DELAY-POWER PERFORMANCE COMPARISON OF MULTIPLIERS IN VLSI CIRCUIT DESIGN

Sumit Vaidya[1] and Deepak Dandekar[2]

[1]Department of Electronic & Telecommunication Engineering,
OM College of Engineering,
Wardha, Maharashtra, India
vaidyarsumit@gmail.com
[2]Department of Electronic Engineering,
B. D. College of Engineering,
Wardha, Maharashtra, India
d.dandekar@rediffmail.com

Abstract

A typical processor central processing unit devotes a considerable amount of processing time in performing arithmetic operations, particularly multiplication operations. Multiplication is one of the basic arithmetic operations and it requires substantially more hardware resources and processing time than addition and subtraction. In fact, 8.72% of all the instruction in typical processing units is multiplication. In this paper, comparative study of different multipliers is done for low power requirement and high speed. The paper gives information of "Urdhva Tiryakbhyam" algorithm of Ancient Indian Vedic Mathematics which is utilized for multiplication to improve the speed, area parameters of multipliers. Vedic Mathematics suggests one more formula for multiplication of large number i.e. "Nikhilam Sutra" which can increase the speed of multiplier by reducing the number of iterations.

Keywords

Multiplier, Vedic Mathematics, VLSI design

1. INTRODUCTION

Multiplication is an important fundamental function in arithmetic logic operation. Computational performance of a DSP system is limited by its multiplication performance [9] and since, multiplication dominates the execution time of most DSP algorithms [3]; therefore high-speed multiplier is much desired [4]. Currently, multiplication time is still the dominant factor in determining the instruction cycle time of a DSP chip. With an ever-increasing quest for greater computing power on battery-operated mobile devices, design emphasis has shifted from optimizing conventional delay time area size to minimizing power dissipation while still maintaining the high performance [5]. Traditionally shift and add algorithm has been implemented to design however this is not suitable for VLSI implementation and also from delay point of view. Some of the important algorithm proposed in literature for VLSI implementable fast multiplication is Booth multiplier, array multiplier and Wallace tree multiplier [9]. This paper presents the fundamental technical aspects behind these approaches.

The low power and high speed VLSI can be implemented with different logic style. The three important considerations for VLSI design are power, area and delay. There are many proposed logics (or) low power dissipation and high speed and each logic style has its own advantages in terms of speed and power [6-7].

2. OBJECTIVES

The objective of good multiplier to provide a physically compact high speed and low power consumption unit. Being a core part of arithmetic processing unit multipliers are in extremely high demand on its speed and low power consumption.

To reduce significant power consumption of multiplier design it is a good direction to reduce number of operations thereby reducing a dynamic power which is a major part of total power dissipation. In the past considerable effort were put into designing multiplier in VLSI in this direction.

3. METHODS AND PERFORMANCES

There are number of techniques that to perform binary multiplication. In general, the choice is based upon factors such as latency, throughput, area, and design complexity. More efficient parallel approach uses some sort of array or tree of full adders to sum partial products. Array multiplier, Booth Multiplier and Wallace Tree multipliers are some of the standard approaches to have hardware implementation of binary multiplier which are suitable for VLSI implementation at CMOS level.

3.1. Array Multiplier

Array multiplier is an efficient layout of a combinational multiplier. Multiplication of two binary number can be obtained with one micro-operation by using a combinational circuit that forms the product bit all at once thus making it a fast way of multiplying two numbers since only delay is the time for the signals to propagate through the gates that forms the multiplication array.

In array multiplier, consider two binary numbers A and B, of m and n bits. There are mn summands that are produced in parallel by a set of mn AND gates. n x n multiplier requires n (n-2) full adders, n half-adders and n^2 AND gates. Also, in array multiplier worst case delay would be (2n+1) td.

Array Multiplier gives more power consumption as well as optimum number of components required, but delay for this multiplier is larger. It also requires larger number of gates because of which area is also increased; due to this array multiplier is less economical [2] [11].Thus, it is a fast multiplier but hardware complexity is high [12].

