



A Novel Reconfigurable Processor for Multi way Circuit partitioning Using Genetic Algorithm

Rajaram Sivasubramanian¹ Dhinesh Joseph² and Abhaikumar Varadhan³

Department of Electronics and Communication Engineering,

Thiagarajar college of Engineering, Anna University,

Madurai-625015, Tamilnadu (State), India

rajaram_siva@tce.edu

<http://www.tce.edu>

Abstract

Genetic Algorithms (GAs) are robust techniques based on natural selection that can be used to solve a wide range of problems including VLSI layout optimization, Circuit partitioning and Boolean satisfiability. This paper proposes 'CIRPART' – a novel architecture for implementing Genetic Algorithm (GA) used for circuit Multiway Partitioning in VLSI physical design automation. CIRPART is built upon 'user programmable', Field Programmable Gate Array (FPGA) device and employs a combination of Pipelining and Parallelization. CIRPART provides flexibility and also achieves speedups over software based GA. The design uses four modules and three external memories. The proposed design was coded in VHDL and was functionally verified by writing a test bench and simulating it using ModelSim. The design was synthesized on Xilinx Virtex V100CS144 FPGA chip using VHDL. CIRPART achieves more than 100% improvement in processing speed as compared to the software implementation while the quality of results remains unaltered.

Keywords: Genetic algorithm, Field Programmable Gate array, Multi-way circuit partitioning, VHDL

1. Introduction

In View of the increasing complexity of VLSI circuits, there is a growing need for sophisticated CAD tools to automate the analysis, synthesis and verification steps in the design of VLSI systems. In VLSI design automation applications, Circuit Partitioning algorithms are used to achieve various objectives like Circuit Layout, Circuit Packaging and Circuit Simulation. Numerous optimization techniques have been applied to solve the graph and circuit partitioning problems. U. R. Kodres has given a concurrent approach that tends to build partitions simultaneously and in a more uniform manner. [1]. Kerningham and Lin [2] presented the first iterative improvement algorithm (K-L Algorithm) for partitioning graphs in which the quality of the final partition often heavily depends on the initial partition. Fiduccia and Mattheyses [3] proposed a more efficient method for implementing K-L Algorithm leading to a fast, linear time algorithm for partitioning (F-M Algorithm), but this method tends to converge to local minima when tie occur between many cells. Krishnamurthy proposed some refinements over the F-M Algorithm by changing the

method of choosing the best cell for movement to the other blocks by adding a Look-Ahead (LA) mechanism [4]. Though the method gave some improvement in cut size, the speed was much reduced. These disadvantages have been overcome by applying parallelism using Genetic algorithms.

The remainder of the paper is organized as follows: Section (2) focuses on need for hardware realization of Genetic Algorithm, Section (3) emphasizes on proposed architecture, Section (4) concentrate on Experimental results, and Section (5) focuses on Conclusion and future work.

2. Need for Hardware Realization of Genetic Algorithm

GAs has been recognized as a robust general-purpose optimization technique. But software implementation of GAs to increasingly complex problems causes unacceptable delays in the optimization process. A hardware-based GA is both feasible and desirable. The key to the hardware implementation of the GA is to divide the algorithm into sub-sections and perform the latter using dedicated hardware modules in parallel [5]. The use of reconfigurable hardware for the design of GA has been reported in the literature [6], [7], [8]. In Stephan Scott's behavioral-level implementation of a GA [6], the targeted application was optimization of an input function. In [7], Tommi Rintala designed and implemented a GA on a PLD, using Altera hardware description language (AHDL). In [8], Loring Wirbel designed a number of GAs and implemented in a text compression chip. In [9], Paul Graham and Brent Nelson implemented GA for traveling Salesman problem in reconfigurable hardware. The memory bottleneck is inevitable since GA requires a large memory to store the population. As a result, high-speed memory is a possibility but the hardware becomes expensive or alternatively a low-cost memory with reduced performance. Therefore, in contrast to the Simple GA, the Compact GA was more suitable for hardware implementation [10]. In the current work, Novel GA based architecture for Circuit partitioning (CIRPART) is proposed and its realization in Reconfigurable Processor



is presented. The use of pipelining and parallelization in GA minimize the logic resources used within FPGA.

