Trifunovic et al. Journal of Big Data (2015) 2:4
DOI 10.1186/s40537-014-0010-z

Journal of Big Data
a SpringerOpen Journal

METHODOLOGY Open Access

Paradigm Shift in Big Data SuperComputing: DataFlow vs. ControlFlow

Nemanja Trifunovic[1*], Veljko Milutinovic[1], Jakob Salom[2] and Anton Kos[3]

* Correspondence:
nemanja@maxeler.com
[1]School of Electrical Engineering, University of Belgrade, Belgrade, Serbia
Full list of author information is available at the end of the article

Abstract

The paper discusses the shift in the computing paradigm and the programming model for Big Data problems and applications. We compare DataFlow and ControlFlow programming models through their quantity and quality aspects. Big Data problems and applications that are suitable for implementation on DataFlow computers should not be measured using the same measures as ControlFlow computers. We propose a new methodology for benchmarking, which takes into account not only the execution time, but also the power and space, needed to complete the task. Recent research shows that if the TOP500 ranking was based on the new performance measures, DataFlow machines would outperform ControlFlow machines. To support the above claims, we present eight recent implementations of various algorithms using the DataFlow paradigm, which show considerable speed-ups, power reductions and space savings over their implementation using the ControlFlow paradigm.

Introduction

Big Data is becoming a reality in more and more research areas every year. Also, Big Data applications are becoming more *visible* as they are slowly entering areas concerning the general public. In other words, Big Data applications that were up to now present mainly in the highly specialized areas of research, like geophysics [1,2] and financial engineering [3], are making its way into more general areas, like medicine and pharmacy [4], biology, aviation [5], politics, acoustics [6], etc.

In the last years the ratio of data volume increase is higher than the ratio of processing power increase. With the growing adoption of data-collecting technologies, like sensor networks, Internet of Things, and others, the data volume growth ratio is expected to continue to increase.

Among others, one important question arises: how do we process such quantities of data. One possible answer lies is in the shift of the computing paradigm and the programming model. With Big Data problems, it is many times more reasonable to concentrate on data rather than on the process. This can be achieved by employing DataFlow computing paradigm, programming model, and computers.

Background and literature review

The strength of DataFlow, compared to ControlFlow computers is in the fact that they accelerate the data flows and application loops from 10× to 1000×. How many orders

of magnitude depends on the amount of data reusability within the loops. This feature is enabled by compiling down to levels much below the machine code, which brings important additional effects: much lower execution time, equipment size, and power dissipation.

The above strengths can prove especially important in Big Data applications that can benefit from one or more of the DataFlow advantages. For instance:

- A daily periodic Big Data application, which would not finish in time, if executed on a ControlFlow computer, executes in time on a DataFlow computer of the same equipment size and power dissipation,
- A Big Data application with limited space and/or power resources (remote locations such as ships, research stations, etc.) executes in a reasonable amount of time,
- With Big Data applications, where execution time is not a prime concern, DataFlow computers can save space and energy.

The previous paper [7] argues that time has come to redefine *TOP500* benchmarking. Concrete measurement data from real applications in geophysics [1,2], financial engineering [3], and some other research fields [8,9,10-12], shows that a DataFlow machine (for example, the Maxeler MAX series) rates better than a ControlFlow machine (for example, Cray Titan), if a different benchmark is used (e.g., a Big Data benchmark), as well as a different ranking methodology (e.g., the benchmark execution time multiplied by the number of 1U boxes needed to accomplish the given execution time - 1U box represents one rack unit or equivalent - it is assumed, no matter what technology is inside, the 1U box always has the same size and always uses the same power).

In reaction to the previous paper [7], scientific community insists that more light is shed on two issues: (a) Programming paradigm and (b) Benchmarking methodology. Consequently the stress of this viewpoint is on these two issues.

Discussion

What is the fastest, the least complex, and the least power consuming way to do (Big Data) computing?

Answer: Rather than writing one program to control the flow of data through the computer, one has to write a program to configure the hardware of the computer, so that input data, when it arrives, can flow through the computer hardware in only one way - the way how the computer hardware has been configured. This is best achieved if the serial part of the application (the transactions) continues to run on the ControlFlow host and the parallel part of the application is migrated into a DataFlow accelerator. A DataFlow part of the application does (parallel) Big Data crunching and execution of loops.

