classification

Exascale Fundamental Drivers:

Examples of the Leaf #1 Finance:

1. A Mixed Precision Monte Carlo Methodology for Reconfigurable Accelerator Systems [3]
2. Finding the Right Level of Abstraction for Minimizing Operational Expenditure [4]
3. Accelerating Reconfigurable Financial Computing [5]
4. Accelerating the Computation of Portfolios of Tranched Credit Derivatives [6]
5. Multi-level Customization Framework for Curve Based Monte Carlo Financial Simulations [7]
6. Rapid Computation of Value and Risk for Derivatives Portfolios [29]

Examples of the Leaf #2 Mathematics:

1. A Fully-Pipelined Expectation-Maximization Engine for Gaussian Mixture Models [13]
2. Enhancing Performance of Tall-Skinny QR Factorization Using FPGAs [16]
3. Heterogeneous Reconfigurable System
   for Adaptive Particle Filters in Real-Time Applications [17]
4. Optimizing Performance of Quadrature Methods with Reduced Precision [18]

Exascale Science and Technology Drivers:

Examples of the Leaf #1 Biology:

1. Hardware Acceleration of Genetic Sequence Alignment [11]
2. A Large-Scale Spiking Neural Network Accelerator for FPGA Systems [12]

Examples of the Leaf #2 Geophysics::

1. Maximum performance Computing with Dataflow Engines [23]
2. An Implementation of the Acoustic Wave Equation on FPGAs [20]
3. Finding Speedup in Parallel Processors [8]
4. Anisotropic Reverse-Time Migration Using Co-Processors [28]

Examples of the Leaf #3 Meteorology:

1. Acceleration of a Meteorological Limited Area Model with Dataflow Engines [22]

Exascale Engineering and Innovation Drivers

Examples of the Leaf #1 Oil and Gas Industry::

1. Surviving the End Of Frequency Scaling with Reconfigurable Dataflow Computing [26]
2. Beyond Traditional Microprocessors
   for Geoscience High-Performance Computing Applications [7]
3. Acceleration of Anisotropic Phase Shift Plus Interpolation with Dataflow Engines [9]
4. Accelerating large-scale HPC Applications using FPGAs [25]
5. Accelerating 3D Convolution Using Streaming Architectures On FPGAs [27]
6. Finite-Difference Wave Propagation Modeling on Special-Purpose Dataflow Machines [24]

2. Presentation of Examples:

In this section examples of the projects found in available open literature are presented. For each Classification leaf, one, the most representative example is described in more detail, while for the rest of them, only a short description is given together with acquired results (speedup, power consumption reduction, and other improvements).

2.1. Examples from Exascale Fundamental Drivers leaf #1 – Finance

2.1.1. Example 1 - A Mixed Precision Monte Carlo Methodology for Reconfigurable Accelerator Systems [3]

Classification #2 - Numerical analysis, modeling, and simulation toolbox
Sub-classification - A particular single algorithm for a targeted set of applications
Applications/Algorithms - Monte Carlo simulation

In this article Gary C.T. Chow at al. introduce a novel mixed precision methodology applicable to any Monte Carlo (MC) simulation. MC simulations are well suited to Field Programmable Gate Arrays, due to the parallel nature of MC algorithms and the availability of cost-effective random number generators for the DFE. The authors use data-paths with reduced precision, and for correcting any resulting errors, they

use auxiliary sampling. To determine the optimal reduced precision and optimal resource allocation among the MC data-paths and correction data-paths, optimization based on mixed integer geometric programming is implemented. Reduced-precision data-paths usually have higher clock frequencies, consume fewer resources, and offer a higher degree of parallelism, for a given amount of resources compared with full precision data-paths.

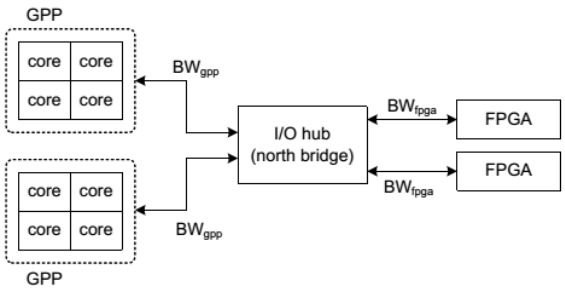
The results were evaluated in the three financial use cases: Asian option pricing, Fixed strike look-back call option under the GARCH model, and Collateralized mortgage obligation and in Multi-dimensional integral evaluation that is used in many other areas.

Essence

The major contributions of this paper are:

- An error analysis that separates finite precision error and sampling error for reduced precision Monte Carlo simulations, and a novel mixed precision methodology to correct finite precision errors through auxiliary sampling;
- Techniques for partitioning workloads of different precisions for auxiliary sampling to a reconfigurable accelerator system consisting of FPGA(s) and GPP(s);
- An optimization method based on an analytical model for the execution time of a Monte Carlo simulation on a reconfigurable accelerator system, and Mixed Integer Geometric Programming to find optimal precision for the FPGA's data-paths and optimal resource allocation.

Infrastructure



*the picture will be changed (all the pictures)

Figure 1: System architecture of the reconfigurable accelerator system in the analytical model

Relevance and details

The proposed mixed precision methodology provides several advantages over previous FPGA designs.

- The final result is adjusted with an approximated mean finite precision. This is a novel approach which enables obtaining a more accurate result from the reduced precision result instead of passively finding the error bound.
- Since there are only sampling errors in the output, a very accurate result can be achieved by increasing the number of sample points. The output accuracy is no longer limited by the reduced precision.
- The methodology is applicable to any Monte Carlo simulation because no accuracy analysis is required for the relative error and the methodology is totally independent of the function.

Application and results

	GPP only	GPU	FPGA only	FPGA + GPP
precision	double	double	double	mixed
execution time (s)	29	3	4.7	0.65
power (W)	183	236	85	192
energy (kJ)	5.3	0.71	0.4	0.13
normalised speedup	1x	9.7x	6.2x	44.6x
normalised energy	40.7x	**5.5x**	3.1x	1x

*the tables will be changed (all the tables)

Table 1: Comparison with GPP, GPU, and plain FPGA

The results are very impressive. Mixed precision FPGA reconfigurable accelerator system can be up to 4.6 times faster than state-of-the-art GPU, 7.1 times faster than a baseline FPGA design using double precision, and 163 times faster than optimized software running on a quad-core GPP. It can also be up to 5.5 times more energy efficient than a GPU and 170 times more energy efficient than a quad-core software implementation.

Characteristics

This work suggests that it is worthwhile to try using lower precision, but one should be very careful and have a numerically exact reference to evade getting the results with unsatisfactory precision. The authors had the reference results available, so having data to compare with, they could perform all the tests and get the p results.

Trends

Further analysis with changing data precision on FPGA will be continued and tried by many authors. It is something to be expected since general purpose computer architectures do not offer that possibility. Future work should be based on extending this methodology to other similar algorithms and providing tools for automating the process in order to make it both easier and faster.

2.1.2. Example 2 - Finding the Right Level of Abstraction for Minimizing Operational Expenditure [4]

Classification #2 - Numerical analysis, modeling, and simulation toolbox

Sub-classification - A set of algorithms (framework) for a targeted set of applications

Applications/Algorithms - total cost of ownership (TCO) of a financial computing operation, Monte Carlo case study

In this article Oskar Mencer et al. are examining the impact of modern programming language abstractions on total cost of ownership (TCO) of a financial computing operation. The analysis is based on static and dynamic analysis of an example financial software, based on the loopflow graph (LFG) concept and the custom dynamic hotspot tool called MaxSpot.

It is not always easy to get the correct calculation of the TCO mostly because of unsupportive corporate structure of the company. The authors suggest this list of components:

> (a) Cost of software development (including testing); (b) Capital expenditure (CAPEX): Purchase of computing equipment, Purchase of communication equipment, Purchase of computer storage, Purchase of air conditioning, and Purchase of datacenter real estate; (c) Operational expenditure (OPEX): Datacenter staff, real estate operations, and Electricity for computers & air conditioning; and (d) Indirect costs: Cost of failures (business damage from scheduled and unpredicted downtime and non-availability), Cost of waiting to see if there is a human or technical failure, and Cost of reliability (Redundancy, backups, testing, and verification of computing).

In all businesses it is difficult to calculate what are the real costs of not getting the needed data on time (or sooner than that). In the financial industry it's even more so (the interest rates, the fast changes on the financial instruments markets etc.).

The authors took a simple Monte Carlo Application in C and reprogrammed it in C++ to be able to raise the level of abstraction (adding random number generators and payoff calculations) making code easy to be expanded with little programming effort.

The most valuable result of these analyses is the fact that if the required throughput of an application is high enough, the operational expenditure is minimized by minimizing runtime and not by minimizing programming effort.

2.1.3. Example 3 - Accelerating Reconfigurable Financial Computing [5]

Classification #2 - Numerical analysis, modeling, and simulation toolbox
Sub-classification - A set of related algorithms composed as a complete standalone solution, for a targeted set of applications
Applications/Algorithms - derivative pricing using both Monte-Carlo and quadrature methods

This PhD thesis proposes novel approaches to the design, optimization, and management of reconfigurable compute accelerators for financial computing. There are three contributions. First one is the proposed novel reconfigurable designs for derivative pricing using both Monte-Carlo and quadrature methods. Such designs involve exploring techniques such as control variate optimization for Monte-Carlo, and multidimensional analysis for quadrature methods. Significant speedups and energy savings are achieved using the Field-Programmable Gate Array (FPGA) designs over both CPU and GPU designs.