3. Proposed Architecture

CIRPART is specifically optimized towards solving the circuit Multiway Partitioning. There is no limit on the size of the problem that the core can handle, but the core requires external RAM for storing the netlist information, population and the fitness values of each chromosome. The size of the RAM is directly proportional to the number of nets and modules in the design, number of chromosomes and number of partitions. Therefore too big problems would require a relatively large amount of external RAM. CIRPART uses a processing-pipeline for performing the computationally intensive parts of the partitioning algorithm.

CIRPART architecture comprises of four main modules. They are Central Processing Module (CPM), Fitness Evaluation Module (FEM), Parent Selection Module (PSM) and Genetic Operation Module (GOM). In addition, CIRPART uses three external memory blocks (RAMs) namely Input Memory (IM), Population Memory (PM) and Fitness Memory (FM). The block diagram of the GA processor is shown in Figure 1 and the description of the Bus used in the architecture is given in Table 1.

3.1. Central Processing Module (CPM)

CPM generates control signals for rest of the blocks of the design. The various states of CPM are given in Fig3. CPM generates control signals for rest of the blocks of the design. CPM enters a loop in which the three functions of Fitness calculation, Parent selection, and crossover and mutation operations are carried out in sequence until the generation counter inside the Main controller reaches the generation count loaded into No Generations in control register. With each generation, the generation count is incremented by one. At the end of last generation, CPM enables FEM for one last time and outputs the final population and final fitness using the top-level output signals.

3.2. Fitness Evaluation Module (FEM)

Once GOM generates a complete new population, FEM generates fitness values for each of the generated chromosomes. Upon receiving the signal from CPM, FEM determines for each chromosome, the Partition-imbalance Cost and the Net-cut Cost simultaneously. Once the Cost evaluation process is complete for a single chromosome, the Total cost is stored in the Fitness Memory and the above said process is repeated for each chromosome with all registers reset until the Chromosome Count equals No Chromosomes. Once the Cost evaluation process is complete an enable signal is sent to FEM.

3.3. Parent Selection Module (PSM)

PSM performs 'Tournament selection' on the population by reading four random fitnesses from the Fitness memory upon receiving an active high signal from the Main Controller. Internally, PSM consists of a random number generator, a comparator for comparing unsigned integers, registers to latch the generated random

addresses, and a control state machine. The control state machine generates control/enable signals for different blocks in the module.

3.4. Genetic Operation Module (GOM)

GOM performs the crossover and mutation operations on the two parent chromosomes, the starting addresses of which are generated by PSM. The Population memory is divided into two parts, namely the low bank, and the high bank. At any time, the parent population is stored into one of the banks and the child population generated by GOM is stored into the other bank. Upon receiving an active high signal from CPM, the chromosome for each of the parents is read from the Population memory based upon the addresses generated by PSM.

The Central Processing Module (CPM) generates control signals for rest of the modules of the design. Before starting the core process, the Control Registers are loaded with 'legal' values using the CPU interface. After loading the control registers, an active high pulse on the Start control input starts the GA process. After receiving the Start signaling, CPM accepts the netlist along with the other mentioned Generics from the top-level inputs. The Input storage area stores the various inputs, from where it is read repeatedly by the other modules for the GA process. After receiving the inputs, CPM generates the initial population randomly and stores it in to the Population memory. The Fitness Evaluation Module (FEM) calculates the fitness of each chromosome and stores it in Fitness Memory. The Parent Selection Module (PSM) uses Tournament Selection. GOM uses the memory addresses of the selected parents to perform the genetic operations and stores the generated off springs in the Population memory. After the entire set of new population is generated, FEM computes the fitness of each of the elements of the new population, and stores the fitness into the Fitness Memory. After the specified number of generation is executed (based on Generation Counter value), CPM outputs the final population along with the fitness of each chromosome. The design is coded in VHDL and implemented in Xilinx FPGA. The three external RAM modules used by the design are Input memory, Population memory and Fitness Memory.

In order to solve the circuit-partitioning problem using GA, the following chromosome representation is used. Each chromosome contains a sequence of words, each word corresponding to a distinct module. Each word denotes the number of the partition (binary encoded number in the range between 0 and k-1 where k is the number of partitions) in which that corresponding module has to be placed. The chromosome representation is depicted in Figure 2. Word-length in the chromosome depends on the number of partitions needed. Therefore Chromosome-length is the number of modules in the net list multiplied with the word-length. Memory access is provided at word level (Gene Level). This approach avoids delays during genetic operations.

