Design and Implementation of Reconfigurable Embedded Processor (REP) for AUV using FPGA

M. H. Al-Doori[1], R. Badlishah Ahmad[1], Abid Yahya[1] and Mohd. Rizal Arshad[2]

[1] School of Computer and Communication Engineering, Universiti Malaysia Perlis, Kuala Perlis, 02000, Malaysia

[2] School of Electric and Electronic Engineering, Universiti Sains Malaysia, Penang, 14300, Malaysia

Abstract

The superscalar pipeline is something that the REP architecture resembles. The features that are similar in both architectures include rename, decode, fetch as well as in-order front-end's parts as substituted by dispatch units. The physical register file, execute unit, dynamic scheduler and task queue are contained by the out of-order execution core. The architecture also contains the REP that beholds an integrated system of shared-memory. A number of experiments with realistic benchmarks and an FPGA environment are used to synthesize design. The experiments conducted yielded in results that prove that the new architecture of the REP leads to higher performance of applications because of decreased processing time. And the core reason for the deduction of processing time is the simplified complexity of the frequency and area of the architecture. The results show that the least error tolerance is 1m by particle filter. Also, the optimum iteration level is 20 to get enhanceable navigation data for AUV.

Keywords: *Embedded system design, Soft core FPGA processor design, parallel processing, underwater detection.*

1. Introduction

Application-specific applications can be developed using customized hardware or software. Achieving a compromise between the two can lead to many benefits. By taking hardware and software factors into consideration, the scientific community proposed the novel concept of reconfigurable computing [1]. These systems integrate the efficiency of hardware implementations in addition to the implementation of programmable processors. This leads to improvements in performance. These systems are also expected to address the difficulties of both the hardware and software as they are a hybrid of both the technologies. Reconfigurable computing was introduced as a concept in early sixties [2]. It was described as a system that employs hardware programmability, customizing the way the hardware is employed for a number of physical control points [2].

A reconfigurable system actually comprises of microprocessors, I/O interface, Reconfigurable Devices (RD) and memory. FPGAs are originally reconfigurable devices that have revolutionized the development of digital systems. The recent developments in the VLSI technology have vastly influenced reconfigurable devices in terms of evolution in the last decade, securing improvements in performance and flexibility. Taking into account the literature available in this field, a number of important surveys such as Tessier and Burleson, Vemuri and Harr, DeHon and Wawrzynek, Compton et al. [1- 4] categorize these devices as digital signal processing or DSP technology between specific ICs and microprocessors. While ASICs are believed to be hardwired devices, nothing can overshadow the importance of the role Programmable Digital Signal Processors (PDSP) plays in the implementation of mechanisms for digital signal processing applications. Researchers also found out that RDs filled the existing gaps between hardware and software such that better performance is rendered than software and it helped maintain a very high level of flexibility.

This paper focuses on parallel and out-of-order execution, register renaming and other established superscalar architectural techniques are used as a base for the creation of the REP. Interestingly, REP design has earlier been created to make instructions fine-grained parallel. The creation took several steps. The first step revolved around the analysis of the nature of coarse-grain level architecture and that of fine-grain superscalar architecture. The second step was to identify the differences between the models of both the architectures. One of the major findings was that the very huge number of inputs and outputs contained by embedded tasks complicates the design that includes dynamic scheduler, task queue and register renaming. The third step is to measure the magnitude of the differences discovered between the two architectures. The fourth step is to take due measures to modify and scale the

architectural techniques in a way that efficiency is not adversely affected.

2. Parallel Computing Architecture

The typical classification of parallel computing architecture was introduced in 1972 by Michael J. Flynn, this architecture came to be known as Flynn's taxonomy [5]. The basis for this classification was whether the architecture is able to execute single or multiple instructions on single or multiple data sets. Using this basis, the classification put forward four categories of computer architecture that are described below:

i. Single instruction single data (SISD) is a computer system that is capable of executing instructions sequentially on a single set of data. This system does not support parallelism.
ii. Single instruction multiple data (SIMD) is an architecture that works with instruction sets using vectors so that a single instruction is executed to all the data in a set simultaneously. A common example using this architecture are multimedia applications making use of computer graphics.
iii. Multiple instruction single data (MISD) is very rarely used. As the name implies, it is used for executing multiple instructions on a single set of data. A common area of use is to determine the fault tolerance on critical systems.
iv. Multiple instruction multiple data (MIMD) is a core multiprocessor systems with multiple cores. This architecture supports parallelism as multiple instructions are being executed on multiple sets of data simultaneously.

