A CMOS Design of 64 Bit ALU Using Mentor Tools

Kishore Prabhala[1], Prof. Prabhandhakam Sangameswara Raju[2]
[1, 2] Dept of EEE
[1]Rayalaseema University, Senior Member IEEE
[2]SVU Engineering College, Sri Venkateswara Univeristy, Tirupati – 517 502, Chitoor Dist., AP

Abstract- *Data processing has been driven by an Arithmetic Logic Unit, ALU in any computer. A four bit ALU began by Intel Corporation in 1971 in the design of 4004 microprocessor, first of kind. The basis of any logic gate is a transistor which can raise to a logic 1 or Logic 0 in less than 100 pico seconds (Ps) which translates 10 Giga Hertz, GHz. 16 Giga bit memory chip has been in production since 2012 but 64 bit microprocessor have been produced only at end of 2015 simple because of arithmetic operation and software development of course with cost. The time delay of a 64 bit operation of addition or subtraction may take twenty or fifty times more than a clock cycle and multiplication would take more than hundred clock cycles. So the design with reduction in area to make faster processing has been pushing semiconductor technology with 180 nano meter (nm) to 90 nm to 65 nm or even 45 nm but the cost or power or time to delivery has to evaluated. The Electronic Design Automation (EDA) tools have been heavily used to make front end design with netlist with rapid strides along backend design of layout with tape out also. This paper show the usage of Mentor EDA tools in the front design of 64 bit ALU at 130 nm technology with simulation.*

Keywords- Arithmetic Logic Unit, Microprocessor, Giga Hertz, Electronic Design Automation and Mentor Tools.

I. INTRODUCTION

Data processing has been driven by an ALU in any computer which began by the 4004, first microprocessor by Intel with Metal Oxide Silicon (MOS) technology with 2300 transistors using 4 bit processing in 1971.But 8 bit processing in 1974 and 16 bit processing by 1978 created a glorious technology diffusion of Personal Computers led by IBM, Intel and Microsoft thought Apple, Atari, Tandy, and others made big strides but IBM model continued with faster speed and 32 bit processing by middle of 1990s. Millions of PC have led to huge number of applications as well as new devices called mobile later smart phone but ALU has been key aspect with memory in any advanced design of these devices.

The key aspect is the technology scaling down from 250 nm of CMOS gate to 180 nm to 130 nm with more transistors can be packed in same area but bigger chips can be made. So 64 and 128 bit data processing have taken place since 2015. Mass production of integrated circuits (IC) in semiconductor technology of fabrication is a complex, costly, and deals with yield. As propagation delay approached 100 Ps in 180 nm technology with 10 GB memory development led to 64 bit microprocessor based devices in 2015 with billions of devices. Operating system advancements and application of many types in millions made a 64 bit processing a key aspect of design. Area, Delay, Power and Reliable fabrication have become the key design constraints. Any ALU has to add or subtract or bit wise operations or increment or decrement.

II. ADDITION

Fundaments design of any ALU is based on addition and bit wise comparisons. So an adder plays a critical role in Very Large Scale Integration (VLSI) design of 64 bit addition which require over 5000 gates to be simulated. An adder can also convert to do subtraction. There are two adders.

The full adder performs addition of two inputs from A and B and another input called carry in, Cin. There are two outputs are Sum and Carry Out, Co. Since there are three inputs there will be eight combinations in binary and the input combinations would generate the output as truth table for a full adder shown in the table 1.

Table 1: Truth table for Full Adder

A	B	Cin	Sum	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

When one transpose these values into a Karnaugh map and with simplification of sum of products one get equation for Sum and Carry Out, Co. This is shown in figure 1.

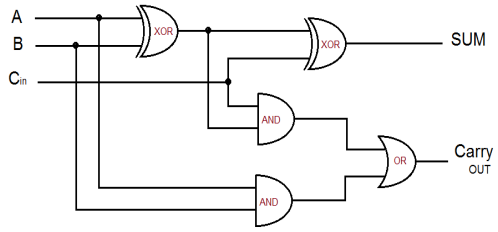

Figure 1: Full Adder

Sum = A'B'Cin + A'BCin' + AB'Cin' + ABCin = (A XOR B) XOR Cin

Carry Out = AB + ACin + BCin or AB + (A XOR B)Cin

III. SUBTRACTION

A subtraction can be performed using a modification to a Full Adder. A – B can be written as A + (2's complement of B). A 2's Complement needs a XOR with control input when 0 passes B and inverts bits when 1. This control input will be connected to Carry in when 1 it will added to A with complement of B. This is shown in figure 2.

