VOL. 9, NO. 12, DECEMBER 2014 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences


www.arpnjournals.com

# NEW EMBEDDED COMPUTING ARCHITECTURE USING HETEROGENEOUS PROCESSORS FOR FAST PROCESSING AND LOWER COMPLEXITY

Muataz Hameed Salih, R. Badlishah Ahmed, L.A. Hassnawi, R. Kh. Al-Janabi and Omar. F. Yousif
School of Computer and Communication Engineering Universiti Malaysia Perlis (UniMAP) Perlis, Malaysia
E-Mail: muataz@unimap.edu.my

**ABSTRACT**

When it comes to communicating between processors in a Multi-Computing system, there are many choices. Exotic architectures are available with tens to hundreds of cores. This allows plentiful processing power to be utilized on problems. However, how to retain the cores feed with data? Core is a wasted resource when idle. Both of bandwidth between resources and latency incurred in transfers are performance concern. That can significantly affect the bandwidth for small transfers and can also make the system-transfer function exceed its design requirements for real-time operation. The solution to this problem lies in multi-level of computing architecture. A new architecture at SoC and chip level is designed and implemented to come up with high-performance embedded multi-computing architecture for AUV. It will be based on FPGA technology that will be as computing unit and router for other processing units as well. The designed architecture provides a concurrent environment for programming and testing. Furthermore, numerous FPGA mega core modules are provided to easily verify the targeted problem, address processing issues and balance data movement with processing power. The outputs and deliverables from the designed architecture are manifold in terms of performance and throughput accuracy; covering a sensing area of over 95%. The designed architecture seeks to provide dynamic reconfigurable platform with the knowledge and tools needed to improve in today's academic, research, industry environment and realistic applications.

**Keywords:** embedded system design, FPGA system design, simultaneous multithreading, autonomous underwater vehicle.

## INTRODUCTION

In order to ensure speedy and accurate processing throughput, the underwater vehicle system needs a precise and efficient computational monitoring platform. Moreover, precision is also important to lower the operational complexity and facilitate the decision making process. To make the system easy-to-handle and manage, the computational platform should be compact and scalable.

The current platforms for underwater vehicle system used commercially by developers consist of Digital Signal Processing (DSP) chip, microcontroller, ATX-PC board and embedded single soft processor on an FPGA. Though the platform operates in a systematic way, it consists of a number of phases and there are two major problems that challenge the effectiveness of these platforms. The entire procedure of fetching data, processing it, filtering information and produce results make the process of decision making ambiguous and unnecessarily complex. Moreover, when there are so many factors working together, there are unnecessary processing delays due to synchronization, information transfer. Hardware issues like computational power and architectural units' synchronization can also cause significant delay in data processing which impacts the accuracy of decision making process. This is the reason that researchers and developers are coming up with alternate solutions and more robust architectural design for underwater vehicle system.

One way to remove the inability issue and facilitate fast data processing is to determine the post processing scenarios. Developers and researchers, think data delays are caused due to issues during the data processing process, therefore they are considering on the post-processing data analysis first in order to resolve the inability issue. However, the post-processing procedures of data processing and operation are mostly accurate and robust therefore the idea of post-processing analysis is not applicable to online mission. So, to completely avoid the post-processing approach a robust, effective computational platform is required for the submarine vehicles system.

### Multiprocessor

### Embedded computing

Although it is not easy to figure the differences between general purpose computing and embedded computing, general purpose computers are not as scalable as embedded computers. Embedded computers have evolved from large-sized working stations to on-chip systems. Moreover, embedded computers have also improved considerably in terms of sophistication, data processing and complexity. With the advent of semiconductors in the IT world, embedded computers have become more reliable and robust than ever and their prices have also dropped considerably. Below is a detailed discussion about the major differences between general purpose and embedded computer and their applications [1, 2].

When it comes to power requirements and energy considerations, embedded systems are more demanding than general purpose computers. These computers need

heat-sink and thermal packages to keep the processing units cool which adds to the overall costs of these systems. Especially designed heat sinks are used in these systems because normal heat sinks cannot be installed on small chips. The problem of energy efficiency is particularly evident in handsets, as they rely on batteries to function and it is difficult to keep the temperature of batteries low [3, 4].

