Journal of Interconnection Networks
© World Scientific Publishing Company

Introducing the Extremely Heterogeneous Architecture

Shaoshan Liu[1], Won W. Ro[2], Chen Liu[3], Alfredo Cristobal-Salas[4], Christophe Cérin[5], Jian-Jun Han[6], and Jean-Luc Gaudiot [7]

[1]*Microsoft, WA, U.S.A.*
[2]*Yonsei University, Seoul, Korea*
[3]*Clarkson University, NY, U.S.A.*
[4] *Universidad Veracruzana campus Poza Rica, Mexico*
[5]*Université Paris XIII, France*
[6]*Huazhong University of Science & Technology, China*
[7]*University of California, Irvine, CA, U.S.A.*

Received Day Month Year
Accepted Day Month Year

The computer industry is moving towards two extremes: extremely high-performance high-throughput cloud computing, and low-power mobile computing. Cloud computing, while providing high performance, is very costly. Google and Microsoft Bing spend billions of dollars each year to maintain their server farms, mainly due to the high power bills. On the other hand, mobile computing is under a very tight energy budget, but yet the end users demand ever increasing performance on these devices. This trend indicates that conventional architectures are not able to deliver high-performance and low power consumption at the same time, and we need a new architecture model to address the needs of both extremes. In this paper, we thus introduce our Extremely Heterogeneous Architecture (EHA) project: EHA is a novel architecture that incorporates both general-purpose and specialized cores on the same chip. The general-purpose cores take care of generic control and computation. On the other hand, the specialized cores, including GPU, hard accelerators (ASIC accelerators), and soft accelerators (FPGAs), are designed for accelerating frequently used or heavy weight applications. When acceleration is not needed, the specialized cores are turned off to reduce power consumption. We demonstrate that EHA is able to improve performance through acceleration, and at the same time reduce power consumption. Since EHA is a heterogeneous architecture, it is suitable for accelerating heterogeneous workloads on the same chip. For example, data centers and clouds provide many services, including media streaming, searching, indexing, scientific computations. The ultimate goal of the EHA project is two-fold: first, to design a chip that is able to run different cloud services on it, and through this design, we would be able to greatly reduce the cost, both recurring and non-recurring, of data centers\clouds; second, to design a light-weight EHA that runs on mobile devices, providing end users with improved experience even under tight battery budget constraints.

1. Introduction

The rise of cloud computing, big data, and green computing poses new challenges for computer architects. On the one hand, cloud computing and big data imply that large-scale computing and data storage are needed, and lay much stress on computation and data intensiveness. On the other hand, green computing creates a need to implement computing and data storage with severe energy consumption constraints. For a while now, high performance and low energy consumption have been the two sides of a difficult trade-off.

To address this dilemma, we propose a cloud computing and green computing-oriented heterogeneous multicore design: our Extremely Heterogeneous Architecture (EHA), which aims at providing high computation performance, high data throughput, as well as low energy consumption. EHA comprises general-purpose cores and specialized cores in the same chip. As their names imply, general-purpose cores are designed for generic control logic and computations, while the specialized cores focus on accelerating those applications which are frequently invoked or computationally heavy, such as image/media processing, XML parsing, encryption and decryption, *etc*.

The motivation behind the EHA project is to provide the ability to achieve performance and energy efficiency at the same time. We have demonstrated that the combination of specific-cores can achieve both performance and energy efficiency [1, 2, 4, 6, 9] because when an application is not invoked, its corresponding specific-core is in the sleep or shut down mode, leading to more energy saving. Further, with hardware acceleration, multiple software instructions can be integrated into a single hardware instruction [3, 5]. This can shorten the execution time and give more performance improvement. Base on this requirement, the number of specific-purpose cores can be doubled or tripled to accelerate different concurrently executed applications.

The EHA project is a result of a collaboration among several research groups located across the globe, from both academia and industry. It is planned along the following stages:

First, we ported some frequently executed OS/middleware services or high overhead services onto homogeneous many-core design, and study its performance and power consumption. As validation, we implemented XML data parsing in the Intel Single-Chip-Cloud (SCC) 48-core homogeneous system [2]. As a result, it was uncovered that pinned OS/middleware service can lead to great power saving; however, the homogeneous cores may not be able to efficiently execute the dynamic workload, and thus may not achieve great performance efficiency.

Second, since not all services are equally weighted or requested, in order to get the maximal performance gain and energy efficiency, we tailored a specialized design to different services. As we will discuss later, we demonstrated there is a large overhead from the memory side when paring XML data. Considering this, we proposed a memory-side accelerator as a specific core for XML parsing [1, 6]. For the cloud computing scenario, we proposed a JVM garbage collection accelerator to improve the bloated cloud middleware. With the GC accelerator, the overall GC time can be reduced by 60% and the overall execution time can be reduced by 7% [3, 5]. The EHA can not only be used to accelerate cloud computing workloads, but also dynamic mobile workloads. For the mobile embedded scenario, we demonstrated that, with technology progress, a hardware prefetcher may no longer waste energy and can be used in energy-limited mobile systems for both energy and performance efficiency [4].

