VLSI Design

CAD for Gigascale SoC Design and Verification Solutions

Guest Editors: Shiyan Hu, Zhuo Li, and Yangdong Deng

CAD for Gigascale SoC Design and Verification Solutions

VLSI Design

CAD for Gigascale SoC Design and Verification Solutions

Guest Editors: Shiyan Hu, Zhuo Li, and Yangdong Deng

Editorial Board

Contents

Hindawi Publishing Corporation
VLSI Design
Volume 2011, Article ID 398390, 2 pages
doi:10.1155/2011/398390

Editorial
CAD for Gigascale SoC Design and Verification Solutions

Shiyan Hu,[1] Zhuo Li,[2] and Yangdong Deng[3]

[1] *Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, MI 49931, USA*
[2] *IBM Austin Research Laboratory, 11501 Burnet Road, MS 904-6G016, Austin, TX 78758-3493, USA*
[3] *Institute of Microelectronics, Tsinghua University, Beijing 100084, China*

Correspondence should be addressed to Shiyan Hu, shiyan@mtu.edu

Received 5 December 2011; Accepted 5 December 2011

As VLSI technology enters the nanometer regime, the design complexity is rapidly increasing with timing, power, routability, and reliability. Salient design automation techniques for scalable SoC design and verification solutions that could greatly improve the design quality are highly desired. This special issue is dedicated to the research problems in all aspects of System-on-Chip (SoC) implementation and verification. The papers selected for this special issue address new optimization, simulation, and verification techniques containing theoretical and/or applied contributions that emphasize the scalability to future large designs. They represent a good panel on the state-of-the-art development in scalable VLSI design and verification algorithms and methodologies.

This special issue contains ten papers. Six papers focus on the physical design optimizations including routing, buffer insertion, gate sizing, partitioning, floorplanning, and leakage analysis. Other three papers address the low-power test and resource sharing issue from different perspectives. Another paper discusses the emerging security issue in SoC design and presents some interesting new challenges.

In the paper entitled "*Efficient congestion mitigation using congestion-aware steiner trees and network coding topologies,*" the authors present a new Steiner tree construction technique for reducing congestion and minimizing overflow. With the integration of network coding, their technique can achieve significant improvements in routability.

In the paper entitled "*Shedding physical synthesis area bloat,*" the authors present a novel physical synthesis flow addressing the area bloat issue. To mitigate the significant area increase due to buffer insertion and gate sizing, a set of practical physical synthesis tools are designed which perform much better than existing algorithms in industrial designs.

In the paper entitled "*Buffer planning for IP placement using sliced-LFF,*" the authors present a buffer planning technique during floorplanning. Given a fixed-outline constraint, buffer insertion can be performed using a sliced less flexibility first algorithm which allows to distinguish geometric difference between floorplan candidates with the same topological structure. The proposed two-stage technique can significantly improve the success rate of buffer insertion and run faster than existing algorithms.

In the paper entitled "*Finding the energy efficient curve: gate sizing for minimum power under delay constraints,*" the authors present a gate sizing technique which targets to satisfy the timing constraint with minimal dynamic and leakage power consumption. Based on a new metric called energy delay gain to quantify the timing and power tradeoff, geometric programming technique is applied to optimize the circuits in the approach.

In the paper entitled "*The impact of statistical leakage models on design yield estimation,*" the authors review a set of closed form approximations on leakage power and present the study on the impact of different sums of lognormal approximation to the leakage of multiple leaky devices. Through comparing with CDF matching technique, they show that modeling the tail probability in CDF matching is critical.

In the paper entitled "*Wirelength minimization in partitioning and floorplanning using evolutionary algorithms,*" the authors present a memetic algorithm for partitioning and floorplanning. The algorithm uses multiple local search phases to reduce delay in partitioning and area in floorplanning.

In the paper entitled "*Weighted transition based reordering, columnwise bit filling, and difference vector: a power*

aware test data compression method," the authors improve the existing hamming distance-based reordering technique for test data compression. The new technique can achieve high compression rate while reducing test power with little on-chip area overhead.

In the paper entitled "*Suitability of various low-power testing techniques for IP core-based SoC: a survey,*" the authors present a survey in the area of low-power testing for IP core-based SoC. Various existing techniques including external testing, BIST techniques, and DFT techniques are reviewed.

In the paper entitled "*Efficient resource sharing architecture for multistandard communication system,*" the authors present a dedicated hardware module which can be reconfigured for the OFDM wireless LAN standard and MCDMA standard. The new hardware can efficiently share resources for the two standards.

In the paper entitled "*SoC: a real platform for IP reuse, IP infringement, and IP protection,*" the authors address the security issue due to the IP reuse in SoC. They discuss how to locate attacks, categorize the infringement, apply strategic analysis in IP-based SoC design flow, and highlight several new research opportunities in this emerging area.

Acknowledgments

We would like to thank all the authors for their excellent contributions to the special issue and all reviewers for their highly valuable comments.

Shiyan Hu
Zhuo Li
Yangdong Deng

Hindawi Publishing Corporation
VLSI Design
Volume 2011, Article ID 896241, 9 pages
doi:10.1155/2011/896241

Research Article

Wirelength Minimization in Partitioning and Floorplanning Using Evolutionary Algorithms

I. Hameem Shanavas[1] and Ramaswamy Kannan Gnanamurthy[2]

[1] *Department of Electronics and Communication, M. V. J. College of Engineering, Bangalore 560067, India*
[2] *Vivekanandha College of Engineering for Women, Trichengode 637205, Tamilnadu, India*

Correspondence should be addressed to I. Hameem Shanavas, hameemshan@gmail.com

Received 16 December 2010; Revised 23 April 2011; Accepted 6 July 2011

Academic Editor: Zhuo Li

Minimizing the wirelength plays an important role in physical design automation of very large-scale integration (VLSI) chips. The objective of wirelength minimization can be achieved by finding an optimal solution for VLSI physical design components like partitioning and floorplanning. In VLSI circuit partitioning, the problem of obtaining a minimum delay has prime importance. In VLSI circuit floorplanning, the problem of minimizing silicon area is also a hot issue. Reducing the minimum delay in partitioning and area in floorplanning helps to minimize the wirelength. The enhancements in partitioning and floorplanning have influence on other criteria like power, cost, clock speed, and so forth. Memetic Algorithm (MA) is an Evolutionary Algorithm that includes one or more local search phases within its evolutionary cycle to obtain the minimum wirelength by reducing delay in partitioning and by reducing area in floorplanning. MA applies some sort of local search for optimization of VLSI partitioning and floorplanning. The algorithm combines a hierarchical design technique like genetic algorithm and constructive technique like Simulated Annealing for local search to solve VLSI partitioning and floorplanning problem. MA can quickly produce optimal solutions for the popular benchmark.

1. Introduction

Partitioning and floorplanning (PF) has been an active area of research for at least a quarter of a century. The main reason that partitioning has become a central and critical design task today is due to the enormous increase of system complexity in the past and the expected further advances of microelectronic system design and fabrication. Widely accepted powerful high-level synthesis tools allow the designers to automatically generate huge systems by just changing a few lines of code in a functional specification. Synthesis and simulation tools often cannot cope with the complexity of the entire system under development, and also designers want to concentrate on critical parts of a system to speed up the design cycle. Thus the present state of design technology often requires a partitioning of the system with fast and effective optimization. Moreover, fabrication technology makes increasingly smaller feature sizes and augmented die dimensions possible to allow a circuit for accommodating several million transistors. However, circuits are restricted in size and in the number of external connections. So the fabrication technology requires partitioning of a system into components by arranging the blocks without wasting free space. The direct implementation of large circuit will occupy large area. Hence the large circuit is to be split into small subcircuit. This will minimize the area of the system and the complexity of the system. When they are partitioned, the connection between two modules should be minimum (or the number net cut by the partition). This is known as cut size and hence this plays a major role in partitioning. It is a design task by applying an algorithmic method to solve difficult and complex combinatorial optimization problems like breaking a large system into pieces [1].

The process of determining block shapes and positions with area minimization objective is referred to as floorplanning. A common strategy for blocks floorplanning is to determine in the first phase and then the relative location of the blocks to each other based on connection-cost criteria.

In the second step, block sizing is performed with the goal of minimizing the overall chip area and the location of each block is finalized [2]. When the partitioning and floorplanning are combined (PF), the criteria like power, cost, and clock speed of each module are the subobjective to be optimized. The main objective to be optimized is the wirelength that can be achieved by incorporating Memetic Algorithm (MA) in both the partitioning and floorplanning [3, 4].

In early stages, many interchanging methods have been used which resulted in local optimum solution. And later some of the mathematical methods are followed. Some heuristics are also used which resulted in better result but it has its own advantage and disadvantage. Since there may be many solutions possible for this problem, stochastic optimization techniques are utilized and until now many techniques have been known like Simulated Annealing Algorithm (SA) which combines the Local Search Algorithm with the Metropolis algorithm.

SA is a simple algorithm and does not need much memory, but it takes a long time to reach the desired solution. Kernighan and Lin [5] proposed a two-way graph partitioning algorithm which has become the basis for most of the subsequent partitioning algorithms. Fiduccia and Mattheyses [6] modified the K-L algorithm to a more efficient algorithm by suggesting moving one cell at a time and by designing a new data structure separately. As a kind of global optimization technique Genetic Algorithm (GA) which borrows the concept of generation from biological system had been used for physical design problems like circuit partitioning, floorplanning, and so forth. This technique has been applied to several problems, most of which are graph related because the genetic metaphor can be most easily applied to these types of problems. GA requires more memory but it takes less time than SA [7]. Lots of researchers have proposed their theories to partition circuit using GA. Work of [8] proposed hardware genetic algorithm by developing GA and local search processor that uses some external memory to overcome the problem of local maxima/minima. Authors of [8] expressed the probability of selection of chromosome as function of both the best and worst chromosome while [4] proposed different cost functions in order to achieve multiple objectives of minimizing delay, cutsize area, and power. The authors of [9] proposed two GA one based on 0-1 encoding and other based on integer encoding. Work done in [10] developed an adaptive strategy for partitioning of circuits in which population size, crossover rate, and mutation rate are modified during the execution in order to enhance performance. Number of enhancements like crossover operator mutation or choosing different fitness functions can still be made to achieve optimal solutions. This means that theory of GA still provides opportunities for new inventions that can help in inventing new solutions for physical design problems. This paper proposes memetic algorithm to solve the graph partitioning and floorplanning problem. The algorithm incorporates several genetic algorithm features, namely, selecting a population and crossover of the selected chromosomes to get better stable solutions. The algorithm starts by incorporating the circuit as unweighted connected graph in which each vertex represents gate and edge represents an interconnection between two gates and thereafter applying the GA metaphor to generate a partition that is highly interconnected within but disconnected from other subpartitions while trying to minimize the number of cuts and time consumed. This work tried to hybrid two algorithms like Genetic Algorithm and Simulated Annealing for overcoming the disadvantage of one another. Such type of algorithms is called memetic algorithm, (see Figure 1).

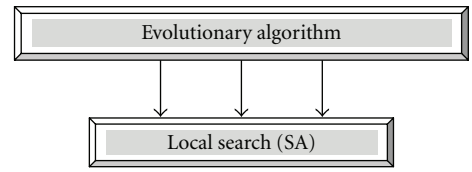


Figure 1: Memetic Algorithm.

This work addresses the problem of VLSI netlist partitioning with the objective of reducing delay and then floorplanning with the objective of reducing area to minimize the wirelength.

2. Partitioning

A better circuit partition will reduce connection among subcircuits and result in a better routing area of the layout. The challenge is that the circuit partitioning problem belongs to the class of well-known NP-hard optimization problems [11]. The problem can be viewed as a graph partitioning problem where each module (gates etc.) is taken as vertices and the connection between them represents the edges between the nodes [12, 13].

To start the algorithm, n gates are placed on the graph as n vertex, and an initial population is chosen as the different permutations of various vertices of the given graph. The problem reduces to associate each chromosome and each partition of the graph. An algorithm based on Genetic algorithm is proposed that can be used to partition a number of nodes. The example for a circuit and graph representation is given (Figure 2).

Given an unweighted connected graph $G = (V, E)$ on set of vertices V and edges E. Let $k \geq 2$ be a given integer, find a partition $V_1,\ V_2,\ V_3, \ldots, V_k$ set of vertices V such that

(i) the induced graph $G_i = (V_i, E_i)$, for all values $i = 1, 2, 3, \ldots, k$, is connected;

(ii) the following values are minimized $\min\{|V_1|, |V_2|, |V_3|, \ldots, |V_K|\}$.

This k-Connected Problem is a particular instance of graph partitioning problem [14]. Because exact and approximation algorithms that run in polynomial time do not exist for graph partitioning problems in general, it is necessary to solve the problem using heuristic algorithms. Genetic Algorithm is a heuristic technique that seeks to imitate the behavior of biological reproduction and their ability to collectively solve a problem. The GA starts with several

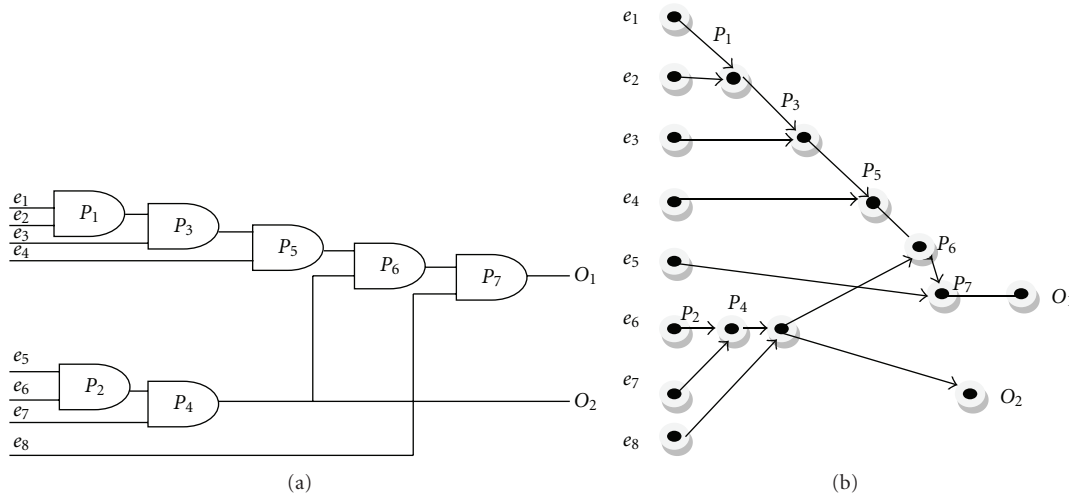


Figure 2: Representation of a circuit and Directed acyclic graph.

alternative solutions to the optimization problem, which are considered as individuals in a population. These solutions are coded as binary strings, called chromosomes. The initial population is constructed randomly. These individuals are evaluated using partitioning by specific fitness function. GA then uses these individuals to produce a new generation of hopefully better solutions. In each generation, two of the individuals are selected probabilistically as parents, with the selection probability proportional to their fitness. Crossover is performed on these individuals to generate two new individuals, called offspring, by exchanging parts of their structure. Thus each offspring inherits a combination of features from both parents. The next step is mutation. An incremental change is made to each member of the population, with a small probability. This ensures that GA can explore new features that may not be in the population yet. It makes the entire search space reachable, despite the finite population size. The basic foundation of the algorithm is to represent each vertex in the graph as a location that can represent a logic gate, and a connection is represented by an edge [15, 16].

3. Floorplanning

A module B is a rectangle of height H_B, width W_B, and area A_B. A supermodule consists of several modules, also called a subfloorplan, (see Figure 3). A floorplan for n modules consists of an enveloping rectangle R subdivided by horizontal lines and vertical lines into n non-overlapping rectangles. Each rectangle must be large enough to accommodate the module assigned to it. In the given problem, a set of hard modules and outline-constraints are provided. The modules inside the given outline have freedom to move. A feasible packing is in the first quadrant such that all the modules inside the outline should not duplicate and overlap each other. The objective is to construct a feasible floorplan R such that the total area of the floorplan R is minimized and simultaneously satisfy floorplanning constraint. Given a set

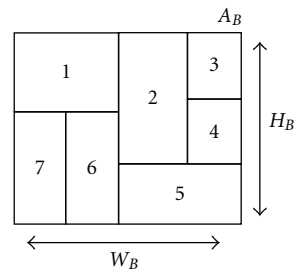


Figure 3: Floorplan module.

of modules to a specified number of cluster satisfying the predescribed properties. To solve the floorplanning problem, first construct a network graph, and then run the given algorithm to get the solution. The graph consists of two kinds of vertices horizontal and vertical. The network graph $G = (V, E)$ has to be constructed. "$*$" represents vertical slicing of blocks. "+" represents horizontal slicing of blocks (see Figure 4).

4. Memetic Algorithm

The memetic algorithm [17, 18] is a combination of an Evolutionary Algorithm (EA) and Local Search (LS). The EAs are used for finding the global optimum. The LS used here will aid the EA to convergence speed. The evolutionary algorithm used in this work is genetic algorithm. There are two types of memetic algorithm.

Exhaustive MA. Few solutions are selected from the final generation and improved using local search.

Intermediate MA. After a predetermined number of iteration by GA, local search is applied to few random individuals. This is done to have a better solution than the local maxima. This work deals with intermediate memetic algorithm.

4.1. Populations and Chromosomes. In GA-based optimizations, a set of trial solutions are assembled as a population. The parameter set representing each trial solution or individual is coded to form a string or chromosome and each individual is assigned a fitness value by evaluation of the objective function. The objective function is to only link between the GA optimizer and the physical problem.

4.2. Parents. Following this initialization process, pairs of individuals are selected (with replacement) from the population in a probabilistic manner weighted by their relative fitness and designated as parents.

4.3. Children. A pair of offspring, or children, are then generated from the selected pair of parents by the application of simple stochastic operators. The principle operators are crossover and mutation. Crossover occurs with a probability of pcross (typ. 0.6–0.8) and involves the random selection of a crossover site and the combining the two parent's genetic information. The two children produced share the characteristics of the parents as a result of this recombination operators. Other recombination operators are sometimes used, but crossover is the most important. Recombination (e.g., crossover) and selection are the principle way that evolution occurs in a GA optimization.

4.4. Mutation. Mutation introduces new and unexplored points into the GA optimizer's search domain. Mutation randomly changes the genetic makeup of the population. Mutation is much less important than recombination and occurs with a probability pmutation (typ. 0.05) which is much less than pcross.

4.5. New Generation. Reproduction consisting of selection, recombination and mutation continues until a new generation is created to replace the original generation. Highly fit individuals, or more precisely highly fit characteristics produce more copies of themselves in subsequent generation resulting in a general drift of the population as a whole towards an optimal solution point. The process can be terminated in several ways: threshold on the best individual (i.e., the process stops when an individual has an error less than some amount E), number of generations exceeds a preselected value, or some other appropriate criteria. A simple Genetic Algorithm must be able to perform five basic tasks: encode the solution parameters in the form of chromosomes, initialize a starting point population, evaluate and assign fitness values to individuals in the population, perform reproduction through the fitness weighted selection of individuals from the population, and perform recombination and mutation to produce members of the next generation.

Consider the graph $G = (V, E)$ with vertex $|v| = u$ and in integer $1 < k < n/4$. Initialize a randomly generated population P of k elements. Population P has 1 to k elements. Assume each parent p_1 to p_k belongs to the population P. Perform two point crossover for p_a and p_b from population P using the fitness function $(f) = k \cdot M(p)/n$, where $M(p)$ is the number of node of partition with maximum cardinality among n partitions. Assume $P_a(I)$ and $P_b(I)$ are the children from p_a and p_b, respectively. If $P_a(I)$ has not satisfied the fitness ($P_a(I)$ is not in I), then choose p_a randomly from $j > i$. Swap $P_a(i)$ and $P_a(j)$. Copy the first element k elements of p_a in q_1, q_3. If $P_b(I)$ has not satisfied the fitness ($P_b(I)$ is not in I), then choose p_b randomly from $h > k$. Swap $P_b(h)$ and $P_b(i)$. Copy the first element k elements of p_b in q_2, q_4. Create two vectors L, L' with $2(n - k)$ elements. If $j \cdot \mathrm{mod}2 = 1$, then $L(i) = P_a[k + (j + 1)/2]$ and $L'(i) = P_b[k + (j + 1)/2]$, else $L(j) = P_a[k + (j + 1)/2]$ and $L'(j) = P_b[k + (j + 1)/2]$. Check the fitness of $L(i)$, $L'(i)$, $L(j)$, and $L'(j)$. If $L(i)$ is not in q_1, then copy $L'(i)$ in q_1 and $L(i)$ in q_2. If $L(j)$ is not in q_3, then copy $L'(j)$ in q_3 and $L(j)$ in q_4. Repeat the process again with $L(i)$ and $L(j)$, $L'(i)$ and $L'(j)$ to get new offspring. The new offsprings can have more fitness value or less fitness value depending upon the parents. Less fitness offspring can be discarded then to reduce the number of cycles.

The fitness of the individual in partitioning is based on the delay of the module.The fitness of the individual is given by weighted evaluations of maximum delay (D). D_a identifies a particular subgraph, D_m is a predetermined maximum value, and D_f is the weighting factor. The sum of the weighting factors equals one. The complete fitness function is

$$G_p = \left\{ \frac{D_a}{D_m} > 1, \frac{D_a}{D_m} \leq 1, \frac{D_a}{D_m} * D_f \right\} \tag{1}$$

Assuming an individual is fully feasible and meets constraints, the value of $G_p \leq 1$, with smaller values being better.

At the beginning, a set of Polish expressions is given for floorplannig. It is denoted as P, randomly generated expression to compose a population. The fitness for floorplanning of the individual is based on the area of the module. Area of a block can be calculated by the general formula $A = LW$, where L stands for length of the module and W stands for width of the module.

$$\text{Total Area} = \sum A_n. \tag{2}$$

The fitness of the individual is given by weighted evaluations of maximum area (A). A_a identifies a particular subgraph, A_m is a predetermined maximum value, and A_f is the weighting factor. The sum of the weighting factors equals one. The complete fitness function is

$$G_f = \left\{ \frac{A_a}{A_m} > 1, \frac{A_a}{A_m} \leq 1, \frac{A_a}{A_m} * A_f \right\}. \tag{3}$$

5. Local Search

After a prescribed number of iterations by evolutionary algorithm, local search algorithm is applied to few random individual, to have better solution. Local search methods are iterative algorithms that tend to enhance solution by stepwise improvement and make an attempt to reach optimum solutions, which more often results in suboptimal

solutions by trapping themselves in local minima/maxima. The simplest form of local search is repetitively flipping elements in a solution resulting again in the objective function. Eventually local minima will be reached whereby flipping any element in the solution will result in loss of object. Although these algorithms are simple, there have been many complex improvements for CAD tools which involve large dynamic memory and linked list usages. For refining the solution obtained by GA, the local search (LS) is applied. This can be used before crossover or after crossaover, it can also be used for parents selection, and used before or after mutation to increase the number of fitness variables (see Algorithm 1).

5.1. Optimization by Simulated Annealing. Simulated Annealing algorithm is applied for Local Search process since SA is not a Local Search Algorithm. Here simulated annealing method is performed on finally generated offspring to improve the fitness. This method is called as intermediate MA.

Simulated annealing is a stochastic computational method for finding global extrema to large optimization problems. It was first proposed as an optimization technique by Kirkpatrick et al. in 1983 [19] and Cerny in 1984 [20]. The optimization problem can be formulated by describing a discrete set of configurations (i.e., parameter values) and the objective function to be optimized. The problem is then to find a vector that is optimal. The optimization algorithm is based on a physical annealing analogy. Physical annealing is a process in which a solid is first heated until all particles are randomly arranged in a liquid state, followed by a slow cooling process. At each (cooling) temperature, enough time is spent for the solid to reach thermal equilibrium, where energy levels follow a Boltzmann distribution. As temperature decreases the probability tends to concentrate on low energy states. Care must be taken to reach thermal equilibrium prior to decrease the temperature. At thermal equilibrium, the probability that a system is in a macroscopic configuration with energy is given by the Boltzmann distribution. The behavior of a system of particles can be simulated using a stochastic relaxation technique developed by Metropolis et al. [21]. The candidate configuration for the time is generated randomly. The new candidate is accepted or rejected based on the difference between the energies associated with states. The condition to be accepted is determined by

$$p = \frac{p_r}{p_q} = \frac{\exp\left(-E_t - E_q\right)}{K_t} > 1. \tag{4}$$

One feature of the Metropolis way of Simulated Annealing algorithm is that a transition out of a local minimum is always possible at nonzero temperature. Another evenly interesting property of the algorithm is that it performs a kind of adaptive divide and conquer approach. Gross features of the system appear at higher temperatures, fine features develop at lower temperatures. For this application, it used the implementation by Ingber [22].

```
Begin
   Initialize population P;
   For i := 1 To size of(P) Do
            Individual := P_i;
            Individual := Local_Search (Individual);
   Repeat Until (terminate=True) Do
         For i := 1 To #recombination Do
            Select two parents i_a, i_b ε P randomly;
               i_c := Recombination(i_a,i_b);
               i_c := Local Search(i_c);
   Add individual i_c to P;
   P := Select(P);
   If (P converged Then
         For i := 1 To size of(P), i != index(Best) Do
      Individual := P_i
Individual := Local_Search(Mutate(Individual));
End
```

Algorithm 1

5.2. Partitioning Based on Simulated Annealing. The basic procedure in simulated annealing is to start with an initial partitioning and accept all perturbations or moves which result in a reduction in cost. Moves that result in a cost increase are accepted. The probability of accepting such a move decreasing with the increase in cost and also decreasing in later stages of the algorithm are given in (4). A parameter T, called the temperature, is used to control the acceptance probability of the cost-increasing moves. Simulated Annealing algorithm for partitioning the modules will be described here. The cells are partitioned using simulated annealing so as to minimize the estimated wire length. A formal description of the simulated annealing algorithm was given in Section 5. There are two methods for generating new configurations from the current configuration. Either a cell is chosen randomly and placed in a random location on the chip, or two cells are selected randomly and interchanged. The performance of the algorithm was observed to depend upon r, the ratio of displacements to interchanges. Experimentally, r is chosen between 3 and 8. A temperature-dependent range limiter is used to limit the distance over which a cell can move. Initially, the span of the range limiter is twice the span of the chip. In other words, there is no effective range limiter for the high temperature range. The span decreases logarithmically with the temperature.

$$\begin{aligned} L_{WV}(T) &= L_{WV}(T_i)\left[\frac{\log T}{\log T_i}\right], \\ L_{WH}(T) &= L_{WH}(T_i)\left[\frac{\log T}{\log T_i}\right], \end{aligned} \tag{5}$$

where T is the current temperature, T_i is the initial temperature, $L_{WV}(T_i)$ and $L_{WH}(T_i)$ are the initial values of the vertical and horizontal window spans $L_{WV}(T)$ and $L_{WH}(T)$, respectively.

The wirelength cost C is estimated using the semiperimeter method, with weighting of critical nets and independent

weighting of horizontal and vertical wiring spans for each net:

$$C = \sum_{\text{nets } i} [x(i)W_H(i) + y(i)W_V(i)], \tag{6}$$

where $x(i)$ and $y(i)$ are the vertical and horizontal spans of the net i's bounding rectangle and $W_H(i)$ and $W_V(i)$ are the weights of the horizontal and vertical wiring spans. When critical nets are assigned a higher weight, the annealing algorithm will try to place the cells interconnected by critical nets close to each other. Independent horizontal and vertical weights give the user the flexibility to prefer connections in one direction over the other.

The acceptance probability is given by $\exp(-\Delta C/T)$, where ΔC (i.e., $E_t - E_q$) is the cost increase and T is the current temperature. When the cost increases, or when the temperature decreases, the acceptance probability (4) gets closer to zero. Thus, the acceptance probability $\exp(-\Delta C/T)$ less than random (0, 1) (a random number between 0 and 1) is high when ΔC is small and when T is large. At each temperature, a fixed number of moves per cell is allowed. This number is specified by the user. The higher the maximum number of moves, the better the results obtained. However, the computation time increases rapidly. There is a recommended number of moves per cell as a function of the problem size in. For example, for a 200-cell and 3000-cell circuit, 100 and 700 moves per cell are recommended, respectively. The annealing process starts at a very high temperature, for example, $T_i = 4,000,000$, to accept most of the moves. The cooling schedule is represented by $T_i + 1 = a(T)$, where $a(T)$ is the cooling rate parameter and is determined experimentally. In the high and low temperature ranges, the temperature is reduced rapidly (e.g., $a(T) \approx 0.8$). However, in the medium temperature range, the temperature is reduced slowly (e.g., $a(T) \approx 0.95$). The algorithm is terminated when T is very small, for example, when $T < 0.1$.

5.3. Floorplannning Based on Simulated Annealing. This section describes an optimal floorplanning on simulated annealing algorithm. Assume that a set of modules are given and each module can be implemented in a finite number of ways, characterized by its width and height. Some of the important issues in the design of a simulated annealing optimization problem are as follows.

(1) the solution space,

(2) the movement from one solution to another,

(3) the cost evaluation function.

The branch cells correspond to the operands and the internal nodes correspond to the operators of the Polish expression. A binary tree can also be constructed from a Polish expression by using a stack as shown in Figure 4. The simulated annealing algorithm moves from one Polish expression to another. A floorplan may have different slicing tree representations. For example, the trees in Figure 4 represent the given floorplan. There is a one-to-one correspondence between a floorplan and its normalized Polish expression. But this leads to a larger solution space and some bias towards floorplans with multitree representations, since they have more chances to be visited in the annealing process. Assume three types of movement are defined to move from one floorplan to another. They operate on the Polish expression representation of the floorplan.

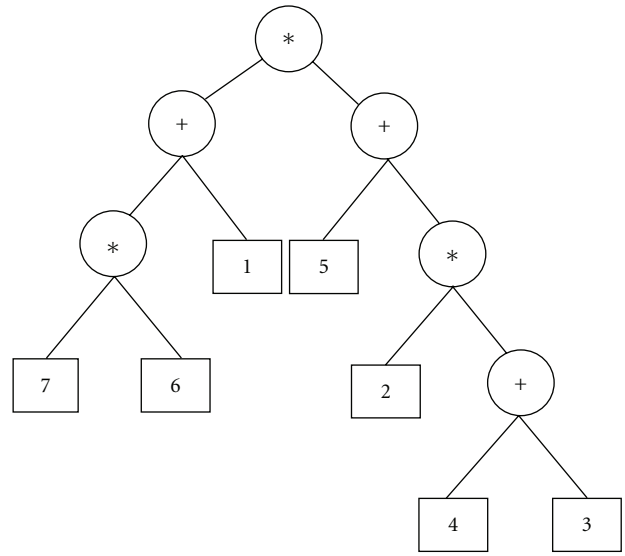


Figure 4: Graph representation of the given floorplan module. *Polish expression:* 76∗1+43+2∗5+∗.

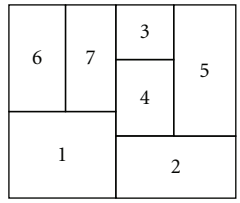


Figure 5: Condition 1 (67∗1+34+2∗5+∗).

Condition 1. Exchange two operands when there are no other operands in between (67∗1+34+2∗5+∗) (see Figure 5).

Condition 2. Complement a series of operators between two operands (67+1∗34∗2+5∗+) (see Figure 6).

Condition 3. Exchange adjacent operand and operator if the resulting expression is a normalized Polish expression (67+43∗+25∗1+∗) (see Figure 7).

It is obvious that the movements will generate only normalized Polish expressions. Thus in effect, the algorithm moves from one floorplan to another. Starting from an arbitrary floorplan, it is possible to visit all the floorplans using the movement. If some floorplans cannot be reached, there is a danger of losing some valid floorplans in the solution. Starting from any floorplan, the modules can move to the floorplan based on the given conditions. The cost function is a function of the floorplan, or equivalently the

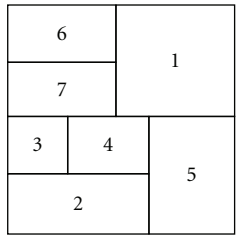


Figure 6: Condition 2 (67+1∗34∗2+5∗+).

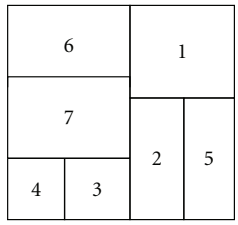


Figure 7: Condition 3 (67+43∗+25∗1+∗).

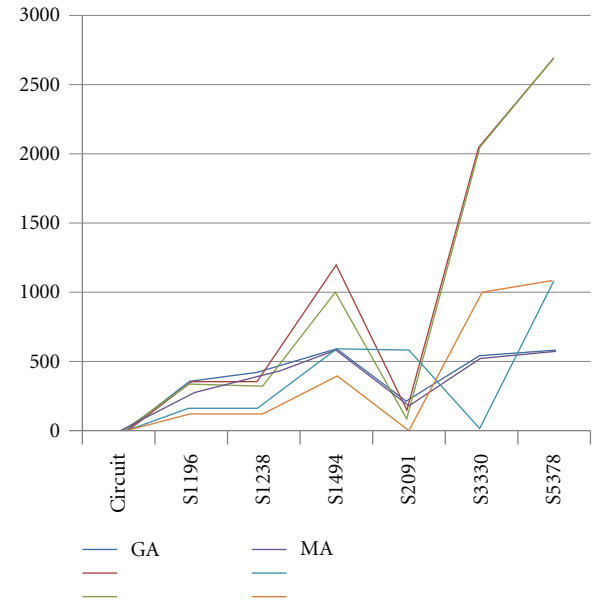


Figure 8: Comparison of MA with GA.

Table 1: Delay comparison of MA with GA for ISCAS89 benchmark.

Circuit	GA			MA		
	Delay(ps)	T (s)	Best (s)	Delay (ps)	T (s)	Best (s)
S1196	396	375	373	301	184	134
S1238	475	397	365	408	187	160
S1494	614	1228	1040	585	616	427
S2091	302	94	32	225	616	16
S3330	571	2096	2074	533	47	994
S5378	587	2687	2686	590	1078	1100

Polish expression. There are two components for the cost function, area, and wire length. The area of a sliceable floorplan can be computed easily using the floorplan sizing algorithm. The wire-length cost can be estimated from the perimeter of the bounding box of each net, assuming that all terminals are located on the center of their module. In general, there is no universally agreed upon method of cost estimation. For simulated annealing, the cost function is best evaluated easily because thousands of solutions need to be examined. Figures 5, 6, and 7 shows a series of movements which lead to a solution.

6. Experimental Results

The memetic algorithm has been implemented in C++ on a Linux installed personal computer. In this work, it compares the performance of a pure genetic algorithm to a memetic technique that combines GA with simple local search techniques like Simulated Annealing. In circuit partitioning, MA can increase the speed of execution time on an average of 45% when compared to simple genetic algorithm see Table 1 and Figure 8. Delay (ps) is the delay of the most critical path. T (s) is the total run time, and Best (s) is the execution time in seconds for reaching the best solution. Thus the objective of wirelength minimization can be achieved by reducing the delay in circuit partitioning. In Floorplanning, wirelength and CPU time are compared with simulated annealing see Table 2. MA can reduce the wirelength on an average of 0.5 mm when compared with Fast Simulated annealing see (Figures 9 and 10).

Table 2: Wirelength comparison of MCNC benchmark for n100, n200, & n300 WITH ASPECT RATIO $R = 1, 2, 3, 4$.

Circuit	Aspect ratio R*	Fast-SA		MA	
		Wire (mm)	Time (sec)	Wire (mm)	Time (sec)
n100	1	33.56	30	32.06	26
	2	35.44	30	34.39	24
	3	35.48	30	34.23	27
	4	36.89	29	32.74	27
n200	1	63.55	175	58.33	150
	2	62.76	173	59.84	156
	3	63.70	180	61.55	156
	4	66.31	176	63.72	171
n300	1	76.05	399	71.00	363
	2	77.60	386	74.22	358
	3	81.67	391	79.56	371
	4	88.58	382	82.18	370
Comparison		1.06	1.11	1.00	1.0

7. Conclusion

By reducing the wirelength, cost of very large-scale integrated chip manufacturing can be reduced. This paper explores the advantage of memetic algorithm which can be proven to 45%

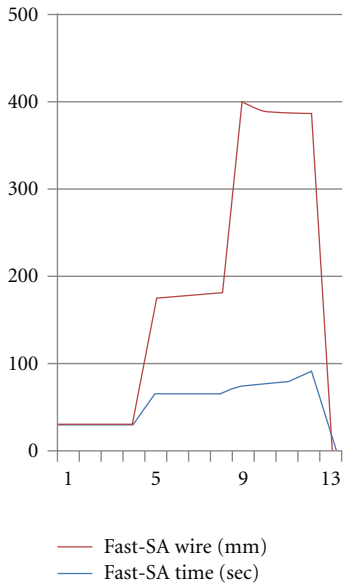


Figure 9: Performance of Fast Simulated Annealing.

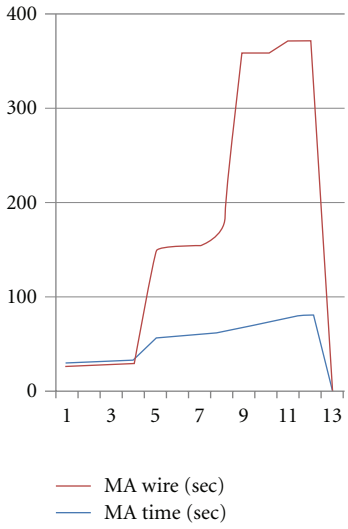


Figure 10: Performance of Memetic Algorithm.

faster than the simple software genetic algorithm by reducing the delay and area in partitioning and floorplanning, respectively, that would indefinitely reduce the wirelength. The implementation of multiobjective in the algorithm enables to get the near optimal solution. After a predetermined number of iteration by GA, local search is applied to few random individual to get the optimal solution by Simulated Annealing. In the future, the performance of the algorithm has to be tested on different benchmarks.