In addition, parallelism can be implemented on a number of levels of the system architecture. It can be achieved at instruction level, data level and thread level [6]:

i. Instruction level parallelism determines how many operations can be simultaneously performed by a computer program.
ii. Thread level parallelism, also known as task parallelism, is the parallelism that shares the execution of a computer program among multiple processors.
iii. Data level parallelism achieves the execution of instruction by distributing data among multiple processors.

While it is difficult to differentiate between embedded and general purpose computer systems, embedded ones have continued to scale in terms of complexity, sophistication and scope. The introduction of advancements in semiconductor and information technology has introduced inexpensive computing capabilities. The following outlines the way embedded computing and embedded applications differ from that of general purpose computer systems [7]:

i. Power and Energy Efficiency.
ii. Performance and Predictability.
iii. Cost and Scalability.

FPGAs are being used for the design and development of embedded systems. This is because FPGAs uses 'off-the-shelf' programmability and superior logic capacity that allows it to avoid costs of long manufacturing times. Some of the major FPGA vendors include Synplicity [8-12], these vendors do not only supply FPGA boards but Electronic Design Automation (EDA) tools so that designers can realize the full potential of FPGAs. These tools can be used for designing, placement, simulation, synthesis, routing and verification. In addition, libraries of Intellectual Properties (IPs) and reference designs.

3. Proposed Design

The designed core has incorporated an embedded processor that comprises of a dual-issue processing unit that is effectively pipelined along with all the other functional elements that are needed to create embedded SoC solutions. The REP used for this approach consists of two stages namely dual-issue task fetch and decode unit and load/store operations. Furthermore, it includes individual task and data units as well as timer facilities. Fig. 1 illustrates the logical organization of the REP. The task unit incorporated into the REP is responsible for fetching, decoding and issuing two tasks each cycle to a group of two execution pipelines.

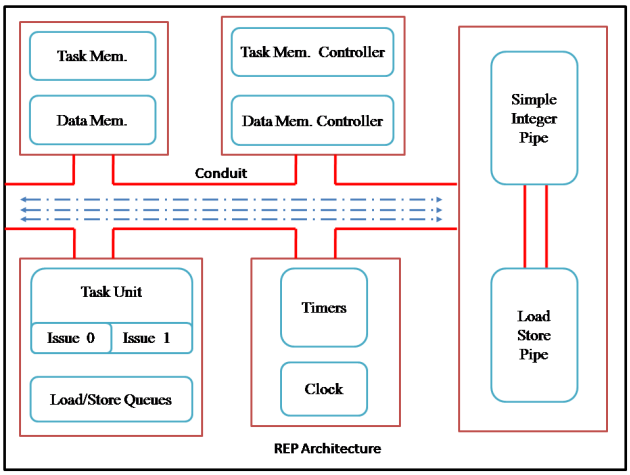

Fig. 1 Block diagram of designed REP [13]

The proposed design of the processor employ data and task transfers so as to secure improvements in terms of

efficiency of data and task transmission between memory and registers. The processor achieves all the interaction over an on-chip connection. The memory and communication systems integrated into the processor offer latencies that can predict and bandwidth that promises to simplify the process of addressing the efficiency and real-time constraints. The task and data memory are made up of 2-port RAM on chip and is created by making use of the Altera® CAD tool as shown in the Fig. 2 and Fig. 3 respectively.

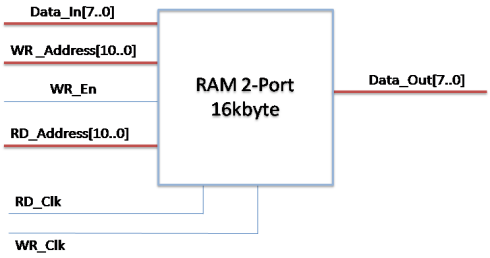

Fig. 2 Block diagram of task memory

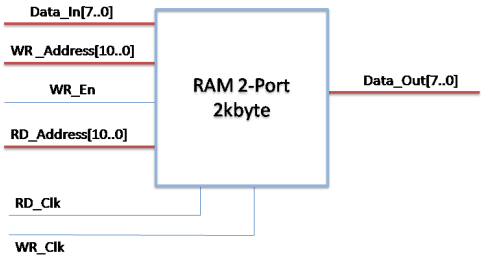

Fig. 3 Block diagram of data memory

Embedded applications extensively make use of threads and loops that comprise of small task working sets and execute extensive sub-tasks locality and reuse. As a result, a large portion of the required bandwidth in embedded applications is contributed by smaller memories comparatively.