A - B = A + (-B) = A + ~B + 1

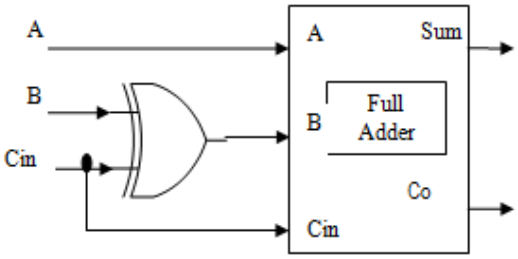

Figure 2: Adder as a Subtractor

IV. LOGICAL OPERATIONS

A bitwise operation of AND is good for IP address identification with 64 NANDs with INV. There is OR operation, XOR (Parity Check), XNOR (Equivalency) of 64 bits.

V. INCREMENTING AND DECREMENTING

The input ain can be incremented it can be incremented or decrement with 1's or 2's complement.

VI. FLAGS

A carry occurs when the final addition or subtraction is too big to fit into 64 bits which comes out of final carry out of 64 bit adders. A Zero is done with AND gate to signify the zero value at the output of the ALU. Parity of One is also checked.

VII. TOP LEVEL DESIGN IN MENTOR USING VERILOG

We have used Mentor tools to create the top design with five blocks. Each block has been coded and compiled to verify logically there are correct. There is a Logic_unit_64 for 64 bit wise operations with AND, OR, XOR, and XNOR using sel(1:0) as control. A block with "a_logic" to perform increment or decrement operation on input ain. Another block with "b_logic" to perform increment or decrement operation on input bin. Two bit control operators used with "sel" along anther operator "m" to select bit wise AND, OR, XOR, XNOR when 0 and else other when 1. When "Co" = 0, add will be done in the adder_64 block and subtraction when "Co" = 1.

A block by name "mux_2by1" used to select the output from the adder block vs logic block and control is "m". When m=0, output at "result(63:0)" will from the logical operations defined by sel(1:0), if m=1 then the output of adder_64 will be passed to "result(63:0)" .

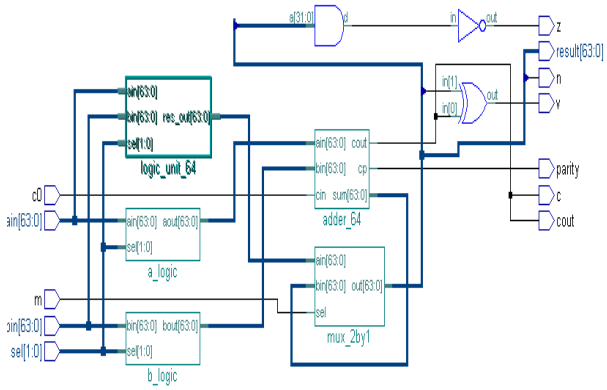

Figure 3: Top level 64 bit ALU

The following table shows the mode of operation of ALU with four control variables "sel0", "sel1", "m" and "Co".

Table 1: Operations for ALU

Mode M	Sel 1	Sel 0	Co	Function	Operations
0	0	0	x	A AND B	AND
0	0	1	x	A OR B	OR
0	1	0	x	A XOR B	XOR
0	1	1	x	A XNOR B	XNOR
1	0	0	0	A	Pass A
1	0	0	1	A + 1	Inc A
1	0	1	0	A + B	Add
1	0	1	1	A + B + 1	Add + Inc
1	1	0	0	A + B'	A + 1'sB
1	1	0	1	A + B' + 1	A + 2'sB
1	1	1	0	A' + B	1'sA + B
1	1	1	1	A' + B + 1	2'sA + B

The coding is done in verilog using mentol tool as shown in figure 4 with two inputs ain and bin defined to have 64 bits. Two inputs of control are m and co and other two are sel(1:0). Output of 64 bits is defined as result with flags as parity, cout, z, n, c, and v.

