FPGA Design and Implementation of Multi-Filtering Techniques Using Flag-Bit and Flicker Clock

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

Real time system is a condition where the processor is required to perform its tasks within a certain time constraints of some processes or simultaneously with the system it is assisting. Typically, it suffers from two main problems; delay in data processing and complexity of the decision-making process. The delay is caused by reasons such as computational power, processor unit architecture, and synchronization signals in the system. To improve the performance of these systems in term of processing power, a new architecture and clocking technique is realized in this paper. This new architecture design called Embedded Parallel Systolic Filters (EPSF) that process data gathered from sensors and landmarks are proposed in our study using a high-density reconfigurable device (FPGA chip). The results expose that EPSF architecture and bit-flag with a flicker clock achieve appreciably better in multiple input sensors signal under both incessant and interrupted conditions. Unlike the usual processing units in current tracking and navigation systems used in robots, this system permits autonomous control of the robot through a multiple technique of filtering and processing. Furthermore, it offers fast performance and a minimal size for the entire system that minimizing the delay about 50%.

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

1. Introduction

Modern real time systems and applications depend on a sufficiently high processing throughput and massive data bandwidths needed in computations [1]. The need for real-time, high performance computational algorithms and architectures is one of the driving forces of modern signal processing technology and is behind the expansion of the semiconductor industry. The semiconductor industry is developing new products at an enormous pace driven by the Moore's law [2]. The symbiotic integration of once dissimilar memory and logic processes is very promising for processor array and memory integration on a single chip, which is the key to reliable massively parallel systems [3, 4].

With tremendous advances in the VLSI technologies, new horizons have opened. Developments in the integrated circuit technology has led to a rising interest in parallel or highly concurrent algorithms and architectures [5, 6]. As a result of large available transistor counts on a single chip, different projects have emerged with the vision of integrating processor and memory on a single chip [7]. Current high-density devices (FPGA) provide the possibility for mixing memory and logic processes on the same chip. Many SoC projects and studies have been done in the past few years to show benefits of system-scale integration [8]. Most of the studies have been dealing with vector processors or small-scale MIMD processor systems. The two well-known projects are IRAM (Intelligent RAM) [9] and CRAM (Computational RAM) [10]. These projects may mark the real start of different parallel SoC systems, indeed also systolic arrays.

This paper focuses on multi-filtering techniques by systolic arrays, where algorithm execution can be simultaneously triggered on all data elements of data set with different clock cycles. The advantage of such arrangement is that the replication of execution resources is employed. Each datum or part of a data set can be associated with a separate processing element. Processing elements form a parallel processing ensemble that is triggered by multi-clock, where each processing element operates on a different data element.

2. Systolic Arrays and Filters

Research in the area of systolic arrays began at the end of the 1970s [11]. The original motivation behind the systolic array concept was its potential for very-large-scale integration (VLSI) implementation. Only a few systolic algorithms have been implemented in VLSI chips. The main obstacle is their limited flexibility, as general-

purpose (high volume) designs are the main driving force for commercial use. The second problem is the available technology at the time of introduction. Nevertheless, several multiprocessor projects have been directly inspired by the systolic array concept such as Warp Processor developed at Carnegie-Mellon, the Saxpy Matrix-1, or the Hughes Research Labs Systolic/Cellular System. Some other projects are covered in [12, 13].

Systolic algorithms are concurrent versions of sequential algorithms suitable to run on array processors that execute operations in the so-called systolic manner. Systolic arrays are generally classified as high-performance, special-purpose VLSI computer systems suitable for specific application requirements that must balance intensive computations with demanding I/O bandwidths. Systolic arrays are tremendously concurrent architectures, organized as networks of identical and relatively simple processing elements that synchronously execute operations. Modular processors interconnected with homogeneous (regular) and local interconnections provide the basic building blocks for a variety of algorithms. Systolic algorithms address the performance requirements of special-purpose systems by achieving significant speedup through parallel processing and the prevention of I/O and memory bandwidth bottlenecks. Data are pumped rhythmically from the memory through the systolic array before the result is returned to the memory. The global clock and explicit timing delays synchronize the system.

