Evaluating the Energy Efficiency of Reconfigurable Computing Toward Heterogeneous Multi-Core Computing

Fabian Nowak
Karlsruhe Institute of Technology
Chair for Computer Architecture and Parallel Processing
76128 Karlsruhe, Germany
Email: nowak@kit.edu

***Abstract*—Future exascale systems need to have a much better performance-to-power ratio than today's systems. Accelerators are a promising approach to pave this path by more energy-efficient computing. We show some early results of our investigations toward energy efficiency of reconfigurable and heterogeneous computing against multi-core processors for special applications. The results are supported by a general framework and toolchain for early evaluation of potential benefits of reconfigurable hardware. As a result, heterogeneous systems based on reconfigurable hardware, efficient data exchange mechanisms, data-driven and component-based programming, and task-parallel execution can help achieve power-efficient exascale systems in future.**

I. Introduction

Multi-core microprocessors nowadays feature a thermal design power (TDP) of more than 100 Watt. Not only is energy consumption high, but it is also hard to cool down the components. New solutions are required to lower average and maximum energy consumption and to also distribute the heat spots along the processor chip. Hardware accelerators can fulfill a lot of tasks not only faster, but even much more energy-efficiently, i.e., consuming less energy in the same time or consuming more energy while also being unproportionally faster, or more formally:

$$energy\ efficiency := \frac{ratio_{speedup}}{ratio_{energy}}$$

. It is therefore of paramount importance to evaluate other resources as alternatives to microprocessor cores for energy-efficient execution in order to someday yield exascale computing systems [6]. In addition, other means than data-parallel programming and execution must be found, evaluated and propagated to make more use of the number of cores already present in nowadays processors. For example, OpenMP 3.0 already supports task-parallel execution. Ideally, such other means will allow seamless migration of program parts from microprocessor cores to accelerators and vice versa.

Apart from energy consumption, another issue with today's microprocessors is that many parts such as floating point units, multimedia or SIMD streaming extensions are unused [13]. Reconfigurable hardware instead of such hard-wired special units allows to dynamically "load" required or useful hardware so that chip area will always be used. Field programmable gate arrays (FPGAs) contain lots of reconfigurable hardware and can therefore serve as evaluation candidates. t

In this paper, we employ the Convey HC-1, a heterogeneous system equipped with four user-programmable FPGAs, for our investigations toward energy-efficient computing. The FPGAs are connected to the host microprocessor via the Front-Side Bus (FSB), thereby posing a tightly coupled heterogeneous system that can serve as a basis and evaluation platform for future architectures of multi- or many-core processors. We also employ a 2-socket Intel Westmere X5670 system with six cores each and HyperThreading as homogeneous multi-core system, providing up to 24 hardware threads in total, to compare coprocessor-extended heterogeneous computing against.

After giving an overview of related work in Section II, we show in Section III three implementations: a kernel for the bioinformatics application HHblits in reconfigurable hardware based on previous work [2], [8] and its extension toward heterogeneous computing; a micro-programmed, data-driven, task-parallel, component-based and domain-adaptive framework in software; and the implementation of said framework for leveraging FPGAs based on previous work [7]. Our evaluation in Section IV shows that reconfigurable computing can be up to 24× more energy-efficient than legacy computing on microprocessors. We draw the conclusions in Section V, giving some final remarks on possible heterogeneous microprocessors to come in the future.

II. Related Work

A solution to low usage of many parts of the microprocessor [13] is to distribute work onto all available cores to execute in a data-parallel fashion. OpenMP and Intel Array Building Blocks yield high processor utilization and high speedups this way, with the latter also exploiting SIMD streaming extensions (SSE) and thereby even more of a chip's resources. When memory bandwidth cannot provide enough data to the processing cores for the application to scale only by means of exploiting data-level parallelism, an obvious solution is to bring in task parallelism, i.e., executing different functions of an application concurrently, hopefully reusing already loaded or cached data. Also, the nodes of an application's directed acyclic graph representation can execute independently. Intel's Cilk Plus [12], OpenMP 3.0 and StarSS [10] are promising

programming models to exploit task parallelism in multi- and manycore systems.