The modern in application in which embedded systems are used have more dynamic computational requirements. It is believed that computational needs escalated with the boom of scalable algorithms, data and information compression, novel and dynamic communication standards. When talking about the computation requirements, one cannot ignore the real time performance requirements. A number of high-tech love systems that rely on embedded systems for real time data monitoring and processing, have complex processing requirements. For example, a real time image or video processing systems, requires the rate of frames per second to be adjusted in accordance to the output requirements if the application. The systems in which there is not a lot of streaming for decoding the information have even more complex processing needs. Same is the case with control and feedback systems used in telecommunication and automobiles industries; they have very high and customized demands [5-7].

Applications that rely on embedded systems need a wide assortment of computational needs and processing requirements that have to be fulfilled. Components working in a closed loop cellular arrangement are expensive than other components which are used in general purpose computers because they are designed for long-term use. This is the reason that developers demand that these components should be upgradable and portable so they can grow with the system in which they are deployed and fulfill the real-time requirements of these systems. Network operations and designers consider it wise to invest in making the embedded systems scalable and upgradable because it can secure the initial investment. However, commercial components, designed particularly for providing client services are affordable but they have a shorter lifespan than dedicated embedded components [6, 8].

**Hetrogenouse processors**

When a variety of computational modules are integrated together, they form a heterogeneous computing system. However, the functionality of a heterogeneous system depends on the types of computational units. It can serve the purpose of a GPP (General Purpose Processor), FPFA (Field Programmable Gate Array), DSP (Digital Signal Processor), ASIC (Application Specific Integrated Circuit) and GPU (Graphics Processing Unit). So it can be said that a heterogeneous computing system consist of processors with different functionality and instructional set architectures.

Today systems need to be highly reactive and multi-functional and this is the reason that heterogeneous systems have replaced simple processing units. Heterogeneity allows computational systems to interact with network and video/audio applications at the same time, thus enhancing their efficiency and reactivity. Before the advent of heterogeneity, techniques like frequency scaling were used to increase the efficiency of systems. However, these techniques were not dramatically effective and created obstructions like power-wall and memory-wall [1, 2]. This is the reason hat technologists had to incorporate hardware amendments to increase the productivity of systems and making them heterogeneous [3, 4]. Heterogeneity allows a system designer to use different kinds of processors and elements and integrate them together to develop an all-inclusive processing system [5]. Changes in the hardware or integration of two or more different types of processors make a system multi-core or parallel computing system. Technologists also consider such types of systems as 'hybrid systems' [6].

As integrated chips have replaced bulky elements, so it has become easier for fabricators to scale and integrate elements. However, there are a number of challenges in heterogeneous systems due to parallel processing, which are not present in homogenous systems. Unlike heterogeneous systems, the parallel-processing components in homogenous systems have uniformly structured instruction set architecture, which is easier to code, decode and troubleshoot [7]. Due to non-uniformity, the coding in heterogeneous computational systems needs multiple assemblers, simulators and compilers. This makes the development and integration phases considerably complicated. Furthermore, it is required to adopt a cohesive approach to simultaneously develop various modules of a heterogeneous system [8-11].

Moreover, it is difficult for programmers to code every element differently. It not only increases work load for programmers, but also slows down the process of development. Integrating and interleave hardware specific programming codes, increases the probability of loopholes and makes the system extremely complex [12-18]. There is no denying the fact that heterogeneous systems are far efficient than homogenous computational units, but it is difficult to handle these systems [19, 20].

**Architectural Design**

In this paper, in this research, a design is proposed that uses an embedded system which consists of dual-issue processing unit as the major component which is integrated with all the other elements installed on the board. The REP chosen for this design comprises of two main phases. The first stage is called dual-issue task fetch and the second one focuses at load and store operations. Moreover the pan also includes data memory and timer synchronization tasks. The main unit incorporated into the system is responsible for fetching, processing, filleting and decoding the information and then issuing the tasks to the execution pipelines. Figure-1 explains in detail the local organization of components on the REP. The design also has another unit called the branch unit that is meant to host a dynamic branch. The accuracy of the branch has been

improved by this approach but it has reduced the latency of the system.

