Signal Processing through Field Programmable Gate Arrays: Prospects and Challenges

Shubhajit Roy Chowdhury*

IC Design and Fabrication Centre, Department of Electronic and Telecommunication Engineering, Jadavpur University, Kolkata-700032, India

Abstract: The paper focuses on the use of field programmable gate arrays (FPGA) for signal processing applications. By allowing designers to create circuit architectures developed for the specific applications, high levels of performance can be achieved using FPGA for many digital signal processing (DSP) applications providing considerable improvements over conventional microprocessor and dedicated DSP processor solutions. A key reason is that an FPGA can side step the classic Von Neumann architecture's instruction—fetch, load/store bottleneck—found in most DSP. The paper highlights the flexibility offered by FPGA in realizing signal processing architectures and algorithms. The possibility of realizing low power signal processors on FPGA by functional transformation approach has also been discussed. The bottlenecks faced in the state of the art technologies have also been explained.

Keywords: Field programmable gate array, signal processing, low power.

1. INTRODUCTION

Signal processing algorithms have been used to transform or manipulate analog or digital signals for a long time. One of the most frequent applications is obviously filtering the signal. Digital signal processing has found many applications, ranging from data communications, speech, audio or biomedical signal processing, to instrumentation and robotics.

Digital signal processing (DSP) has developed over the past decade and has almost replaced analog signal processing (ASP) systems in many applications. DSP systems enjoy several advantages over ASP systems such as insensitivity to change in temperature, aging or component tolerance [1]. Originally analog chips yielded smaller die sizes, but with the advent of VLSI design in the deep submicron regime, digital chips can be realized on a smaller area with denser integration. This yields compact low power and low cost designs.

Signal processing applications are typically computationally intensive and heavily rely on the efficient implementation of such digital signal-processing (DSP) algorithms as filtering, transforms and modulation. In past systems, conventional digital signal processors were used to perform many of these algorithms. Programmable DSP introduced in the late 70's incorporated a multiply-accumulate operation in only one clock cycle which was a dramatic improvement over the Von-Neumann microprocessor based systems [2].

*Address correspondence to this author at the IC Design and Fabrication Centre, Department of Electronic and Telecommunication Engineering, Jadavpur University, Kolkata-700032, India;
E-mail: shubhajit@juiccentre.res.in

The advent of field programmable gate arrays (FPGA) has revolutionized the field of digital signal processing over the past decade. By allowing designers to create circuit architectures developed for the specific applications, high levels of performance can be achieved for many DSP applications providing considerable improvements over conventional microprocessor and dedicated DSP processor solutions. However, field-programmable gate arrays (FPGA) deliver an order of magnitude higher performance than traditional DSP. A key reason is that an FPGA can side step the classic Von Neumann architecture's instruction—fetch, load/store bottleneck—found in most DSP. Modern FPGA families provide DSP arithmetic support with fast carry chains which are used to implement multiply accumulates at high speed, with low overhead and low costs [3]. Another reason is the FPGA has lower power consumption [4, 5].

2. OVERVIEW OF FPGA

FPGA is a member of a class of devices called field programmable logic (FPL). FPLs are defined as programmable devices containing repeated fields of small logic blocks, called configurable logic blocks (CLB). A typical FPGA consists of three major types of elements viz. configurable logic block (CLB), programmable interconnects and I/O blocks. Fig. (**1**) shows the basic architecture of FPGA that incorporates these three elements.