Second, it proposes a framework for distributing computing tasks on multi-accelerator heterogeneous clusters. In this framework, different computational devices including FPGAs, GPUs and CPUs work

collaboratively on the same financial problem based on a dynamic scheduling policy. The trade-off in speed and in energy consumption of different accelerator allocations is investigated. Third, it proposes a mixed precision methodology for optimizing Monte-Carlo designs, and a reduced precision methodology for optimizing quadrature designs. These methodologies enable optimizing throughput of reconfigurable designs by using data paths with minimized precision, while maintaining the same accuracy of the results as in the original designs.

This work clearly shows the trade-offs between number of computations and the contribution of each computation in increasing the accuracy of the final result. Using the mixed precision methodology, the speedup of 2.9 to 7.1 times over the double precision FPGA designs and 44 to 106 times speed up over the quad-core CPU designs is achieved, Using the mixed precision methodology, an Virtex-6 ST475X FPGA and an i7-870 CPU are able to out-perform the GPU by 4.6 times.

2.1.4. Example 4 - Accelerating the Computation of Portfolios of Tranched Credit Derivatives [6]

Classification #2 – N/A

Sub-classification - A set of related algorithms composed as a complete standalone solution, for a targeted set of applications

Applications/Algorithms - standard base correlation methodology, with a Gaussian copula for default correlation and a stochastic recovery process

Huge growth in the trading and complexity of credit derivative instruments over the past five years has driven the need for ever more computationally demanding mathematical models. This has led to massive growth in data center compute capacity, power, and cooling requirements. The paper reports the results of a joint project between J.P. Morgan and specialist acceleration solutions provider Maxeler Technologies on improving the price/performance for calculating the value and risk of a large complex credit derivatives portfolio.

The results show that valuing tranches of Collateralized Default Obligations (CDOs) on Maxeler accelerated systems is over 30 times faster per cubic foot and per Watt than solutions using standard multi-core Intel Xeon processors. Also reports on some preliminary results of further work that extends the approach to classes of interest rate derivatives are given.

2.1.5. Example 5 - Multi-level Customization Framework for Curve Based Monte Carlo Financial Simulations [7]

Classification #2 - Numerical analysis, modeling, and simulation toolbox

Sub-classification - A particular single algorithm for a targeted set of applications

Applications/Algorithms - Monte Carlo framework for the automated generation of highly efficient curve based payoff evaluation accelerator,

One of the main challenges when accelerating financial applications using reconfigurable hardware is the management of the design complexity. This paper proposes a multi-level customization framework for automatic generation of complex yet highly efficient curve based financial Monte Carlo simulators on reconfigurable hardware. By identifying multiple levels of functional specializations and the optimal data

format for the Monte Carlo simulation, different levels of programmability in the framework to retain good performance and support multiple applications are allowed.

Designs targeting a Virtex-6 SX475T FPGA generated by this framework are about 40 times faster than single-core software implementations on an i7-870 quad-core CPU at 2.93 GHz; they are also over 10 times faster and 20 times more energy efficient than 4-core implementations on the same i7-870 quad-core CPU, and are over three times more energy efficient and 36% faster than a highly optimized implementation on an NVIDIA Tesla C2070 GPU at 1.15 GHz.
In addition, the framework is platform independent and can be extended to support CPU and GPU applications.

2.1.6. Example 6 - Rapid Computation of Value and Risk for Derivatives Portfolios [29]

Classification #2 - Machine learning and data mining toolbox
Sub-classification - A set of related algorithms composed as a complete standalone solution for a specific application
Applications/Algorithms - valuation to risk measurement and multi-variate Monte Carlo derivative pricing model

This paper reports on new results from a project to accelerate derivatives computations. The earlier work was focused on accelerating the valuation of credit derivatives. In this paper, the work is extended in two ways: by applying the same techniques, first, to accelerate the computation of portfolio level risk for credit derivatives and, second, to different asset classes using a different type of mathematical model, which together present challenges that are quite different to those dealt with in the earlier work. Also the implications for risk are explored.

The acceleration of 270 times over a single Intel Core for a multi-asset Monte Carlo model is reported.

2.2. Examples from Exascale Fundamental Drivers leaf #2 – Mathematics

2.2.1. Example 1 - A Fully-Pipelined Expectation-Maximization Engine for Gaussian Mixture Models [13]

Classification #2 - Machine learning and data mining toolbox
Sub-classification - A particular single algorithm for a targeted set of applications
Applications/Algorithms - Gaussian Mixture Models for probability density modeling and soft clustering

In this text Ce Guo at al. describe Gaussian Mixture Models (GMMs), a powerful tool for probability density modeling and soft clustering, implemented at Maxeler DFE. In many applications, it is necessary to estimate the parameters of a GMM from data before working with it. This task can be handled by the Expectation-Maximization algorithm for Gaussian Mixture Models (EM-GMM), which is computationally demanding. The authors propose a pipeline-friendly EMGMM algorithm, a variant of the original EM-GMM algorithm that can be converted to a fully-pipelined hardware architecture suitable for FPGA

implementation. To further improve the performance, they use a Gaussian probability density function evaluation unit that works with fixed point arithmetic.

GMMs originally came from statistical machine learning and they play key roles in a large number of applications in data mining, signal processing, and computer vision. Time spent on each run directly depends on the data size. In recent years, the EM-GMM algorithm has become increasingly computationally demanding since the development of high-definition sensors and novel Internet technology has led to a fast growth of data size (BigData) and more often it is desirable for the algorithm to be executed within a short period of time. This is especially true when the algorithm has to be invoked for multiple times or it is real-time application.

Essence

The major contributions of this paper are:

- Restructuring the work flow of the original EM-GMM algorithm with algorithmic transformations to enable pipelining of different computation stages, resulting in a pipeline-friendly EM-GMM algorithm.
- Proposed customized design of the Gaussian probability density evaluation unit that minimizes the hardware cost while achieving satisfactory accuracy.
- Overcoming precision problem in Gaussian PDF evaluation with bit shifting and successfully deployed fixed point arithmetic throughout the system.

Infrastructure

In the experiments about performance, the systems are compared with a GPU implementation described in [31]. As the authors did not have the experimental results of this system on their data, the performance was estimated according to those results and trends. As they did not use the same data sets, they took the best performance record in [31] with similar data size. That means that the performance estimation might not be very meaningful and the GPU comparison results are provided in the paper only for reference.

The two CPU implementations are deployed on a PC with an Intel Core i3 CPU (running at 2.93GHz) and 4 GB DDR3 memories. Both implementations are coded in the C programming language on a single CPU core and compiled with the highest compiler optimization in the Intel C compiler. The FPGA implementation is deployed in a Maxeler MAX3 acceleration card with a Xilinx Virtex-6 FPGA running at 150MHz and 48GB DDR3 on-board memory.

Relevance and Details

Gaussian Mixture Models (GMMs) are powerful tools for probability density modeling and soft clustering. Some relevant usages quoted in the paper are: a speaker verification system where GMMs are used to model the characters of a speaker's voice, or a computer vision system to subtract the background from video streams where GMMs are used to judge whether a pixel belongs to the background in a probabilistic manner, or an image segmentation system to identify tissues in magnetic resonance (MR) images of the brain where GMMs are employed to capture the spatial layout information of brain tissues.

The fundamental difference between the original EM-GMM and the pipeline-friendly EM-GMM is that the former requires data to be streamed into the algorithm for three times, while the latter requires only once. Other differences include the following:

- The expectation step and the maximization step become overlapped in the pipeline-friendly algorithm. In the original algorithm, the maximization step does not start until all the data instances are processed in the expectation step. In the pipeline-friendly algorithm however, the data set is handled in a per-instance manner. Statistical information of the new parameter set is updated once the data instance arrives.
- The original algorithm stores all the responsibility values in the expectation step. The pipeline-friendly algorithm computes the responsibility values for a newly arrived data instance and updates the statistical information of the new parameter set. In this case, it is not necessary for the algorithm to store all responsibility values. When the statistical information about a data instance is collected, the corresponding responsibility values can be discarded safely.

Application and Results

The algorithm performance is described by the number of data instances processed in every second. A data instance is considered to be processed in one iteration after all the computations related to it are completed in that iteration. Experimental results on performance are recorded in Table 2. The last two columns in the table record the speedup values of FPGA-based solution over the CPU-based solution (the original EM-GMM) and the GPU based solution respectively.

Data	CPU1	CPU2	GPU	FPGA	SU_{C1}	SU_G
1	1.732e6	2.040e6	3.081e7	1.493e8	86x	5x
2	8.565e5	1.014e6	1.541e7	1.492e8	174x	9x
3	5.689e5	6.671e5	1.027e7	1.492e8	262x	15x
4	8.714e5	1.023e6	1.541e7	1.487e8	171x	9x
5	4.310e5	5.079e5	7.704e6	1.487e8	346x	15x
6	2.883e5	3.402e5	5.136e6	1.487e8	517x	28x

TO BE CHANGED

Table 2: Performance Results (Instances per Second)

In the experiments, the FPGA-based solution generates fairly accurate results while achieving a maximum of 517 times speedup over a CPU-based solution, and 28 times speedup over a GPU-based solution

Characteristics

The possible reason behind the high speedup of this solution is that the fixed-point arithmetic saves hardware resources on the FPGA platform. This enables deployment of up to 36 Gaussian PDF evaluation units in the pipeline, which exploits available FPGA resources well. However, it is not feasible to perform similar optimization on CPUs and GPUs. With richer logical resources on the FPGA platform, it would be possible to deploy even larger number of Gaussian PDF evaluation units enabling complicated data to be processed by the system. The corresponding acceleration would be more significant. Future work should

be based on extending this methodology on other similar algorithms and, again, providing tools for automating the process in order to make it both easier and faster.