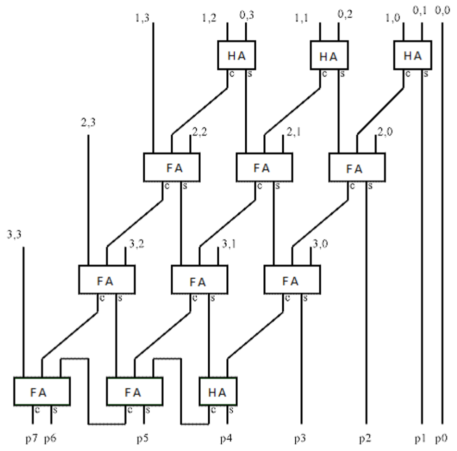

Figure 1. Array Multiplier

3.2. Wallace tree multiplier

A fast process for multiplication of two numbers was developed by Wallace [13]. Using this method, a three step process is used to multiply two numbers; the bit products are formed, the bit product matrix is reduced to a two row matrix where sum of the row equals the sum of bit products, and the two resulting rows are summed with a fast adder to produce a final product.

Three bit signals are passed to a one bit full adder ("3W") which is called a three input Wallace tree circuit, and the output signal (sum signal) is supplied to the next stage full adder of the same bit, and the carry output signal thereof is passed to the next stage full adder of the same no of bit, and the carry output signal thereof is supplied to the next stage of the full adder located at a one bit higher position.

Wallace tree is a tree of carry-save adders arranged as shown in figure 2. A carry save adder consists of full adders like the more familiar ripple adders, but the carry output from each bit is brought out to form second result vector rather being than wired to the next most significant bit. The carry vector is 'saved' to be combined with the sum later. In the Wallace tree method, the circuit layout is not easy although the speed of the operation is high since the circuit is quite irregular [2].

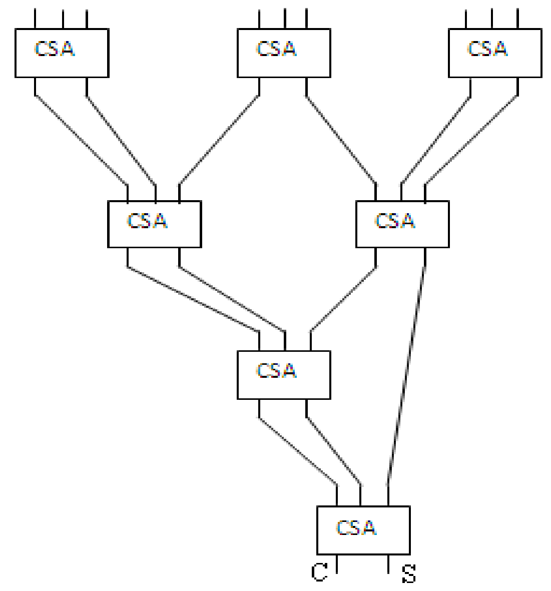

Figure 2. Wallace Tree Multiplier

3.3. Booth Multiplier

Another improvement in the multiplier is by reducing the number of partial products generated. The Booth recording multiplier is one such multiplier; it scans the three bits at a time to reduce the number of partial products [14]. These three bits are: the two bit from the present pair; and a third bit from the high order bit of an adjacent lower order pair. After examining each triplet of bits, the triplets are converted by Booth logic into a set of five control signals used by the adder cells in the array to control the operations performed by the adder cells.

To speed up the multiplication Booth encoding performs several steps of multiplication at once. Booth's algorithm takes advantage of the fact that an adder subtractor is nearly as fast and small as a simple adder.

From the basics of Booth Multiplication it can be proved that the addition/subtraction operation can be skipped if the successive bits in the multiplicand are same. If 3 consecutive bits are same then addition/subtraction operation can be skipped. Thus in most of the cases the delay associated with Booth Multiplication are smaller than that with Array Multiplier. However the performance of Booth Multiplier for delay is input data dependant. In the worst case the delay with booth multiplier is on per with Array Multiplier [9].