As the final step, the stage at which the project is currently, we plan to construct a prototype Extremely Heterogeneous Architecture (EHA) chip by integrating the pieces together. This EHA prototype will consist of multiple homogeneous light-weight cores, multiple specialized cores; each of these cores will host a service. It is supposed to have the best balance in performance, energy consumption and programmability. As a next step, we plan to demonstrate the effectiveness of our EHA architecture on networking workloads [15~18].

The initial version of this paper was presented in the 2012 International Symposium on Pervasive Systems, Algorithms, and Networks [25]. Different from the original version, we have extended the paper with more materials. Specifically, in section 7, we discuss how the EHA is used in accelerating networking coding applications; in section 8, we discuss the application of the EHA in cryptographic computations; and in section 9, we briefly introduce other applications that could benefit from the EHA.

2. EHA Architecture

In this section, we provide an overview of the hardware architecture of the EHA which has been designed to provide the best balance in performance, energy consumption and programmability. As shown in Fig.1, the EHA consists of general-purpose cores and specialized cores, including an out-of-order heavy-weight core, multiple homogeneous light-weight cores, multiple GPU cores, multiple ASIC (hard) accelerators, and multiple reconfigurable (soft) accelerators; each of these cores will host a service. General-purpose cores take care of generic control logic and computation.

In the general-purpose core category, the out-of-order processor is the "control core," whose duty is scheduling, assignment, load balance, some basic computation task, *etc*. The in-order processor is the "execution core," who gets the tasks from the "control core" and executes them as scheduled. Specialized cores are dedicated for application acceleration and those targeted applications are always frequently invoked or computationally intensive. In the specialized category, for high-overhead and well established services (browser, file manager, *etc.*), which are generic and static (no major changes over extended periods of time), we can use ASIC cores for acceleration; for services that are less generic and prone to change (*e.g.,* different encryption and decryption algorithms, even future emerging applications), we can use FPGA accelerators which can be modified at runtime to adjust to application needs; for applications that have great parallelism, we can use GPU cores for acceleration, whose processing engine (PE) can work concurrently to explore the potential parallelism. As stated in Fig.1, each core is equipped with a private cache to maximum data locality and improve the memory access; all on-chip cores share the last level cache and communicate each other with the high-speed on chip interconnection network. As manufacturing technology progresses, a single chip can integrate more cores and more cache capacity. EHA can thus scale up well for the number of both generic and specific cores can be adjusted freely.

In Fig. 2, we provide a usage scenario for the EHA: users may need image/media processing; may access cloud storage, whose data will always be transmitted in an XML format; may access the Internet, most of whose applications are built in a JAVA environment; may need to protect user’s bank transactions, where data needs to go through a complex encryption/decryption process. Since such applications are commonly used, each of them is equipped with a specific core for acceleration. As shown in Fig.2, there are four specific cores: garbage collection core (GC), XML parsing core (XML), Image processing core (Image) and encryption/decryption core (MED). At execution time, specific cores are activated when the corresponding application is invoked; once the application terminates, the core goes to the sleep mode and will be shut down after a specific fixed period of sleep. This mechanism can provide great energy saving as well as performance efficiency.

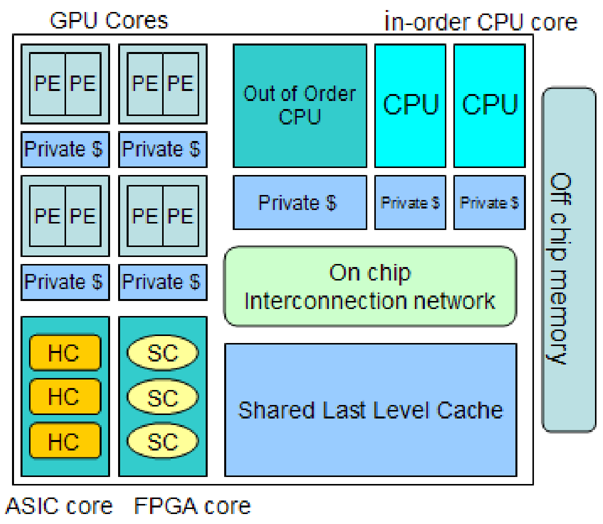

Figure 1: EHA Architecture

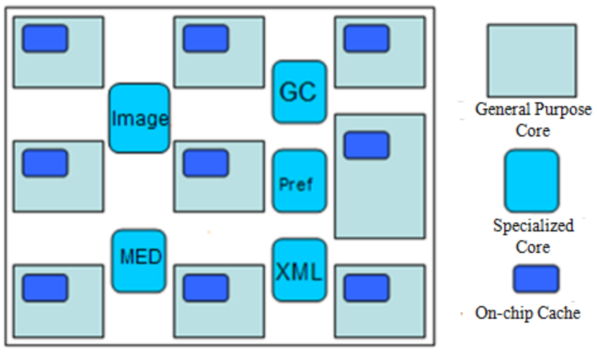

Figure 2: EHA usage scenarios

3. Pinned OS/Middleware Services

In this section, we demonstrate how the EHA can be used to accelerate heterogeneous workloads, and we propose pinning each OS/Middleware service to one core to achieve maximal performance and energy efficiency.