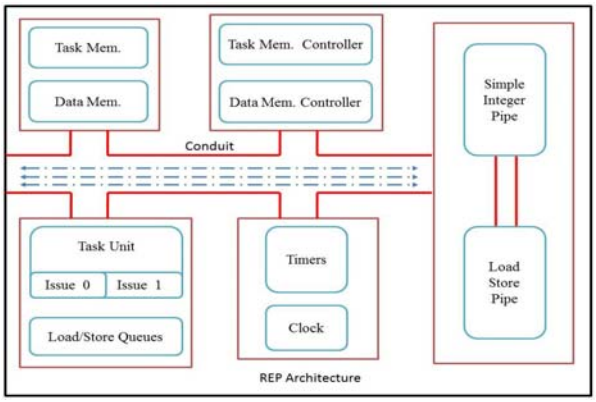


**Figure-1.** Block diagram of designed processor (REP).

The REP is capable of differentiating the control data from operational data, thus allowing parallel access to each kind of data and allowing to reduce the number of execution pipelines. Microcontrollers have 32-bit pipelines which are further set associative in 64 ways. These lines are compatible with parity tags and this makes them perfect for protecting memory arrays from program execution and soft errors. The implementation of the REP platform also allows dedicated and targeted troubleshooting which allows reading the data directly from the source and sending it to the designated arrays. The connection between memory components and microcontroller allows task-side interface functioning of the processor.

Memory can translate the address, and in embedded systems, memory bus has direct access to the control block and stored information. A virtual memory is also present in the system which is supported by all applications that rely on embedded systems. The virtual memory approach is supported by all those applications which require real time and scalable logical mapping of addresses. The translation abilities, however, are controlled by a buffer look-aside table.

There are two timers and primary time-management bus present in the REP. The timers consist of a FIT (Fixed Time Interval) and a DEC chip which is meant to decrement the timer. Moreover, there are three separate and synchronized PLB interfaces present in the REP. Two PLBs support both reading and writing operations and the remaining supervises read dedicated tasks. The layout of PLB interfaces and timers on the FPGA board is explained in Figure-2.

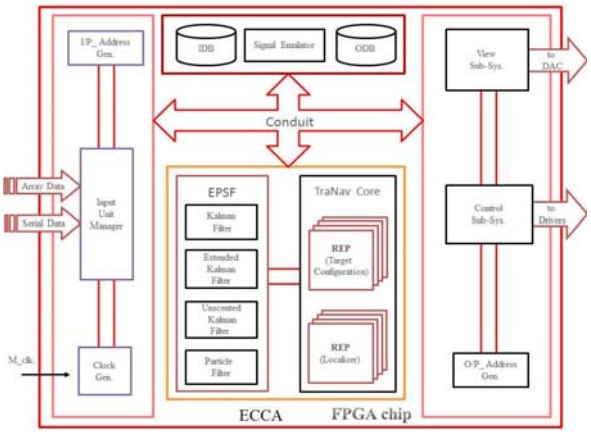


**Figure-2.** The proposed architecture.

The main design, layout and execution of the core processor, is based on three interlinked and embedded modules. The intermediate module is designed in the ECC architecture and the overall design of each module is dependent in the design of the REP.

Another major element integrated in the design is floating bus. The previous layout of the REP had fixed allocated bus. On the other hand, the design presented in this research has an entire and dedicated system of buses. It means that buses can actually float through the entire system and all the sub-modules of the system are operated by the network of the buses. The I/O manager has a separate design for input data which consist of N component matrix arranged in 2D. The sensors used for inputs are arranged in XY plane. The I/O member locations of the database have an internal memory chip which prevents the interconnection between internal and external memory. This is one to enhance the speed of the system.

In order to improve the overall performance of the systems, a number of subsystems are incorporated in the design. The self-assessment sub-module uses an emulator that process and identifies targets in the overall behavior of the system. Another subsystem is view system which aims at converting the processed ad filtered data into SVGA (Super Video Graphics Array) signal which are transferred to the chip for useful interpretation of information.