The systolic array is a computing system that possesses several features amenable to system-on-a-chip (SoC) designs [14]:

1. Network: It is a computing network employing a number of processing elements with local interconnections.
2. Homogeneity: Interconnections between processing elements are homogeneous (regular). The number of interconnections between processing elements is independent of the problem size. This is the first important feature that we exploit for the SoC design.
3. Locality: The interconnections are local. Only neighboring processing elements can communicate directly. This is the second important feature required to achieve high-speed VLSI SoC realizations.
4. Boundary: Only boundary processing elements in the network communicate with the outside world. This eliminates the classic memory and I/O bandwidth bottleneck.
5. Modularity: The network consists of one or, at most, a few types of processing elements. If there is more than one type of processor, the systolic array can usually be decomposed into distinct parts with only one processor type. This feature enables quick and high-performance SoC designs.
6. Rhythm: Data are computed and passed through the network rhythmically and continually.
7. Synchrony: A global clock synchronizes the execution of instructions and data interchange between processing elements.
8. Expendability: The computing network may be extended arbitrarily.
9. Pipelineability: Pipelining on the array level, that is, between processing elements, is present.

From the features presented, we can summarize that a large number of processing elements working in parallel on different parts of the computational problem is functional. Data enter the systolic array only at the boundary. Once placed into the systolic array, data are reused many times before they become output. Several data flows move at constant velocities through the array and interact with each other where processing elements execute the same function repeatedly. Only the initial data and results are transferred between the host computer/global memory and the systolic array.

Certain constraints should be kept in mind and incorporated into the design methodology for VLSI-SoC implementation. The most important are short communication paths, limited I/O interaction, and synchronization. These constraints are inherent features of systolic arrays. Features like homogeneity, modularity, and locality are especially favorable from the VLSI-SoC point of view and make systolic arrays ideal candidates for SoC implementation [12, 13, 14]. Each systolic algorithm viewed from the processing element perspective includes the following three distinct processing phases:

1. Data input: A new data item is input into the processing element from either the processing elementneighbors or the global memory of the bordering processing elements of the systolic array.
2. Algorithm processing: The algorithm is processedas specified by the processing element definition of the systolic algorithm.
3. Data output: The result of the algorithm data processing phase is output to the neighbors of

the designated processing element or to the global memory for bordering the processing elements of the systolic array.

Systolic arrays are finding their way into many practical applications. In recent years, several applications of systolic arrays have been presented, ranging from radar signal processing to low-level image processing problems [12]. Typical applications include image sequence analysis, DNA sequence analysis, visual surface control, image processing, cryptography, and computer tomography. General-purpose programmable systolic arrays are also found in the market. Recently, Systolic launched PulseDSP architecture, which uses a large number of highly efficient processors arranged in a systolic array. The first PulseDSP-based product is a wide bandwidth sigma-delta A/D converter from Analog Devices (AD7725) [15].

Full custom designs of systolic algorithms are also available. For example, low-level image processing algorithms for motion estimation are already implemented in hardware as systolic array structures [16].

Over the past forty years, adaptive filters have been widely used in many applications such as target tracking, navigation systems, adaptive control and many other dynamic systems.

2.1 Kalman Filter

In recent years, Kalman filters have been widely used in many applications such as target tracking, navigation systems, adaptive control, and many other dynamic systems. The Kalman filter algorithm is based on minimizing the mean square error recursively. Therefore, it can work well in estimating the unknown states of a noisy dynamic process [17].

Many attempts have been made to employ various systolic architectures for the VLSI implementation of Kalman filters [18, 19]. These methods, along with dozens of others presented in the literature, are merely for permanent structures and are only suitable for VLSI implementation. Rapid prototyping of a Kalman filter-based system requires the implementation to be parameterized. Systolic-based architectures should be modified to meet the hardware requirements of the FPGA technology [20].

2.2 Extended Kalman Filter

What happens if the process to be estimated and (or) the measurement relationship to the process is non-linear? Some of the most interesting and successful applications of Kalman filtering have been in such situations. A Kalman filter that linearizes the current mean and covariance is referred to as an extended Kalman filter or EKF. The state transition and observation models need not be linear functions of the state but may instead be (differentiable) functions [17].

3. Proposed Design

The conception of systolic or chained processing can be described as the implementation technique that partitions the execution of a given operation into the number of subsequent steps as far as possible from the same duration. Furthermore, each section assigned to a particular step can exploit standalone technical resources. Executions of single packet data requires less clock cycles, and overhead can be effectively hidden by a parallel operation. Synchronous systolic is composed of stages, each of which is dedicated to a different stage of processing. Individual stages are separated by embedding additional registers. As addressed in our design using appropriate circuitry structures, systolic or chained processing might impose different types of conflicts (data and control).