The method of Booth recording reduces the numbers of adders and hence the delay required to produce the partial sums by examining three bits at a time. The high performance of booth multiplier comes with the drawback of power consumption. The reason is large number of adder cells required that consumes large power [14].

```
Declare Reg A (0:7), M (0:7), Q (-1:7), COUNT (0:2), f
Declare bus INBUS (0:7), OUTBUS (0:7)
BEGIN: A<-0, COUNT<-0, f<-0
INPUT: M<-INBUS, Extend Q (0) ->Q (-1)
          Q (0:7) <-INBUS, Q (-1) <-INBUS (0);
ZERO TEST: if M=0□Q=0, then go to output
Loop: if f=0 then begin
ADD 1: if Q (6) Q (7) =01, then A (0:7) <-A (0:7) +M (0:7); else
SUBSTRACT 1: if Q (6) Q (7) =11, then A (0:7) <-A (0:7)-M (0:7); f<-1,
                    Else go to TEST; end
                    If f=1 then begin
ADD 2: if Q (6) Q (7) =00, then A (0:7) <-A (0:7) +M (0:7); f<-0, else
         if Q(6) Q(7)=10, then A(0:7)<-A(0:7)-M(0:7) end
TEST: if COUNT=7, then go to OUTPUT
RIGHTSHIFT: A (0) <-M (0)' □F□M (0) □A (0);
                  A (1:7).Q<-A.Q (-1:6);
INCREMENT: COUNT<-COUNT+1, go to loop;
OUTPUT: Q (6) <-0, OUTBUS<-A
                  OUTBUS<-Q (-1:6);
END;
```

Parameter	Array Multiplier	Wallace Tree Multiplier	Booth's Multiplier
Operation Speed	Less	High	Highest because the cycle length is as small as possible.
Time Delay	More (n+1)tFA	Log(n)	Less (ntFA/2 + ntFA)
Area	Maximum area because it uses a large number of adders.	Medium area because Wallace Tree used to reduce operands.	Minimum area because adder/subtracter is almost as small/fast as adder.
Complexity	Less complex	More complex	Most complex
Power consumption	Most	More	Less
FPGA implementation	Less efficient	Not efficient	Most efficient

Table 1. Comparison between Multipliers

4. ANCIENT VEDIC METHODS

The use of Vedic mathematics lies in the fact that it reduces the typical calculations in conventional mathematics to very simple ones. This is so because the Vedic formulae are claimed to be based on the natural principles on which the human mind works [15]. Vedic Mathematics is a methodology of arithmetic rules that allow more efficient speed implementation [16]. This is a very interesting field and presents some effective algorithms which can be applied to various branches of engineering such as computing [17].

4.1. Urdhva Tiryakbhyam Sutra

The given Vedic multiplier based on the Vedic multiplication formulae (Sutra). This Sutra has been traditionally used for the multiplication of two numbers. In proposed work, we will apply the same ideas to make the proposed work compatible with the digital hardware.

Urdhva Tiryakbhyam Sutra is a general multiplication formula applicable to all cases of multiplication. It means "Vertically and Crosswise" [15-16]. The digits on the two ends of the line are multiplied and the result is added with the previous carry. When there are more lines in one step, all the results are added to the previous carry. The least significant digit of the number thus obtained acts as one of the result digits and the rest act as the carry for the next step. Initially the carry is taken to be as zero. The line diagram for multiplication of two 4-bit numbers is as shown in figure 3.

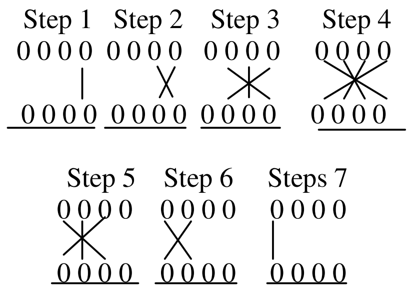

Figure 3. Line diagram for multiplication fo two 4-Bit Number

For this multiplication scheme, let us consider the multiplication of two decimal numbers (325 × 728). Line diagram for the multiplication is shown in Fig. 4. The digits on the two ends of the line are multiplied and the result is added with the previous carry. When there are more lines in one step, all the results are added to the previous carry. The least significant digit of the number thus obtained acts as one of the result digits and the rest act as the carry for the next step. Initially the carry is taken to be as zero.