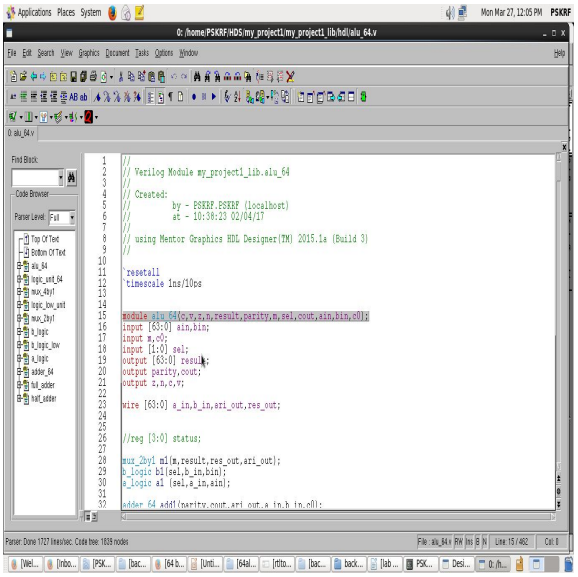

Figure 4: Verilog coding of Top level 64 bit ALU

7.1 Adder Block

This block has 64 inputs of ain and bin which are used to create add logic with XOR and passed to a AND gate. Then carry out is identified with a OR logic with one input from AND of ain and bin. The second input is from ANDing XOR of ain and bin with cin. Carry out of ain0 and bin0 will be carry in "cin" for ain1 and bin1. At bit 31, this will be carry flag. This is shown in figure 5.

There is a second level of XOR output is Sum. One input of this XOR is from the output of ain XOR bin with other input will be driven by Carry out from addition of each bit. The verilog coding for sum and carry is

```
// full adder: <str> (\/)
module full_adder(cout,sum,ain,bin,cin);
input ain,bin,cin;
output cout,sum;
wire w1,w2,w3;
half_adder ha1(w1,w2,ain,bin),
ha2(w3,sum,w2,cin);
or or1(cout,w1,w3);
endmodule

//half adder <str> (\/)
module half_adder(carry,sum,ain,bin);
input ain,bin;
output sum,carry;
xor xor11(sum,ain,bin);
and and11(carry,ain,bin);
endmodule
```

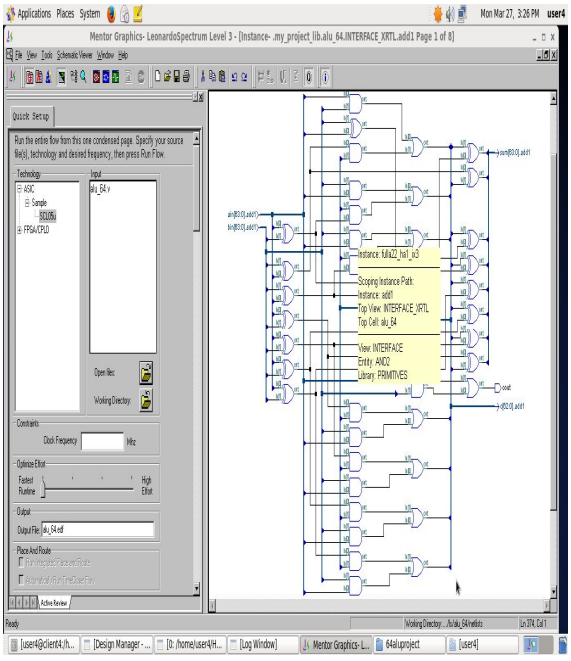

Figure 5: 64 bit Adder in ALU

For addition of 64 bits there are 64 full adder references in verilog from the above full adder definition as

```
//64 bit adder : <str> (\/)
module adder_64 (cp,cout,sum,ain,bin,cin);
input [63:0] ain,bin;
input cin;
output [63:0] sum;
output cp,cout;
wire [62:0] c;
full_adder fulla0(c[0],sum[0],ain[0],bin[0],cin),
```

fulla1(c[1],sum[1],ain[1],bin[1],c[0]),
fulla2(c[2],sum[2],ain[2],bin[2],c[1]),
this continues for 63rd bit from bit zero.

7.2 Logical Block

There are four logical implementations in this block. One is AND all 64 bit ain with bin. Second is OR all 64 bit ain with bin. Third is XOR all 64 bit ain with bin and fourth is XNOR. The verilog code that generates the logic is

```
module logic_low_unit (and_out,or_out,xor_out,xnor_out,ain,bin);
input ain,bin;
output and_out,or_out,xor_out,xnor_out;
wire w1,w2;
//wire and_out,xnor_out;
nand aand1(w1,ain,bin);
not nnot1(and_out,w1);

nor oor1(w2,ain,bin);
not nnot11(or_out,w2);

or oor2(xnor_out,and_out,w2);
not nnot2(xor_out,xnor_out);
endmodule

module logic_unit_64(res_out,sel,ain,bin);
input [63:0] ain,bin;
input [1:0]sel;
output [63:0] res_out;
wire[63:0] and_out,or_out,xor_out,xnor_out;
logic_low_unit
llu0(and_out[0],or_out[0],xor_out[0],xnor_out[0],ain[0],bin[0]
),
llu1(and_out[1],or_out[1],xor_out[1],xnor_out[1],ain[1],bin[1]
), continues for 63rd bit.
```