To solve this problem, a flag bit is assigned for each stage indicating its status. A flag bit of 0 means the stage has finished the process and is ready to accept other data; otherwise, data coming from the previous stage will enter in a delay unit of 100 ns each cycle and will make the previous stages work on another smaller clock cycle at 50 ns until the flag bit becomes 0. This embedded architecture uses the most familiar filters for data processing in navigation and tracking systems for robots.

We propose the development of a homogeneous, modular, expandable systolic array combined with a large global memory. There are three main reasons for the suitability of the SoC integrated systolic array using this memory system:

1. Off-chip memory buses are slow. A systolic array inherently depends on a continuous supply of data from the memory. Inability to realize this at a fast enough data rate can cause significant performance degradation.
2. Off-chip inter-processor element communication slows data exchange. This ultimately slows down the entire systolic array.
3. Package pin-count limitation. A rectangular systolic array can be connected to the memory system on all four borders. Using a narrower memory interface can again cause a slowdown.

Integration of the systolic array and memory on the same piece of real state alleviates these problems. Homogeneity of the systolic array is a very important factor in the design of such a system as it provides the following benefits:

1. Only one processing element design is reused in the entire systolic array.
2. Performance balances linearly with the number of processing elements in the systolic array.
3. The systolic array design cycle can be shortened by reusing the same processing elements forming the processing ensemble.

The suitable features of systolic arrays for SoC can be summarized as follows:

1. Control of a systolic array is not limited to any particular implementation.
2. The number of functional units within the scalar part of the architecture can be arbitrary.
3. The topology of the physical interconnections between processing elements is not limited to a specific network.
4. Due to synchronous data injecting through the systolic array, a separate global data memory bank is assigned to each row/column of processing elements to ensure conflict-free access. Furthermore, there is no need for additional crossbar buses between the memory and the processing element array as, by definition, only the nearest neighbor data transfers are allowed in systolic processing.

A new technique of embedded parallel systolic filters (EPSF) is proposed in this paper and is depicted in Fig. 1. The EPSF combines the multiple Parallel Element (PE) layers of the original systolic array into a single PE layer with a set of feedback registers in a pipeline structure.

Data are originally passed to stage B for pre-processing, and the flag-bit is then observed to decide whether to enter stages A or the delay unit until the flag-bit changes its status. An additional unit of control signal digital circuit is required to produce the control signals for data input selection, cell memory clearance, and operation mode control.

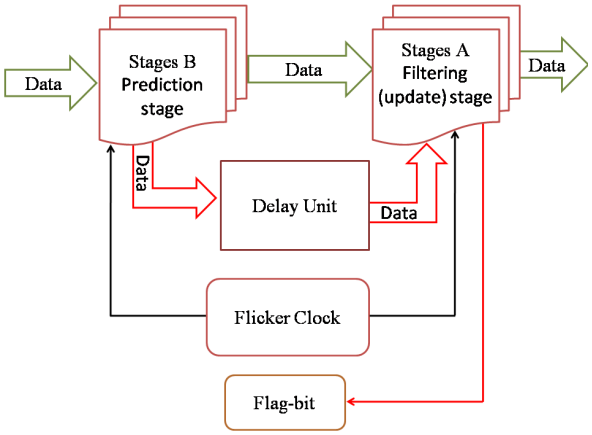

Fig. 1 EPSF design architecture

4. FPGA Implementation

The throughput of the systolic array can be limited due to various reasons: low systolic array efficiency, data synchronization between parts of the systolic array with different types of processing elements (e.g., RLS), data dependencies within the processing element, and long latency of operations within the processing element. Systolic array efficiency and systolic cycle length depend on the complexity of the systolic algorithm and implementation of the processing element. Both may bind the systolic array throughput. To solve these problems, several systolic algorithm transformation techniques have been devised that operate on the algorithm level.

The main problem addressed in this section is the possibility of increasing the throughput of systolic arrays beyond the limit set by a systolic cycle without applying systolic algorithm transformation/mapping techniques. This can be achieved by combining embedded multithreading and systolic array computing called EPSF. Unlike classic algorithm-level transformations, multithreading can increase the throughput of the systolic array without changing the original systolic algorithm. If a certain algorithm is highly effective, the incorporation of multithreading on the pipelined systolic processing elements can improve the throughput of the same algorithm by a constant factor. Generally, multiple independent algorithm threads (i.e., instances of the same algorithm) are interleaved within a given systolic cycle on the same systolic array. Data from multiple threads are pumped through the multithreaded systolic array in packets resulting in a dramatic improvement of the net throughput. All threads share the same resources. Several sources of the data sets available within systolic arrays that can be treated as unrelated threads can be

defined as follows:

1. Data vectors from different algorithm partitions.
2. Loop unrolled systolic processing element algorithm.
3. Multiple instances of the same algorithm.
4. Simultaneous execution of different types of algorithms.
5. Suitable combinations of the previous four.