Trends

This example shows the possible advantages of the FPGA architecture in all algorithms that work satisfiably with fixed point arithmetic.

2.2.2. Example 2 - Enhancing Performance of Tall-Skinny QR Factorization Using FPGAs [16]

Classification #2 - Optimization toolbox

Sub-classification - A particular single algorithm for a targeted set of applications

Applications/Algorithms - Tall-Skinny QR factorization

Communication-avoiding linear algebra algorithms with low communication latency and high memory bandwidth requirements like Tall-Skinny QR factorization (TSQR) are highly appropriate for acceleration using FPGAs. TSQR parallelizes QR factorization of tall-skinny matrices in a divide-and-conquer fashion by decomposing them into sub-matrices, performing local QR factorizations and then merging the intermediate results. As TSQR is a dense linear algebra problem, one would therefore imagine GPU to show better performance. However, the performance of GPU is limited by the memory bandwidth in local QR factorizations and global communication latency in the merge stage. In this paper the shape of the matrix is exploited on FPGA-based custom architecture, which avoids these bottlenecks by using high-bandwidth on-chip memories for local QR factorizations and by performing the merge stage entirely on-chip to reduce communication latency.
The result that is achieved is a peak double-precision floating-point performance of 129 GFLOPs on Virtex-6 SX475T. A quantitative comparison of the proposed design with recent QR factorization on FPGAs and GPU shows up to 7.7× and 12.7× speed up respectively. Additionally, even higher performance over optimized linear algebra libraries like Intel MKL for multi-cores, CULA for GPUs, and MAGMA for hybrid systems is shown.

2.2.3. Example 3 - Heterogeneous Reconfigurable System for Adaptive Particle Filters in Real-Time Applications [17]

Classification #2 - Optimization toolbox

Sub-classification - A particular single algorithm for a targeted set of applications

Applications/Algorithms - Particle filter statistical method for dealing with dynamic systems having nonlinear and non-Gaussian properties.

This paper presents a heterogeneous reconfigurable system for real-time applications applying particle filters. The system consists of an FPGA and a multi-threaded CPU. A method is proposed to adapt the number of particles dynamically and utilize the run-time reconfigurability of the FPGA for reduced power and energy consumption. An application is developed which involves simultaneous mobile robot localization and people tracking. It shows that the proposed adaptive particle filter can reduce up to 99% of computation time.
Using run-time reconfiguration, a reduction of 34% in idle power and 26-34% of system energy is achieved. The proposed system is up to 7.39 times faster and 3.65 times more energy efficient than the

Intel Xeon X5650 CPU with 12 threads, and 1.3 times faster and 2.13 times more energy efficient than an NVIDIA Tesla C2070 GPU.

2.2.4. Example 4 - Optimizing Performance of Quadrature Methods with Reduced Precision [18]

Classification #2 - Optimization toolbox

Sub-classification - A particular single algorithm for a targeted set of applications

Applications/Algorithms - Quadrature methods

This paper presents a generic precision optimization methodology for quadrature computation targeting reconfigurable hardware to maximize performance at a given error tolerance level. The proposed methodology optimizes performance by considering integration grid density versus mantissa size of floating-point operators. The optimization provides the number of integration points and mantissa size with maximized throughput while meeting given error tolerance requirement.
Three case studies show that the proposed reduced precision designs on a Virtex-6 SX475T FPGA are up to 6 times faster than comparable FPGA designs with double precision arithmetic. They are up to 15.1 times faster and 234.9 times more energy efficient than an i7-870 quad core CPU, and are 1.2 times faster and 42.2 times more energy efficient than a Tesla C2070 GPU.

2.3. Examples from Exascale Science and Technology Drivers leaf #1 – Biology

2.3.1. Example 1 - Hardware Acceleration of Genetic Sequence Alignment [11]

Classification #2 - Numerical analysis, modeling, and simulation toolbox

Sub-classification - A particular single algorithm for a specific application

Applications/Algorithms – DNA sequencing, alignment processor based on a backtracking variation of the FM-index algorithm

Next generation DNA sequencing machines have been improving at an exceptional rate so the subsequent analysis of the generated sequenced data has become a bottleneck in current systems. In this paper J. Arram at al. explore the use of reconfigurable hardware to accelerate the short read mapping problem, where the positions of millions of short DNA sequences are located relative to a known reference sequence. The proposed design comprises of an alignment processor based on a backtracking variation of the FM-index algorithm. The design represents a full solution to the short read mapping problem, capable of efficient exact and approximate alignment.

Next-generation sequencing (NGS) machines with performances improving at a rate faster than Moore's law, are able to rapidly and inexpensively produce sequenced data. To improve the throughput and measurement accuracy of these machines, shorter sequences are processed, allowing tens of billions of bases to be sequenced per day. These short sequences can be created by breaking the long DNA chain randomly. As a consequence of this action, the position and orientation information of the fragments with respect to the sample is lost. Based on the assumption that all DNA sequences within a species are similar, the sample DNA can be reconstructed by determining the location of the short fragments (the short reads) in a known reference genome of the species.

This application involves highly-parallel bit-oriented operations based on a backtracking FM-index algorithm, so FPGA technology is a promising candidate for its acceleration. The design represents a full solution to short read mapping, capable of both exact and approximate alignment.

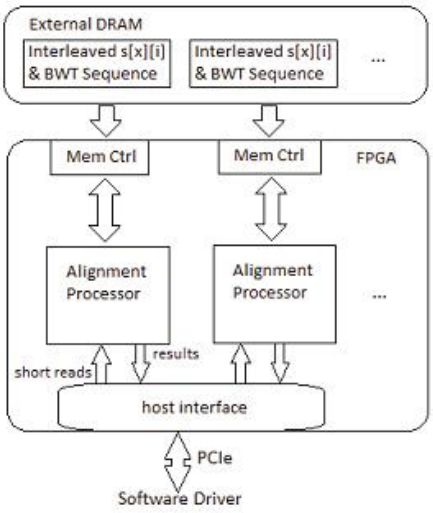
Essence

The major contributions of this work include:

- A hardware design for a novel sequence alignment processor based on a backtracking FM-index algorithm. Various optimizations, such as those for memory size, memory bandwidth, and latency are analyzed.
- An implementation of the proposed design on a Maxeler MAX3 board.
- Performance evaluation of the proposed design, with comparisons against some of the fastest software solutions on multi-core processors and GPUs, as well as FPGA hardware solutions.

Infrastructure

The design architecture consists of an FPGA populated with alignment processors, connected to the host processor through a software driver. Before alignment starts, the software driver transfers the Burrows-Wheeler transform (BWT) sequence to the accelerator board, where they are stored using on-chip BRAM or external DRAM. Short reads are then streamed to the alignment processors in batches given by the design latency. Each batch of short reads is processed for a number of iterations determined by the permitted number of mismatches and the short read length. The alignment results for each short read, including the SA interval, the cost and a string representation of the alignment are reported to the software driver. This architecture is illustrated in Figure 2.



TO BE CHANGED

Figure 2: Design architecture

The alignment processor design can be fully mapped to hardware, therefore the software driver has a minimal role in the design architecture. The reference sequence changes infrequently, therefore it is assumed that the BWT sequence is generated in advance. After transferring the data structures to the accelerator board and configuring the number of mismatches permitted, the short reads are streamed to

the alignment processors populating the FPGA. The software driver then pauses until all the accelerator output is received. If a short read can be matched to the reference sequence, the SA interval is mapped to positions in the reference sequence using a simple lookup table. This step can be performed in hardware, however it was chosen to perform it using a CPU in order to reduce the number of memory controllers required by each alignment processor. If a short read is unaligned, it is streamed again to the alignment processors for testing with a higher number of permitted mismatches.

Relevance and Details

The authors proposed an alignment processor design with the following features:

- A backtracking version of the FM-index algorithm with a data structure that supports both forward and backward search, which is capable of exact and approximate alignment.
- A novel scheme to reduce the memory size of the FM-index occurrence array, allowing it to be stored directly on the accelerator board.
- A new method to reduce external memory access frequency, while maximizing memory bandwidth utilization.
- A novel scheme to process batches of short reads in parallel, maximizing the throughput of the design.

Application and Results

Since different packages report their performance using different data sets, it is difficult to directly compare designs using the raw results. To better assess the performance of various designs, the authors defined the bases aligned per second (baps) value as a normalized performance measure unit.

Table 3 presents the achieved results.

Platform	Clock freq (MHz)	Devices	Cores	*baps* (millions)	Energy (W-hr)
Intel X5650	2670	1	20	1.04	19
Intel X5650	2670	1	20	1.76	11
Intel Xeon X5650	2670	1	20	1.59	13
NVIDIA GTX 580	900	1	512	3.84	6.3
Xilinx Virtex-6 SX475T	150	1	3	13.5	0.078

TO BE CHANGED

Table 3: Aligner and energy performance comparison

Characteristics

This work demonstrated that an alignment processor based on the backtracking FM Index algorithm can achieve high throughput for short read alignment applications. The design was able to map short reads to a full genome faster than available software aligners, without compromising the alignment sensitivity.

What was also very important, it consumes significantly less power than other software aligners.

Trends

More work should be done to overcome the following disadvantage. For up to two mismatches the design has a comparable sensitivity (~100%) to available software aligners. For short reads with more than two mismatches, the sensitivity of the software aligners sharply decreases (<20%). This is a result of the aligner being unable to explore the large search space within the cut off time. Current and future research should include further optimization of this approach, together with its application in clinical procedures.