References

[1] N. Sherwani, *Algorithms for VLSI Physical Design Automation*, Kluwer Academic Publishers, Boston, Mass, USA, 1999.

[2] I. Hameem Shanavas, R. K. Gnanamurthy, and T. Stephen Thangaraj, "Hybrid algorithmical approach for VLSI physical design—floorplanning," *International Journal of Recent Trends in Engineering*, vol. 2, no. 4, 2009.

[3] M. Tang and X. Yao, "A memetic algorithm for VLSI floor planning," *IEEE Transactions on Systems, Man, and Cyber Net*, vol. 37, no. 1, 2007.

[4] P. Subbaraj, K. Sivasundari, and P. Siva Kumar, "An effective memetic algorithm for VLSI partitioning problem," in *Proceedings of the International Conference on ICTES*, pp. 667–670, Chennai, India, 2007.

[5] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell System Technical Journal*, pp. 291–307, 1970.

[6] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *Proceedings of the 19th ACM IEEE Design Automation Conference*, pp. 175–181, 1982.

[7] T. W. Manikas and J. T. Cain, "Genetic algorithms vs simulated annealing:a comparison of approaches for solving the circuit partitioning problem," Tech. Rep. 96-101, The University of Pittsburgh, 1996.

[8] Y. Yodtean and P. Chantngarm, "Hybrid algorithm for circuit partitioning," in *Proceedings of IEEE Region 10 Conference*, vol. 4, November 2004.

[9] G.-F. Nan and M.-Q. Li, "Two novel encoding strategies based genetic algorithms for circuit partitioning," in *Proceedings of the 3rd IEEE International Conference on Machine Learning*, vol. 4, pp. 2182–2188, Shangai, 2005.

[10] G. C. Sipakoulis, I. Karafyllidis, and A. Thanailkis, "Genetic partition and placement for VLSI circuits," in *Proceedings of the 6th IEEE Conference on Electronics Circuits and Systems*, vol. 3, pp. 1647–1650, 1999.

[11] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to Theory of NP-Completeness*, Freeman, San Francisco, Calif, USA, 1979.

[12] G. Tumbush and D. Bhatia, "Partitioning under timing and area constraints," in *Proceedings of the International Conference on Computer Design*, pp. 614–620, October 1997.

[13] C. J. Augeri and H. H. Ali, "New graph-based algorithms for partitioning VLSI circuits," in *Proceedings of the IEEE International Symposium on Cirquits and Systems*, pp. V-521–V-524, May 2004.

[14] S. M. Sait, A. H. El-Maleh, and R. H. Al-Abaji, "General iterative heuristics for VLSI multiobjective partitioning," in *Proceedings of the 2003 IEEE International Symposium on Circuits and Systems (ISCAS '03)*, pp. V497–V500, May 2003.

[15] F. M. Johannes, "Partitioning of VLSI circuits and systems," in *Proceedings of the 1996 33rd Annual Design Automation Conference*, pp. 83–87, June 1996.

[16] A. Cincotti, V. Cuttelo, and M. Pavone, "Graph partitioning using genetic algorithms with OPDX," in *Proceedings of IEEE World Congress on Computational Intelligence*, pp. 402–406, 2002.

[17] I. Hameem Shanavas and R. K. Gnanamurthy, "Evolutionary algorithmical approach on VLSI Floorplanning problem," *International Journal of Computer Theory and Engineering*, vol. 1, no. 4, pp. 461–464, 2009.

[18] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: model, taxonomy, and design issues," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 5, pp. 474–488, 2005.

[19] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[20] V. Cerny, "Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm," *Journal of Optimization Theory and Applications*, vol. 45, no. 1, pp. 41–51, 1985.

[21] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[22] L. Ingber, "Very fast simulated re-annealing," *Mathematical and Computer Modelling*, vol. 12, no. 8, pp. 967–973, 1989.

Hindawi Publishing Corporation
VLSI Design
Volume 2011, Article ID 756561, 8 pages
doi:10.1155/2011/756561

Research Article

Weighted Transition Based Reordering, Columnwise Bit Filling, and Difference Vector: A Power-Aware Test Data Compression Method

Usha Mehta,[1] K. S. Dasgupta,[2] and N. M. Devashrayee[1]

[1] *PG (VLSI Design), Nirma University, Ahmedabad 382481, India*
[2] *Indian Institute of Space Science & Technology, Tiruvanthapuram, India*

Correspondence should be addressed to Usha Mehta, usha.mehta@nirmauni.ac.in

Received 29 March 2011; Revised 14 May 2011; Accepted 29 July 2011

Academic Editor: Yangdong Deng

Test data compression is the major issues for the external testing of IP core-based SoC. From a large pool of diverse available techniques for compression, run length-based schemes are most appropriate for IP cores. To improve the compression and to reduce the test power, the test data processing schemes like "don't care bit filling" and "reordering" which do not require any modification in internal structure and do not demand use of any test development tool can be used for SoC-containing IP cores with hidden structure. The proposed "Weighted Transition Based Reordering-Columnwise Bit Filling-Difference Vector (WTR-CBF-DV)" is a modification to earlier proposed "Hamming Distance based Reordering—Columnwise Bit Filling and Difference vector." This new method aims not only at very high compression but also aims at shift in test power reduction without any significant on-chip area overhead. The experiment results on ISCAS89 benchmark circuits show that the test data compression ratio has significantly improved for each case. It is also noteworthy that, in most of the case, this scheme does not involve any extra silicon area overhead compared to the base code with which it used. For few cases, it requires an extra XOR gate and feedback path only. As application of this scheme increases run length of zeroes in test set, as a result, the number of transitions during scan shifting is reduced. This may lower scan power. The proposed scheme can be easily integrated into the existing industrial flow.

1. Introduction

The testing cost and testing power are the two well-known issues of current generation IC testing [1].

The test cost is directly related to test data volume and hence test data transfer time [2]. Test data compression can solve the problem of test cost by reducing the test data transfer time. The dynamic test power plays a major role in overall test power. The switching activity during test has a large contribution in dynamic power and hence in overall test power. The extensive use of IP cores in SoC has further exaggerated the testing problem. Because of the hidden structure of IP cores, the SoCs containing large IP cores can use only those test data compression techniques and switching reduction technique which do not require any modification or insertion in architecture of IP core. These methods should not also demand the use of ATPG, scan insertion, or any such testing tools. They should be capable to use ready-to-use test data coming with IP core for data compression and power reduction. This test data may be partially specified or fully specified. Thus, the current research on IC testing can not be directly applied to the SoC because of the hidden structure of IP core.

So it can be inferred that the test data compression and switching reduction in context of hidden structure of IP core is the current need for SoC testing.

In literature, there are many test data compression techniques like linear decompression-based, broadcast scan-based, and code-based techniques. Considering to suitability to IP core-based SoC, code-based test data compression scheme is more appropriate. From the various code-based test data compression schemes like dictionary codes, constructive codes, statistical codes, and run length-based codes, the run length-based codes can be more suitable to IP cores

because of its simple on-chip decoder and better compression capacity. The do not care bit filling methods and test vector reordering further enhance the test data compression.

The switching activity reduction technique described in literature can be broadly classified in three categories: (1) techniques for built-in self-test, (2) techniques applied as design-for-test, and (3) techniques for external testing. Considering the suitability to IP cores, the techniques for external testing can be further explored. Out of various switching reduction techniques for external testing like low-power ATPG, input control, reordering, and do not care bit filling, the do not care bit filling and reordering are applicable for hidden structure of IP core.

To improve the compression ratio and to reduce the switching in the most famous run length-based data compression method, in this paper, a new scheme based on three techniques: Hamming distance and weighted transition-based reordering (WTR), columnwise bit filling (CBF), and difference vector (DV) is proposed. This scheme is applied to various test set prior to apply a variety of run based codes, and it gives better result in each of the case. The experiment results show that the test data compression ratio is significantly improved. Moreover, this scheme does not require any on-chip silicon area overhead compared to base run length code with which it is used. With the help of weighted transition-based reordering, the total number of transition during scan-in is also reduced. This method may reduce the overall scan power requirement during testing. Further, the proposed scheme can be easily integrated into the existing industrial flow.

The paper is organized as follows: the background for bit filling methods and test vector reordering used for test data compression and test power reduction is covered in Section 2. In Section 3, the Hamming distance-based reordering is explained with a motivational example. Section 3 introduces the concept of weighted transition-based reordering. Section 3.3 include the details of run length code used for compression. Experimental results and performance comparison are presented in Section 6 followed by concluding remarks in Section 8.

2. Background

2.1. Test Data Compression. In 1998, a scheme based on run-length codes that encoded runs of 0 s using fixed-length code words [3] was proposed. The Golomb code [4] encodes runs of 0 s with variable-length code words. The optimization of Golomb is achieved using frequency-directed run-length (FDR) codes [5]. Maleh and Abaji proposed an extension of FDR (EFDR) [6]. The alternating FDR uses runs of 0 s as well as 1 s in alternating fashion [7]. An evolution in alternate run length-based FDR is shifted alternate run length-based FDR [8]. The detailed description on each run length code with one example is given in [9] which also includes the compression, power, and area overhead in case of each run length-based compression code.

2.1.1. Do Not Care Bit Filling. Instead of simply filling all do not care bits with 0 s, if the do not care bits are filled considering the type of run used in particular compression scheme, the better compression can be achieved [10].

2.1.2. Reordering. Stuck at fault-based test patterns can be reordered without any loss of fault coverage. In literature, a number of test vector reordering techniques are proposed for test data compression. The run-based reordering approach [11] is based on reordering the test frames to give the bigger run lengths of 0 s. As this bigger run lengths are than coded with extended FDR, it gives better compression to normal extended FDR. As this approach uses scan frame reordering, it is not suitable to IP cores with hidden structure. The same thing is applicable to [12] which requires a large amount of area overhead to compensate the 2-D reordering. The Hamming distance-based reordering used in [13] is used as the basic scheme in this paper.

2.2. Test Power. For the reduction of switching activity in terms of number of transitions during scan operations, the do not care bit filling and reordering techniques are widely used.

2.3. Ordering Techniques. The greedy algorithm based reordering process using the minimum Hamming distance between them to reduce the scan power [14]. The concept of finding Hamiltonian cycle in a complete weighted graph is used in [15]. In [16], both scan latch reordering with test vector reordering is considered. Another work [17] has also considered the Hamming distance minimization between adjacent vectors to reduce the dynamic power dissipation during testing. Test vector reordering problem as TSP and genetic algorithm (GA) has been used to generate low-power test patterns in [18]. In [19], an evaluation of different heuristic approaches has been done in terms of execution time and quality. In [20], 2-opt heuristic and a GA-based approach with reduction in fault coverage is introduced. Roy et al. has proposed a test vector reordering technique switching activity reduction in case of combinational circuit with AI [21]. An ant colony optimization-based test vector reordering problem for power reduction is described in [22]. The particle swarm approach is used in [23]. There are few other approaches available in literature for test vector reordering, but, while considering the hidden structure of IP cores, these approaches are not found suitable. For capture power reduction in case of IP core-based SoC, the artificial intelligence-based reordering of scan vector is proposed in [24].

2.4. Do Not Care Bit Filling. An automatic test pattern generation (ATPG) scheme for low-power launch-off-capture (LOC) transition test based on do not care bit filling is proposed in [25]. A genetic algorithm based heuristic to fill the do not cares is proposed in [26]. This approach produces an average percentage improvement in dynamic power and leakage power over 0-fill, 1-fill, and minimum transition fill (MT-fill) algorithms for do not care filling. The work in [27] proposed segment-based X-filling to reduce test power and keep the defect coverage. The scan chain configuration tries to cluster the scan flip-flops with common successors into one scan chain, in order to distribute the specified bits

per pattern over a minimum number of chains. Based on the operation of a state machine, [28] elucidates a comprehensive frame for probability-based primary-input dominated X-filling methods to minimize the total weighted switching activity (WSA) during the scan capture operation. The work in [29] describes the effect of do not care filling of the patterns generated via automated test pattern generators, to make the patterns consume lesser power. It presents a tradeoff in the dynamic and static power consumption.

3. Weighted Transition-Based Reordering, Columnwise Bit Filling, and Difference Vector

The earlier proposed Hamming distance-based reordering, column-wise bit filling, and difference vector (HDR-CBF-DV) are taken as the basic scheme for the proposed method. This section includes the introduction to (HDR-CBF-DV) and the proposed modifications.

Before continuing the further explanation, the following two terms need to be defined.

Hamming Distance. The Hamming distance between two scan vectors is equal to the number of corresponding incompatible bits. This definition is similar to Hamming distance with extension of do not-care bits. For example, given two vectors V_1 = (10XX01) and V_2 = (001X11), the distance d(V_1,V_2) is 2 because the first and the fifth corresponding bits in the vectors are incompatible.

Weighted Transition. For a given test data set containing m vectors with n bits each, the weighted transition for each test vector is given by (1), and the total weighted transition during test is given by (2),

$$\text{Weighted Transitions for Scan-In vector } j = \sum_{i=1}^{n} (t(j,i) \oplus t(j,i+1)) * (n-i), \tag{1}$$

$$\text{Total Weighted Transitions during test} = \sum_{j=1}^{m} \sum_{i=1}^{n} (t(j,i) \oplus t(j,i+1)) * (n-i). \tag{2}$$

3.1. Weighted Transition-Based Reordering. If we take each test pattern as a vertex in a complete undirected graph G, and the distance between two patterns as the weight of an edge, then this problem is similar to Hamilton problem, which is NP-hard and solved by various greedy algorithms. The simplest pure greedy algorithm is choosing as the next pattern in a path the one that is closest to the current pattern, provided that it has not been visited yet. It seems that the Hamilton path of G is the solution to our reordering problem.

3.1.1. Selection of First Vector for Reordered Test Set

Hamming Distance-Based Selection. In [13], for the selection of first test pattern, the heuristic that is applied in this scheme is "Hardest Path First." The test pattern with minimum do not cares will be selected as the first test pattern of reorder list. The reason for selecting the test pattern with minimum do not care bits is that there is a minimum flexibility to stuff the bits later. If more than one test pattern have minimum do not care values than any one vector with the minimum do not care bits each will be selected.

Weighted Transition-Based Selection. In WTR-CBF-DV, if there are more than one test vectors with minimum number of do not care bits, each will be evaluated for its weighted transitions, and the vector with minimum weighted transition will be selected as the first test vector of the reordered test set. Further in earlier method, the first vector is kept unfilled until all the test vectors are reordered, but, in the proposed scheme, the selected first vector is MT filled before continuing reordering to make the overall testing and selection of remaining test vectors power aware.

3.1.2. Reordering and Bit Filling of Remaining Test Vector

Hamming Distance-Based Selection. For reordering of the remaining test patterns in [13], the pattern with minimum Hamming distance from first pattern of reordered set will be placed next to first pattern of reordered set. It is decided to take the next vector with minimum Hamming distance because when the further columnwise bit filling and difference vector will be done, this reordered vector sequence will generate maximum zeroes so run length, and hence the compression will increase. In HDR-CBF-DV [13], after completing the reordering of all the vectors, for the second test patterns and onwards, the do not care bit will be replaced with the same value which its upper vector has at the same position. The goal here is to get the maximum zeroes in difference vector.

Weighted Transition-Based Selection. During the reordering process for various circuits, it is found that generally the test set contains more than one test vectors with same Hamming distance. This happens because of the structural behavior of faults. This tie should be broken in favor of power reduction. So in the proposed scheme, while selecting the next vector of the reordered test set, if there are more than one vector with the same Hamming distance from the last selected vector of reordered set, then the weighted transition will be taken into consideration. All these equidistance vectors will be applied columnwise bit filling (explained in Section 5.2) and their weighted transitions are calculated. The vector with the minimum weighted transitions will be selected as next vector of reordered test set. The do not care bits in this vector will be replaced with the same value which its upper vector has at the same position.

3.2. Difference Vector. The next step is to take the difference vector of two consecutive vectors in reordered set. This will further increase the numbers of zeroes and hence data compression. Any run length code can be used to compress the difference vector sequence T_{diff}. Let $T_{\text{Dpt}} = \{t_1; t_2; \ldots; t_n\}$ be the reordered test set. T_{diff} is defined as follows: $T_{\text{diff}} = \{d_1; d_2; \ldots; d_n\} = \{t_1; t_1 \oplus t_2; t_2 \oplus t_3; \ldots; t_{n-1} \oplus t_n\}$, where a bit-wise exclusive or operation is carried out between

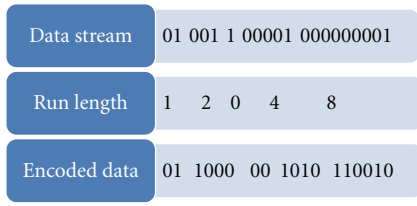


Figure 1: Example of frequency-directed run length coding.

Table 1: Frequency directed run length code.

Run length	k	Group prefix	Tail	Code word
0	1	0	0	00
1			1	01
2	2	10	00	1000
3			01	1001
4			10	1010
5			11	1011
6	3	110	000	110000
7			001	110001
-		---	--	------

patterns t_i and t_{i+1}. The successive test patterns in a test sequence often differ in only a small number of bits. Therefore, T_{diff} contains few 1 s, and it can be efficiently compressed using the FDR code.

3.3. Run Length Code for Compression. For the proposed method, the test data will be first preprocessed by WTR-CBF-DV scheme, and then the frequency-directed run length code (FDR) [5, 30] will be applied to preprocess data. The equation of % compression used very frequently in this paper is as follows:

$$\text{Test data compression in percentage} = \frac{\#\text{ of original bits} - \#\text{ of compressed bits}}{\#\text{ of original bits}} \times 100\%. \quad (3)$$

The examples in Figure 1 and Table 1 demonstrate this coding style.

4. Algorithm for WTR-CBF-DV

(1) Consider a digital circuit with n scan flip-flops, p inputs, and q outputs. The ATPG generated partially specified test set with m scan-in test vectors, and each of n bits is the input to this algorithm.

(2) Find test vector with minimum number of do not care bits in the given test set.

(3) If there are more than one vector with minimum do not care bits, then

(i) apply MT fill to each vector,

(ii) calculate weighted transition for each vector,

(iii) select the MT-filled vector with minimum WT as first vector of reordered set.

(4) Find the Hamming distance of remaining each vectors from the first vector of reordered set.

(5) Select the vector with minimum Hamming distance as next vector.

(6) If there are more than one vector with minimum Hamming distance, then

(i) apply columnwise bit fill to each vector, that is, replace the do not care bit of the vector with the same position bit value of last selected vector,

(ii) calculate weighted transition for each vector, and

(iii) select the columnwise bit filled vector with minimum WT as next vector of reordered set.

(7) Repeat step (6) until all the vectors are reordered.

(8) Apply difference vector mechanism.

(i) First vector of reordered set is kept unchanged.

(ii) From the second vector onward, if the same position bits in last vector and current vector are same, replace the bit with 0 else 1.

(9) Apply frequency-directed run length code.

5. Motivation Example

Considering the following test data, for example,

test vector

1	X	1	0	0	X	X	0	1	X	0	0	X	1
1	1	1	X	0	X	0	X	1	0	1	0	X	X
1	0	1	1	0	X	0	0	X	X	X	0	1	0
0	X	X	0	X	X	1	0	X	X	X	0	X	X
1	0	1	X	1	X	1	X	1	0	X	0	0	X
1	1	1	1	0	X	0	0	X	X	X	X	0	0

5.1. Selection of First Test Vector of Reordered Set. In the above test data, vector V_3 has the minimum number of do not care bits, that is, 4. Now this vector is minimum transition (MT) filled as shown below:

test vector

V_3	1	0	1	1	0	0	0	0	0	0	0	0	1	0

its corresponding weighted transition as per (1) is 38.

5.2. Reordering Remaining Test Vector with Columnwise Bit Filling. After placing the V_3 at first place and V_1 shifted to position of vector 3, the test set after first line selected and filled is as below:

test vectors after first vector reordered

R_1	1	0	1	1	0	0	0	0	0	0	0	0	1	0
V_2	1	**1**	1	X	0	X	0	X	**1**	0	**1**	0	X	X
V_3	1	X	1	**0**	0	X	X	0	**1**	X	0	0	X	**1**
V_4	**0**	X	X	**0**	X	X	**1**	0	X	X	X	0	X	X
V_5	1	0	1	X	**1**	X	**1**	X	**1**	0	X	0	**0**	X
V_6	1	1	**1**	1	0	X	0	0	X	X	X	X	**0**	0

Now the Hamming distance of remaining each test vector V_2, V_3, V_4, V_5, and V_6 from first reordered vector R_1 is 3, 3, 3, 4, and 2 as described in definition of Hamming distance and emphasized in above test set with bold letter. So V_6 is selected as next vector of reordered set. After placing V_6 as R_2, the columnwise bit filling is done.

Partially reordered test vectors

R_1	1	0	1	1	0	0	0	0	0	0	0	0	1	0
R_2	1	1	1	1	0	**0**	0	0	**0**	**0**	**0**	**0**	0	0
V_3	1	X	1	0	0	X	X	0	1	X	0	0	X	1
V_4	0	X	X	0	X	X	1	0	X	X	X	0	X	X
V_5	1	0	1	X	1	X	1	X	1	0	X	0	0	X
V_6	1	1	1	X	0	X	0	X	1	0	1	0	X	X

Now the Hamming distance of V_3, V_4, V_5, and V_6 from R_2 is calculated as 3, 3, 4, and 2. So V_6 is selected as R_3.

Partially reordered test vectors

R_1	1	0	1	1	0	0	0	0	0	0	0	0	1	0
R_2	1	1	1	1	0	0	0	0	0	0	0	0	0	0
R_3	1	1	1	1	0	0	0	0	1	0	1	0	0	0
V_4	0	X	X	0	X	X	1	0	X	X	X	0	X	X
V_5	1	0	1	X	1	X	1	X	1	0	X	0	0	X
V_6	1	X	1	0	0	X	X	0	1	X	0	0	X	1

Now the remaining all three V_4, V_5, and V_6 vectors have equal Hamming distance 3 from R_3. So each vector will be tried for its weighted transition as if columnwise bit filling is applied to it.

Test vector

R_3	1	1	1	1	0	0	0	0	1	0	1	0	0	0
V_4	0	**1**	**1**	0	**0**	**0**	1	0	**1**	**0**	**1**	0	**0**	**0**

Applying (1), the weighted transition is equal to 57 for this case.

Test vector

R_3	1	1	1	1	0	0	0	0	1	0	1	0	0	0
V_5	1	0	1	**1**	1	**0**	1	**0**	1	0	**1**	0	0	**0**

Test vector

R_3	1	1	1	1	0	0	0	0	1	0	1	0	0	0
V_6	1	**1**	1	0	0	**0**	**0**	0	1	**0**	0	0	**0**	1

The same way the possible weighted transitions for V_5 and V_6 are 67 and 23, respectively, as shown above. So V_6 is selected as the next test vector of reordered set. Repeating the reordering process until the last vector is reordered, the final reordered set is as shown bellow:

reordered test vectors

R_1	1	0	1	1	0	0	0	0	0	0	0	0	1	0
R_2	1	1	1	1	0	0	0	0	0	0	0	0	0	0
R_3	1	1	1	1	0	0	0	0	1	0	1	0	0	0
R_4	1	1	1	0	0	0	0	0	1	0	0	0	0	1
R_5	0	1	1	0	0	0	1	0	1	0	0	0	0	1
R_6	1	0	1	0	1	0	1	0	1	0	0	0	0	1

5.3. Difference Vector. The next step is to take the difference vector of two consecutive vectors in reordered set to increase the numbers of zeroes and hence data compression. The difference vector set is as shown below:

difference test vectors

R_1	1	0	1	1	0	0	0	0	0	0	0	0	1	0
D_2	0	1	0	0	0	0	0	0	0	0	0	0	1	0
D_3	0	0	0	0	0	0	0	0	1	0	1	0	0	0
D_4	0	0	0	1	0	0	0	0	0	0	1	0	0	1
D_5	1	0	0	0	0	0	1	0	0	0	0	0	0	0
D_6	1	1	0	0	1	0	0	0	0	0	0	0	0	0

5.4. Run Length Coding. The difference vector set applied the frequency directed run length coding as described in Section 3.3.

6. Comparison

In Table 2, the comparison of % compression, peak power, and average power for various test data processing scheme with FDR coding applied to test data of motivational example is shown in Table 2. Here the column 2 in Table 2 shows the results when test data is applied with MT filling and FDR coding. The peak power and average power is minimum in this case, but the compression is negative. Column 3 is for test data where do not care bits are filled with 0 s and without reordering, the difference vectors are created, and FDR is applied. The columns 4 and 5 represents the results of HDR-CBF-DV and WTR-CBF-DV. As it is seen from these results, the % compression is maximum in case of HDR-CBF-DV and WTR-CBF-DV, while average power is comparable in case of difference vector only, HDR-CBF-DV, and WTR-CBF-DV.

6.1. Experiment Results. For the WTR-CBF-DV, the working model is developed using MATLAB7.0 language and then for extensive experimental work, the C language is used. The experiments are conducted on a workstation with an Intel 2 GHz Core2Duo CPU T5750 with 3 GB of memory. The six largest ISCAS89 full-scan circuits have been considered for this experiment. For all ISCAS89 circuits, the test sets (with do not care) obtained from the Mintest ATPG program are used. Tables 3, 4, and 5 show the comparison of % compression, average power, and peak power for various ISCAS circuits' test data with FDR coding when test data applied the following processing prior to FDR coding.

(A) Do not care bits are MT filled, but no reordering is applied.

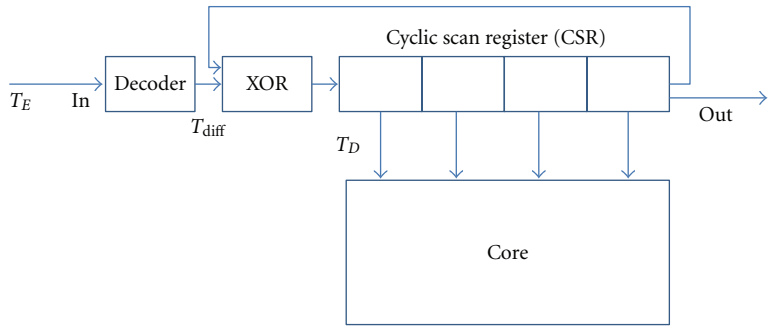


Figure 2: On-chip decoder for WTR-CBF-DV.

Table 2: Comparison of test data processing methods.

	MT Fill	Diff. Vector	HDR-CBF-DV	WTR-CBF-DV
% Compression	−2.381	7.1429	16.6667	16.6667
Peak power	38	81	82	82
Average power	23.8333	36.8333	42.1667	38.6667

Table 3: Comparison of % compression for various test data processing methods.

ISCAS circuit	Minimum transition fill	0 Filling + XOR [30]	Run based bit fill maximum limit	HDR-CBF-DV	2-D Reordering	WTR-CBF-DV
s5378	−12.31	48.02	52.36	62.33	59.66	62.15
s9234	−20.67	43.59	47.80	61.06	61.35	63.31
s13207	6.16	81.30	83.65	87.47	88.22	88.04
s15850	−17.91	66.22	68.18	72.84	73.96	73.38
s38417	−20.39	43.26	54.5	66.18	65.13	66.38
s38584	−8.90	60.91	62.49	64.79	66.08	65.21

Table 4: Comparison of average power for various test data processing methods.

ISCAS circuit	Minimum transition fill	0 Filling + XOR [30]	Run based bit fill maximum limit	HDR-CBF-DV	2-D Reordering	WTR-CBF-DV
s5378	3433	3526	3526	11133	7934	10344
s9234	3958	4022	4022	14382	13329	13492
s13207	7735	7887	7887	113890	78856	103400
s15850	13514	13659	13659	82421	71015	64275
s38417	117540	118080	118080	452860	486000	443030
s38584	85656	86305	86305	410240	423260	329110

Table 5: Comparison of peak power for various test data processing methods.

ISCAS circuit	Minimum transition fill	0 Filling + XOR [30]	Run based bit fill maximum limit	HDR-CBF-DV	2-D Reordering	WTR-CBF-DV
s5378	11519	12085	12085	13327	11769	12822
s9234	14092	15395	15395	17828	16106	17169
s13207	94879	110129	110129	128638	95541	125392
s15850	70875	84360	84360	96084	98903	96452
s38417	437884	514716	514716	644262	633561	660096
s38584	481158	530464	530464	550037	532809	551602

(B) Do not care bits are filled on the basis of run type, but no reordering is applied.

(C) Do not care bits are filled with 0 s, and difference vector is applied [30].

(D) HDR-CBF-DV applied.

(E) 2-D reordering is applied.

(F) WTR-CBF-DV applied.

7. On-Chip Decoder

Any of test data compression methods needs an on-chip decompressor, which loads compressed data from automatic test equipment (ATE) and restores the original test data. The decompressed test data will be transmitted to design under test. The WTR-CBF-DV is a test data processing method applied in conjunction with FDR coding. The same FDR decoder described in [5, 30] is used here. Figure 2 describes the decoder described in [30]. As in this approach, the difference vector is already used, the proposed WTR-CBF-DV does not require any extra on-chip area overhead.

8. Conclusion

In this paper, a scheme comprising of Hamming distance and weighted transition-based reordering (WTR), columnwise bit filling (CBF), and difference vector (DV) for test data compression is proposed. This scheme is applied to preprocess the test data before applying the FDR compression method. The proposed test data processing scheme improves the % compression compared to earlier methods described in literature. Moreover, this method increases the compression beyond the limit of maximum possible compression for run based bit filled data. The peak power and average power is a tradeoff with % compression, but still it is controlled using weighted transition-based reordering. The proposed scheme demands no extra on-chip area overhead compared to earlier methods in literature.

References

[1] M. Hirech, "Test cost and test power conflicts: EDA perspective," in *Proceedings of the 28th IEEE VLSI Test Symposium (VTS '10)*, p. 126, Santa Cruz, Calif, USA, January 2010.

[2] N. A. Touba, "Survey of test vector compression techniques," *IEEE Design and Test of Computers*, vol. 23, no. 4, pp. 294–303, 2006.

[3] A. Jas and N. A. Touba, "Test vector decompression via cyclical scan chains and its application to testing core-based designs," in *Proceedings of the IEEE International Test Conference (ITC '98)*, pp. 458–464, October 1998.

[4] A. Chandra and K. Chakrabarty, "Test data compression for system-on-a-chip using Golomb codes," in *Proceedings of the 18th IEEE VLSI Test Symposium (VTS '00)*, pp. 113–120, May 2000.

[5] A. Chandra and K. Chakrabarty, "Test data compression and test resource partitioning for system-on-a-chip using frequency-directed run-length (FDR) codes," *IEEE Transactions on Computers*, vol. 52, no. 8, pp. 1076–1088, 2003.

[6] A. H. El-Maleh and R. H. Al-Abaji, "Extended frequency-directed run-length code with improved application to system-on-a-chip test data compression," in *Proceedings of the 9th IEEE International Conference on Electronics, Circuits and Systems (ICECS '02)*, vol. 2, pp. 449–452, September 2002.

[7] A. Chandra and K. Chakrabarty, "Reduction of SOC test data volume, scan power and testing time using alternating run-length codes," in *Proceedings of the 39th Conference on Design Automation*, pp. 673–678, June 2002.

[8] S. Hellebrand and A. Wrtenberger, "Alternating run length coding : a technique for improved test data compression," in *Proceedings of the 3rd IEEE International Workshop on Test Resource Partitioning*, Baltimore, MD, USA, October 2002.

[9] U. S. Mehta, K. S. Dasgupta, and N. M. Devashrayee, "Run-length-based test data compression techniques: how far from entropy and power bounds?—a survey," *VLSI Design*, vol. 2010, Article ID 670476, 9 pages, 2010.

[10] U. S. Mehta, K. S. Dasgupta, and N. M. Devashrayee, "Combining unspecified test data bit filling methods and run length based codes to estimate compression, power and area overhead," in *Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems (APCCAS '10)*, pp. 40–43, Kuala-Lumpur, Malaysia, December, 2010.

[11] H. Fang, T. Chenguang, and C. Xu, "RunBasedReordering: a novel approach for test data compression and scan power," in *Proceedings of the IEEE International Conference on Asia and South Pacific Design Automation (ASP-DAC '07)*, pp. 732–737, Yokohama, Japan, January 2007.

[12] U. S. Mehta, K. S. Dasgupta, and N. M. Devashrayee, "Hamming distance based 2-D reordering with power efficient don't care bit filling: optimizing the test data compression method," in *Proceedings of the 9th International Symposium on System-on-Chip (SoC '10)*, pp. 1–7, Tampare, Finland, September 2010.

[13] U. S. Mehta, K. S. Dasgupta, and N. M. Devashrayee, "Hamming distance based reordering and column wise bit stuffing with difference vector: a better scheme for test data compression with run length based codes," in *Proceedings of the 23rd International Conference on VLSI Design (VLSID '10)*, pp. 33–38, Bangalore, India, 2010.

[14] P. Girard, C. Landrault, S. Pravossoudovitch, and D. Severac, "Reducing power consumption during test application by test vector ordering," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '98)*, pp. 296–299, June 1998.

[15] K. Roy, R. K. Roy, and A. Chatterjee, "Stress testing of combinational VLSI circuits using existing test sets," in *Proceedings of the IEEE International Symposium on VLSI Technology, Systems, and Applications (ISVLSI '95)*, pp. 93–98, June 1995.

[16] I. P. V. Dabholkar, S. Chakravarty, and S. Reddy, "Techniques for minimizing power dissipation in scan and combinational circuits during test application," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 12, pp. 1325–1333, 1998.

[17] H. N. J. M. P. Flores, J. Costa, H. Neto, J. Monteiro, and J. Marques-Silva, "Assignment and reordering of incompletely specified pattern sequences targetting minimum power dissipation," in *Proceedings of the 12th International Conference on VLSI Design*, pp. 37–41, January 1999.

[18] S. Chattopadhyay and N. Choudhary, "Genetic algorithm based approach for low power combinational circuit testing," in *Proceedings of the 16th International Conference on VLSI Design*, pp. 552–559, January 2003.

[19] H. Hashempour and F. Lombardi, "Evaluation of heuristic techniques for test vector ordering," in *Proceedings of the ACM*

Great lakes Symposium on VLSI (GLSVLSI '04), pp. 96–99, January 2004.

[20] A. S. D. Whitley, A. Sokolov, and Y. Malaiya, "Dynamic power minimization during combinational circuit testing as a traveling salesman problem," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1088–1095, September 2005.

[21] S. Roy, I. S. Gupta, and A. Pal, "Artificial intelligence approach to test vector reordering for dynamic power reduction during VLSI testing," in *Proceedings of the IEEE Region 10 Conference (TENCON '08)*, pp. 1–6, Hyderabad, India, November 2008.

[22] Y. L. J. Wang, J. Shao, and Y. Huang, "Using ant colony optimization for test vector reordering," in *proceedings of the IEEE Symposium on Industrial Electronics and Applications (ISIEA '09)*, pp. 52–55, Kuala Lumpur, Malaysia, October 2009.

[23] S. K. Kumar, S. Kaundinya, and S. Chattopadhyay, "Particle swarm optimization based vector reordering for low power testing," in *proceedings of the 2nd International Conference on Computing, Communication and Networking Technologies (ICCCNT '10)*, pp. 1–5, Karur, India, July 2010.

[24] U. S. Mehta, K. S. Dasgupta, and N. M. Devashrayee, "Artificial intelligence based scan vector reordering for capture power minimization," in *Proceedings of the IEEE International Symposium on VLSI (ISVLSI '11)*, June 2011.

[25] S. J. Wang, Y. T. Chen, and K. S. M. Li, "Low capture power test generation for launch-off-capture transition test based on don't-care filling," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '07)*, pp. 3683–3686, May 2007.

[26] S. Kundu and S. Chattopadhyay, "Efficient don't care filling for power reduction during testing," in *Proceedings of the International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom '09)*, pp. 319–323, October 2009.

[27] Z. Chen, J. Feng, D. Xiang, and B. Yin, "Scan chain configuration based X filling for low power and high quality testing," *IET Journal On Computers and Digital Techniques*, vol. 4, no. 1, pp. 1–13, 2009.

[28] J. L. Yang and Q. Xu, "State-sensitive X-filling scheme for scan capture power reduction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 7, Article ID 4544872, pp. 1338–1343, 2008.

[29] T. K. Maiti and S. Chattopadhyay, "Don't care filling for power minimization in VLSI circuit testing," in *Proceedings of THE IEEE International Symposium on Circuits and Systems (ISCAS '08)*, pp. 2637–2640, Seattle, Wash, May 2008.

[30] A. Chandra and K. Chakrabarty, "Frequency-directed run-length (FDR) codes with application to system-on-a-chip test data compression," in *Proceedings of the 19th IEEE VLSI Test Symposium*, pp. 42–47, May 2001.