We are now approaching the point where we will be capable of integrating hundreds to thousands of cores on a single chip [10]. However, traditional system software and middleware are not well suited to managing and providing services at such large scales. To improve the scalability and adaptability of operating system and middleware services on future many-core platforms, we proposed the Pinned OS/middleware Services as the implementation of the first step. By porting each OS and middleware service to a separate core, we expect to achieve maximal performance gain and energy efficiency in many-core environments. We target our work at general OS services, all of them are ubiquitous enough and worthy of acceleration in hardware. In future thousand core scenarios, we can turn off or wake up the corresponding cores, depending on the application load. If the service is not needed, its dedicated core will remain in the sleep mode or shut down, with low or no energy consumption. By doing so, we hope to provide a significant FLOPS per Watt ratio to system, greatly reducing the non-recurring cost (hardware investment) and recurring cost (energy bill) of the deployment of servers in the cloud computing datacenter eco-system.

As a case study, we targeted an XML (Extensible Markup Language) data parsing application. Extensible Markup Language (XML) has become a widely adopted standard for data representation and exchange, acting as the important component of system middleware [2, 9]. Although XML can provide benefits like language neutrality, application independency and flexibility, it also comes at the cost of heavy performance overhead [11, 12] due to its verbosity and descriptive nature. A real world example would be Morgan Stanley's Financial Services system, which spends 40% of its execution time on processing XML documents [13]. This situation is only going to worses as XML datasets become larger and more complex.

We have successfully implemented and evaluated the design of the porting of XML parsing service onto an Intel SCC platform [10], configured in a 48-core homogenous system. In our experiments, we chose to perform DOM parsing on six different XML benchmarks and evaluate both their performance and energy behaviors [2, 9]. Our results show that when porting XML service into a homogenous system, we can get a considerable energy reduction. Compared to original non-pinned parsing, porting the XML service onto a homogenous system can yield a reduction of up to 40% in energy. However, when pinning XML parsing services to a dedicated core of the Intel SCC, we can find heavy penalties are incurred on the memory side: for example, the average rate of *duration of pipeline stalled* during data memory reads is 10%, meaning 10% of the total execution time is consumed by memory operations. In the case of the most memory intensive benchmark, *Long*, memory operations account for 23% of the overall execution time. To overcome this drawback, in section 4, we further tailored the XML parsing service core into a specific memory-side hardware accelerator.

4. Memory-Side Acceleration for XML Parsing

As mentioned in the previous section, to alleviate the performance inefficiency of a Pinned OS/middleware service in a homogeneous system, we now perform a further study of XML parsing and propose memory-side acceleration with the incorporation of data prefetching techniques. It can tailor the homogeneous core to the memory access requirements of XML parsing. This is our form of memory-side acceleration for XML parsing. As seen in Fig.3.c, memory-side acceleration can be applied on top of computation-side acceleration, whose combination can act as a Data Front End (DFE) to speed up the XML data parsing together. DFE is one of the specialized cores in EHA architectures, which is responsible for XML data transmission. In Fig. 3.a and Fig. 3.b, we show two different DFE interaction scenarios (interaction between application and interaction with a data warehouse).

In our memory-side accelerator design [1, 6], we use a hardware prefetcher to improve the memory data loading stage of XML parsing. In our experiment, we select eight different hardware prefetchers to accelerate both SAX and DOM parsing. We find that prefetching techniques are very effective on XML parsing workloads, as most prefetchers are able to reduce cache misses by more than 50%. In the best case, prefetcher n3 is able to reduce L2 cache misses by as much as 82% in the SAX parser and 85% in the DOM parser. Such reductions in LLC miss rate can yield performance gains as high as 9.73% and 19.94% for SAX and DOM parsing respectively. These results confirm that memory-side acceleration can be effective regardless of the parsing models. We have also verified its feasibility by checking the impact of its implementation on bandwidth, energy consumption and hardware costs. The results show that memory-side acceleration can yield up to 12.77% of energy saving when implemented in 32nm technology. For the eight accelerators studied, all the hardware costs are within 32 Kbits which is quite a small and reasonable overhead considering modern hardware budgets. As for bandwidth, XML parsing performance is hurt by the latency but not by the throughput of the memory subsystem, thus confirming that memory-side acceleration will not likely result in resource contention of memory bus bandwidth. In conclusion, with the tailored specific design, we can further improve on the performance gains and energy efficiency. It therefore validates our initial assertion that only the featured design can achieve both performance and energy efficiency.

