International Transaction of Electrical and Computer Engineers System, 2014, Vol. 2, No. 5, 144-148
Available online at http://pubs.sciepub.com/iteces/2/5/3

DOI:10.12691/iteces-2-5-3

Design of an Efficient Low Power 4-bit Arithmatic Logic Unit (ALU) Using VHDL

Giridhari Muduli, Bibhudatt Pradhan, Manas Ranjan Jena*, Snigdharani Nath

Departemt of ETC, SIET, Odisha
*Corresponding author: manas.synergy@gmail.com

Received June 03, 2014; Revised November 26, 2014; Accepted December 01, 2014

Abstract In this paper, we have designed an efficient low power 4-bit ALU using VHDL. Advancement in VLSI technology has allowed following Moore's law for doubling component density on a silicon chip after every three years. Though MOS transistors have been scaled down, increased interconnections have limited circuit density on a chip. Furthermore, the size of transistor is limited by hot-carrier phenomena and increase in electric field that lead to degradation of device performance and device lifetime. It has become essential to look into other methods of adding more functionality to a MOS transistor, such as, the multiple- input floating gate MOS transistor structure proposed by Shibata and Ohmi. An enhancement in the basic function of a transistor has, thus, allowed for designs to be implemented using fewer transistors and reduced interconnections. In published literature, many integrated circuits have been reported which are using multi-input floating gate MOSFETs in standard CMOS process. Thus using the advanced VLSI technology the proposed ALU design is more efficient.

Keywords: *CPU, GPU, ALU, RCA, CSA, CMOS*

Cite This Article: Giridhari Muduli, Bibhudatt Pradhan, Manas Ranjan Jena, and Snigdharani Nath, "Design of an Efficient Low Power 4-bit Arithmatic Logic Unit (ALU) Using VHDL." *International Transaction of Electrical and Computer Engineers System*, vol. 2, no. 5 (2014): 144-148. doi: 10.12691/iteces-2-5-3.

1. Introduction

In computing, an arithmetic logic unit (ALU) is a digital circuit that performs arithmetic and logical operations. The ALU is a fundamental building block of the central processing unit (CPU) of a computer, and even the simplest microprocessors contain one for purposes such as maintaining timers. The processors found inside modern CPUs and graphics processing units (GPUs) accommodate very powerful and very complex ALUs; a single component may contain a number of ALUs. Mathematician John von Neumann proposed the ALU concept in 1945, when he wrote a report on the foundations for a new computer called the EDVAC. Research into ALUs remains an important part of computer science, falling under Arithmetic and logic structures in the ACM Computing Classification System [1].

1.1. System Operations

A simple arithmetic logic unit does AND, OR, XOR, and addition. Most ALUs can perform the following operations: Integer arithmetic operations (addition, subtraction, and sometimes multiplication and division, though this is more expensive). Bitwise logic operations (AND, NOT, OR, XOR). Bit-shifting operations (shifting or rotating a word by a specified number of bits to the left or right, with or without sign extension). Shifts can be interpreted as multiplications by 2 and divisions by 2 [2]. A basic block diagram is shown in Figure 1.1 indicating 2-bit ALU operation.

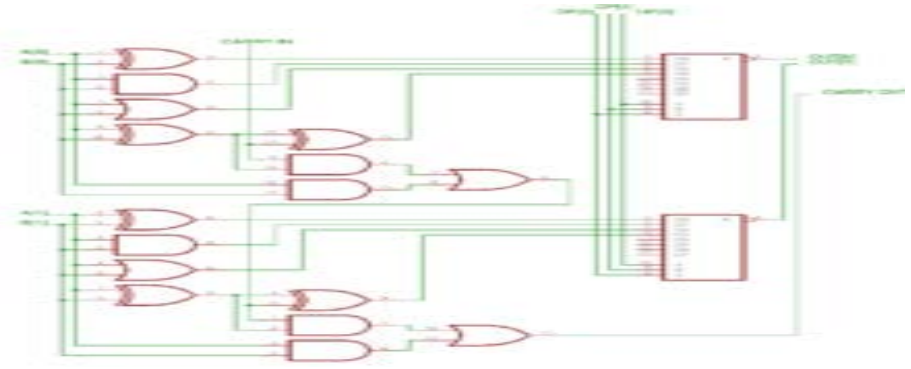

Figure 1.1. Block diagram shows 2-bit ALU Operation

1.2. System Inputs & Outputs

The inputs to the ALU are the data to be operated on (called operands) and a code from the control unit indicating which operation to perform. Its output is the result of the computation. In many designs the ALU also takes or generates as inputs or outputs a set of condition codes from or to a status register. These codes are used to indicate cases such as carry-in or carry-out, overflow, divide-by-zero, etc.