The early works of Dennis [13] and Arvind [14] could prove the concept, but could not result in commercial successes for three reasons: (a) Reconfigurable hardware technology was not yet ready. Contemporary ASIC was fast enough but not reconfigurable, while reconfigurable FPGA was nonexistent; (b) System software technology was not yet ready. Methodologies for fast creation of system software did exist, but effective tools for large scale efforts of this sort did not; and (c) Applications of those days were not of the Big Data type, so the streaming capabilities of the DataFlow computing

model could not generate performance superiority. Recent measurements show, that, currently, Maxeler can move internally over 1 TB of data per second [15].

Programming model

Each programming model is characterized with its quantity and quality. The quantity and quality aspects of the Maxeler DataFlow model, as one of the currently best evaluated, are explained in the next two paragraphs, based on Figure 1. Other DataFlow programming initiatives exist [16] that follow similar approaches as Maxeler systems. To the best of our knowledge Maxeler is the leading player on the field and employs the most advanced and flexible model. For that reason we are using the Maxeler system platform for the presentation of the DataFlow programming model (one of many possible).

Quantitatively speaking, the complexity of DataFlow programming, in the case of Maxeler, is equal to $2n + 3$, where n refers to the number of loops migrated from the ControlFlow host to the DataFlow accelerator. This means, the following programs have to be written:

- One kernel program per loop, to map the loop onto the DataFlow hardware;
- One kernel test program per loop, to test the above;
- One manager program (no matter how many kernels there are) to move data:
  (1) Into the DataFlow accelerator,
  (2) In between the kernels (if more than one kernel exists), and

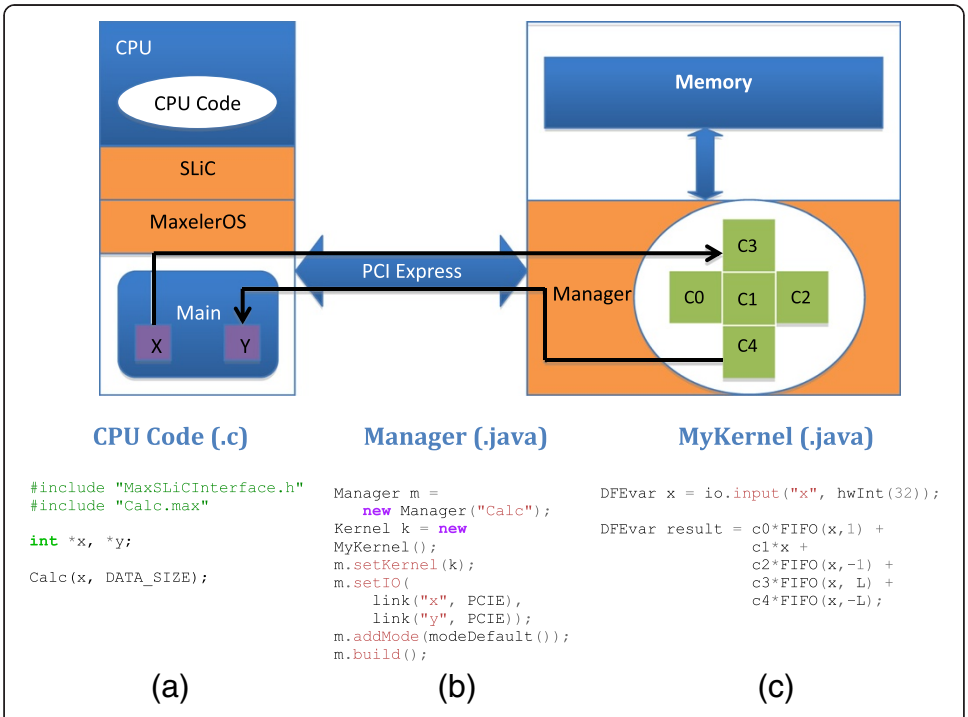

Figure 1 An example of the Maxeler DataFlow programming model: (a) Host code, (b) Manager code, and (c) Kernel code (a single kernel case): 2D convolution. Legend: SLiC = Compiler support for selected domain specific languages and customer applications. DFEvar = keyword used for defining the variables that internally flow through the configured hardware (in contrast to standard Java variables used to instruct the compiler). FIFO = First In First Out.

