Implementation of 90 nm CMOS LFSR Test Vector Generator for FPGA Floating Point Arithmetic Unit

Fazal Noorbasha*, Ch.S.Venu Madhav, Jacqueline Hu, GSRK Harsha Vardhan Reddy, Rolla Sireesha, P.Joshi Ram Charan Tej

VLSI Systems Research Group (VSRG), Department of ECE, K L University,
Guntur, AP – 522 502 India
E-mail: fazalnoorbasha@kluniversity.in *

Abstract
In this paper, we present the LFSR Test Vector Generator for Floating Point Arithmetic Unit (FPAU); it improves the performance of multimedia processing with low hardware cost. Here, we focused on a decimal floating point unit which consists of adder and subtractor for high speed computing. It takes input as binary real numbers, converts them into single precision IEEE 754 format floating point numbers, performs operations and gives the output with high precision. We used Guard bit, Sticky bit and Round bit to get output with good precision. At last, we tested the unit by the values from LFSR. The FPAU was modeled and synthesized in Verilog HDL and LFSR was implemented 90nm CMOS technology.

Keywords - FPAU, IEEE 754. Sticky bit, Round bit, Guard Bit, LFSR.

I. Introduction

IEEE Standard 754 floating point is the most common representation today for real numbers on computers, including Intel-based PC's, Macintoshes, and most Unix platforms. In recent years computer applications have increased in their computational complexity. The industry-wide usage of performance benchmarks, such as SPECmarks, forces processor designers to pay particular attention to implementation of the floating point unit, or FPU. Special purpose applications, such as high performance graphics rendering systems, have placed further demands on processors. High speed floating point hardware is a requirement to meet these increasing demands. [1]

To test this implemented design we need test vector for this Built-In Self Test (BIST) is one of the best process. In this LFSR is used for test vector sequence generator. The most commonly used linear function of single bits is XOR. Thus, an LFSR is most often a shift register whose input bit is driven by the exclusive-or (XOR) of some bits of the overall shift register value. [2]

These entire circuit FPGA modules are designed in using Xilinx, gate level and CMOS Layout circuits are implemented in 90nm technology. And simulation results are tested in ModelSim. After fabrication of entire chip layout, entire chip is exposed to an ultra violet light source which emits photons with 4.9eV energy (250nm wavelength). The photons are absorbed by the electrons of the floating gate, excited and finally attracted by the control gate or substrate. Typical erasure time is 20ns. The supply voltage is 1.20V, I/O supply voltage is 2.50V and operating temperature is 27.00 OC.

II. IEEE 754 Floating Point System

The floating point describes a method of representing an approximation to real numbers in a way that can support a wide range of values. We have used single precision IEEE754 format to represent floating point numbers in this paper. In IEEE754 format, there are 3 parts named Sign, Exponent and Mantissa. [3] Single precision format consist of 1 Sign bit, 8 Exponent bits and 23 Mantissa bits. The mantissa is composed of the fraction and an implicit leading digit. The exponent base (2) is implicit and need not be stored. Single precision format is shown in Fig. 1.

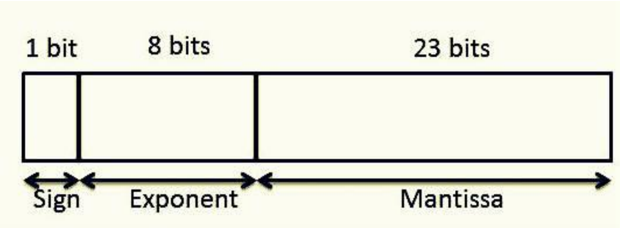

Fig. 1 Single precision format

The single precision can be represented as a forma: $(-1)^{S} \times 1.M \times B^{E}$

The '1' before mantissa is not represented anywhere in IEEE 754 format and it is called 'Hidden Bit'. The sign bit (S) when '0' denotes a positive number and '1' denotes a negative number. The exponent field needs to represent both positive and negative exponents. Exponent is an 8-bit number. It has a *bias* of 127 (single precision IEEE 754 format) is added to the exponent to get the stored exponent. The mantissa is also known as the significand. It represents the precision bits of the number and it has an implicit leading bit and the fraction bits.

III.IEEE 754 Standards

The standard defines:

- Arithmetic formats which are sets of binary and decimal floating-point numbers, which consists of finite numbers including subnormal number and signed zero, a special value called "not a number" (NaN) and infinity.
- Interchange formats which are bit strings (encodings) that are used to exchange a floating-point data in a compact and efficient form.
- Rounding rules which are the properties that should be satisfied while doing arithmetic operations and conversions of any numbers on arithmetic formats.
- Exception handling which indicates any exceptional conditions (like division by zero, underflow, overflow, etc.) occurred during the operations.