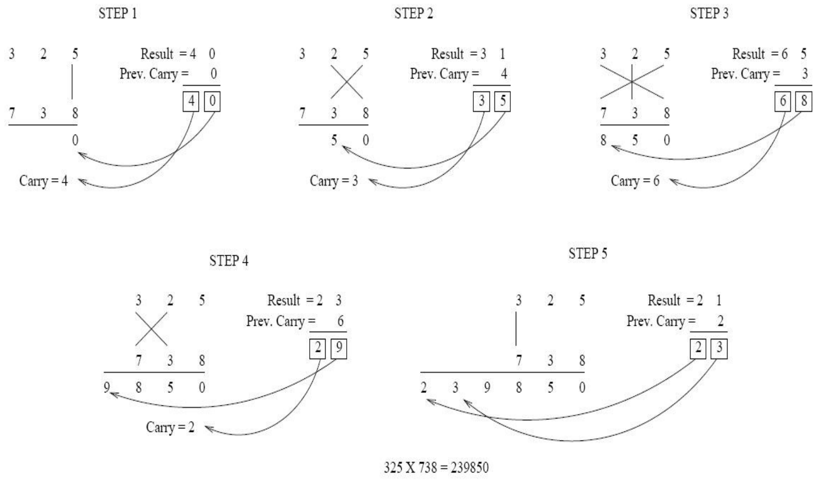

Figure 4. Multiplication of two decimal numbers by Urdhva Tiryakbhyam

Now we will extend this Sutra to binary number system. For the multiplication algorithm, let us consider the multiplication of two 8 bit binary numbers $A_7A_6A_5A_4A_3A_2A_1A_0$ and $B_7B_6B_5B_4B_3B_2B_1B_0$. As the result of this multiplication would be more than 8 bits, we express it as $\ldots R_7R_6R_5R_4R_3R_2R_1R_0$. As in the last case, the digits on the both sides of the line are multiplied and added with the carry from the previous step. This generates one of the bits of the result and a carry. This carry is added in the next step and hence the process goes on. If more than one lines are there in one step, all the results are added to the previous carry. In each step, least significant bit acts as the result bit and all the other bits act as carry. For example, if in some intermediate step, we will get 011, then1 will act as result bit and 01 as the carry. Thus we will get the following expressions:

$R_0=A_0B_0$
$C_1R_1=A_0B_1+A_1B_0$
$C_2R_2=C_1+A_0B_2+A_2B_0+A_1B_1$
$C_3R_3=C_2+A_3B_0+A_0B_3+A_1B_2+A_2B_1$
$C_4R_4=C_3+A_4B_0+A_0B_4+A_3B_1+A_1B_3+A_2B_2$
$C_5R_5=C_4+A_5B_0+A_0B_5+A_4B_1+A_1B_4+A_3B_2+A_2B_3$
$C_6R_6=C_5+A_6B_0+A_0B_6+A_5B_1+A_1B_5+A_4B_2+A_2B_4+A_3B_3$
$C_7R_7=C_6+A_7B_0+A_0B_7+A_6B_1+A_1B_6+A_5B_2+A_2B_5+A_4B_3+A_3B_4$
$C_8R_8=C_7+A_7B_1+A_1B_7+A_6B_2+A_2B_6+A_5B_3+A_3B_5+A_4B_4$
$C_9R_9=C_8+A_7B_2+A_2B_7+A_6B_3+A_3B_6+A_5B_4+A_4B_5$
$C_{10}R_{10}=C_9+A_7B_3+A_3B_7+A_6B_4+A_4B_6+A_5B_5$
$C_{11}R_{11}=C_{10}+A_7B_4+A_4B_7+A_6B_5+A_5B_6$
$C_{12}R_{12}=C_{11}+A_7B_5+A_5B_7+A_6B_6$
$C_{13}R_{13}=C_{12}+A_7B_6+A_6B_7$
$C_{14}R_{14}=C_{13}+A_7B_7$

$C_{14}R_{14}R_{13}R_{12}R_{11}R_{10}R_9R_8R_7R_6R_5R_4R_3R_2R_1R_0$ being the final product. Hence this is the general mathematical formula applicable to all cases of multiplication. All the partial products are calculated in parallel and the delay associated is mainly the time taken by the carry to propagate through the adders which form the multiplication array. So, this is not an efficient algorithm for the multiplication of large numbers as a lot of propagation delay will be involved in such cases. To overcome this problem, Nikhilam Sutra will present an efficient method of multiplying two large numbers.

5.2. Nikhilam Sutra

Nikhilam Sutra means "all from 9 and last from 10". It is also applicable to all cases of multiplication; it is more efficient when the numbers involved are large. Since it find out the compliment of the large number from its nearest base to perform the multiplication operation on it. Larger the original number, lesser the complexity of the multiplication. We will illustrate this Sutra by considering the multiplication of two decimal numbers (96 × 93) where the chosen base is 100 which is nearest to and greater than both these two numbers.