## RESULT AND SYSTEM EVALUATION

It is challenging to make sure that the localization of information and navigation are performed precisely underwater, due to obvious hurdles. Therefore it is elementary to determine the system (AUV) has to travel to achieve accurate readings and repeatable measurements that can be applied to other applications. A number of approaches have been used previously to design a vehicle system, but most of these systems followed the principles of acoustic or vision. This study, however, proposed two different approaches to design a REP system for underwater localization. The approach is based on

embedding sound sensors on the AUV which are designed to monitor views and directions. There are almost 32 sensors used in this design and the layout is explained in Figure-3.

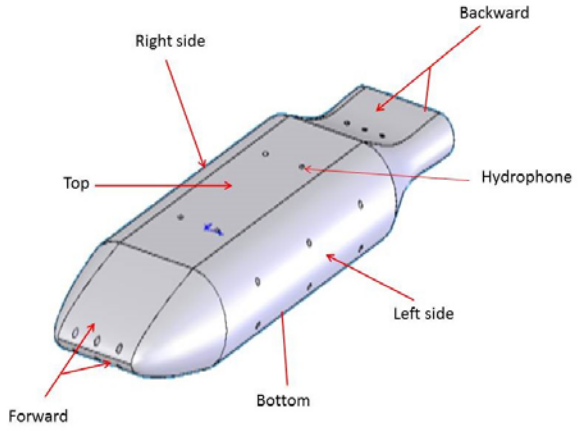


**Figure-3.** Sensor arrangements on the AUV.

The 4x8 2D matrixes are used in order to determine the approach and design in which the network of 32 sensors is organized. A fast ADC chip is used in order to identify the location of the 8bit incoming signal from a sensor. Another 2D matrix of the same measurement s arranged to target and identifies the location of the array sensors according to their bit capacity. For example 3 best sensors are allocated to monitor the side direction while remaining is specified to represent some of their direction.

Sound sensors generate high speed signals which are transmitted to the 8bit ADC chip. The ADC chip used in the REP s designed to convert the incoming analog signals from the sensors to the digital packets at a rate of 3 GSPS. These signals then enter the process using floating localization technique. Getting signals from all directions allow the processor to get a clear idea about evaluating direction, distance and speed simultaneously. The REP system is capable of pressing and responding in real time due to its overall architecture installed in the FPGA board. Another reason why the system responds in real time is the crystal frequency. The crystal used in the REP can oscillate faster than the speed of sound signals in water transmitted by sensors. The data processing speed and frequency of the system makes it appropriate for other applications too, with a few amendments.

The designs proposed in the study have been implemented for tracking the AUV signals and for passive location. In this research study both designs have been analyzed and their complexity in a number of applications has been evaluated. The approach of analysis and synthesis has been adopted in this study to evaluate the proposed designs. Data acquisition and a number of computational operations have been implemented to test the effectiveness of sensors used in the REP for underwater localization. The results prove that it is very important to overcome the transmission errors and process delays to make the tracking procedure as reliable as possible. However, there are a number of challenges regarding location and tracking which were noticed during the implementation of the system. For example, one of the problems is taking measurements in a parallel manner. The current architecture of the system is designed in view of the issues that are face during critical operations. A number of measurements are shown for the motion of AUV in Figures 4, 5, 6, 7, 8, 9, 10 and 11.

In order to acquire reliable and precise results, it was very important to decide the number of iterations to get repeated and acceptable results. The precision of results is directly proportional to the number of iterations. However, there is a saturated number of iterations after which repeating the experiment can waste time. In this case, 20 iterations were required to acquire accurate analysis results. Extended Kalman filter is used on this study which is better and more precise than Kalman filter. A comparison between both filters is shown in Figures 4, 5, 6, 7, 8, 9, 10, and 11.