The arithmetic logic unit (ALU) is the core of a CPU in a computer. The adder cell is the elementary unit of an ALU. The constraints the adder has to satisfy are area, power and speed requirements. Some of the conventional types of adders are ripple-carry adder, carry- look ahead adder, carry-skip adder and Manchester carry chain adder. The delay in an adder is dominated by the carry chain. Carry chain analysis must consider transistor and wiring delays. Ripple carry adder is an n-bit adder built from full adders. Figure 1.2 shows a 4-bit ripple carry adder. Carry-look ahead adders first compute carry propagate and generate and then computes SUM and CARRY from them. It allows for carry to be computed in each bit [1,3].

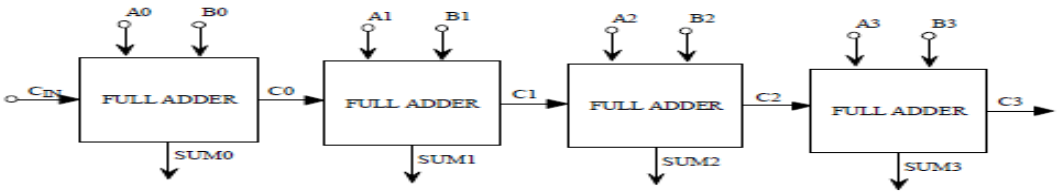

Figure 1.2. Block diagram of a 4-bit ripple carry adder (RCA)

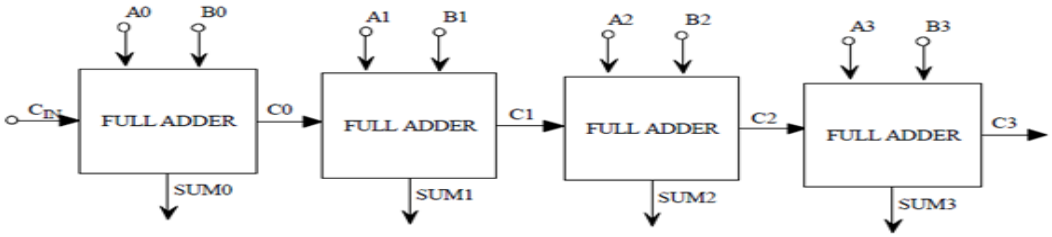

Figure 1.3. Block diagram of a 4-bit carry-look ahead adder (CLA)

Figure 1.3 shows a 4-bit carry-look ahead adder. Carry-look ahead unit requires complex wiring between adders and look ahead unit, as the values must be routed back to adder from look ahead unit. Layout becomes complex with multiple levels of look ahead. Figure 1.4 shows a 4-bit carry-skip adder and skip module used. The skip module determines whether it could just pass a carry in (CIN) the next four bits for addition or it has to wait until the carry out (C3) propagates through the last full adder in the design. In essence, the skip module can make the carry in (CIN) appear to skip through the four full adders [4]. The Manchester carry chain adder uses a pre charged carry chain with P and G signals. Propagate signal Pi is the XOR of input bits Ai and Bi and generate signal Gi is the NAND of input bits Ai and Bi. Propagate signal connects adjacent carry bits and Generate signal discharge the carry bit. Figure 1.4 shows a Manchester carry chain. When input bits are '0', Gi is HIGH and hence the carry out node is discharged. When one of the input bits is '1', then Pi is HIGH and carry out follows carry in. When both bits are '1', then both Gi and Pi are LOW; hence carry out node remains isolated from carry in and ground. As the node is pre-charged to a HIGH state the carry out remains HIGH. Each of the adder configurations may or may not require additional logic apart from full adder design. Table 2.1 shows approximately how many additional gates and transistors are required for each of the adder configurations. In terms of area efficiency ripple carry adder is preferred. Keeping in mind small layout area and less number of interconnections our ALU has been designed using ripple carry configuration. However, the delay time for worst case is more when compared to other adders [5]. A 16 -bit, 2.4 ns, 0.5 mm CMOS ALU is shown in Figure 1.5.