Data streams from systolic algorithms execute operations without noticing the presence of others. All data streams share processing element resources including inter-processing element communication data paths. It should be noted that we assume the same processing element I/O bandwidth, that is, the same bisection bandwidth as in the original single threaded systolic array. The performance increase is due to the elimination of true data hazards within each algorithm, better functional unit utilization, and larger amounts of usable instruction level parallelism uncovered within longer basic blocks constituting instruction loops. Functional unit pipelines within processing elements are kept busy by interleaving independent threads running simultaneously through the multithreaded systolic computation. A side effect of multithreading is that as the efficiency of each processing element improves, a group of algorithms can complete the execution in a shorter time than it would in the serial case.

We examined the multithreading systolic computation logically, where all threads are concurrently executed on the systolic array with the granularity of one processing element clock cycle. Implementing a multithreaded design uses a data packet transfer approach. For each iteration of the systolic program, *N* data elements from all *N* threads are input, processed, and output.

Data elements of the input data vectors of *N* threads are time multiplexed into a multithreaded systolic array at a rate of one element per processing element clock cycle. Data elements of the result vectors are output at the same rate. This process constitutes a multithreaded systolic cycle. It repeats for all subsequent elements and all threads. As threads are independent data sets, no data dependencies are present within the processing element pipelines. The implementation of the EPSF is performed with the memory components created inside the FPGA chip. Memory is used for frame and parameter buffers, while other circuits on the FPGA are used for pixel and parameter calculations.

The system design consists of two main modules representing the filters. Each filter contains two systolic stages for filter calculations. The 32-bit input data represent data coming from the array of sound sensors (receivers) becoming more sensitive and directive, enabling the system to discriminate between sounds coming from different directions. The strategy used to build our architecture in FPGA is a system design using components or a piece of conventional code (LIBRARY declarations ENTITY ARCHITECTURE). However, by declaring such code as a COMPONENT, it can then be used within another circuit, thus allowing the construction of hierarchical designs [21]. A COMPONENT is also another way to partition a code, which allows code sharing and code reuse. For example, commonly used circuits such as flip-flops, multiplexers, adders, basic gates, and others, can be placed in a LIBRARY; therefore, any project can use them without explicitly rewriting the codes.

Many FPGA design tools provide built-in modules for arithmetic operations. These modules are usually optimized for the target architecture. To carry out the scalar division in the boundary cell for the matrix inversion, Z=x/y, where $0 <= |\mathbf{x}| < |\mathbf{y}| <= 1$, and **x**, **y,** and **z** are 16-bit numbers. **x** has to be zero-padded to 32-bit long and then passed to a 32-bit divider. This arrangement occupies 1,120 logic elements (LE) and is executed at the maximum clock rate of 27 MHz in an Altera Cyclone II device. The 16-bit finite word length induces a negligible precision error of only 0.0012%.

FPGA devices provide only limited memory, which limits the size of the LUT for this purpose. The value of **1/y** falls into a decreasing quadratic curve, while **y** tends to one. Thus, the value difference between two consecutive numbers of **1/y** decreases dramatically. To reduce the size of the LUT, the inverse value curve can be segmented into several sections with different mapping ratios. This can be achieved by storing one inverse value, the median of the group, in the LUT to represent the results of **1/y** for a group of consecutive values of **x**.

By applying LUT to the EPSF structure, a generic hardware design is established for matrix inversion in different sizes. The result of matrix operations, including addition, subtraction, multiplication, and inversion, is calculated. For an n×n matrix, there are a total of 2n PEs in the EPSF with 1 boundary cell, 2n – 1 internal cells, and 2(n – 1) layers of registers. The processing time requires $2(n^2 - 1)$ clock cycles. This implementation can run with a maximum clock frequency of 50 MHz for n<64.

Flicker clock management can have a strong impact on the reduction of processing delay in FPGAs. In most modems, FPGAs have dedicated clock managers for solving high-speed clock distribution problems in high-density designs. This design uses an indicator for systolic stages of the filters called flag-bit, which works when a delay occurs in

any filter stage. Flag-bit gives design high control on data transmission and early alerts to prevent data conflict. This new technique is very efficient and reliable for parallel processing systems and gives the best results to satisfy parallelism.