CIRPART [19] is specifically optimized towards solving the circuit Multiway Partitioning. There is no limit on the size of the problem that the core can



handle, but the core requires external RAM for storing the netlist information, population and the fitness values of each chromosome.

4. Experimental Results

The main objective of this paper is to propose a novel architecture for Mutiway circuit partitioning and implement the same in both software and hardware. The proposed novel architecture CIRPART was coded using VHDL language and then implemented in Xilinx Virtex V100CS144 FPGA chip. The GA was implemented in the C++ programming language on a Pentium III (800 MHz) machine with 128 MB memory. Hardware GA was compared with software GA for different generation count and different benchmarks with Default GA parameters (Table2). Each benchmark of variable sizes was chosen to validate the proposed architecture. The benchmarks range in size from 125 to 2844 cells and 147 to 3282 nets. Performance results for Hardware GA and Software GA for different Generation count and different Chromosome Counts were given in Table3 &4. Execution time of the Hardware genetic algorithm is significantly improved over the software Genetic algorithm. From the simulation results, it is clear that the hardware implementation speed is approximately 100 times more than the software implementation. The proposed architecture is implemented in Xilinx Virtex V100CS144 FPGA using VHDL coding. The FPGA chip layout of the architecture is shown in Figure 4.

5. Conclusion

In this paper a new GA Based architecture for Circuit Mulltiway Partitioning and its implementation in Reconfigurable hardware is presented. The design takes into account the practical limitations of memory data bus imposed by the memory chips available. In order to enable the use of almost any memory chip along with the design, the design uses configurable parameters (generics), which can easily change the memory address and data bus widths during compilation time. The design was synthesized for a maximum clock frequency of 117 MHz on Xilinx Virtex V100CS144. At this frequency the design achieves more than 100 times improvement in processing speed over the software implementation.

The chromosome representation used in this paper requires a relatively large amount of external memory to store the population and netlist. Alternate chromosome representations are being explored in order to reduce the memory requirements. Future work will concentrate on adapting hardware/software co-design approach to implement the architecture and proposing Memetic algorithm (Genetic algorithm + Local search algorithm) for the circuit partitioning problem.

6. References

[1] U. R. Kodres, "Partitioning and Card Selection," in Design Automation of Digital Systems, M. A. Breuer, ed., pp. 173-212, Prentice Hall 1972.
 [2] B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," Bell

Systems Technical Journal, vol. 49, pp.291-307, 1970.
 [3] C. M. Fiduccia and R. M. Mattheyses, "A Linear-time heuristic for improving network partitions," Proc. Design Automation Conf., pp. 175-181, 1982.
 [4] B. Krishnamurthy, "An Improved min-cut algorithm for partitioning VLSI net-works," IEEE Trans. Computers, vol. c-33, pp. 438-446, 1984.
 [5] PaulGraham and Brent Nelson, "Genetic Algorithms in Software and in Hardware- A performance Analysis of workstation and custom Computing Machine Implementation", in IEEE Symposium on FPGAs for custom Computing Machines, Reconfigurable Logic Laboratory, Brigham Young University, Provo, UT, USA, 1996.
 [6] Stephen Donald Scott, "A hardware based genetic algorithm", Master's thesis, University of Nebraska, August 1994.
 [7] Tommi Rintala, "Hardware implementation of GA", September 20 2000.
 [8] Loring Wirbel, "Compression chip is first to use genetic algorithms", Electronic Engineering Times, page 17, December 1984.
 [9] PaulGraham and Brent Nelson, "A Hardware Genetic Algorithm for the Traveling Salesman Problem on Splash2", Fifth International workshop on Field Programmable Logic and Applications, pp.352-261, Aug 1995.
 [10] Chatchawit and Prabhas, "A Hardware Implementation of compact Genetic algorithm", in Proceedings of the 2001 IEEE Congress on Evolutionary Computation, pp.624-629, Seoul, Korea, May 2001.
 [11] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Welsey Publishing Company, 1989.
 [12] S. Areibi, "A Review of Circuit Partitioning", Technical report, School of Engineering, University of Guelph, June 2000.
 [13] G. A. Korn and T. M. Korn, Mathematical Handbook for Scientists and Engineers, New York: McGraw-Hill Book Company, Inc., 1961.
 [14] S. Dutta and W. Deng, "A Probability-based approach to VLSI circuit partitioning," Proc. Design Automation Conf., pp. 100-105, 1996.
 [15] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," Science, vol. 220, no.4598, pp. 671-680, May 1983.
 [16] John R. Koza, Forrest h Bennett III, Stephen L Jeffrey L, Martin A and David Andre, "Evolving Computer Programs using Rapidly Reconfigurable Field Programmable Gate Arrays and Genetic Programming", Sixth international symposium on FPGA,1998.
 [17] K.A. De Jong and W.M. Spears, " Using genetic algorithms to solve NP-complete problems", in J.David Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms, pp. 124132. Morgan Kaufmann Publishers, 1989.
 [18] S.D. Brown, R.J. Francis, J. Rose and Z.G. Vranesic, Field-Programmable Gate Arrays, Kluwer Academic Publishers, USA, 1999.