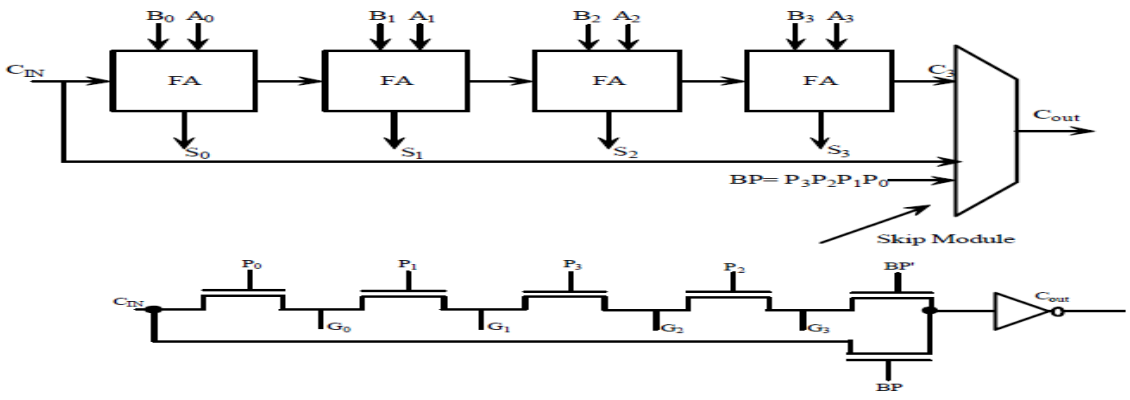

Figure 1.4. Block diagram of a 4-bit carry-skip adder (CSA) with skip module

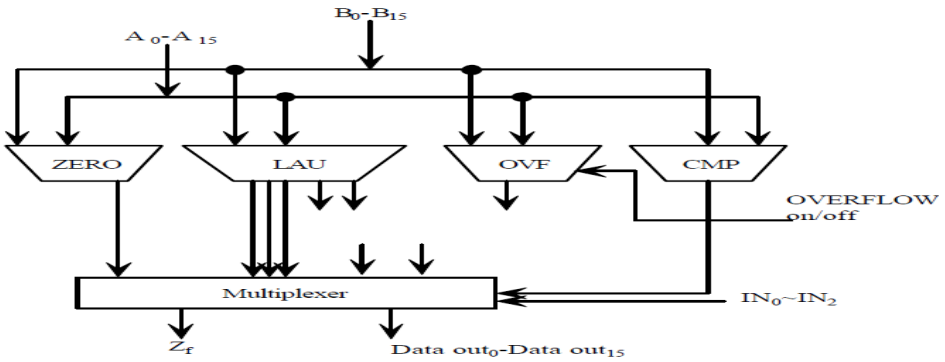

Figure 1.5. 16-bit, 2.4 ns, 0.5 mm CMOS arithmetic logic unit

1.3. System Block Diagram

The system block diagram of a 4-bit ALU is shown in the Figure 1.6. ALU is a combinational circuit that performs logic and arithmetic micro-operations on a pair of n-bit operands (ex. A [3:0] and B [3:0]). The operations performed by an ALU are controlled by a set of function-select inputs. In this design a 4-bit ALU with 3 function-select inputs: Mode M, Select S1 and S0 inputs. The mode input M selects between a Logic (M=0) and Arithmetic (M=1) operation.

