International Journal of Engineering Innovation & Research
Volume 2, Issue 3, ISSN: 2277 – 5668

Design of LRSM Algorithm for Multiplications of Large Numbers

Mrs. Sudha P.
Lecturer, Department of ECE

***Abstract* – The design of high-speed, area-efficient and low power multiplier is essential for the VLSI implementation of DSP systems. In many applications, like digital filtering, the inputs are contaminated by noise and precise outputs are often not required. A new high precision serial multiplier with most significant digit first is presented. This method uses a borrow save adder to perform the reduction of large length partial products required by the multiplications of large numbers. The results are converted from Borrow-Save (BS) form to 2's complement representation by the on the fly conversion which let the conversion of the digit result as soon as it is obtained. the comparison between the residual and these constants *(-3/2*, -1/2, 1/2 and *3/2)* needed in the radix-2 on line multiplication, present problem in high precision computation.**

In the proposed method the operands are introduced digit by digit with Most Significant Digit First (MSDF) mode and results are obtained in the same manner with fixed time delay independently of the operand size. So this approach is advantageously used for the long multiplication computation. The results of the implementations of this multiplier for several operands sizes on Virtex-II. FPGA circuit confirms that the multiplication is performed in constant time.

***Keywords* — FPGA, High Precision, Multiplier Architecture, On-Line Arithmetic, VHDL, Virtex-II.**

I. Introduction

In the online arithmetic, the operands, as well as the results, flow serially through the computation in a digit by digit manner starting from the most significant digit (MSDF). The advantages of online arithmetic have been to permit the computation of all operations with MSDF mode which reduce the interconnection bandwidth between modules and allow parallelism between several operations. Moreover, to manipulate large numbers in parallel way presents always a hardware problem, because parallelism requires large processing circuits that must comprise several inputs-outputs pins according to the size of the operands, increases circuitry complexity.

The proposed method is based on the generation of partials products and their reductions with the MSDF mode, using the Borrow-Save addition approach. The result digit is converted to 2's complement by the On-the-fly conversion. This conversion algorithm manipulates the numbers in serial MSDF mode, thus the conversion of the result can begin as soon as the first redundant digit result is present.

II. FPGA Design and Programming

To define the behavior of the FPGA, the user provides a hardware description language (HDL) or a schematic design. The HDL form might be easier to work with when handling large structures because it's possible to just specify them numerically rather than having to draw every piece by hand. On the other hand, schematic entry can allow for easier visualization of a design.

Then, using an electronic design automation tool, a technology-mapped netlist is generated. The netlist can then be fitted to the actual FPGA architecture using a process called place-and-route, usually performed by the FPGA company's proprietary place-and-route software. The user will validate the map, place and route results via timing analysis, simulation, and other verification methodologies. Once the design and validation process is complete, the binary file generated (also using the FPGA company's proprietary software) is used to (re)configure the FPGA.

Going from schematic/HDL source files to actual configuration. The source files are fed to a software suite from the FPGA/CPLD vendor that through different steps will produce a file. This file is then transferred to the FPGA/CPLD via a serial interface (JTAG) or to an external memory device like an EEPROM.

The most common HDLs are VHDL and Verilog, although in an attempt to reduce the complexity of designing in HDLs, which have been compared to the equivalent of assembly languages, there are moves to raise the abstraction level through the introduction of alternative languages.

2.1 Xilinx

Xilinx, Inc. is the world's largest supplier of programmable logic devices, the inventor of the field programmable gate array (FPGA) and the first semiconductor company with a fabless manufacturing model.

Xilinx designs, develops and markets programmable logic products including integrated circuits (ICs), software design tools, predefined system functions delivered as intellectual property (IP) cores, design services, customer training, field engineering and technical support. Xilinx sells both FPGAs and CPLDs programmable logic devices for electronic equipment manufacturers in end markets such as communications, industrial, consumer, automotive and data processing.

Xilinx's FPGAs have even been used for the ALICE (A Large Ion Collider Experiment) at the CERN European laboratory on the French-Swiss border to map and disentangle the trajectories of thousands of subatomic particles.

The Virtex-II Pro, Virtex-4, Virtex-5, and Virtex-6 FPGA families are particularly focused on system-on-chip (SoC) designers because they include up to two embedded IBM PowerPC cores.