Another solution to tackle dark silicon issues [13] is to provide heterogeneous cores. Basically, with MultiAmdahl the allocation problem of which and how many resources should be integrated on a chip, given a specific target application, has been solved [14]. Now what remains to be investigated and qualified are programmability and energy efficiency of reconfigurable hardware-based accelerators that represent a potential "specialized horseman".

Comparing the FPGA-based heterogeneous computing system Convey HC-1 against a GPU-based system, Bakos finds the HC-1 $5\times$ less energy-efficient [1]. However, we expect more benefit from heterogeneous computing with reconfigurable hardware. Therefore, we evaluate our bioinformatics FPGA-based accelerator for ungapped score calculations as used by HHblits [11]. We also shortly present our task-based programming model for both homogeneous and heterogeneous resources and upon this basis, we evaluate energy efficiency of task-based programming and computing in heterogeneous environments.

III. Implementations

We present three different implementations to evaluate the fitness of existing homogeneous and heterogeneous computing systems in regard to required energy efficiency for exascale computing.

A. Accelerating Ungapped Score Calculations for Bioinformatics Application

HHblits [11] is a bioinformatics application to find homologous sequences among a query protein string and a database of reference protein strings. HHblits employs prefiltering of the millions of database entries by means of a striped Smith-Waterman implementation [3], which has been implemented on FPGA recently [2], [8].

HHblits works as depicted in Figure 1: For the query protein sequence, at first a Hidden Markov Model (HMM) is created. Then, a profile is created upon the query HMM. The profile is used to prefilter suitable entries from the large protein database. This is done by first calculating a so-called *ungapped score* without considering any gaps between the query profile and the database reference sequence, i.e, the query and the reference are quickly evaluated with regard to contiguously matching parts. Those sequences passing this ungapped-score prefilter are roughly checked whether they would also deliver a suitable score when accepting gaps between subsequences. The large number of millions of database entries is thereby decimated to only some ten thousands. These "few" remaining protein HMMs are regularly compared against the query HMM. To obtain more suitable candidates, additional runs can be used to find more results by "widening" the query so that more reference sequences can match the query. If another run is intended or required, then an update of the initial HMM by the information from the accepted HMMs is needed and accordingly performed by HHblits. The most time-consuming part is the prefiltering step, with the first step (ungapped score calculation) accounting for 1601.9 seconds of the total application time of 1626.2 seconds when not

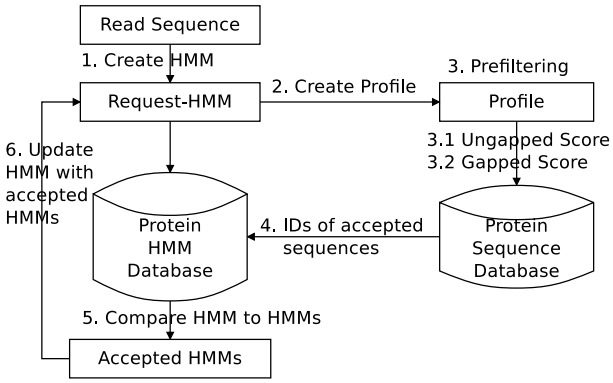

Fig. 1. Comparing a query sequence against a protein sequence database within HHblits.

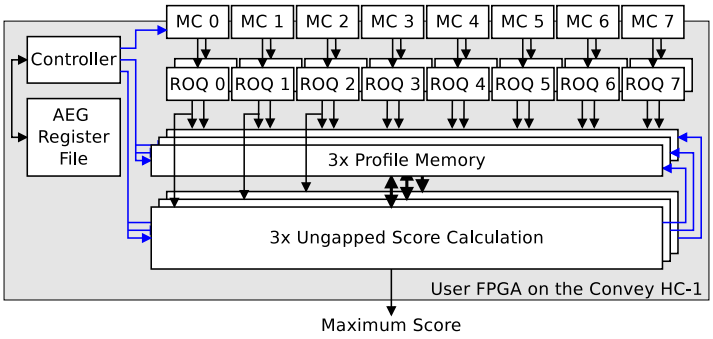

Fig. 2. Implementation of the first prefiltering step of HHblits on the four FPGAs on the Convey HC-1. (MC – Memory Controller, ROQ – Read-Order Queue, AEG registers – Application Engine General registers).

employing Farrar's efficient striped SSE implementation for the prefiltering step when performing only a single run. When prefiltering is accelerated via Farrar's SSE implementation, then prefiltering takes roughly 67 % of the total application time of 74.1 seconds for one run. If more runs are performed, the proportion of the ungapped score calculations to overall application execution time decreases.