Hindawi Publishing Corporation
VLSI Design
Volume 2011, Article ID 328640, 9 pages
doi:10.1155/2011/328640

Research Article

Efficient Resource Sharing Architecture for Multistandard Communication System

T. Suresh and K. L. Shunmuganathan

Department of CSE, R.M.K Engineering College, Anna University, Chennai 600025, India

Correspondence should be addressed to T. Suresh, fiosuresh@yahoo.co.in

Received 15 October 2010; Revised 8 February 2011; Accepted 29 March 2011

Academic Editor: Yangdong Deng

The Fourth Generation (4G) network is expected to serve mobile subscribers under dynamic network conditions and offer any type service: anytime, anywhere, and anyhow. Two such technologies that can respond to the above said services are Wideband Code Division Multiple Access (WCDMA) and Orthogonal Frequency Division Multiplexing (OFDM). The main contribution of this paper is to propose a dedicated hardware module which can reconfigure itself either to the OFDM Wireless LAN or WCDMA standard. In this paper, Fast Fourier Transform (FFT) algorithm is implemented for OFDM standard, and rake receiver is implemented for WCDMA standard. Initially efficient implementations of these two algorithms are tested separately and identified the resources utilized by them. Then the new hardware architecture, which configures to any one of these two standards on demand, is proposed. This architecture efficiently shares the resources needed for these two standards. The proposed architecture is simulated using ModelSimSE v6.5 and mapped onto a virtex 5 FPGA device (xc5v1x30ff324) using the tool Xilinx ISE 9.2i, and the results are compared with the standard approach. These results show that the proposed hardware architecture utilizes less number of resources compared to the conventional Reconfigurable Receiver Architecture System.

1. Introduction

Next generation wireless and mobile networks are all IP-based heterogeneous networks that allow users to use any system at anytime, anywhere, anyhow, and always-on connectivity in a seamless [1] manner. Users carrying an integrated open terminal can use a wide range of applications provided by multiple wireless networks and access to various air interface standards. The continuous evolution of wireless networks and the emerging variety of different heterogeneous, wireless network platforms with different properties require integration into a single platform [2]. The handoff mechanism allows a network connection on a mobile node to operate over multiple wireless access networks in a way that is completely transparent to end user applications. But there is no single system that is good enough to support all the wireless communication technologies. Instead of putting efforts in developing new radio interfaces and technologies for 4G [3] systems, we believe establishing 4G systems that integrate existing and newly developed wireless systems into one open platform is a more feasible option.

This has led to an increased interest in the design of reconfigurable architecture. The idea of the reconfigurable architecture is that it should be possible to alter the functionality of a mobile device at run time by simply reusing the same hardware for different wireless technologies and ultimately for users to connect to any system that happens to be available at any given time and place. This means that the same hardware should be able to handle many different modulation types as well as different demands on data rate and mobility.

2. Reconfigurable Architecture

There are several options available to implement the base-band signal processing section of wireless technologies. The possible options are the following.

(1) Multimode ASICs where device functionality can be switched according to the mode of operation.

(2) DSP-based implementation.

(3) Programmable logic-based implementation (using FPGA or PLD).

(4) Reconfigurable hardware [4–6].

Multistandard wireless communication applications demand high-computing power [7], flexibility, and scalability. An Application-Specific Integrated Circuit (ASIC) solution would meet the high computing power requirement, but is inflexible [8] and the long design cycle of ASICs makes them unsuitable for prototyping. On the other hand, general purpose microprocessors or Digital Signal Processing (DSP) chips are flexible, but often fail to provide sufficient computing power. Various processor architectures such as Very Long Instruction Word (VLIW) architecture [9] or vector processors have been introduced to increase the computing power, but the computing power is still insufficient or the architectures are too power hungry or expensive in silicon. Field Programmable Gate Arrays (FPGAs) [10] signal processing platforms are now widely being accepted in base_station designs. However, low power and form factor requirements have prevented their use in handsets. Reconfigurable hardware for Digital BaseBand (DBB) [11] processing is rapidly gaining acceptance in multimode handheld devices that support multiple standards. In Reconfigurable Hardware tasks that are nonoverlapping either in time domain or space domain can be mapped onto the same reconfigurable logic. Tasks that are required initially can be configured in the beginning. When another task is required, the configuration to load it can then be triggered. For example, in a typical smartphone environment, different wireless technologies, such as GSM, WCDMA, WLAN, and WiMax in the future, have to be supported [9]. However, it is not likely that these wireless technologies will be used at the same time. Therefore, it is possible to put them into reconfigurable logic and dynamically load the one that is needed. In this paper we present the design methodology for reconfigurable baseband signal processor architecture that supports WCDMA and OFDM wireless LAN standards.

In [12] the flexibility of the MONTIUM architecture was verified by implementing HiperLAN/2 receiver as well as a Bluetooth receiver on the same architecture. In [13], a broadband mobile transceiver and a hardware architecture which can be configured to any cyclic-prefix- (CP-) based system reconfigurable architecture for multicarrier-based CDMA systems is proposed. Reconfigurable Modem (RM) Architecture targeting 3G multistandard wireless communication system was proposed in [7]. This architecture targeted two 3G wireless standards WCDMA and CDMA 2000 and the design objectives are scalability, low power dissipation, and low circuit complexity. It is seen that though different functions can be reconfigured on a reconfigurable hardware, the major challenge is to have an efficient system configuration and management function which will initiate and control the reconfiguration as per the different application requirements.

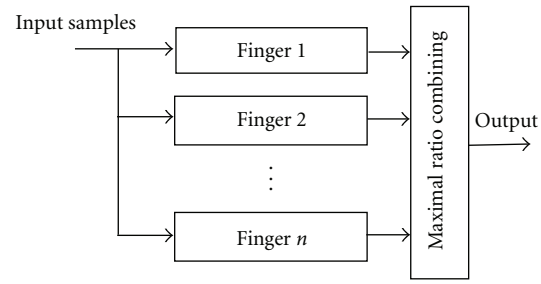


Figure 1: Components of the rake Receiver.

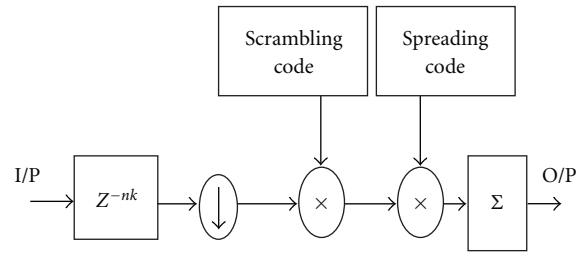


Figure 2: Schematic diagram of a rake finger.

3. Wideband Code Division Multiple Access (WCDMA)

Wideband Code Division Multiple Access (WCDMA) is a third-generation mobile wireless technology that is based on an ITU standard derived from CDMA [14–16] technology. WCDMA can support mobile/portable voice, images, data, and video communications up to 2 Mbps (local area access) or 384 kbps (wide area access). One of the more demanding functions on the WCDMA Baseband receiver module is the Rake Receiver [17]. A Rake Receiver allows each arriving multipath signal to be individually demodulated and then combined to produce a stronger and more accurate signal. Figure 1 shows the components of a Rake Receiver: a set of fingers and a combiner block.

The actual number of Rake fingers is not specified by WCDMA specifications but typically 4–8 fingers are employed. The demodulation is performed in the Rake fingers by correlating the received signal with a spreading code over a period corresponding to the spreading factor. Each Rake finger consists of two multipliers for applying the spreading and scrambling codes and an accumulator of length scrambling factor, input sample delay memory block, downsampling block, scrambling and spreading code generators. Figure 2 shows the schematic diagram of a Rake finger.

After the demodulation, maximal ratio combining is applied to the symbol dumps from the fingers. In maximal ratio combining, the phases of the symbols are aligned and their amplitudes are weighted according to the complex tap coefficients acquired by the complex channel estimator. After the combining, decision of the transmitted symbol is made and the resulting bit stream is de interleaved and decoded. The downlink scrambling code generator is a set

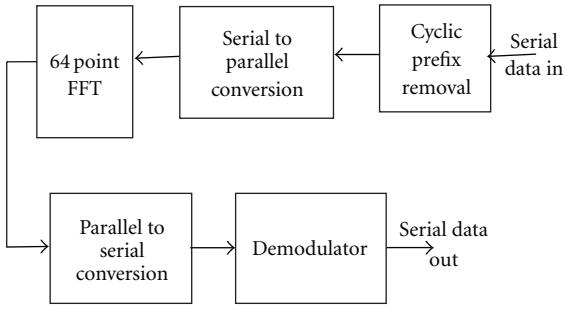


Figure 3: Simplified OFDM receiver block diagram.

of shift registers easy to implement in hardware or software. The number of scrambling code generators required for the RAKE Receiver depends on the receiver architecture chosen as well as on the number of simultaneous downlink cells supported and the number of fingers.

4. Orthogonal Frequency Division Multiplexing (OFDM)

Broadband WLAN standards (IEEE 802.11a/g, IEEE 802.16, Digital Audio Broadcast (DAB), and HIPERLAN/2) are based on the Orthogonal Frequency Division Multiplexing (OFDM) [18–20] modulation scheme because of its superior performance in various multipath environments, such as indoor wireless networks and metropolitan area network. OFDM can efficiently deal with multipath fading, channel delay spread, enhanced channel capacity, adaptively modifies modulation density and robust to narrowband interference. Figure 3 shows the basic block diagram for OFDM Receiver module. At the receiver, the received RF signal is down-converted to baseband frequency, digitized, and fed to the baseband section for further processing. Here the data is first cleaved off the cyclic prefix, followed by Fast Fourier Transform (FFT) [21, 22] operation and demodulated to obtain the received data bits. The key kernel in an OFDM receiver is the FFT processor. FFT-based processing is used to convert the signals from time domain to frequency domain and vice versa in OFDM modulation. The idea of using FFT instead of DFT is that the computation can be made faster where this is the main criteria for implementation. In direct computation of DFT the computation for **N**-point DFT will be calculated one by one for each point. But for FFT, the computation is done simultaneously and this method helps to save a lot of time. In WLAN standards it works with 64 carriers at a sampling rate of 20 MHz, so a 64-point FFT processor is required. 64 time domain samples represent the useful data part of the OFDM symbol that has to be demodulated.

The equations for FFT algorithm is written as

$$X(k) = \sum_{n=0}^{N\ 1} x(n) W_N^{nk}. \tag{1}$$

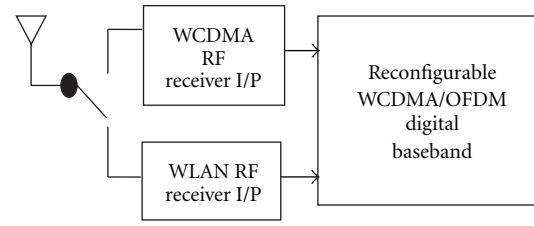


Figure 4: Block diagram of reconfigurable receiver system.

The quantity W_N^{nk} (called Twiddle Factor) is defined as

$$W_N^{nk} = e^{-j2\pi nk/N}. \tag{2}$$

This factor is calculated and put in a table in order to make the computation easier and can run simultaneously. The Twiddle Factor table is depending on the number of points used. During the computation of FFT, the factor does not need to be recalculated, since it can refer to the Twiddle factor table and thus it saves time.

5. Reconfigurable Receiver Architecture

Figure 4 shows the block diagram of reconfigurable receiver system. This receiver system is able to reconfigure itself to the WCDMA or OFDM Wireless LAN (WLAN) standard.

The proposed architecture comprises functional blocks, which in the form of reusable [13], reconfigurable [23–26] functional blocks for use in implementing different algorithms necessary for OFDM and WCDMA standards. One or more reusable functional blocks, can be configured to implement a process including multiplication, addition, subtraction, and accumulation. By accommodating the above-mentioned capabilities, the architecture should be configured to support WCDMA and WLAN OFDM Standards. For example, Fast Fourier Transform (FFT) (basic butterfly function) for WLAN OFDM and rake receiver algorithms. (multiply and accumulate select function) for WCDMA are implemented in the architecture as shown in Figure 5. Figure 6 shows the proposed reconfigurable architecture. The architecture includes the following:

(1) processing elements (PE) and their interconnects,

(2) a controlling block to control the functions of all the blocks,

(3) I/O block configured to receive preprocessed data, deliver processed data out, and determine the required functionality: a configurable I/O block that connects the chip with the outside world;

(4) memory block: configured to store the compiled software instructions and to cache and buffer data.

This architecture allows for transformation of the chip from WCDMA chip to WLAN Wi-Fi chip on demand wherein new algorithms can be accommodated on chip in real time via different control sets.

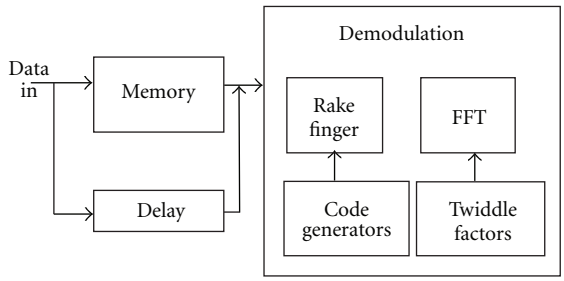


Figure 5: Functional block of reconfigurable receiver algorithms.

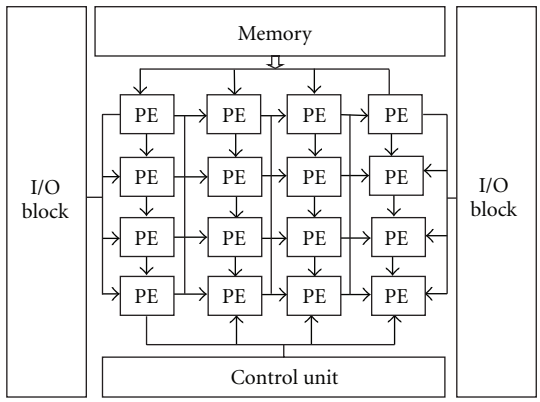


Figure 6: The proposed reconfigurable architecture.

5.1. Rake Finger Implementation. In WCDMA receivers, the demodulation is performed in the Rake fingers by correlating the received signal with a spreading code over a period corresponding to the spreading factor. The output of the ith Rake finger can be expressed as

$$O_i(n) = \sum_{i=0}^{L_{sf}-1} C_s(i + nL_{sf})R(i + nL_{sf}), \tag{3}$$

where C_s is the combined spreading and scrambling code and R is the received signal and both are complex numbers [7]. Since the scrambling and spreading codes are always of ± 1, the multiplication and addition of each correlation stage are simplified. So (3) is simplified to

$$(R_r + jR_i)(C_{sr} - jC_{si}) = (R_rC_{sr} + R_iC_{si}) + j(R_iC_{sr} - R_rC_{si}). \tag{4}$$

If the value +1 is represented as logic "0" and the value −1 is represented as logic "1", (4) is simplified as follows:

$$= \begin{cases} R_r + R_i + j(R_i - R_r), & \text{when } C_{sr} = 0,\ C_{si} = 0, \\ R_r - R_i + j(R_i + R_r), & \text{when } C_{sr} = 0,\ C_{si} = 1, \\ -(R_r - R_i) - j(R_i - R_r), & \text{when } C_{sr} = 1,\ C_{si} = 0, \\ -(R_r + R_i) - j(R_i - R_r), & \text{when } C_{sr} = 1,\ C_{si} = 1. \end{cases} \tag{5}$$

Since the code input is binary valued, the complex multiplication in the correlations is simplified to one real addition/subtraction and one imaginary addition/subtraction.

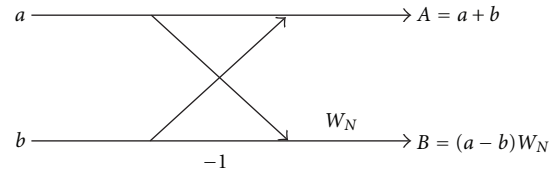


Figure 7: 2 Point butterfly structure.

Selection of addition or subtraction is done with the help of multiplexer. So the total resources required to implement Rake Receiver using (5) are two adders, two subtractors, and one multiplexer.

5.2. FFT Implementation. In OFDM, the demodulation is performed by applying 64-point FFT. The twiddle factor is calculated and put in a table in order to make the computation easier and can run simultaneously. The Twiddle Factor table is depending on the number of points used. During the computation of FFT, this factor does not need to be recalculated since it can refer to the twiddle factor table, and thus it saves time. Figure 7 shows the 2-point Butterfly structure [27] where multiplication is performed with the twiddle factor after subtraction.

Multiplication is certainly the most vital operation in communication processing, and its implementation in an integrated circuit component requires large hardware resources and significantly affects the size, performance, and power consumption of a system [28]. So an efficient way of multiplier reduction in FFT processing is done as follows. Consider the problem of computing the product of two complex numbers R and W

$$\begin{aligned} X = RW &= \left(R_r + jR_i\right)\left(W_r + jW_i\right) \\ &= (R_rW_r - R_iW_i) + j(R_rW_i + R_iW_r). \end{aligned} \tag{6}$$

From (6), the direct architectural implementation requires total of four multiplications and one real subtraction and one imaginary addition to compute the complex product. However, by applying the Strength Reduction Transformation we can reformulate (6) as

$$\begin{aligned} X_r &= (R_r - R_i)W_i + R_r(W_r - W_i), \\ X_i &= (R_r - R_i)W_i + R_i(W_r + W_i). \end{aligned} \tag{7}$$

As can be seen from (7), by using the Strength Reduction Transformation the total number of multiplications is reduced to only three. This however is at the expense of having two additional subtractors and one adder.

5.3. Design Procedure and Mapping Tool. Figure 8 shows the proposed tool chain for application mapping which is similar to the conventional ones, however it has been customized for this design. This design typically starts with the application codes written in VHDL. This abstract design is optimized to fit into the FPGAs available logic

through a series of steps. The basic functionality of this flow is to generate configuration bit streams and an executable code for the reconfigurable architecture. Initially DFGs (Data Flow Graph) are extracted manually, however automatic generation is possible for some application. The extracted DFGs possess a large number of operations which necessitates a sophisticated mapping tool to locate DFGs onto the proposed reconfigurable architecture. Applying architectural constraints, the DFGs are mapped onto the architecture through placing DFG nodes on the PE array and routing interconnection as well as assigning input/output nodes to the proper I/O ports. Our placement algorithm uses the well-known conventional method for placing DFG nodes on the PEs. Routing process establishes connections between the source and target PEs of Reconfigurable Array, by means of routing resources. The main objective of this process is to reduce the connection length between PEs. Placement is not limited to only one row of PE array but, the nodes can be distributed over the entire PE array. Placement and routing procedures ought to be iterated until a valid mapping satisfying the architectural constraints are generated. The total number of required PE in the PE array, the number of PEs in each row, the number of rows, the number of input/output ports, and the suitable routing resources are the essential architectural constraints for this design process. Configuration's bit_stream associated with each node of DFGs can be generated after completion of the mapping stage. For logic and layout synthesis the Xilinx ISE toolset can be utilized. The configuration controller (processor embedded in the FPGA) controls the execution of the partitioned basic block by enabling the proper configurations (bit_streams) on the FPGA according to application's data and control flow. Data memories embedded in the FPGA can be used for storing the input and output values. The virtex 5 FPGA device is used as the design platform since this architecture provides more resources and an improved logic count with respect to the previous family. A set of configuration registers defines the state of this configuration logic at a given moment in time. Configurations are the memory where the bit_stream file has direct access. Actual configuration data are first written by the bit_stream into these registers and then copied by the configuration logic on the configuration SRAMs.

5.4. Reconfiguration Overhead. To be practical, RTR systems must insure that the time spent performing reconfiguration is negligible with respect to the time spent performing applications. If the system is reconfigured too frequently, more time is spent reconfiguring than performing applications. The time wasted in performing reconfiguration is called reconfiguration overhead. The reconfiguration overhead in dynamically reconfigurable systems is a major problem that affects the total execution time. The three methods (beyond the scope of this paper) used to minimize the reconfiguration overhead are configuration reuse, configuration prefetch, and configuration replacement. Configuration reuse allows different applications to use the same configured hardware task so that the number of configurations is reduced.

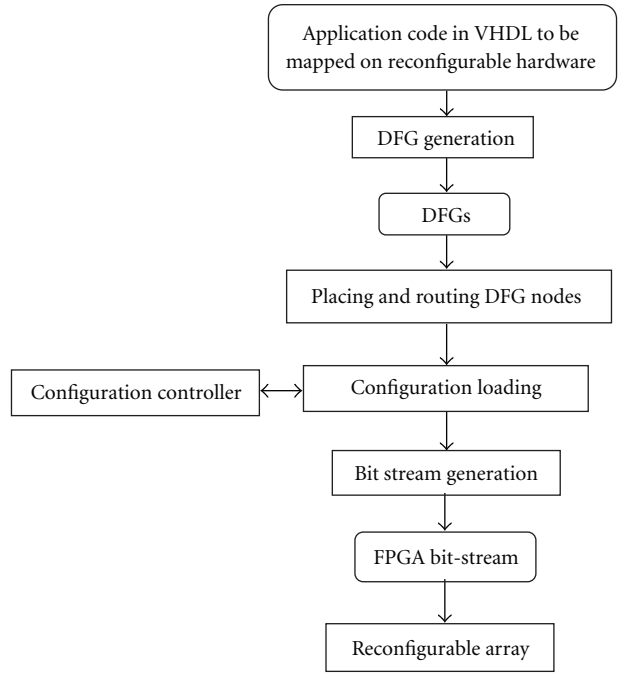


Figure 8: Application mapping tool chain.

Configuration prefetch allows hardware tasks to be configured well before it is required so that the configuration latency is hidden or overlapped with other hardware or software executions. The replacement technique increases the possibilities of reusing those subtasks that are more critical for the system performance.

5.5. Debugging the Reconfigurable Array. After mapping the application in reconfigurable array, this FPGA design can be debugged using simulators. However, we verified this design directly, by downloading them into an FPGA and then testing them in a system. This testing provides a stronger form of functional verification than simulation. Simulations for debugging purposes were carried out with ModelSim SE v6.5 from Mentor Graphics. The Xilinx 9.2i tool set was used to compile the VHDL code and then the design was placed and routed.

6. Processing Element

Figure 9 shows the Processing Element (PE) and its resources required for the implementation of FFT in WLAN OFDM and Figure 10 shows the Processing Element (PE) and its resources required for the implementation of Rake finger in WCDMA. It is shown that the two adders and subtractors (red coloured) are shared by both the standards.

So the proposed reconfigurable architecture consists of processing units, their computational elements are shared by both the Rake Receiver operation of WCDMA and FFT operation of OFDM. The processing units perform the multiply accumulate operation in the Rake mode as described in Section 5.1 and butterfly operations in the FFT mode as described in Section 5.2. The computational

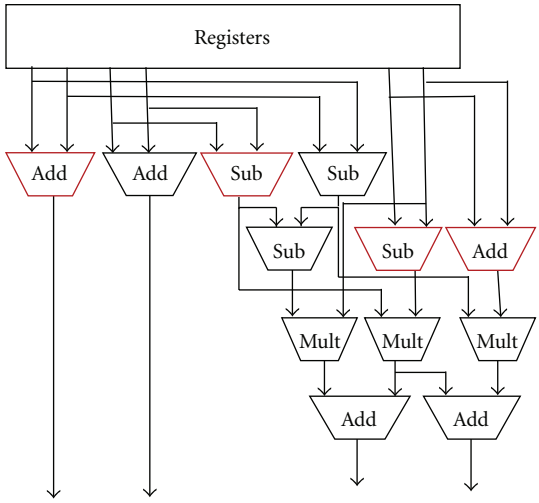


Figure 9: PE and its resources of WLAN OFDM.

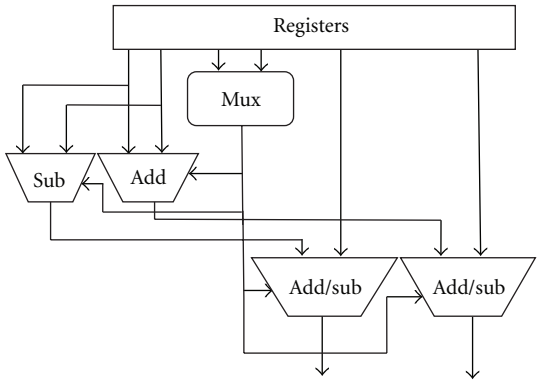


Figure 10: PE and its resources of WCDMA.

resources required by the proposed architecture are 5 adders, 4 subtractors, and 3 multipliers and multiplexers.

7. Results and Discussions

The proposed architecture described in Sections 5 and 6 was simulated using ModelSimSE v6.5, coded in VHDL and mapped [29, 30] onto a Virtex 5 FPGA device (xc5v1x30ff324) with speed grade (−3) using the tool Xilinx ISE 9.2i and synthesized. The metrics extracted from Xilinx ISE after synthesis and implementation are the followings: (1) the number of used Slice LUTs (Look Up Tables) and (2) gate count. Table 1 shows the implementation results of our optimized Strength Reduction Transformation Technique of FFT algorithm. These results are obtained using the design scheme explained in Section 6. The comparison results as shown in Figure 11 shows that our proposed Strength Reduction Transformation Technique of FFT algorithm has significant improvements in terms of area saving compared to the standard implementation. On average, our optimized approach shows an improvement about 15.93% in terms of number of gates used and 19.27% in terms of number of Slice LUTs used, as illustrated in Table 4.

Table 1: Resource utilization of conventional FFT and FFT with strength reduction transformation technique.

Resources utilized	Conventional FFT	FFT with strength reduction transformation technique
Number of slice LUTs	1588 out of 19200	1282 out of 19200
Gate count	14796	12440

Table 2: Resource utilization of conventional rake finger and multiplier-less rake finger.

Resources utilized	Conventional rake finger	Rake finger (multiplier-less)
Number of slice LUTs	1426 out of 19200	128 out of 19200
Gate count	13898	1328

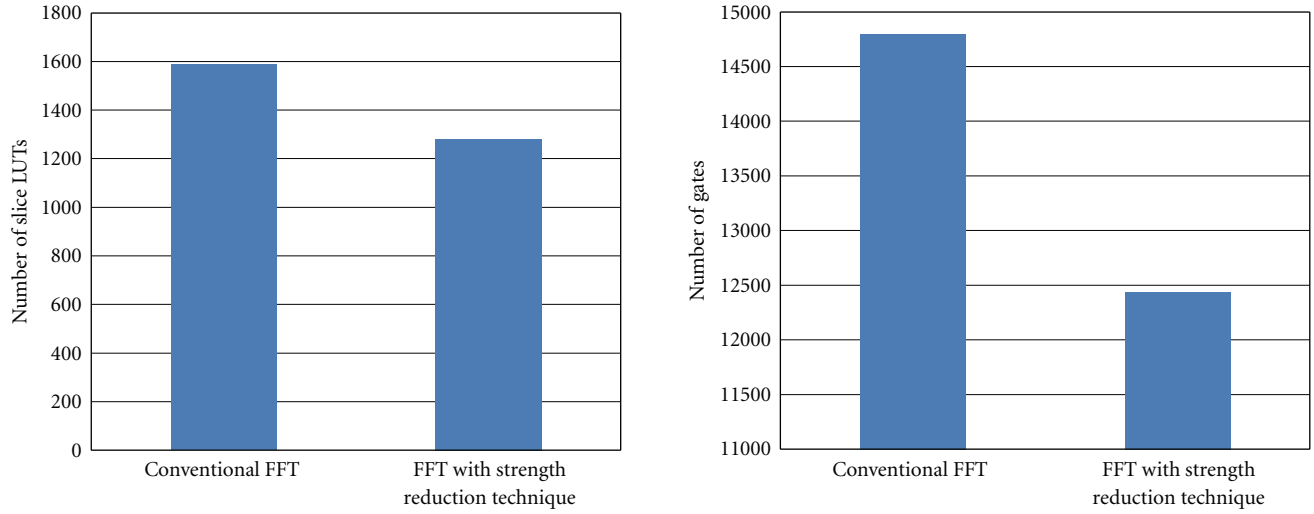
Table 3: Resource utilization of reconfigurable architecture without and with resource sharing.

Resources utilized	Reconfigurable architecture without resource sharing	Reconfigurable architecture with resource sharing
Number of slice LUTs	3014 out of 19200	1290 out of 19200
Gate count	28694	12540

Table 2 and Figure 12 illustrate the comparison of the results of our optimized multiplier-less Rake finger implementation and the Conventional Rake finger. From this result it is clear that our proposed approach outperformed the conventional Rake finger in terms of the number of Slices LUTs used and the number of gates utilized. On average, our optimized approach shows an improvement of about 91.03% and 90.45% in terms of number of Slice LUTs used and the number of ates utilized, as illustrated in Table 4.

Finally the implementation results of the proposed reconfigurable architecture with Resource sharing are compared with Reconfigurable Architecture without resource sharing techniques. Table 3 and Figure 13 show the comparison results of the proposed Architecture (Reconfigurable Architecture with Resource sharing) and the Reconfigurable Architecture without Resource sharing in terms of the number of Slice LUTs used and the number of gates utilized. The average reductions for the parameters are 57.2% and 56.3%, respectively, as shown in Table 4. From this result it is clear that our proposed Resource sharing Reconfigurable Architecture is better than the standard Reconfigurable Architecture.

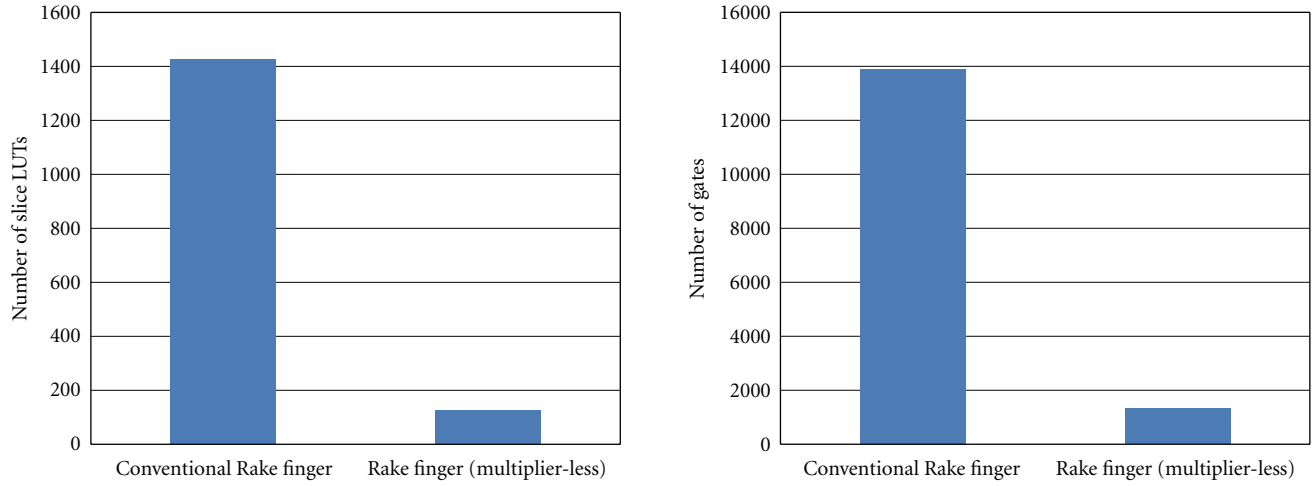
From the results presented earlier it seems that there is a significant reduction in large number of computational resources which forms the proposed architecture which is more efficient than the conventional architecture. These efficient realizations require fewer FPGA resources, so not



(a) Comparison of the Slice LUTs used

(b) Comparison of the number of gates utilized

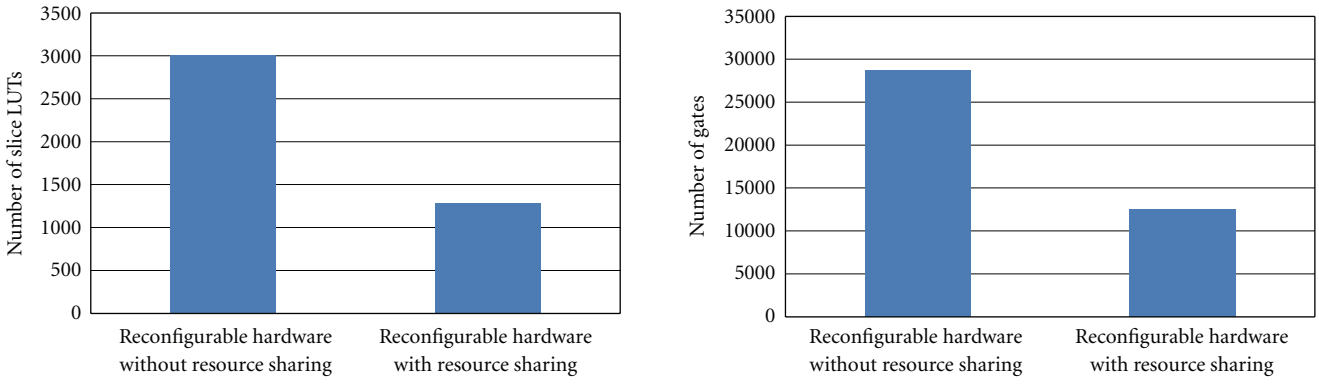
Figure 11: Comparison results of resources utilized by conventional FFT and FFT with strength reduction transformation technique.



(a) Comparison of the Slice LUTs used

(b) Comparison of the number of gates utilized

Figure 12: Comparison of the percentage of resources utilized by conventional rake finger and rake finger (multiplier-less).



(a) Comparison of the Slice LUTs used

(b) Comparison of the number of gates utilized

Figure 13: Comparison of the percentage of resources utilized by reconfigurable architecture without resource sharing and with resource sharing.

Table 4: Comparison of results of the conventional architecture and the proposed method targeting Virtex 5 FPGA.

Proposed method	Gate reduction	Slice LUTs saving
FFT with strength reduction transformation technique	15.93%	19.27%
Rake finger (multiplier-less)	90.45%	91.03%
Reconfigurable hardware with resource sharing	56.3%	57.2%

only reduce the area, but reduce the total power dissipation also.

8. Conclusion

An architecture which can reconfigure itself to wireless LAN OFDM and WCDMA standards, was presented in this paper. While configuring these two standards, it was also presented to implement FFT operation for OFDM and Rake Receiver functioning for WCDMA efficiently. To lower the number of multipliers in FFT and eliminate the multipliers in Rake Receiver, we adopted Strength Reduction Transformation technique and multiplier-less technique. The proposed architecture was simulated using ModelSimSE v6.5 and mapped onto a Xilinx Virtex 5 FPGA device and synthesis report was generated. Substantial improvements in terms of number of Slice LUTs used and the number of gates utilized were achieved. Comparison results showed that the proposed architecture can reduce large number of FPGA resources, enhance efficiency of the hardware architecture, and significantly reduce area and power consumption. Moreover, the proposed architecture can be improved to reconfigure to various other advanced wireless standards.

References

[1] R. Hirschfeld, W. kellerer, C. Prehofer, and H. Berndt, "An integrated system concept for future generation mobile communication," in *Proceedings of the Eurescom Summit on Powerful Networks for Profitable Services (EURESCOM '02)*, Germany, October 2002.

[2] L. M. Gavrilovska and V. M. Atanasovski, "Interoperability in future wireless communications. Systems: a roadmap to 4G," *Microwave Review*, vol. 13, no. 1, 2007.

[3] V. Gazis, N. Houses, A. Alonistioti, and L. Merakos, "Generic system architecture for 4G mobile communications," in *Proceedings of the 57th IEEE Semiannual Vehicular Technology Conference (VTC '03)*, pp. 1512–1516, April 2003.

[4] S. Hauck and A. Detlon, *Reconfigurable Computing the theory and Practice of FPGA-Based Computation*, Elsevier, New York, NY, USA, 2008.

[5] M. Gokhale and P. S. Graham, *Reconfigurable Computing Accelerating Computation with Field-Programmable Gate Arrays*, Springer, London, UK, 2005.

[6] R. Tessier and W. Burleson, "Reconfigurable computing for digital signal processing: a survey," *Journal of VLSI Signal Processing*, vol. 28, no. 1-2, pp. 7–27, 2001.

[7] J. S. Lee and D. S. Ha, "FleXilicon: a reconfigurable architecture for multimedia and wireless communications," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '06)*, pp. 4375–4378, May 2006.

[8] J. Kim, D. S. Ha, and J. H. Reed, "A new reconfigurable modem architecture for 3g multi-standard wireless communication systems," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '05)*, pp. 1051–1054, May 2005.

[9] A. S. Y. Poon, "An energy-efficient reconfigurable baseband processor for wireless communications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 3, pp. 319–327, 2007.

[10] R. David, D. Chillet, S. Pillement, and O. Sentieys, *A Compilation Framework for a Dynamically Reconfigurable Architecture*, vol. 2438, Springer, London, UK, 2002.

[11] L. Harju and J. Nurmi, "A programmable baseband receiver platform for WCDMA/OFDM mobile terminals," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC '05)*, pp. 33–38, New Orleans, La,USA, March 2005.

[12] G. K. Rauwerda, P. M. Heysters, and G. J.M. Smit, "Mapping wireless communication algorithms onto a reconfigurable architecture," *Journal of Supercomputing*, vol. 30, no. 3, pp. 263–282, 2004.

[13] Y. C. Liang, S. Naveen, S. K. Pilakkat, and A. K. Marath, "Reconfigurable signal processing and hardware architecture for broadband wireless communications," *EURASIP Journal on Wireless Communications and Networking*, vol. 2005, no. 3, pp. 323–332, 2005.

[14] J. Becker and M. Glesner, "A parallel dynamically reconfigurable architecture designed for flexible application-tailored hardware/software systems in future mobile communication," *Journal of Supercomputing*, vol. 19, no. 1, pp. 105–127, 2001.

[15] Y. Guo, D. McCain, J. R. Cavallaro, and A. Takach, "Rapid industrial prototyping and SoC design of 3G/4G wireless systems using an HLS methodology," *EURASIP Journal on Embedded Systems*, vol. 2006, Article ID 14952, pp. 1–25, 2006.

[16] R. Sivasubramanian and A. Varadhan, "An efficient implementation of IS-95A CDMA transceivers through FPGA," *International Journal on Digital Signal Processing*, vol. 6, no. 1, pp. 23–30, 2006.