The ISE Design Suite is the central electronic design automation (EDA) product family sold by Xilinx. The ISE Design Suite features include design entry and synthesis supporting Verilog or VHDL, place-and-route (PAR), completed verification and debug using Chip Scope Pro tools, and creation of the bit files that are used to configure the chip.

2.2 Spartan family

The Spartan series targets applications with a low-power footprint, extreme cost sensitivity and high-volume; e.g. displays, set-top boxes, wireless routers and other applications.

The Spartan-6 family is built on a 45-nanometer (nm), 9-metal layer, dual-oxide process technology. The Spartan-6 was marketed in 2009 as a low-cost solution for automotive, wireless communications, flat-panel display and video surveillance applications.

The Spartan-3A consumes 70-90 percent less power in suspend mode and 40-50 percent less for static power compared to standard devices. Also, the integration of dedicated DSP circuitry in the Spartan series has inherent power advantages of approximately 25 percent over competing low-power FPGAs.

2.3 Multiplication on FPGA

Multiplication is basically a shift add operation. There are however, many variations on how to do it like, Scaling Accumulator Multipliers, Serial by Parallel Booth Multipliers, Ripple Carry Array Multipliers, Row Adder Tree Multipliers, Carry Save Array Multipliers, Look-Up Table Multipliers, Partial Product LUT Multipliers, Computed Partial Product Multipliers, Constant Multipliers from Adders, KCM multipliers, Limited Set LUT Multipliers, Wallace Trees, Booth Recoding.

III. Module Description

The use of the binary on-line arithmetic in high precision, presents generally a problem in the selection of the result digit. Since, the generation of this digit is done after the comparison between the residual noted by H[j], which is represented in redundant notation, and constants represented in 2's complement. This comparison is possible only after the conversion of the total residual to 2's complement. This requires the use of carry propagate adder depending on the operands size. Consequently the delay of generation of the digit result will be proportional to the time of this adder. Several works detailed in present solutions based on the overlapping of the selection intervals, thus the operation of comparison is carried out only on a fixed part of the residual.

This part is deduced after the estimation of the residual to a value which contains only its significant digits. While this solution has a correct theoretical base, but it complicates the representation of the new constants corresponding to the new intervals. This increases the complexity in the hardware implementation of the online operators. As solution, we present in this paper, a new method to compute a long precision multiplication of 2's complement numbers. The proposed method is based on the generation of partials products and their reductions with the MSDF mode, using the Borrow-Save addition approach [4]. The result digit is converted to 2's complement by the On-the-fly conversion.

This conversion algorithm manipulates the numbers in serial MSDF mode, thus the conversion of the result can begin as soon as the first redundant digit result is present. The architecture of our multiplier is implemented on XC2V2000-6 FPGA circuit for different size of operands (from 128 bits to 1024 bits). The main contributions of work are as follows: we expose the online multiplication problem in high precision and we prove that the truncation of the residual might gives error result. We also propose a new multiplication method in high precision which provides in constant time independent of operands size. We finally achieve the simulation and the implementation of this multiplier for several operand sizes.

The principle is based on the classic multiplication which is done in three steps: (i) the generation of the partial products, (ii) the reduction of the partial products and (iii) the final addition. Our approach has the same principle, except that it runs these three steps in serial with significant digit first. In our project, operands are represented in 2' s complement and introduced bit by bit with most significant bit first. All the result digits are represented in the redundant form, thus converting them to 2's complement is required. By using the on the fly conversion [5] the results digits are converted as soon as they are obtained.

IV. Module Separation

❖ Multiplier architecture
❖ Reduction block
❖ On the fly multiplier

4.1 Multiplier architecture

One block is used for the generation of the partials products, another to their reduction, and the last to the conversion from the redundant representation to the 2's complement representation. In the fig.1show the first block calculates the two partials products xj*Y[j-1]and yj*X[j]. As the bits xj and yj are equal to 0 or 1, the multiplication by these bits is easy to do by a group of AND gates. This last has the same size as the operand. Since, each iteration; a new bit is added to the operands Y [j-1] and X [j], until obtained completely the operands X and Y.