Key to successful acceleration is enhancing data reuse. So in contrast to the SSE execution, our implementation [2], [8], shown in Figure 2, of the first prefiltering step for calculating a locally ungapped score first loads the entire profile in parallel over all 16 memoy controllers (MCs) and then streams the query sequence over one MC per ungapped score calculation unit (UCU) that can then heavily reuse data from different parts of the profile. The profile is shared by all instantiated UCUs. Apart from data reuse, exploiting massive parallelism is required to successfully speedup an application despite the low processing frequency of FPGAs. We do so by employing the four FPGAs of the Convey HC-1, with each FPGA instantiating three UCUs for calculating three different ungapped scores in parallel. These calculations are made up of 16 parallel, 8-Byte-processing units; and the maximum is calculated in a reduction tree over the 128 calculated individual scores and is implemented as a pipeline. In total, we exploit $4{\cdot}3{\cdot}16{\cdot}8 = 1536$ fold parallelism and can output a new maximum score per FPGA every cycle, while the SSE version suffers from the non-parallel maximum calculations that introduce undesired additional delay.

The two cores on the host processor of the HC-1 can also perform useful work during coprocessor execution: the second prefiltering step depends only on the individual results of previous single ungapped score calculations and can therefore be carried out in an overlapped, task-parallel fashion.

B. Task-Parallel, Data-Driven and Component-Based Programming and Execution (TDDcomp)

As second application domain to be analyzed toward performance and efficiency of homogeneous and heterogeneous computing systems we investigate numerical applications. As can be seen from Figure 3 of the conjugate gradient method (CG method), not only are some functions completely independent from others, but many functions can also be executed in an overlapped fashion, using already calculated vector entries to compute the next depending functions. This is interesting and helpful for two reasons: first, the available bandwidth to/from external memories is too low to provide enough data to the processing cores as soon as data become too large to fit into the caches so that some cores are only waiting for data. Second, newly calculated data that would normally be replaced in the caches is reused now.

We implemented a set of linear algebra routines in a data-driven fashion. The functions are executed as separate POSIX threads and consume data from matrices or vectors and exchange progress information in order to not use yet invalid data. From a performance point of view, it proved to be best to split matrices into ten blocks, so that communication would not occur on a per-datum or per-line basis. Implemented and evaluated mechanisms ensure that the function threads do not consume valuable processing time as long as the required input data are not available. When processing data blocks, OpenMP is employed to exploit data parallelism in addition. From an application programmer's perspective, programming resembles that of sequential programming. The programmer has to care that as much task parallelism can be exploited as possible. During development, our investigations revealed that matrix data and structures should be allocated rather sooner than later in order to leverage more task parallelism after the allocation phase; and the structures should be freed as late as possible in order to not pollute the caches during ongoing more useful calculations. As it does not make sense to poll continuously for new data, we evaluated sleeping against waiting on semaphores. Upon the results gained, we decided to employ semaphores. The final implementation and configuration of the routines allow processing of large vectors with competitive speed to data-parallel execution for the CG method.

C. Implementation of a Micro-Programmable TDDcomp Framework on the Convey HC-1

When employing FPGAs to accelerate applications partially or even entirely, one of the most crucial problems is data transfer due to limited off-chip bandwidth and lack of hardware caches. Therefore, we implemented a streaming-oriented, component-based architecture framework similar to the one proposed above in Subsection III-B as depicted in Fig. 4. To exchange data between components and external memories in an efficient manner, a central FIFO buffer set connects the components much like a crossbar. Data flow is