[17] A. Alsolaim, J. Becker, M. Glesner, and J. Starzyk, "Architecture and Application of a dynamically reconfigurable hardware array for future mobile communication systems," in *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, August 2002.

[18] J. S. Park, H.-J. Jung, and V. K. Prasasnna, "Efficient FPGA-based implementation of the MIMO-OFDM physical layer," in *Proceedings of the World Congress in Computer Science Computer Engineering and Applied Computing (WORLDCMP '06)*, World Academy of Science, Las Vegas, Nev, USA, June 2006.

[19] J. Helmschmidt, E. Schuler, P. Rao, S. Rossi, S. di Matteo, and R. Bonitz, "Reconfigurable signal processing in wireless terminals," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE '03)*, March 2003.

[20] C. Ebeling, C. Fisher, G. Xing, M. Shen, and H. Liu, "Implementing an OFDM receiver on the RaPiD reconfigurable architecture," *IEEE Transactions on Computers*, vol. 53, no. 11, pp. 1436–1448, 2004.

[21] M. J. Myjak and J. G. Delgado-Frias, "A medium-grain reconfigurable architecture for DSP: VLSI design, benchmark mapping, and performance," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 1, pp. 14–23, 2008.

[22] B. Zhou, Y. Peng, and D. Hwang, "Pipeline FFT architectures optimized for FPGAs," *International Journal of Reconfigurable Computing*, vol. 2009, Article ID 219140, 9 pages, 2009.

[23] S. Hauck, T. W. Fry, M. M. Hosler, and J. P. Kao, "The chimaera reconfigurable functional unit," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 2, pp. 206–217, 2004.

[24] H. Parizi, A. Niktash, A. Kamalizad, and N. Bagherzadeh, "A reconfigurable architecture for wireless communication systems," in *Proceedings of the 3rd International Conference on Information Technology: New Generations (ITNG '06)*, pp. 250–254, April 2006.

[25] R. Hartenstein, "Coarse grain reconfigurable architectures," in *Proceedings of the Conference on Asia South Pacific Design Automation*, pp. 564–570, January 2001.

[26] Y. Qu, K. Tiensyrjä, J. P. Soininen, and J. Nurmi, "Design flow instantiation for run-time reconfigurable systems: a case study," *EURASIP Journal on Embedded Systems*, vol. 2008, no. 1, Article ID 856756, 2008.

[27] P. Heysters, G. Smit, and E. Molenkamp, "A flexible and energy-efficient coarse-grained reconfigurable architecture for mobile systems," *Journal of Supercomputing*, vol. 26, no. 3, pp. 283–308, 2003.

[28] H. Hinkelmann, P. Zipf, J. Li, G. Liu, and M. Glesner, "On the design of reconfigurable multipliers for integer and Galois field multiplication," *Microprocessors and Microsystems*, vol. 33, no. 1, pp. 2–12, 2009.

[29] F. Hannig, H. Dutta, and J. Teich, "Regular mapping for coarse-grained reconfigurable architectures," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. V-57–V-60, May 2004.

[30] M. D. Galanis, G. Dimitroulakos, and C. E. Goutis, "Speedups and energy reductions from mapping DSP applications on an embedded reconfigurable system," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 12, pp. 1362–1366, 2007.

Hindawi Publishing Corporation
VLSI Design
Volume 2011, Article ID 892310, 9 pages
doi:10.1155/2011/892310

Research Article

Efficient Congestion Mitigation Using Congestion-Aware Steiner Trees and Network Coding Topologies

M. A. R. Chaudhry, Z. Asad, A. Sprintson, and J. Hu

Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843, USA

Correspondence should be addressed to M. A. R. Chaudhry, masadch@tamu.edu

Received 26 December 2010; Accepted 5 February 2011

Academic Editor: Zhuo Li

In the advent of smaller devices, a significant increase in the density of on-chip components has raised congestion and overflow as critical issues in VLSI physical design automation. In this paper, we present novel techniques for reducing congestion and minimizing overflows. Our methods are based on ripping up nets that go through the congested areas and replacing them with *congestion-aware* topologies. Our contributions can be summarized as follows. First, we present several efficient algorithms for finding *congestion-aware* Steiner trees that is, trees that avoid congested areas of the chip. Next, we show that the novel technique of *network coding* can lead to further improvements in routability, reduction of congestion, and overflow avoidance. Finally, we present an algorithm for identifying efficient congestion-aware network coding topologies. We evaluate the performance of the proposed algorithms through extensive simulations.

1. Introduction

In almost any VLSI design flow, global routing is an essential stage that determines the signal interconnections. Therefore, the capability of the global router may significantly affect the design turn-around time. Moreover, the results of the global routing stage impact many circuit characteristics, such as power, timing, area, and signal integrity. Global routing poses major challenges in terms of the efficient computation of quality routes. In fact, most of the global routing problems, even special cases, tend to be NP complete [1, 2].

In the advent of smaller devices, a significant increase in the density of on-chip components results in a larger number of nets that need to be routed, which, together with more stringent routing constraints, results in increasing congestion and overflow. In this paper, we propose novel techniques for congestion avoidance and overflow reduction. Our methods are designed for the rip-up-and-reroute phase of the global routing stage. At this stage, all the nets have already been routed using a standard prerouting technique however some of the nets need to be rerouted due to high congestion and overflow. Our methods are based on ripping up nets that go through congested areas and replacing them with congestion-aware topologies. The proposed techniques facilitate even distribution of the routing load along the available routing areas. We propose efficient algorithms for finding congestion-aware Steiner trees that favor uncongested routes. In addition, we use the novel technique of *network coding* for further reduction of congestion and overflow avoidance.

1.1. Congestion-Aware Steiner Trees. The major goal of congestion-aware Steiner tree routing is to find a tree that connects the required set of nodes (pins of a net) while avoiding congested areas with a minimum penalty in terms of the total wirelength. In addition, the running time of the routing algorithm should scale well with the growing number of nets. These requirements pose several challenges in terms of the algorithm design. The first challenge is to select a cost function that adequately captures the local congestion conditions at the edges of the routing graph. Next, the algorithm should find a minimum cost tree within acceptable running time. Since finding a Steiner tree is an NP-complete problem, the algorithm needs to use an approximation scheme or employ a heuristic approach. Finally, the proposed algorithm should ensure that the overall performance of the rip-up-and-reroute phase is satisfactory

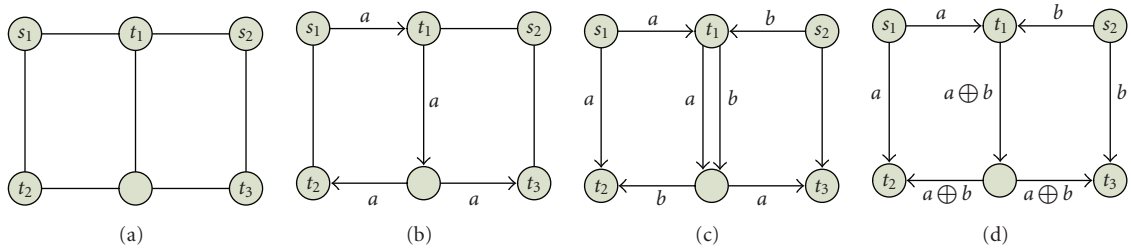


FIGURE 1: (a) The underlying routing graph with two source nodes s_1, and s_2 and three terminals t_1, t_2, t_3. (b) A rectilinear Steiner tree that connects source s_1 to all terminals. (c) Two rectilinear Steiner trees that connect sources s_1 and s_2 to all terminals. (d) A network coding solution.

in terms of congestion mitigation and overflow reduction. In this paper, we evaluate several cost functions which take into account various factors such as wire density, overflow, and congestion history. We propose several efficient algorithms for Steiner tree routing and compare their performance. Our algorithms are based on known approximations for the Steiner tree problem, heuristic methods, and artificial intelligence techniques.

1.2. Network Coding. The basic idea of the *network coding* technique [3] is to enable the intermediate nodes to generate new signals by combining the signals arriving over their incoming wires. This is in contrast to the standard approach, in which each node can only forward or duplicate the incoming signals.

For example, consider a routing instance depicted in Figure 1(a). In this example, we need to route two nets, one connecting source s_1 with terminals t_1, t_2, and t_3 and the other connecting source s_2 with the same set of terminals. The underlying routing graph is represented by a grid as shown in Figure 1(a). Suppose that due to congestion, each edge of this graph has a residual capacity of one unit, that is, each edge can accommodate only a single wire. It is easy to verify that using traditional Steiner tree routing, only one net can be routed without an overflow. For example, Figure 1(b) shows a possible routing of a net that connects s_1 with terminals t_1, t_2, and t_3. In contrast, Figure 1(c) shows that routing of both nets results in an overflow. In this example, two nets transmit different signals, a and b, over separate Steiner trees. Figure 1(d) shows that the network coding approach allows to route both nets without overflows. With this approach, the terminal t_1 creates a new signal, $a \oplus b$, which is delivered to terminals t_2 and t_3, while the signals a and b are delivered to terminals t_2, and t_3 directly. It is easy to verify that each terminal can decode the two original symbols a and b.

The network coding technique offers two distinct advantages. First, it has a potential of solving difficult cases that cannot be solved using traditional routing. For example, for the routing instance shown in Figure 1, the traditional routing results in an overflow of value 1, whereas with the network coding technique there are no overflows. Second, the network coding technique can decrease the total wirelength. For example, in the routing instance shown in Figure 1 the total wirelength for the traditional routing solution is 8, whereas for the network coding solution, the total wirelength is 7.

1.3. Previous Work. In the past decades, researchers have strived to improve the performance of global routing algorithms (see, e.g., [4–6] and references therein). To handle the complexity of large-scale global routing, multilevel routing techniques are proposed in [7, 8]. Recently proposed BoxRouter [9, 10] is based on progressive integer linear Programming (ILP) and rip-up-and-reroute techniques. A fast routing method is presented in [11]. Reference [12] proposes an approach based on the Lagrangian multiplier technique. An effective edge shifting technique is presented in [13]. Most of these previous works adopt the rip-up-and-reroute strategy. However, they usually reroute one path (i.e., a 2-pin connection) at a time. In contrast, our method reroutes entire multipin nets. We also propose to use network coding techniques to further reduce congestion and eliminate overflows.

1.4. Contribution. The paper makes the following contributions. First, we propose several algorithms for finding efficient congestion-aware Steiner trees. Second, we show that the novel technique of network coding can lead to further improvements in routability, reduction of congestion, and overflow minimization. Finally, we provide an algorithm for identifying efficient congestion-aware network coding topologies. We evaluate the performance of the proposed algorithms through extensive simulations.

2. Model

In this paper, we adopt the most commonly used global routing model. The routing topology is represented by a grid graph $G(V, E)$, where each node $v \in V$ corresponds to a global routing cell (GCell) [10, 12] and each routing edge $e \in E$ corresponds to a boundary between two adjacent GCells. A set $S = \{n_1, n_2, \ldots, n_{|S|}\}$ of nets are to be routed on this graph. Each net $n_i \in S$ connects a source node s_i with terminal nodes $T_i = \{t_1, t_2, \ldots, t_{|T_i|}\}$. If there is a wire connection between two adjacent GCells, the wire must cross their boundary and utilize the corresponding routing edge. Each routing edge $e \in E$ has a certain routing capacity $c(e)$ which determines the number of wires that can pass through

this edge. We denote by $\eta(e)$ the number of wires that are currently using edge e.

2.1. Global Routing Metric. The goal of a global router is to minimize congestion. Some of the important metrics for a global router are defined as follows.

(i) *Overflow.* For each edge $e \in E$, the overflow $\text{ov}(e)$ of e is defined as

$$\text{ov}(e) = \begin{cases} \eta(e) - c(e), & \text{if } \eta(e) > c(e), \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The maximum overflow $\text{ov}_{\max}$ is defined as

$$\text{ov}_{\max} = \max_{e \in E} \text{ov}(e). \quad (2)$$

Total overflow ov_{tot} is defined as

$$\text{ov}_{\text{tot}} = \sum_{e \in E} \text{ov}(e). \quad (3)$$

(ii) *Wirelength.* Total wirelength wlen is defined as

$$\text{wlen} = \sum_{e \in E} \eta(e). \quad (4)$$

(iii) *Density.* The density $d(e)$ of edge $e \in E$ is defined as

$$d(e) = \frac{\eta(e)}{c(e)}. \quad (5)$$

2.2. Cost Functions. Our algorithms associate each edge $e \in E$ in the graph with a cost function $\rho(e)$ which captures its congestion and overflow. The cost of the tree is defined as the sum of the costs of all of its edges. Our goal is to identify trees that go through congested areas and replace them by Steiner trees or network coding topologies that go through areas with low congestion.

In this work, we consider several cost functions, described below.

Polynomial Cost Function. We propose a cost assignment function, where the cost of an overflowed edge is a polynomial function of the sum of its density and overflow. Formally, our proposed cost function is defined as follows:

$$\rho(e) = (d(e) + \text{ov}(e))^{\alpha}, \quad (6)$$

where α is a constant which determines the relative penalty for the congested edges.

Exponential Cost Function. We use the cost assignment function proposed by [12]. With this cost assignment, the cost of an edge is an exponential function of its density

$$\rho(e) = \begin{cases} \text{Exp}(\beta \cdot (d(e) - 1)) & \text{if } d(e) > 1, \\ d(e) & \text{otherwise,} \end{cases} \quad (7)$$

where β is a constant which determines the penalty for overflowed edges.

History-Based Cost Function. This cost function assigns cost to an edge based on its congestion history [10, 12]. Specifically, each edge is associated with a parameter h_e that specifies the number of times the edge has been overflowed during the previous iterations of the algorithm. That is, each time the edge with an overflow is used, the parameter h_e is incremented by one. Then, the modified cost $\rho'(e)$ of the edge is defined as follows:

$$\rho'(e) = 1 + h_e \cdot \rho(e). \quad (8)$$

Here, $\rho(e)$ is either the polynomial cost function (6) or exponential cost function (7). If the density of the edge is less than or equal to one, the parameter h_e is initially set to zero.

Since we focus on the rerouting phase, we assume that for each net $n_i \in S$, there exists a Steiner tree ϕ_i which connects all nodes in n_i. Given a set of trees $\{\phi \mid n_i \in S\}$, we can determine the values of $\text{ov}(e), \eta(e)$ for each edge $e \in E$ and identify the set of *congested nets* $S' \subseteq S$. A net $n_i \in S'$ is referred to as *congested* if its Steiner tree ϕ_i has at least one edge with overflow.

We propose a two-phase solution for rerouting congested nets using congestion-aware topologies. In the first phase we iteratively rip up each net $n_i \in S'$ and reroute it using a congestion-aware Steiner tree with the goal of minimizing the maximum overflow $\text{ov}_{\max}$ and the total overflow ov_{tot}. In second phase, we deal with the nets that remain congested at the end of the first phase and rip-up-and-reroute pairs of congested nets using congestion-aware network coding topologies to further reduce congestion and minimize the number of overflows. Note that the nets considered in phase two correspond to difficult cases, where congestion avoidance was not possible even after several attempts of ripping up and rerouting *individual* nets. Therefore in the second phase, we consider the *pairs* of congested nets for further improvement. The example given in Figure 1 shows the advantage of using network coding topologies for routing pairs of nets over using standard routing techniques that handle each net separately.

3. Congestion-Aware Steiner Trees

In this section, we present several techniques for finding congestion-aware Steiner trees. Our goal is to find Steiner trees that minimize congestion with a minimum penalty in terms of the overall wirelength. We would like to achieve better tradeoffs between congestion mitigation and total wirelength. These tradeoffs are useful for practical applications as in some cases, congestion mitigation is preferable to wirelength reduction, whereas in other cases, the wirelength reduction is of higher priority.

3.1. Previous Work on Steiner Tree Routing. The Steiner tree problem is a well-studied NP-complete problem [14]. There is a wealth of heuristic and approximate solutions that have been developed for this problem. The best-known approximation algorithm has an approximation ratio of 1.55 (i.e., the cost of the tree returned by the algorithm is less than 1.55 times the optimum) [15]. The best known

approximations require significant computation time, so we focus on computationally feasible and easy to implement approximation and heuristic solution for constructing Steiner trees.

3.2. Algorithms for Finding Congestion-Aware Trees. As mentioned above, our goal is to rip up and reroute nets that use congested edges of $G(V, E)$. For each net $n_i \in S$ which has been ripped up, we need to find an alternative Steiner tree that uses uncongested routes. In this section, we describe five algorithms for finding congestion-aware Steiner trees. The first three algorithms use combinatorial techniques (see, e.g., [1, 16, 17]) while, the last two are based on the intelligent search techniques [18]. The performance of the algorithms is evaluated in Section 5.

Algorithm stTree1. This algorithm approximates a minimum cost Steiner tree by using a *shortest path tree*. A shortest path tree is a union of the shortest paths between source s_i and a set of terminals T_i. A shortest path tree can be identified by a single invocation of Dijkstra's algorithm. However, the cost of the tree may be significantly higher than the optimum.

Algorithm stTree2. This algorithm constructs the tree in an iterative manner. We iteratively process the terminals in T_i in the increasing order of their distance form s_i. More specifically, we first find a shortest path P_1 between source s_i and terminal t_1. Then, we assign a zero cost to all edges that belong to P_1 and find a shortest path P_2 between s and t_2 with respect to modified costs. The idea behind this algorithm is to encourage sharing of the edges between different paths. That is, if an edge e belongs to P_1, it can be used in P_2 with no additional cost. In general, when finding a shortest path to terminal t, all edges that belong to paths of previously processed terminals are assigned a zero cost. This algorithm requires $|T| - 1$ iterations of Dijkstra's algorithm, but it typically returns a lower-cost tree than Algorithm *stTree1*.

Algorithm stTree3. This is a standard approximation algorithm with the approximation ratio of 2 (i.e., the cost of the tree returned by the algorithm is at most two times higher than the optimal cost). Specifically, with this algorithm, we find a shortest path between each pair of nodes in the set $\overline{T} = \{T \cup \{s_i\}\}$. Then, we construct a complete graph G' such that each node in G' corresponds to a node in $\overline{T}$. The weight of an edge $e \in G'$ is equal to the minimum length of the path between two corresponding nodes in $\overline{T}$. The algorithm then finds a minimum spanning tree ϕ in G'. Next, each edge in ϕ is substituted by the corresponding shortest path in G, which results (after removing redundant edges) in a Steiner tree in G that connects source s_i with terminals in T_i.

Algorithm stTree4. Algorithm $stTree4$ is an intelligent search-based algorithm. Our approach is inspired by Algorithm A^*. Algorithm A^* is a shortest path algorithm that uses a heuristic function $\lambda(v)$ to determine the order of visiting nodes of the graph in order to improve its running time. Specifically, for each node v, we define $\lambda(v)$ to be the maximum distance between node v and a terminal $t \in T_i$ which has not yet been visited. The distance between v and $t \in T_i$ is defined as the minimum number of hops that separate v and t in G. The Algorithm $stTree4$ follows the same steps as Algorithm $stTree1$, but it uses Algorithm A^* with heuristic function $\lambda(v)$ to find shortest paths.

Algorithm stTree5. Algorithm $stTree5$ is also based on Algorithm A^*. It follows the same steps as Algorithm $stTree2$, but it uses Algorithm A^* with the same heuristic function $\lambda(v)$ as in Algorithm $stTree4$.

4. Network Coding Techniques

In this section, we use the network coding techniques in order to achieve further improvement in terms of minimizing congestion and reducing the number of overflows. The network coding technique enables, under certain conditions, to share edges that belong to different nets. For example, in the graph depicted in Figure 1(c), there are two minimum Steiner trees one transmitting signal a from source s_1 and the second transmitting signal b from source s_2. These two trees clash at the middle edge (emanating from t_1), resulting in an overflow. This conflict can be resolved by coding at node t_1, which effectively allows two trees to share certain edges. Similarly, our algorithm will identify pairs of nets that share terminals and then apply network coding techniques to reduce overflow.

4.1. Previous Work on Network Coding. The problem of routing of multiple nets with shared terminals is related to the problem of establishing efficient multicast connections in communication networks. The network coding technique was proposed in a seminal paper by Ahlswede et al. [3]. It was shown in [3, 19] that the capacity of the multicast networks, that is, the number of packets that can be sent simultaneously from the source node s to all terminals, is equal to the minimum size of a cut that separates s from each terminal. Li et al. [19] proved that *linear network codes* are sufficient for achieving the capacity of the network. In a subsequent work, Koetter and Médard [20] developed an algebraic framework for network coding. This framework was used by Ho et al. [21] to show that linear network codes can be efficiently constructed through a randomized algorithm. Jaggi et al. [22] proposed a deterministic polynomial-time algorithm for finding feasible network codes in multicast networks. An initial study of applicability of network coding for improving the routability of VLSI designs appears in [23]. In [24], Gulati and Khatri used network coding for improving the routability of FPGAs, focusing on finding nets that are suitable for network coding. To the best of our knowledge, this is the first work that proposes efficient algorithms for finding the congestion-aware network coding topologies for VLSI circuits.

4.2. Network Coding Algorithm. We proceed to present the algorithm we use for constructing congestion-aware coding networks that reduce congestion and overflow.

Algorithm NC $(G, s_i, s_j, T_{ij}, T'_i, T'_j)$:
(1) Find a Steiner tree ϕ_i connects s_i to terminals in T'_i
(2) Find a Steiner tree ϕ_j connects s_j to terminals in T'_j
(3) For each link $e \in T'_i \cup T'_j$ update $\eta(e)$
(4) Find a Steiner tree $\hat{\phi}$ that connects s_i to terminals in T_{ij}
(5) Assign zero cost to all edges in $\hat{\phi}$
(6) For each terminal $t \in T_{ij}$ let $d(t)$ be the shortest distance between s_j and t
(7) For all terminals $t \in T_{ij}$ in the increasing order of $d(t)$ do:
(8) Let $G'(V', E')$ be a graph formed from $G(V, E)$ by reversing all edges in $P_{i,t}$, where $P_{i,t}$ is a path from s_i to t in $\hat{\phi}$
(9) Find shortest path $P_{j,t}$ from source s_j to terminal t in $G'(V', E')$
(10) Assign zero cost to all edges of $P_{j,t}$
(11) **For** each edge $e(v, u) \in P_{j,t}$ **do**
(12) If there exists an edge $e'(u, v) \in \hat{\phi}$, remove $e'(u, v)$ from $\hat{\phi}$
(13) Otherwise, add $e(v, u)$ to $\hat{\phi}$
(14) Return $\hat{\phi}$, ϕ_i, and ϕ_j

Algorithm 1: Algorithm *NC*.

The algorithm includes the following steps. First, we identify the subset S' of S that includes nets that go through edges with overflow. Second, we identify pairs of nets in S' that share at least three common terminals. Next, we check, for each such pair of nets (n_i, n_j), whether we can replace the Steiner trees for n_i and n_j by a more efficient routing topology with respect to congestion and overflow.

More specifically, let (n_i, n_j) be a pair of nets in S' that share at least three terminals. Let s_i and s_j be the source nodes of these nets. We denote the set of terminals shared by n_i and n_j by T_{ij}. We also denote by T'_i the set of terminals in T_i that do not belong to T_{ij} that is, $T'_i = T_i \setminus T_{ij}$. Similarly, we denote by T'_j the set of terminals in T_j that do not belong to T_{ij} that is, $T'_j = T_j \setminus T_{ij}$. Next, we find two congestion-aware Steiner trees ϕ_i and ϕ_j that connect s_i to T'_i and s_j to T'_j. These trees can be identified by one of the algorithms presented in Section 3. The parameter $\eta(e)$ for each $e \in G$ is updated after finding ϕ_i and ϕ_j.

Finally, we find a congestion-aware network coding topology $\hat{\phi}$ that connects s_i and s_j to the common set of terminals T_{ij} in an iterative manner. First, we let $\hat{\phi}$ be a Steiner tree with source s_i and terminals T_{ij}. All edges of $\hat{\phi}$ are always assigned zero cost. We then sort the terminals T_{ij} in the increasing order of their distance (in the original graph) from s_j and process them in that order. For each terminal $t \in T_{ij}$, we reverse all the edges in the path $P_{i,t}$ between source s_i and terminal t and find a shortest path $P_{j,t}$ between source s_j and terminal t. Then, for each link $e(v, u) \in P_{j,t}$ we perform the following procedure. If there exists a link $e'(u, v)$ in $\hat{\phi}$, we remove $e'(u, v)$ from $\hat{\phi}$ otherwise, we add $e(v, u)$ to $\hat{\phi}$. A sample execution of this procedure is shown in Figure 2. It is easy to verify that the algorithm produces a feasible network coding topology that is, a topology that ensures that for each terminal $t \in T_{ij}$ there are two-edge disjoint paths that connect s_i and s_j with t. The formal description of algorithm for identifying the network coding topology, referred to as Algorithm *NC*, is given in Algorithm 1.

After the execution of the algorithm, we determine whether the total cost of $\hat{\phi}$, ϕ_i, and ϕ_j is less than the total cost of the original Steiner trees for nets n_i and n_j. If there is a reduction in terms of cost, the two original Steiner trees are replaced by $\hat{\phi}$, ϕ_i, and ϕ_j.

Our experimental results, presented in Section 5, show that the number of coding opportunities is relatively small. However, by applying the network coding technique on a limited number of nets, we can achieve a significant reduction in the number of overflows. Also, since the network coding technique is applied to a limited number of nets, the overhead in terms of the number of additional required gates is relatively small.

5. Performance Evaluation

We have evaluated the performance of our algorithms using the ISPD98 routing benchmarks [25]. All the experiments are performed on a 3.2 GHz Intel Xeon dual-core machine. In all experiments, we first run the Steiner tree tool *Flute* [26] in order to determine the initial routing of all nets in the benchmark. Next, we perform an iterative procedure, referred to as Phase 1, which processes each net with overflows and checks whether an alternative Steiner tree of lower cost and with smaller number of overflows exists, and if yes, it rips up the existing tree and replaces it with an alternative one. This phase uses one of the algorithms described in Section 3. Phase 1 terminates when four subsequent iterations yield the same cost and the number of overflows, indicating that further reduction in the number overflows is unlikely.

Next, we check whether the application of the network coding technique can further reduce the number of overflows. This phase is referred to as the Phase 2. We first identify pairs of nets that have overflowed edges and share at least there terminals. We then apply Algorithm *NC*, presented in Section 4, to find an alternative network coding topology and perform rip-up and reroute if such a topology is beneficial in terms of reducing congestion and eliminating overflows.

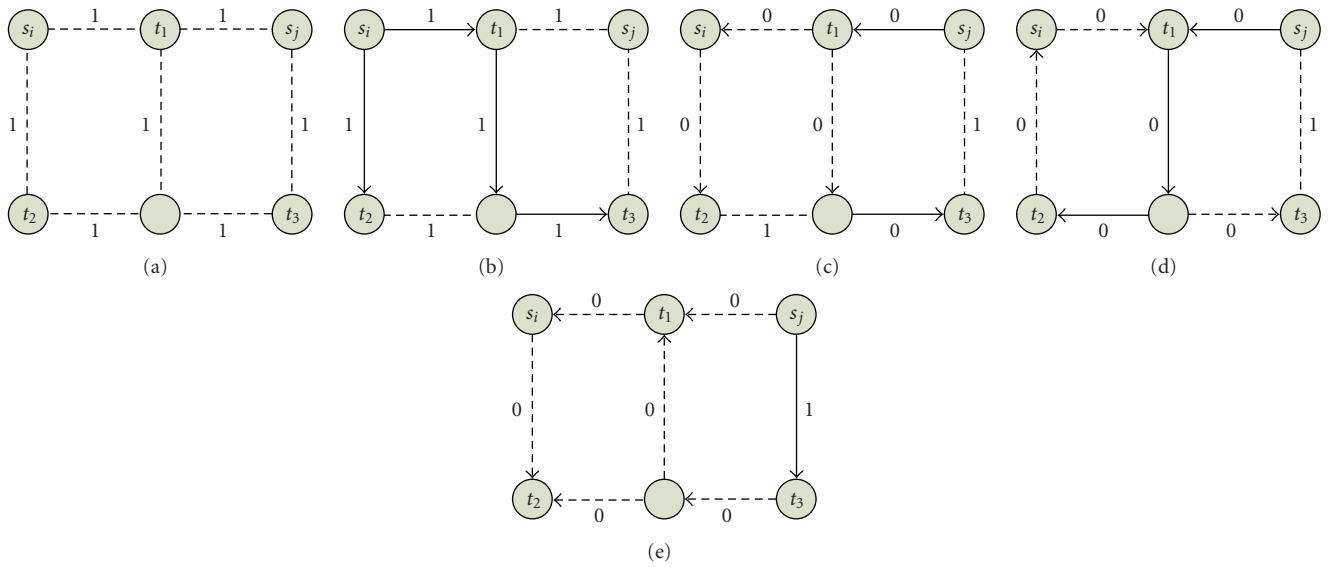


Figure 2: Steps for finding network coding topology using Algorithm *NC*. (a) A graph G with two nets n_i and n_j that connect nodes s_i and s_j to terminals $T_1 = T_2 = T_{12} = \{t_1, t_2, t_3\}$. (b) A Steiner tree $\hat{\phi}$ connecting s_j to T_2. (c) Modified graph in which the costs of all edges found in $\hat{\phi}$ are set equal to zero and the edges connecting s_i to t_1 are reversed. A shortest path from s_j to t_1 is shown. (d) Modified graph in which the edges connecting s_i to t_2 are reversed. A shortest path from s_j to t_2 is shown. (e) Modified graph in which the edges connecting s_i to t_3 are reversed. A shortest path from s_j to t_3 is shown.

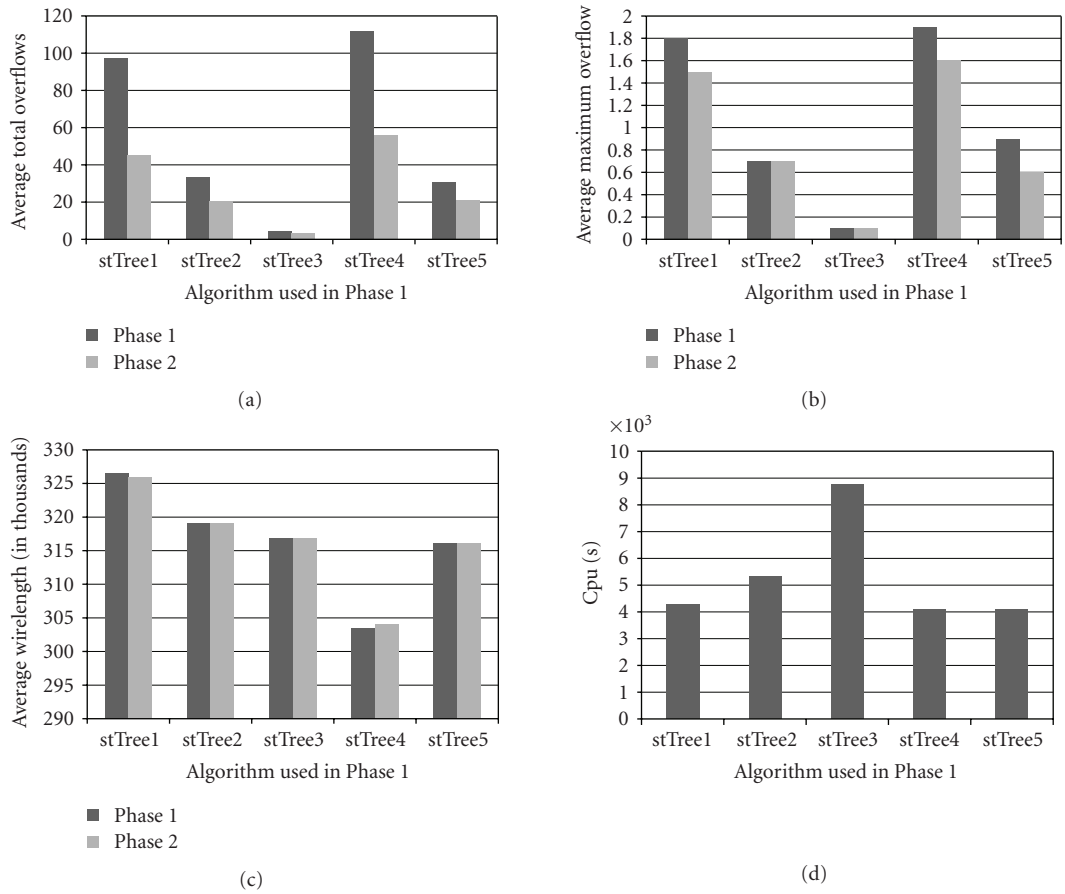


Figure 3: Evaluation of five different algorithms for Phase 1 and Algorithm *NC* for Phase 2 on ISPD98 benchmark files. In all experiments, the polynomial cost function is used. (a) Comparison of the average total overflow. (b) Comparison of the average maximum overflow. (c) Comparison of the average total wirelength. (d) Comparison of the average running time. Results present average over 10 ISPD98 benchmark files.

Table 1: Performance evaluation using Algorithm *stTree*3 for Phase 1 and Algorithm *NC* for Phase 2 on ISPD98 benchmarks with polynomial cost functions.

	Phase 1			Phase 2		
	ov_{tot}	ov_{max}	wlen	ov_{tot}	ov_{max}	wlen
ibm1	14	1	81058	6	1	80727
ibm2	0	0	216299	0	0	216299
ibm3	0	0	188391	0	0	188391
ibm4	125	2	196106	93	2	195949
ibm5	0	0	415681	0	0	415681
ibm6	0	0	336105	0	0	336105
ibm7	0	0	466381	0	0	466381
ibm8	0	0	477404	0	0	477404
ibm9	0	0	508661	0	0	508661
ibm10	0	0	711201	0	0	711201

Table 2: Performance evaluation of different cost functions (using Algorithm *stTree*3 for Phase 1 and Algorithm *NC* for Phase 2) on ISPD98 benchmarks.

	Polynomial cost		Exponential cost		History-based cost	
	ov_{tot}	ov_{max}	ov_{tot}	ov_{max}	ov_{tot}	ov_{max}
ibm1	0	0	117	1	0	0
ibm2	0	0	79	2	0	0
ibm3	0	0	0	0	0	0
ibm4	31	1	834	7	439	4
ibm5	0	0	0	0	0	0
ibm6	0	0	54	2	0	0
ibm7	0	0	82	1	0	0
ibm8	0	0	37	1	0	0
ibm9	0	0	15	1	0	0
ibm10	0	0	95	3	0	0

Table 3: Performance evaluation using Algorithm *stTree*3 for Phase 1 and Algorithm *NC* for Phase 2 on modified ISPD98 benchmarks (decreased both horizontal and vertical capacity by 1 unit) with polynomial cost function.

	Flute			Phase 1			Phase 2			Number of XOR gates
	ov_{tot}	ov_{max}	wlen	ov_{tot}	ov_{max}	wlen	ov_{tot}	ov_{max}	wlen	
ibm1	3257	14	60209	362	4	78161	184	3	77231	5
ibm2	5678	22	166193	18	1	217673	4	1	217737	12
ibm3	2308	16	145777	1	1	183395	0	0	183391	2
ibm4	4833	15	162844	550	6	193832	440	6	193131	52
ibm5	5	4	410038	0	0	476251	0	0	473265	70
ibm6	5492	27	276012	3	1	334165	1	1	334163	0
ibm7	4665	15	363678	1	1	473743	0	0	473743	2
ibm8	5400	13	403502	3	1	476880	0	0	476864	0
ibm9	8545	14	411524	16	1	496563	9	1	496409	24
ibm10	7103	20	574743	4	1	713569	1	1	713553	18

Table 4: Improvement over MaizeRouter for modified (congested) IBM benchmarks using Algorithm *stTree*3 for Phase 1 and Algorithm *NC* for Phase 2, using polynomial cost function.

	Ver. Cap	Hor. Cap	MaizeRouter		Phase 1 and Phase 2	
			ov_{tot}	ov_{max}	ov_{tot}	ov_{max}
ibm1	11	13	43	1	34	1
ibm5	21	31	45326	25	45151	25
ibm8	17	28	649	9	641	9
ibm9	10	24	4006	9	3989	9

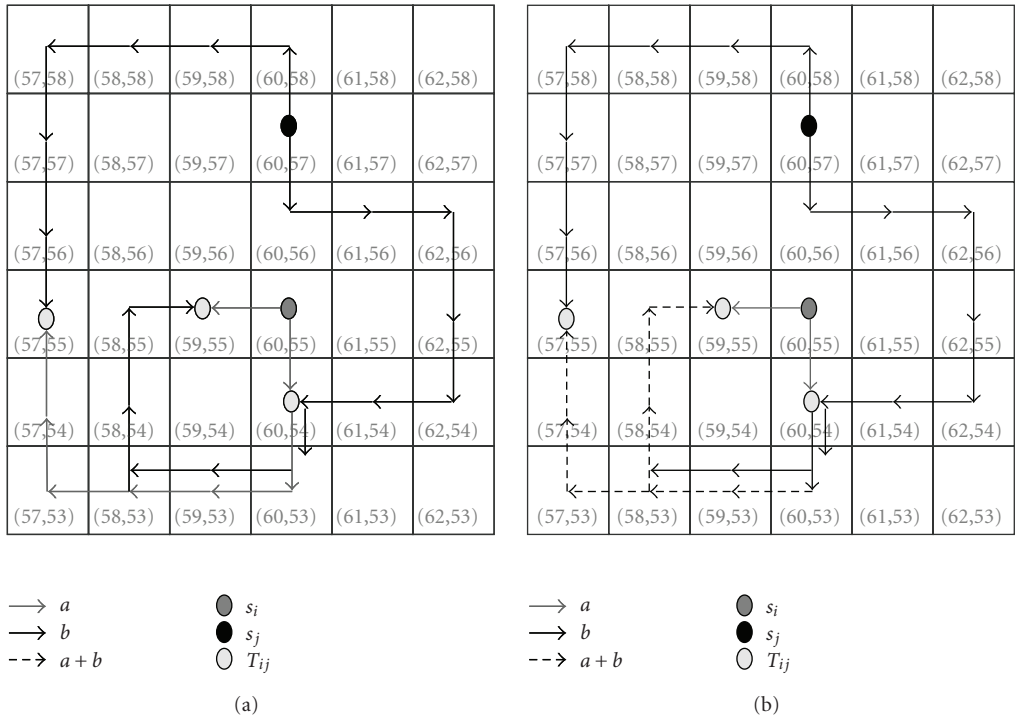


Figure 4: A network coding topology for benchmark ibm1 for pair of nets $n_i = 66$ and $n_j = 1531$ computed using Algorithm *NC*. There are two nets n_i and n_j with sources $s_i = (60, 55)$, and $s_j = (60, 57)$ and set of common terminals $T_{ij} = (59, 55), (60, 54)$, and$(57, 55)$. Part(a) shows routing layout without network coding. Part(b) shows routing layout with network coding. Example shows that network coding has helped to reduce congestion on edges $(60, 54)$, $(60, 53)$, $(59, 53)$, and $(58, 53)$.