(3) Out of the DataFlow accelerator;
- One simulation builder program, to test code without the time consuming migration into the binary level;
- One hardware builder program, to exploit the code on the binary level.

In addition, in the host program (initially written in Fortran, Hadoop, MapReduce, MathLab, Matematika, C++, or C), instead of each migrated loop, one has to include a streaming construct (send data + receive results), represented by automatically generated C function Calc(x, DATA_SIZE, see Figure 1 (a).

Qualitatively speaking, the above quantity (2n + 3) is not any more difficult to realize because of the existence of a DSL (domain specific language) like MaxJ (an extension of standard Java with over 100 new functionalities). Figure 2 shows how a complex Big Data processing problem can be realized in MaxJ code. Note that the programming model implies the need for existence of two types of variables: (a) Standard Java variables, to control compile time activities, and (b) DFE (DataFlow Engine) variables, which actually flow through configured hardware (denoted with the DFE prefix in the examples of figures 1 and 2). The programming of the DataFlow part of the code is largely facilitated through the use of appropriate Java extensions.

Research design and methodology

Bare speed is definitely neither the only issue of importance nor the most crucial one [17]. Consequently, the TOP500 ranking should not concentrate on only one issue of importance, no matter if it is speed, power dissipation, or size (the size includes hardware complexity in the widest sense); it should concentrate on all three issues together, at the same time.

In this paper we argue that the best methodology for TOP500 benchmarking should be based on the holistic performance measure H ($T_{BigData}$, N_{1U}) defined as the number of 1U boxes (N_{1U} = one rack units or equivalent) needed to accomplish the desired execution time using a given Big Data benchmark. Instead of using theoretical measures of size and volume, we have opted in this paper for a more practical measure, which is related to international standardization efforts: The size of a 1U box. Issues like power dissipation (monthly electricity bill), and the physical size of the equipment

HostCode (.c)

```
for(t = 1; t < tmax; t++) {
  // Set-up timestep
  if(t < tsrc) {
    source = generate.soruce.wavelet(t);
    maxlib.stream.region_from_host(
                maxlib, "source", source
                srcx, srcy, srcz,
                srcx+1, srcy+1, srcz+1);
  }
  maxlib.stream_from_dram(maxlib, "curr",
curr_ptr);
  maxlib.stream_from_dram(maxlib, "prev",
prev_ptr);
  maxlib.stream_earthmodel_from_dram(maxlib,
dvv_array);

  maxlib.stream_to_dram(maxlib, "next", next_ptr);

  maxlib_run(maxlib); // Execute timestep

  swap_buffers(prev_ptr, curr_ptr, next_ptr);
 }
```

FDKernel (.java)

```
public class IsotropicModelingKernel extends FDKernel {
   public IsotropicModelingKernel(FDKernelParameters p) {
        super(p);
        Stencil stencil =
           fixedStencil(-6, 6, coeffs, 1/8.0);
        DFEVar curr = io.wavefieldInput("curr", 1.0, 6);
        DFEVar prev = io.wavefieldInput("curr", 1.0, 0);
        DFEVar dvv = io.earthModelInput("dvv", 9.0, 0);
        DFEVar source = io.hostInput("source", 1.0n 0);

        DFEVar I =
           convolve(curr, ConvolveAxes.XYZ.stencil);

        DFEVar next = curr*2 - prev + dvv + I + source;

        io.wavefieldOutput("next", next);
    }
}
```

Figure 2 An example of the Host and Kernel code.

(the assumption here is that the equipment size is proportional to the hardware complexity, and to the hardware production cost) are implicitly covered by H ($T_{BigData}$, N_{1U}) (every 1U box has limited power dissipation and size). Selection of the performance measure H is coherent with the TPA concept introduced in [18] and described in Figure 3.