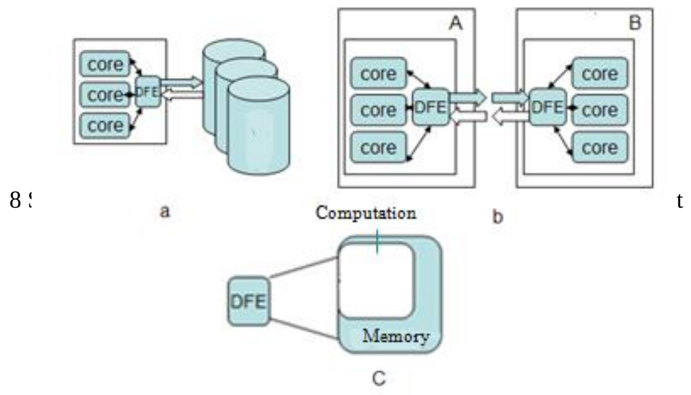

Figure 3: Data Front End

5. Hardware Accelerator for Garbage Collection

In this section, we provide another case study for our EHA. In this case, we use specialized cores to accelerate virtualization workloads, which are very important in cloud environments. Although virtualization technologies significantly benefit cloud computing environments (since the virtual machines provide more features), the middleware layer has become bloated, introducing high overhead [19]. Our ultimate goal is to provide hardware-assisted solutions to improve the middleware performance in cloud computing environments. As a starting point, we have designed, implemented, and evaluated specialized hardware instructions to accelerate garbage collection (GC) operations. We select GC because it is a common component in virtual machine designs and it incurs high performance and energy consumption overheads. For example, the GC module alone consumes an average of 10% of the CPU cycles in Apache Harmony [7, 8, 21]. For some data-intensive applications that frequently trigger the garbage collector, the overhead may be even higher. As seen in Fig. 4, the cloud-oriented EHA need to maximize its computation capacity, so that it includes more GPU units for parallelism and more CPUs for generic logic control. The GC module is one of the specialized cores, interacting with off chip memory for garbage collection.

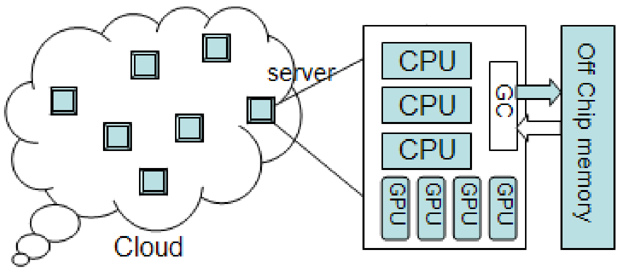

Figure 4: Garbage Collection in EHA

Our profiling results indicate that GC incurs significant overhead costs, contributing to about 10% of the total execution time. By examining the details of GC execution using different GC algorithms, we identified three performance hotspots: *gc_alloc_fast, Vtable manipulation, and scan-slot*. These hotspots contribute to more than 50% of the total GC execution time, and they also consume a significant amount of energy. By moving these hotspot functions into hardware with specialized instructions, we achieved an order of magnitude speedup and significant improvement on energy efficiency. In addition, we also performed an extrapolation study by using the hardware parameters into the profiling data. The results indicate that these three simple hardware-assisted GC instructions can reduce the GC execution time by 59%, leading to a 7% improvement to the overall execution time and similar significant improvements to energy efficiency [3, 5]. It validates that EHA can efficiently adapt to servers in cloud.

6. Energy Efficient Prefetcher in Embedded Mobile Systems

Although our initial motivation for designing the EHA was to accelerate cloud/server-side workloads, we quickly realized that it may be suitable for mobile workloads as well since mobile workloads are highly dynamic and heterogeneous. In this section, we explore how EHA can be used in mobile systems.

Energy efficiency is the most important concern in mobile embedded system design. Conventional wisdom is that there is a tradeoff between energy efficiency and high-performance techniques, such as prefetching. Thus to reduce energy consumption and save chip area, hardware prefetchers have not been implemented in most existing embedded mobile systems. However, since they have become increasingly powerful high-performance techniques, such as hardware prefetching, are increasingly needed if we are to accelerate applications. In Fig. 6, we describe a prefetcher in EHA. Since it is mobile embedded system-oriented, in this context, our EHA should only contain a single GPU and more HC (hardware core)/SC (software core) to maximum performance and minimum energy cost meanwhile.