The experimental results are shown in Figures 3(a)–3(c). The figures present average performance over all ten benchmarks. The cost function for this set of experiments was set according to (6) with $\alpha \geq 10$. We have observed that larger values of α yield fewer overflows but result in larger running times and increased wirelength. We also note that for Phase 1, Algorithm $stTree3$ shows the best performance in terms of reducing the total number of overflows as well as reducing the maximum overflow. In fact, Algorithm $stTree3$ eliminates all overflows in all benchmarks, except for ibm4 as given in Table 1. We also note that Algorithms $stTree4$ and $stTree5$ yield Steiner trees with a smaller total wirelength. This is due to the fact the intelligent search algorithms favor paths that have small hop count.

We observe that the network coding technique results in a considerable reduction of the total number of overflows as well as reduces the maximum overflow. Furthermore, for each pair of nets for which we perform network coding, the number of required gates is small. Moreover, in all cases that we have encountered, the network coding operations can be performed over finite field $GF(2)$, that is, each encoding node can be implemented with a single XOR gate. Such gates incur minimum overhead, because they can serve as buffers for long wires. An example of how network coding can help in reducing the wirelength of two nets in the ibm1 benchmark is shown in Figure 4.

Figure 3(d) compares the running times of the different Algorithms for Phase 1 and Algorithm *NC* for Phase 2. As expected, Algorithm $stTree4$ is one of the fastest algorithms, whereas Algorithm $stTree1$ and $stTree5$ have running times comparable to Algorithm $stTree2$. Moreover, Algorithms $stTree4$ and $stTree5$ are faster than their counterparts (Algorithms $stTree1$ and $stTree2$, resp.). This is due to the fact that intelligent search methods speed up the search by preferring nodes closer to the destination.

In the second set of experiments, we evaluated the performance of three cost functions mentioned in Section 2.2 on the ISPD98 benchmarks using Algorithm $stTree3$ for Phase 1 and Algorithm *NC* for Phase 2. For cost function given by (6), we used $\alpha \geq 10$, whereas for cost function given by (7), we used $\beta \geq 50$, and for cost function given by (8), $\rho(e)$ was a polynomial cost function with $\alpha \geq 10$. The results are shown in Table 2. Polynomial cost function showed the best performance in terms of overflows.

In another set of experiments, we worked on slightly modified ISPD98 benchmark files. The modification included reducing the vertical and horizontal capacity by one unit. For Phase 1, we used Algorithm $stTree3$ and then applied Algorithm *NC* in Phase 2. Polynomial cost function given by (6) was used to check the performance on these more congested cases. The results are given in Table 3.

We have also conducted the same experiment on several selected benchmarks on the output of MaizeRouter [13]. In these experiments, we iteratively reduced the vertical and horizontal capacity of the ISPD98 benchmarks until we got overflows while running them through MaizeRouter. Then,

we used the output of MaizeRouter as input to Phase 1 using Algorithm $stTree3$, and after that, we have applied Algorithm *NC* in Phase 2. The cost function used was polynomial with $\alpha = 10$. The results are shown in Table 4. The results demonstrate that Algorithms *stTree3* combined with Algorithm *NC* perform well and can contribute to further reduction of the number of overflows.

6. Conclusions

In this paper, we presented several efficient techniques for rip-up-and-reroute stage of the global routing process. Our techniques are based on ripping up nets that go through highly congested areas and rerouting them using efficient Steiner tree algorithms. We have considered several efficient Steiner tree algorithms as well as several cost functions that take into account congestion and overflow. We have also studied the application of the novel technique of network coding and showed that it can efficiently handle difficult routing cases, facilitating reduction in the number of overflows.

Acknowledgment

A preliminary version of this paper appeared in the proceedings of the 2009 IEEE Computer Society Annual Symposium on VLSI (ISVLSI) [27]. This work was supported in part by NSF grant CNS-0954153.

References

[1] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, John Wiley & Sons, New York, NY, USA, 1990.

[2] N. A. Sherwani, *Algorithms for VLSI Physical Design Automation*, Kluwer Academic Publishers, Norwell, Mass, USA, 2nd edition, 1995.

[3] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[4] J. Hu and S. S. Sapatnekar, "A survey on multi-net global routing for integrated circuits," *Integration, the VLSI Journal*, vol. 31, no. 1, pp. 1–49, 2001.

[5] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Predictable routing," in *Proceedings of IEEE/ACM International Conference on Computer Aided Design (ICCAD '00)*, pp. 110–113, 2000.

[6] C. Albrecht, "Provably good global routing by a new approximation algorithm for multicommodity flow," in *Proceedings of the International Symposium on Physical Design (ISPD '00)*, pp. 19–25, April 2000.

[7] J. Cong, J. Fang, and Y. Zhang, "Multilevel approach to full-chip gridless routing," in *Proceedings of the International Conference on Computer-Aided Design*, pp. 396–403, November 2001.

[8] Y. W. Chang and S. P. Lin, "MR: A new framework for multilevel full-chip routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 5, pp. 793–800, 2004.

[9] M. Cho and D. Z. Pan, "BoxRouter: a new global router based on box expansion and progressive ILP," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 373–378.

[10] M. Cho, K. Lu, K. Yuan, and D. Z. Pan, "BoxRouter 2.0: architecture and implementation of a hybrid and robust global router," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD '07)*, pp. 503–508, November 2007.

[11] M. Pan and C. Chu, "FastRoute 2.0: a high-quality and efficient global router," in *Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC '07)*, pp. 250–255, January 2007.

[12] J. A. Roy and I. L. Markov, "High-performance routing at the nanometer scale," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD '07)*, pp. 496–502, November 2007.

[13] M. D. Moffitt, "MaizeRouter: engineering an effective Global Router," in *Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC '08)*, pp. 226–231, March 2008.

[14] M. R. Garey and D. S. Johnson, *Computers and Intractability*, Freeman, San Francisco, Calif, USA, 1979.

[15] G. Robins and A. Zelikovsky, "Improved Steiner tree approximation in graphs," in *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 770–779, January 2000.

[16] S. Voß, "Steiner's problem in graphs: heuristic methods," *Discrete Applied Mathematics*, vol. 40, no. 1, pp. 45–72, 1992.

[17] M. P. D. Aragão and R. F. Werneck, "On the implementation of MST-based heuristics for the steiner problem in graphs," in *Proceedings of the 4th International Workshop on Algorithm Engineering and Experiments*, pp. 1–15, 2002.

[18] R. Dechter and J. Pearl, "Generalized best-first search strategies and the optimality of A*," *Journal of the ACM*, vol. 32, no. 3, pp. 505–536, 1985.

[19] S. Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[20] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, 2003.

[21] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proceedings of IEEE International Symposium on Information Theory (ISIT '03)*, p. 442, July 2003.

[22] S. Jaggi, P. Sanders, P. A. Chou et al., "Polynomial time algorithms for multicast network code construction," *IEEE Transactions on Information Theory*, vol. 51, no. 6, pp. 1973–1982, 2005.

[23] N. Jayakumar, S. P. Khatri, K. Gulati, and A. Sprintson, "Network coding for routability improvement in VLSI," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD '06)*, pp. 820–823, November 2006.

[24] K. Gulati and S. R. Khatri, "Improving FPGA routability using network coding," in *Proceedings of the 18th ACM Great Lakes Symposium on VLSI (GLSVLSI '08)*, pp. 147–150, March 2008.

[25] C. J. Alpert, "ISPD98 circuit benchmark suite," in *Proceedings of the International Symposium on Physical Design (ISPD '98)*, pp. 80–85, April 1998.

[26] C. Chu, "FLUTE: fast lookup table based wirelength estimation technique," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD '04)*, pp. 696–701, November 2004.

[27] M. A. R. Chaudhry, Z. Asad, A. Sprintson, and J. Hu, "Efficient rerouting algorithms for congestion mitigation," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI (ISVLSI '09)*, pp. 43–48, May 2009.

Hindawi Publishing Corporation
VLSI Design
Volume 2011, Article ID 845957, 13 pages
doi:10.1155/2011/845957

Research Article

Finding the Energy Efficient Curve: Gate Sizing for Minimum Power under Delay Constraints

Yoni Aizik and Avinoam Kolodny

Department of Electric Engineering, Technion, Israel Institute of Technology, Haifa 32000, Israel

Correspondence should be addressed to Yoni Aizik, yoni.aizik@intel.com

Received 23 September 2010; Accepted 28 January 2011

Academic Editor: Shiyan Hu

A design scenario examined in this paper assumes that a circuit has been designed initially for high speed, and it is redesigned for low power by downsizing of the gates. In recent years, as power consumption has become a dominant issue, new optimizations of circuits are required for saving energy. This is done by trading off some speed in exchange for reduced power. For each feasible speed, an optimization problem is solved in this paper, finding new sizes for the gates such that the circuit satisfies the speed goal while dissipating minimal power. Energy/delay gain (EDG) is defined as a metric to quantify the most efficient tradeoff. The EDG of the circuit is evaluated for a range of reduced circuit speeds, and the power-optimal gate sizes are compared with the initial sizes. Most of the energy savings occur at the final stages of the circuits, while the largest relative downsizing occurs in middle stages. Typical tapering factors for power efficient circuits are larger than those for speed-optimal circuits. Signal activity and signal probability affect the optimal gate sizes in the combined optimization of speed and power.

1. Introduction

Optimizing a digital circuit for both energy and performance involves a tradeoff, because any implementation of a given algorithm consumes more energy if it is executed faster. The tradeoff between power and speed is influenced by the circuit structure, the logic function, the manufacturing process, and other factors. Traditional design practices tend to overemphasize speed and waste power. In recent years power has become a dominant consideration, causing designers to downsize logic gates in order to reduce power, in exchange for increased delay. However, resizing of gates to save power is often performed in a nonoptimal way, such that for the same energy dissipation, a sizing that results in better performance could be achieved.

In this paper, we explore the energy-performance design space, evaluating the optimal tradeoff between performance and energy by tuning gate sizes in a given circuit. We describe a mathematical method that minimizes the total energy in a combinational CMOS circuit, for a given delay constraint. It is based on an extension of the Logical Effort [1] model to express the dynamic and leakage energy of a path as well as the delay. Starting from the minimum achievable delay, we apply the method for a range of longer delays, in order to find the optimal energy-delay relation for the given circuit. We show that downsizing all gates in a fast circuit by the same factor does not yield an energy-efficient design, and we characterize the differences between gate sizing for high speed and sizing for low power.

In trading off delay for energy, we are interested only in a subset of all the possible downsized circuits2014those implementations that are energy efficient. A design implementation is considered to be energy efficient when it has the highest performance among all possible configurations dissipating the same power [2, 3]. When the optimal implementations are plotted in the energy-delay plane, they form a curve called the *energy efficient curve*. In Figure 1, each point represents a different hardware implementation. The implementations which belong to the energy efficient family reside on the energy efficient curve.

Zyuban and Strenski [3, 4] introduce the *hardware intensity* metric. Hardware intensity (η) is defined to be the ratio of the relative increase in energy to the corresponding relative gain in performance achievable "*locally*" through

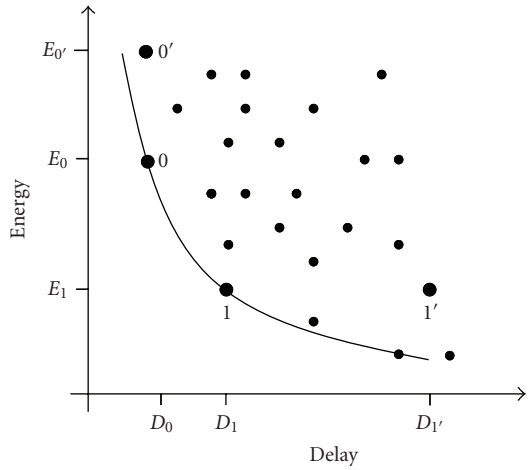


Figure 1: Energy efficient curve. Although implementations 0 and 0′ of the given circuit have the same delay (D_0), implementation 0 consumes less energy. Similarly, implementations 1 and 1′ consume the same energy, but implementation 1 has a shorter delay (D_1), hence is preferable. Points 0 and 1 are on the energy efficient curve. All implementations have the same circuit topology, with different device sizes.

gate resizing and logic manipulation at a fixed power-supply voltage for a power efficient design. Simply put, it is the ratio of % energy per % speed performance tradeoff for an energy-efficient design. Since speed performance is inversely proportional to delay,

$$\eta = -\frac{1/E}{1/D}\frac{\partial E}{\partial D}, \tag{1}$$

where D is delay, E is the dissipated energy, and η represents the hardware intensity. The hardware intensity is a measure of the "*differential*" energy-performance tradeoff (the energy gained if the delay is relaxed by a small ΔD around a given delay and energy point on the energy efficient curve), and is actually the sensitivity of the energy to the delay.

As shown in [3], each point on the energy efficient curve corresponds to a different value of the hardware intensity η. The hardware intensity decreases along the energy efficient curve towards larger delay values. According to [3], η is equivalent to the tradeoff parameter n in the commonly used optimization objective function combining energy and delay:

$$F_{\text{opt}} = E \cdot D^n, \quad n \geq 0. \tag{2}$$

In [5], Brodersen et al. formalize the tradeoff between energy and delay via sensitivities to tuning parameters. The sensitivity of energy to delay due to tuning the size W_i of gate i is defined as

$$\theta(W_i) = -\frac{1/E}{1/D} \cdot \frac{\partial E/\partial W_i}{\partial D/\partial W_i}, \tag{3}$$

where $\theta(W_i)$ is the sensitivity, D is the delay, E is the energy, $\partial E/\partial W_i$ is the derivative of energy with respect to size of device i, and $\partial D/\partial W_i$ is the derivative of delay with respect to size of device i. To achieve the most energy-efficient design, the energy reduction potentials of all the tuning variables must be the same. Therefore, for an energy efficient design, (3) is equivalent to (1) for all points on the energy efficient curve.

The focus of this paper is on the conversion to low power of circuits that were optimized only for speed during their initial design process. Optimal downsizing is applied to each gate for each relaxed delay target, such that the whole energy efficient curve is generated for the circuit. Note that the gate sizes are allowed to vary in a continuous manner between a minimum and a maximum size. While the resultant gate sizes would be mapped into a finite cell library in a practical design, the continuous result for some basic circuits provides guidelines and observations about CMOS circuit design for low power.

The rest of this paper is organized as follows: The design scenario is described in Section 2. Usage of logical effort to analyze the delay and energy is described in Section 3. The optimization problem is formalized in Section 4. Typical circuit types are analyzed in Section 5. Section 6 concludes the paper.

2. Power Reduction Design Scenario

Typically, an initial circuit is given, where speed was the only design goal. In order to save energy, the delay constraint is relaxed, and the gates sizes are reduced. For example, consider Figure 1, with the initial circuit implementation 0, which is energy efficient. While relaxing the delay constraint (moving from D_0 to D_1), the design gets downsized, which results in circuit implementation 1.

To calculate the energy gain achievable by relaxing the delay by X percent, we define a metric we call "Energy Delay Gain" (EDG). The EDG is defined as the ratio of relative decrease in energy to the corresponding relative increase in delay, with respect to the initial design point (D_0, E_0). D_0 is the initial delay (not necessarily the minimum achievable delay), and E_0 is the corresponding initial energy. Note that the EDG defines the total energy-performance tradeoff, as opposed to the differential tradeoff—the hardware intensity. Mathematically, EDG at a given delay D with corresponding energy E is defined as

$$\text{EDG} = \frac{(E_0 - E)/E_0}{(D - D_0)/D_0}. \tag{4}$$

For example, assuming that the initial design point in Figure 1 is implementation 0, then the EDG of point 1 is

$$\frac{(E_0 - E_1)/E_0}{(D_1 - D_0)/D_0}. \tag{5}$$

Figure 2 illustrates the difference between hardware intensity and EDG. It shows the energy efficient curve of a given circuit, where D_0 is the initial delay, and E_0 is the corresponding initial energy. The hardware intensity is the ratio between the slope of the tangent to the energy efficient curve at point (D, E) to the slope of the line connecting the origin to point (D, E). The EDG is the ratio between the slope of the line connecting points (D_0, E_0) and (D, E), to the slope of the

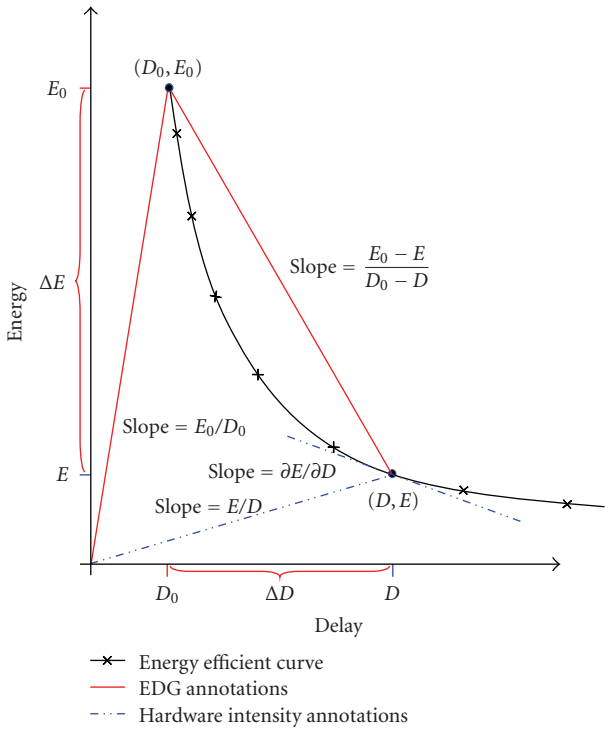


Figure 2: EDG and hardware intensity. Note that when $(D, E) \rightarrow (D_0, E_0)$, hardware intensity and EDG converge.

line connecting the origin to point (D_0, E_0). Note that when point (D, E) is close to (D_0, E_0), the two definitions converge.

Resizing of the gates to tradeoff performance with active energy is the most practical approach available to the circuit engineer. Continuous gate sizes has been used for optimizing delay under area constraints and vice versa [6]. Other degrees of freedom include logic restructuring, tuning of threshold voltages or supply voltage, and power gating. Changing the threshold voltage affects mainly the leakage energy, and not the dynamic energy dissipation [7, 8], so does power gating [9, 10]. Logic restructuring of the circuit could be an effective method to trade off energy and performance, by reducing the load on high activity nets, and by introducing new nodes that have a lower switching activity [11]. However, changing the circuit topology may increase the time required for the design process to converge. Changing the supply voltage is an effective technique as well [3, 5, 7, 11–14]. However, in most cases, changing the supply voltage for a subcircuit requires major changes in the package and in the system and therefore is not practical. For instance, latest state-of-the art CPUs include only 1-2 power planes [15, 16].

In the following sections, we set up an optimization framework that maximizes the energy saving for any assumed delay constraint in a given combinational CMOS circuit. It determines the appropriate sizing factor for each gate in the circuit. For primary inputs and outputs of the circuit we assume that fixed capacitances. Given activity factor and signal probability are assumed at each node of the circuit. The result of this optimization process is equivalent to finding the energy-efficient curve for the given circuit.

3. Analytical Model

The optimization problem we solve is defined as follows. Given a path in a circuit with initial delay (minimum or arbitrary) D_0 and the corresponding energy consumption E_0, find gate sizing that maximizes the EDG for an assumed delay constraint. We use the logical effort method [1] in order to calculate the delay of a path and adapt it to calculate the dynamic and leakage energy dissipation of the circuit.

For a given path (Figure 3), we assume that constant input and output loads and an initial sizing that is given as input capacitance for each gate. For each gate we apply a sizing factor k. The input capacitance of the resized ith gate is expressed as the initial input capacitance C_{0_i} multiplied by k_i. The energy-delay design space is explored by tuning the k's.

The following properties are defined:

M_i: number of inputs to gate i,

AF_i^j: activity factor (switching probability) of input j in gate i,

AF_o^i: output activity factor of gate i,

g_i: logical effort of gate i,

p_i: parasitic delay of gate i,

C_{0_i}: initial capacitance of gate i that achieves initial path delay (corresponds to (D_0, E_0)),

C_{off_i}: off-path constant capacitance driven by gate i,

$\overline{P_{\mathrm{leak}_i}}$: the average leakage power for gate i, for a unit input capacitance,

k_i: sizing factor for gate i. The k's are used in the gate downsizing process. For each gate i, $k_i \cdot C_{0_i}$ is the gate size. Although specified, k_1 is assumed to be 1 (constant driver).

3.1. Energy of a Logic Path

3.1.1. Switching Energy. The switching energy of a static CMOS gate i with M_i inputs and a single output is

$$\text{Switching Energy} = \underbrace{\sum_{j=1}^{M_i} \mathrm{AF}_i^j \cdot C_j \cdot V_j^2}_{\text{input energy}} + \underbrace{\mathrm{AF}_{\mathrm{out}_i} \cdot C_{\mathrm{out}_i} \cdot V_{\mathrm{out}_i}^2}_{\text{output energy}}. \quad (6)$$

Assuming that the voltage amplitude for each net in the design is the same (V_{cc}), we can define a parameter called dynamic capacitance (C_{dyn}), which is the switching energy normalized by V_{cc}. The dynamic capacitance of a gate i (C_{dyn_i}) is

$$C_{\mathrm{dyn}_i} = \frac{\text{Switching Energy}}{\mathrm{V}cc^2} = \sum_{j=1}^{M_i} \mathrm{AF}_i^j \cdot C_j + \mathrm{AF}_{\mathrm{out}_i} \cdot C_{\mathrm{out}_i}, \quad (7)$$

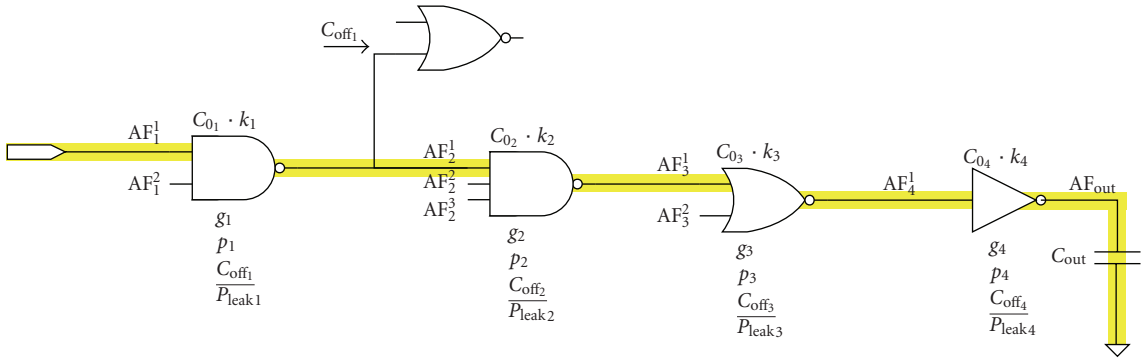


FIGURE 3: Example path. Each gate is assigned with logical effort notation, initial input capacitance (C_{0_i}) and sizing factor (k_i).

Without loss of generality, we assume that the first input of each gate resides on the investigated path. We assume that the inputs of the gates we deal with are symmetrical (input capacitance on each input pin is equal) and the gates are noncompound (i.e., gates implementing functions like $a \cdot b + c$ are out of scope). Our method can be easily extended to support these types. Under these assumptions, all input capacitances of a given gate are identical. Therefore, the input C_{dyn} of gate i ($C_{dyn_{in}} i$) is

$$\begin{aligned} C_{dyn_{in}} i &= C_{0_i} \cdot k_i \sum_{j=1}^{M_i} AF_i^j \\ &= C_{0_i} \cdot k_i \cdot AF_i, \end{aligned} \tag{8}$$

where AF_i is defined to be $\sum_{j=1}^{M_i} AF_i^j$—sum of activity factors for input pins of gate i. Note that unlike calculating the delay of a gate, when calculating the gate energy, all input and output nets of a gate have to be taken into consideration. The C_{dyn} of the nets not in the desired path should not be overlooked.

The output capacitance of a gate is defined to be its self loading and is combined mainly of the drain diffusion capacitors connected to the output. The parasitic delay of gate i in logical effort method, denoted by p_i, is proportional to the diffusion capacitance. The logical effort of gate i, denoted by g_i, expresses the ratio of the input capacitance of gate i to that of an inverter capable of delivering the same current. It is easy to see that the output capacitance of gate i can be expressed as

$$C_{out_i} = \frac{C_{in_i}}{g_i} p_i. \tag{9}$$

We can now rewrite (7) using the notation defined previously:

$$C_{dyn_i} = C_{0_i} k_i \cdot AF_i + \frac{C_{0_i} k_i}{g_i} p_i \cdot AF_o^i. \tag{10}$$

Besides the gates in the path, we have to take into account the C_{dyn} of the side loads. Multiplying C_{off_i} by AF_i^1 results in the C_{dyn} of the off-path load driven by gate i. We use (10) to calculate C_{dyn} of a desired path:

$$\begin{aligned} C_{dyn} = \underbrace{AF_1 \cdot C_{in_1}}_{\text{input } C_{dyn}} + \sum_{i=2}^{N} \underbrace{AF_i \cdot C_{in_i} + AF_o^{i-1} \cdot C_{out_{i-1}}}_{\text{stage } i\ C_{dyn}} \\ + \underbrace{AF_o^N (C_{out_N} + C_{load})}_{\text{output } C_{dyn}} + \sum_{i=1}^{N} \underbrace{C_{off_i} \cdot AF_i^1}_{C_{dyn} \text{ of off path load } i}. \end{aligned} \tag{11}$$

Substituting input C_{dyn} with (8) and C_{out_i} with (9), and rearranging the formula, we get

$$\begin{aligned} C_{dyn} = \sum_{i=1}^{N} k_i \left(AF_i \cdot C_{0_i} + AF_o^i \cdot \frac{C_{0_i} \cdot p_i}{g_i} \right) + AF_{N+1} \cdot C_{load} \\ + \sum_{i=1}^{N} C_{off_i} \cdot AF_i^1. \end{aligned} \tag{12}$$

By defining

$$\begin{aligned} C_{dyn_i} &\triangleq AF_i \cdot C_{0_i} + AF_o^i \cdot \frac{C_{0_i} \cdot p_i}{g_i}, \\ C_{dyn\text{-}off} &\triangleq \sum_{i=1}^{N} C_{off_i} \cdot AF_i^1, \end{aligned} \tag{13}$$

we get

$$C_{dyn} = \sum_{i=1}^{N} C_{dyn_i} \cdot k_i + AF_{N+1} \cdot C_{load} + C_{dyn\text{-}off}. \tag{14}$$

The initial C_{dyn} is achieved by setting all $k_i's$ to 1:

$$C_{dyn}^0 \triangleq C_{dyn}\Big|_{k_i's=1} = \sum_{i=1}^{N} C_{dyn_i} + AF_{N+1} \cdot C_{load} + C_{dyn\text{-}off}. \tag{15}$$

3.1.2. Leakage Energy. The leakage energy of a static CMOS gate i with M_i inputs and a single output can be expressed as

$$\text{Leakage Energy of Gate } i = T_{cycle} \cdot \overline{P_{leak_i}} \cdot C_{0_i}, \tag{16}$$

where T_{cycle} is the cycle time of the circuit, and $\overline{P_{\text{leak}_i}}$ is the average leakage power for gate i, for a unit input capacitance. $\overline{P_{\text{leak}_i}}$ is a function of the manufacturing technology, gate topology, and signal probability (SP: the probability for a signal to be in a logical TRUE state at a given cycle) for each input. See [8, 17, 18] for leakage power calculation methods. Under a given workload, $\overline{P_{\text{leak}_i}}$ should be precalculated for each gate i. Since $\overline{P_{\text{leak}_i}}$ is sensitive to the signal probability, it needs to be recalculated whenever the workload is modified, to reflect changes in gates' signal probability.

By dividing the leakage energy by V_{cc}^2, we can express the leakage in terms of capacitance:

$$\begin{aligned}&\text{Leakage Capacitance of Gate } i\\ &\triangleq C_{\text{leak}_i} = \frac{1}{V_{cc}^2} T_{\text{leak}_i} \cdot C_{0_i} \cdot \overline{P_{\text{leak}_i}}.\end{aligned} \tag{17}$$

And the total C_{leak} is equal to:

$$\begin{aligned}C_{\text{leak}} &= \frac{1}{V_{cc}^2} T_{\text{cycle}} \sum_{i=1}^{N} \left(k_i \cdot C_{0_i} \cdot \overline{P_{\text{leak}_i}}\right)\\ &= \sum_{i=1}^{N} k_i \cdot C_{\text{leak}_i}.\end{aligned} \tag{18}$$

The initial C_{leak} is achieved by setting all $k_i's$ to 1:

$$C_{\text{leak}}^0 \triangleq C_{\text{leak}}|_{k_i's=1} = \sum_{i=1}^{N} C_{\text{leak}_i}. \tag{19}$$

By combining (14), (15), (18), and (19) we can express the total capacitance and the initial capacitance of a desired path:

$$\begin{aligned}C_{\text{path}} &= \sum_{i=1}^{N} k_i \left(C_{\text{dyn}_i} + C_{\text{leak}_i}\right) + \text{AF}_{N+1} \cdot C_{\text{load}} + C_{\text{dyn-off}},\\ C_{\text{path}}^0 &= \sum_{i=1}^{N} \left(C_{\text{dyn}_i} + C_{\text{leak}_i}\right) + \text{AF}_{N+1} \cdot C_{\text{load}} + C_{\text{dyn-off}}.\end{aligned} \tag{20}$$

The energy decrease rate (e_{dec}) due to downsizing of the gates by a factor of k is expressed as

$$\begin{aligned}e_{\text{dec}} &= \frac{C_{\text{path}}^0 - C_{\text{path}}}{C_{\text{path}}^0}\\ &= \frac{\sum_{i=1}^{N} \left(C_{\text{dyn}_i} + C_{\text{leak}_i}\right)(1 - k_i)}{\sum_{i=1}^{N} \left(C_{\text{dyn}_i} + C_{\text{leak}_i}\right) + \text{AF}_{N+1} \cdot C_{\text{load}} + C_{\text{dyn-off}}}.\end{aligned} \tag{21}$$

In order to estimate the upper bound of e_{dec}, we assume an initial design point with minimum delay for C_{path}^0, and set the sizes of the gates in the path to minimum allowed feature size (C_{min}), to reflect the minimum possible C_{path}. By defining

$$\begin{aligned}C_{\text{dyn}_i}^{\text{min}} &\triangleq \text{AF}_i \cdot C_{\text{min}} + \text{AF}_o^i \cdot \frac{C_{\text{min}} \cdot p_i}{g_i},\\ C_{\text{leak}_i}^{\text{min}} &\triangleq \frac{1}{V_{cc}^2} T_{\text{cycle}} \cdot C_{\text{min}} \cdot \overline{P_{\text{leak}_i}}.\end{aligned} \tag{22}$$

We get

$$\begin{aligned}e_{\text{dec}} &\leqslant e_{\text{dec}}^{\text{MAX}}\\ &= \frac{\sum_{i=1}^{N} \left(C_{\text{dyn}_i} + C_{\text{leak}_i}\right) - \sum_{i=1}^{N} \left(C_{\text{dyn}_i}^{\text{min}} + C_{\text{leak}_i}^{\text{min}}\right)}{\sum_{i=1}^{N} \left(C_{\text{dyn}_i} + C_{\text{leak}_i}\right) + \text{AF}_{N+1} \cdot C_{\text{load}} + C_{\text{dyn-off}}}.\end{aligned} \tag{23}$$

By using (23), the upper bound to the EDG at a given delay increase rate (d_{inc})—$\text{EDG}_{(d_{\text{inc}})}^{\text{MAX}}$ can also be calculated, simply by dividing $e_{\text{dec}}^{\text{MAX}}$ by d_{inc}:

$$\text{EDG}_{(d_{\text{inc}})}^{\text{MAX}} = \frac{e_{\text{dec}}^{\text{MAX}}}{d_{\text{inc}}}. \tag{24}$$

$\text{EDG}_{(d_{\text{inc}})}^{\text{MAX}}$ can be used by the circuit designer to quickly evaluate the potential for saving power. However, the designer should note that the value of $\text{EDG}_{(d_{\text{inc}})}^{\text{MAX}}$ is a nonreachable upper bound since the minimum sizing leads to a delay increase which is always greater than the one that the designer refers to. If the value of $\text{EDG}_{(d_{\text{inc}})}^{\text{MAX}}$ is not sufficient, other energy reduction techniques should be considered.

3.2. Delay of a Logic Path. When using the logical effort notation, the path delay (D) is expressed as

$$D = \sum_{i=1}^{N} g_i h_i + P. \tag{25}$$

The electrical effort of stage i (h_i) is calculated as the ratio between capacitance of gate $i+1$ and gate i, plus the ratio of side load capacitance of gate i and input capacitance of gate i. For the sake of simplicity, k_{N+1} and k_1 are defined to be 1. Using the notation defined earlier, the path delay D can be written as:

$$D = \sum_{i=1}^{N} g_i \left(\frac{C_{0_{i+1}} k_{i+1}}{C_{0_i} k_i} + \frac{C_{\text{off}_i}}{C_{0_i} k_i}\right) + P. \tag{26}$$

By defining

$$\begin{aligned}D_i^0 &\triangleq g_i \frac{C_{0_{i+1}}}{C_{0_i}},\\ D_i^1 &\triangleq g_i \frac{C_{\text{off}_i}}{C_{0_i}}.\end{aligned} \tag{27}$$

Equation (26) becomes

$$D = \sum_{i=1}^{N} \left(D_i^0 \frac{k_{i+1}}{k_i} + D_i^1 \frac{1}{k_i}\right) + P. \tag{28}$$

The initial delay is achieved by setting all $k_i's$ to 1.

$$D_0 \triangleq D|_{k_i's=1} = \sum_{i=1}^{N} \left(D_i^0 + D_i^1\right) + P. \tag{29}$$

And therefore, the delay increase rate (d_{inc}) due to downsizing of the gates by a factor of k_i is

$$d_{\text{inc}} = \frac{D - D_0}{D_0} = \frac{\sum_{i=1}^{N}\left(D_i^0 k_{i+1}/k_i + D_i^1 1/k_i\right) + P - D_0}{D_0}. \tag{30}$$

4. Optimizing Power and Performance

Given a delay value that is d_{inc} percent greater than the initial delay D_0, we seek the path sizing $(C_{0_2} \cdot k_2 \cdots C_{0_N} \cdot k_N)$ that maximizes the energy reduction rate e_{dec}.

From (21), maximizing e_{dec} is achieved by minimizing C_{dyn}. By ignoring the factors that do not depend on k_i and will not affect the optimization process in (20), we define an objective function f_0:

$$f_0 = \sum_{i=1}^{N} k_i\left(C_{\text{dyn}_i} + C_{\text{leak}_i}\right). \tag{31}$$

Note that f_0 depends linearly on the dynamic and the leakage capacitances, which apply weights and determine the importance of each k_i. Equation (31) can also be written as:

$$f_0 = \sum_{i=1}^{N} k_i C_{0_i}\left(\frac{1}{V_{cc}^2} T_{\text{cycle}} \overline{P_{\text{leak}_i}} + \text{AF}_i + \text{AF}_o^i \cdot \frac{p_i}{g_i}\right). \tag{32}$$

Note that when all gates in a path are of the same type, all activity factors are equal, and average leakage power for all gates in the path is equal, both C_{dyn_i} and C_{leak_i} can be eliminated from (31) without affecting the optimization result. These conditions are satisfied on an inverter chain with input signal probability of 0.5, for instance. In this case, the leakage power of activity factor has no influence on the optimization result.