Several clock managers have been introduced to FPGA chips. They can perform clock buffering, drive the distribution networks, and simultaneously eliminate clock skew. They can also produce phase shifts and duty cycle adjustment. In addition, clock managers can be used to synchronize several components in a system. Although they perform their intended function well, current FPGA clock managers are incapable of performing dynamic clock management because their dividers cannot perform dynamic division or multiplication.

The Xilinx and Altera clock managers can only be programmed during the initial configuration. The Lucent Programmable Clock Manager can be programmed during its operation, but this can lead to dangerous clock outputs. Changing the settings of existing clock dividers during their operation can lead to metastability and latching errors due to glitches, distortions, asymmetry, transient frequencies, and additional clock edges of the output clock signal. Even shutting off the system during the change to the new frequency does not help in this case, as the inconsistent duty cycle clocks may be non-transient.

If a flicker clock is added to the existing FPGA clock managers, they can be used for clock management. As a result, clean frequency changes can take effect within a clock cycle. The flicker clock circuit of Fig. 2 is capable of performing efficient clock signals without undesired effects at the output.

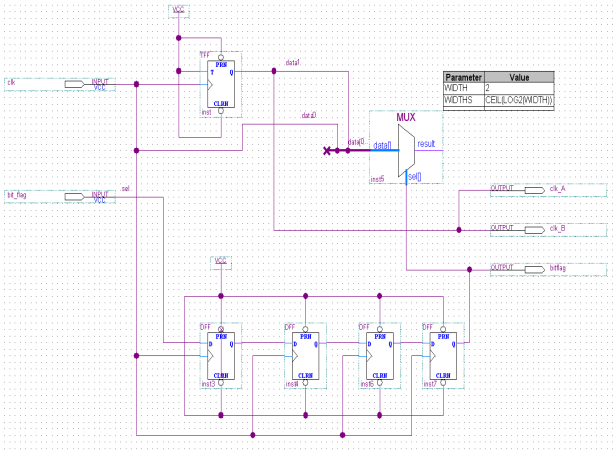

Fig. 2 Flicker clock circuit

Division of the input clock is performed by creating a loop of T-flip-flops driven by the input clock. To create the necessary clock, more than one clock signal must enter into the multiplexer that feeds the systolic stages through necessary clock signals dependent on the status of the flag-bit. The delay between the output of the Kalman filter and the extended Kalman filter is approximately 20 ns due to the use of the flag-bit and flicker clock techniques, as shown in Fig. 3.

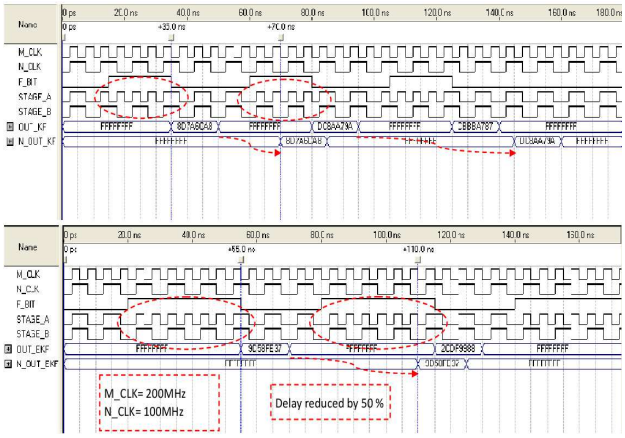

Fig. 3 KF and EKF output with\out flag-bit and flicker clock effects

Clock signal of the systolic stage A flicks directly to the smaller clock signal after the flag-bit indicates logic 1, which means the stage has a delay in processing data and returning to the original clock signal after the flag-bit changes its status, as shown in Fig. 3.

Benchmark is carried out to compare the EPSF performance with other systems. For an n×n matrix, the EPSF requires 2n PEs while in [18], {n× (3n + 1) / 2} PEs are used. EPSF allows the clock speed to run at 50 MHz compared with the maximum clock frequency of 2 MHz in the VHDL design presented in [20] and 10 MHz in the Geometric Arithmetic Parallel Processor (GAPP) in [18]. When it comes to resource consumption, the EPSF approach is still superior compared with those presented in the literature. When working with an 8×8 matrix, it takes 4,314 LE to implement the EPSF in the Altera Cyclone II device, while 8,610 LE is required to implement the hardware design as reported in [22].