Note that the hardware design cost is not encompassed by the parameter A, which encompasses only the hardware production cost, and causes that the above defined H formula represents an upper bound for ControlFlow machines and a lower bound for DataFlow machines. This is due to the fact that ControlFlow machines are built using the Von Neumann logic, which is complex to design (execution control unit, cash control mechanism, prediction mechanisms, etc.), while the DataFlow machines are built using the FPGA logic, which is simple to design; mostly because the level of design repetitiveness is extremely high, etc. The latter is beneficiary for many Big Data problems, where a large amount of data is continuously processed through the use of relatively simple operations.

As indicated in the previous paper [7], the performance measure H puts PetaFlops out of date, and brings PetaData into the focus. Consequently, if the TOP500 ranking was based on the performance measure H, DataFlow machines would outperform ControlFlow machines. This statement is backed up with performance data presented in the next section.

Results

A survey of recent implementations of various algorithms using the DataFlow paradigm can be found in [19]. Future trends in the development of the DataFlow paradigm can be found in [20]. For comparison purposes, future trends in the ControlFlow paradigm can be found in [21].

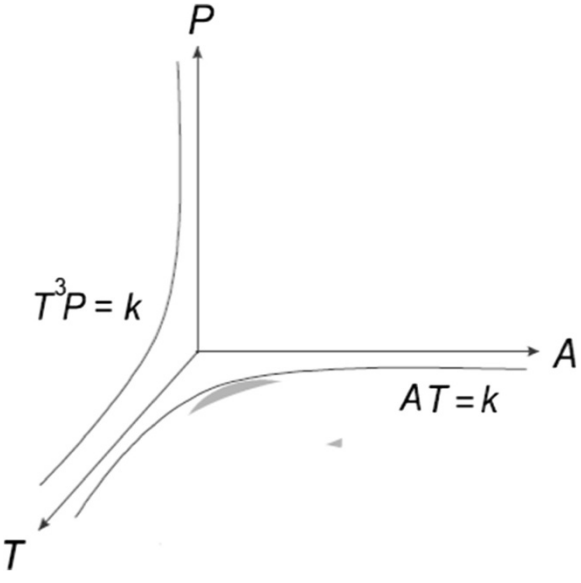

Figure 3 The TPA (Time, Power, and Area) concept of the optimal computer design. Design optimizations have to optimize the three essential issues jointly: T = Time, P = Power, and A = Area (complexity of a VLSI chip).

Some of recent implementations of various DataFlow algorithms interesting within the context of the performance measure H are summarized below.

(1) Lindtjorn et al. [1], proved: (T) That one DataFlow node has the performance equivalent to about 70 twin server Nehelem CPU machines and to 14 two card Tesla GPU machines (application: Schlumberger, GeoPhysics), (P) Using a 150 MHz FPGAs, and (A) Packaged as 1U.
The algorithm involved was Reverse Time Migration (RTM).
Starting from Acoustic wave equation:

$$\frac{\partial^2 u}{\partial t^2} = v^2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right)$$

where u is acoustic pressure and v is velocity.

(2) Oriato et al. [2], proved: (T) That two DataFlow nodes have the performance equivalent to more than 1,900 3 GHz X86 CPU cores (application: ENI, The velocity-stress form of the elastic wave equation), (P) Using sixteen 150 MHz FPGAs, and (A) Packaged as 2U.
The algorithm involved (3D Finite Difference) was:

$$\frac{\partial v\left(\overrightarrow{\chi},t\right)}{\partial t} - b\left(\overrightarrow{\chi}\right)\frac{\partial \sigma ij\left(\overrightarrow{\chi},t\right)}{\partial \chi_j} = b\left(\overrightarrow{\chi}\right)\left[f_i\left(\overrightarrow{\chi},t\right) + \frac{\partial m_{ij}^a\left(\overrightarrow{\chi},t\right)}{\partial \chi_j}\right],$$

$$\frac{\partial \sigma ij\left(\overrightarrow{\chi},t\right)}{\partial t} - \lambda\left(\overrightarrow{\chi}\right)\frac{\partial v_k\left(\overrightarrow{\chi},t\right)}{\partial \chi_k}\delta_{ij} - \mu\left(\overrightarrow{\chi}\right)\left[\frac{\partial vi\left(\overrightarrow{\chi},t\right)}{\partial \chi j} + \frac{\partial v_j\left(\overrightarrow{\chi},t\right)}{\partial \chi_i}\right] = \frac{\partial m_{ij}^s\left(\overrightarrow{\chi},t\right)}{\partial t}.$$

where λ and μ are the so-called Lamé parameters describing the elastic properties of the medium, σ is the stress and f is the source function (driving force).