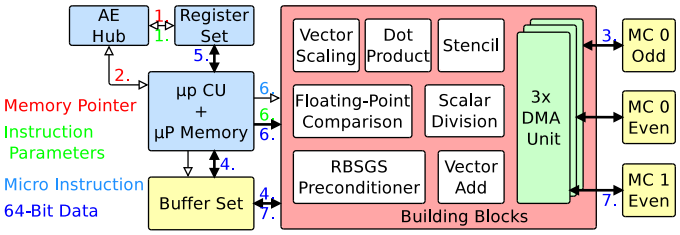

Fig. 4. Implementation of the architecture framework on the Convey HC-1.

controlled via a user-supplied micro program. This framework allows application programmers to provide and exploit custom, application-targeted accelerators in reconfigurable hardware by developing micro programs only instead of describing hardware at low level and instead of employing high-level language converters. For example, the CG method can be accelerated up to ten times in comparison to the Convey HC-1's host processor and the framework even achieves comparable performance to 24 hardware threads on a two-socket Intel Westmere X5670 system [7], although employing only one out of the four available FPGAs.

With special units such as an enhanced-precision dot product implementation, even more speedup can be gained because the algorithm can converge in only half the number of iterations. This architecture framework is hence also a possible approach of how to interconnect multiple cores on a manycore chip and how to have them communicate efficiently via buffer sets in a data flow style. Moreover, the special units represent the case of exploiting reconfigurable logic for custom accelerators, while the numeric functions might also be executed on standard microprocessor cores.

IV. Analysis and Evaluation

We aim at analyzing the fitness of heterogeneous, FPGA-based computing systems as a possible approach for future exascale computing where much more energy-efficient execution than on today's regular, multi-core processor-based computing systems is required. In future, manycore chips might want to integrate reconfigurable logic for faster and less energy-consuming, i.e, more energy-efficient, computing. An early example is the Intel Atom E6x5C [4] with an Altera FPGA connected to the low power core over PCI Express. This way, the dark logic issues [13] might be solvable when integrating special instructions, operations or functions. Also will heat dissipation be less problematic then. We do hence study execution time and energy consumption of the implementations presented in Sect. III on the Convey HC-1 and a two-socket Intel Westmere system. Table I summarizes the maximum power consumption of these systems. Although many processors can be dimmed down to a fraction of the maximum frequency, this is not much different from using FPGAs that are clocked with only 100 MHz to 300 MHz, except that FPGAs can implement special operations at bit-level or with smart data structures so that they also reduce execution time in addition to having a lower power consumption footprint. Some processors can also be boosted to a higher frequency, but this is possible only for a very short time due to heating problems and going rather into the opposite direction than the approach favored in this paper.

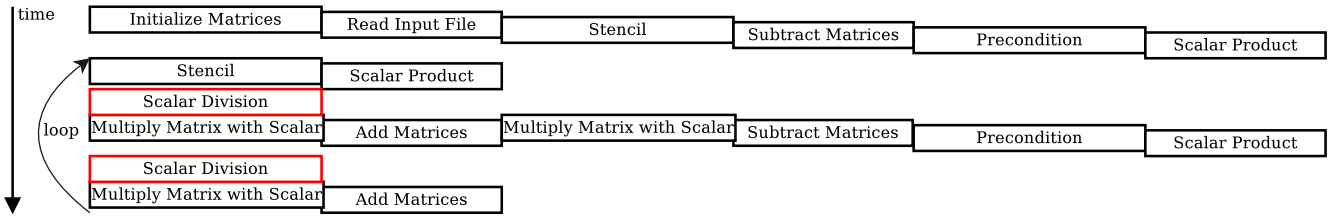

Fig. 3. Graph for the Conjugate Gradient Method indicating exploitable task parallelism. Data flows from left to right, while time advances vertically. Many tasks can execute in an overlapped fashion as they only depend on few previously calculated data.

TABLE I. THERMAL DESIGN POWER (TDP) AND MAXIMUM POWER CONSUMPTION OF THE USED SYSTEMS.