- [19] S.Rajaram, Dhinesh Joseph, V.Abhaikumar,
 "CIRPART-A FPGA Based Processor for circuit
 Multiway Partitioning using Genetic algorithm"
 Proceedings of National Conference on
 Communications (NCC 2004), IIT Madras, 2004

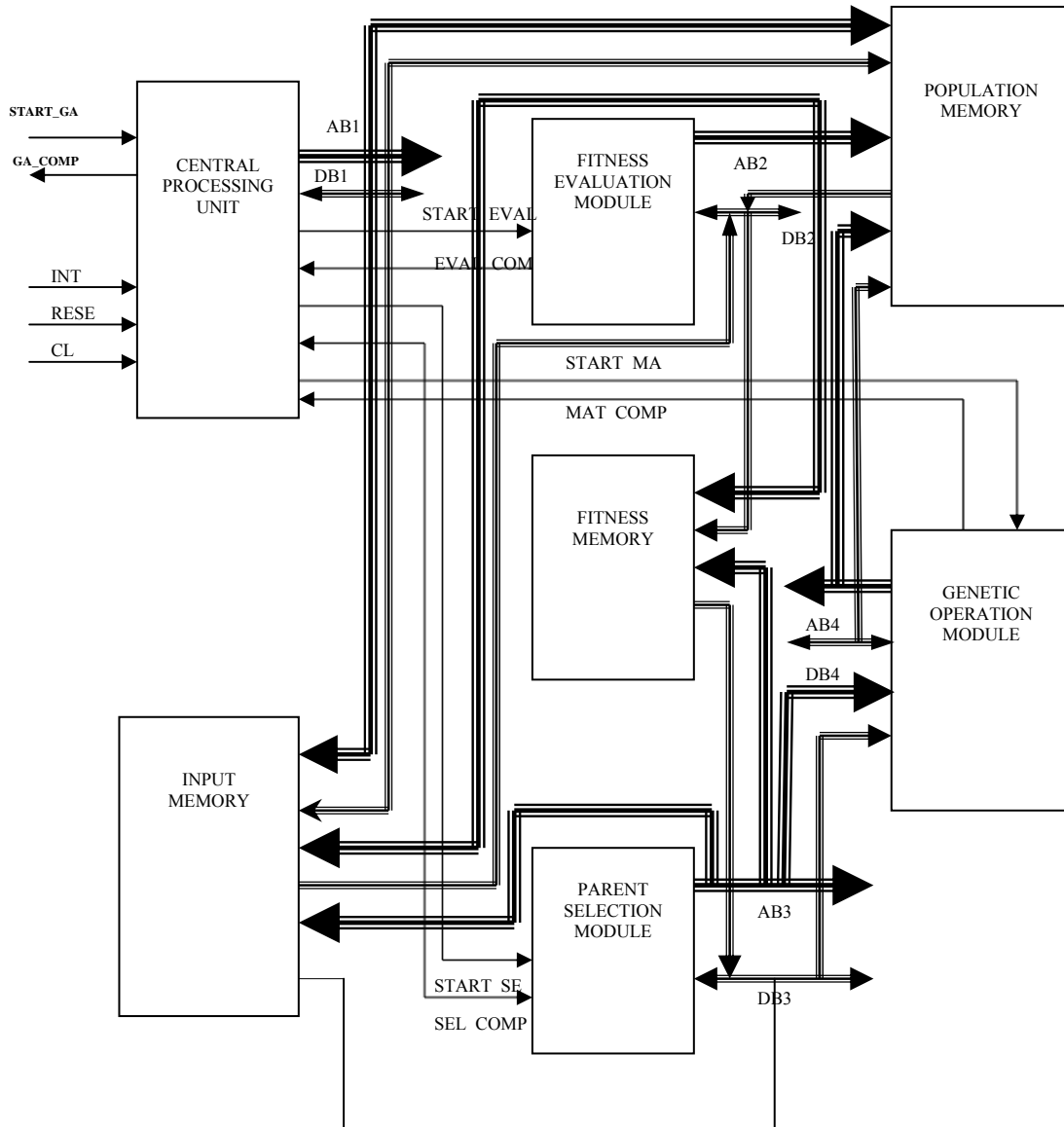


Figure 1. Block Diagram of the GA Processor

M1	M2	M3	M4	M5	M6	M7	M8
5	1	2	4	5	3	2	1

Here,

Module M1 belongs to partition 5
 Module M2 belongs to partition 1 and so on.

Figure 2: Chromosome Representation of Circuit-multiway-Partitioning

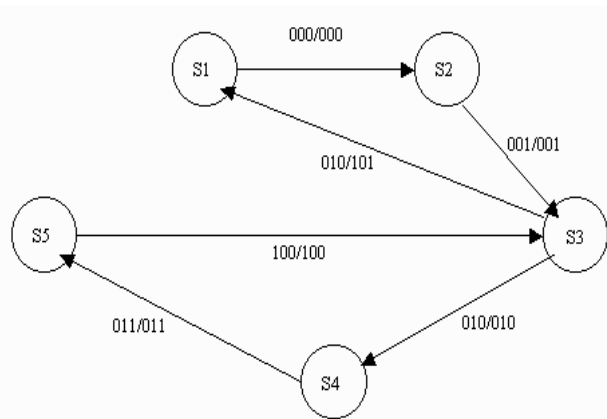


BUS NAME	DESCRIPTION
AB1	Address Bus of Central Processing Module
DB1	Data Bus of Central Processing Module
AB2	Address Bus of Fitness Evaluation Module
DB2	Data Bus of Fitness Evaluation Module
AB3	Address Bus of Parent Selection Module
DB3	Data Bus of Parent Selection Module
AB4	Address Bus of Genetic Operation Module
DB4	Data Bus of Genetic Operation Module

Table 1: Description for Bus in the Architecture of the GA Processor.

PARAMETERS	PARAMETER VALUE
Population Size	20
Generation Count	20
Crossover Rate	0.99
Mutation Rate	0.01
Crossover Type	Uniform
Selection Type	Tournament

Table 2: Default GA parameters



INPUT	SIGNAL	SIGNIFICANCE
000	StartGA	Starts the GA Process.
001	StartInit	Initializes the Population memory with random population.
010	StartEval	Enables the Fitness Evaluation Module.
011	StartSel	Enables the Parent Selection Module.