To this end, we study how the impact of six different hardware prefetchers on three groups of popular mobile applications, including media, game and XML data. We first demonstrate that [4]: contrary to the conventional wisdom, as technology advances (*e.g.,* from 90 nm to 32 nm), prefetching starts to become energy-efficient while improving performance. In the best case, it can offer close to 10% energy reduction. Then, we introduce a general analytical model to identify the conditions under which prefetching techniques yield energy efficiency. This model demonstrates that, for a prefetcher to be energy efficient, the performance gain (*G*) it brings must be greater than the ratio of the energy overhead ($E_{overhead}$) it incurs over the original static energy ($E_{no\text{-}pref\text{-}static}$). Furthermore, we have also introduced a series of models to evaluate the energy efficiency of multi-layer prefetching. By using these models, system designers can easily and accurately evaluate the energy efficiency of their designs and make decisions on the deployment of hardware prefetchers. This modle confirms that EHA can be adapted to embedded mobile scenarios as well.

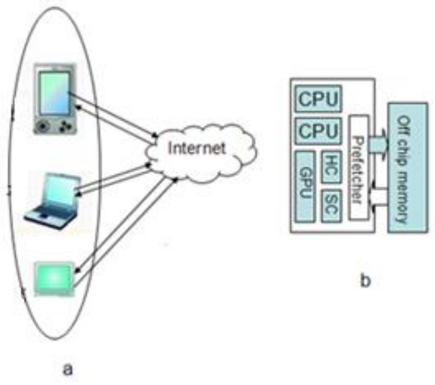

Figure 5: Prefetcher in EHA

7. Network Coding

In this section, we present our study on using EHA to accelerate network coding workloads [33~36]. There are three ways we can accelerate network coding on an EHA many-core chip: specialized core to accelerate the workload, and parallelize the network coding workloads on many general-purposed cores, or the combination of both.

Network coding is a well-known technique used to enhance network throughput and reliability by applying special coding to data packets. One critical problem in practice, when using the random linear network coding technique, is the high computational overhead. More specifically, using this technique in embedded systems with low computational power might cause serious delays due to the complex Galois field operations and matrix handling. To address this problem, we have implemented a hardware accelerator to accelerate the decoding logic for random linear network coding. We have implemented the prototype using field-programmable gate-array (FPGA) technology.

The main design motivation was to improve the decoding delay by dividing and parallelizing the entire decoding process. Fast arithmetic operations are achieved by the proposed parallelized GF ALUs, which allow calculations with all the elements of a single row of a matrix to be performed concurrently. To improve the flexibility in the utilization of the FPGA components, two different decoding methods have been designed and compared [33, 35, 36].

The performance of the proposed idea was evaluated by comparing with the performance of the decoding process executed by general-purpose processors through an equivalent software algorithm. Overall, a maximum throughput of 65.98 Mbps is achieved with the proposed FPGA design on an XC5VLX110T Virtex 5 device. In addition, the proposed design provides speedups of up to 13.84 compared to an aggressively parallelized software decoding algorithm run on a quad-core AMD processor. We expect the speedup would be even higher when we implement the design on ASIC. In the next step, we plan to integrate the design into EHA as one of the specialized cores.

In addition to computation acceleration, we also explored improving networking coding performance using parallel computation techniques [33, 34, 35]. All emerging network applications require deep packet classification as well as security-related processing and they should be run at line rates. Hence, network speed and the complexity of network applications will continue increasing and future network processors should simultaneously meet two requirements: high performance and high programmability. We found that the performance of single processors would not be sufficient to support future demands. Therefore, we turned to multi-core processors which can exploit the parallelism in network workloads. In our study [34], we focused on the cache coherence protocols which are central to the design of multi-core based network processors. We investigated the effects of two main categories of various cache coherence protocols with several network workloads on multi-core processors. Our results show that token protocols have a significantly higher performance than directory protocols. With an 8-core configuration, token protocols improve the performance compared to directory protocols by a factor of nearly 4 on average. In the next step, we plan to compare the performance, power consumption, and chip area cost of the parallel computation technique and the hardware acceleration technique.

8. Cryptographic Computation

In this section, we present our study on using EHA to accelerate cryptographic computations [28~30]. We compared two ways to accelerate the cryptographic computations on an EHA many-core chip: specialized core to accelerate the workload, and parallelize the cryptographic computation on many general-purposed cores. In addition, we studied the energy efficiency of proposed designs.

Data encryption/decryption has become an essential component for modern information exchange. However, executing these cryptographic algorithms is often associated with huge overhead and the need to reduce this overhead arises correspondingly. In our study, we selected nine widely adopted cryptography algorithms and investigated their workload characteristics [29]. Different from many previous works, we considered the overhead not only from the perspective of computation but also focusing on the memory access pattern. We broke down the function execution time to identify the software bottleneck suitable for hardware acceleration. Then we categorized the operations needed by these algorithms. In particular, we introduced a concept called "Load-Store Block" (LSB) and performed LSB identification of various algorithms. Our results illustrate that for cryptographic algorithms, the execution rate of most hotspot functions is more than 60%; memory access instruction ratio is mostly more than 60%; and LSB instructions account for more than 30% for selected benchmarks.