2.3.2. Example 2 - A Large-Scale Spiking Neural Network Accelerator for FPGA Systems [12]

Classification #2 - Complex networks analysis toolbox
Sub-classification - A particular single algorithm for a specific application
Applications/Algorithms - alignment processor based on a backtracking variation of the FM-index algorithm

Spiking neural networks (SNN) aim to mimic membrane potential dynamics of biological neurons. They have been used widely in neuromorphic applications and neuroscience modeling studies. Despite the vast amount of anatomical and functional knowledge of the brain, the complete picture of how higher cognitive function emerges from neuronal and synaptic dynamics still eludes the scientists. Large-scale simulation is useful in this regard, since it can be investigated how such functions emerge from deterministic simulation. The authors designed a parallel SNN accelerator for producing large-scale cortical simulation targeting an off-the-shelf FPGA based system. The accelerator parallelizes synaptic processing with run time proportional to the firing rate of the network.

Using only one FPGA, this accelerator is estimated to support simulation of 64K neurons. The accelerator is 1.4 times (localized connectivity) to 5.5 times (uniform connectivity) faster than the GPU NeMo accelerator (Tesla C1060 65nm process) in terms of spike delivery rate. However, current accelerator does not support axonal delay and synaptic plasticity which require additional hardware resources.

2.4. Examples from Exascale Science and Technology Drivers leaf #2 – Geophysics

2.4.1. Example 1 - Maximum Performance Computing with Dataflow Engines [23]

Classification #2 - Optimization toolbox

Sub-classification - A set of related algorithms composed as a complete standalone solution for a specific application

Applications/Algorithms - Wave-modeling application

In this article Oliver Pell et al. discuss multidisciplinary dataflow computing as a powerful approach to scientific computing that has led to orders-of magnitude performance improvements for a wide range of applications. As an example they provide of one particular application - FD wave modeling.

One of the main factors that determine the resolution of seismic images is the bandwidth of the seismic wavelet. Finite difference (FD) modeling and Reverse Time Migration (RTM) encounter particular problems increasing the wavelet bandwidth at the upper end of the spectrum because of the large impact this has on the computation resource requirements. Increasing the upper modeled frequency requires a finer spatial sampling while the CFL limit implies that the modeling time step must decrease. That causes the number of arithmetic operations to grow as it is proportional to the fourth power of frequency. Modeling at high frequencies (for example, 70Hz) can easily require hundreds of gigabytes of memory.

Essence

In dataflow-accelerated wave modeling, the CPUs in the system retain control of the application, and instruct the dataflow engine(s) to compute each time step. Pressure and velocity data volumes reside in the local memory of each dataflow engine and are streamed through the dataflow implementation of the modeling kernel, which computes the next time-step pressure field. Stimulus data can be sent from the CPU to the dataflow engine to be added to points in the field at runtime, and data can also be read out each time step to visualize or store to disk.

Infrastructure

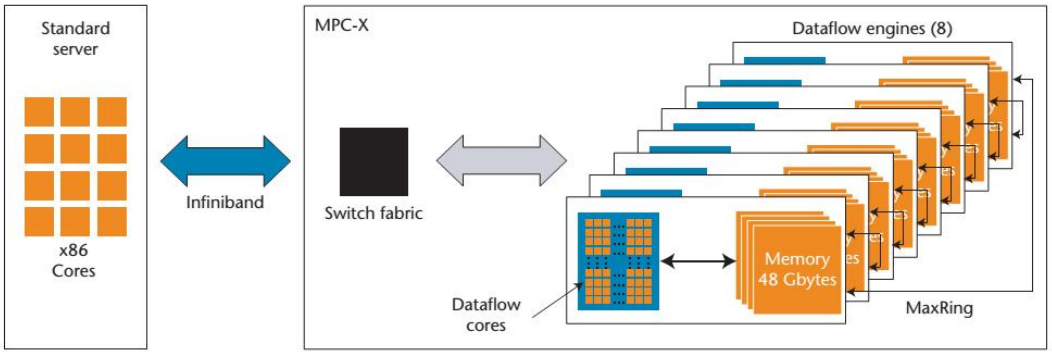


Figure 4. MPC-X series architecture. These are stand-alone dataflow compute nodes that connect to CPU-only nodes in a system via Infiniband.

TO BE CHANGED

Multiple dataflow engines can work together on a single modeling problem, by splitting the domain in one dimension into multiple subdomains and assigning each subdomain to different engines. At the edge of each sub domain, engines must exchange boundary data with their neighbors, and they can do this using the direct MaxRing interconnect. Because MaxRing is a point-to-point interconnect, the communication bandwidth scales as the number of engines increases.

Relevance and Details

Finite difference is a major numerical method used to solve PDEs such as the wave equation. In particular, for the geosciences, medical imaging, and physics simulations, explicit FD is an elegant and regular algorithm that affords efficient implementation within the dataflow paradigm.

Application and Results

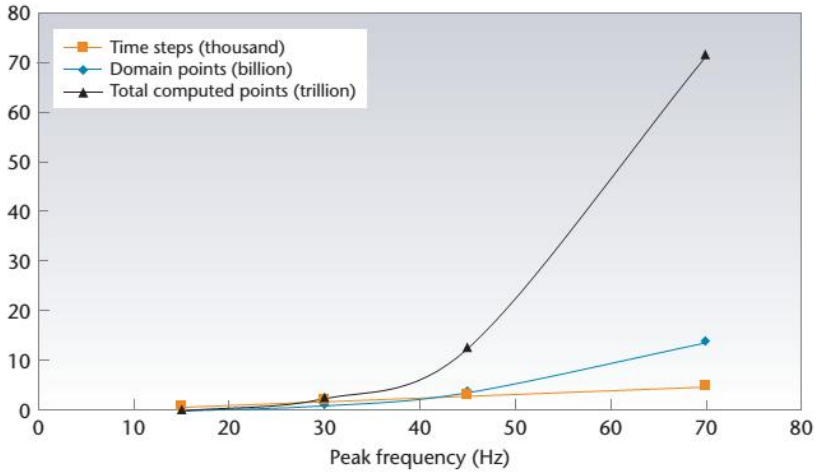


Figure 5. Impact of increased modeling frequency on memory and computation costs. In wave modeling, the amount of computation required increases with the fourth power of the wave frequency.

TO BE CHANGED

The Maxeler MAX2 FPGA card used has two Xilinx Virtex-5 FPGAs and 24GB of on-board DRAM. MAX2 has a high speed MaxRing link between cards that enables multiple MAX2 cards to work together to achieve the 70Hz bandwidth objective. The performance of the MAX2 system is compared to a C++ software version on a cluster with 32 3GHz X86 cores communicating via MPI over Infiniband. The maximum performance of the accelerated node is equivalent to nearly 2000 CPU cores: one MAX2 card provides the equivalent performance of over 200 CPU cores. [21]

Characteristics

Dataflow computing can deliver orders-of-magnitude improvements in space and power consumption for a wide range of applications; and dataflow compute engines can be balanced with other kinds of compute resources in a cluster environment (such as CPUs and storage). The benefits that can be seen show considerable promise for achieving the potential of exascale computing.

2.4.2. Example 2 - An Implementation of the Acoustic Wave Equation on FPGAs [20]

Classification #2 - Numerical analysis, modeling, and simulation toolbox

Sub-classification - A set of algorithms (framework) a specific application
Applications/Algorithms - The acoustic forward modeling application, 3D finite difference

FPGA chips are utilized as co-processors in a PCI Express configuration to accelerate anacoustic isotropic modeling application. The acoustic forward modeling application in consideration is 3D finite difference, with 4th-order in time, 12th order in space and uses single precision floating point arithmetic. The acoustic variable density modeling code contains a kernel which consumes the majority of the compute cycles, indicating that the algorithm is a good candidate for acceleration. The finite difference operators are calculated to minimize the relative phase velocity error over the bandwidth.

Optimization provides a peak speedup of over 160x over a single core, or 28-48x speedup per node depending on multi-core scaling. It can be observed that FPGA speedup increases further for problem sizes beyond 400 mesh, too.

2.4.3. Example 3 - Finding Speedup in Parallel Processors [8]

Classification #2 - Numerical analysis, modeling, and simulation toolbox
Sub-classification - A set of related algorithms composed as a complete standalone solution for a targeted set of applications
Applications/Algorithms - Geophysical Modeling, Forward modeling a finite difference method

One particularly compute intensive application is concerned with oil and gas exploration. In this application data is collected by first distributing a grid of sensors over a large area. Then a sonic impulse is applied to the area and reflections are recorded: frequency, amplitude and delay at each sensor. Sonic impulse could be a compressed air canon (at sea) or explosives (on land).
A typical sea based survey uses 30,000 sensors to record data (over a 120 dB dynamic range). Each sensor is sampled at more than 2kbps with a new sonic impulse occurring every 10 sec. Structures in the earth reflect the impulse and reflections are detected by sensor array. The resulting data is of the order of terabytes of data each day.
In this article the authors propose an acceleration methodology based on FPGA arrays. The methodology uses a comprehensive application analysis supported by high performance FPGA hardware. The analysis provides a dataflow graph of the application which is replicated in SIMD for multiple data strips (until limited by the pin bandwidth), then pipelined (MISD) until circuit limited.

In this particular application the FPGA solution shows the possibility of speedup of over 300x over an Intel Xeon.