The on-chip memory carries out a number of fractions effectively. The most important function is that it offers local memory for retaining all the significant tasks and data sets that fail to meet the capacity of register files that are located close to processors. The local memory is where the REP stores the tasks that are then loaded into the task registers while the thread is being executed. This significantly reduces the predicted impact of energy and performance encountered by loading tasks from memory in the duration of thread execution. The embedded memory being used is also making use of local memory that releases data temporarily when the scenario occurs that it becomes impossible to designate registers to all the available variables in a thread.

The on-chip memory also offer local storage for achieving task and data transfers between the processors residing in the memory and in the core and beyond. The superior storage capacity embedded into the local memory allow designs to incorporate techniques such as double buffering and pre-fetching that ensure computation and communication concurrency. This successfully discourages any remote latency in terms of memory access. The REP architecture used reduces the rate of task and data transmission to an REP by ensuring that multiple processor embedded in an concurrent architecture are accessing the tasks and data. This makes it a promising mechanism that guarantees to secure improvements with regard to task and data delivery. This is achieved by using data parallel threads that are integrated to the REP.

The on-chip memory used allows multiple transactions to be executed simultaneously. Each processor has a separate bank within a core. Processes gains access to its preferred bank by making use of priority before any other channel interface or processors tries to access it. Tasks and data that are unique to a processor are retained by its preferred bank to offer access times that are deterministic. The logic circuit is responsible for assigning accesses to the read and writes ports and is biased. This is done to ensure that an affinity is established between the different memory banks and processors. This is why REP is capable of making such strong assumption in terms of latency and memory availability when data distributed among a number of processors is accessed.

Threads that are executed in an REP interact by means of shared reading and writing variables that are stored in the local memory. However, a drawback associated with interaction through memory is that it demands that threads execute load and store tasks massively. The local communication offers a high-bandwidth interconnection that has a low latency. This ensures that data is transferred efficiently between threads and that they are capable of executing simultaneously on different processors. These interconnections established can also be used to transmit data between threads that are mapped to different processors within an REP. This ensures that the rate of data streaming through memory does not suffer. An implicit synchronization mechanism is also employed by these communication links. This allows data to be transported locally between tasks from a single thread that are distributed and mapped over a number REPs. In addition, injected tasks running on a variety of processors make use of communication to guarantee synchronized execution.

Furthermore, since REP uses dedicated point-to-point links to achieve communication between the processors, the local communication offer links that are configurable

for REP. This allows data streams between processors in different threads to communicate to each other. The mapping of tasks to processors is further simplified by addressing some of the constraints that discourage local communication. This is because the rate at which multiple memory operations are executed and it is also cost effective to transfer data among the different threads working between processors through local communication that is responsible for transferring data to memory. This facilitates the transportation of scalar data between tasks that are carried out on different threads across configurable links. However, the transport latency increases with the increase in distance as the data traverses.

REP actually makes use of two execution pipelines that are simple integer and load/store respectively. Each pipeline makes use of four stages that are capable of accessing the nine ports (six designated to reading and three for writing). The simple integer pipeline is responsible for carrying out the usual arithmetic and logic operations but it does not update the condition register. All the load/store operations are handled by the load/store pipeline. Furthermore, the load//store pipeline also provide provisions for carrying out all the operations that are lying in the big-endian and little-endian data regions. The dynamic data and task controller provided by the REP allows the use of concurrent access and effectively handles all the pipeline stalls, as shown in the Fig. 4.

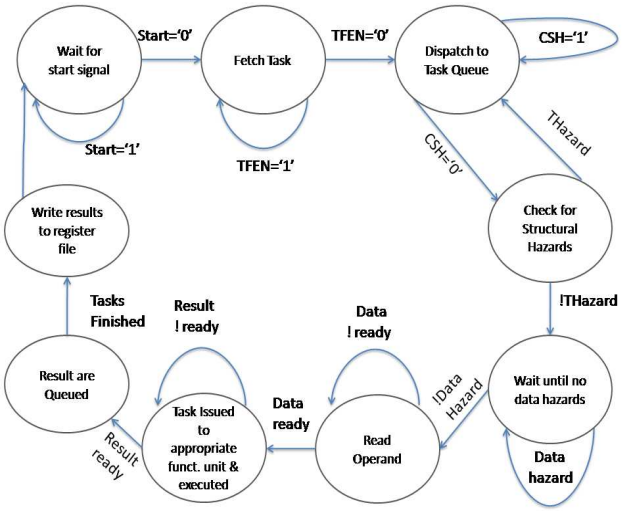

Fig. 4 State diagram of the task and data controller

The task units are responsible for issuing the tasks that are then distributed among the functional units. Shared operand registers provide cost effective solutions for employing short-term locality and reuse provide close to functional units. Tasks are coupled together to form a thread and they all share the local memory. Groups of threads are coupled with distributed memory to form processes. Local communication ensures that the communication links used offer a low latency and high bandwidth and this allows the different registers to communicate to each other.