The CLB can usually form the function of typical logic gates but it is still small compared to the typical combinational logic block found in a large design. The programmable interconnects are made between the logic elements. These interconnects may be logically organized into channels or other units. FPGA typically offer several types of interconnect depending on the distance between the combinational logic blocks that are to be connected; clock signals are also provided with their own interconnection

1877-6124/10 2010 Bentham Open

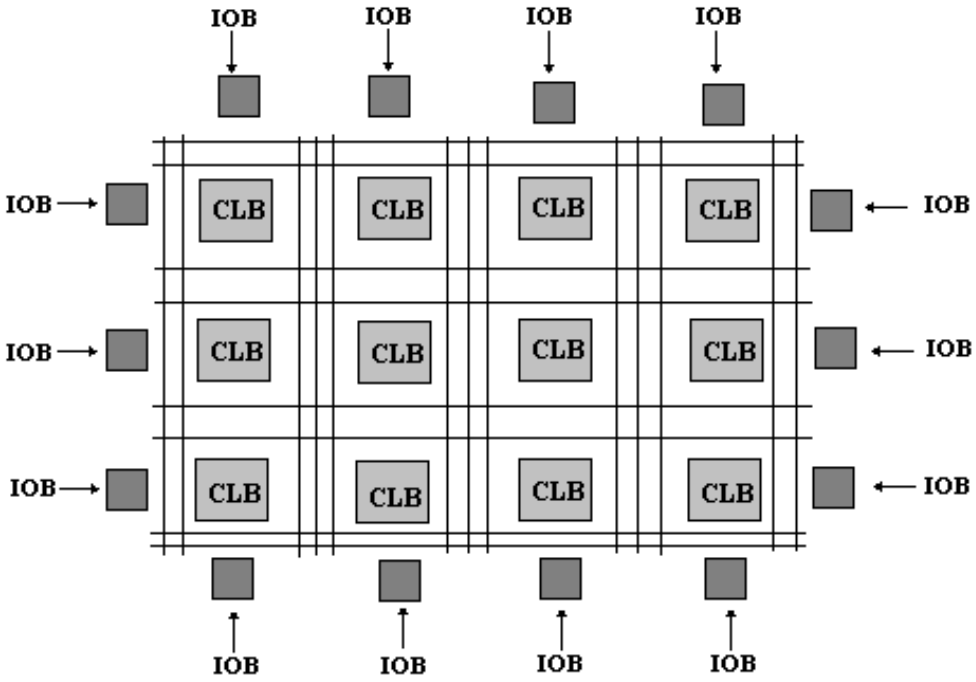

Fig. (1). Basic architecture of FPGA.

networks. I/O pins may be referred to as I/O blocks (IOB). They are generally programmable to be inputs or outputs and often provide other features such as low power or high speed connections. Multiple I/O pads may fit into the height of one row or the width of one column. An application circuit must be mapped into an FPGA with adequate resources.

A typical FPGA configurable logic block consists of a 4-input lookup table (LUT), and a flip-flop, as shown in Fig. (**2**) below:

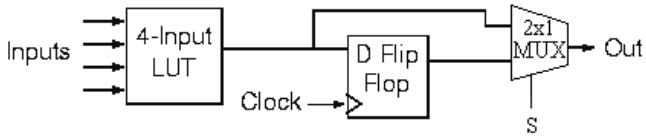

Fig. (2). Structure of a Configurable Logic block with multiplexed registered and unregistered output.

The look up table stores the truth table of the Boolean function to be implemented. Since the LUT is a 4 input LUT, hence, up to 4 variable Boolean functions can be implemented using LUT. By storing the truth tables of Boolean function on the LUT, the CLBs are configured for specific logic functions. There is only one output, which can be either the registered or the unregistered LUT output. The registered and unregistered LUT output is connected to the final output through a 2X1 multiplexer. The logic block has four inputs for the LUT and a clock input. Since clock signals (and often other high-fanout signals) are normally routed *via* special-purpose dedicated routing networks in commercial FPGAs, they are accounted for separately from other signals.