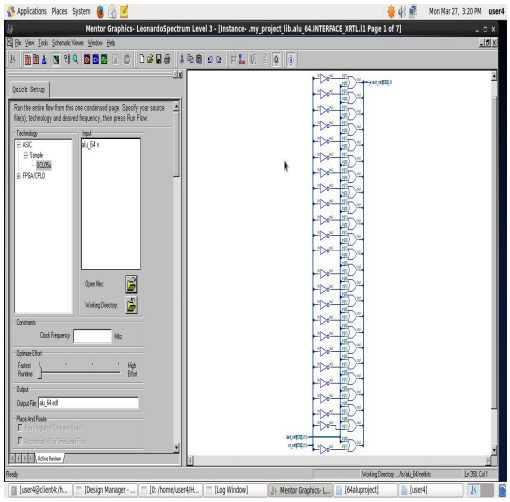

Figure 6: 64 bitwise Logic implementation

VIII. GATE COUNT

There are two 64 bit inputs, four control inputs (m, sel0, sel1, Co), one 64 bit output results and six output as flags, so total 202 ports as show in figure 7.

The total number gates And, Or, Inv, XOR, are 6125.

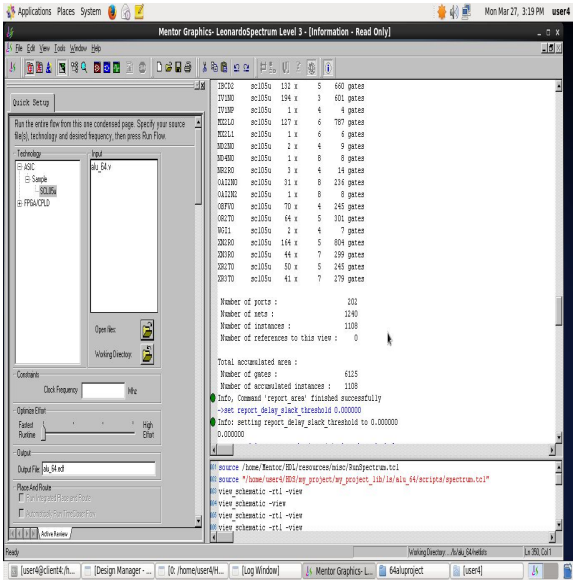

Figure 7: Gate for all the five blocks designed for 64 bit ALU

IX. SIMULATION

The first wave form shown is the 64 bit input “ain” and next is input “bin”. The third wave form is the control input “m” when it is 1 it select addition and subtraction as seen going 1 at 1375ns on x-axis. When m is ZERO logical bit operations are performed. Fourth wave form is carry in “co” which goes from low to high at 2500 ns. Next is two input “sel(1:0)” and also each input sel1 and sel0 is shown figure 8.

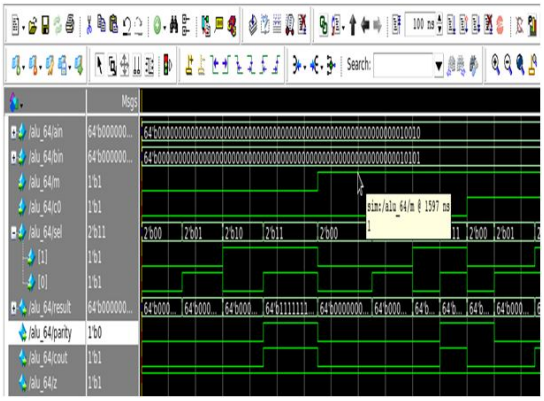

Figure 8: Results of Logical verification of 64 bit ALU

Eight waveform is the 64 bit output of "result" which goes to all Ones at 1000 ns along with the outputs parity and carryout too. By 1300 ns another pattern is checked and the flags change too as per the operation.