To get a canonical constraint goal, in which the constraint is less than or equal 1, we rearrange (30) to

$$\sum_{i=1}^{N}\left(D_i^0 \frac{k_{i+1}}{k_i} + D_i^1 \frac{1}{k_i}\right) = d_{\text{inc}} D_0 + D_0 - P, \tag{33}$$

and define

$$D_i^{'0} \triangleq \frac{D_i^0}{d_{\text{inc}} D_0 + D_0 - P}, \qquad D_i^{'1} \triangleq \frac{D_i^1}{d_{\text{inc}} D_0 + D_0 - P}, \tag{34}$$

to get

$$\sum_{i=1}^{N}\left(D_i^{'0} \frac{k_{i+1}}{k_i} + D_i^{'1} \frac{1}{k_i}\right) = 1. \tag{35}$$

We now can use (35) to get an optimization constraint

$$f_1 = \sum_{i=1}^{N}\left(D_i^{'0} \frac{k_{i+1}}{k_i} + D_i^{'1} \frac{1}{k_i}\right) \leqslant 1. \tag{36}$$

Combining (32) and (31) results in the following optimization problem:

Minimize $f_0(k_1 \cdots k_N)$,

subject to $f_1(k_1 \cdots k_N) \leqslant 1$, where

$$f_0(k_1 \cdots k_N) = \sum_{i=1}^{N} k_i\left(C_{\text{dyn}_i} + C_{\text{leak}_i}\right), \tag{37}$$

$$f_1(k_1 \cdots k_N) = \sum_{i=1}^{N}\left(D_i^{'0} \frac{k_{i+1}}{k_i} + D_i^{'1} \frac{1}{k_i}\right).$$

However, f_1 defined above is nonconvex. We use geometrical programming [19–21] to solve the optimization problem, by changing variables

$$\begin{aligned} \widetilde{k_i} &= \log(k_i) \Longrightarrow k_i = e^{\widetilde{k_i}}, \\ \widetilde{C_{\text{dyn}_i}} &= \log\left(C_{\text{dyn}_i}\right) \Longrightarrow C_{\text{leak}_i} = e^{\widetilde{C_{\text{leak}_i}}}, \\ \widetilde{C_{\text{leak}_i}} &= \log(C_{\text{leak}_i}) \Longrightarrow C_{\text{dyn}_i} = e^{\widetilde{C_{\text{dyn}_i}}}, \\ \widetilde{D_i^{'0}} &= \log\left(D_i^{'0}\right) \Longrightarrow D_i^{'0} = e^{\widetilde{D_i^{'0}}}, \\ \widetilde{D_i^{'1}} &= \log\left(D_i^{'1}\right) \Longrightarrow D_i^{'1} = e^{\widetilde{D_i^{'1}}}. \end{aligned} \tag{38}$$

So the equivalent convex optimization problem (which can be solved using convex optimization tools) is:

Minimize $\widetilde{f}_0(k_1 \cdots k_N)$,

subject to $\widetilde{f}_1(k_1 \cdots k_N) \leqslant 0$, where

$$\begin{aligned} \widetilde{f}_0(k_1 \cdots k_N) &= \log\left(\sum_{i=1}^{N} e^{\widetilde{k_i} + \widetilde{C_{\text{dyn}_i}}} + e^{\widetilde{k_i} + \widetilde{C_{\text{leak}_i}}}\right), \\ \widetilde{f}_1(k_1 \cdots k_N) &= \log\left(\sum_{i=1}^{N} e^{\widetilde{k}_{i+1} - \widetilde{k_i} + \widetilde{D_i^{'0}}} + e^{\widetilde{D_i^{'1}} - \widetilde{k_i}}\right). \end{aligned} \tag{39}$$

The convexity of (39) ensures that a solution to the optimization problem exists, and that the solution is the global optimum point. In order to obtain the EDG curve, the delay increase rate is swept from 0 to the desired value, and for each delay increase value, a different optimization problem is solved by geometrical programming.

This result can be extended to handle circuit delay, instead of a single path delay. All paths must be enumerated, and the optimized delay should reflect the critical path delay. The critical path delay is calculated as the maximum delay of all enumerated paths. However, the MAX operator cannot be handled directly in geometrical programming, since it produces a result which is not necessarily differentiable. Boyd et al. [20] solve the general problem of using the MAX operator in geometrical programming $(\text{MAX}(f_1(x), f_2(x) \ldots f_N(x)) \leqslant 1)$

by introducing a new variable t, and N inequalities (N being the number of paths), to obtain

$$\begin{aligned} t &\leqslant 1, \\ f_1(x) &\leqslant t, \\ f_2(x) &\leqslant t, \\ &\cdots \\ f_N(x) &\leqslant t. \end{aligned} \tag{40}$$

This transformation can be used in order to feed the critical path into the optimizer. To calculate the energy-delay tradeoff, the C_{dyn} of the entire circuit should be taken into account.

In the following sections, we employ this procedure to characterize the EDG and power reduction in typical logic circuits, and derive design guidelines.

5. Exploring Energy-Delay Tradeoff in Basic Circuits

We run numerical experiments that explore the EDG of some basic circuits. We use GGPLAB [22] as a geometrical programming optimizer, to solve the optimization problem (37) and (39). GGLAB is a free open source library and can be easily installed over Matlab. For each experiment, we provide an EDG curve which is obtained by optimizing the circuit for a wide range of increased delay values. Although the propagation delay and the active energy dissipation are technology independent, the leakage depends on the manufacturing technology and the circuit's cycle time. Throughout this section, the leakage is calculated according to the 32 nm technology node of the ITRS 2007 projection [23], in which $C_{\text{leak}_{\text{inv}}}$ is calculated to be 0.5694, based on clock frequency of 2 GHz and signal probability of 0.5.

5.1. Inverter Chain. Consider a chain consisting of N inverters, with output load of C_{out}. C_{0_1} is set arbitrarily to a constant value of 1 ff, and therefore the path electrical effort (H) is C_{out} (Figure 4). We set initial gate capacitances ($C_{0_2} \cdots C_{0_N}$) that ensure minimum delay, using the logical effort methodology. The minimum delay was obtained by setting the electrical effort to be the Nth root of the path electrical effort. The leakage calculation takes into account the signal probability of the inverters in the chain.

Figure 5(a) shows the EDG for different combinations of path electrical effort (H) and chain length (N) where the leakage energy is negligible. Figure 5(b) shows the same analysis, for negligible dynamic energy. In both cases, the largest potential for energy savings occurs near the point where the design is sized for minimum achievable delay. The potential for energy savings decreases as the delay is being relaxed further. This is consistent with the observation in [5].

Figure 6 shows the optimal sizing of a fixed input and output load inverter chain with an arbitrary activity factor and signal probability of 0.5, for various delay increase values. For input signal probability of 0.5, all the gates in the inverter chain have the same signal probability. Therefore, the optimization process is indifferent to the average leakage power of each gate—P_{leak_i} in (17) is constant and can be eliminated from (37).

The optimization process leads to increasing the electrical effort of the last stages and decreasing the electrical effort of the first stages, to meet the timing requirements (Figure 6(f)). The largest energy savings, for a given delay increase value, are achieved by downsizing the largest gates in the chain (Figure 6(e)). The relative downsizing, however, is maximal around the middle of the chain (Figure 6(c)), due to the fact that the first stage and the load are anchored with a fixed size. In order to understand the behavior of the middle stages, a 16-stage inverter stage simulation is plotted

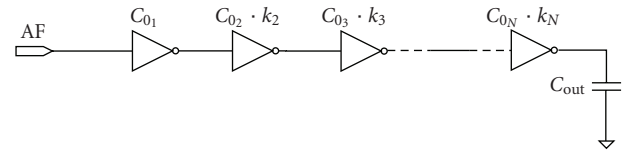
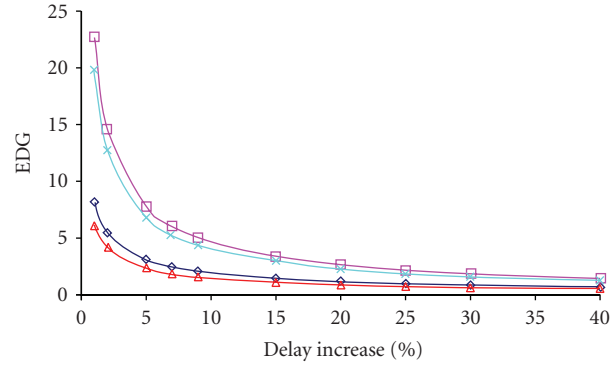
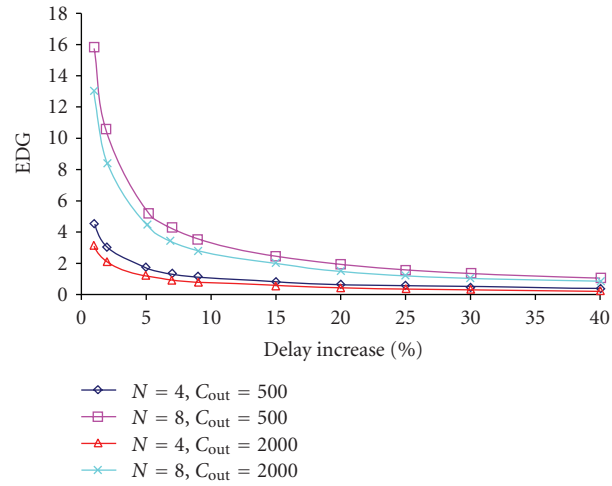


Figure 4: Inverter chain—consists of N stages, output load C_{out}, and initial capacitances ($C_{0_1} \cdots C_{0_N}$)



(a) Energy delay gain, active dominant circuit



(b)

Figure 5: Inverter chain—various loads (C_{out}) and chain length (N).

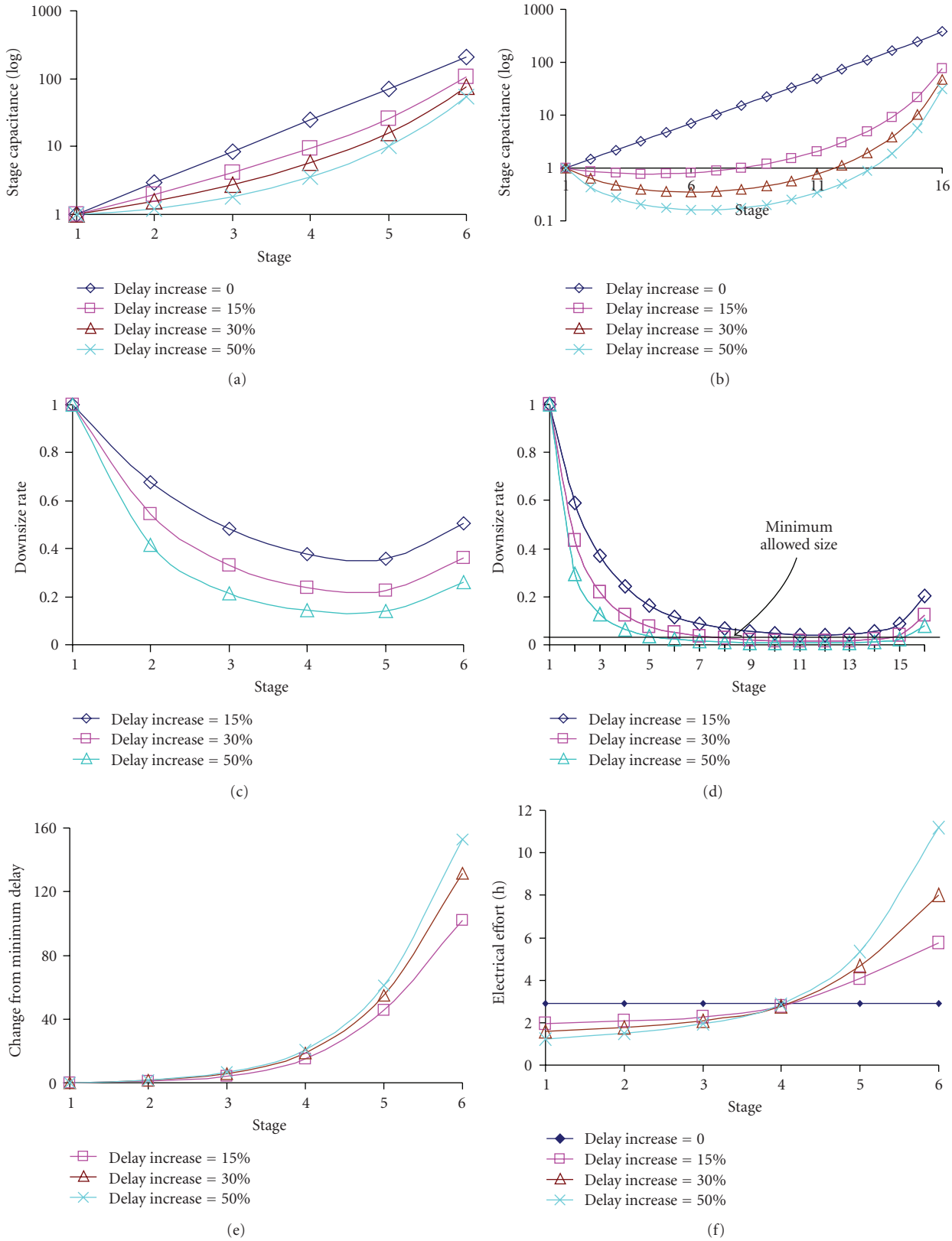


FIGURE 6: Inverter chain—sizing of the stages in an inverter chain. (a) Stage capacitance (chain of 6 inverters), for various delay increase rates (log scale). (b) Stage capacitance (chain of 16 inverters), for various delay increase rates (log scale). (c) Stage sizing factor (chain of 6 inverters): ratio of gate capacitance to minimum delay capacitance, needed to meet the given delay increase rates value. (d) Stage sizing factor (chain of 16 inverters): ratio of gate capacitance to minimum delay capacitance, needed to meet the given delay increase rates value. (e) Stage downsizing value: change in gate capacitance with respect to minimum delay sizes to meet the given delay increase rates value. (f) Stage electrical effort (h), for various delay increase rates.

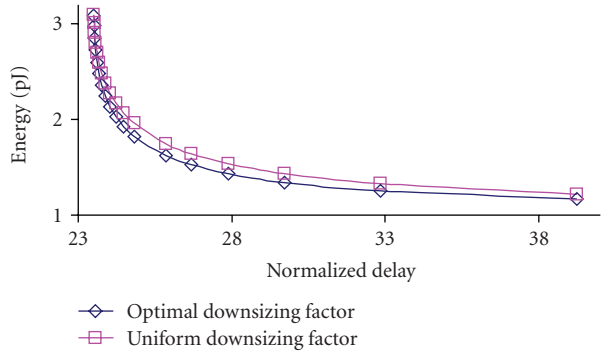


FIGURE 7: Uniform versus. Optimal downsizing. Linear downsizing of an inverter chain in order to save energy by increasing the delay results in a nonoptimal design—in this case 7% more energy could be saved by tuning the sizing correctly.

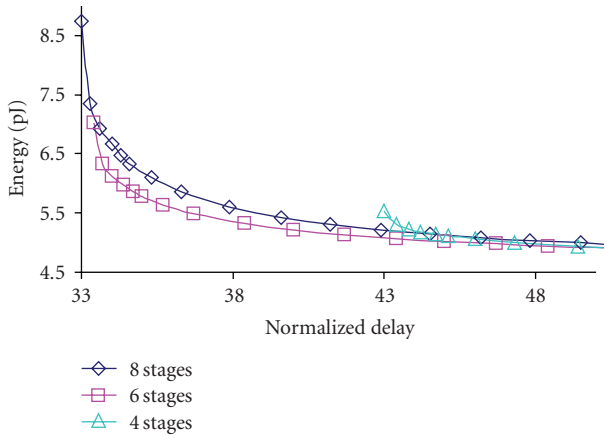


FIGURE 8: Inverter chain—variable Length: the chain length is varied in order to save a maximal amount of energy for each delay value.

in Figure 6(d). As the delay increases, the gates towards the middle of the chain are downsized and form a plateau-like shape. Note that the optimal gate sizes might be limited by the minimum allowed size according to design rules.

Both Figures 6(a) and 6(b) (absolute sizing) and Figure 6(f) illustrate that as we move further from the minimal achievable delay (delay increase = 0, where all electrical efforts are identical), the difference between the electrical efforts of the stages increases. However, uniform downsizing (e.g., increase the delay by downsizing each gate by 5%) is sometimes used in the power reduction process by the circuit designer as an easy and straightforward method to trade off energy for performance. Figure 7 shows the energy efficient curve (optimal sizing) versus energy-delay curve generated by uniform downsizing of an 8-long inverter chain with out/in capacitance ratio of 200. The energy difference between the curves in the figure reaches up to 7%.

Most of the energy in the path is dissipated in the last stages of the chain, where the fanout factors are larger, in order to drive the large fixed output capacitance.

Figure 8 demonstrates the effect of chain length on delay and energy. The external load of the circuit is relatively large—9pF, for which 8-long chain yields an optimal timing. The energy efficient curves for chains of 8, 6, and 4 inverters are plotted in the energy-delay plane. We can see that the number of stages is important when the optimal delay is required. Generally, as we move further from the smallest achievable delay, fewer inverters achieve better energy dissipation for the same delay. However, the difference in energy between the optimal number of inverters and a fixed number of inverters decreases as the delay is relaxed.

Figure 9 shows good correlation between $\mathrm{EDG}_{10\%}^{\mathrm{MAX}}$ see (24), and the actual energy delay gain. The energy saving opportunity increases when the output load is small, and when the number of stages in the path increases.

5.2. Activity and Signal Probability Effect on Sizing. The more active a gate is, the more energy it consumes. In order to trade off delay and energy better, active gates in the timing critical path can be downsized more than inactive gates in the critical path. For instance, consider the circuit in Figure 10. The path from A to out is the timing critical. Input A has a fixed activity factor of 0.5, while the activity factor of input B is varied. In order to calculate the activity factor and signal probability of internal nodes, the method described in [24] for AF/SP propagation in combinational circuits is used—for a NAND gate with uncorrelated inputs A and B and output O, the activity at its output is calculated as:

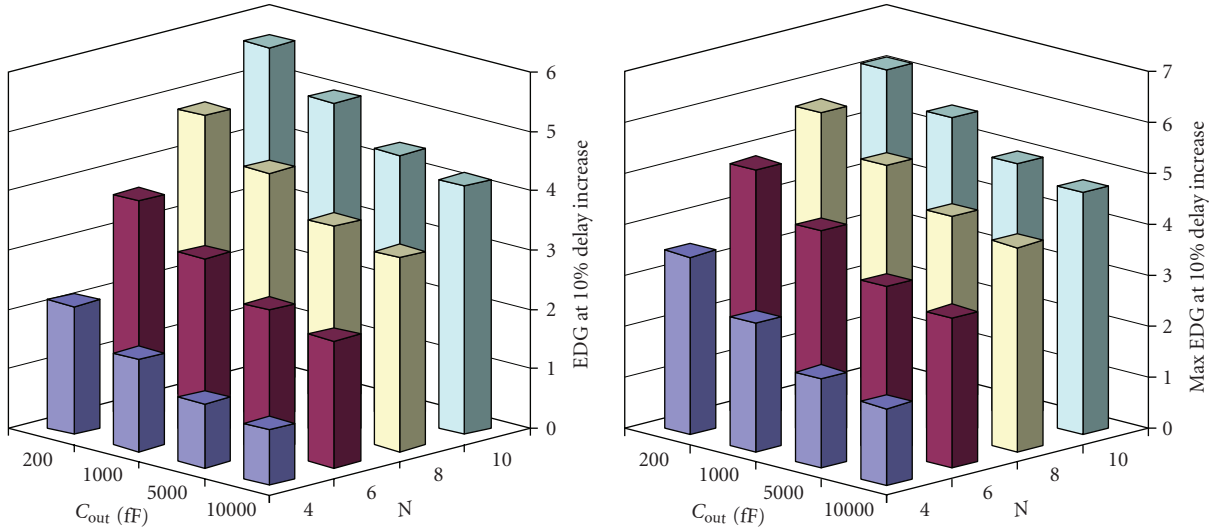
$$\mathrm{AF}_O = \mathrm{AF}_A \cdot \mathrm{SP}_B + \mathrm{AF}_B \cdot SP_A - \frac{1}{2} \cdot \mathrm{AF}_A \cdot \mathrm{AF}_B. \quad (41)$$

According to (41), the activity factor at the NAND's gate output is $\mathrm{AF}_{\mathrm{nand}} = 0.25 + 0.5\mathrm{AF}_B$—the activity factor at the output of the NAND is controlled by the activity factor of input B, and monotonically rises as AF_B increases.

When the delay constrains of the circuit are relaxed, As AF_B is increased, and with it $\mathrm{AF}_{\mathrm{nand}}$, we expect that the gates that are driven by the NAND gate will get downsized at the expense of the gates driving the NAND gate. Figure 11 shows the sizing factor of each gate for various AF_B values, for a delay increase rate of 20%. We see that as AF_B increases, the sizing factor of gates 1 and 2 is increased, while the sizing factor of gates 5 and 6 is decreased.

A similar observation holds for leakage dominant circuits, where the signal probability becomes the affecting parameter instead of the activity factor. $\overline{P_{\mathrm{leak}_i}}$ in (16) depends on the signal probability. Therefore, it is expected that the sizing of each gate during the optimization process will be influenced by the signal probability at the gate input. For example, in an inverter, where the Pmos transistor's size is twice the size of the Nmos transistor, the leakage power of a single inverter can be estimated by:

$$\begin{aligned}&\text{Inverter Leakage Power}\\ &= \mathrm{SP} \cdot C_{\mathrm{in}} \cdot \frac{2}{3} \cdot P_{\mathrm{leak}}(P\mathrm{mos})\\ &\quad + (1 - \mathrm{SP}) \cdot C_{\mathrm{in}} \cdot \frac{1}{3} \cdot P_{\mathrm{leak}}(N\mathrm{mos}),\end{aligned} \quad (42)$$



(a) EDG value of various inverter chains at delay increase rate of 10%

(b) EDG^{MAX} (analytical upper bound to EDG) of various inverter chains at delay increase rate of 10%

FIGURE 9: Inverter Chain—comparison between energy delay gain and EDG^{MAX} (analytical upper bound to EDG).

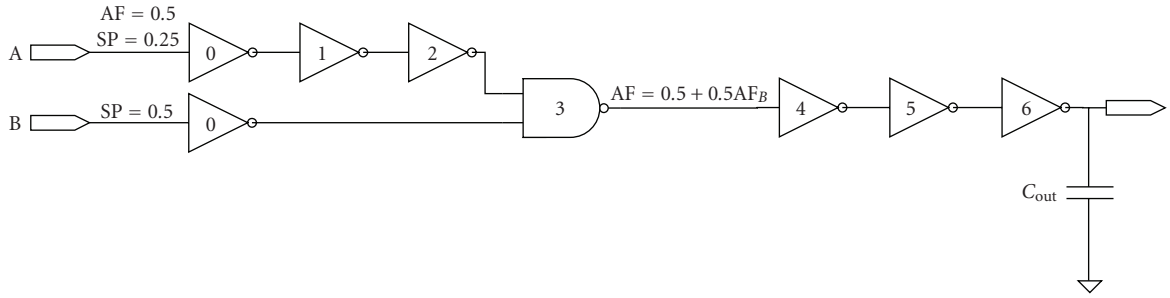


FIGURE 10: Activity effect on sizing the path from a to end is timing critical, and the activity of input b is varied.

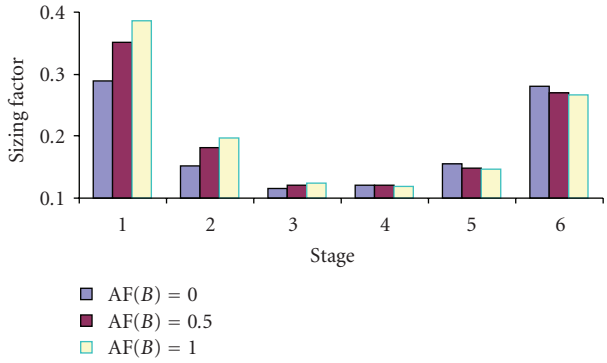


FIGURE 11: Sizing factor to achieve 20% delay increase as AF_b increases, the sizing factor of gates 1 and 2 is increased, while the sizing factor of gates 5 and 6 is decreased.

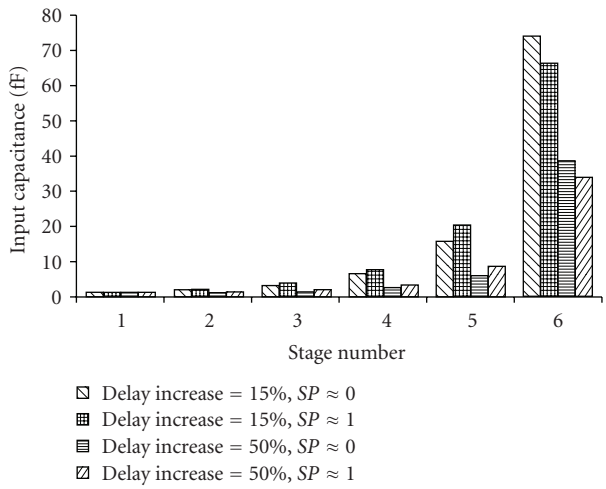


FIGURE 12: Sizing of 6-stages inverter chain as a function of SP the sizing of the stages is sensitive to the signal probability changes, both for small and large delay increase values.

where SP is the signal probability in the input of the inverter, C_{in} is the input capacitance of the inverter, and $P_{leak}(N\text{mos}, P\text{mos})$ is the leakage power of Nmos and Pmos transistors respectively, per unit input capacitance. Figure 12 shows the sizes of the gates in a six-stage inverter chain with input capacitance of 1 ff and output load of 600 ff with a small activity factor, when the delay increase rate is varied from 0% to 50%. The optimal sizing at each stage is clearly affected

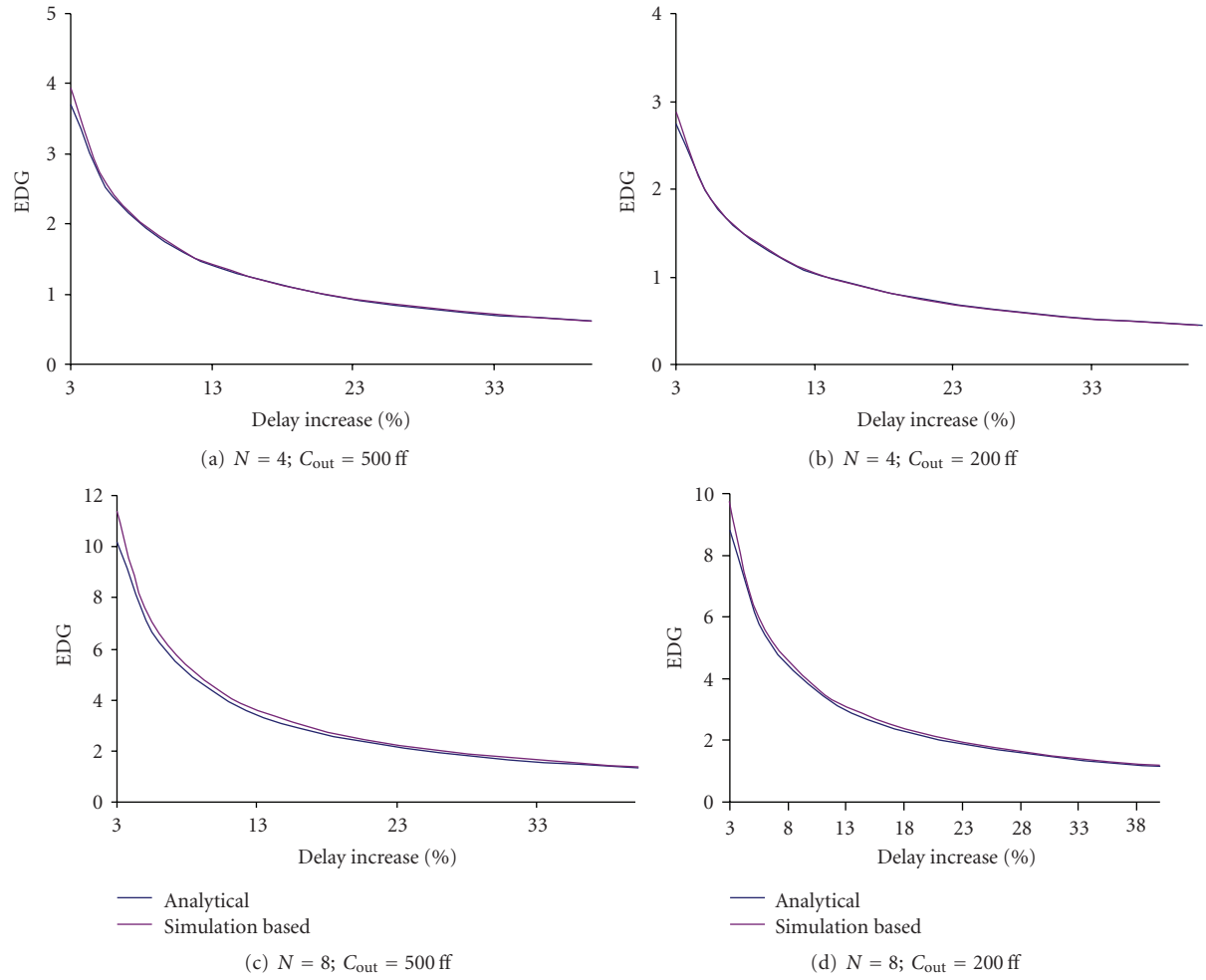


Figure 13: Simulation optimization of inverter chain with comparison to theoretical computation.

by the signal probability. Up to 50% difference in the sizing of the stages as a function of the signal probability can be observed (see delay increase of 50%, 4th stage).

5.3. Comparing Analytical and Simulation-Based Optimization. In order to validate the correctness of the EDG optimization algorithm, the results of Section 5.1 are compared to simulation results. The simulation was performed using a proprietary circuit simulator combined with a proprietary numerical optimization environment, in a 32 nm process. The circuit was first optimized for minimum delay, which was used later as a reference. In order to get the EDG curves, the circuit was optimized by the simulation-based tool for minimum energy, for several delay constraints.

Figure 13 presents the difference between the analytical computation (Section 5.1) and the simulation-based optimization. The error is small, and ranges from a maximum of 7% to a minimum of 0̃%. Obtaining the EDG curves using simulation-based optimization is orders of magnitude slower than running the proposed analytical method. Table 1 compares the run time of simulation-based optimization and the run time of the proposed analytical model for few inverter chain circuits. Note that simulation-based optimization run time increases dramatically as the circuit complexity increases.

Table 1: Comparison of run time-simulation-based and analytical model optimization. The table compares the amount of time taken in order to generate an EDG plot-consists of ten delay increase points.

Circuit	Sim-based optimization	Analytical model optimization
4-long Inverter Chain	240 sec	25 sec
8-long Inverter Chain	360 sec	40 sec
15-long Inverter Chain	1100 sec	70 sec

The analytical model was calibrated by computing the parasitics delay of an inverter (p) for the given technology, simply by comparing the output capacitance to the input capacitance of an unloaded inverter (see (9)).

6. Final Remarks and Conclusion

We have presented a design optimization framework that explores the power-performance space. The framework provides fast and accurate answers to the following questions.

(1) How much power can be saved by slowing down the circuit by x percent?

(2) How to determine gate sizes for optimal power under a given delay constraint?

We introduced the energy/delay gain (EDG) as a metric for the amount of energy that can be saved as a function of increased delay. The method was demonstrated on a variety of circuits, exhibiting good correlation with accurate simulation-based optimizations. We have shown that around 25% dynamic energy can be gained when the delay constraint is relaxed by 5% in an optimal way, for circuits in 32 nm technology which were initially designed for maximal operation speed. An upper bound of power savings in a given circuit can be obtained without optimization, in order to quickly assess whether a downsizing effort may be justified for the circuit.

The method described in this work can be used by both circuit designers and EDA tools. Circuit designers can increase their intuition of the energy-delay tradeoff. The following rules of thumb can be derived from the experiments.

(i) Minimum Delay Is Power Expensive. By relaxing the delay, significant amount of dynamic energy could be saved. We have shown that under given conditions, for a 2-bit multiplexer up to 40% of dynamic energy could be saved when the delay constraint is relaxed by 10%.

(ii) A fixed Uniform Downsizing Factor for all Gates in the circuit would lead to an inefficient design in terms of energy. The optimal downsizing factor is not uniform.

(iii) Increase delay by downsizing the "middle" gates. In order to save energy with minimal impact on timing-the gates located in the middle (between he input and the load) are downsized the most. The downsizing factor increases as the delay constraint relaxes.

(iv) Increase Delay by Increasing the Electrical Effort towards the load. Minimum delay design requires a constant tapering factor. Typically, a "fanout of 4" is used [1]. Minimum energy design (when neglecting short circuit power) requires high tapering factor, that decreases the number of stages. When performance is compromised to save energy, the tapering factor of the stages must *increase* towards the external load. The tapering factor increases as the delay constraint is relaxed. Note that this result is applicable only when the external load is larger than the input capacitance.

(v) Downsizing of the Gates Reduces Both Dynamic energy and Leakage Energy Dissipation. Both dynamic energy and leakage energy dissipation depend linearly on the size of the gates. By downsizing the gates, both dynamic and leakage energy are reduced.

(vi) The Power Optimization Has to Be Performed under a Given Work-load. The activity factor and signal probability influence the optimized circuit's sizing. Different tests may result in different sizing. Using random tests, rather then typical tests to optimize the circuit may lead to sub-optimal design.

Acknowledgments

The authors would like to thank Yoad Yagil for his valuable inputs.

References

[1] I. E. Sutherland, R. F. Sproull, and D. F. Harris, *Logical Effort: Designing Fast CMOS Circuits*, Morgan Kauffmann, Boston, Mass, USA, 1999.

[2] P. I. Penzes and A. J. Martin, "Energy-delay efficiency of VLSI computations," in *Proceedings of the 12th ACM Great Lakes symposium on VLSI (GLSVLSI '02)*, April 2002.

[3] V. Zyuban and P. N. Strenski, "Balancing hardware intensity in microprocessor pipelines," *IBM Journal of Research and Development*, vol. 47, no. 5-6, pp. 585–598, 2003.

[4] V. Zyuban and P. Strenski, "Unified methodology for resolving power-performance tradeoffs at the microarchitectural and circuit levels," in *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 166–171, Monterey, Calif, USA, August 2002.

[5] D. Marković, V. Stojanović, B. Nikolić, M. A. Horowitz, and R. W. Brodersen, "Methods for true energy-performance optimization," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 8, pp. 1282–1293, 2004.

[6] C. P. Chen, C. C. N. Chu, and D. F. Wong, "Fast and exact simultaneous gate and wire sizing by Lagrangian relaxation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 7, pp. 1014–1025, 1999.

[7] R. Gonzalez, B. M. Gordon, and M. A. Horowitz, "Supply and threshold voltage scaling for low power CMOS," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 8, pp. 1210–1216, 1997.

[8] Z. Chen, M. Johnson, L. Wei, and K. Roy, "Estimation of standby leakage power in CMOS circuits considering accurate modeling of transistor stacks," in *Proceedings of the International Symposium on Low Power Design*, pp. 239–244, 1998.

[9] L. Benini, G. de Micheli, and E. Macii, "Designing low-power circuits: practical recipes," *IEEE Circuits and Systems Magazine*, vol. 1, no. 1, pp. 6–25, 2001.

[10] V. Khandelwal and A. Srivastava, "Leakage control through fine-grained placement and sizing of sleep transistors," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD '04)*, pp. 533–536, 2004.

[11] S. Iman and M. Pedram, *Logic Synthesis for Low Power VLSI Designs*, Springer, New York, NY, USA, 1998.

[12] R. Zlatanovici and B. Nikolić, "Power—performance optimization for custom digital circuits," in *Proceedings of the Power—Performance Optimization for Custom Digital Circuits (PATMOS '05)*, vol. 3728 of *Lecture Notes in Computer Science*, pp. 404–414, 2005.

[13] H. Q. Dao, B. R. Zeydel, and V. G. Oklobdzija, "Energy optimization of pipelined digital systems using circuit sizing and supply scaling," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 2, pp. 122–134, 2006.

[14] V. Oklobdzija and R. K. Krishnamurthy, *High-Performance Energy-Efficient Microprocessor Design*, Springer, New York, NY, USA, 2006.

[15] A. Naveh, E. Rotem, A. Mendelson et al., "Power and thermal management in the Intel core duo processor," *Intel Technology Journal*, vol. 10, no. 2, 2006.

[16] AMD Press Release, "AMD Phenom X4 9100e processor enables full featured, sleek and quiet quad-core PCs," AMD Press Resources Web Page, March 2008.

[17] H. Rahman and C. Chakrabarti, "A leakage estimation and reduction technique for scaled CMOS logic circuits considering gate-leakage," in *Proceedings of the IEEE International Symposium on Cirquits and Systems (ISCAS '04)*, pp. 297–300, May 2004.

[18] Y. Xu, Z. Luo, and X. Li, "A maximum total leakage current estimation method," in *Proceedings of the IEEE International Symposium on Cirquits and Systems (ISCAS '04)*, pp. 757–760, May 2004.

[19] S. Boyd, *Lieven Vandenberghe Convex Optimization*, Cambridge University Press, Cambridge, UK, 2006.

[20] S. Boyd, S. J. Kim, L. Vandenberghe, and A. Hassibi, "A Tutorial on Geometric Programming," Revised for Optimization and Engineering, July 2005.

[21] S. P. Boyd, S. J. Kim, D. D. Patil, and M. A. Horowitz, "Digital circuit optimization via geometric programming," *Operations Research*, vol. 53, no. 6, pp. 899–932, 2005.

[22] A. Mutapcic, K. Koh, S. Kim, L. Vanden-Berghe, and S. Boyd, "GGPLAB: A Simple Matlab Toolbox for Geometric Programming," May 2006, http://www.stanford.edu/boyd/ggplab/.

[23] "International Technology Roadmap for Semiconductors," 2007 Edition, http://www.itrs.net/Links/2007ITRS/Home2007.htm.