3. FLEXIBILITY OFFERED BY FPGA IN SIGNAL PROCESSING APPLICATIONS

Fitting multiple DSP functions into a single FPGA has many integration challenges, but also offers significant advantages to the designer in performance and flexibility. The primary reasons for integrating DSP functions into a single FPGA are system-level reductions in size, weight and power. For example, eliminating the transfer pathways between separate FPGA and DSP significantly reduces power consumption and, therefore, heat. This, in turn, reduces the system-cooling burden of the design. Recent releases of design and place-and-route software have advanced power-awareness features that significantly reduce dynamic power use of the FPGA. These options can be important to the designer; the benchmark of device logic density among competitive FPGA providers is beginning to give way to functionality-per-watt metrics, due to the sensitivity of power and cooling requirements in emerging systems.

Performance is also a key driver as FPGA based signal processing has become more reliable and faster than traditional processing technologies [6]. We have reported in literature how FPGA based pipelined and pipelined parallel hybrid architectures lead to computation with an increased throughput [7, 8]. In applications where performance is the driving parameter, efficiency can be sacrificed for application speed, where a memory-intensive, massively parallel floating-point mathematical operation is desired. Alternatively, highly iterative DSP calculations can be implemented for applications where moderate performance is allowable, but where logic-element usage is limited [9]. This logically leads to the advantage of flexibility. The designer has the flexibility to decide between high-speed performance and the number of logic elements in every DSP operation, whereas calculation bandwidths and iterations would be more difficult and costly to modify in a dedicated DSP device. In addition, consolidating DSP functions within an FPGA allows for post-design system changes in the signal-processing architecture, whereas using separate DSP locks the designer into a fixed set of chip interfaces once the

board is designed. FPGA designers can alternately switch between 9-bit, 18-bit or 36-bit or 18-bit complex math functions without changing the system hardware. Additional flexibility can be designed into the system when the designer uses fast-embedded processors for the execution or routing of complex floating-point operations.

4. PROSPECTS IN ALGORITHMIC MATHEMATICAL FUNCTIONS

Typical algorithmic mathematical functions in signal processing systems include recursive least-square and square-root operations [10]. Many designers have implemented these functions in C-based processors (in fixed-decimal and floating-point operations), or with proprietary FPGA VHDL operations. The current FPGA devices include embedded processor and logic-cell resources to efficiently implement these processes; future generations will also have these capabilities [11]. Additionally, IP cores and reference designs are becoming available to transition anywhere from dozens to hundreds of these operations into a single FPGA. Tools are available to translate processor-based algorithms from C code to hardware languages, such as very high-level descriptive language (VHDL). These tools can be used to optimize certain logic functions from a standard main processor into an FPGA co-processor operating in parallel with the main processor, or to move entire operations from the main processor to the FPGA hardware.

Matrix inversion is an important element of adaptive-array designs and standard spatial-transceiver-array processing (STAP) [12]. These operations are commonly performed in fixed hardware elements, though efficiently implemented embedded processing has been demonstrated in some radar/sonar development programs. The logic-element size and potential parallelism of a matrix inversion engine depends on the size of the array used in the radar system. As the size of the array is increased, so does the number of floating-point multiplications required by the system. Therefore, in larger arrays, there are more trade-off options between the speed of the system and the number of logic elements required by the system (both of which increase as the parallelization of the architecture increases). Implementing this function using a combination of a DSP and a group of internal memory blocks is the most likely design path for radar-system designers. As these operations are often tailored to the adaptive-array algorithms of the radar system, they are likely to be custom designed in VHDL. However, reference designs that are optimized for the place-and-route capabilities of an FPGA device can be offered or designed-to-order from the FPGA manufacturer, if required for the radar or sonar system.

5. IMPLEMENTING FAST FOURIER TRANSFORMS USING FPGA

Signal processing typically entails to time to frequency domain conversion and vice versa for the ease of analysis. Fast Fourier Transforms (FFT) and their inverse are effectively used for time to frequency domain conversion and vice versa. A review of literature reveals implementation of FFT with DSP and ASIC [13-18]. FFT has also been implemented with FPGA for 1D [19-21] and 2D [22, 23] transforms. Fig. (**3**) shows the performance comparison of different programmable devices in the perspective of signal processing.