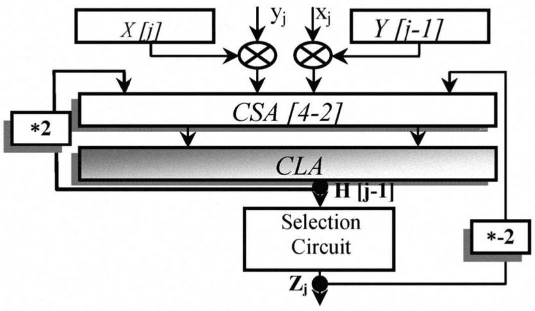

Fig.1. On-line multiplier Architecture

A. MSDF: online arithmetic

Online arithmetic algorithms operate in a digit-serial MSDF mode

• To compute the first digit of the result, δ + 1digits of the input operands needed. Thereafter, for each new digit of the operands, an extra digit of the result obtained is shown in Fig.2

• The online delay δ typically a small integer, e.g., 1 to 4.

Cycle	-2	-1	0	1	2 ..
Input	x1	x2	x3	x4	x5 .
Compute			z1	z2	z3 .
Output			z1	z2..	

δ = 2

Fig.2. Timing in online arithmetic.

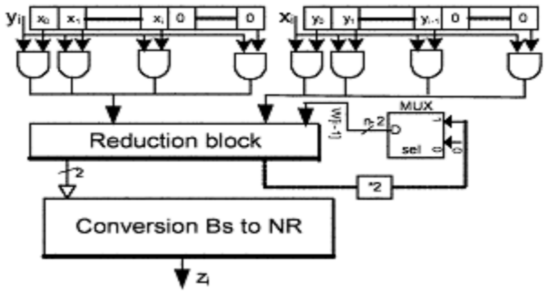

Fig.3. The multiplier Architecture.

4.2 Reduction block

The reduction block, reduces three numbers, twice of them are represented in 2's complement and the third one is represented in the Borrow-Save(BS) system. As shown in the section with BS representation, number is equal to the addition of a positive number and a negative number. The schematic block is illustrated on the fig 3. This latter, contained two stages. The first one is constituted by a carry save adder (CSA). The second one is constituted by PPM adders is shown in Fig.4. We noted that this reduction is done in constant time independently of the operand size used.

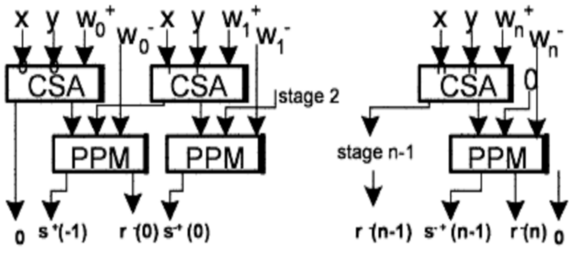

Fig.4. Reduction Block

4.3 On the fly multiplier

At the iteration (j) we obtained result digit Wj, which is represented in a BS system. To obtain the result digit in the 2's complement representation, two solutions are proposed. The first one suggests waiting until the reception of all the result digits then doing the subtraction. This solution, a large delay is added to the execution time. Since, the final addition requires carry propagation depending on the operands size. The second solution, is the use of the on the fly conversion [5], which consists in the conversion of the result as soon as they are obtained. This architecture is illustrated in fig.5

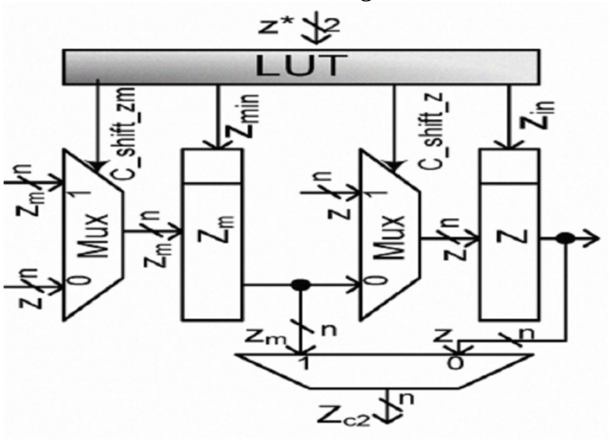

Fig.5. On the fly Multiplier

V. Existing Methodology

Several fields can also be cited such as the electronic signatures, the medical imagery, the generation of random numbers, the reduction of fraction in infinite precision and other more critical applications such as the nuclear simulators of stations and the military batteries of air defence. Therefore, the calculation in high precision was the object of several works who clearly reveal the adaptation of the on-line arithmetic for this kind of calculation. Since, in the online arithmetic, the operands, as well as the results, flow serially through the computation in a digit by digit manner starting from the most significant digit (MSDF).