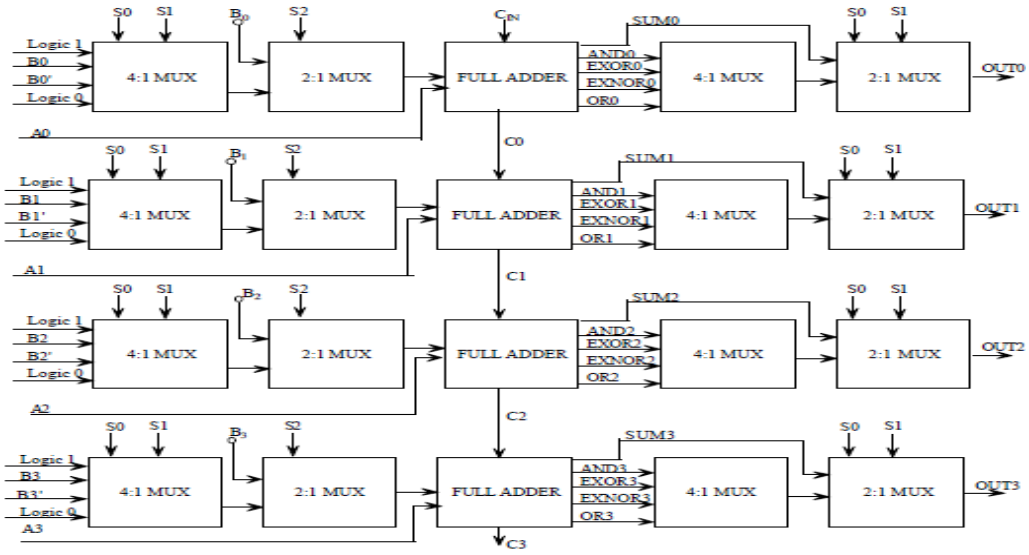

Figure 1.6. Block diagram of a 4-bit ALU

2. System Specification

The 4-bit ALU comprises of 4 to 1 and 2 to 1 multiplexers at the input and output sides and full adder with additional logic. The full adder is configured as ripple carry adder. S_0, S_1 and S_2 are the select signals that decide the operation being performed. The post layout simulated waveforms for the full adder showing SUM & CARRY bits. The truth table for the eight operations performed by the ALU is shown in Table 2.1 [6].

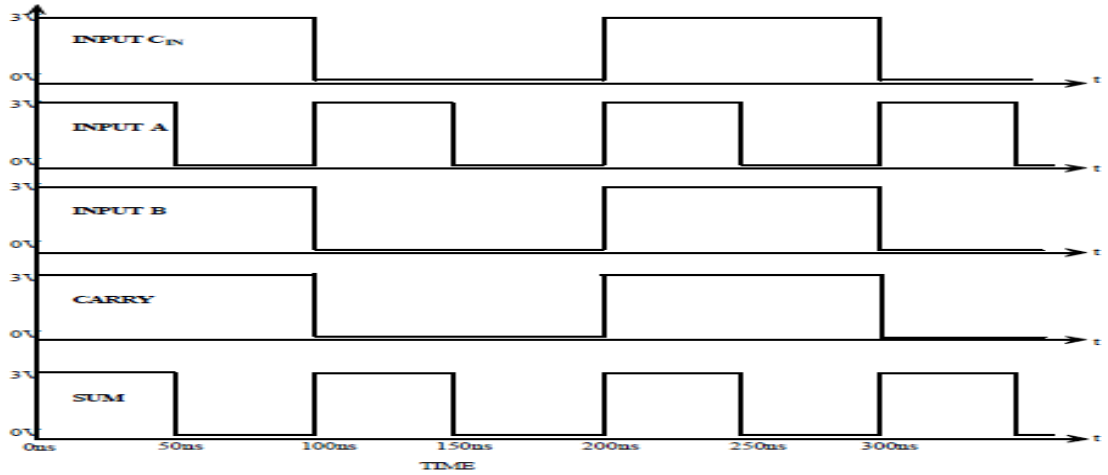

Figure 2.1. Post layout simulated waveforms for the full adder showing SUM and CARRY bits

Table 2.1 Truth table of a 4-bit ALU

S_2	S_1	S_0	Operation performed
0	0	0	INCREMENT
0	0	1	DECREMENT
0	1	0	ADDITION
0	1	1	SUBTRATION
1	0	0	AND
1	0	1	OR
1	1	0	EXOR
1	1	1	EXNOR

Signal S_2 is connected to logic '0' for arithmetic operations, and connected to logic '1' for logical operations. The multiplexer logic at the output side of each stage of the ALU gives the final output. For logical operations the output of each stage is independent of the other stages. In case of arithmetic operations, the carry ripples from LSB to MSB position. Therefore the output of each stage depends on the previous stage. For the layout of the 4-bit ALU, four stages of the full adder are cascaded in ripple carry adder configuration. The layout of the 4-bit ALU is shown in Figure 3.4. The entire layout was placed in the 1.5 m pad frame. Connections were made for inputs, outputs, supply voltages and ground pins on the pad frame [7].

3. Simulation Results & Analysis

The synthesis & simulation is performed by using softwares like Xilinx 11.1 and ModelSim SE 5.7 which is further used for coding, testing and simulation of VHDL programs. The layout of 4- bit ALU has been shown in Figure 3.4 using Mentor Graphic 2007. The chip layout before fabrication is shown in Figure 3.4. The design was fabricated in AMI 1.5 mm CMOS process. The 4-Bit ALU occupies approximately an area of 830 x 935 mm^2. SPICE simulations for the 4-bit ALU were done for post- layout extracted net lists. The RTL schematic block of 7-bit ALU is shown in Figure 3.1.