System setup	Intel Xeon 5138 @Convey HC-1	Xilinx Virtex-5 LX330 @Convey HC-1	Convey HC-1	Intel Xeon X5670	2-socket X5670
TDP / maximum	35 W	35.8 W	178.2 W	95 W	190 W

A. Energy Efficiency of HHblits Implementations

The performance of HHblits largely depends on the performance of the ungapped score calculation kernel (60 %-70 % of total execution time [2]). We study the individual aspects of coprocessor-supported execution and task-parallel execution and finally merge these.

1) Ungapped Score Calculation on the FPGA-based Coprocessor: As can be seen from Table II, employing the FPGAs dramatically increases power consumption because as of now, a hardware thread is required to control execution on the FPGAs (fifth row). FPGA power consumption is determined via Xilinx Power Analyzer (XPA), while CPU power consumption is taken from the specified thermal design power (TDP). Also, for the remaining parts of HHblits, power and energy consumption of the standard processor must be accounted for. However, due to shortened overall execution time and because the coprocessor runs only for the ungapped score calculations, but not for the remainder of the application, the total energy efficiency of the heterogeneous system shall be better, as will be seen later.

2) Overlapped Execution of HHblits: We already sketched the approach of overlapping different subtasks of HHblits. As multi-core processors may easily run into the memory wall, it must be investigated whether task-parallelism allows to better exploit the available processing resources so that memory accesses do not become the bottleneck. Figure 5 shows that although not necessarily providing real benefit over data-parallel execution, it is possible to entirely hide the computations of swStripedByte behind the ungapped-score calculation by executing in a task-parallel, component-based manner. The overhead imposed by synchronization must be regarded the culprit for the low advantages of the task-parallel approach for our coprocessor-accelerated version of HHblits. As consequence, more data must be transferred so that memory bandwidth is stressed more, or a different communication model must be used for synchronization.

3) HHblits on the Heterogeneous System: Applying the task-parallel model while also using the coprocessor allows to easily synchronize the ungapped score calculations more tightly with the subsequent swStripedByte because the coprocessor is invoked for a set of sequences only, and swStripedByte can process the previous sequence set then. Again, Table III shows that energy consumption is higher when also processing HHblits on the coprocessor, but due to achieved speedup, the heterogeneous systems performs more than 1.5× more energy-efficiently.

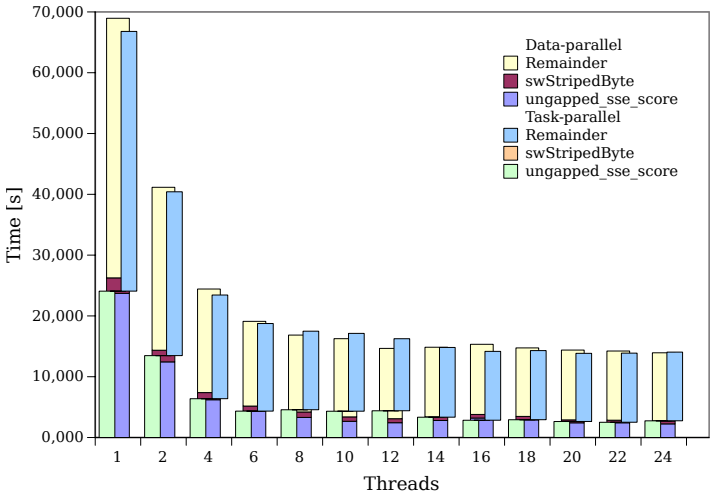

Fig. 5. Data-parallel execution of HHblits on Homogeneous System with two X5670 processors against task-parallel execution, which hides swStripedByte nearly entirely behind ungapped-score calculation and thereby is faster in most cases.

B. Energy Efficiency of Conjugate Gradient Method Implementations

To finally support our thesis that heterogeneous computing based on reconfigurable hardware can provide benefit over computing on homogeneous resources only, we regard energy consumption and efficiency of an FPGA for task-parallel, data-driven computing (TDDcomp) in reconfigurable hardware on the Convey HC-1's coprocessor against a two-socket Intel Xeon X5670 system. A previous implementation [7] of the CG method in the framework serves as basis for these investigations. As indicated by Xilinx Power Analyzer, the FPGA implementation of TDDcomp draws 17.439 W of power. Note that the FPGA implementation includes an extended dot product unit in order to draw more benefit from reconfigurable hardware by arbitrary bit-width implementations. This also reflects the case that special operations (→ "specialized horseman" [13]) are tightly integrated into execution of a program on various resources, such as an exact dot product implementation [9] which is approximated here by the extended-precision unit. Table IV lists the execution time, calculated

TABLE II. POWER CONSUMPTION AS USED IN OUR ANALYSIS MODEL, EXECUTION TIME AND ENERGY CONSUMPTION OF ALL KERNEL CALCULATIONS OF HHBLITS ON THE CONVEY HC-1.