Implementing fast-Fourier transforms (FFT) and their inverses in FPGA logic has advantages in prototyping and scalability, and offers design flexibility between a system's speed and the number of required logic elements. For example, massively parallel implementations can be designed and distributed among the logic elements of a single or multiple FPGAs. However, while these implementations can significantly reduce latency, they impose the penalty of a greater number of logic elements. However, the limitations on the number of logic elements in an FPGA can be done away using reconfigurable architectures. Recently, a reconfigurable signal processing chip with an embedded flash memory has been patented [24].

In fact, the primary flexibility advantage of an FPGA for FFT is the ability to select the optimal balance between these two parameters in the initial design. This is fortunate, because the implementation of large or complex FFT should be the primary factor in any design, and the advantages of an FFT implementation in an FPGA are apparent. Fig. (**4**) shows the FFT implementation using an FPGA. However, creating code or modifying existing code from previous designs can be cumbersome when testing and verifying code

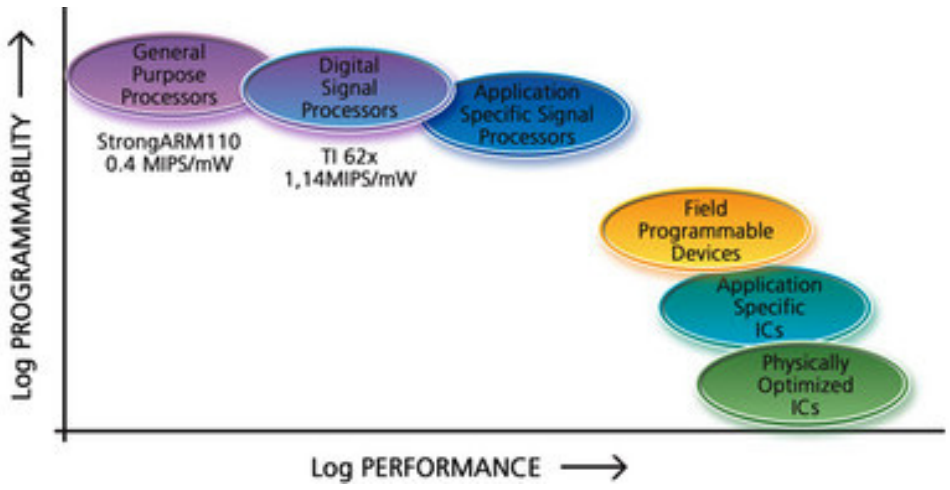

Fig. (3). Performance comparison of different programmable devices in the perspective of signal processing.

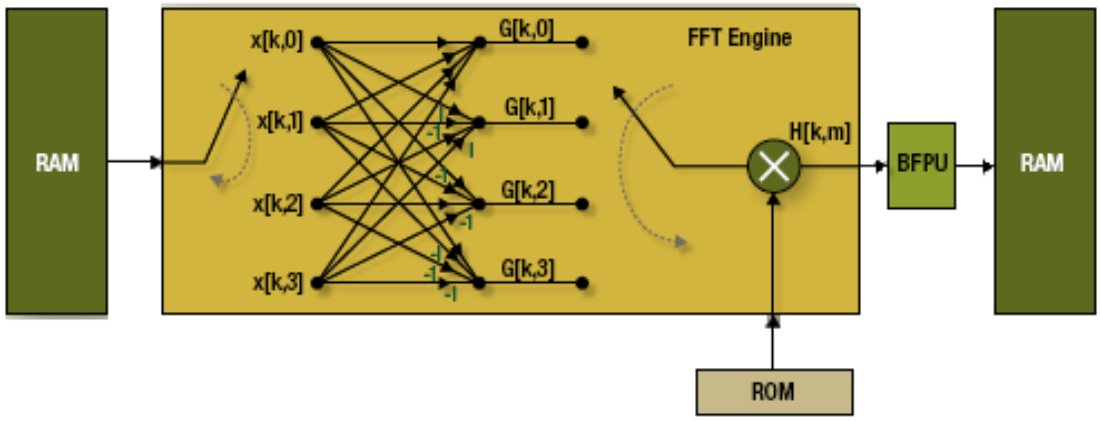

Fig. (4). FPGA based implementation of FFT.