The results demonstrating the speedups achievable on an EPSF computation run the demonstrated design on selected processing element models. Speedups are based on the execution time ratios of EPSF versus traditional computing. We show that a collection of data executing simultaneously on the systolic array is faster than the serial execution of the same data on a single array. Each task will be executed separately in a shorter time due to the flag-bit and flicker clock techniques. The processing time

will be longer to produce an output without using these techniques, and EPSF computation creates the following effects:

1. Processing element utilization is increased, as there are many independent operations available, which can be executed concurrently.
2. The net throughput increases as a direct consequence of multithreading. As long as there are enough systolic functional units and inter-processing element communication channels, the speedup curve can experience a proportional increase.
3. The need for register storage increases proportionally to the number of threads.

Multithreading is very effective in increasing the throughput and utilization of systolic arrays. A linear increase in the throughput is observed as long as the processing element functional units are pipelined, and the algorithms do not experience very low computation-to-communication ratios.
Multithreaded systolic computation also increases the number of available operations within a given systolic cycle, and as threads are independent of each other, more instruction-level parallelism can be easily extracted from the code. Furthermore, pipelined functional units can be kept busy, producing results with a theoretical maximum throughput rate.

5. Performance Comparison

As reference for the synthesis times and the speedups reported in the current section, Table 1 describes the PC used for design and testing.

Table 1: PC setup used for design and testing

Model	HP Compaq dc7800 Convertible
CPU	Intel®Core™2 Quad Q6600 @ 2.4GHz
RAM	8GB
Operating System	Windows XP Professional SP3

Testing and verification of the design were carried out by implementing the proposed technique for filters on a target reconfigurable platform based on FPGA device. This was accomplished by describing the techniques in VHDL and then synthesizing them for the FPGA chip. One motivation of this work is the evaluation of the filters' architecture in a realistic hardware environment. Therefore, we selected our chip and methodology with the goal of synthesizing the EMPSoC system and running it on designed custom board as shown in Fig. 4. We chose an Altera Cyclone II FPGA chip as the basis of our platform. This high-density device is intended for full SoC implementation and contains a balance of memory and logic resources.

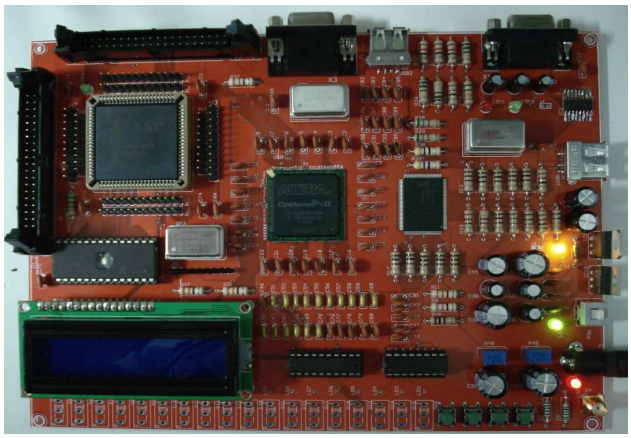

Fig. 4 Designed custom board

The application is targeted for FPGA devices for many reasons. One of the goals of this paper is to design EPSF as part of EMPSoC for underwater applications such as autonomous underwater vehicle (AUV) navigation and tracking using a structural design more robust than behavioral. FPGAs can be considered for high-performance DSP systems. On the other hand, nowadays, FPGA devices offer very attractive hardware facilities; great I/O pin-count, embedded memory blocks, large logic area, high clock speed, and advanced software computer-aided design (CAD) tools are available for assistance in every design stage.

The VHDL language is chosen for its hardware design mainly because of its familiarity but also because of its wide support range. Parameterizable VHDL blocks are implemented, allowing application parameters to be changed. The modularity of the design makes possible the reuse of its parts for other applications. The Altera QUARTUS II 8.0 design software tool is used for design verification. QUARTUS II 8.0 is a CAD design tool that can assist each step in the design-flow and provide an extensive analysis of timing and resource utilization (embedded memory blocks, logic elements, and dedicated multipliers).

Altera's Cyclone II family is chosen as the hardware target because of its accessibility and low cost. Its important characteristics such as embedded multipliers and memory blocks are also influential factors. Synthesis was conducted using an Altera Cyclone II FPGA model number EP2C35F672C6, with the following main specifications:

1. 33,216 LEs
2. 105 M4K RAM blocks
3. 483,840 total RAM bits
4. 35 embedded multipliers
5. 4 PLLs
6. 475 user I/O pins
7. FineLine BGA 672-pin package

The various performance results for each filter architecture are shown in Fig. 5. The total synthesis time, which is the time required by Quartus II 8.0 to compile the VHDL, perform the synthesis stage, and generate a programming file added to the time required to load the logic module flash with configuration data. The maximum number of cells, N_{max}, which fit in the chip for each filter architecture. The maximum clock frequency at which the computing cells can run. This value is reported by the Quartus II 8.0 after synthesis as an estimate of the maximum clock frequency that should remain valid through acceptable device tolerances and operating conditions. Furthermore, the number of clock cycles, N_{cycles}, required to complete one iteration as determined by the number of states in the cell controller's FSM and by the type of operation performed in each state.