Parallelizing the computation of cryptographic algorithms on many-core computing platforms can be a promising approach to reduce the execution time and eventually the energy consumption of such algorithms. In this study, we built a pipelined model to analyze and compare the execution time and energy consumption of the Blowfish cryptographic algorithm on the Single-Chip Cloud Computer (SCC) [28]. In this model the Blowfish cryptographic algorithm is divided to smaller chunks and each chunk is run only by one core. Using message passing interface, the input data passes in turn through all the cores involved. Due to the communication overhead and latency associated with this model, we experimented and identified the optimal message size to pass between the cores to avoid saturating the on-chip communication network. Our results illustrate that our parallel approach is 27X faster than the sequential approach and yields close to 16X less energy consumption on the SCC platform.

In another study, we focus on energy efficiency of executing the cryptographic workloads [30]. On one hand, we want to reduce the computation overhead of cryptography algorithms; on the other hand, we also want to reduce the energy consumption associated with this computation overhead. We explored and present implementation techniques for energy-efficient hardware acceleration of RSA cryptography and Blowfish cryptography. Instead of implementing the entire algorithm into hardware format, we provided a system design that focus on accelerating the execution of the critical path of each of the cryptography algorithm, which is the most computation-intensive component. We carefully implemented the critical path as a customized coprocessor to improve the overall system throughput on Virtex-5 Field-Programmable Gate Array (FGPA) platform. Subsequently, we made a comparison of the effectiveness and energy consumption between the pure software implementation of the cryptography algorithms and our proposed approach. The results show that our critical path enhancement design speeds up the execution of RSA by 11% and Blowfish by 58.8%; in the meantime, we are able to reduce the energy consumption by 9.6% for RSA and 36.0% for Blowfish, thus achieving our objective.

9. Other Applications

We have also done studies on the following areas of the EHA design

- Energy efficiency [24, 26, 27]
- Acceleration of Software Defined Radio workloads [37]
- Software communication architecture for hardware acceleration [32]
- Interaction between different cores on EHA, such as GPU and CPU [38]
- Acceleration of Electroencephalogram processing [31]

10. Conclusions

In this paper, we introduced our EHA project, a result of collaboration among several research groups globally, both in academia and industry. EHA seeks to achieve power and performance efficiency, scalability and adaptivity. To reach this goal, it incorporates general-purpose cores and specialized cores by using the concept of factored service and effective using of kinds of units. Its general-purpose core is used to take care of generic control and computation; its specific-core always includes at least one GPU for intensive computation, hard accelerators (ASIC accelerators) and soft accelerators (FPGAs) for accelerating frequently used or computationally intensive applications. The combination of generic core and specialized cores can greatly improve performance while maintaining a low energy consumption.

This project has been divided into three stages: First, we ported some frequently executed OS/middleware services or high overhead services onto a homogeneous many-core design, and studied its performance and power consumption. Second, since not all services are equally weighted or requested, in order to get maximal performance gain and energy efficiency, we tailored specialized design for different services. As the final step, at which the project is currently, we plan to construct a prototype Extremely Heterogeneous Architecture (EHA) chip by integrating all these pieces. This EHA prototype will consist of multiple homogeneous light-weight cores, multiple specialized cores; each of these cores will host a service. It is meant to have the best balance in performance, energy consumption and programmability. Also, in the future, we will incorporate more aggressive optimization techniques into the EHA, such as value prediction [20, 22], speculative execution [39, 40], and run-time partial reconfiguration [23, 24].