2.4.4. Example 4 - Anisotropic Reverse-Time Migration Using Co-Processors [28]

Classification #2 - Image, video, text processing, and analysis toolbox
Sub-classification - A particular single algorithm for a targeted set of applications
Applications/Algorithms - Seismic imaging, anisotropic reverse-time migration

Co-processors offer attractive acceleration opportunities to waveform-based imaging and inversion applications in challenging exploration and production environments. Unlike seismic forward modeling, the large amount of data involved in seismic imaging and inversion can pose a significant challenge to scalable acceleration. The authors provide and compare several computational schemes to perform anisotropic reverse-time migration on two co-processor platforms: FPGAs and GPUs. The ongoing experiments so far indicate that both platforms can potentially achieve high speedups using acceleration-friendly schemes which minimize interruptions to computation from data movement and storage.
FPGA-accelerated isotropic modeling reached a speedup rate of 20x over 8 cores. Currently, accelerated isotropic wave propagation on FPGAs can go beyond 40x (over 8 cores) or 200x over a single core.

2.5. Examples from Exascale Science and Technology Drivers leaf #3 – Meteorology

2.5.1. Example 1 - Acceleration of a Meteorological Limited Area Model with Dataflow Engines [22]

Classification #2 - Machine learning and data mining toolbox

Sub-classification - A set of algorithms (framework) for a specific application

Applications/Algorithms – Hydrostatic Limited Area Model derived from the BOLAM model

Climate and weather modeling need High Performance Computing due to the hard deadlines inherent in predicting weather. Given the large data volumes and runtimes involved, climate and weather modeling is ideally suited for dataflow computation. In this paper, Diego Oriato at al. demonstrate a dataflow implementation of the dynamic core of a meteorological limited area model. To achieve maximum performance necessary computation was transformed by reordering operations and encoding the data. They focused on the dynamic core of a Limited Area Model (LAM) derived from the BOLAM model, which is a research-oriented hydrostatic LAM developed by ISAC-CNR (Bologna, Italy) and parallelized using domain decomposition and message passing libraries [15].

They present results for a domain of 13,600 x 3,333 x 30 km with 620 thousand grid points. The aim was to develop a computationally fast version of the dynamic core of the hydrostatic model to satisfy requirements of higher spatial resolution for regional weather forecast, large domain dimension for global models, and very long time integration typical of climate simulations.

Essence

A complete version of a hydrostatic LAM is made up essentially of three blocks: initialization, post-processing and the meteorological model that numerically solves the prognostic equations governing atmospheric circulation. The last part, the most complex and computationally expensive, is composed of two macro blocks: the dynamic core and the physical parameterizations routines (of which the most computational expensive is the radiation). The primitive equations are a set of nonlinear 3D partial differential equations that approximate global atmospheric flow. They consist of equations for the conservation of momentum, continuity, and thermal energy.

This paper gives an example of a possible standard procedure that could be taken in transferring application from a ControlFlow computer to Maxeler Data Flow Engine.
The whole work is divided work in four phases.

In *Structural Analysis* phase the time step computation was decomposed into five logical blocks. Each block is structured as a 3D loop where longitude, latitude and altitude are respectively fast, medium and slow dimensions.

In *Partitioning* phase the authors used Maxeler Parton toolset to analyze the CPU time spent on each logical block of the application when run on an Intel Xeon core. The toolset gave as a result a lot of relevant data. The most interesting was the number floating point operations. The authors decided to migrate all the blocks to DFE because an acceleration of 100x or more was the goal. By moving all the computation to the DFE data transfer between CPU and DFE was also minimized.

In *Transformation* phase the dataflow computing operations were implemented spatially as part of a pipeline through which data was streamed rather than each instruction being executed temporally on a new piece of data. For this reason a dataflow algorithm is inherently parallel. The aim is therefore to group all the computational logic inside a single loop and to replace the loop with a deterministic data access pattern. This requires an understanding of the data dependency among the different parts of the algorithm.

In *Parallelization* phase an analysis was done how in detail to parallelize the algorithm and the data flow. A dataflow engine comprises memory coupled to a chip implementing many dataflow cores, as shown in Fig.4. the dataflow cores are arranged in a pipeline which processes one item of data per clock cycle. To facilitate parallelization part of the computation was converted to use fixed-point arithmetic, which is more efficient in terms of silicon area per operation than the equivalent floating-point arithmetic. Analysis was performed on each block of the algorithm to establish the optimal fixed-point representations to maximize precision and avoid overflow, applying scaling coefficients where prognostic variables exhibited high dynamic range. The authors also chose to use one-to-one mapping of serial simulation per DFE, giving the capability of running six independent simulations in parallel.

Infrastructure

The dataflow application was run on a Maxeler MPC-C series dataflow node (eight Intel Xeon E5506@2.13GHz CPU cores and six MAX3 Dataflow Engines connected to the CPUs via PCI Express (as shown in Fig.4). Each MAX3 DFE utilizes a Xilinx Virtex 6 SX475T FPGA to implement the dataflow cores and 48GB of memory. Maxeler's MaxCompiler development environment was used to implement the dataflow pipeline and integrate it into the original FORTRAN application.

All the variables needed during the computation were stored on the DFE. The prognostic variables were initialized from CPU and transferred out at the last time step of the simulation. The CPU controlled the time step loop by triggering the dataflow process and waiting for it to finish before repeating the trigger. No computation was executed on the CPU other than for initialization and post processing.

Relevance and Details

The LAM dataset used is a medium size domain of 13600 km (160 points) in longitude, 3333 km (120 points) in latitude and 30 km (32 points) in altitude with a baroclinic atmosphere for the initial condition. Although boundary conditions limit the clear development of the baroclinic instability, the authors believe that this initial condition, together with the fixed boundary conditions, is a representative test to

evaluate numerical stability of the migrated application. Spatial resolution chosen was of 0.25° both in latitude and longitude with a time step of 15s to satisfy the CFL criteria, which guarantees the stability of the solutions.

Application and Results

It is interesting to note that increasing the number of cores on a ControlFlow CPU system did not scale up the application speed proportionally. The parallel software was run on a dual Intel Xeon X5650@2.7Ghz six-core CPU coupled to 192GB DDR3 memory. Results showed that the software does not scale very well when using more than two cores; four cores give a speedup of only 2.8x compared to a single core. Using all twelve cores becomes less efficient than running with four cores with only a 2.6x speedup. The highest speedup of 3.1x is reached with eight cores.

The speedups acquired by using a single DFE was 64x compared to a single node single core CPU and 12x compared to a single node 12 cores CPU. When MPC-C Series Node was used the speedup was 381x compared to a single node single core CPU and 74x compared to a single node 12 cores CPU.

Peak power usage of a MPC-C series node was measured to be around 900W. Considering a peak value of 400W per Intel node, 74 twelve-core nodes and therefore around 30kW to would be needed to match the performance of the dataflow solution.

Characteristics

This is an example of an application transfer that gave really good results, not only in the speedups, but also in power reduction. It shows that, providing a good preparation and analysis is done prior to the migration process, quite good results can be expected.

2.6. Examples from Exascale Engineering and Innovation Drivers leaf #5 – Oil industry

2.6.1. Example 1 - Surviving the End of Frequency Scaling with Reconfigurable Dataflow Computing [26]

Classification #2 - Optimization toolbox
Sub-classification - A set of algorithms (framework) for a specific application
Applications/Algorithms - CRS seismic trace stacking, a fitness function

In this paper, besides presenting a very good example of a DFE usage in Oil and Gas industry, Oliver Pell at al. give an excellent overview of Maxeler FPGA computer; of how the heterogeneous computing is the solution for frequency crunch in CPU/GPU systems; how parallelism is built and can be used in multi and many core systems (CPU/GPU); and finally how the only real available way to exa-scale supercomputing, that isn't at the same time a devouring consumer of electrical power, is to use and to take advantage of capabilities of the both worlds – ControlFlow and DataFlow computers.

It is not at all suggested that dataflow engines are suitable to replace the conventional CPU entirely. They should be used to increase the performance of the processors. Most of the lines of code in a program

will still run on the CPU, but for computationally intensive components of an application it can make sense to offload these to a custom dataflow engine (Figure 4). A typical Maxeler HPC compute node consists of some number of CPUs coupled to some number of FPGAs; for example one current standard 1U node has 12 Intel Xeon CPU cores, 4 Xilinx Virtex-6 FPGAs and 100-400GB of RAM split between the CPUs and FPGAs. Large RAM capacities are important for many of the target applications, which process very large data volumes.

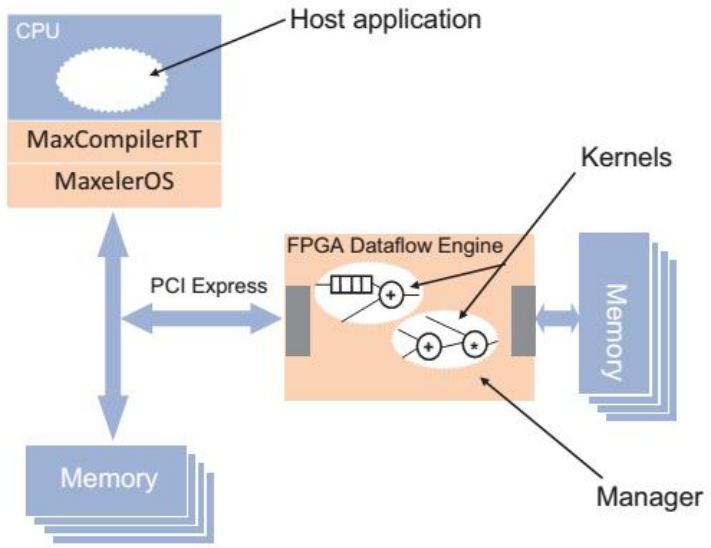


Figure 4: A CPU coupled to a custom FPGA dataflow engine.