The REP used makes use of time base and two timers: a Fixed Interval Timer (FIT) and a Decrementer (DEC). The time base is typically a 64 bit counter that increments at a particular frequency that is either equal to the clock input of an asynchronous timer or to the rate of the processor clock of the REP. DEC or Decrementer is a 32-bit register whose rate of decrement equals the rate of increment of the time base. The DEC register actually retains a value to guarantee that the desired interval is created. When the register contains the value zero, there are a number of actions that are triggered: a status bit is set in the Timer Status Register (TSR) and the DEC register will stop decrementing. Moreover, the DEC can also be programmed to reload the value stored in the DECAR or Decrementer Auto-Reload register. In turn, the DEC will again start decrementing. Moreover, the FIT generates the periodic time depending on the chosen bit selected from the time base.

Rep processed the date that coming from Embedded Parallel Systolic Filters (EPSF) [14]. The basic idea is that of chained processing or pipeline which is used as the basic implementation technique. This distributes the execution of a given operation in a series of steps where efforts are made to ensure that all the steps are of the same time duration. The cycles it takes to execute a single data packet is less and the overhead introduced can be effectively amortized by using parallel operations. A synchronous pipeline has a number of processing stages where each stage represents a different stage of processing. The stages are distinguished by embedding a number of additional registers. However, the use of pipeline or chained processing may introduce a number of conflicts in terms of data and control. These conflicts are then addressed in the proposed design by employing appropriate circuitry structures [14].

To counter this problem, the status of each stage is indicated using flag bits. If the flag bit is 0, this implies that the stage has finished the process and can progress with the next data otherwise the data from the previous unit will be retained in the delay unit for 100 ns each cycle. This will make the previous stages work with smaller clock cycles until the flag bit is again 0. This part of embedded design makes use of the most commonly used filters Kalman Filter, Extended Kalman Filter, Unscented Kalman Filter, and Particle Filter for data processing in navigation and tracking systems for robot [15- 24].

4. Design Validation

The most important aspect to keep in mind for design testing is the range information. Ping and other synchronized clocks are used at specific intervals to determine ranges. The echo of the ping for different directions is listened by the sensors and the round trip times are then used to compute the ranges. The important parameters for range finding are used to determine the frequency of the ping. A reverse relation exists between the signal penetration and frequency due to the characteristics of water. High frequency implies lower penetration and higher absorption as compared to the infra frequency which also has high penetration [25]. 1Hz is the frequency chosen for range finding involved in design validation. Using this range, it is possible for the system to self localize the AUV.

In design validation , one significant question is how many iterations are needed to achieve a chosen level of precision in the results. More iterations in the sample lead to greater precision until reach saturation point that extra iterations are wasted system time. The proper number of iterations to reach acceptable level of precision in the results by REP is 20 iterations. REP starts from 5 iterations until reach study state, as shown in the Fig.'s 5, 6, 7, 8, 9 and 10. There are five plots in each figure, four of them representing filtered data by EPSF and supplying to REP to get target distance from AUV. The fifth one is the actual target distance. Figures show that tolerance in the REP's throughput is decreasing while iteration increased until reach 20 iteration. Furthermore, figures show that the most acceptable data filtered in the design is PF. This is because the PF reacts relatively accurate than KF, EKF and UKF.