Acknowledgments

This work is partly supported by the National Science Foundation under Grants No. CCF-1065448, ECCS-1125762, and by the National Natural Science Foundation of China under grant number (61173045). This work is also sponsored, in part, by Universidad Veracruzana through Facultad de Ingeniería en Electrónica y Comunicaciones. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. Jie Tang, Shaoshan Liu, Zhimin Gu, Chen Liu, and Jean-Luc Gaudiot. Acceleration of XML Parsing Through Prefetching. *IEEE Transactions on Computers, In Press*
2. Jie Tang, Pollawat Thanarungroj, Chen Liu, Shaoshan Liu, Zhimin Gu and Jean-Luc Gaudiot. Pinned OS/Services: A Case Study of XML Parsing on Intel SCC. Journal of Computer Science and Technology, *In Press*
3. Jie Tang, Shaoshan Liu, Zhimin Gu, Xiao-Feng Li, and Jean-Luc Gaudiot. Achieving middleware execution efficiency: Hardware-assisted Garbage Collection Operations. Journal of Supercomputing: Volume 59, Issue 3 (2012), Page 1101-1119, Springer
4. Jie Tang, Shaoshan Liu, Zhimin Gu, Chen Liu, and Jean-Luc Gaudiot. Prefetching in Mobile Embedded System Can be Energy Efficient. IEEE Computer Architecture Letters, Volume 10, Issue 1 (2011), Pages 8-11
5. Jie Tang, Shaoshan Liu, Zhimin Gu, Xiao-Feng Li, and Jean-Luc Gaudiot. Hardware-Assisted Middleware: Acceleration of Garbage Collection Operations. In Proceedings of the 21st IEEE International Conference on Application-Specific Systems Architectures and Processors (ASAP 2010), Rennes, France, 2010: Pages 281-284
6. Jie Tang, Shaoshan Liu, Zhimin Gu, Chen Liu, and Jean-Luc Gaudiot. Memory-Side Acceleration for XML Parsing. In Proceedings of the 8th IFIP International Conference on Network and Parallel Computing (NPC 2011). Changsha, China, 2009: 277-292
7. Shaoshan Liu, Jie Tang, Ligang Wang, Xiao-Feng Li, and Jean-Luc Gaudiot. Packer: Parallel Garbage Collection Based on Virtual Spaces. IEEE Transactions on Computers, DOI: 10.1109/TC.2011.193, October, 2011
8. Shaoshan Liu, Jie Tang, Chengrui Deng, Xiao-Feng Li, and Jean-Luc Gaudiot. RHE: A JVM Courseware. IEEE Transactions on Education. Volume 54, Issue 1 (2011), Page 141-148
9. Jie Tang. Research on Performance Acceleration and Energy Efficiency for EHA Multicore Processor, Ph.D. Thesis, 2012, Beijing Institute of Technology, China
10. Intel Labs, "SCC Platform Overview," Intel Many-core Applications Research Community, Revision 0.75, September 2010.
11. K. Chiu, M. Govindaraju, and R. Bramley. Investigating the limits of soap performance for scientific computing. In Proceedings of the 11 th IEEE International Symposium on High Performance Distributed Computing HPDC-11 2002
12. M. R. Head, M. Govindaraju, R. van Engelen, andW. Zhang. Grid scheduling and protocols—benchmarking xml processors for applications in grid web services. In SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing, page 121, New York, NY, USA, 2006. ACM Press.
13. P. Apparao, R. Iyer, R. Morin, and et al., "Architectural characterization of an XML-centric commercial server workload," in 33rd International Conference on Parallel Processing, 2004
14. International HapMap Project: http://hapmap.ncbi.nlm.nih.gov/
15. Zhefu Shi, Beard Cory, Mitchell Ken. Analytical Models for Understanding Misbehavior and MAC Friendliness in CSMA Networks. Performance Evaluation (2009), Volume: 66, Issue: 9-10, Pages: 469-487
16. Zhefu Shi, Beard Cory, and Mitchell Ken. Misbehavior and MAC Friendliness in CSMA Networks . IEEE Wireless Communications and Networking Conference, 2007. (WCNC 2007). March 2007. page(s): 355 – 360