100	StartMat	Enables the Genetic Operation Module.
OUTPUT	SIGNAL	SIGNIFICANCE
000	Ready	Indicates the successful reception of all inputs.
001	InitComp	Indicates the completion of random initial population generation in the Population memory.
010	EvalComp	Indicates the completion of the Fitness Evaluation Process.
011	SelComp	Indicates the completion of the Parent Selection Process.
100	MatComp	Indicates the completion of the 'Mating' Process.
101	GAComp	Indicates the completion of the GA Process. This happens at the end of the last generation.

Figure 3: State machine of Central Processing module.



CIRCUIT	#MODULES	#NETS	SOFTWARE TIME (ms)			HARDWARE TIME (ms)		
			G _N =20	G _N =60	G _N =100	G _N =20	G _N =60	G _N =100
Fract	125	147	420	1160	1900	4.18	10.98	16.96
Struct	1888	1920	5650	16950	28300	53.90	160.05	231.97
Primary1	752	904	2600	7700	12800	25.13	73.90	109.87
Ckt1	833	902	2700	8000	13300	26.05	76.77	114.16
Ckt2	3014	3029	8140	24320	40500	77.27	229.00	300.98
Ckt5	1663	1721	5280	15740	26200	50.47	148.63	214.75
Ckt6	1607	1618	5020	14960	24900	47.90	141.26	204.10
Ckt7	1515	1658	5100	15200	25300	48.71	143.53	207.38
Ckt8	2595	2751	7360	21980	36600	69.91	206.97	272.73
Ckt9	1752	1674	5200	15500	25800	49.62	146.36	211.46
Ckt10	2844	3282	9000	26900	44800	85.34	253.30	332.83

Table 3: Performance results for Hardware GA and Software GA for different Generation Count.