TO BE CHANGED

The main challenge in the practical exploitation of dataflow engines is to program them. Unlike writing code for a CPU where the programmer is creating a set of instructions that will be executed by existing function units, to create a dataflow engine the programmer must actually construct a circuit that represents their application. However, this does not need to be an electrical engineering process of circuit design - Maxeler has developed a programming tool called MaxCompiler that allows software engineers to implement dataflow engines for an application using a high-level software environment.

Essence

The oil and gas industry is one of the major industry consumers of HPC computing. We are exploiting reserves of “the black gold” for more than 150 years, and while depleting the existing reservoirs, we have to keep finding new ones. In order to find them, the scientists need to make images of the subsurface that expose details that haven’t been visible with existing technology.

To create an image of the subsurface, scientists conduct an acoustic experiment on the earth’s surface. In the experiment, a low-frequency source is activated and the reflections from the different subsurface layers are recorded by tens of thousands of sensors. This experiment is repeated many, many times,

creating a data set of dozens or hundreds of Terabytes. To process the data, thousands of compute nodes are used and the processing lasts a very long time.

Infrastructure

CRS stacking is an algorithm used to process seismic survey data to compute zero-offset traces. These can be easily computed from the input data given 8 parameters to the CRS equation. The stacking application must determine good values for the eight parameters before it can compute the stack, by performing a search in 8 dimensional space. A fitness function is evaluated at each point in the space to determine the quality of the current parameter set.

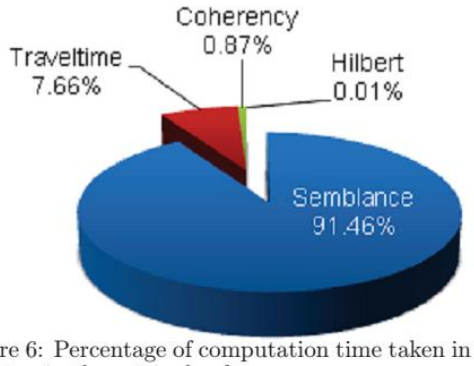


Figure 6: Percentage of computation time taken in each function in the original software.

TO BE CHANGED

On CPU, execution time to compute CRS for a typical survey can be of the order of 1 month using 1000 CPU cores, and this runtime is dominated by the computation of the Semblance – fitness (91.46% of the CPU computation time) and Traveltime (7,66%). The travel time function calculates the location that should be read from a data trace based on the eight CRS parameters, while the semblance function computes on a window of data values from that location.

Application and Results

With both Semblance and Traveltime computation on the FPGA, over 99% of the total runtime from the CPU is accelerated. The final implementation gives a speedup of approximately 230x compared to a single core for land datasets and 190x for marine datasets, what is approximately 30x greater performance/W.

Characteristics

Definitely a lot of sciences and industries can successfully use heterogeneous computing architecture. The best results, up to now, have been achieved in the fields of Finances and Oil and Gas industries. This is an example of how huge speedups such architecture can offer.

2.1.1 Example 2 - Beyond Traditional Microprocessors for Geoscience High-Performance Computing Applications [7]

Classification #2 - Optimization toolbox

Sub-classification - A set of algorithms (framework) for a specific application

Applications/Algorithms – Reverse Time Migration

The oil and gas industry is a major user of high-performance computing, and geoscience computational cycles are dominated by kernels that are relatively few and well defined. This project explores accelerating geoscience applications using FPGA-based hardware, optimizing the algorithm and the hardware to achieve maximum performance. Modeling and computation are taking an unprecedented role in the search for, and the extraction of, energy sources like oil and gas.
The oil and gas industry already uses high-performance computing (HPC), but it's still unclear how conventional HPC technologies can meet the demands of tomorrow's algorithms. 1 In this article, authors describe the acceleration of the most demanding applications in this domain using field-programmable gate array (FPGA) technology. This avoids some of the performance-scaling issues frequently encountered with CPUs and GPUs.
This approach can deliver speedup of 20 to 70 times compared to a conventional HPC node.

2.6.2. Example 3 - Acceleration of Anisotropic Phase Shift Plus Interpolation with Dataflow Engines [9]

Classification #2 - Optimization toolbox

Sub-classification - A set of algorithms (framework) for a specific application

Applications/Algorithms - Phase Shift Plus Interpolation method

Although time-domain depth migration techniques have been successfully ported to run on modern hardware accelerators, their ultimate obstacle is the I/O overhead present during the imaging step. Frequency-domain depth migration algorithms overcome this limitation and can exploit the full potential of new computing technologies. In particular, this implementation of Phase Shift Plus Interpolation (PSPI) method is characterized by fast running time, good quality results under low signal-to-noise ratio conditions and excellent results for steep dips.
The measurements indicate that a dataflow approach can achieve high speedups despite larger and larger computational domains, increased complexity of the anisotropic approach and the I/O overhead during angle-gathers calculation. When the MAX2 and MAX3 system are compared to a Fortran software version on a node with 8 Intel Xeon 2.6GHz cores parallelized using MPI, the speedup is from 13 to 34x.

2.6.3. Example 4 - Accelerating large-scale HPC Applications using FPGAs [25]

Classification #2 - Optimization toolbox

Sub-classification - A particular single algorithm for a targeted set of applications

Applications/Algorithms - Wave propagation using 3D finite difference

Field Programmable Gate Arrays (FPGAs) are conventionally considered as 'glue-logic'. However, modern FPGAs are extremely competitive compared to state-of-the-art CPUs for commercial HPC workloads, such as those found in Oil and Gas and Finance. For example, an FPGA accelerated system can be 31-37 times faster than an equivalently sized conventional machine, and consume 1⁄39 of the power. The key to achieving the best performance in FPGA accelerators, while maintaining accuracy, is optimization of arithmetic units and data types to suit the range/precision at each point in the computation. The flexibility of the FPGA to implement non-standard arithmetic, combined with a data-flow programming model that instantiates a separate unit for each arithmetic operator in the code provides a wide design space. As such, FPGA computing offers significant opportunity for arithmetic research into 'large scale'

HPC applications, where there is an opportunity to move away from standard IEEE formats, either to improve precision compared to the CPU version or to increase speed.
The key to performance in stacking on the FPGA is to maximize reuse of each point loaded from DRAM, otherwise the available DRAM bandwidth into the chip is the limit to performance. The degree of reuse of each input data point changes with varying the number of output data points computed in parallel. When doing 64 parallel searches, the degree of reuse of each input data point, while varying the number of output data points computed in parallel, is high enough so that the application is no longer memory bound.
The authors used the flexibility of FPGA arithmetic to trade off precision and performance in different versions of the application. A full precision version, used for pricing accurate to 10^{-8}, gives a 31x speedup over an 8-core Xeon E5430 server.

2.6.4. Example 5 - Accelerating 3D Convolution Using Streaming Architectures on FPGAs [27]

Classification #2 - Optimization toolbox

Sub-classification - A set of algorithms (framework) for a specific application

Applications/Algorithms - 3D Convolution

In this paper Haohuan Fu at al. investigate FPGA architectures for accelerating applications whose dominant cost is 3D convolution, such as modeling and Reverse Time Migration (RTM). The authors explore different design options, such as using different stencils, fitting multiple stencil operators into the FPGA, processing multiple time steps in one pass, and customizing the computation precisions. The exploration reveals constraints and tradeoffs between different design parameters and metrics.
They are using two major FPGA advantages over other computation platforms: (1) Streaming computation architecture and (2) Customizable number representations.
They experimented with processing different number of time steps in one pass and got different speedup and the same happened when different floating-point precisions were used.

The experiment results show that the FPGA streaming architecture provides great potential for accelerating 3D convolution, and can achieve up to two orders of magnitude speedup. By dividing the array into two parts and computing in two FPGAs concurrently speedups of up to 55x and 47x can be achieved.

2.6.5. Example 6 - Finite-Difference Wave Propagation Modeling on Special-Purpose Dataflow Machines [24]

Classification #2 - Numerical analysis, modeling, and simulation toolbox

Sub-classification - A particular single algorithm for a targeted set of applications

Applications/Algorithms - 3D finite-difference calculations

Modeling wave propagation through the earth is an important application in geoscience. In this paper Oliver Pell at al. present a framework for wave propagation modeling on special-purpose hardware, which dramatically improves the application performance compared to conventional CPUs. They use custom hardware platforms consisting of a mix of x86 CPUs and dataflow engines connected by high bandwidth communication links.

The application-specific dataflow engines run at hundreds of MHz with massive parallelism and deliver high performance/Watt, up to 30 times more energy efficient than conventional CPUs. The power efficiency of this approach suggests that dataflow computing may have a key role to play in the improvements in power efficiency necessary to reach exascale computing.

2.7. Additional Research Papers

In the last couple of years many post-graduate students in countries of Southern Europe have been doing intensive experimenting with porting algorithms and applications to Maxeler Data Flow Engines. A few of their research papers were published in a special July 2013 issue "Maxeler Super Computer Related Research" of *IPSI Transaction on Internet Research* magazine.

The examples of Exascale Fundamental Drivers leaf #2 - Mathematics can be found in [32], [33], [34], and [36].
The example of Exascale Science and Technology Drivers leaf #3 - Meteorology can be found in [37].
The examples of Exascale Science and Technology Drivers leaf #4 - Physics can be found in [35] and [38].