As shown in Fig. 5, we write the multiplier and the multiplicand in two rows followed by the differences of each of them from the chosen base, i.e., their compliments. We can write two columns of numbers, one consisting of the numbers to be multiplied (Column 1) and the other consisting of their compliments (Column 2). The product also consists of two parts which are distributed by a vertical line. The right hand side of the product will be obtained by simply multiplying the numbers of the Column 2 (7×4 = 28). The left hand side of the product will be found by cross subtracting the second number of Column 2 from the first number of Column 1 or vice versa, i.e., 96 - 7 = 89 or 93 - 4 = 89. The final result will be obtained by combining RHS and LHS (Answer = 8928).

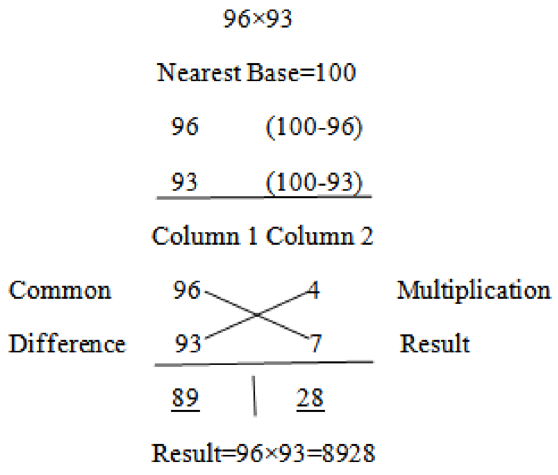

Figure 5. Multiplication of decimal numbers using Nikhilam Sutra

6. COMPARISION AND DISCUSSION

FPGA implementation results shows that multiplier Nikhilam Sutra based on of Vedic mathematics for multiplication of binary numbers is faster than multipliers based on Array and Booth multiplier[2]. It also proves that as the number of bits increases to N, where N can be any number, the delay time is greatly reduced in Vedic Multiplier as compared to other multipliers. Vedic Multiplier has the advantages as over other multipliers also for power and regularity of structures.

Name of Multiplier	Array Multiplier		Booth Multiplier		Vedic Multiplier	
	8×8 Bit	16×16 Bit	8×8 Bit	16×16 Bit	8×8 Bit	16×16 Bit
Delay (ns)	47	92	117	232	27	39

Table 2. Comparison of Multiplier w.r.t. delay (ns)

There are number of techniques for logic implementation at circuit level that improves the power dissipation, area and delay parameters in VLSI design. Implementation of parallel Multiplier in CPL logic shows significant improvement in power dissipation [1]. CPL requires more number of transistors to implement as compared to the CMOS and provides only a little improvement in speed. Pass Transistor Logic which offers better performance over both the CMOS and CPL in terms of delay, power, speed and transistor count. The PTL outperforms the CMOS implementation in speed and great in power dissipation, with approximately same transistor count [1]. When compared to CPL, PTL is faster and shows improvement in power and transistor count.

Parameters	CMOS	Complementary Pass Transistor Logic	Pass Transistor Logic
Number of Transistor	Most	More	Less
Area	Maximum	Medium	Minimum
Power	Most	More	Less
Delay	Most	More	Less
Speed	Less	Medium	High

Table 3. Comparison between multiplier designs in three different Logics

7. CONCLUSION

It can be concluded that Booth Multiplier is superior in all respect like speed, delay, area, complexity, power consumption. However Array Multiplier requires more power consumption and gives optimum number of components required, but delay for this multiplier is larger than Wallace Tree Multiplier. Hence for low power requirement and for less delay requirement Booth's multiplier is suggested. Ancient Indian Vedic Mathematics gives efficient algorithms or formulae for multiplication which increase the speed of devices.

Urdhva Tiryakbhyam, is general mathematical formula and equally applicable to all cases of multiplication. Also, the architecture based on this sutra is seen to be similar to the popular array multiplier where an array of adders is required to arrive at the final product. Due to its structure, it suffers from a high carry propagation delay in case of multiplication of large number. This problem can solved by Nikhilam Sutra which reduces the multiplication of two large numbers to the multiplication of two small numbers.

The power of Vedic Mathematics can be explored to implement high performance multiplier in VLSI applications. Nikhilam Sutra in Vedic Mathematics is less complex than Urdhva Tiryakbhyam which can be tested with its implementation with different logics in VLSI.

Further the work can be extended for optimization of said multiplier to improve the speed or to minimize the delay.

REFERENCES

[1] "A New Low Power 32×32- bit Multiplier" Pouya Asadi and Keivan Navi, World Applied Sciences Journal 2(4):341:347, 2007, IDOSI Publication.