The advantages of online arithmetic have been to permit the computation of all operations with MSDF mode which reduce the interconnection bandwidth between modules and allow parallelism between several operations. Moreover, to manipulate large numbers in parallel way presents always a hardware problem, because parallelism requires large processing circuits that must comprise several inputs-outputs pins according to the size of the operands, what increases circuitry complexity.

The Table I and Table II, show the iteration delay and the occupied area for several operands 128, 256, 512, and 1024. These results confirm that the left to right multiplication is done at constant time.

Table I: The Iteration Delay

Size	Iteration Delay	Route Delay	Logic Delay
128	5,53ns	3.75ns (67.8%)	1.783 (32.2%)
256	6,40ns	4.62ns (72.2%)	1.783 (27.8%)
512	8,47ns	6.69ns (79.0%)	1.783 (21.0%)
1024	11,81ns	10.02ns (84.9)	1.783 (15.1%)

Table II: The occupied Area

Size	Area (Slices)
1024	10750 (99%)
512	59906 (54%)
256	2949 (27%)
128	1471 (13%)

VI. Proposed Methodology

The multiplication method that we develop in this section is a serial method in MSDF mode. The result is obtained in the same manner in constant time independently of the operand's size. The method's principal is based on the classic multiplication which is done in three steps: the generation of the partial products, the reduction of the partial products and the final addition. Our approach has the same principle, except that it runs these three steps in serial with significant digit first. In order to illustrate the process of this method, we present an example on the fig 6 which shows the multiplication calculation of two.

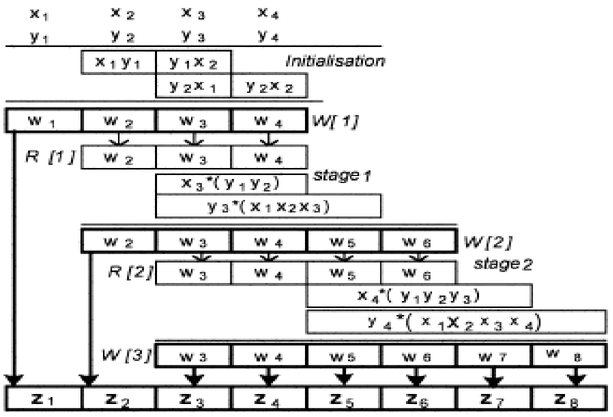

Fig.6. Execution of the method for n=4

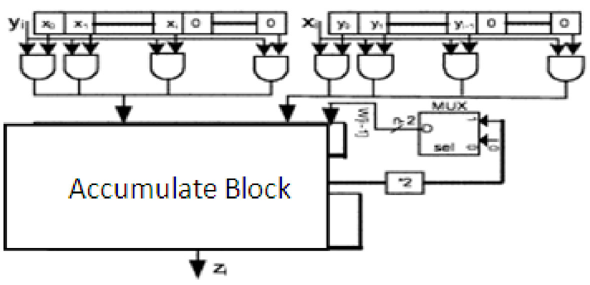

Fig.7. Proposed Multiplier Architecture

In fig.7 shows the proposed method, operands are represented in 2's complement and introduced bit by bit with most significant bit first. In order to illustrate the process of this method, we present an example on the fig. 6 which shows the multiplication calculation of two operands: X and Y which are presented with 4 bits as follow:

X=O, XIX2X3X4, Y=O, YIY2Y3Y4, such as X and Y < 1

We have to start the process with the initialization step, which consist in the addition of the four most significant bits (two bits from each operand). This step may be done in parallel or serial, and it presented the method's delay where the first result digit cannot be obtained until the end of this step.

During the first step, the first digit WI is obtained, and the X3 and Y3 inputs bits are introduced and their corresponding partial products are added to the residual R {1}. Then, the second digit result W2 is obtained which is equal to the most significant digit of the partial residual W {2}. In the same manner, all the digits results are computed, until the appearance of the two last bits of the X and Y operands. The final residual W{3}, is taken as a final part of the multiplication's result. It is important to note that all the result digits are represented in the redundant form, thus converting them to 2's complement is required. By using the on the fly conversion [5] the results digits are converted as soon as they are obtained.