X. RESULTS

A propagation delay for adder from stage zero was identified and given in figure 9. A NOR gate has a worst case delay of 0.26 ns and a NAND gate has a worst delay of 0.32 ns. A logical OR output took 0.62ns, a logical AND took 0.97ns, a XOR took 1.79ns and a XNOR took 2.21ns at the end of the mux shown in the top level logic diagram.

A 64 bit addition has 39.36 ns delay from input "ain" and "bin" to output "result_out" for adding 64 bits at 130 nm technology library of Leonardo Spectrum Level 3 of Mentor tool. This is shown in figure 9 and figure 10 shows results for last adders.

Critical Path Report

Critical path #1, (unconstrained path)

NAME	GATE		ARRIVAL	
sel(0)/		0.00	0.00	dn
io_sel(0)/X	IBCD2	0.26	0.26	dn
b1_ix597/X	WGI1	0.38	0.65	up
b1_ix521/X	ND2N0	0.32	0.97	dn
b1_ix593/X	CBDC	0.82	1.79	dn
b1_ix515/X	MX2LO	0.43	2.21	up
add1_ix1/X	XR2TO	0.59	3.12	dn
add1_ix963/X	AO2IO	0.36	3.48	up
add1_ix21/X	OAI2NO	0.51	3.99	dn
add1_ix975/X	AO2IO	0.50	4.49	up
add1_ix45/X	OAI2NO	0.51	5.00	dn
add1_ix987/X	AO2IO	0.50	5.50	up
add1_ix67/X	OAI2NO	0.51	6.01	dn
add1_ix999/X	AO2IO	0.50	6.51	up
add1_ix93/X	OAI2NO	0.51	7.02	dn
add1_ix1011/X	AO2IO	0.50	7.52	up
add1_ix115/X	OAI2NO	0.51	8.03	dn
add1_ix1023/X	AO2IO	0.50	8.53	up
add1_ix139/X	OAI2NO	0.51	9.04	dn
add1_ix1035/X	AO2IO	0.50	9.54	up

add1_ix139/X	OAI2NO	0.51	9.04 dn	0.14
add1_ix1035/X	AO2IO	0.50	9.54 up	0.13
add1_ix161/X	OAI2NO	0.51	10.06 dn	0.14
add1_ix1047/X	AO2IO	0.50	10.56 up	0.13
add1_ix189/X	OAI2NO	0.51	11.07 dn	0.14
add1_ix1059/X	AO2IO	0.50	11.57 up	0.13
add1_ix211/X	OAI2NO	0.51	12.08 dn	0.14
add1_ix1071/X	AO2IO	0.50	12.58 up	0.13
add1_ix235/X	OAI2NO	0.51	13.09 dn	0.14
add1_ix1083/X	AO2IO	0.50	13.59 up	0.13
add1_ix257/X	OAI2NO	0.51	14.10 dn	0.14
add1_ix1095/X	AO2IO	0.50	14.60 up	0.13
add1_ix283/X	OAI2NO	0.51	15.11 dn	0.14
add1_ix1107/X	AO2IO	0.50	15.61 up	0.13
add1_ix305/X	OAI2NO	0.51	16.12 dn	0.14
add1_ix1119/X	AO2IO	0.50	16.62 up	0.13
add1_ix329/X	OAI2NO	0.51	17.13 dn	0.14
add1_ix1131/X	AO2IO	0.50	17.63 up	0.13
add1_ix351/X	OAI2NO	0.51	18.14 dn	0.14
add1_ix1143/X	AO2IO	0.50	18.64 up	0.13
add1_ix381/X	OAI2NO	0.51	19.15 dn	0.14
add1_ix1155/X	AO2IO	0.50	19.65 up	0.13
add1_ix403/X	OAI2NO	0.51	20.16 dn	0.14
add1_ix1167/X	AO2IO	0.50	20.66 up	0.13
add1_ix427/X	OAI2NO	0.51	21.17 dn	0.14
add1_ix1179/X	AO2IO	0.50	21.67 up	0.13
add1_ix449/X	OAI2NO	0.51	22.18 dn	0.14
add1_ix1191/X	AO2IO	0.50	22.68 up	0.13
add1_ix475/X	OAI2NO	0.51	23.19 dn	0.14
add1_ix1203/X	AO2IO	0.50	23.69 up	0.13