System setup	Power consumption [W]	Kernel execution time [s]	Energy consumption [mWh]
Xeon 5138, 1 Thread (TDP)	17.5	1601.9	7787.0
Xeon 5138, 1 Thread, SSE (TDP)	17.5	50.0	243.1
Xeon 5138, 2 Threads, SSE (TDP)	35	25.1	244.0
Xilinx Virtex-5 LX330 (XPA)	18.561	28.6	147.5
Convey HC-1 with Coprocessor (1 Hardware Thread, 4 FPGAs)	91.744	13.1	333.8

TDP – Thermal Design Power, XPA – Xilinx Power Analyzer

TABLE III. EXECUTION TIME, ENERGY CONSUMPTION AND RATIOS OF CPU EXECUTION, COPROCESSOR-EXTENDED EXECUTION AND OVERLAPPED CPU/COPROCESSOR EXECUTION FOR THE ENTIRE APPLICATION HHBLITS.

	Xeon 5138 (1 Thread, SSE)	With Coprocessor (+ 4 FPGAs)	Overlapped
Execution time [s]	74.585	37.465	34.147
Energy consumption [mWh]	362.566	507.048	507.048
Speedup over CPU	(1.0)	1.991	2.184
Relative energy consumption against CPU	(1.0)	1.398	1.398
Relative energy efficiency against CPU	(1.0)	1.424	1.562

energy consumption and energy efficiency ratio of FPGA-based computing against a 24-hardware-thread system. The FPGA implementation of the conjugate gradient method within the micro-programmable TDDcomp is more than 17 times as energy-efficient as execution on a 24-hardware thread Intel Westmere system.

V. CONCLUSIONS AND FUTURE WORK

We showed that reconfigurable computing can be 17× more energy-efficient than computing on general-purpose multi-core processors. To fully leverage the benefits of heterogeneous, accelerator-based systems, both the kernels ported to accelerators in reconfigurable hardware and the application need to take much care of data locality and data reuse. The task-parallel execution of components in reconfigurable hardware and software allows to exploit such data locality and data reuse by means of FIFO buffers and by the concept of streaming. As a result, our investigations show that upcoming multi- and many-core microprocessors should be heterogeneous by nature, for example by instantiating a couple of FPGA-like resources to provide application-required accelerators on demand. Moreover, the task-parallel execution of components in a streaming fashion requires data buffers for linear data access. Such data buffers should make up another part of microprocessors in addition to the regular caches as the buffers will support both kernels in reconfigurable hardware and in software. With such a setup of multiple processing cores, reconfigurable hardware and custom memory structures on chip, we can expect to meet the required power and energy goals of exascale systems. Future work includes seamless provision of the required accelerators based on previous work toward self-management of heterogeneous systems [5].

ACKNOWLEDGMENTS

The author acknowledges the invaluable contributions of Ingo Besenfelder and Michael Bromberger to the implementations on the Convey HC-1 and also the helpful comments from the reviewers.

REFERENCES

[1] J. Bakos, "High-Performance Heterogeneous Computing with the Convey HC-1," *IEEE Computing in Science & Engineering*, vol. 12, no. 6, pp. 80–87, Nov.-Dec. 2010.

[2] M. Bromberger and F. Nowak, "Parallel Prefiltering for Accelerating HHblits on the Convey HC-1," in *Mitteilungen*, vol. 30. Gesellschaft fr Informatik e. V., Parallel-Algorithmen und Rechnerstrukturen, Sept. 2013, pp. 47–57.