[24] A. Ghosh, S. Devadas, K. Keutzer, and J. White, "Estimation of average switching activity in combinational and sequential circuits," in *Proceedings of the 29th ACM/IEEE Conference on Design Automation*, pp. 253–259, Anaheim, Calif, USA, June 1992.

Hindawi Publishing Corporation
VLSI Design
Volume 2011, Article ID 731957, 10 pages
doi:10.1155/2011/731957

Review Article

SoC: A Real Platform for IP Reuse, IP Infringement, and IP Protection

Debasri Saha and Susmita Sur-Kolay

Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata 700108, India

Correspondence should be addressed to Debasri Saha, debasri_r@isical.ac.in

Received 12 October 2010; Revised 4 January 2011; Accepted 24 January 2011

Academic Editor: Shiyan Hu

Increased design complexity, shrinking design cycle, and low cost—this three-dimensional demand mandates advent of system-on-chip (SoC) methodology in semiconductor industry. The key concept of SoC is reuse of the intellectual property (IP) cores. Reuse of IPs on SoC increases the risk of misappropriation of IPs due to introduction of several new attacks and involvement of various parties as adversaries. Existing literature has huge number of proposals for IP protection (IPP) techniques to be incorporated in the IP design flow as well as in the SoC design methodology. However, these are quite scattered, limited in possibilities in multithreat environment, and sometimes mutually conflicting. Existing works need critical survey, proper categorization, and summarization to focus on the inherent tradeoff, existing security holes, and new research directions. This paper discusses the IP-based SoC design flow to highlight the exact locations and the nature of infringements in the flow, identifies the adversaries, categorizes these infringements, and applies strategic analysis on the effectiveness of the existing IPP techniques for these categories of infringements. It also clearly highlights recent challenges and new opportunities in this emerging field of research.

1. Introduction

In the recent era of automation, there are urgent needs of highly complex and application-specific multifunctional chips in every sphere of life. Customer's specification for complex chip causes explosion of gates on a single chip, advancement in process technology, and requirement for integrating heterogeneous technologies. The increased design complexity consequently needs more design effort. However, requirements for application-specific chips in every sphere mandate enhanced productivity and low cost. The only way to bridge the gap is to adopt hierarchical approach and reuse of already designed, optimized, and verified design components or fabricated and tested hardware cores to meet specification of a complex chip in time and at low cost. The way of designing an electronic system from the scratch has been replaced and system-on-chip (SoC) has emerged as an inevitable solution, where the major functional components of a complete end product are integrated into a single chip. Already-designed electronic components or fabricated hardware chips to be reused for these functional components constitute intellectual property (IP) cores. If an IP remains in electronic form, it is either a circuit description in hardware description language, that is, HDL (soft IP), may be any form of netlist, placed and routed design (firm IP), or design layout (hard IP); otherwise, it remains as hardware chip constituting a hardware IP core. A design tool is also treated as an IP. Soft IPs are more flexible but less optimized; on the other end, hardware IP cores are less flexible but more optimized. To be reused, an IP should have complete specification and proper documentation. The forms of the IPs suitable for reuse specifically on SoC will be discussed later.

An SoC usually contains reusable IPs, embedded processor(s) (a general-purpose processor and multiple special-purpose processors based on requirements) or controller(s), memory elements (SRAM, ROM, etc.), bus architecture (for interfacing IPs and other components on SoC), mixed signal blocks, programmable blocks (FPGA), voltage level shifter, clock circuits, test architecture, and so forth. An SoC may easily be enhanced by integrating more IP components with it. Furthermore, a number of SoCs can also be integrated to

realize a more complex system. The components of an SoC and the way of IP reuse in SoC environment are shown in Figure 1.

As reuse of IP is promoted in SoC environment, access control becomes essential for the IPs. In order to reuse an IP component on SoC, the SoC company should purchase the IP from its genuine vendor in legitimate way. Further, its reuse in SoC design house, in fabrication facility, and in SoC application environment should be protected. Unauthorized reuse of an IP by an SoC company and any other adversary renders loss of revenue to the genuine IP owner (IP vendor/IP creator), thus causes economic damage to the IP vendor.

An IP may be infringed during its design as well as during designing an SoC reusing that IP. So, in silicon industries, first, IP protection (IPP) techniques have been incorporated in VLSI design flow, and later on, security considerations have been extended for SoC design methodology. IPP techniques often rely on standard security mechanisms like cryptography, obfuscation, watermarking, fingerprinting, and so forth. Design concepts, system level knowledge and mechanism, design or chip level analysis, and characterization sometimes form the basis of the IPP techniques.

Section 2 discusses in detail the state of the art of IP reuse, IP infringement, and IP protection in SoC environment. Section 3 focuses on critical challenges, and Section 4 highlights the new opportunities in the field of SoC security. Conclusion appears in Section 5.

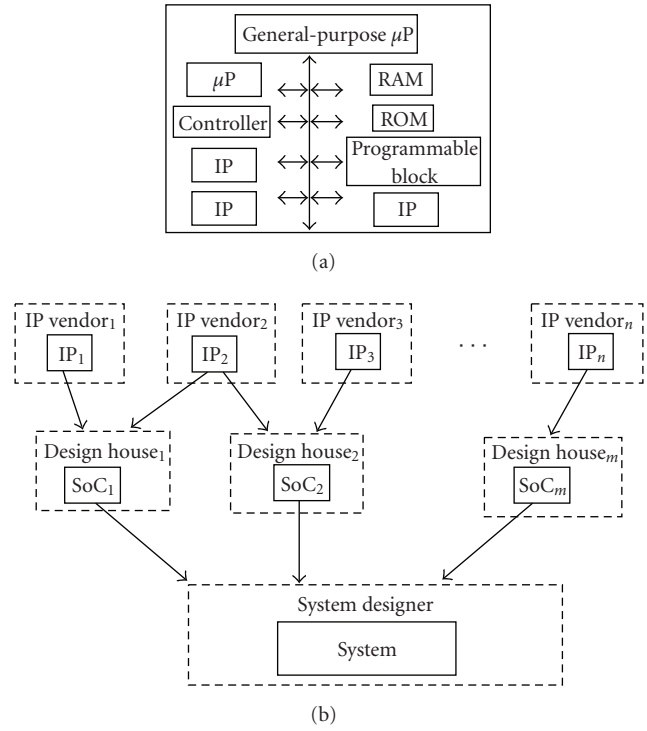


Figure 1: (a) Components on a system-on-chip, (b) IP reuse in SoC environment.

2. State of the Art

2.1. IP Reuse in SoC. Due to introduction of reuse techniques in designing an electronic system, design methodology undergoes transition from timing-driven ASIP (application-specific IP) design to block-based design of an SoC and then to platform-based design of a plug-and-play SoC. An ASIP, moderate in size and complexity, is optimized through synthesis, placement, and routing with great design effort to act as an IP component for reuse in the other two design techniques. In case of block-based design (BBD), an electronic system is partitioned into functional components, and these are mapped into available IP components (mostly firm IPs). For these firm IPs, their placements are retained, and routing, timing, and so forth are reoptimized contextwise for the overall optimization of these factors for the entire system. Finally, this software/hardware tradeoff in BBD is transformed into platform-based design (PBD), which provides an extensive and planned support to reuse of either hard IP or hardware IP on SoC. Here, based on functional requirements, IP blocks are chosen in a way so that each block can be well interfaced with its target blocks by designing suitable system architecture, their delay/power profiles satisfy constraints determined by the entire PBD, their test options are compatible with design-for-test mechanism on PBD, and those can function in one of the clock domains and voltage domains easily supported on SoC. In PBD, placement, routing, timing, delay calculation, physical verification, and test architecture construction all are performed hierarchically, with the only objective of properly designing the system (bus) architecture to realize the interface constraints [1, 2].

Reuse on SoC enhances productivity but not in a linear way; the reasons are the multiple design challenges faced in SoC design methodology and the overhead of integrating IPP techniques with the design flow. Design challenges include integration of heterogeneous device technologies and protocols, maintaining signal integrity, issues related to testing, clock timing, and voltage regulation [3, 4]. An IP is to be integrated with the other components on SoC, so it needs to satisfy several design constraints. Moreover, IPs are individually optimized, but the objective of SoC is to optimize the performance of the entire system obtained from integrating the IPs. Furthermore, it is to be noted that SoCs are mostly application specific, and, therefore, application of SoC defines the IPs. Different application needs different architecture, for example, buses, I/Os, processors, memories, and so forth. So, in order to meet constraints determined by the application of the SoC and the other components on the SoC, firm IPs are often partially redesigned. For hard IPs, one may use interface wrappers, which incur area overhead; otherwise, interface definitions of the available IPs are redesigned.

In IP-based SoC design flow, both software and hardware development are required (software/hardware codesign). Depending on the application on SoC, IPs close to requirement specifications are chosen, these are partially redesigned to be compatible for reuse on SoC, thereby transformed

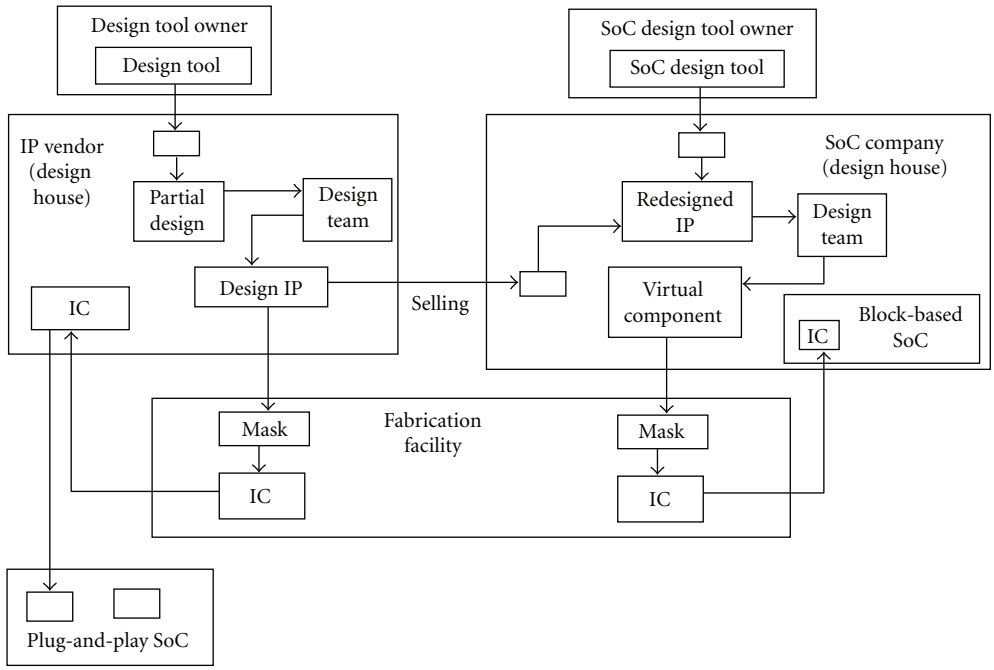


Figure 2: Movement of IP in integrated IP and SoC design flow.

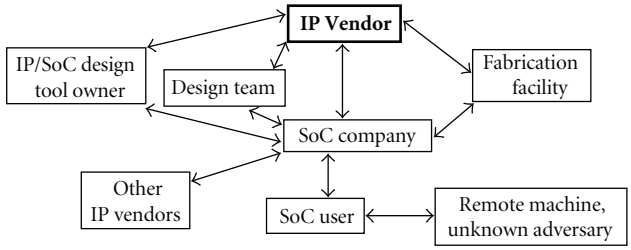


Figure 3: Interaction of IP vendor with various parties (may be adversaries).

into virtual components (VCs). VCs are emulated with programmable hardware on SoC and then fabricated to ICs. On the other hand, on SoC, bus architectures are designed to completely satisfy interface constraints, and other design works are completed, that is, placement of voltage level shifters, clock dividers, and so forth, and finally the SoC is fabricated and the firmware designers write the drivers for the components of SoCs [5].

The movement of an IP in IP-based SoC design flow is shown in Figure 2.

2.2. IP Infringement in SoC. Within an SoC company, significant design works are needed for IPs to transform those into virtual components (VCs) compatible on SoC. These VCs are then fabricated. Therefore, SoC company needs to have its own design team, design tool, and fabrication facility. However, in order to meet shrinking time to market, sometimes an SoC company hires designers and purchases SoC design tools from tool owner companies. An SoC company very often cannot afford a fabrication facility of its own. Consequently, it gets its VCs fabricated from separate fabrication foundry. So, an untrusted environment prevails between the SoC company, its hired design team, owner of the purchased design tools, and the fabrication facility used. A similar untrusted environment may occur for the IP vendor itself. Also, there are external adversaries. Following are the possible adversaries for misappropriation of an IP to be reused on SoC. Interactions of these parties with the IP vendor are shown in Figure 3.

(a) SoC company may intercept/hack an IP instead of legally purchase it from the IP vendor.

(b) An untrusted design team in IP/SoC design house may infringe an IP.

(c) Owner of an IP/SoC design tool may be malicious. Owner's Untrusted CAD tool, while used for designing an IP in IP design house or for partially redesigning the IP in SoC company, may infringe the IP.

(d) In case of a fabless IP company selling hardware IP or a fabless SoC company buying firm/hard IP, the IP is fabricated in external fabrication facility, which may be untrusted and misappropriates the IP.

(e) An IP may be misused by an SoC designer during realization of an SoC or by an SoC user, while the IP remains functional on SoC.

(f) Other IP vendors, providing IPs for the same SoC may be malicious. While an IP is exchanging data with the other IPs on SoC, those may extract valuable information from that IP.

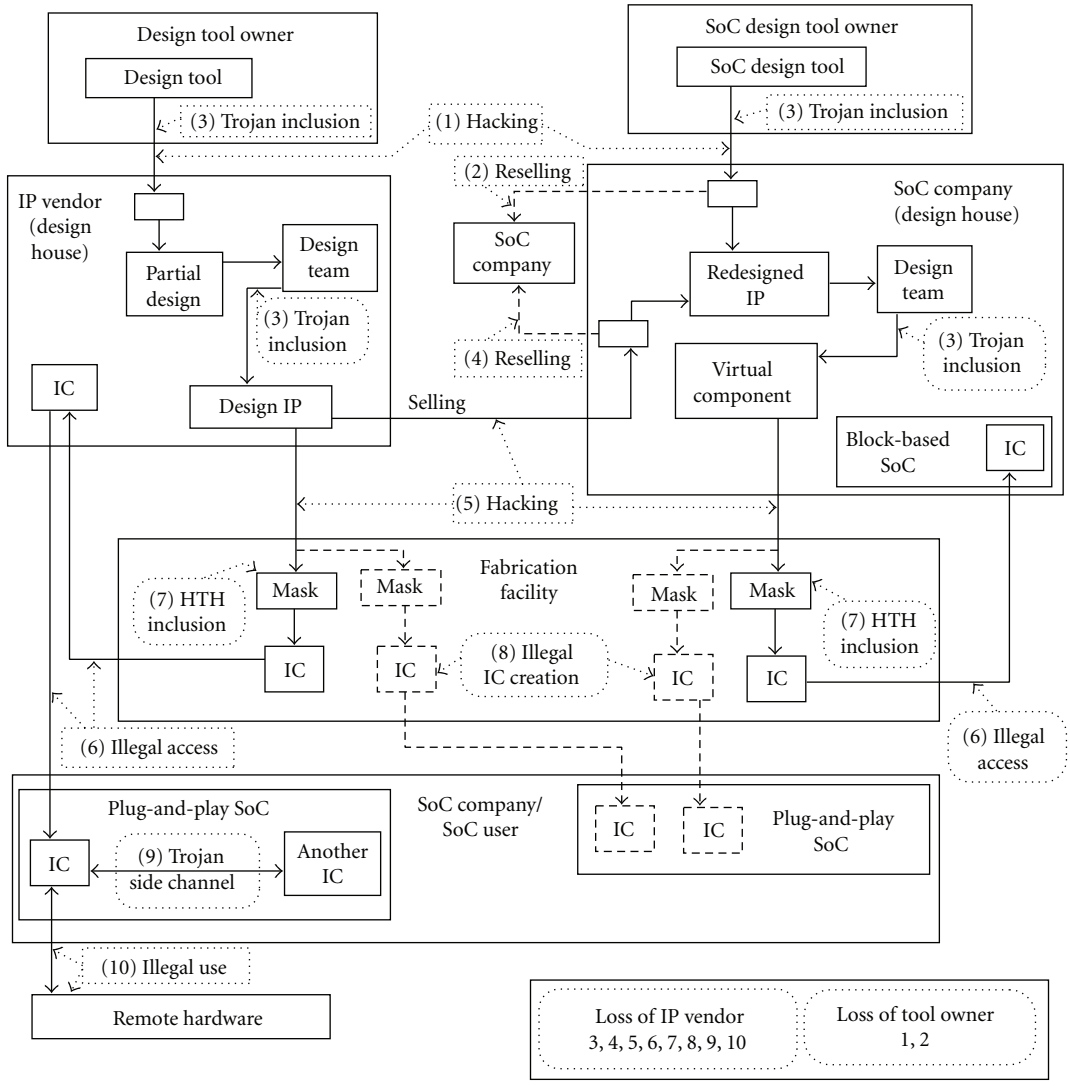


Figure 4: Locations and nature of IP infringements in IP-based SoC environment.

(g) Finally, an unknown adversary may misuse an IP, while on SoC, it is communicating with any remote hardware.

Misappropriation of IPs by any of these adversaries causes loss of revenue to the IP vendor. The threats for an IP can be divided into three categories, and these are discussed in the following subsections. The locations and nature of IP infringements in IP-based SoC environment are shown in Figure 4.

2.2.1. Unauthorized Access. An adversary may adopt the following ways to access an electronic or hardware IP or use it in unauthorized way [6].

(i) An SoC system design tool or an IP design tool may be hacked while transferred from tool owner to SoC/IP company. A firm IP may be hacked during its transfer from IP design house to SoC company. A hard IP can also be hacked during a similar transmission or in its way from fabless IP vendor/SoC company to the fabrication foundry.

(ii) An unauthorized person may intercept a hardware instance of an IP, that is, an ASIC, while it is distributed from the IP vendor to an SoC company or during its way from fabrication foundry to IP/SoC company.

(iii) In a security or networking application, an IP may communicate with a malicious remote hardware/application, or its communication may be intercepted by an adversary, thus an adversary may get benefited by the IP by misusing the IP.

2.2.2. Generation of Illegal Copies of IP. Illegal copies of an IP are generated due to intentional reselling of a firm/hard IP or fabrication of additional ICs in foundry [7, 8].

(i) An SoC design tool or a firm/hard IP is often resold by its legitimate buyer (an SoC design house) to another SoC design house in an illegal way.

(ii) From a mask of a hard IP or a virtual component, an untrusted fabrication facility may create illegal ICs or construct an additional mask, which is later on utilized to create any number of illegal ICs. Then the untrusted fabrication facility in unauthorized way sells these illegal ICs to SoC companies where those are reused on a plug-and-play SoC.

An hardware instance of an IP, that is, an IC, may be altered, replaced, or spoofed while is being used on a system. Therefore, each IC also needs separate authentication.

2.2.3. Information Retrieval through Trojan Horse. Trojan-based attack consists of inclusion of trojan horse (addition/modification of circuitry or design specification) into an IP by an adversary and later on, retrieval of circuit and design information through the already inserted trojan. This attack becomes relevant in SoC platform [9].

Trojan horse may be included in any of the following ways.

(i) Synthesis by an unreliable design team or usage of an untrusted design tool or an untrusted FPGA configuration tool may insert trojan horse into a soft/firm IP in IP design house or into a firm IP purchased in SoC design house.

(ii) In SoC company, unreliable firmware designer may maliciously modify interface definition of an IP, that is, its specification, to extract information from an IC after fabrication. Sometimes, untrusted SoC system design tool, used to redesign interface of an IP, facilitates unauthorized extraction of information from an IC fabricated from that IP.

(iii) Trojan Horses may be inserted into a mask by unreliable fabrication facility during fabrication so that the ASICs fabricated from that mask are trojan-infected.

Hardware trojan horse (HTH) acts as a side channel and leaks circuit and design information in the following scenarios.

(i) From an ASIC containing trojan horse, valuable information may be retrieved by the SoC designer during SoC realization or by an SoC user, while the IC is functional on SoC.

(ii) Information may also be extracted from an IP through any other untrusted hardware IP, while there is exchange of data between these two IPs.

(iii) While an hardware IP is communicating with a remote hardware (which may be malicious or their communication channel may be intercepted), trojan may leak information from the hardware IP.

In order to counterfeit these three major categories of IP misappropriation, several IP protection techniques have been adopted in various stages of IP-based SoC flow. Several critical attacks have also been designed to crack these security mechanisms or render these ineffective. The next section discusses on IPP techniques, attacks on those, and their countermeasures.

2.3. IP Protection in SoC

2.3.1. Locking-Based Security. This is a direct/active way to prevent unauthorized access of an electronic IP, to render illegally created/intercepted ICs useless and to protect communication of a hardware IP with remote devices. Locking-based security techniques include encryption, obfuscation, and remote activation.

(i) A hard IP, which is a design layout file in either GDSII (graphics data system II) or OASIS (open artwork system interchange standard) binary format, is locked by applying cryptographic encryption [10] on its binary content. It is encrypted while transferred from design house of IP/SoC company to fabrication foundry, where it is decrypted prior to fabrication. Symmetric (private) key cryptographic algorithms (e.g., DES, i.e., data encryption standard, AES, i.e., advanced encryption standard [11]) use same key for encryption and decryption, whereas, in public key cryptographic algorithms (e.g, RSA, ECC, i.e., elliptic key cryptography [11]) two separate keys are used for encryption and decryption. For FPGA design, corresponding bitfile core is kept encrypted [12] during its transmission to the SoC company, where it is decrypted using a decryption unit on FPGA hardware. Contrary to encryption which renders cipher text unreadable, another effective technique is obfuscation of an electronic IP, which renders the IP unusable to serve the purpose of security. The technique in [13] deterministically obfuscates a firm IP to a low-quality design using a secret key prior to transmission so that the high-quality IP can be regenerated only by the authorized person. It is directly applicable to firm IPs, thereby provides a faster way for its access control.

(ii) In the following techniques, IP is so designed that its each hardware instance needs a distinct secret key to be operational. If such an IC is illegally created by a malicious foundry or intercepted by an unauthorized person, its unauthorized user does not have the secret key, thus interception/illegal creation of ICs becomes useless.

An hardware instance of IP, that is, an IC, is locked by scrambling the control bus by controlled reversible bit permutations and substitutions [14]. Passive metering [15] registers each IC in a database uniquely based on its gate level characteristics. It can only detect illegal ICs by authenticating a chip against the database. In its active counterpart [16], design house keeps control of illegal ICs through monitoring of IC property and reuse, and by disabling functionalities of illegal ICs. The idea is further developed in remote activation technique [17], which replicates few states of underlying finite state machine (FSM) of an IP and then exploits inherent unclonable manufacturing variability to generate unique ID for each IC and adds a control, based on the unique ID, to the state transitions. Thus, it facilitates

piracy prevention and digital right management for each IC. An obfuscation technique discussed in [18], inserts a small FSM and constitutes a preinitialization state space, which is resilient against reverse engineering. It also uses a PUF-based IC-specific activation pattern to activate each IC with a distinct preinitialization state transition. Each preinitialization state is made observable as a particular output pattern for a predefined input sequence to authenticate each IC.

(iii) In the functional environment of SoC, in order to ensure security of hardware IPs communicating with an external device or remote hardware, their communications are encrypted using a cryptoprocessor embedded in hardware security module, thereby kept secret from the SoC user and other unknown adversaries. A cryptoprocessor is designed and implemented as a special-purpose embedded system optimized in terms of area, performance, and power for execution of a cryptographic algorithm in hardware. It is an SoC with several crypto-IP cores as coprocessors for high-speed execution of computation intensive operations, for example, SHA-1 hashing, wide operand modular arithmetic, and so forth, and for faster data compression required in the target encryption algorithm. Embedded cryptoprocessor has some area, performance, and power overhead on the chip quality. For example, an efficient FPGA implementation [19] of symmetric key encryption algorithm AES can achieve throughput of 24.922 Gb/s with the efficiency (throughput/area) of 6.97 Mb/s per slice of the FPGA used. An hardware implementation [20] of public key encryption technique ECC has an area requirement of 2.1 mm^2, delay 45 ns, and power 98.89 mW.

While a cryptoprocessor is tested on SoC, SoC designer may crack the encryption/decryption key exploiting information leaked from the cryptoprocessor. This class of attacks is known as side channel attacks, which include simple power analysis (SPA), dynamic power analysis (DPA), fault attack, cache timing attack, and attack using electromagnetic emanation. SPA and DPA are based on studying the device's power consumption while some key dependant sensitive variables are processed. Fault attack is based on the controlled induction of a fault and followed by analysis of a faulty cipher text produced from the cryptoprocessor. Cache timing attack uses the correlation of a sensitive variable with cache timing characteristics of an embedded implementation. Countermeasure to these attacks is to apply masking, permutation table, or random switching logic [21] to hide the nature of sensitive variables. A new class of side-channel attacks has been designed based on correlation power analysis (CPA) and mutual information analysis (MIA) [22]. This section of attacks awaits innovative solutions.

2.3.2. Authentication-Based Security. Security threat exists even after applying locking-based security due to the following reasons.

(a) An SoC design house may illegally resell the purchased firm/hard IP to another SoC company or resell the hardware IP and intentionally share its secret key (to make it operational) with the illegal buyer.

(b) The decryption key of the decryption unit on FPGA may be cracked applying side channel attacks, thus an attacker manages to get the unencrypted bitfile.

Watermarking and fingerprinting provide passive protection and authenticate an IP and an IC to establish digital rights of legal IP vendor and legitimate buyer of an IC, respectively, even if locking-based security is cracked. The process of embedding the signature of IP vendor in form of watermark is known as watermarking, whereas including that of IP buyer in form of fingerprint is termed as fingerprinting. Fingerprints may be constructed for a set of ICs fabricated from an IP by characterizing their manufacturing variability. If any misappropriation of an IP or IC is suspected, these signatures are verified from the IP or IC to identify the genuine IP vendor or legitimate IP buyer.

The following are the desiderata of a signature-embedding technique.

(a) A signature embedding technique should incur low overhead on the chip quality in terms of area, delay, and power.

(b) The signature embedding and signature verification techniques should be fast.

(c) The technique should be robust against typical attacks like tampering, finding ghost signatures, additive attack [8], as well as resilient in the design flow.

Signature embedded by applying constraints in place/route phase of physical design [8] or through incremental router [37] cannot be verified from an IC fabricated from that marked design. Hence, in order to provide effective security of an IP on SoC, the mark (i.e., watermarks and fingerprints) insertion technique should be resilient against fabrication. Furthermore, it is desirable that the embedded signatures should be resistant against process variation.

Marks embedded through any of the following techniques are resilient against fabrication.

(i) Signature of IP vendor, that is, watermark, may be hosted into nonused cells of memory structure described in HDL [24].

(ii) Underlying finite state machine is modified so that desired watermark can be detected at the chip's outputs for a particular key input sequence [25]. Such a way of generating watermark only for particular key inputs is known as dynamic watermarking.

(iii) Dynamic watermarking has been applied to reconfigurable scan architecture during physical synthesis so that desired watermark can be verified as scan outputs [26]. For both dynamic watermarking techniques, watermarks are easily verifiable.

(iv) Physically unclonable functions (PUFs) [27] authenticate each IC instance by leveraging manufacturing variability of each fabricated IC, based on a nonfunctional characteristics such as delay or power. So, the techniques based on PUFs can perform fingerprinting of ICs.

(v) A set of fingerprints is constructed for an IC family utilizing side channel informations such as power,

temperature, and electromagnetic profile. The technique is also capable of detecting trojan of 3-4 orders of magnitude smaller than the main circuit using signal processing [28]. The above two techniques in (iv), and (v) can be categorized as postfabrication IC fingerprinting techniques. In (iv), PUF structure is embedded in the IP design by the IP design house as a prerequisite.

(vi) Signature of IP vendor, that is, watermark, may be embedded into logic synthesis phase through incremental technology mapping of selective disjoint closed cones [29].

(vii) Signatures of both IP vendor and IP buyer are stored as configuration bitstream of unused configurable logic blocks (CLBs) of FPGA [30].

The techniques in (i) and (ii) are associated with behavioral phase and, therefore, are applicable to both ASIC and FPGA. The techniques in (iii), (iv) and (v) are for ASIC authentication and those in (vi), and (vii) are for FPGA bitfile core authentication.

While a hardware IP is operational on SoC and needs to communicate with an external device or remote hardware, hardware security module on the SoC authenticates the device prior to establishing communication. In the technique discussed in [23], a secure hardware/cryptoprocessor in an SoC authenticates the remote processor by challenging it to compute a check sum that depends on cycle-by-cycle activities of its internal microarchitectural mechanisms for a given code within a time limit. Thus, it controls unauthorized access of data from an IP operating on SoC.

2.3.3. Security against Trojan Horse. An IP, to be reused on SoC, is first handled by the SoC system designer, who is well equipped with circuit and system knowledge, and when it is operational on SoC, it exchanges data with other hardware IPs on SoC and other external devices or remote hardware. So, if the IP is already trojan-infected, trojan side channel attack is quite prominent. So, detection of existence of trojan becomes essential prior to dispatching the chip to SoC company. Trojan may be inserted in design level as well as during its fabrication. So, we need *efficient* trojan detection techniques effective for both design and hardware IP.

Information hiding strategy to design trusted system [31] is capable of detecting possible existence of trojan horse in design IP. In this work, an IP company (design team 1) creates a high-quality partial solution for a given problem specification and extracts a modified specification from that partial solution. The modified specification is then sent to another design team (team 2 which may be untrusted). From the complete design generated by the team 2, a partial solution is extracted to cross-check it with the high-quality partial solution created by team 1 to detect possible inclusion of trojan by team 2.

The technique in [32] precisely measures actual combinational delay of large number of paths. Thus, it can detect hardware trojan horse (HTH) due to increase in delay at certain paths. This method characterizes each fabricated IC instance based on manufacturing variability, therefore, it is effective for IC fingerprinting. However, it is exhaustive and not time efficient.

Among the trojan detection techniques based on gate-level characterization (GLC), [33] characterizes gates using physical properties specifically leakage current. Measurements on leakage current are processed with linear programming (LP). It imposes additional constraints on LP formulation to indicate nature and location of additional ghost circuitry. However, this technique cannot characterize all the gates due to collinearity and cannot detect collinear HTH. The technique in [34] breaks the correlations by applying thermal control on the process of GLC. Thermal conditioning imposes extra variations on gate level leakage power by characterizing switching power followed by heat dissipation due to it. Both of them are effective for process invariant trojan detection.

The authors of [35] proposed an IPP technique for detection of trojan horse inserted in FPGA design files, from bitfile core, or from FPGA hardware loaded with bitfile core. It is a parity-based method and uses two-level randomized ECC structures. Failing to detect desired parity relation signals possible existence of additional circuitry, that is, trojan in the FPGA design.

The technique in [36] resists an untrusted synthesis CAD tool to add/modify design specification. It employs the CAD tool under inspection to difficult scheduling and assignment synthesis tasks for a completely specified pertinent design so that there is no room for the tool to add malicious circuitry. The technique uses a trusted tool to fully account all resources at each step.

Effectiveness of several IPP techniques for various security aspects are summarized in Table 1. "Y" in a cell indicates the technique in the corresponding row is effective to achieve the security aspect specified in the corresponding column.

3. Challenges

(i) In the SoC environment, SoC designer has enough opportunity to misappropriate an IC during realization of the SoC. An IC exchanges data with other hardware IPs on SoC and external devices or remote hardware. SoC users, other IPs on SoC and external or remote device may be malicious, so an IC faces multiple security threats in SoC environment. Remote activation authenticates an IC as a legal instance of hardware IP, but if the IC belongs to some untrusted source, its remote access may pose threats to other trusted IP components on the same SoC.

(ii) In SoC environment, each instance of IP, that is, IC, needs to be protected. All the existing techniques to control access of ICs use PUF-based IC fingerprinting. However, signature of an IC is limited in length, and an attacker may guess a signature by developing a timing model for PUF structure.

(iii) Emphasis has been given to IC fingerprinting and design IP watermarking. However, fingerprinting of design IPs is quite relevant as transaction of IP often takes place in form of firm/hard IP between IP vendor and SoC company. A fingerprinting technique

Table 1: Effectiveness of IPP techniques for the following security aspects.

IPP techniques	Access control			Authentication		Trojan detection	
	IP	IC	Data from IC	IP	IC	From design	From hardware
Charbon and Torunoglu 2000 [10]	Y						
Adi et al. 2006 [12]	Y						
Saha and Sur-Kolay 2009 [13]	Y						
Roy et al. 2008 [14]		Y					
Alkabani et al. 2008 [15]		Y					
Alkabani and Koushanfar 2007 [16]		Y					
Alkabani et al. 2007 [17]		Y			Y		
Chakraborty and Bhunia 2009 [18]		Y			Y		
Granado-Criado et al. 2010 [19]			Y				
Dyka and Langendoerfer 2005 [20]			Y				
Suzuki et al. 2004 [21]			Y				
Deng et al. 2009 [23]			Y	Y			
Castillo et al. 2007 [24]				Y			
Abdel-Hamid et al. 2005 [25]				Y			
Saha and Sur-Kolay 2010 [26]				Y			
Majzoobi and Koushanfar 2009 [27]					Y		
Agrawal et al. 2007 [28]					Y		Y
Cui et al. 2008 [29]				Y			
Lach et al. 2001 [30]				Y	Y		
Gu et al. 2009 [31]						Y	
Li and Lach 2008 [32]					Y		Y
Potkonjak et al. 2009 [33]							Y
Wei et al. 2010 [34]							Y
Dutt and Li 2009 [35]						Y	Y
Potkonjak 2010 [36]						Y	

needs wider space for mark insertion compared to watermarking a design IP.

(iv) CAD tools are too complex to be traced. Therefore, synthesis tools and libraries, testing and verification tools and configuration scripts act as significant sources of hardware trojan. Designing trustable ICs and systems using untrusted CAD tools and untrusted reusable IP cores has been emerged as a true challenge in IP-based SoC security.

(v) With very few techniques, for example, gate-level characterization, process invariant trojan detection is possible. However, these techniques are exhaustive and consequently inefficient. There is a tradeoff between efficiency and strength of a protection technique for certain security aspects.

(vi) Trojan can be inserted in design level as well as during its fabrication. It is difficult to design a technique for both pre and postfabrication trojan detection. Furthermore, the postfab HTH detection should be process invariant.

4. New Opportunities

(i) There is no sole technique to ensure protection in all the three major security aspects shown in Table 1. Furthermore, any technique, which serves two security purposes, is not equally efficient and robust in both the tasks.

(ii) We are in need of efficient techniques for certain detection of presence of trojan horse both at design level as well as at hardware level.

(iii) The existing countermeasures to side channel attacks cannot prevent the new class of side channel attacks based on CPA and MIA. SoC-based industry, therefore, awaits for innovative solutions to resist these attacks.

5. Conclusion

Rapid growth of technology in semiconductor industry continually creates new security holes specially in SoC platform. As security threats have direct impact on Psilicon-based

economy, these threats demand immediate solutions to check misuse of technology and consequently loss of revenue for the IP companies. In recent trends, design-for-security for IP-based SoC design methodology forms an open research area, where constant effort and domain expertise are needed to check misappropriation of IPs.

References

[1] R. Rajsuman, *System-On-a-Chip*, Artech House, London, UK, 2009.

[2] W. Wolf, *Modern VLSI Design: IP-Based Design*, Prentice Hall, New York, NY, USA, 2008.

[3] S. Raif and K. Arno, *Reuse Techniques for VLSI Design*, Kluwer Academic Press, Boston, Mass, USA, 1999.

[4] R. Seepold and N. M. Madrid, *Virtual Components Design and Reuse*, Kluwer Academic Press, Boston, Mass, USA, 2000.

[5] D. Mathaikutty and S. Shukla, *Metamodelling-Driven IP Reuse for SoC Integration and Microprocessor Design*, Artech House, London, UK, 2009.

[6] G. Qu and M. Potkonjak, *Intellectual Property Protection in VLSI Designs: Theory and Practice*, Kluwer Academic Press, Boston, Mass, USA, 2003.

[7] D. Saha, P. Dasgupta, S. Sur-Kolay, and S. Sen-Sarma, "A novel scheme for encoding and watermark embedding in VLSI physical design for IP protection," in *Proceedings of the International Conference on Computing: Theory and Applications (ICCTA '07)*, pp. 111–116, March 2007.

[8] A. B. Kahng, J. Lach, W. H. Mangione-Smith et al., "Constraint-based watermarking techniques for design IP protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 10, pp. 1236–1252, 2001.

[9] A.-R. Sadeghi and D. Naccache, *Towards Hardware-Intrinsic Security: Foundations and Practice*, Springer, New York, NY, USA, 2010.

[10] E. Charbon and I. H. Torunoglu, "On intellectual property protection invited paper," in *Proceedings of the 22nd Annual Custom Integrated Circuits Conference (CICC '00)*, pp. 517–522, May 2000.

[11] A. Menezes, P. V. Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, New York, NY, USA, 1996.

[12] W. Adi, R. Ernst, B. Soudan, and A. Hanoun, "VLSI design exchange with intellectual property protection in FPGA environment using both secret and public-key cryptography," in *Proceedings of the IEEE Annual Symphosium on VLSI (ISVLSI '06)*, pp. 24–32, 2006.

[13] D. Saha and S. Sur-Kolay, "Encoding of floorplans through deterministic perturbation," in *Proceedings of the 22nd International Conference on VLSI Design*, pp. 315–320, January 2009.