A. The mathematic formalism of the method

According to the steps of the method defined above, we notice that the calculation of Z = X*Y comports at each step j, the computation of the sum noted by W[j] which is equal to the accumulation of the total residual R[j] with the partials products generated at this step. The recurrence equation which describes this formalism is:

$$\begin{cases} W[O] = 0 \\ W[1] = X[2] \times Y[2] \\ W[j] = 2 \times R[j] + 2^{-2}(y_j * X[j] + x_j * Y[j-1]) \end{cases}$$

The Table III & IV, show the iteration delay for size 256 and the occupied area for 328 was reduced compared to existing methodology.

Table III: The Iteration Delay

Size	Iteration Delay	Route Delay	Logic Delay
256	18.64ns	1.78(9.5%)	17.056(90.5%)

Table IV: The occupied Area

Size	Area(Slices)
328	**3584(9%)**

VII. Modelsim Simulation

ModelSim combines high performance and high capacity with the code coverage and debugging capabilities required to simulate larger blocks and systems and attain ASIC gate-level sign-off. Comprehensive support of Verilog, VHDL, and SystemC provide a solid foundation for single and multi-language design verification environments.

7.1. Advanced Code Coverage

ModelSim's advanced code coverage capabilities, ease of use, and high capacity lower the barriers for leveraging this valuable verification resource.

The ModelSim advanced code coverage capabilities provide valuable metrics for systematic verification. All coverage information is stored in the Unified Coverage

DataBase (UCDB), which is used to collect and manage all coverage information in a highly efficient database. Coverage utilities that analyze code coverage data, such as merging and test ranking, are available. Coverage results can be viewed interactively, post-simulation, or after a merge of multiple simulation runs. Code coverage metrics can be reported by instance or by design unit, providing flexibility in managing coverage data.

7.2. Effective Debug Environment

The ModelSim debug environment's broad set of intuitive capabilities for Verilog, VHDL, and System C make it the choice for ASIC and FPGA design.

ModelSim eases the process of finding design defects with an intelligently engineered debug environment. The ModelSim debug environment efficiently displays design data for analysis and debug of all languages.

ModelSim allows many debug and analysis capabilities to be employed post-simulation on saved results, as well as during live simulation runs. For example, the coverage viewer analyzes and annotates source code with code coverage results, including FSM state and transition, statement, expression, branch, and toggle coverage.

Signal values can be annotated in the source window and viewed in the waveform viewer, easing debug navigation with hyperlinked navigation between objects and its declaration and between visited files.

Race conditions, delta, and event activity can be analyzed in the list and wave windows. User-defined enumeration values can be easily defined for quicker understanding of simulation results. For improved debug productivity, ModelSim also has graphical and textual dataflow capabilities.

VIII. Simulation Result

The results obtained in multiplication of 2 large numbers through MODELSIM and the simulation diagram of LRSM algorithm is shown in figure 8 below.

Eg. The multiplication of 2 large numbers is 597 & 768 and the result obtained was 458496.