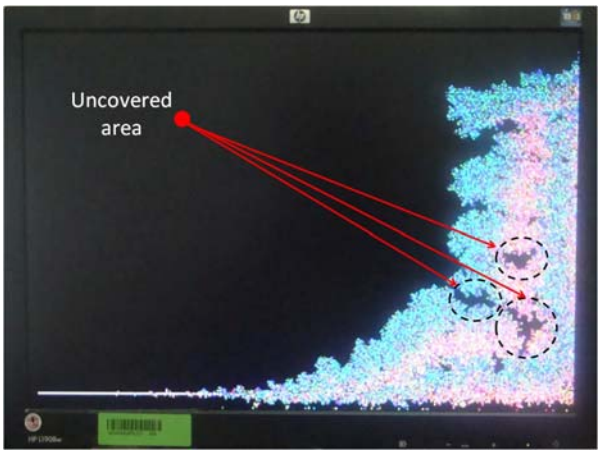


**Figure-4.** Processed sensor signal coming from the KF after 10 iterations.

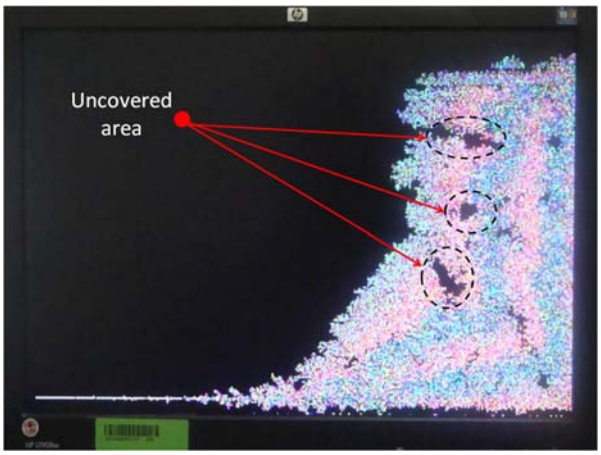


**Figure-5.** Processed sensor signal coming from the KF after 19 iterations.

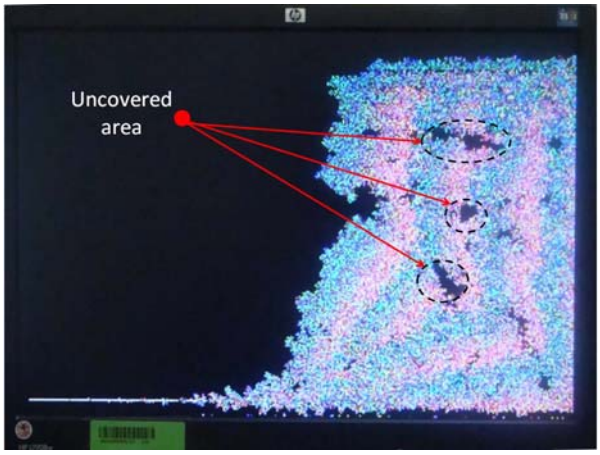


**Figure-6.** Processed sensor signal coming from the KF after 20 iterations.

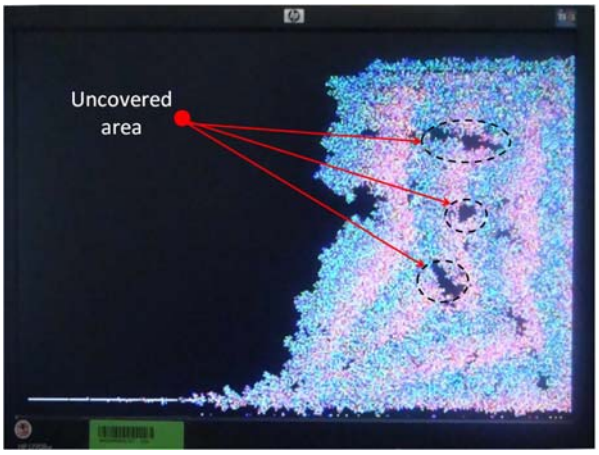


**Figure-7.** Processed sensor signal coming from the KF after 25 iterations.

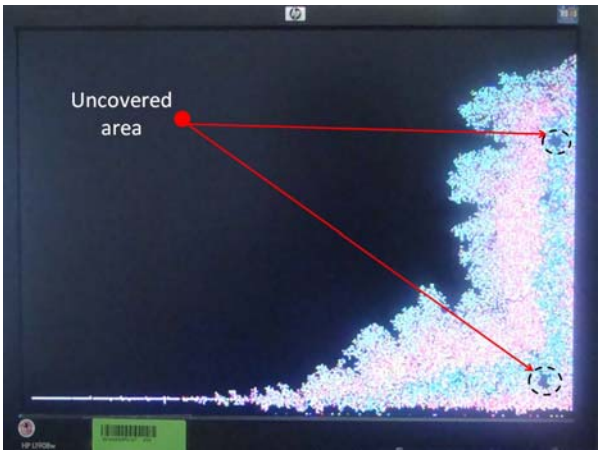


**Figure-8.** Processed sensor signal coming from the EKF after 10 iterations.

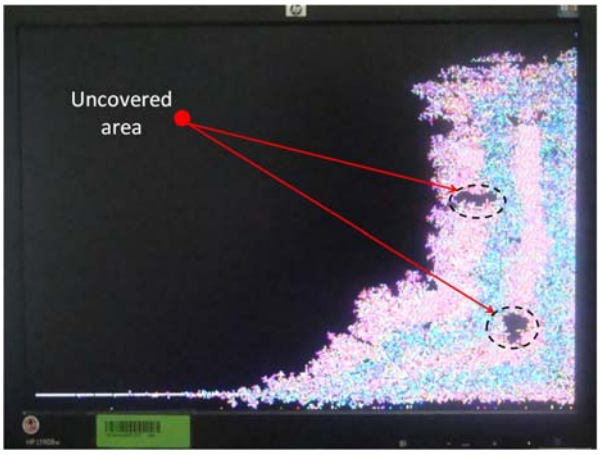


**Figure-9.** Processed sensor signal coming from the EKF after 19 iterations.

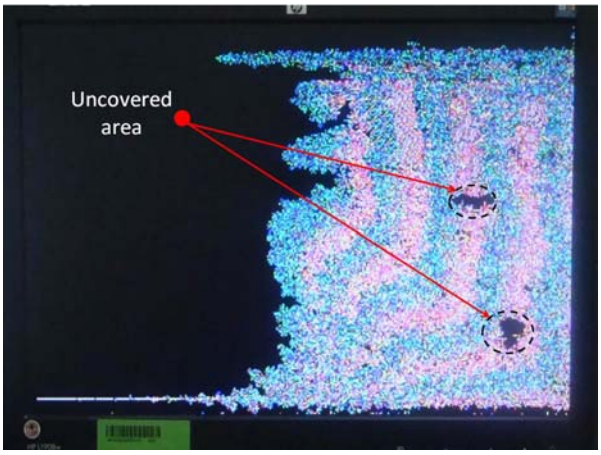


**Figure-10.** Processed sensor signal coming from the EKF after 20 iterations.

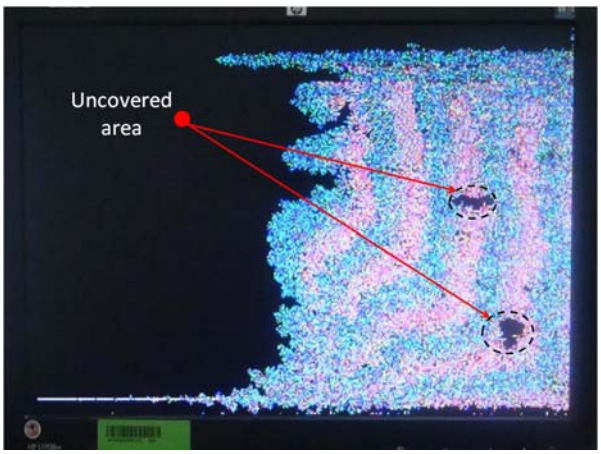


**Figure-11.** Processed sensor signal coming from the EKF after 25 iterations.


## CONCLUSIONS

The study analyzed the architecture and layout of the system on the basis of three parameters. The first important parameter is modularity of the system. The architecture is installed on a FPGA. This architecture allows dual functionality of the system. For large applications like base stations, the system can work as a single module. While for smaller applications, the system can be divided into modules due to point to point connection between subsystems. The second parameter is flexibility. The architecture consists of a number of systolic stages which can be adjusted according to the requirements of the system. The third and last parameter is scalability. The system can be replicated and upgraded according to the computational requirements of the application in which it is being used.