Figure 9: Propagation Values of Logical Comparisons and Adders of 64 bit ALU

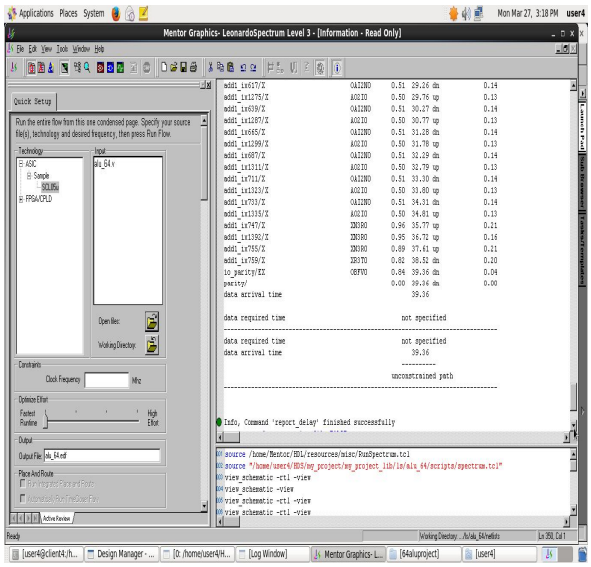

Figure 10: Propagation Values for the last stage Adders of 64 bit ALU

XI. CONCLUSIONS

This is a CMOS VLSI design of 64 bit ALU with addition, subtraction, bit wise operations of AND, OR, XOR and XNOR verified with mentor tools at 130 nm technology at room temperature. The efficiency design time increases for three or four fold with EDA tools and the blocks can be reused. Addition can be reduced with a carry look ahead adder and in future will do at the cold and hot temperatures as well as scaling down the technology too.

REFERENCES

[1] Mukesh P. Mahajan, P.G. Salunke, Y.M. Gaikwad, V.P. Jagtap, "Design And Simulation of 64 bit ALU", IJARECE Volume 4, Issue 4, January 2015.
[2] Kishore Prabhala, Haritha Dasari, and Thrinadh Komatipalli, "Performance Comparison of 64-Bit Adders", IJEDR 2018 | Volume 6, Issue 2 | ISSN: 2321-9939
[3] Pragati Nagdeote and Prof. Manisha Waje, "Design of 64-bit Arithmetic Logic Unit (ALU) Based on BSIM4 Model Using Tanner", IJSART - Volume 2 Issue 10-October-2016 ISSN [ONLINE]: 2395-1052
[4] Rajib Partha Bhattacharyya, Bijoy Kundu, Sovan Ghosh, Vinay Kumar, and Anup Dandapat, "Performance Analysis of a Low-Power High-Speed Hybrid 1-bit Full Adder Circuit" IEEE transactions on very large scale integration (VLSI) systems, vol. 23, no. 10, October 2015.
[5] Chetia, Kaushik Chandra Deva Sarma, Gaurab Baruah, "Behavioral Design and Synthesis of 64 Bit ALU using Xilinx ISE". IOSR Journal of Electronics and Communication Engineering (IOSR-JECE), Volume 7, Issue 4 (Sep. - Oct. 2013), PP 37-41, e-ISSN: 2278-2834,p- ISSN: 2278-8735.
[6] N. Ravindran, R. Mary Lourdes, "An Optimum VLSI, Design Of A 16-Bit ALU",978-1-4799-8966-9/15/$31.00 ©2015 IEEE
[7] Sakshi Samaiya and Anupreksha Jain, "A Review Article of ALU Unit Design based on FPGA", International Journal of Scientific Research & Engineering Trends, Volume 4, Issue 4, July-Aug-2018, ISSN (Online): 2395-566X.

About Authors

Kishore Prabhala is a research Scholar in EEE PhD, Rayalaseema University and also Senior Member IEEE. He published six papers in CMOS VSLI design in India after leaving USA in 1994 working at Motorola, MMI and National Semiconductor from 1981. Currently, he is the Principal, PLNM Degree College, Opposite Acharya Nagarjuna University Mens Hostel, Nagarjuna Nagar – 522 510, Guntur Dist., AP, India. He received a MSEE from Georgia Institute of Technology, GA, USA in 1989 and BSEE from Purdue University, W.Lafatette, IN, USA in 1981.

Prof. Prabhandhakam Sangameswara Raju is a Professor in Dept. of Electrical and Electronics Engineering, SVU Engineering College, Sri Venkateswara Univeristy, Tirupati – 517 502, Chitoor Dist., AP. He received M.Tech. and Ph.D. from SVU Engineering College. He has been teaching PG course for last 25 years and guided ove 52 projects. He published over 70 papers. Currently there are 8 students pursuing Ph.D.