units. Therefore, what is needed is a comprehensive suite of FFT design tools that allows a nearly infinitely scalable FFT design. These tools are available and they allow scripted logic distribution among multiple FPGA where necessary. They can also automatically generate numerical coefficients having floating-point accuracy.

6. LOW POWER SYSTEM IMPLEMENTATION USING FPGA

The average dynamic power dissipation of a CMOS logic circuit is given by:

$$P_{avg} = E_{switching} \times f_{CLK} \times C_{load} \times V_{DD}^2 \quad (1)$$

where P_{avg} is the average power dissipation, C_{load} is the load capacitance, f_{CLK} is the clock frequency, V_{DD} is the supply voltage and $E_{switching}$ is the expected value of output switching per clock cycle. For a CLB in an FPGA, C_{load} is a function of the number of fan outs of the CLB. f_{CLK} refers to the clock frequency of the FPGA. $E_{switching}$ can be computed as

$$E_{switching} = 2p(1-p) \quad (2)$$

where p refers to the probability that the output of the CLB is 1, so that $p(1-p)$ is the probability of having a 1 to 0 transition and $p(1-p)$ is the probability of having a 0 to 1 transition and the two transitions being mutually exclusive. The probability p can be computed from the truth table realized in the CLB. Specifically, p is equal to the sum of the probabilities of the input combinations which produce a 1 in the output of the truth table of the Boolean function. Extensive studies have been carried out in technology mapping algorithms to reduce the power dissipation by minimizing the number of CLB [25] and minimizing the length of critical path [26, 27]. One transformational approach aims at minimizing the number of CLB as a starting objective and then applying functional transformation to the mapping solution to reduce the power consumption without increasing the number of CLB [4]. Formally, let the output of a CLB F_0 is a Boolean function:

$$\varphi(x_1, x_2, ..., x_n) = F(G(x_1, x_2, ..., x_m), x_{m+1}, x_{m+2}, ..., x_n), \quad (3)$$

where $G(x_1, x_2, ..., x_m)$ is the output of CLB G_0. Obviously $F(G(x_1, x_2, ..., x_m), x_{m+1}, x_{m+2}, ..., x_n)$ is a Roth Karp decomposition [28, 29] of $\varphi(x_1, x_2, ..., x_n)$. The transformational approach attempts to find an alternative Roth Karp decomposition such that:

$$\varphi(x_1, x_2, ..., x_n) = F'(G'(x_1', x_2', ..., x_m'), x_{m+1}', x_{m+2}', ..., x_n') \quad (4)$$

where $x_1', x_2', ..., x_n'$ is some permutation of $x_1, x_2, ..., x_n$. Here, F' and G' are functions, possibly different from F and G. However, F' and G' map into the same CLBs F_0 and G_0 respectively, since the total number of inputs of F and G remains unchanged after transformation. If the switching density of $G'(x_1', x_2', ..., x_m')$ is less than that of $G(x_1, x_2, ..., x_m)$ then we achieve a reduction in power dissipation.

7. CHALLENGES IN SIGNAL PROCESSING USING FPGA

Despite such huge potential of FPGA in signal processing applications, it suffers from lots of limitations. Signal processing algorithms often require elementary functions like logarithm, exponential, trigonometric, etc. There are no such operators defined in the library for FPGA based system design. Many libraries of floating point operators for FPGA now exist typically offering the basic operators like +, -, ×, / and √ [30-34]. As FPGA floating point is clocked 10 times slower than the equivalent in contemporary processors, hence massive parallelism can allow these applications to be competitive to software equivalent [35-37]. We have reported in our work [38], how by using multi-core superscalar architectures on an FPGA, computational speed up have been achieved. Moreover even if it is possible to implement floating point algorithms on FPGA, the current state of the art algorithms do not scale well beyond single precision. It is therefore necessary to explore algorithms which work up to double precision which is standard in processors.