The standard defines the following four rounding rules:

- Round to the nearest even which rounds to the nearest value of the number with an even (zero) least significant bit.
- Round to the Zero means truncation of the number. The number is truncated to the require number of digits in rounding to the zero rule.
- Round towards positive infinity means rounding the number towards the positive infinity and is also called rounding up or ceiling.

- Round towards negative infinity means rounding number towards the negative infinity and is also called rounding down or floor or truncation.

IV. Implementation of Floating Point Adder and Subtractor

In the design of Floating point adder and subtractor, the numbers which are given in Binary format are first converted into IEEE 754 single precision format and then different operations are performed on these numbers to get the result. The output obtained may contain errors due to overflow of the binary digits during shifting and other operations. [4] To prevent these errors we used Guard bit, Round bit and Sticky bits. The FPAU design and operation flow and Algorithm for Binary to IEEE 754 format are shown in Fig. 2 and Fig. 3.

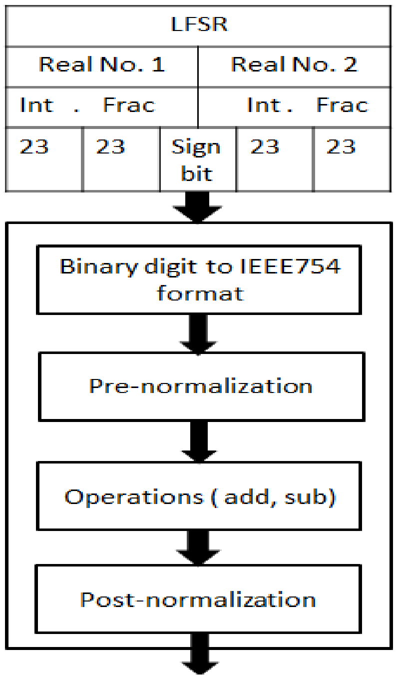

Fig. 2 FPAU design and operation flow

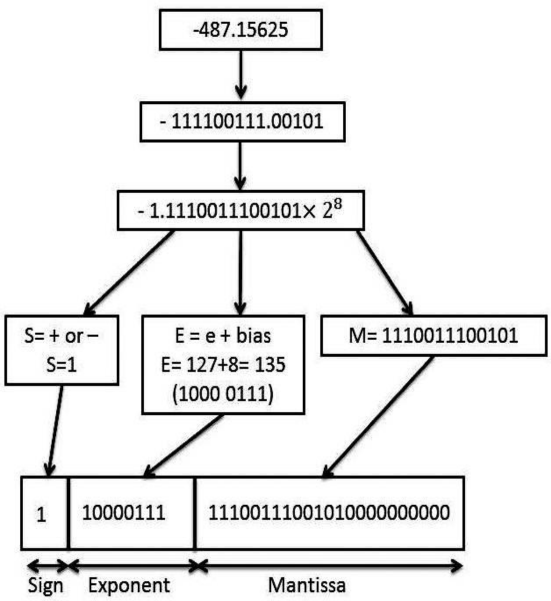

Fig. 3 Algorithm for Binary to IEEE 754 format

The algorithm for Binary to IEEE 754 format is used in our design is as follows:

A. Convert decimal number to binary

B. Move radix point to 1.xxx... * 2 ^ exp representation

C. Now, the steps are as follows:

i. Assign the sign bit. Positive → 0, negative → 1.

ii. Assign the mantissa. This is the fractional part of the value of the number.

→Ignore the leading "1". It is always 1 in this scientific notation, so there is no need to store it. The system will re-insert it later when the number is used.

→ Form the 23 MMM...MMM bits from the first 23 of the remaining "xxx..." bits above. If there are not 23 of them, fill out with trailing zeros.

iii. The exponent "exp" may be positive or negative. Float is stored with excess-127 notation. That is, you ADD 127 to your exponent and store as a pure (unsigned) binary. When the number is re-created for use later, 127 will be subtracted from the exponent. This method saves having to use 2's complement for negative exponents.

iv. Combine in recipe SEM format.

THE ALGORITHM FOR FLOATING POINT ADDITION OR SUBTRACTION IS AS FOLLOWS:

A. Find out the difference between exponents (ED) and which exponent is greater. Set the effective operand's exponent to larger exponent value.