[2] "A Novel Parallel Multiply and Accumulate (V-MAC) Architecture Based On Ancient Indian Vedic Mathematics" Himanshu Thapliyal and Hamid RArbania.

[3] "Low power and high speed 8x8 bit Multiplier Using Non-clocked Pass Transistor Logic" C.Senthilpari, Ajay Kumar Singh and K. Diwadkar, 1-4244-1355-9/07, 2007, IEEE.

[4] Kiat-seng Yeo and Kaushik Roy "Low-voltage,low power VLSI sub system" Mc Graw-Hill Publication.

[5] Jong Duk Lee, Yong Jin Yoony, Kyong Hwa Leez and Byung-Gook Park "Application of Dynamic Pass Transistor Logic to 8-Bit Multiplier" Journal of the Korean Physical Society, Vol. 38, No. 3, pp.220-223,March 2001

[6] C. F. Law, S. S. Rofail, and K. S. Yeo "A Low-Power 16×16-Bit Parallel Multiplier Utilizing Pass-Transistor Logic" IEEE Journal of Solid State circuits, Vol.34, No.10, pp. 1395-1399, October 1999.

[7] Oscal T. C. Chen, Sandy Wang, and Yi-Wen Wu "Minimization of Switching Activities of Partial Products for Designing Low-Power Multipliers" IEEE Transaction on VLSI System.Vol. 11, No.3, pp. 418-433, June 2003.

[8] "Low Power High Performance Multiplier" C.N. Marimuthu and P.Thiangaraj, ICGST-PDCS, Volume 8, December 2008.

[9] "ASIC Implementation of 4 Bit Multipliers" Pravinkumar Parate ,IEEE Computer society. ICETET,2008.25.

[10] Steven A. Guccione MARIO j. Gonzalez "A Cellular Multiplier for Programmable Logic"Computer Engineering Research Center,Department of Electrical and Computer Engineering, University of Texas at Austin, USA, Febuary, 1994.

[11] Morris Mano, "Computer System Architecture",PP. 346-347, 3rd edition,PHI. 1993.

[12] Jorn Stohmann Erich Barke, "A Universal Pezaris ArrayMultiplier Generator for SRAM-Based FPGAs" IMS- Institute of Microelectronics System, University of Hanover Callinstr, 34,D-30167 Hanover,Germany.

[13] Moises E. Robinson and Ear Swartzlander, Jr."A Reduction Scheme to Optimize the Wallace Multiplier" Department of Electrical and Computer Engineering, University of Texas at Austin, USA.

[14] Tam Anh Chu, "Booth Multiplier with Low Power High Performance Input Circuitary", US Patent, 6.393.454 B1,May 21, 2002.

[15] "A Reduced-Bit Multiplication Algorithm For Digital Arithmetic" Harpreet Singh Dhilon And Abhijit Mitra, International Journal of Computational and Mathematical Sciences, Waset, Spring, 2008.

[16] "Lifting Scheme Discrete Wavelet Transform Using Vertical and Crosswise Multipliers" Anthony O'Brien and Richard Conway, ISSC, 2008,Galway, June 18-19.

[17] H. Thapliyal and M. B. Shriniwas and H. .Arbania, "Design and Analysis of a VLSI Based High Performance Low Power Parallel Square Architecture", Int. Conf. Algo. Math.Comp. Sc., Las Vegas,June 2005, pp. 72-76.

Authors

S. R. Vaidya received the Bachelor of Engineeing degree in Electronic Engineering from R.T.M. Nagpur University, Nagpur in 2007 and pursuing his M.Tech in Electronic Engineering from B. D. College of Engineering, Wardha, R.T.M. Nagpur University. He is working towards M.Tech research project at R.T.M. Nagpur University. He is currently working as a Lecturer in Electronic and Telecommunication Engineering Department at OM College of Engineering, Wardha. His areas of interest include High performance VLSI Design and VHDL based system design.

Deepak R. Dandekar received the Bachelor of Engineering degree in Electronic Engineering from Nagpur University, in 1990 and the M.Tech (Electronic Engineeing) degree from Vishveshariya National Institute of Technology, Nagpur in 2004. He is currently working as a Assistant Professor in Department of Electronic Engineering at B. D. College of Engineering, Sewagram, Wardha since 1992. His area of interests include VLSI design and optimization.