## REFERENCES

[1] IBM. 2009. Cell Broadband Engine Programming Tutorial. Retrieved. 05-06.

[2] John Shalf. 2010. The New Landscape of Parallel Computer Architecture. Retrieved. 02-25.

[3] Tim Kaldewey, Guy Lohman, Rene Mueller and Peter Volk. 2012. GPU join processing revisited. In Proceedings of the DaMoN ’12, pp. 55-62, New York, NY, USA.

[4] Brodtkorb, André Rigland; Christopher Dyken, Trond R. Hagen, Jon M. Hjelmervik, Olaf O. Storaasli. May 2010. State-of-the-Art in Heterogeneous Computing. Scientific Programming. 18: 1-33.

[5] K. Van Craeynest, A. Jalelle, L. Eeckhout, P. Narvaez and J. Emer. 2012. Scheduling heterogeneous multi-cores through performance impact estimation (PIE). In ISCA ’12.

[6] 2009. Visions for Application Development on Hybrid Computing Systems. Retrieved. 02-09.

[7] Brian Flachs, Godfried Goldrian, Peter Hofstee, Jorg-Stephan Vogt. 2009. Bringing Heterogeneous Multiprocessors into the Mainstream. Symposium on Application Accelerators in High-Performance Computing (SAAHPC'09).

[8] Cray Computers. 2013. Cray XD1 Datasheet. Retrieved.

[9] Ron Wilson EDN. 2010. Xilinx FPGA introductions hint at new realities. February 2, 2009 Retrieved.

[10] Mike Demler EDN. 2011. Xilinx integrates dual ARM Cortex-A9 MPCore with 28-nm, low-power programmable logic. March 1, 2011. Retrieved.

[11] 2013. What is Heterogeneous System Architecture (HSA)? AMD. March 31, 2013. Retrieved.

[12] Kunzman D. M., Kale L. V. 2011. Programming Heterogeneous Systems. 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum. p. 2061. doi:10.1109/IPDPS.2011.377. ISBN 978-1-61284-425-1.

[13] Siegfried Benkner, Sabri Pllana, Jesper Larsson Träff, Philippas Tsigas, Andrew Richards, Raymond Namyst, Beverly Bachmayer, Christoph Kessler, David Moloney, Peter Sanders. 2012. The PEPPHER Approach to Programmability and Performance Portability for Heterogeneous many-core Architectures. Advances in Parallel Computing, IOS Press 22: 361-368, doi: 10.3233/978-1-61499-041-3-361.

[14] D. Koufaty, D. Reddy, and S. Hahn. 2010. Bias scheduling in heterogeneous multi-core architectures. In EuroSys.

[15] Y. He, S. Elnikety and H. Sun. 2011. Tians scheduling: Using partial processing in best-effort applications. In ICDCS.

[16] D. Meisner, C. M. Sadler, L. A. Barroso, W.-D.Weber and T. F. Wenisch. 2011. Power management of online dataintensive services. In ACM/IEEE International Symposium on Computer Architecture, ISCA ’11, pp. 319-330.

[17] J. C. Saez, D. Shelepov, A. Fedorova and M. Prieto. 2011. Leveraging workload diversity through OS scheduling to maximize performance on single-ISA heterogeneous multicore systems. Journal of Parallel and Distributed Computing. 71(1): 114-131.

[18] Scott Grauer-Gray, Lifan Xu. 2012. Robert Ayalasomayajula and John Cavazos. Auto-tuning a high-level language targeted to GPU codes. In Innovative Parallel Computing Conference. IEEE.

[19] Phitchaya Mangpo Phothilimthana Jason Ansel Jonathan Ragan-Kelley Saman Amarasinghe. Portable Performance on Heterogeneous Architectures. In ASPLOS’13, March 16-20, 2013, Houston, Texas, USA.

[20] Andrew Davidson, Yao Zhang, and John D. Owens. 2011. An auto-tuned method for solving large tridiagonal systems on the GPU. In Parallel and Distributed Processing Symposium. IEEE.