B. Assign Guard bit, Round bit and Sticky bits to the mantissa of both numbers and the mantissa of lower exponent number is shifted to the right ED times. The sticky bit is obtained by OR operation of all the bits flowing out from the mantissa while shifting right. The hidden bit (1) is also recovered.

C. Sort the operands. The operand with larger mantissa is stored in one register and smaller mantissa in other register.

D. Determine Sign of the output by sign of both the operands.

E. Fix the sign for Not A Number and Zero results.

F. The operation of both addition and subtraction is performed on the numbers and the bit that overflows got saved in carry bit.

G. The rounding of the numbers is then performed on the resultant number and the output is then displayed.

This total algorithm for floating point addition /subtraction is shown in Fig. 4.

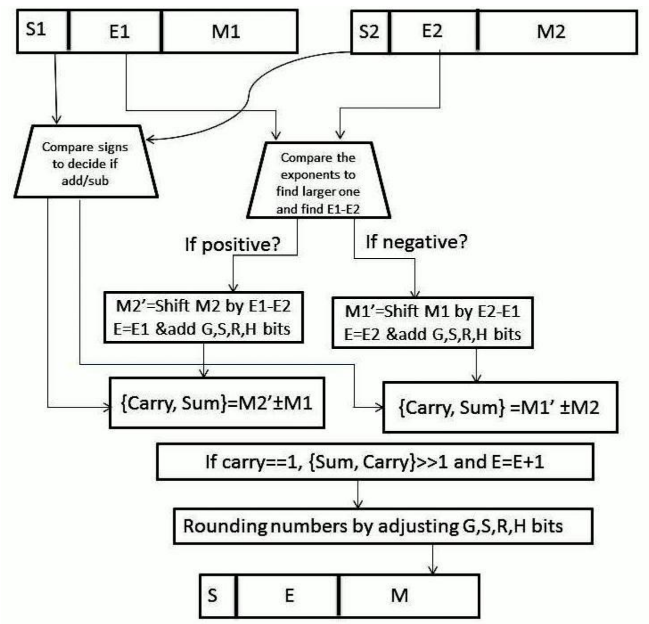

Fig. 4 Algorithm for floating point addition /subtraction

V. FPGA Synthesis Report of FPAU

Field Programmable Gate Arrays (FPGAs) are becoming a critical part of every system design. For FPAU design we have used Xilinx (Spartan-3) family. We have developed total hardware on Verilog HDL code. [5] Here Fig. 5 shows the RTL (FPGA) schematic view of FPAU. The average connection delay for Floating Point Arithmetic Unit FPGA design is 0.905ns, the maximum pin delay is 2.306ns and the average connection delay on the 10 worst NETS is 1.331ns. The HDL synthesis report is shown in Table – 1.

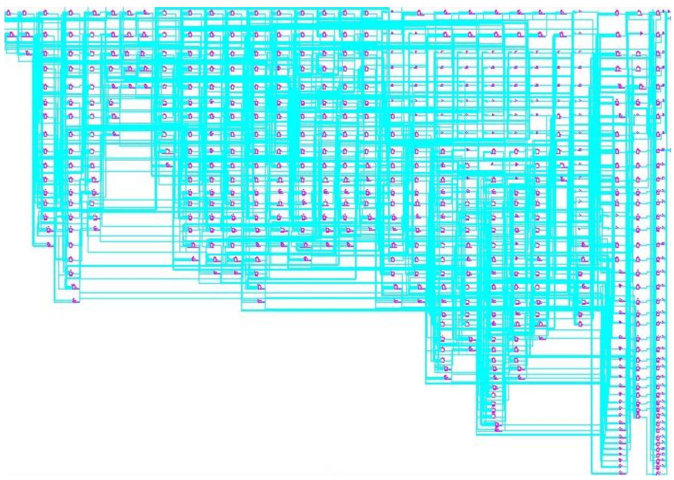

Fig. 5 RTL (FPGA) view of FPAU

Component	Utilization in FPGA
RAM (8x1-bit single-port distributed Read Only RAM)	1
Adders/Subtractors (28-bit, 8-bit)	4
Registers (Flip-flops)	33
Comparators (23-bit, 27-bit, 8-bit)	5
Multiplexers (2-to-1, 27-to-1)	22
Logic shifters (27-bit shifter logical right)	2