	Without Flag-Bit and Flicker Clock		With Flag-Bit and Flicker Clock	
	KF	EKF	KF	EKF
Total Synthesis Time	4.8 min.	9.9 min.	5.2 min.	10.3 min.
Maximum Design Density	1157 LE	1635 LE	1262 LE	1738 LE
Maximum Clock frequency	10 MHz	9.2 MHz	47.8 MHz	46.7 MHz
Clock Cycles per Iteration	7- Normal clk.	11- Normal clk.	3 -Normal clk. 4-Flicker clk	4- Normal clk. 7- Flicker clk.

Fig. 5 Performance Results of Filters Architecture

6. Application Area

One of the main technological challenges associated with the operation of AUVs is navigation [23, 24]. End users need data to be gathered along specific trajectories defined in some local or global reference frame. These trajectories are converted into a path the vehicle has to follow, and this in turn requires the continuous knowledge of its position. On the other hand, there is always the risk of losing an AUV while performing autonomous operations. The external tracking of the vehicle allows for a high degree of confidence, as it becomes possible to monitor its behavior far from the operations area. Sound plays a critical role in most navigation and tracking systems for underwater devices. As electromagnetic waves propagate poorly in seawater, sound waves are generally used to measure the ranges to the beacons installed in known positions.

The navigation task is difficult due to a number of complex problems. Some issues that complicate navigation are the limits on computational power, difficulties in detecting and recognizing objects, difficulties in avoiding collisions with objects, and difficulties involved in using information provided by the environment [25]. For example, the computational power needed to do real-time image processing, computer vision, and learning is high. Although CPUs have become faster, they are still not fast enough to provide the power to perform tasks in the field. The chip technologies (FPGA) used today for processor design will soon reach their limits. According to some, it will take years before new processing technologies can become available. The fundamental way to extract more computation cycles from a given manufacturing technology is to introduce parallelism into the design.

With this in view, designers are seeking to introduce more processing elements into their designs. The speedup in the application is proportional to the effective utilization of the processing elements. Effective utilization depends on the computation and communication behavior of the architecture. This power is increased by using both FPGA and the new technique represented by the flag-bit and flicker clock in our design.

7. Conclusion

The assuming integration of the systolic array and a global memory is presented in this paper through an SoC implementation investigation. Two novel approaches have been presented. First is the EPSF architecture, which is the solution for minimizing the delay occurring in the processing units of the system. In using this architecture, the decision maker module has a variety of information on which to build its decision. We have introduced a system-level technique that takes advantage of the clock managers present in most modem FPGAs. Flicker clock with flag-bit, which is used to reduce the processing delay and enables dynamic operation by eliminating glitches and transient and non-transient divider output errors that are very harmful when introduced into clock signals. The flicker can be included either as part of the FPGA clock managers or as a user circuit.

The results show that synthesis time for filters with our techniques is 5.2 min for KF and 10.3 min for EKF, whereas 4.8min for KF and 9.9 min for EKF without our techniques. Also, N_{max} that fit in the chip for KF and EKF without our techniques is 1157 LE and 1635 LE

respectively, whereas with our techniques is 1262 LE for KF and 1738 LE for EKF. The maximum clock frequency at which the computing cell can run with our techniques is 47.8 MHz for KF and 46.7 MHz for EKF, whereas it is less than 10 MHz without our techniques. Finally, each filter has the same N_{cycles} with or without applying our techniques, but the difference that KF has 4 clk flicker from 7, and EKF has 7 clk flicker from 11. As can be seen, the results expose that this technique can be used efficiently in most FPGA families.