(3) Mencer et al. [8] proved: (T) That one DataFlow node has the performance equivalent to more than 382 Intel Xeon 2.7 GHz CPU cores (application: ENI, CRS 4 Lab, Meteorological Modelling), (P) Using a 150 MHz FPGAs, and (A) Packaged as 1U.
The algorithm involved (Continuity (1) and (2) and thermal energy (3) equations) was:

$$\frac{\partial p_s}{\partial t} = \int_0^1 \nabla\cdot\left(\vec{V}_h\frac{\partial p}{\partial \sigma}\right)d\sigma \tag{1}$$

$$\frac{\partial q}{\partial t} = -\frac{u}{ah_x}\frac{\partial q}{\partial \lambda}\frac{v}{a}\frac{\partial q}{\partial \phi} - \sigma\frac{\partial q}{\partial \sigma} + F_q \tag{2}$$

$$\frac{\partial \theta}{\partial t} = -\frac{u}{ah_x}\frac{\partial \theta}{\partial \phi} - \sigma\frac{\partial \theta}{\partial \sigma} + F_\theta \tag{3}$$

Where $\overrightarrow{V_h}$ is the horizontal wind in vector form, for which Cartesian components are u and v, terms F_u, F_v, F_q and F_θ represent contributions to the tendencies from the parameterization of physical processes such as radiation, convection, dry adiabatic adjustments, surface friction, soil water and energy balance, large scale

precipitation and evaporation. The PE describes the time evolution for the five prognostic variables u, v, p_s, q and θ.

(4) Stojanović et al. [9] proved: (T) That one DataFlow node has the performance of about 10 i7 CPU cores, (P) Power reduction of about 17, and (A) Packaged as 1U. The algorithm involved (Gross Pitaevskii equation) was:

$$ih\frac{\partial}{\partial t}\Phi(\mathrm{r},t)=\left(-\frac{h^2\nabla^2}{2m}+V\,ext(\mathrm{r})+g|\Phi(\mathrm{r},t)|^2\right)\Phi(\mathrm{r},t)$$

where m is the mass of the boson, r is the coordinate of the boson, V_{ext} is the external potential, g is coupling constant and Φ is wave function.

(5) Chow et al. [3] proved: (T) That one DataFlow node has the performance of about 163 quad core CPUs, (P) Power reduction of about 170, and (A) Packaged as 1U. The algorithm involved (Monte Carlo simulation) was:

$$I\approx\langle f\,H\rangle N=\frac{1}{N}\sum_{i=1}^{N}\left(\overrightarrow{\chi_i}\right)$$

where $\rightarrow x_i$ is the input vector, N is the number of sample points, I is approximated expected value, and ⟨fH⟩N is the sampled mean value of the quantity.

(6) Arram et al. [10] proved: (T) That one DataFlow node has the performance of about 13 Intel X5650 20 core CPUs and about 4 NVDIA GTX 580 GPU machine, (P) Using one 150 MHz FPGAs, and (A) Packaged as 1U.
The algorithm involved (Genetic Sequence Alignment) was based on FM-index. This index combines the properties of suffix array (SA) with the Burrows-Wheeler transform (BWT).
SA interval is updated for each character in pattern Q, moving from the last character to the first:

$$k_{new}=c(\chi)+s(\chi,k_{current}-1)$$

$$l_{new}=c(\chi)+s(\chi,l_{current}-1)$$

where pointers k and l are respectively the smallest and largest indices in the SA which starts with Q, $c(x)$ (frequency) is the number of symbols in the BWT sequence that are lexicographically smaller than x and $s(x, i)$ (occurrence) is the number of occurrences of the symbol x in the BWT sequence from the 0th position to the i^{th} position.