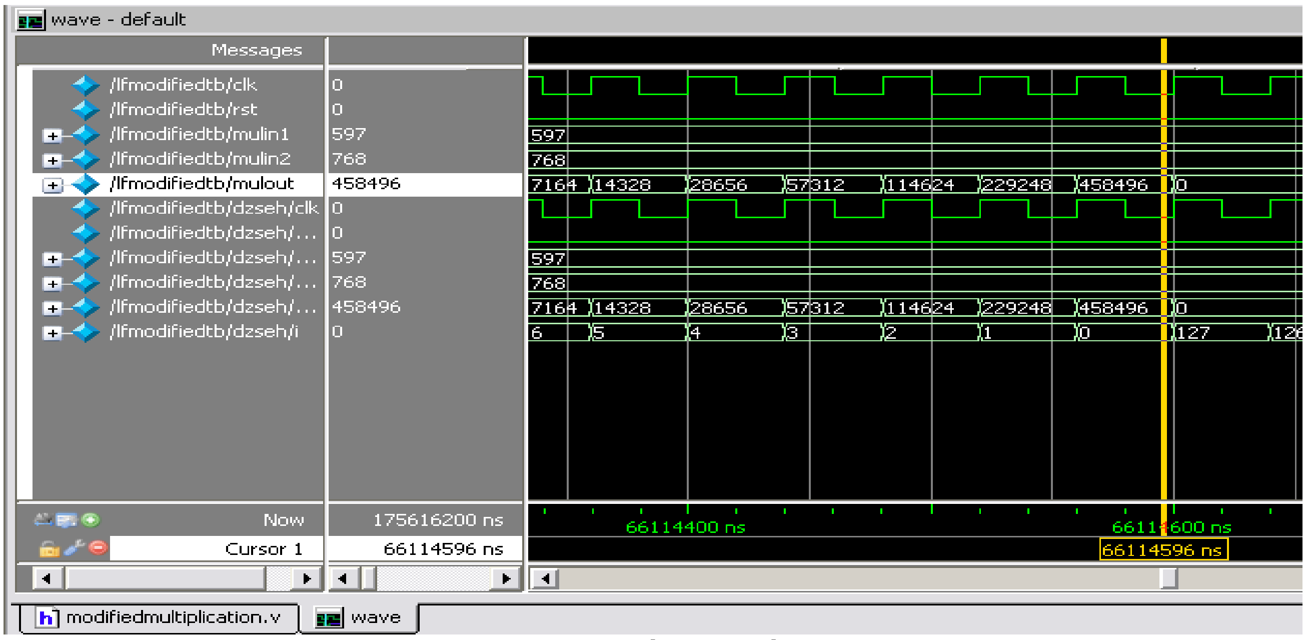
```
wave - default
Messages
/lfmodifiedtb/clk            0
/lfmodifiedtb/rst            0
/lfmodifiedtb/mulin1         597
/lfmodifiedtb/mulin2         768
/lfmodifiedtb/mulout         458496
/lfmodifiedtb/dzseh/clk      0
/lfmodifiedtb/dzseh/...      0
/lfmodifiedtb/dzseh/...      597
/lfmodifiedtb/dzseh/...      768
/lfmodifiedtb/dzseh/...      458496
/lfmodifiedtb/dzseh/i        0
Now                          175616200 ns
Cursor 1                     66114596 ns
```

Fig: 8. Simulation Result

IX. Conclusion

A new multiplication method is presented. This new approach is used for the multiplication of no redundant numbers. For each step, the proposed method consists first, on the generation of the partials products which are computed after the reception of the step's inputs. Secondly, the reduction of these partials products is completed using constant time adder. So, the digits results are generated in constant time. My computation method is suitable for the multiplication in high precision. The implementation of my multiplier architecture show that for the operands size more than the number of flips-flops available in the same line of the circuit chosen, the on-the -fly conversion augment the routing delay so the execution time is delayed. In existing system, the main problem of this method is the routing complexity which enlarges the route delay. To overcome this problem, avoid the reduction block so the route delay will be reduced.

References

[1] Y. K. Hornik, "Operateurs Arithmetiques Standards en ligne a tresgrande precision, conception et implementation," Phd Thesis, Grenoble, 93.

[2] M.D Ercegovac, AF Tenca, "On the design of high-radix on-line division for long precision Computer Arithmetic," Proceedings of the 14th IEEE Symposium, pp. 44 - 51, April 1999.

[3] A F. Tenca, and M.D Ercegovac, "A High-Radix multiplier design for variable long-precision computations," Proc. 31st Asilomar Conference on Signals, Systems and Computers, pp. 1173-1177, 1997.

[4] 1. L. Beuchat, J. M. Muller, "Automatic Generation of Modular Multipliers for FPGA Applications," LIP Research Report W2007-1, 2007.

[5] M.D Ercegovac, T.Lang, "On the fly conversion of redundant into conventional representation," IEEE Transaction On Computers, vol C-36, W7, Juilly 87 pp. 895-897

[6] A Guyot, Y.Herreros, 1. M. Muller, "JANUS, An on-line multiplier/divider for manipulating large numbers," Computer Arithmetic, Proceedings of 9th Symposium pp.l 06--111, September.1989

[7] Z. Huang and M.D. Ercegovac, "High-Performance Low-Power Left-to Right Array Multiplier Design," IEEE Trans. Computers, vol. 54(3) pp.272-283,2005.

[8] V. Lefevre, and P. Zimmermann, "Arithmetique flottante," Research Report, lNRIA, n05105, January 2004.

AUTHOR'S PROFILE

Mrs. Sudha P
She has received her B.E. (E.C.E) degree from Anna University in 2006 and M.Tech. (VLSI) degree from Sathyabama University, Chennai, India, in 2010. She has four year of teaching experience in several Engineering Colleges. Her area of interest is Digital Electronics and VLSI. sudhaamudha@gmail.com