[3] M. Farrar, "Striped Smith-Waterman speeds database searches six times over other SIMD implementations," *Journal of Bioinformatics (Oxford University Press)*, vol. 23, no. 2, pp. 156–161, Jan. 2007.

[4] "Intel® Atom™ Processor E6x5C Series," online, Intel, Dec. 2010, Product Preview Datasheet, Revision 001US. [Online]. Available: http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/atom-e6x5c-datasheet.pdf

[5] M. Kicherer, F. Nowak, R. Buchty, and W. Karl, "Seamlessly Portable Applications: Managing the Diversity of Modern Heterogeneous Systems," *ACM Trans. Archit. Code Optim.*, vol. 8, no. 4, pp. 42:1–42:20, Jan. 2012.

[6] N. Leavitt, "Big Iron Moves Toward Exascale Computing," *IEEE Computer*, vol. 11, pp. 14–17, November 2012.

[7] F. Nowak, I. Besenfelder, W. Karl, M. Schmidtobreick, and V. Heuveline, "A Data-driven Approach for Executing the CG Method on Reconfigurable High-Performance Systems," in *Architecture of Computing Systems*, ser. LNCS, vol. 7767. Springer, Jan. 2013, pp. 171–182.

[8] F. Nowak, M. Bromberger, M. Schindewolf, and W. Karl, "Multi-Parallel Prefiltering on the Convey HC-1 for Supporting Homology Detection," in *Proceedings of the 20th European MPI Users' Group Meeting*, ser. EuroMPI '13. ACM, Sept. 2013, pp. 169–174, international Workshop on Parallelism in Bioinformatics (PBio 2013).

[9] F. Nowak, R. Buchty, D. Kramer, and W. Karl, "Exploiting the HTX-Board as a Coprocessor for Exact Arithmetics," in *Proceedings of the First International Workshop on HyperTransport Research and Applications (WHTRA2009)*. Computer Architecture Group, Institute for Computer Engineering (ZITI), University of Heidelberg, Feb. 2009, pp. 20–29. [Online]. Available: http://www.ub.uni-heidelberg.de/archiv/9114

[10] J. Planas, R. M. Badia, E. Ayguadé, and J. Labarta, "Hierarchical Task-Based Programming With StarSs," *International Journal of High Performance Computing Applications, Sage Publications*, vol. 23, pp. 284–299, August 2009.

[11] Remmert, M., Biegert, A., Hauser, A., and Sding, J., "HHblits: Lightning-fast iterative protein sequence searching by HMM-HMM alignment," *Journal of Nature methods (Nature Publishing Group)*, vol. 9, no. 2, pp. 173–175, Feb. 2012.

[12] A. D. Robison, "Composable Parallel Patterns with Intel Cilk Plus," *Computing in Science and Engineering*, vol. 15, no. 2, pp. 66–71, 2013.

TABLE IV. EXECUTION TIME OF CG METHOD ON COPROCESSOR FPGA (XILINX VIRTEX-5 LX330) AGAINST 2-SOCKET INTEL XEON WESTMERE X5670 SYSTEM, AND THE CORRESPONDING ENERGY CONSUMPTION AND RATIO AS DEFINED IN SECT. I.

	512×512	1024×1024	2048×2048	4096×4096
Execution time on 2 X5670	2.405 s	21.242 s	168.618 s	1896.006 s
Execution time 1 FPGA	2.286 s	16.607 s	145.628 s	1196.383 s
Maximum Energy Consumption, CPU	126.938 mWh	1121.100 mWh	8899.266 mWh	100066.993 mWh
Maximum Energy Consumption, 1 FPGA	22.729 mWh	165.144 mWh	1448.186 mWh	11897.358 mWh
Calculated Energy Consumption, 1 FPGA	11.072 mWh	80.446 mWh	705.444 mWh	5795.476 mWh
Ratio	11.5	13.9	12.6	17.3

[13] M. B. Taylor, "A Landscape of the New Dark Silicon Design Regime," *IEEE Micro*, vol. 33, no. 5, pp. 8–19, 2013.

[14] T. Zidenberg, I. Keslassy, and U. Weiser, "Optimal resource allocation with multiamdahl," *Computer*, vol. 46, no. 7, pp. 70–77, 2013.