(7) Guo et al. [11] proved: (T) That one DataFlow node has the performance of about 517 Intel i3 2.93 GHz CPU cores and of about 28 GPU machines, (P) Using one 150 MHz FPGA, and (A) Packaged as 1U.
The algorithm involved (Gaussian Mixture Models) was:

$$p(\mathrm{X}|\lambda)=\sum_{i=1}^{M}w_i g\left(\mathrm{X}|\mu i,\sum\nolimits_i\right)$$

where x is a d-dimensional continuous-valued data vector (i.e. measurement or features), w_i, i = 1, …, M, are the mixture weights, and g(x|ě$_i$, Ó$_i$), i = 1, …, M, are the component Gaussian densities.

(8) Kos et al. [12] proved: (T) That one DataFlow node has the performance of between 100 and 400 Intel Core2 Quad 2.66 GHz CPU cores, (P) using one 200 MHz FPGA, and (A) Packaged as 1U.

The algorithm involved was network sorting. An example of the simple sorting network is given in the Figure 4.

Size of the sorting network, the number of comparators needed, to sort N numbers is:

$$S = \frac{N \cdot \log_2 N \cdot (\log_2 N + 1)}{}$$

The achieved speed-up depended on N and the bit-size of numbers being sorted (between 8 and 64 bits).

Conclusion

The viewpoint presented in this paper sheds more light on the recent development of the DataFlow computing concept (more details can be found in [20-22]). The DataFlow computing paradigm requires new ways of thinking and new ways of programming. In general it redefines the subordination of program and data; instead of writing a program that controls how the data flows, the data flow defines the way a program is written.

DataFlow computing excells with applications which are having high repetettivenes of operation and some level of data reusability within the operations. The latter is particularly beneficiarry for many BigData problems, where a large amount of data is repetetively processed through the use of relatively simple operations.

The newly presented benchmarking methodology performance measure H (defined as the number of 1U boxes needed to accomplish the desired execution time using a given Big Data benchmark), would considerably reorder the TOP500 list. If the TOP500 ranking was based on the performance measure H, DataFlow machines would outperform ControlFlow machines. This statement is backed up with the presented performance results. The results show that when using DataFlow computers, instead od ControlFlow computers, time, energy, and/or space can be saved.

The above can be of great interest to those who have to make decisions about future developments of their Big Data centers. It also opens up a new important problem: The need for a development of a public cloud of ready-to-use Big Data applications.

The only remaining question is: can a Big Data application be broken in to a set of tasks and operations that are easily mappable into a DataFlow execution graph for a FPGA structure? We argue that for most Big Data applications the answer is positive!

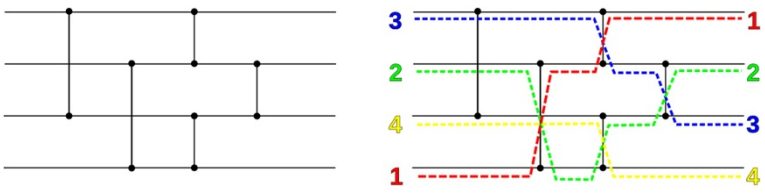

Figure 4 An example of a simple sorting network for sorting four input values with five comparators. Each comparator connects two wires and emits higher value to the bottom wire and lower value to the top wire. Two comparators on the left and two comparators in the middle can work in parallel. Parallel operation of this sorting network sorts the input numbers in three steps.

Comment

This paper was prepared in response to over 100 e-mail messages with questions from the CACM readership inspired by our previous CACM contribution [7].

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
NT, VM, JS, and AK proposed a new methodology for benchmarking Big Data applications, which takes into account not only the execution time, but also the power and space, needed to complete the task. All authors read and approved the final manuscript.

Author details
[1]School of Electrical Engineering, University of Belgrade, Belgrade, Serbia. [2]Mathematical Institute of the Serbian Academy of Sciences and Arts, Belgrade, Serbia. [3]School of Electrical Engineering, University of Ljubljana, Ljubljana, Slovenia.