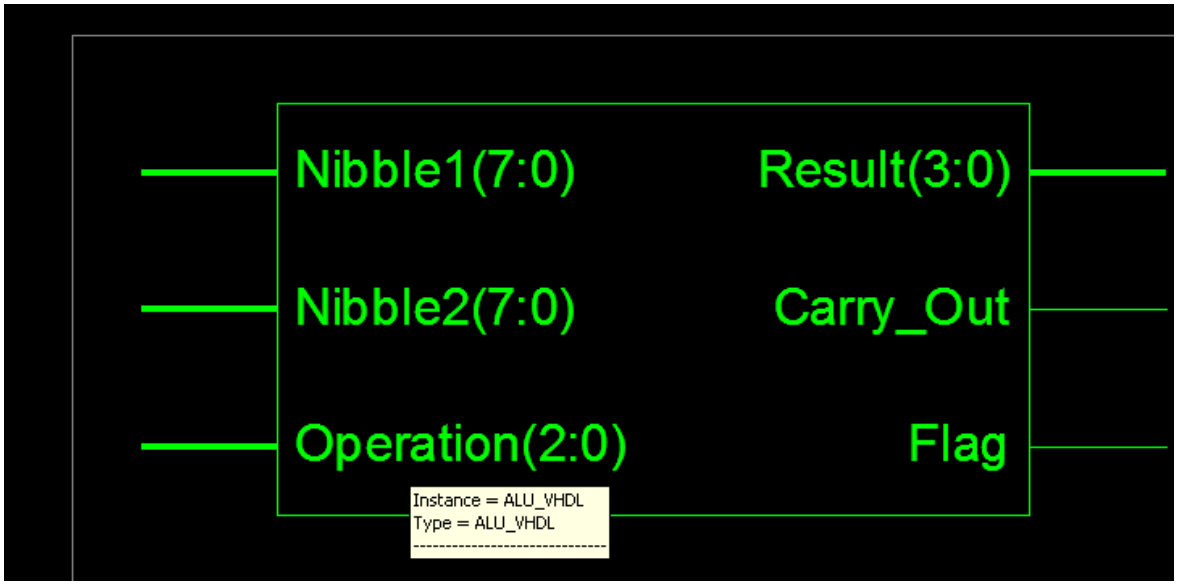

Figure 3.1. RTL Schematic block of 4-bit ALU

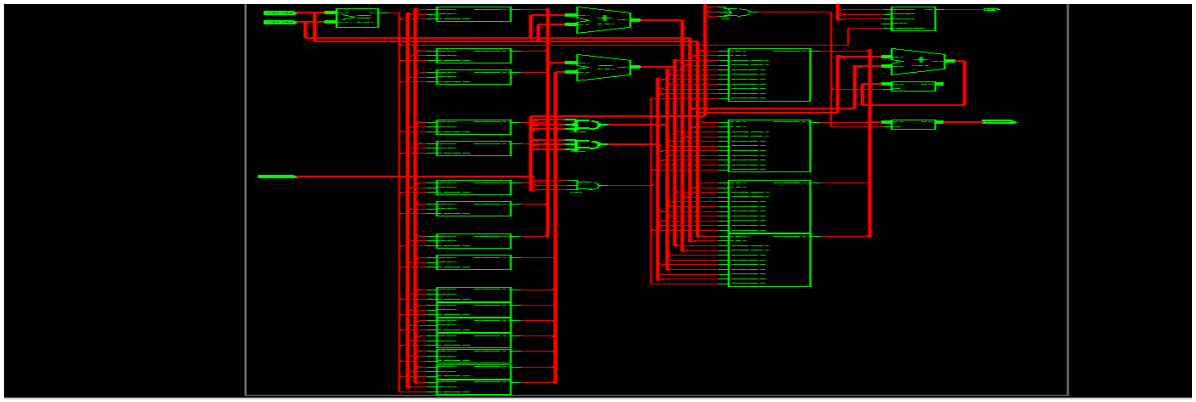

Figure 3.2. Synthesis output of ALU

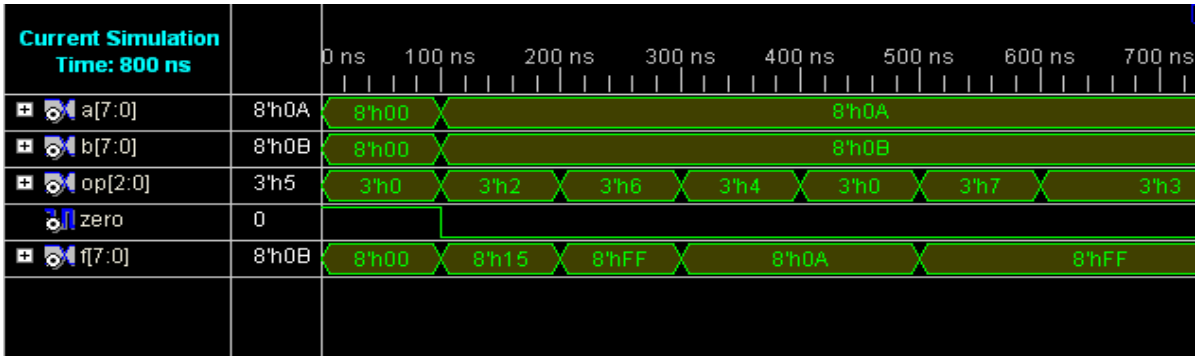

Figure3.3. Simulated Waveform

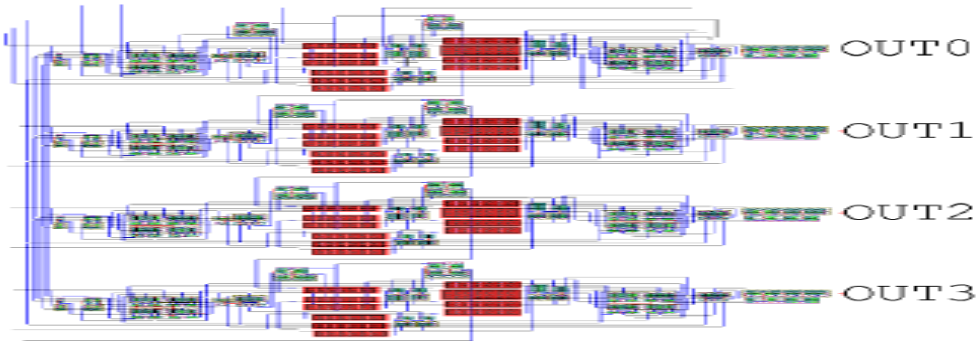

Figure 3.4. Layout of a 4-bit ALU

4. Conclusion

This paper presents design, optimization and implementation of 4-bit ALU. The increasing demand for low-power very large scale integration (VLSI) can be addressed at different design levels, such as the architectural, circuit, layout, and the process technology level. The simulation results shows improvement in delay of output signal & decrease the distortion of the waveforms at the output stages. Due to the major advantages the proposed design can be suitable in DSP applications.

References

[1] M. Farmwald., *"Design of High Performance Digital Arithmetic Units"*. PhD thesis, Stanford University, Aug. 1981.

[2] M. Morris Mano *"Digital Design"* (Pearson Education Asia. 3rd Ed, 2002).

[3] N. Burgess. *"The flagged prefix adder for dual additions"*. In Proc. SPIE ASPAAI-7, volume 3461, pages 567-575, San Diego, Jul. 1998.

[4] N. Quach and M. Flynn. *"Design and implementation of the snap floating-point adder"*. Technical Report CSL-TR-91-501, Stanford University, Dec. 1991.