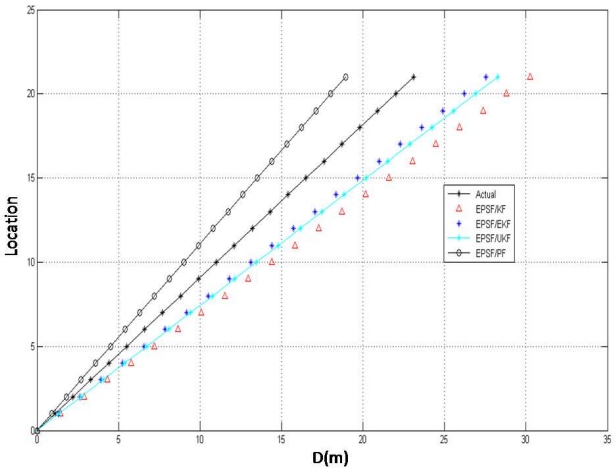

Fig. 5 Distance for oncoming target to AUV for 5 iterations

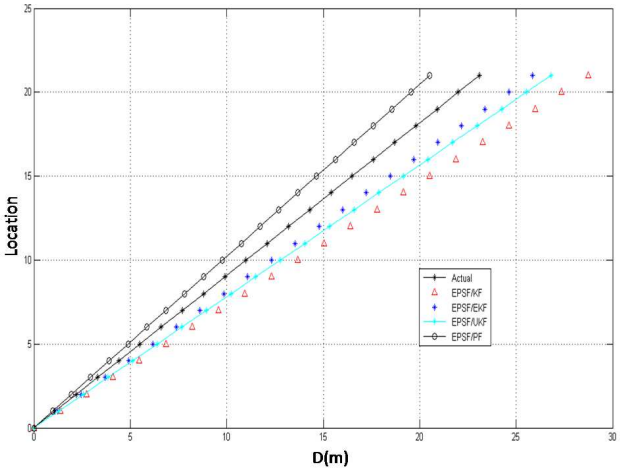

Fig. 6 Distance for oncoming target to AUV for 10 iterations

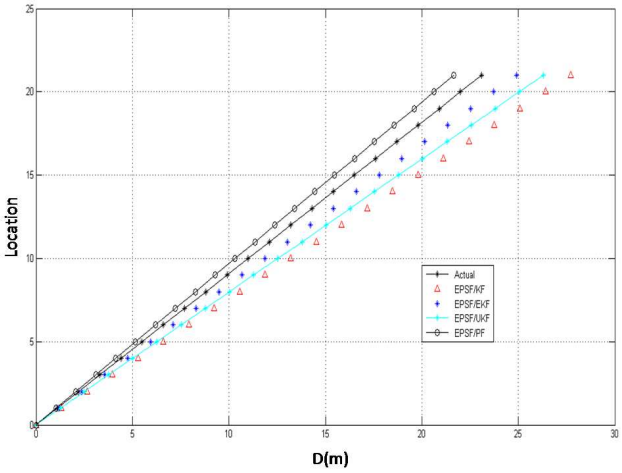

Fig. 7 Distance for oncoming target to AUV for 15 iterations

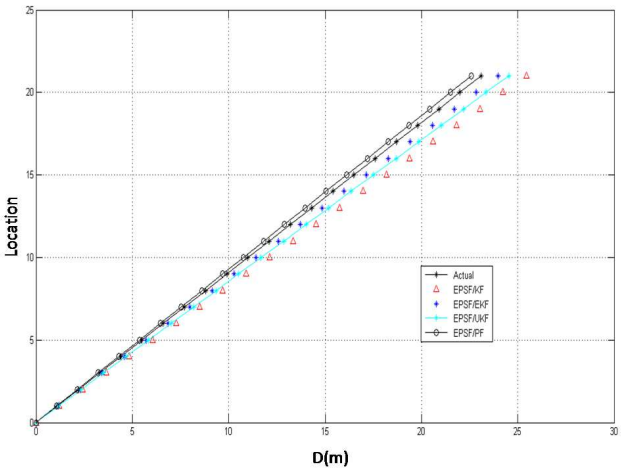

Fig. 8 Distance for oncoming target to AUV for 19 iterations

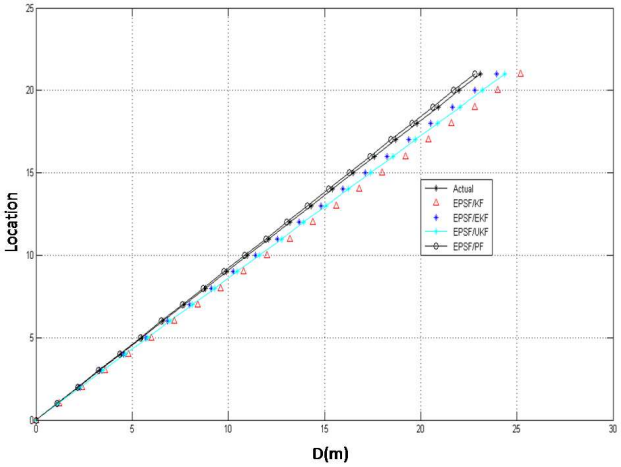

Fig. 9 Distance for oncoming target to AUV for 20 iterations

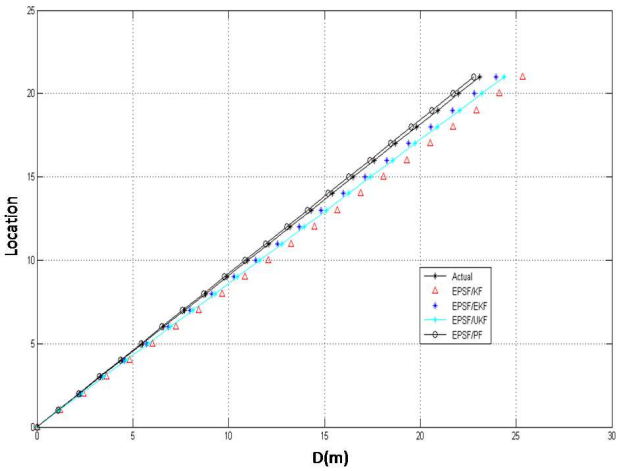

Fig. 10 Distance for oncoming target to AUV for 21 iterations

The good performance of the prescribed design is secured for a tolerance of 2 m for 25 m range. The PF module offers minimum tolerance that is 1 m in this case and this is logical as the error is minimized precisely as shown in Fig. 9. In addition, the test provides sensor signals that are used on Matlab files by employing the same filters that help to evaluate the comparison for the specific filters used. Results also prove that EPSF is more accurate as compared to Matlab as shown in Fig.'s 11, 12, 13, and 14.