3. Conclusion

3.1. What has been done

In this paper some examples of successful implementations of a computer architecture that uses both General purpose computers and Special purpose reconfigurable computers are presented. It has been shown that such a combination can be a solution for exascale HPC computing, and that instead of building huge Supercomputing datacenters with exascale power consumption, a tendency should be to introduce Data Flow Engines to decrease the pressure and satisfactorily perform the necessary tasks. The data in the next sub-section, taken from [26], support this statements with interesting numbers.

3.1.1. How to reach Exascale

As it was supposed a few years ago supercomputing should reach an Exascale computer sowhere in the. Ona couls wonder what will (or might) that machine look like. Let us imagine that it will be built with standard multi-core processors. For example to be made of CPU cores running at 2.5GHz, each able to execute 8 floating point operations per cycle. With that speed the CPU core will have a peak performance of 20GFLOPS (2×10^{10} FLOPs). Now the calculation - 1 exaflop = 1×10^{18} FLOPs what accounts for at least 50 million cores to achieve exaflop performance. OK, it's feasible, Tiane-2 has 3,120,000 cores.

But, let us see how much it costs in MW. If one chip uses 100W, and with the help of Moore's law we get from 8 cores per chip today to maybe 100 cores per chip by 2018 (very optimistic, but ok, it's for the sake of calculation). Than these 50M cores would be on 500,000 CPU chips. At 100W per chip this gives a power usage of 50MW just for CPUs - for a complete system probably more than 100MW with all other

components, and then double that again, when power for cooling is included. Now, let us compare with the biggest power plants in the world [40]:

Large coal-fired, nuclear, and hydroelectric power stations can generate hundreds of Megawatts to multiple Gigawatts. Some examples:

The Three Mile Island Nuclear Generating Station in the USA has a rated capacity of 802 MW.
The coal-fired Ratcliffe-on-Soar Power Station in the UK has a rated capacity of 2 GW.
The Aswan Dam hydro-electric plant in Egypt has a capacity of 2.1 GW.
The Three Gorges Dam hydro-electric plant in China will have a capacity of 22.5 GW when complete;
18.2 GW capacity is operating as of 2010.

We have to wonder if that makes ANY sense.

How much will the electricity cost? At an optimistic electricity cost of $0.10 per kWh, a 100MW machine costs will cost $10k per hour to run, a staggering $87M a year.

And the last, but not the least - what is the programmability of a machine 50M cores might also be a problem. Considering an application performing a local convolution with a 13×13×13 3D stencil on a 10000^3 regular grid. The grid has 1.0 terapoints - 8TB if stored using double precision floating point. But, if decomposed spatially across all the cores in the system, it will end up with only 20k data points per core - or a 27^3 region. Computing on such a small region, 66% of data required by each core will be in "other" regions and must be requested from nearby cores - transforming "local" calculation into a non-local one will almost surely severely decrease performance.

So, it has to be repeated again and again, the future is in heterogeneous supercomputing. The sooner we accept that fact, the better.

3.2. Who Should Benefit from This Analysis

This paper has been written for two types of audiences.

The first is one that is not very familiar with FPGA and DataFlow programming paradigm. That audience can learn how useful it is to have heterogeneous HPC computing, how big speedup can be gained and what savings of space and power are available.

The other represents current users of DataFlow programming paradigm. The can find in an easy way a paper that handles similar applications and algorithms to those that are of their interest and use the experience written in the quote articles.

3.3. Conclusion

It is getting more and more difficult is to envisage how we can ever solve exascale problems and deliver answers to real world problems in science including such an extraordinary long space structure as the space elevator in the order of 10*E12 cm (T: tera) [41], [42]., industry and commerce. In solving specific

real problems one should consider the question of whether it is more appropriate to build application-specific computers to enable tackling large-scale problems much more efficiently than it is possible using general-purpose machines.

At Maxeler, this approach to building application specific computers is called *Maximum Performance Computing*. The intention is to design computer systems that offer maximum performance within certain cost or power or space, or all them together. Custom hardware is employed to build special-purpose dataflow engines on FPGAs which are configured for a particular application. Domain specific knowledge and experience is applied to develop solutions that are optimized at each level of abstraction ranging from the mathematics and the algorithm down to the physical hardware.

The aim is not to develop a computer which offers acceptable performance across the set of all applications, but rather to focus on a narrower range of problems and offer one to two orders of magnitude improvement in performance and power efficiency for those applications compared to conventional multi-core systems. Of course, this is not the way to build a "general purpose" exascale computer, it is nevertheless an attractive way for solving real exascale problems.

References

[1] Sutter, H., "The Free Lunch is Over: A Fundamental Turn towards Concurrency in Software, " *http://www.gotw.ca/publications/concurrency-ddj.htm*. Used under fair use guidelines, 2011.

[2] Flynn, M., Mencer, O., Milutinovic, V., Rakocevic, G., Stenstrom, P., Trobec. R., Valero, M., "Moving from petaflops to petadata," Communications of the ACM, ACM, New York, NY, USA Volume 56 Issue 5, May 2013, pp. 39-42, Available on http://dl.acm.org/citation.cfm?id=2447989

[3] Chow, G. C. T., Tse, A. H. T., Jin, O., Luk, W., Leong, P. H. W., Thomas, D. B., "A Mixed Precision Monte Carlo Methodology for Reconfigurable Accelerator Systems," *Proceedings of ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA),* Monterey, CA, USA, February 2012, pp. 57-66, Available on http://dl.acm.org/citation.cfm?id=2145705.

[4] Mencer, O., Vynckier, E., Spooner, J., Girdlestone, S., Charlesworth, O., "Finding the Right Level of Abstraction for Minimizing Operational Expenditure*," Proceedings of the fourth workshop on High performance computational finance WHPCF '11*, ACM New York, NY, USA ©2011, pp. 13-18, Available on http://dl.acm.org/citation.cfm?id=2088262

[5] Tse (Anson), H. T., "Accelerating Reconfigurable Financial Computing," PhD Thesis, *http://www.doc.ic.ac.uk/~htt08/thesis_at.pdf*, Imperial College, London, G. Britain, January 2012, pp. 1-181., Available on http://www.doc.ic.ac.uk/~htt08/thesis_at.pdf

[6] Weston, S., Marin, J-T., Spooner, J., Pell O., Mencer, O., "Accelerating the Computation of Portfolios of Tranched Credit Derivatives," *Proceedings of 2010 IEEE Workshop on High Performance Computational Finance (WHPCF)*, New Orleans, LA, USA, 14-14 November 2010, pp. 1-8, Available on http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5671822&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5671822

[7] Lindtjorn, O., Clapp, R.G., Pell, O., Mencer, O., Flynn, M. J., Fu, H., "Beyond Traditional Microprocessors for Geoscience High-Performance Computing Applications," *Micro IEEE*, (Volume:31, Issue: 2), IEEE Computer Society, March-April 2011, pp. 41 - 49., Available on http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5719584&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5719584

[8] Flynn, F., Dimond, R., Mencer, O., Pell, O., "Finding Speedup in Parallel Processors," *Proceedings of International Symposium on Parallel and Distributed Computing*, Krakow, Poland, 1-5 July 2008, pp. 3-7., Available on http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4724222&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4724222

[9] Tomas, C., Cazzola, L., Oriato, D., Pell, O., Theis, D., Satta, G., Bonomi, E., "Acceleration of Anisotropic Phase Shift Plus Interpolation with Dataflow Engines," *Proceedings of 82nd Annual Meeting and International Exposition of the Society of Exploration Geophysics-SEG*, Las Vegas, - Nevada, USA, - 2012, Available on http://publications.crs4.it/pubdocs/2012/TCOPTSB12/Tomas_PSPI_Acceleration_FPGA.pdf

[10] Viswanathan, V., "Hardware Support for Dynamic Partial Reconfiguration," Master thesis, *http://repository.tudelft.nl/view/ir/uuid%3A9e2b32fb-278f-4f6f-9a32-91f6f49bc656/*, Delft University, Holland, 2012, pp. 1-91, Available on http://repository.tudelft.nl/view/ir/uuid%3A9e2b32fb-278f-4f6f-9a32-91f6f49bc656/

[11] Arram, J., Tsoi, K. H., Luk, W., Jiang, P., "Hardware Acceleration of Genetic Sequence Alignment," *Proceedings of 9th International Symposium ARC 2013*, Los Angeles, CA, USA, March 25-27, 2013. pp. 13-24., Available on http://link.springer.com/chapter/10.1007%2F978-3-642-36812-7_2

[12] Cheung, K., Schultz, S.R., Luk, W., "A Large-Scale Spiking Neural Network Accelerator for FPGA Systems," *Proceedings on 2nd International Conference on Artificial Neural Networks*, Lausanne, Switzerland, September 11-14, 2012, Part I, pp. 113-120., Available on http://link.springer.com/content/pdf/10.1007%2F978-3-642-33269-2_15.pdf

[13] Guo, C., Fu, H., Luk, W., "A Fully-Pipelined Expectation-Maximization Engine for Gaussian Mixture Models," *Proceedings of 2012 International Conference on Field-Programmable Technology (FPT)*, Seoul, S. Korea, 10-12 Dec. 2012, pp. 182 - 189, Available on http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6412132&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6412132

[14] Jin, Q., Dong, D., Tse, A. H. T., Chow, G. C. T., Thomas, D. B., Luk, W., Weston, S., "Multi-level Customization Framework for Curve Based Monte Carlo Financial Simulations," *ARC'12 Proceedings of the 8th international conference on Reconfigurable Computing: architectures, tools and applications*, Hong Kong, March 2012, pp. 187-201, Available on http://dl.acm.org/citation.cfm?id=2247097