CIRCUIT	#MODULES	#NETS	SOFTWARE TIME (ms)			HARDWARE TIME (ms)		
			C _N =20	C _N =60	C _N =100	C _N =20	C _N =60	C _N =100
Fract	125	147	420	1300	2200	4.18	11.46	18.38
Struct	1888	1920	5650	17300	26800	53.90	164.37	227.04
Primary1	752	904	2600	8200	13200	25.13	75.16	112.42
Ckt1	833	902	2700	8550	13750	26.05	77.84	117.93
Ckt2	3014	3029	8140	23900	35900	77.27	222.32	282.11
Ckt5	1663	1721	5280	15900	23300	50.47	150.51	211.05
Ckt6	1607	1618	5020	15100	22100	47.90	141.96	200.27
Ckt7	1515	1658	5100	15300	22600	48.71	144.07	203.85
Ckt8	2595	2751	7360	21200	32300	69.91	202.30	258.86
Ckt9	1752	1674	5200	15650	22850	49.62	147.81	207.91
Ckt10	2844	3282	9000	25500	36200	85.34	247.52	311.60

Table 4: Performance results for Hardware GA and Software GA for different Chromosome Count

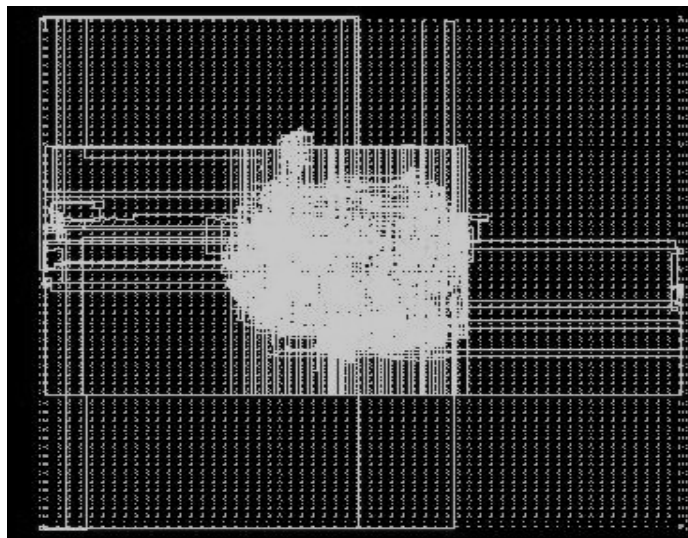


Figure 4: Chip Layout of Virtex V100CS144





Rajaram Sivasubramanian received his BE and ME degrees in Electronics and Communication Engineering from Thiagarajar College of Engineering and Alagappa Chettiar College of Engineering and Technology in 1994 and 1996, respectively, and pursuing PhD from Madurai Kamaraj University, Madurai, India. Currently, he is an Assistant Professor of Electronics and Communication Engineering at Thiagarajar College of Engineering, Madurai, India. He is a Member of IEEE, VLSI Society of India and published more than 25 papers in both International and National Conferences. His Research interests include FPGA Design, VLSI testing, Design Optimization.

Dhinesh Joseph Received his B.E degree from Thiagarajar College of Engineering, Madurai in the year 2002. His research interest includes VLSI Design and Testing, FPGA Implementation and Genetic Algorithms. Now He is with Infosys (P) Ltd , Chennai as a senior Software Engineer.



Abhaikumar Varadhan received his BE and M.E. degree from PSG college of Technology, Coimbatore, India in 1977 and 1979 respectively. He received his PhD degree from Indian Institute of Technology, Madras, India in 1984. Currently, he is the Principal and Head of Electronics and Communication Engineering at Thiagarajar College of Engineering, Madurai, India. He is a senior Member of IEEE. He is the recipient of two awards for research, teaching and advising excellence. He has co-authored 70 technical papers in reputed journals, International and National Conferences.