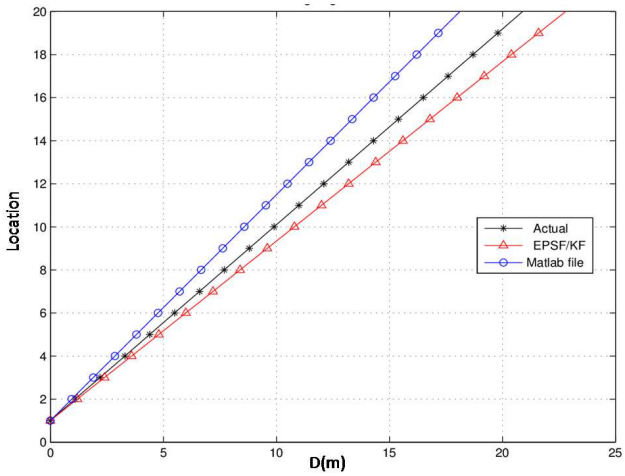

Fig. 11 Range comparison between KF and Matlab for 20 iterations

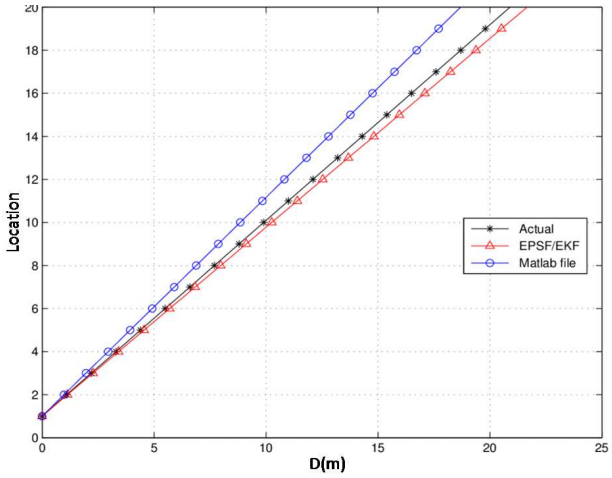

Fig. 12 Range comparison between EKF and Matlab for 20 iterations

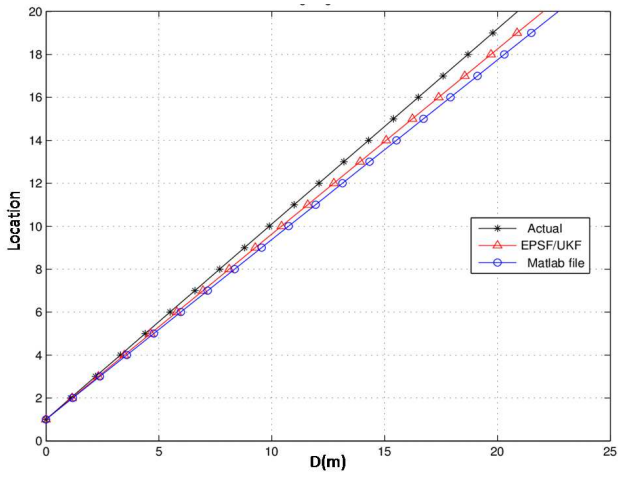

Fig. 13 Range comparison between UKF and Matlab for 20 iterations

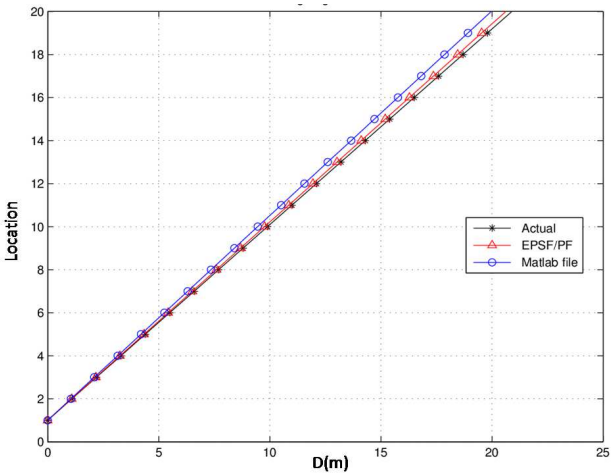

Fig. 14 Range comparison between PF and Matlab for 20 iterations

5. Conclusion

The new architectural design brought a set of improved processes that exploit locality, reuse and abundant parallelism that are rife in embedded applications. Such a design was hierarchically organized and distributed. Also, the novel architectural structure allowed a far more efficient scheme of mapping applications to massive parallel systems through a wide range of mechanisms. The improvement in efficiency was also a function of design that orchestrated and scheduled the motion of data and tasks and significantly controlled the placement of data and tasks too.

Although this paper mostly contemplated efficiency, the relative complexity of mapping applications to design informed many of the architectural decisions. Mapping applications from high level descriptions of systems to detailed implementations imposed substantial engineering and time costs. The architecture attempted to simplify the complex and arduous task of that must satisfy real time performance constraints to programmable architectures.

References

[1] DeHon, A., & Wawrzynek, J.” Reconfigurable computing: what, why, and implications for design automation”, Paper presented at the Proceedings of the 36th annual ACM/IEEE Design Automation Conference, 1999.

[2] Compton, K., Li, Z., Cooley, J., Knol, S., & Hauck, S. “Configuration relocation and defragmentation for run-time reconfigurable computing”, IEEE Trans. Very Large Scale Integr. Syst., 10(3), 209-220, 2002.