8. CONCLUSION

FPGA have been found to a highly robust architecture that can be tuned for lots of signal processing applications. By allowing designers to create circuit architectures developed for the specific applications, high levels of performance can be achieved using FPGA for many DSP applications providing considerable improvements over conventional microprocessor and dedicated DSP processor solutions. The paper highlights the flexibility offered by FPGA in realizing signal processing architectures and algorithms. The possibility of realizing low power signal processors on FPGA by functional transformation approach has also been discussed. However, there are lots of limitations in regard to floating point computation and implementing logarithmic, exponential and trigonometric functions using FPGA. The current state of the art algorithms for FPGA do not support multi-precision floating point operations. It is therefore necessary to explore algorithms which work up to double precision which is standard in processors.

REFERENCES

[1] Lapsley P, Bier J, Shoham A, Lee E. DSP processor fundamentals. New York, IEEE Press 1997.

[2] Shear D. EDN's DSP Benchmarks. Emerg Dep News 1988; 33: 126-148.

[3] Dipert B. EDN's first annual PLD directory. Emerg Dep News 2000; 45: 54-84.

[4] Chen CS, Hwang TT, Liu CL. *Low power FPGA design-a reengineering approach*. Proceedings of Design Automation Conference. Anaheim, California, USA 1997; 1: 656-661.

[5] Wolff FG, Knieser MJ, Weyer DJ, Papachristou CA. *High level low power FPGA design methodology*. Proceedings of IEEE National Aerospace and Electronics Conference. Dayton, OH, USA 2000; 1: 554-559.

[6] Kenny R. FPGA signal processing for Radar/Sonar applications. Defense Electron Magazine 2007; 14(6): 9-12.

[7] Chowdhury SR, Saha H. A high performance generalized fuzzy processor architecture and realization of its prototype on an FPGA. IEEE Micro 2008; 28(5): 38-52.

[8] Chowdhury SR, Chakrabarti D, Saha H. FPGA realization of a smart processing system for clinical diagnostic applications using pipelined datapath architectures. Microprocess Microsyst 2008; 32(2): 107-120.

[9] Peterson R, Hutchings B. An assessment of the suitability of FPGA based systems for use in digital signal processing. Lect Notes Comput Sci 1995; 975: 293-302.

[10] Detrey J, Dinechin FD. Parameterized floating-point logarithm and exponential functions for FPGAs. Microprocess Microsyst 2007; 31; 537-545.

[11] Lienhart G, Kugel A, Manner R. Using floating point arithmetic on FPGAs to accelerate scientific N-body simulations. FPGAs for Custom Computing Machines. Proceedings: 10th Annual IEEE Symposium, New York, IEEE Press 2002; 182-191.

[12] Roy S, Boudreault FR, Dupont L. An end-to-end prototyping framework for compliant wireless LAN transceivers with smart antennas. Comput Commun 2008; 31(8): 1551-1563.

[13] Baas B. SPIFFEE an energy efficient single chip 1024 point FFT processor. http://www.standford.edu/bbaas/spiffee1.html 1998.

[14] Sunada G, Jin J, Berzins M, Chen T. *COBRA: An 1.2 million transistor expandable column FFT chip*. Proceedings of International Conference on Computer Design: VLSI in Computers and Processors. Cambridge, MA, USA 1994; 546-550.

[15] Texas Memory Systems: TM-66 swiFFT Chip. http://www.texmemsys.com 1996.

[16] Sharp Microelectronics. BDSP9124 digital signal processor. http://www.butterflydsp.com 1997.