References

[1] Flynn, M., & Luk, W." Computer System Design: System-on-Chip", 2011.
[2] Doug Burger and James R. Goodman. "Billion-Transistor Architectures: There and Back Again". IEEE Computer Society, Vol. 37, Issue 3, pp. 22–28, March 2004.
[3] Hennessy, J. L., & Patterson, D. "Scalable Multi-core Architecture *London: Morgan Kaufmann. p. 213*, 2010.
[4] H. Peter Hofstee. "Power Efficient Processor Architecture and the Cell Processor". In HPCA '05: Proceedings of the 11th International Symposium on High-Performance Computer Architecture, IEEE Computer Society, pp. 258–262, Washington, DC, USA, 2005.
[5] L. Chen and Z. Hu, "Optimizing fast Fourier transform on a multi-core architecture", IEEE International Parallel and Distributed Processing Symposium, Vol.15 no.6, pp.491-504, 2007.
[6] Tarek Abdelrahman, Ahmed Abdelkhalek, Utku Aydonat, Davor Capalija, David Han, Ivan Matosevic, Kirk Stewart, Faraydon Karim, and Alain Mellan. The MLCA: A Solution Paradigm for Parallel Programmable SOCs. In IEEE North-East Workshop on Circuits and Systems (NEWCAS), pp. 253-253, June, 2006.
[7] Hubner, M., & Becker, J. "Multiprocessor System-on-Chip: Hardware Design and Tool Integration *New York: Springer p. 201-245*, 2010.
[8] Farayadon Karim, Alain Mellan, Anh Nguyen, Utku Aydonat, and Tarek S. Abdelrahman. "A Multi-Level Computing Architecture for Embedded Multimedia Applications". IEEE Micro, Vol. 24, no. 3, pp. 55–56, 2004.
[9] D. Patterson, et al., "A case for intelligent RAM", IEEE Micro, vol. 17, no. 2, pp. 34-44, March/April 1997.
[10] Ma, Z., Marchal, P., Scarpazza, D., Yang, P., Wong, C., Gomez, J., et al. "Systematic Methodology for Real-Time Cost Effective Mapping of Dynamic Concurrent Task-Based Systems on Heterogeneous Platforms", New York: Springer. p. 216-267, 2010.
[11] H. T. Kung, C. E. Leiserson "Systolic Arrays (for VLSI)," *Technical Report CS 79-103*, Carnegie Mellon University, 1978.
[12] High Performance VLSI Signal Processing: Innovative Architectures and Algorithms, vol. I, II, Edited by: K. J. R. Liu, K. Yao, IEEE Press, 1998.
[13] Special issue on: Systolic Arrays, Computer, vol. 20, no. 7, July 1987.
[14] N. Petkov, Systolic Parallel Processing, North-Holland, 1993.
[15] www.systolix.co.uk, Page accessed March 2001.
[16] Y. Katayama, T. Kitsuki, Y. Ooi, "A block processing unit in a single-chip processing elementG-2 video encoder LSI", Proc. of SIPS'97, pp. 459-468, Leicester, 1997.
[17] S. Haykin, Adaptive Filter Theory, 4th Edition, Prentice Hall, USA, 2002.
[18] S-G. Chen, J-C. Lee and C-C. Li, "Systolic Implementation of Kalman Filter", Circuits and Systems, APCCAS '94, IEEE Asia- Pacific Conference, pp 97-102, 1994.
[19] C.J.B. Fayomi, M. Sawan and S. Bennis, "Parallel VLSI Implementation of A New Simplified Architecture of Kalman Filter", Electrical and Computer Engineering, 1995. Canadian Conference, Vol 1, pp 117 – 119, 1995.
[20] Z. Salsic and C.R. Lee, "Scalar-based direct algorithm mapping FPLD implementation of a Kalman filter", Aerospace and Elec-tronic Systems, IEEE Transactions on, Volume: 36 Issue: 3,pp 879-888, 2000.
[21] Volnei A. Pedroni, Circuit Design with VHDL, 1st ED, Massachusetts Institute of Technology, 2004.
[22] D. Lawrie and P., Fleming, "Fine-grain parallel processing implementations of Kalman filter algorithms", Control '91., In-tenational Conference , Vol 2, pp 867 – 870, 1991.
[23] Wadoo, S., & Kachroo, P. "Autonomous Underwater Vehicles: Modeling, Control Design and Simulation", Califonia: CRC p.112-134, 2010.
[24] H. Singh, J. Catipovic, R. Eastwood, L. Freitag, H. Henriksen, F. Hover, D. Yoerger, J. Bellingham and B. Moran, "An Integrated Approach to Multiple AUV Communication, Navigation and Docking," Proceedings of the MTS/IEEE Oceans'96 Conference, Ft. Lauderdale, FL, USA, 1996.
[25] A. Singhal, Issues in Autonomous Mobile Robot Navigation (1997).

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.