[3] Tessier, R., & Burleson, W. “Reconfigurable Computing for Digital Signal Processing: A Survey. J. VLSI Signal Process. Syst"., 28(1/2), 7-27, 2001.

[4] Vemuri, R. R., & Harr, R. E. “Configurable computing: technology and applications”, Computer 33(4), 39-40, 2000.

[5] Flynn, M. J. “Some computer organizations and their effectiveness”, IEEE Transactions on Computers, 21(9), 948–960, 1972.

[6] Hennessy, J. L., & Patterson, D. A. " Computer Organization & Design: The Hardware/Software Interface" Morgan Kaufmann publishers, Inc. 1997.

[7] Sass, R., & Schmidt, A. “Embedded Systems Design with Platform FPGAs: Principles and Practices" 2010.

[8] Synplicity. Information available at http://www.synplicity.com. Access Date at July 2012, from http://www.synplicity.com.

[9] Altera. Information available at http://www.altera.com. Access Date at July 2012., from http://www.altera.com.

[10] Xilinx. Information available at http://www.xilinx.com. Access Date at July 2012, from http://www.xilinx.com.

[11] Cadence Information available at http://www.cadence.com. Access Date at July 2012.

[12] Graphics, M. Information available at http://www.synplicity.com. Access Date at July 2012, from http://www.synplicity.com.

[13] Salih, Muataz H., R. B Ahmad, Abid Yahya & Mohd. Rizal Arshad, "Embedded Concurrent Computing Architecture using FPGA", Proc. of the 2012 7th International Conference on System of Systems Engineering, Genoa, Italy - 16-19 July 2012.

[14] Salih, Muataz H., R. B Ahmad, Abid Yahya & Mohd. Rizal Arshad, " FPGA Design and Implementation of Multi-Filtering Techniques using Flag-Bit and Flicker Clock", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 4, No 2, July 2012.

[15] Brown, R. G., & Hwang, P. Y. C. " Introduction to Random Signals and Applied Kalman Filtering (3rd Ed ed.)", New York: John Wiley and Sons, 1997.

[16] Doucet, A., Godsill, S., & Andrieu, C. " On sequential Monte Carlo sampling methods for Bayesian filtering" Statistics and Computing, 10(3), 197-208, 2000.