Table 1: HDL synthesis report

VI. 90nm Technology LFSR Test Sequence Generator

In testing process of VLSI technology, to get high fault coverage Built-In Self-Test (BIST) is employed. BIST solves the testing problems, generates better sequences covering the range of vectors to be tested. LFSR (linear feedback shift registers) is one of the BIST process, providing a simple means for generating non-sequential lists of sequences. LFSR is a shift register. When clock is given to this register it automatically shifts the signal through register from one flip-flop to other. A feedback mechanism can be formed by combining them in Ex-OR configuration. [6]
This method of XORing a bit with the feedback term is how CRCs (cyclic redundancy checks) are calculated.
In this work, we designed LFSR in order to generate a sequence automatically. These randomly generated sequences are given as inputs to the addition and subtraction unit. A series of flip-flops, connected with XOR gates with a clock, generate sequences randomly .For one complete cycle of operation we get 2^*n* -1 different outputs. So we get random sequences, which can be used as inputs and also for testing the addition-subtraction module instead of checking all the possible inputs. LFSR are efficient design for Test Pattern Generators &Output Response Analyzers (also used in CRC) They use Flip-flops plus a few XOR gates and are better than counters with higher clock frequency. [7-8]

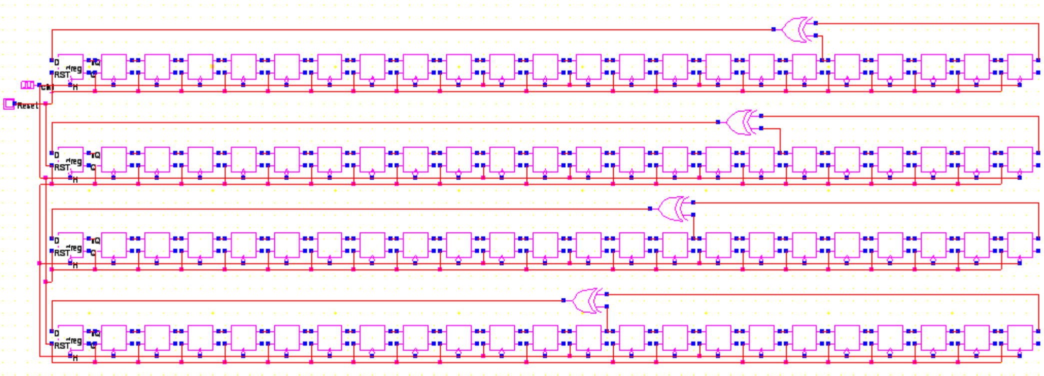

Fig. 6 Cell level LFSR design

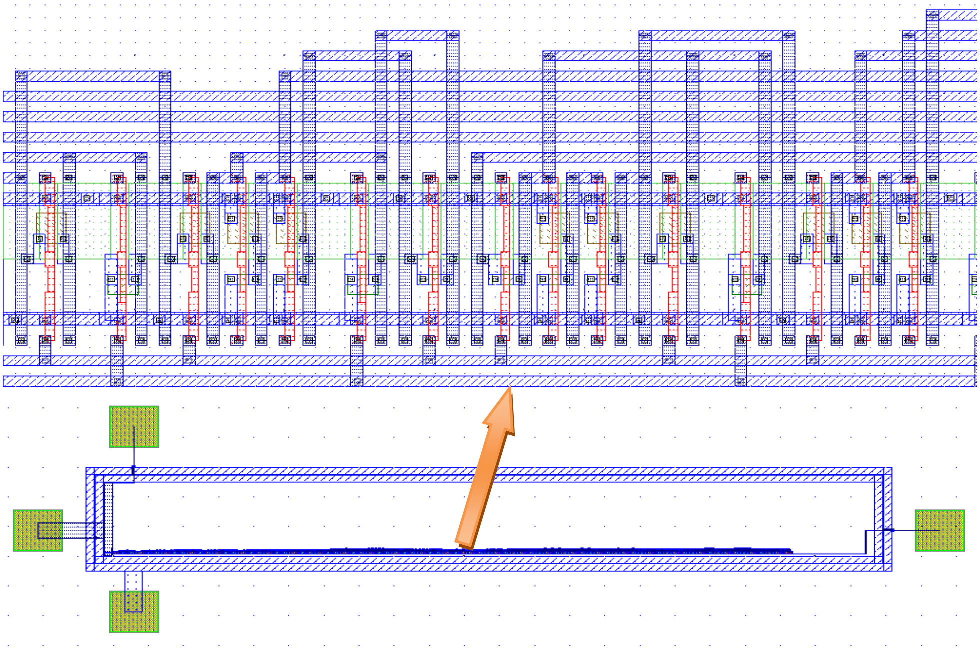

Fig. 7 90nm Technology LFSR Test Sequence Generator

LFSR CMOS circuit is fabricated in 90nm technology. The width of layout is 1160.8µm, height is 12.7µm and total surf is 14764.9µm2. The length (L)/ width (W) of Nmos is 0.120µm / 0.240µm and for Pmos are 0.12µm / 0.720µm. The rise delay is 0.002ns and fall delay is 0.001ns. The parasitic node properties are observed, capacitance is 0.62fF, resistance is 176 ohm and inductance is 0.001nH. We have observed the voltage and current parametric analysis with respect to time (ns). The IddMax is 1.792mA, IddAvr is 0.089mA.

VII. Sumulation Results

Integer Inputs:
OpA → 2815.55 →101011111111.10001100110011001100110 (numinta.numfraca)
OpB → 0185.725 →10111001.10111001100110011001101 (numintb.numfracb)
IEEE 754 Outputs:
OpA → 1_10001010_01011111111100011001100 (S_E_M)
OpB → 1_10000110_01110011011100110011001 (S_E_M)

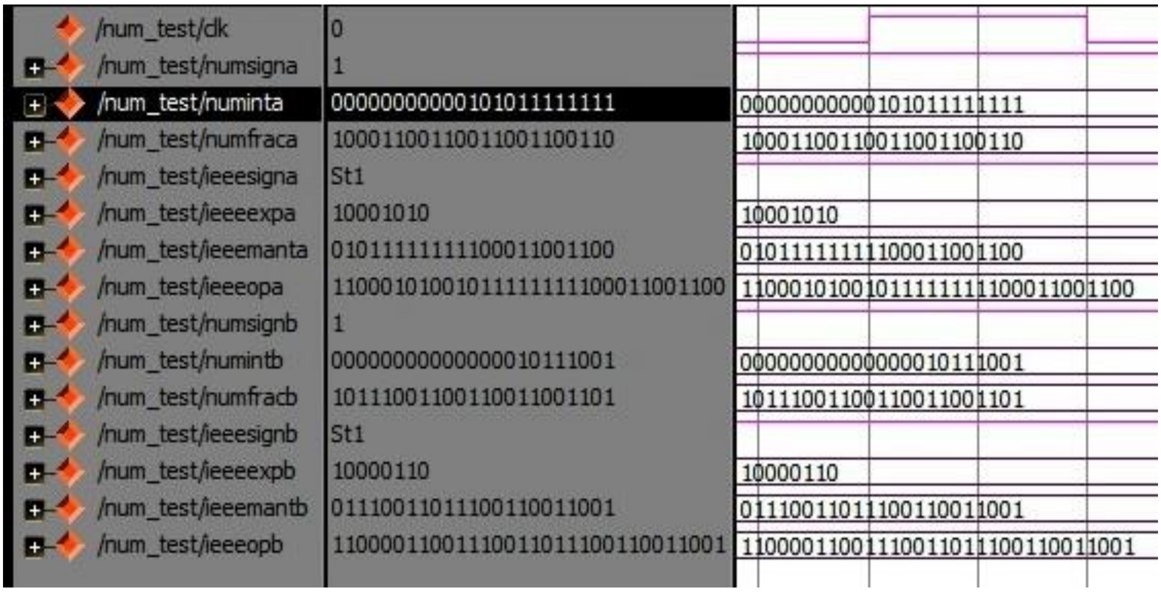

Fig. 8 Conversation of Integer number into IEEE 754 format

Arithmetic Addition Output:
OpA + OpB → Op → 1_10001010_01110111001010001100101 (S_E_M)
→3001.274012 (Required output → 3001.275)