[15] Marrocu, M., Scardovelli, R., Malguzzi, P., "Parallelization and performance of a meteorological Limited Area Model", *Parallel Computing 24* (1998), pp. 911-922, Available on http://www.sciencedirect.com/science/article/pii/S0167819198000325

[16] Rafique, A. Kapre N., Constantinides, G. A., "Enhancing Performance of Tall-Skinny Qr Factorization Using FPGAs," *Proceedings from 22nd International Conference on Field Programmable Logic and Applications (FPL)*, 29-31 Aug. 2012, pp. 443 - 450., Available on http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6339142&sortType%3Dasc_p_Sequence%26filter%3DAND(p_IS_Number%3A6339128)

[17] Chau, T. C. P., Niu, X., Eele, A., Luk, W., Cheung, P. Y. K., Maciejowski, J., "Heterogeneous Reconfigurable System for Adaptive Particle Filters in Real-Time Applications," *Proceedings from 9th International Symposium on Reconfigurable Computing: Architectures, Tools and Applications ARC 2013*, Los Angeles, CA, USA, March 25-27, 2013. pp. 1-12., Available on http://link.springer.com/chapter/10.1007%2F978-3-642-36812-7_1

[18] Tse, A. H. T., Chow, G. C. T., Jin, Q., Thomas, D. B., Luk, W., "Optimizing Performance of Quadrature Methods with Reduced Precision," *Proceedings from 8th International Symposium ARC 2012*, Hong Kong, China, March 19-23, 2012. pp. 251-263., Available on http://link.springer.com/chapter/10.1007%2F978-3-642-28365-9_21

[19] Chow, G.C.T.; Kwok, K.W.; Luk, W.; Leong, P., "Mixed Precision Processing in Reconfigurable Systems," *Proceedings from 2011 IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Salt Lake City, UT, USA, 1-3 May 2011, pp. 17 - 24., Available on http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5771241

[20] Nemeth, T., Stefani, J., Chevron, W. L.,; Dimond, R., Pell, O., Ergas, R., "An Implementation of the Acoustic Wave Equation on FPGAs," *Proceedings of 78th Society of Exploration Geophysicists (SEG)* Meeting, Las Vegas, USA, November 2008., Available on http://www.doc.ic.ac.uk/~wl/icprojects/papers/seg08max.pdf

[21] Pell, O., Oriato, D., Andreoletti, C., Bienati, N., " SUMMARY FD modeling beyond 70Hz with FPGA acceleration," *Maxeler summary*, Available on http://www.maxeler.com/media/documents/MaxelerSummaryFDModelingBeyond70Hz.pdf

[22] Oriato, D., Tilbury, S., Marrocu, M., Pusceddu, G., "Acceleration of a Meteorological Limited Area Model with Dataflow Engines," *Proceedings of 2012 Symposium on Application Accelerators in High Performance Computing*, Chicago, IL, USA, 10-11 July 2012, pp. 129 – 132, Available on http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6319200&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6319200

[23] Pell, O., Averbukh, V., "Maximum performance Computing with Dataflow Engines," *Computing in Science & Engineering* (Volume:14, Issue: 4), Los Alamitos, CA,USA, July-Aug. 2012, pp. 98 - 103, Available on: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=6241370

[24] Pell, O., Bower, J., Robert, D., Oskar, M., Flynn, M., "Finite-Difference Wave Propagation Modeling on Special-Purpose Dataflow Machines," *IEEE Transactions on Parallel and Distributed Systems* (Volume:24, Issue: 5), May 2013, pp. 906-915., Available on http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6226384

[25] Dimond, D., Racaniere, S., Pell, O., "Accelerating large-scale HPC Applications using FPGAs," *Proceedings of 20th IEEE Symposium on Computer Arithmetic (ARITH)*, Tubingen, Germany, 25-27 July 20, 2011 pp. 191-192, Available on http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5992126&contentType=Conference+Publications

[26] Pell, O., Mencer, O., "Surviving the end of frequency scaling with reconfigurable dataflow computing," *Newsletter ACM SIGARCH Computer Architecture News archive*, Volume 39 Issue 4, September 2011, pp. 60-65 ACM New York, NY, USA, Available on https://dl.acm.org/purchase.cfm?id=2082172&CFID=227499296&CFTOKEN=94763184

[27] Fu, H., Clapp, R. G., Mencer, O., Pell, O., "Accelerating 3D Convolution Using Streaming Architectures On FPGAs," *SEG Houston 2009 International Exposition and Annual Meeting,*

Houston, Texas, USA, 25-30 October 2009, Available on http://www.onepetro.org/mslib/servlet/onepetropreview?id=SEG-2009-3035

[28] Liu, W., Nemeth, T., Loddoch, A., Stefani, J., Ergas, R., Zhuo, L., Volz, B., Pell, O., Huggett, J., "Anisotropic Reverse-Time Migration Using Co-Processors," *SEG Houston 2009 International Exposition and Annual Meeting*, Houston, Texas, USA, 25-30 October 2009, Available on http://www.onepetro.org/mslib/servlet/onepetropreview?id=SEG-2009-3040

[29] Weston, S., Spooner, J., Racaniere, S., Mencer, O.: "Rapid computation of value and risk for derivatives portfolios.," *Concurrency And Computation: Practice And Experience*, 2012;24:880–894, 7 July 2011, Wiley Online Library, DOI: 10.1002/cpe.1778, Available on http://onlinelibrary.wiley.com/doi/10.1002/cpe.1778/full

[30] Dimond, R., Flynn, M., Mencer, O. Pell, O., "MAXware: Acceleration in HPC," IEEE HOT CHIPS 20, Stanford, CA, USA, August 2008, Available on http://www.hotchips.org/wp-content/uploads/hc_archives/hc20/3_Tues/HC20.26.531.pdf

[31] Kumar, N., Satoor, S., Buck, I., "Fast parallel expectation maximization for Gaussian mixture models on GPUs using CUDA," in IEEE International Conference on High Performance Computing and Communications. IEEE, 2009, pp. 103–109, Available on http://ieeexplore.ieee.org/Xplore/defdeny.jsp?url=http%3A%2F%2Fieeexplore.ieee.org%2Fstamp%2Fstamp.jsp%3Ftp%3D%26arnumber%3D5166982&denyReason=-134&arnumber=5166982&productsMatched=null&userType=mem

[32] Stanojevic, I., Senk, V., Milutinovic, V. , "Application of Maxeler Dataflow Supercomputing to Spherical Code Design, " IPSI Transactions on Internet Research, Belgrade, Serbia, July 2013, Volume 9, Number 2, ISSN 1820 – 4503, pp. 1 - 4. Available on http://internetjournals.net/

[33] Rankovic, V.; Kos, A., Milutinovic, V., "Bitonic Merge Sort Implementation on the Maxeler Dataflow Supercomputing System," IPSI Transactions on Internet Research, Belgrade, Serbia, July 2013, Volume 9, Number 2, ISSN 1820 – 4503, pp. 5 – 9. Available on http://internetjournals.net/

[34] Bezanic, N., Popovic-Bozovic, J., Milutinovic, V., and Popovic, I., "Implementation of the RSA Algorithm on a Dataflow Architecture, " IPSI Transactions on Internet Research, Belgrade, Serbia, July 2013, Volume 9, Number 2, ISSN 1820 – 4503, pp. 11 – 18. Available on http://internetjournals.net/

[35] Stojanovic, S., Bojic, D., Milutinovic, V. , "Solving Gross Pitaevskii Equation Using Dataflow Paradigm," IPSI Transactions on Internet Research, Belgrade, Serbia, July 2013, Volume 9, Number 2, ISSN 1820 – 4503, pp. 19 – 22. Available on http://internetjournals.net/

[36] Sustran, Z., Todorovic, M,, Milutinovic, V., "Feasibility Study on the SAT Solver on DataFlow Architecture," IPSI Transactions on Internet Research, Belgrade, Serbia, July 2013, Volume 9, Number 2, ISSN 1820 – 4503, pp. 23 – 27. Available on http://internetjournals.net/

[37] Ivkovic, S., Ilic, L., Stankovic, M., Radojicic, R., Babovic, Z., "Source-Sink Model," IPSI Transactions on Internet Research, Belgrade, Serbia, July 2013, Volume 9, Number 2, ISSN 1820 – 4503, pp. 28 – 33. Available on http://internetjournals.net/

[38] Korolija, N., Djukic, T., Milutinovic, V., Filipovic, N., "Accelerating Lattice-Boltzman Method Using the Maxeler DataFlow Approach," IPSI Transactions on Internet Research, Belgrade, Serbia, July 2013, Volume 9, Number 2, ISSN 1820 – 4503, pp. 34 – 42. Available on http://internetjournals.net/

[39] Milutinovic, V., “The Current Challenges in DataFlow Supercomputer Programming ,” Proceeidings of ACM/IEEE ISCA 2013, Tel Aviv, Israel, June 24, 2013,

[40] Wikipedia, “Power station”, http://en.wikipedia.org/wiki/Power_station, June 30, 2013

[41] Hironori Fujii (Guest editor),”Special issue on Tether Tether,” Journal of Space Technology and Science (JSTS), Vol.26 No.1 (2012 spring), pp.1.

[42] Special Issue Dedicated to Prof. H.A.Fujii, Transactions on Advanced Research IPSI Bgd Internet Research Society, July 2013 Vol. 9 No. 2.