[17] Salcic, Z., & Chung-Ruey, L. "Scalar-based direct algorithm mapping FPLD implementation of a Kalman filter", Aerospace and Electronic Systems, IEEE Transactions on, 36(3), 879-888, 2000.

[18] Wan, E. A., & Van Der Merwe, R. "The unscented Kalman filter for nonlinear estimation. Paper presented at the Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000.

[19] Haykin, S. "Adaptive Filter Theory [Book Review]" Signal Processing Magazine, IEEE, 19(4), 87-88, 2002.

[20] Zia, K., Balch, T., & Dellaert, F. "A Rao-Blackwellized particle filter for EigenTracking", Paper presented at the Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, 2004.

[21] Meiliang Wu, & Andrew W. Smyth, "Application of the unscented Kalman filter for real-time nonlinear structural system identification", Structural Control and Health Monitoring, 14(7), 971-990, 2007

[22] Eleni N. Chatzi, & Andrew W. Smyth, "The unscented Kalman filter and particle filter methods for nonlinear structural system identification with non-collocated

heterogeneous sensing", Structural Control and Health Monitoring, 16(1), 99-123, 2009.
[23] Fayomii, C. J. B., Sawan, M., & Bennis, S. "Parallel VLSI implementation of a new simplified architecture of Kalman filter", Paper presented at the Electrical and Computer Engineering, 1995. Canadian Conference on, 1995.
[24] Sau-Gee, C., Jiann-Cherng, L., & Chieh-Chih, L. "Systolic implementation of Kalman filter", Paper presented at the Circuits and Systems, 1994. APCCAS '94., 1994 IEEE Asia-Pacific Conference on, 1994.
[25] Wadoo, S., & Kachroo, P. "Autonomous Underwater Vehicles: Modeling, Control Design and Simulation", Califonia: CRC p.112-134, 2010.

Muataz H. Salih received the B.Sc. and M.Sc. degrees from the Department of Computer Engineering from University of Technology, Baghdad, Iraq, in 1998 and 2002, respectively. From September 1998 to March 2003, he was a research engineer in Military Industrialization Corporation of Iraq. From October 2003 to June 2008, he was a lecturer and manager of engineering faculty's LABS in the faculty of Engineering of Al-Kalamoon private university, Derattiah, Syria. Currently, he is a Ph.D student at Universiti Malaysia Perlis (UniMap), Kuala Perlis, Malaysia. His research interests on designing digital systems using FPGA technology, embedded systems, computer system architecture, microprocessor architecture, computer interfacing, active jamming system for laser missiles, and real time systems.

R.B. Ahmad obtained B. Eng. in Electrical & Electronic Engineering from Glasgow University in 1994. He obtained his M.Sc. and PhD in 1995 and 2000 respectively from University of Strathclyde, UK. His research interests are on computer and telecommunication network modeling using discrete event simulators, optical networking & coding and embedded system based on GNU/Linux for vision. He has five (5) years teaching experience in University Sains Malaysia. Since 2004 until now he is working with University Malaysia Perlis (UniMAP). Currently as the Dean at the School of Computer and Communication Engineering and Head of Embedded Computing Research Cluster.

Abid Yahya earned his B.Sc. degree from University of Engineering and Technology, Peshawar, Pakistan in Electrical and Electronic Engineering majoring in telecommunication. Dr. Abid Yahya began his career on a path that is rare among other Researcher executives and earned his M.Sc. and Ph.D. degree in Wireless & Mobile systems, in 2007 and 2010 respectively, from the university Sains Malaysia, Malaysia. Currently he is working at School of Computer and Communication Engineering, university Malaysia Perlis (UniMAP). His professional career outside of academia includes writing for the International Magazines, News Papers as well as a considerable career in freelance journalism. He has applied this combination of practical and academic experience to a variety of consultancies for major corporations.

Mohd Rizal Arshad graduated from the University of Liverpool, in 1994 with a B.Eng. In Medical Electronics and Instrumentation. He then pursues his MSc. in Electronic Control Engineering at the University of Salford, graduating in Dec. 1995. Following from this, in early 1999, he continues with a PhD degree in Electrical Engineering, with specialization in robotic vision system,. Since then, he has been working at the Universiti Sains Malaysia (USM), Malaysia as a full-time academics, i.e. lecturer and researcher. He has supervised a number of postgraduates students at the MSc. and PhD. levels. He has also published actively in local and international publications. Dr. Mohd Rizal Arshad is currently an Associate Professor and the deputy dean of the School of Electrical and Electronic Engineering, USM. And with his team of researcher, is also the pioneer of underwater system technology research efforts in Malaysia.