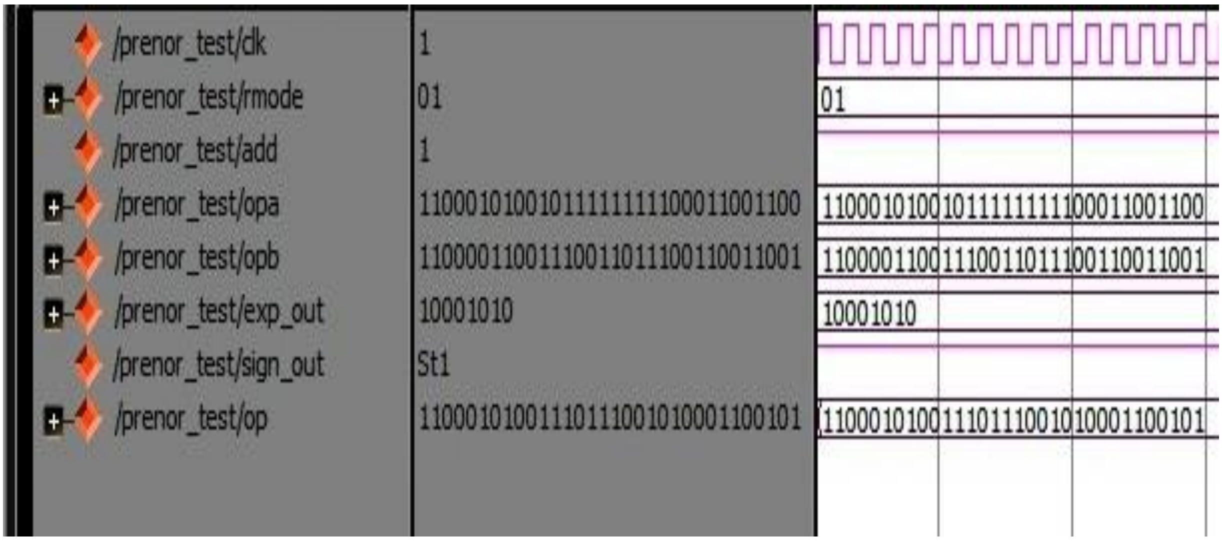

Fig. 9 Additon / subtraction results with the IEEE 754 numbers

VIII. Conclusion

This paper deals with development of a Floating Point Arithmetic Unit (FPAU) (i.e. adder and subtractor) in Verilog HDL with the help of ModelSim and synthesized with Xilinx tools. Simulation results of all the designed modules have been carried out for various inputs with the help of ModelSim tool. Both are available in single cycle and pipeline architectures and fully synthesizable with performance comparable to other available high speed implementations. The design is described as graphical schematics and Verilog code. And the floating point unit is tested successfully with the outputs from LFSR test pattern sequence data. This LFSR was implemented in 90nm CMOS technology.

Acknowledgements
Authors acknowledge support from Er. K satyanarayana, President, KL University. Also acknowledge to Dr. Habibullah Khan, Prof. & Head, Dept. of ECE, KLU, for providing the lab facilities.

References
[1] Stuart Franklin Oberman, "Design Issues In High Performance Floating Point Arithmetic Units" Technical Report: CSL-TR-96-711, December 1996.
[2] Fazal Noorbasha "VLSI Implementation of Timing and Control Unit (TCU) for Memory Processor ALU System", e-Journal of Science & Technology, Issue (2), Vol. 7, Year 2012, PP. 49-55.
[3] Seungchul Kim; Yongjoo Lee; Wookyeong Jeong; Yongsurk Lee, "Low cost floating point arithemetic unit design" 2002 IEEE Asia-Pacific Conference On ASIC Proceedings, Taipei, Taiwan – Aug. 6-8, 2002, PP. 217-220.
[4] Dr. Fazal Noorbasha, S. Dayasagar Chowdary, K. Hari Kishore and Shaik. Moulali, "Implementation of 180nm CMOS Linear Feedback Shift Register (LFSR) ASIC for Data Encryption and Decryption", CiiT International Journal of Programmable Device Circuits and Systems, Print: ISSN 0974 – 973X & Online: ISSN 0974 – 9624, DOI: PDCS042012017, Issue: April-2012, PP. 276-279.
[5] Dr. Fazal Noorbasha, Ch. Hemanth, A. Sivasairam and V. Vijaya Raju, "Implementation of 90nm Technology Multi Test Pattern Sequence LFSR for Fault Testing", CiiT International Journal of Programmable Device Circuits and Systems, Print: ISSN 0974 – 973X & Online: ISSN 0974 – 9624, DOI: PDCS042012018, Issue: April-2012, PP. 234-237.
[6] Samman, F.A.; Surapong, P.; Glesner, M., "Reconfigurable streaming processor core with interconnected floating-point arithmetic units for multicore adaptive signal processing systems", 6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip, June 20 – 22, 2011, Montpellier, France, PP. 1 – 6
[7] Bhasker J, "A Verilog HDL Primer", 3rd edition, Star Galaxy Publishing, 2005.
[8] Samir Palnitkar, "Verilog HDL: A Guide to Digital Design and Synthesis", 2nd edition, Prentice Hall Professional Publishing, 2003.