[17] Mellot J. Long instruction word computer. Ph.D thesis, University of Florida, Gainesville 1997.

[18] Lavoie P. A high speed CMOS implementation of the Winograd Fourier transform algorithm. IEEE Trans Signal Process 1996; 44(8): 2121-226.

[19] Panneerselvam G, Graumann P, Turner L. Implementation of fast Fourier transforms and discrete cosine transforms in FPGAs. Lect Notes Comput Sci 1996; 1142: 272-281.

[20] Altera Corporation. Fast Fourier transforms. Solution brief 12, Altera Corporation, 1997.

[21] Goslin G. *Using Xilinx FPGAs to design custom digital signal processing devices*. Proceedings of DSP. San Jose, CA USA 1995; 595-604.

[22] Shirazi N, Athanas P, Abbott A. Implementation of a 2D fast Fourier transform on an FPGA based custom computing machine. Lect Notes Comput Sci 1995; 975: 282-292.

[23] Dick C. Computing 2D DFTs using FPGAs. Lect Notes Comput Sci 1996; 1142: 96-105.

[24] Landry J, Plants W, Sexton R.: US7360068 (**2009**).

[25] Lai YT, Pendram M, Sastry AKAS. *BDD based decomposition of logic functions with application to FPGA synthesis*. Proceedings of Design Automation Conference. Dallas, Texas, USA 1993; 642-47.

[26] Cong J, Ding Y. *An optimal technology mapping algorithm for delay optimization in lookup table based FPGA designs*. Proceedings of IEEE International Conference on Computer Aided Design. San Jose, CA, USA 1992; 154-158.

[27] Dewan, H.: US 7430726 (**2008**).

[28] Roth JP, Karp RM. Minimization over Boolean graphs. IBM J Res Dev 1962; 227-238.

[29] Ratchev, B., Hwang, Y.Y., Pedersen, B.: US 7100141 (**2006**).

[30] Belanovic P, Leeser M. A library of parameterized floating point modules and their use in field programmable logic and applications. Lect Notes Comput Sci 2002; 2438: 657-666.

[31] Detrey J, De dinechin F. A tool for unbiased comparison between logarithm and floating point arithmetic. Technical Report RR2004-31, LIP, Ecole Normale Superieure de Lyon March 2004.

[32] Dido J, Geraudie N, Loiseau L, Payeur O, Savaria Y, Poirier D. A flexible floating point format for optimizing data-paths and operators in FPGA based DSPs. ACM SIGDA FGPA 2002; 50-55.

[33] Lee B, Burgess N. Parameterized floating point operators in FPGAs. Proceedings of 36th Asilomar Conference on Signals. Syst Comput 2002; 1064-1068.

[34] Shirazi N, Walters A, Athanas P. Quantitative analysis of floating point arithmetic on FPGA based custom computing machine. FPGAs for Custom Computing Machines. New York, IEEE Press 995; 155-162.

[35] De Lorimier M, De Hon A. Floating point sparse matrix vector multiply for FPGAs. ACM/SIGDA 13th International Symposium, Filed Programmable Gate Arrays. New York, ACM Press 2005; 75-85.

[36] Dou Y, Vassiliadis S, Kuzmanov GK, Gaydadjiev GN. 64 bit floating point FPGA matrix multiplication. ACM/SIGDA Field Programmable Gate Arrays, New York, ACM Press 2005; 86-95.

[37] Lienhart G, Kugel A, Manner R. Using floating point arithmetic on FPGAs to accelerate scientific N-body simulations. FPGAs for Custom Computing Machines, New York, IEEE Press 2002.

[38] Pani S, Saha H, Chowdhury SR. Performance enhancement in the associative processing of floating point numbers using multicore superscalar architecture. IEEE TENCON 2008; 1-6.

Received: September 9, 2009 Revised: October 1, 2009 Accepted: October 16, 2009