17. Zhefu Shi, Beard Cory, and Mitchell Ken. Tunable Traffic Control for Multihop CSMA Networks. IEEE Military Communications Conference, 2008. MILCOM 2008. Nov. 2008. On page(s): 1 – 7
18. Zhefu Shi, Beard Cory, and Mitchell Ken. Competition, Cooperation, and Optimization in Multi-Hop CSMA Networks. Proceeding: PE-WASUN '11 Proceedings of the 8th ACM Symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks. L. Cherkasova and R. Gardner. Measuring CPU overhead for I/O processing in the Xen virtual machine monitor[C]. In proceedings of the annual conference on USENIX Annual Technical Conference, 2005:24-24
19. M. Ferrer. Measuring overhead introduced by VMWare workstation hosted virtual machine monitor network subsystem, http://studies.ac.upc.edu/doctorat/ENGRAP/Miquel.pdf
20. Shaoshan Liu, and Jean-Luc Gaudiot. Potential Impact of Value Prediction on Communication in Many-Core Architectures, IEEE Transactions on Computers, June, 2009
21. Shaoshan Liu, Ligang Wang, Xiao-Feng Li, and Jean-Luc Gaudiot. Space-and-Time Efficient Parallel Garbage Collector for Data-Intensive Applications, International Journal of Parallel Programming, October, 2010
22. Shaoshan Liu, Christine Eisenbeis, and Jean-Luc Gaudiot, Value Prediction and Speculative Execution on GPU, International Journal of Parallel Programming, December, 2010
23. Shaoshan Liu, Richard Neil Pittman, Alessandro Forin, and Jean-Luc Gaudiot, Minimizing the Run-time Partial Reconfiguration Overheads in Reconfigurable Systems, Journal of Supercomputing, *In Press*
24. Shaoshan Liu, Richard Neil Pittman, Alessandro Forin, and Jean-Luc Gaudiot, Achieving Energy Efficiency through Run-Time Partial Reconfiguration on Reconfigurable Systems, ACM Transactions on Embedded Computing Systems, *In Press*
25. Shaoshan Liu, Won W. Ro, Chen Liu, Alfredo C. Salas, Christophe Cérin and Jean-Luc Gaudiot, EHA: The Extremely Heterogeneous Architecture, In Proceedings of the 2012 International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN 2012), San Marcos, Texas, December 13-15, 2012
26. Kenneth Berry, Felipe Navarro, and Chen Liu, Application-Level Voltage and Frequency Tuning of Multi-Phase Program on the SCC, In Proceedings of the 3rd International Workshop on Adaptive Self-tuning Computing Systems (ADAPT'13), co-located with HiPEAC 2013, January 22nd, 2013, Berlin, Germany
27. Gustavo Chaparro-Baquero, Qi Zhou, Chen Liu, Jie Tang and Shaoshan Liu, Power-Efficient Schemes Via Workload Characterization on the Intel's Single-chip Cloud Computer, In Proceedings of The Eighth Workshop on High-Performance, Power-Aware Computing (HPPAC 2012), in conjunction with IPDPS 2012, Shanghai, China, May 21, 2012
28. Kamak Ebadi, Victor Pena, and Chen Liu, High-Performance Implementation and Evaluation of Blowfish Cryptographic Algorithm on Single-Chip Cloud Computer: A Pipelined Approach, In Proceedings of The 2nd International Conference on Applied and Theoretical Information Systems Research (ATISR 2012), December 27-29, Taipei, Taiwan
29. Jed Kao-Tung Chang, Chen Liu, Shaoshan Liu and Jean-Luc Gaudiot, Workload Characterization of Cryptography Algorithms for Hardware Acceleration, In Proceedings of the 2nd ACM/SPEC International Conference on Performance Engineering (ICPE 2011), Karlsruhe, Germany, March 14-16, 2011
30. Chen Liu, Roland Duarte, Omar Granados, Jie Tang, Shaoshan Liu and Jean Andrian, Critical Path Based Hardware Acceleration for Cryptosystems, International Journal of Advancements in Computing Technology (IJACT), Vol. 4, No. 1, pp.438-452, 2012.
31. Gildo Torres, Paul McCall, Chen Liu, Mercedes Cabrerizo and Malek Adjouadi, Parallelizing Electroencephalogram processing on a many-core platform for the detection of High Frequency Oscillations, In Proceedings of the 7th International Workshop on Unique Chips and Systems (UCAS-7), in conjunction with HPCA-18, New Orleans, Louisiana, USA, February 26, 2012.
32. Rolando Duarte, Omar Granados, Chen Liu and Jean Andrian, Case Study on a Software Communications Architecture Component for Hardware Acceleration, In Proceedings of the 4th International Conference on Ubi-media Computing (U-Media 2011) , Sao Paulo, Brazil, July 3-4, 2011
33. Minwoo Kim, Karam Park, and Won Woo Ro, Benefits of using parallelized non-progressive network coding. Journal Network and Computer Applications 36(1): 293-305 (2013)

34. Kyueun Yi, Won Woo Ro, and Jean-Luc Gaudiot: Importance of Coherence Protocols with Network Applications on Multicore Processors. IEEE Transactions on Computers 62(1): 6-15 (2013)
35. Karam Park, Joon-Sang Park, Won Woo Ro: On Improving Parallelized Network Coding with Dynamic Partitioning. IEEE Transactions on Parallel Distributed Systems 21(11): 1547-1560 (2010)
36. Sunwoo Kim, Wong S. Jeong, Won W. Ro and Jean-Luc Gaudiot, Design and Evaluation of Random Linear Network Coding Accelerators on FPGAs, ACM Transactions on Embedded Computing Systems, *in press*
37. Chen Liu, Omar Granados, Rolando Duarte and Jean Andrian,Energy Efficient Architecture Using Hardware Acceleration for Software Defined Radio Components,Journal of Information Processing Systems (JIPS), Vol. 8, No. 1, pp.133-144, 2012.
38. Changmin Lee, Won Woo Ro, and Jean-Luc Gaudiot: Cooperative heterogeneous computing for parallel processing on CPU/GPU hybrids. In Proceedings of The 16th Workshop on Interaction between Compilers and Computer Architectures 2012
39. Won W. Ro and Jean-Luc Gaudiot, SPEAR: A Hybrid Model for Speculative Pre-Execution, In Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS 2004), Santa Fe, New Mexico, April 26-30, 2004
40. Won W. Ro, Stephen P. Crago, Alvin M. Despain, and Jean-Luc Gaudiot, Design and Evaluation of a Hierarchical Decoupled Architecture, The Journal of Supercomputing, Vol. 38, No. 3, pp.237-259, December 2006