[5] Y. Hagihara, S. Inui, F. Okamoto, M. Nishida, T. Nakamura, and H. Yamada. "Floating point data paths with online built in self speed test". IEEE J. Solid-State Circuits, 32 (3): 444-449, Mar. 1997.

[6] R.V.K. Pillai, D. Al-Khalili and A.J. Al-Khalili *"Power Implications of Additions in Floating Point DSP-an Architectural Perspective"* Concordia University, Montreal CANADA; Royal Military College, Kingston, The IEEE International Conference Pages: 581-586, 1999.

[7] E. Hokenek and R. Montoye. *"Leading-zero anticipator (lza) in the ibm risc system/6000 floating-point execution unit"*. IBM J. Res. Develop., 34 (1): 71-77, Jan. 1990.

[8] Javier D. Bruguera and Tomas Lang *"Leading one anticipation scheme for latency improvement in single data path floating point adders"*, Department of Electrical and Computer Engineering, Spain. Pages 125-133, 2005.

[9] S. Kang and Y. Leblebici *"CMOS Digital Integrated Circuit, Analysis and Design"* (Tata McGraw-Hill, 3rd Ed, 2003).

[10] Ricardo Gonzalez, Benjamin M. Gordon, and Mark A. Horowitz,*" Supply and Threshold Voltage Scaling for Low Power CMOS"*, IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 32, NO. 8, AUGUST 1997.