References

1. Flynn M, Mencer O, Milutinovic V, Rakocevic G, Stenstrom P, Trobec R, Valero M (2013) Moving from petaflops to petadata. In: Communications of the ACM, vol 5, 56th edn. ACM, New York, NY, USA, pp 39–42
2. Dennis J, Misunas D, 'A Preliminary Architecture for a Basic Data-Flow Processor', Proceedings of the ISCA '75 the 2nd Annual Symposium on Computer Architecture, ACM, New York, NY, USA, 1975, pp. 126–132
3. Agerwala T (1982) Arvind, –., "data flow systems: guest Editors' introduction", *IEEE*. Computer 15(2):10–13
4. Mencer O, "Multiscale Dataflow Computing", Proceedings of the Final Conference Supercomputacion y eCiencia, Barcelona Supercomputing Center, SyeC, Barcelona, Catalonia, Spain, May 27 - May 28, 2013
5. Resch M, "Future Strategies for Supercomputing", Proceedings of the Final Conference Supercomputacion y eCiencia, Barcelona Supercomputing Center, SyeC, Barcelona, Catalonia, Spain, May 27 - May 28, 2013
6. Flynn M, "Area - Time - Power and Design Effort: The Basic Tradeoffs in Application Specific Systems", Proceedings of the 2005 IEEE International Conference on Application-Specific Systems and Architecture Processors (ASAP'05), Samos, Greece, July 23-July 25, 2005
7. Salom J, Fujii H, "Overview of Acceleration Results of Maxeler FPGA Machines", IPSI Transactions on Internet Research, July 2013, Volume 5, Number 1, pp. 1–4
8. Maxeler Technologies FrontPage, http://www.maxeler.com/content/frontpage/, London, UK, October 20, 2011.
9. Patt Y (2009) Future microprocessors: what must We do differently if We Are to effectively utilize multi-core and many-core chips? IPSI Transactions on Internet Res 5(1):1–9
10. Lindtjorn O, Clapp G, Pell O, Mencer O, Flynn M, Fu H, "Beyond Traditional Microprocessors for Geoscience High-Performance Computing Applications,"IEEE Micro, Washington, USA, March/April 2011, Vol. 31, No. 2, pp. 1–9
11. Oriato D, Pell O, Andreoletti C, Bienati N, FD Modeling Beyond 70 Hz with FPGA Acceleration, "Maxeler Summary, http://www.maxeler.com/media/documents/MaxelerSummaryFDModelingBeyond70Hz.pdf, Summary of a talk presented at the SEG 2010 HPC Workshop, Denver, Colorado, USA, October 2010
12. Mencer O, "Acceleration of a Meteorological Limited Area Model with Dataflow Engines", Proceedings of the Final Conference Supercomputacion y eCiencia, Barcelona Supercomputing Center, SyeC, Barcelona, Catalonia, Spain, May 27 - May 28, 2013
13. Stojanovic S et al., "One Implementation of the Gross Pitaevskii Algorithm", Proceedings of Maxeler@SANU, MISANU, Belgrade, Serbia, April 8, 2013
14. 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), ACM, Monterey, CA, USA, February 2012, pp. 57–66
15. Arram J, Tsoi K H, Luk W, Jiang P, "Hardware Acceleration of Genetic Sequence Alignment", Proceedings of 9th International Symposium ARC 2013, ACM, Los Angeles, CA, USA, March 25–27, 2013. pp. 13–24
16. 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
17. Kos A, Ranković V, Tomažič S, "Sorting Networks on the Maxeler Data Flow Super Computing Systems", Advances in Computers, Elsevier, 225 Wyman Street, Waltham, MA 02451, USA, 2014
18. Flynn M, Mencer O, Greenspon I, Milutinovic V, Stojanovic S, Sustran Z, "The Current Challenges in DataFlow Supercomputer Programming", Proceedings of ISCA 2013, ACM, Tell Aviv, Israel, June 2013
19. Seebode C, Ort M, Regenbrecht C, Peuker M (2013) BIG DATA infrastructures for pharmaceutical research. IEEE International Conference on Big Data, California
20. Ayhan S, Pesce J, Comitz P, Sweet D, Bliesner S, Gerberick G, "Predictive analytics with aviation big data", Integrated Communications, Navigation and Surveillance ICNS Conference, 2013, Edinburgh, UK
21. Jinglan Zhang, Kai Huang, Cottman-Fields M, Truskinger A, Roe P, Shufei Duan, Xueyan Dong, Towsey M, Wimmer J, "Managing and Analysing Big Audio Data for Environmental Monitoring", IEEE 16th International Conference on Computational Science and Engineering CSE Conference, 2013, Sydney, Australia
22. Convey Computer Web Page: http://www.conveycomputer.com/technology/, October 31 2014