[14] J. A. Roy, F. Koushanfar, and I. L. Markov, "Protecting bus-based hardware IP by secret sharing," in *Proceedings of the 45th Design Automation Conference (DAC '08)*, pp. 846–851, June 2008.

[15] Y. Alkabani, F. Koushanfar, N. Kiyavash, and M. Potkonjak, "Trusted integrated circuits: a nondestructive hidden characteristics extraction approach," *Proceedings of the Information Hiding*, pp. 112–117, 2008.

[16] Y. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security," in *Proceedings of the USENIX Security Symphosium*, pp. 291–306, 2007.

[17] Y. Alkabani, F. Koushanfar, and M. Potkonjak, "Remote activation of ICs for privacy prevention and digital right management," in *Proceedings of the International Conference on CAD*, pp. 674–677, 2007.

[18] R. S. Chakraborty and S. Bhunia, "HARPOON: an obfuscation-based SoC design methodology for hardware protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1493–1502, 2009.

[19] J. M. Granado-Criado, M. A. Vega-Rodríguez, J. M. Sánchez-Pérez, and J. A. Gómez-Pulido, "A new methodology to implement the AES algorithm using partial and dynamic reconfiguration," *Integration, the VLSI Journal*, vol. 43, no. 1, pp. 72–80, 2010.

[20] Z. Dyka and P. Langendoerfer, "Area efficient hardware implementation of elliptic curve cryptography by iteratively applying karatsubas method," in *Proceedings of the Design Automation and Test in Europe*, vol. 3, pp. 70–75, 2005.

[21] D. Suzuki, M. Saeki, and T. Ichikawa, "Random Switching Logic: A Counter-Measure against DPA based on Transition Probability," ePrint Archive, 2004.

[22] E. Prouff and R. McEvoy, "First-order side-channel attack on the permutation table countermeasures," in *Proceedings of the Cryptographic Hardware and Embedded Systems*, pp. 81–96, 2009.

[23] D. Deng, A. H. Chan, and G. Edward Suh, "Hardware authentication leveraging performance limits in detailed simulations and emulations," in *Proceedings of the 46th Design Automation Conference (DAC '09)*, pp. 682–687, 2009.

[24] E. Castillo, U. Meyer-Baese, A. García, L. Parrilla, and A. Lloris, "IPP@HDL: efficient intellectual property protection scheme for IP cores," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 5, pp. 578–591, 2007.

[25] A. T. Abdel-Hamid, S. Tahar, and EL. M. Aboulhamid, "A public-key watermarking technique for IP designs," in *Proceedings of the Design, Automation and Test in Europe (DATE '05)*, pp. 330–335, March 2005.

[26] D. Saha and S. Sur-Kolay, "A unified approach for IP protection across design phases in a packaged chip," in *Proceedings of the 23rd International Conference on VLSI Design*, pp. 105–110, January 2010.

[27] M. Majzoobi and F. Koushanfar, "Techniques for design and Implementation of secure reconfigurable PUFs," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 2, no. 1, article 5, 2009.

[28] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *Proceedings of the IEEE Symposium on Security and Privacy (SP '07)*, pp. 296–310, May 2007.

[29] A. Cui, C. H. Chang, and S. Tahar, "IP watermarking using incremental technology mapping at logic synthesis level," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 9, pp. 1565–1570, 2008.

[30] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Fingerprinting techniques for field-programmable gate array intellectual property protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 10, pp. 1253–1261, 2001.

[31] J. Gu, G. Qu, and Q. Zhou, "Information hiding for trusted system design," in *Proceedings of the 46th Design Automation Conference (DAC '09)*, pp. 698–701, July 2009.

[32] J. Li and J. Lach, "At-speed delay characterization for IC authentication and Trojan horse detection," in *Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust (HOST '08)*, pp. 8–14, June 2008.

[33] M. Potkonjak, A. Nahapetian, M. Nelson, and T. Massey, "Hardware trojan horse detection using gate-level characterization," in *Proceedings of the 46th Design Automation Conference (DAC '09)*, pp. 688–693, July 2009.

[34] S. Wei, S. Meguerdichian, and M. Potkonjak, "Gate-level characterization: foundations and hardware security applications," in *Proceedings of the Design Automation Conference (DAC '10)*, pp. 222–227, 2010.

[35] S. Dutt and L. Li, "Trust-based design and check of FPGA circuits using two-level randomize ECC structure," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 2, no. 1, 2009.

[36] M. Potkonjak, "Synthesis of trustable ICs using untrusted CAD tools," in *Proceedings of the Design Automation Conference*, pp. 633–634, 2010.

[37] T. Nie and M. Toyonaga, "An efficient and reliable watermarking system for IP Protection," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E90-A, no. 9, pp. 1932–1939, 2007.

Hindawi Publishing Corporation
VLSI Design
Volume 2011, Article ID 471903, 12 pages
doi:10.1155/2011/471903

Review Article

The Impact of Statistical Leakage Models on Design Yield Estimation

Rouwaida Kanj,[1] Rajiv Joshi,[2] and Sani Nassif[1]

[1] *IBM Austin Research Labs, Austin, TX 78758, USA*
[2] *IBM TJ Watson Labs, Yorktown Heights, NY 10598, USA*

Correspondence should be addressed to Rouwaida Kanj, rouwaida@us.ibm.com

Received 19 October 2010; Accepted 11 December 2010

Academic Editor: Shiyan Hu

Device mismatch and process variation models play a key role in determining the functionality and yield of sub-100 nm design. Average characteristics are often of interest, such as the average leakage current or the average read delay. However, detecting rare functional fails is critical for memory design and designers often seek techniques that enable accurately modeling such events. Extremely leaky devices can inflict functionality fails. The plurality of leaky devices on a bitline increase the dimensionality of the yield estimation problem. Simplified models are possible by adopting approximations to the underlying sum of lognormals. The implications of such approximations on tail probabilities may in turn bias the yield estimate. We review different closed form approximations and compare against the CDF matching method, which is shown to be most effective method for accurate statistical leakage modeling.

1. Introduction

With technology scaling, memory designs are the first to suffer from process variation. Density requirements in memory cells, latches, and register files aggravate variability effects and often undergo performance and yield degradation. More recently, the trend is starting to become more visible in peripheral logic [1], and random variations can lead to reduced noise margins. From a leakage perspective, general interest has been to model the statistical leakage distribution of the full-chip [2]. However, under extreme device conditions, leakages may inflict delays or faulty behavior: example false reads. In memory designs, bitline leakages can further aggravate the variability effects and impact readability of the design. Often many devices are stacked on the bitlines, and it has become necessary to statistically account for and model those many independent and, in most cases, identical, leaky sources. A typical array column can have around 16 to 64 cells per bitline. With the aim of capturing rare fail events accurately and modeling the variability space [3], it becomes important to emulate those independent variability sources by a single *statistically* equivalent device, the main goal here being to reduce the dimensionality space. This is especially important for response surface modeling-based integration methods or variance reduction techniques.

The threshold voltage variation due to random dopant fluctuation [4] for the single device, however, is different from that of the equivalent device. In a typical linear model, the equivalent statistical model would be derived from linear relationships with respect to the individual devices, and what the designers often refer to as the square root law; as the device area (A) increases, the threshold voltage variation scales $\sim 1/\sqrt{A}$. This would let us model "n" small devices with one large device whose variation is $\sim 1/\sqrt{n}$ times the original variation. However, as was indicated in [5, 6] this model lacks accuracy when modeling the highly nonlinear leakage currents where the dependency is exponential on the threshold voltage (the source of variation). In fact, as we will explain in the following sections, the leakage current is distributed lognormally, and the problem of the equivalent device is known mathematically as the sum of lognormals. Mathematically, a true and exact closed form does not exist for the method, and historically there have been many approximations. There is a lot of literature on the sum of lognormals, and applications span a wide range of domains including economics, finance, actuarial sciences,

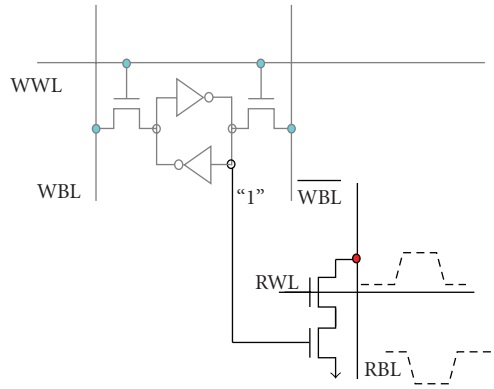


Figure 1: A schematic of an 8T cell/register file. During Read, the read wordline "RWL" turns ON. If the cell storage node is held high, "1", the read bitline (RBL) is discharged.

and others, with engineering being one of the oldest. The most famous approximations to the sum of lognormals are Fenton-Wilkinson's [7] and Schwartz-Yeh (SY) [8]. For circuit design, those models have been studied in [5, 6] to enable modeling the threshold voltage distribution of the equivalent device. The authors in [5] relied on the Fenton-Wilkinson's (FW) method to maintain a reasonable 3-sigma estimate (99th percentile). It is often indicated that when σ dB > 4, FW method may not result in proper approximation of the lognormal distribution. On the other hand, the authors in [6] proposed to use the Schwartz-Yeh (SY) methodology to model the equivalent threshold voltage distribution of a 3-fin device. Again, the approximation is qualified, in this case for a small number of summands, based on a thousand samples thereby ignoring the accuracy of the model in the tail regions which is critical to estimation of low fail probabilities. In the following sections, we review the modeling trends in the critical regions, focusing on the cumulative density function (CDF) matching method [9] as a viable solution for reliable statistical modeling when the tail region modeling is key to the accuracy of the yield estimate.

The paper is organized as follows. Section 2 introduces a basic circuit example to describe the need for the single equivalent device modeling. Section 3 reviews the mathematical background and assumptions of the most common sum of lognormals approximation methods. It also provides a summary chart of the different approaches under study. Section 4 evaluates the impact on fail probability estimations. Finally conclusions are presented in Section 5.

2. Equivalent Device Modeling

Figure 1 illustrates an 8T cell/register file. The circuit operates like a 6T SRAM cell during cell Write. During Read, the read wordline turns ON. If the cell is storing a "1" on the right-side-node, in this example, the Read stack will turn ON, thereby discharging the read bitline (RBL).

Driven by density requirements, many cells share the same bitline and sense amplifier (or read logic). Only one cell per column is accessed at a given time; other cells remain unaccessed. For the example of Figure 1, the RWL of the unaccessed cells is turned OFF. However, based on the data storage, it is possible that the data stack device turns on, thereby rendering the stack leaky. Figure 2 illustrates a scenario where the accessed cell is OFF. RBL is expected to remain high. However, leakages through the accessed stack and unaccessed stacks may falsely discharge the RBL node. This together with noise, mismatch in the read logic, and other environmental effects can degrade the yield of the system and increase the probability of a false read.

Note that this phenomenon is also manifested in other memory and domino logic designs. For example in the case of 6T SRAM, the devices sharing the WBL (see Figure 1) can be storing 1's thereby leaking onto the WBL and degrading the Read "0" performance in a 6T fashion. Hence there is a need to account for the statistical effects of the leaky devices in any design yield optimization and study. This can dramatically increase the dimensionality of the problem thereby reducing the efficiency of statistical integration, response surface methods, or any methods that try to model the failure region and that suffer from sparsity and the curse of dimensionality. Consider for example a study that involves variability in 64 devices on the bitline along with variability sense amp devices. Thus, a model for a statistically equivalent device to emulate the multiple leaky devices can significantly simplify the complexity. Figure 3 summarizes the problem.

Given: n devices $T_1, \ldots, T_n$.

The threshold voltage variation δV_T is normally distributed with zero mean and standard deviation $\sigma_{VTi} = \sigma_0$.

That is, $\delta V_{Ti} \sim N(0, \sigma_0)$

Find: δV_{Teqv} distribution for the device T_{eqv} such that "$n * I_{Teqv}$" is distributed according to "$\sum I_{Ti}$".

Note that we assume here independent and identical devices; the problem can be generalized to nonidentical/correlated devices.

Leakage Current Model. Leakage current as a function of the threshold voltage variation can be modeled according to (1). Hence it portrays an exponential dependency on the random variable δV_T

$$I_{leak} = e^{a\delta V_T + b}, \qquad a \propto \frac{KT}{q}, \tag{1}$$

q is the electron charge, K is boltzmann constant, and T is the temperature. For simplicity, T_{eqv} has the same device width as the other T_i, but it is followed by a current multiplier to model the equivalent leaky device and solve for the new δV_{Teqv} distribution: the set of (2) represents how the problem can be reduced. Parameters Z_i are normal variables, e^{Zi} are lognormal variables (see next section for definitions), and e^{Zeqv} (and Y) are distributed according to a sum of lognormal. A sum of lognormals is not distributed lognormally, and hence Z_{eqv} is not normal. In the following

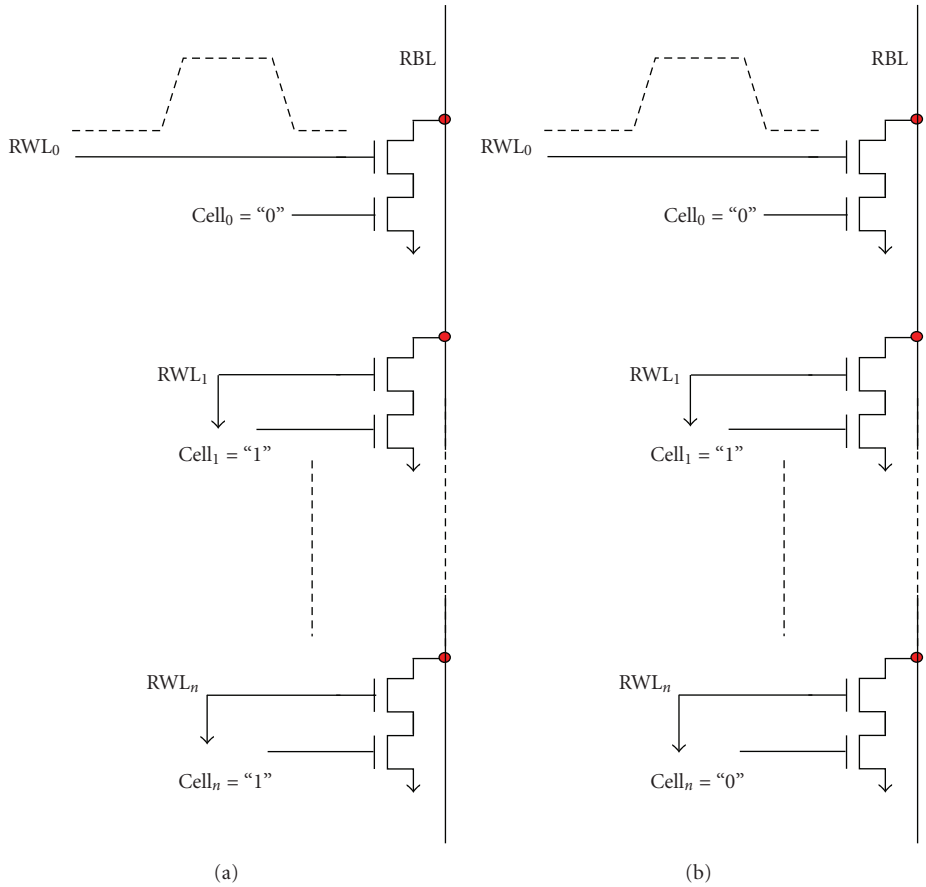


Figure 2: The accesses cell is storing a "0". The read bitline (RBL) should stay high. Leakage from unaccessed cells can falsely discharge the bitline. Leakage can happen due to (a) all the cells on the bitline leaking, or (b) an intermediate number of cells leaking. For our work, we ignore the stack leakages if both read transistors (whose gates are the cell and RWL) are OFF.

section we review the sum of lognormals characteristics and possible methods of approximation that can be used to generate the statistical characteristics of Z_{eqv} and hence δV_{Teqv}

$$
\begin{aligned}
I_{\text{sum}} &= I_{\text{leak1}} + \cdots + I_{\text{leak}n} = n * I_{\text{leak-eqv}},\\
I_{\text{sum}} &= \sum_{i=1}^{n} e^{a\delta V\text{Ti}+b} = n * e^{a\delta V\text{Teqv}+b},\\
Z_{\text{eqv}} &= a\delta V_{\text{Teqv}}; \qquad Z_i = a\delta V_{\text{Ti}},\\
e^{Z\text{eqv}} &= \frac{1}{n}\sum_{i=1}^{n} e^{Z\text{i}} = \frac{1}{n}Y.
\end{aligned}
\tag{2}
$$

3. Sum of Lognormals

In this section we will focus on methods to approximate models for Y (2), and compare their accuracy to the CDF matching approach. First we present some background information.

Lognormal Distribution. Let $Z = \ln(X)$. If Z is normally distributed $\sim N(\mu_z, \sigma_z)$, then X is lognormally distributed and its probability density function is [10]

$$
f_X(x) = \begin{cases} \dfrac{1}{\sqrt{2\pi}\sigma_z} e^{-(z-\mu_z)^2/2\sigma_z{}^2} & x > 0 \\ 0 & x \le 0. \end{cases}
\tag{3}
$$

Often the Gaussian Variable $V = 10\log 10(X)$ is defined. V is in decibels (dB), and $V \cong 4.34\,Z$; if $Z \sim N(0,1)$, $V \sim N(0, 4.34\,\text{dB})$.

Sum of Lognormals. A common way to compute the probability density function (pdf) of a sum of independent random variables, would be from the product of the characteristic functions, CF, of the summands [11]; a characteristic function of random variable Z is the expected value $(E[e^{jwZ}])$. However, for the case of the sum of lognormals, $Y = \sum e^{Z\text{i}}$, a closed form of the CF does not exist, and approximations are used instead to represent the density function of the sum of lognormals. Some approximations

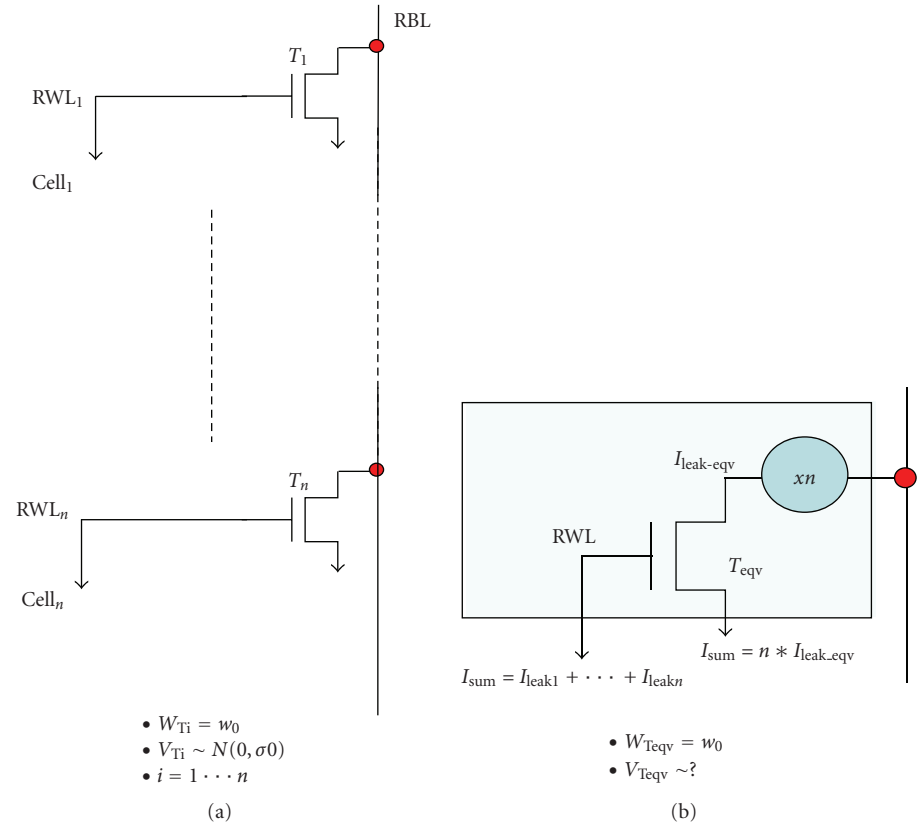


Figure 3: The problem of multiple devices leakages can be statistically equivalent to that of a single device. The distribution of the threshold voltage of the equivalent device is to be solved for.

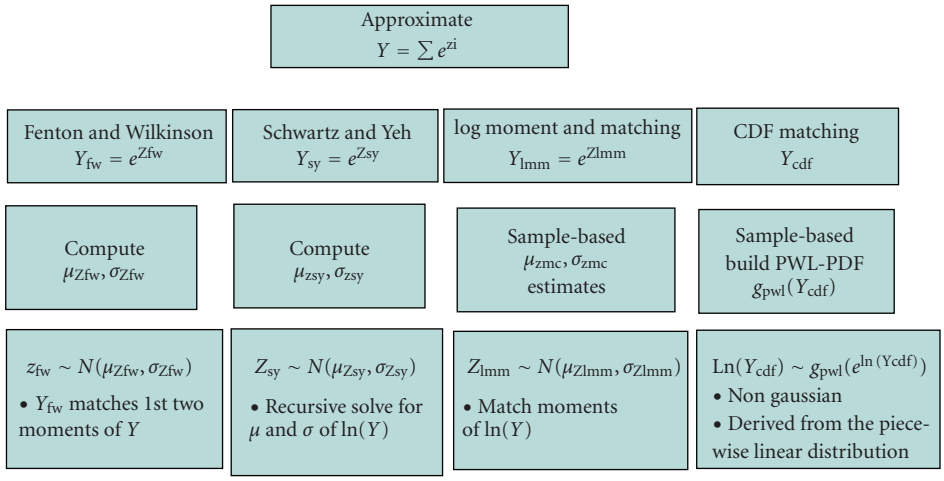


Figure 4: Different methods to be compared against the CDF matching approach involve moment matching to $\ln(Y)$, Schwartz-Yeh, and Fenton-Wilkinson methods.

include modeling the sum of lognormals as an "approximately" lognormal variable (see (4)); where Z_m. is normally distributed (m refers to the method of approximation)

$$Y = \sum_{i=1}^{n} e^{Zi} \approx e^{Z_m}. \tag{4}$$

Common techniques are SY and FW methods mentioned earlier. In the following section, we will revisit those methods and compare them to CDF matching. Figure 4 summarizes the four techniques that will be visited:

(1) Fenton-Wilkinson method,

(2) Schwartz-Yeh method,

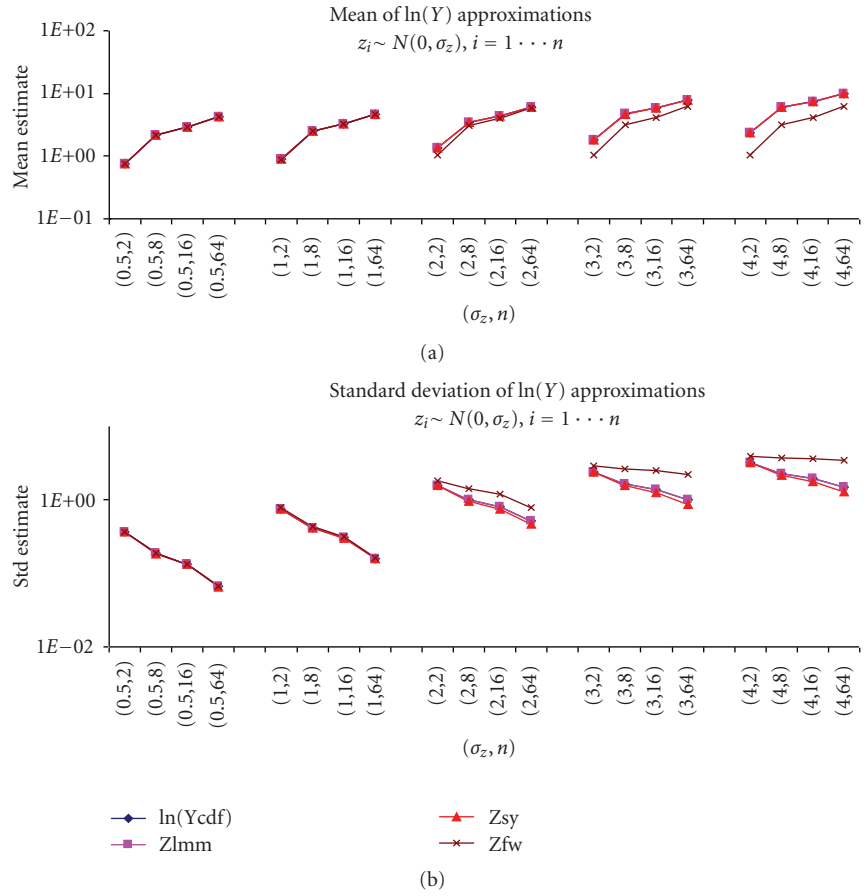


FIGURE 5: Plot of the mean and standard deviation of $\ln(Y)$ obtained from the different approximations. FW method targeted to match moments of Y falls behind for larger σ_z (>1 or 4 dB).

(3) Log Moment Matching approximation,

(4) CDF matching.

All these methods except the CDF matching one assume that Y is a lognormal variable; that is, that Z_m (and hence δVt) is a normally distributed Gaussian variable whose mean and standard deviation are to be found.

3.1. Fenton-Wilkinson. Fenton and Wilkinson method estimates the standard deviation and mean of the Gaussian variable Z_{fw} by matching the first two moments of Y, $E[Y]$ and $E[Y^2]$, to those of the sum of lognormals (the two sides of (4)) according to (5)

$$\begin{aligned} Y &= e^{\text{Zfw}}, \\ E\left[e^{Z_{\text{fw}}}\right] &= E\left[e^{Z_1} + \cdots + e^{Z_n}\right], \\ E\left[e^{2Z_{\text{fw}}}\right] &= E\left[\left(e^{Z_1} + \cdots + e^{Z_n}\right)^2\right]. \end{aligned} \tag{5}$$

It is a well-known fact [10] that given a Gaussian variable, x, with mean and standard deviation μ_x and σ_x then (6) holds

$$E[e^{nx}] = e^{n\mu_x + (1/2)n^2\sigma_x^2} \tag{6}$$

by relying on (5) and (6) we can obtain a closed form for μ_{Zfw} and σ_{Zfw} according to (7) for the case of independent variables $z_i \sim N(\mu_i, \sigma_z)$

$$\begin{aligned} \sigma_{\text{Zfw}} &= \sqrt{\ln\left[(e^{\sigma z^2} - 1)\frac{\sum e^{2\mu_i}}{\left(\sum e^{\mu_i}\right)^2} + 1\right]}, \\ \mu_{\text{Zfw}} &= \ln\left(\sum e^{\mu_i}\right) + \frac{\sigma z^2}{2} - \frac{\sigma_{\text{Zfw}}^2}{2}. \end{aligned} \tag{7}$$

3.2. Schwartz and Yeh. Again the methodology starts with the assumption that the sum of lognormals is approximately lognormally distributed

$$Y = e^{\text{Zsy}}, \tag{8}$$

where Z_{sy} is a Gaussian variable $\sim N(\mu_{\text{Zsy}}, \sigma_{\text{Zsy}})$. It approximates the mean and standard deviation of the $Z_{\text{SY}} = \ln(Y)$, with numerical recursion; whereas FW estimates the first two moments of Y. Schwartz and Yeh method relies on the ability to exactly compute the mean, μ_{Zsy}, and standard deviation, σ_{Zsy}, of $Z_{\text{SY}} = \ln(Y)$ for the case when the number of lognormal variables in the sum $n = 2$. For $n > 2$, the method

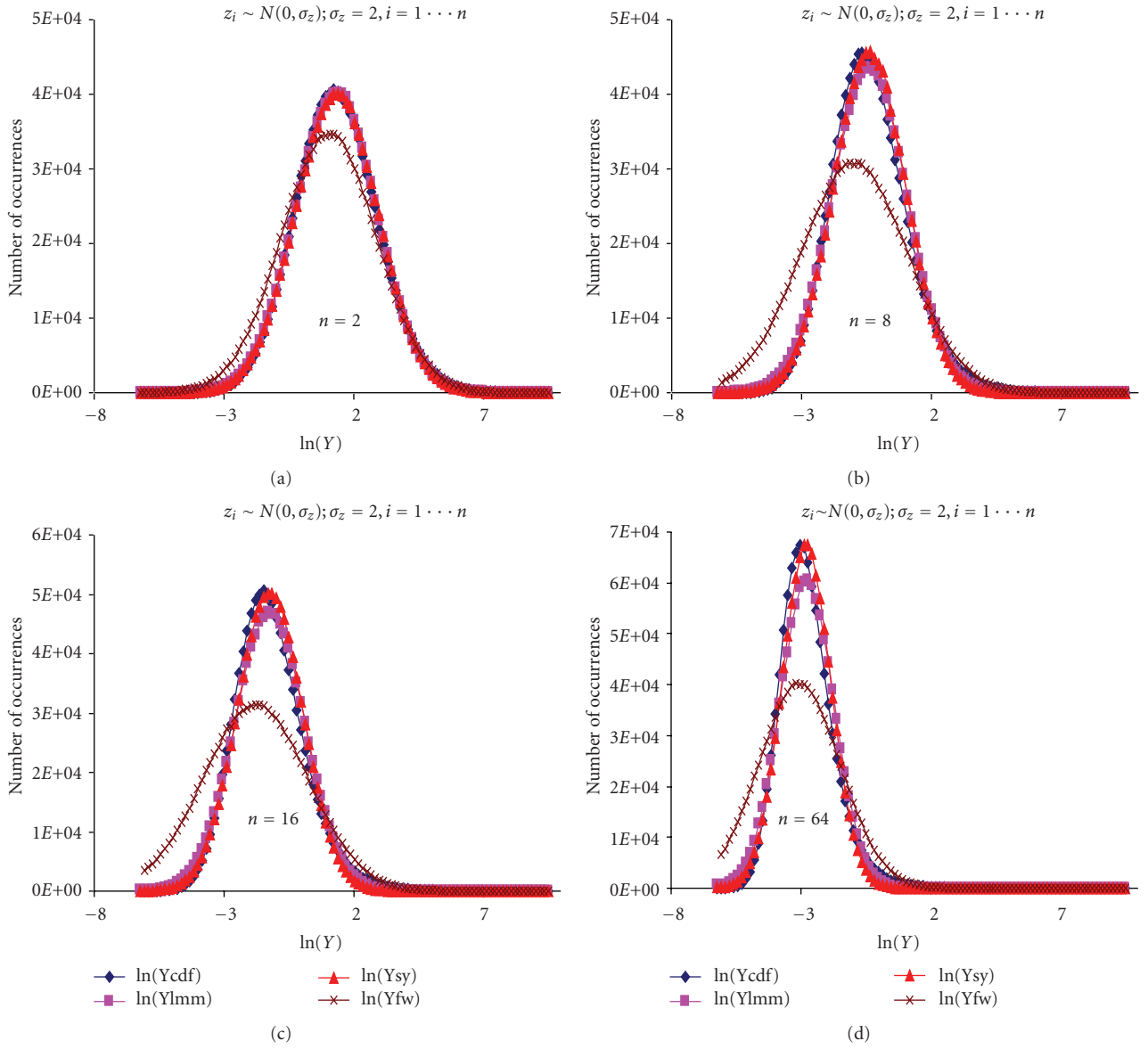


FIGURE 6: Histogram (pdf plot) of $\ln(Y)$ shows that the FW method does not model the body of the (z or δVt) distribution well compared to other methods.

then relies on a recursive approach, adding one factor at a time. Hence, the mean and standard deviation of

$$Z_{\text{sy}} = \ln\left(e^{Z_1} + \cdots + e^{Z_n}\right) \tag{9}$$

can be derived from the following set of generalized equations (10); $Z_{\text{sy}(k-1)}$ is assumed to be normally distributed; $k = 2, \ldots, n$, and the Z_i are assumed to be uncorrelated in (10)

$$\begin{aligned} Z_{\text{sy}(k)} &= \ln\left(e^{Z_{\text{sy}(k-1)}} + e^{Z_k}\right), \\ \mu_{\text{Zsy}(k)} &= \mu_{\text{Zsy}(k-1)} + G_1, \\ \sigma^2_{\text{Zsy}(k)} &= \sigma^2_{\text{Zsy}(k-1)} + G_2 - 2 * \sigma_{\text{Zsy}(k-1)} \frac{G_3}{\sigma_{w(k)^2}}, \end{aligned} \tag{10}$$

where w_k and G_i ($i = 1, 2, 3$) are defined in (11). Finally set of (12)-(13) illustrates how the G_i's can be computed according to [12]; this slightly modified implementation was intended to circumvent the round off error of integration of the original Schwartz and Yeh implementation [13]

$$\begin{aligned} w_k &= Z_k - Z_{\text{sy}(k-1)}, \\ G_i &= G_i(\mu_{wk}, \sigma_{wk}) \quad i = 1, 2, 3, \\ \mu_{wk} &= \mu_{Zk} - \mu_{\text{Zsy}(k-1)}, \\ \sigma^2_{wk} &= \sigma^2_{Zk} + \sigma^2_{Z_{\text{sy}(k-1)}}, \end{aligned} \tag{11}$$

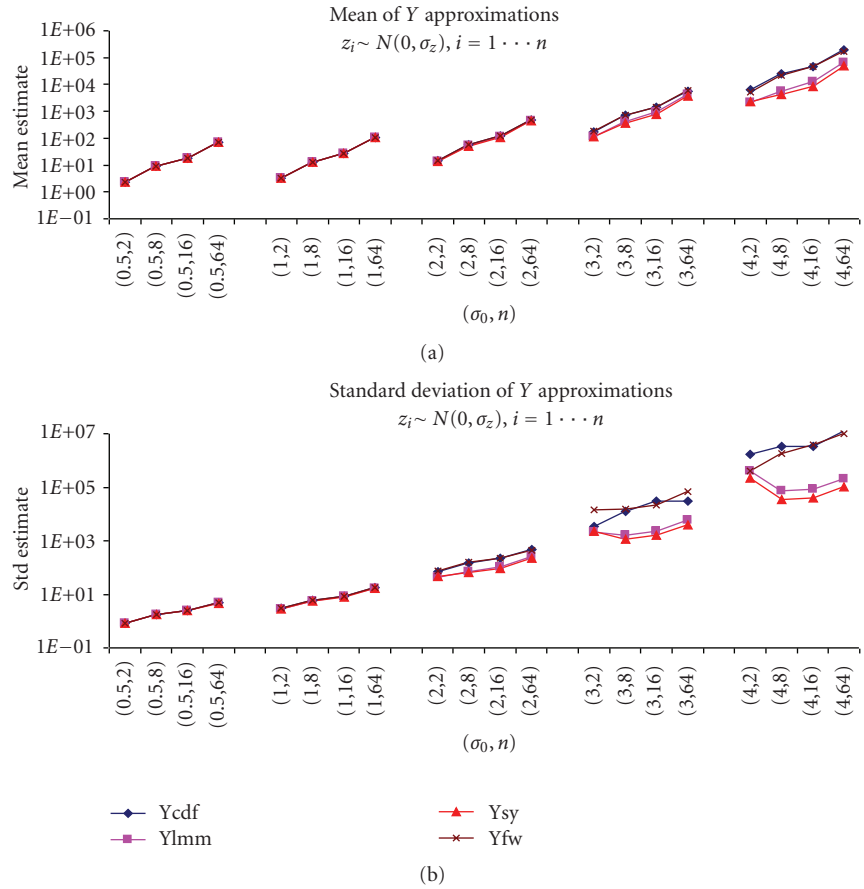


FIGURE 7: Plot of the mean and standard deviation of Y obtained from the different approximations. SY and LMM methods that match moments of $\ln(Y)$ falls behind for larger σ_z (>1 or 4 dB).

$$G_1 = A_0 + I_1,$$

$$G_2 = I_3 + 2I_4 + \sigma_{wk}^2 I_0 + \mu_{wk} A_0,$$

$$G_3 = \sigma_{wk}^2 (I_0 + I_2),$$

$$A_0 = \frac{\sigma_{wk}}{\sqrt{2\pi}} e^{-\eta^2/2} + \mu_{wk} I_0,$$

$$\eta = -\frac{\mu_{wk}}{\sigma_{wk}},$$

$$I_i = \int_0^1 h_i(v) v^{-1} dv = \int_{-\infty}^{\infty} 2h(v) vu \cosh x \, dx = \int_{-\infty}^{\infty} H(x) dx,$$

$$u = \exp(-2\sinh x); \quad v = (1+u)^{-1}. \tag{12}$$

Thus, at each step of the recursion, we compute the mean and standard deviation of w_k. The integrals I_i are then numerically computed using the functions h defined in (13). Their values are used to solve for the G_i's in order to evaluate (10). The final estimate for the Z_{SY} mean and standard deviation is reached at $k = n$

$$h_i(v) = \begin{cases} \frac{1}{\sqrt{2\pi}} e^{-(\ln v - \eta)^2/2}, & i = 0, \\ [f_w(\ln v) - f_w(-\ln v)] \ln(1+v), & i = 1, \\ [f_w(\ln v) - f_w(-\ln v)] (1 + v^{-1})^{-1}, & i = 2, \\ [f_w(\ln v) - f_w(-\ln v)] \ln^2(1+v), & i = 3, \\ -f_w(-\ln v) \ln(v) \ln(1+v), & i = 4, \end{cases} \tag{13}$$

$$\text{fw} = \frac{1}{\sqrt{2\pi {\sigma_{wk}}^2}} e^{-(w_k - \mu_{wk})^2 / 2{\sigma_{wk}}^2}.$$

3.3. Log Moment Matching Approximation. While the previous two approaches find the moment matching using analytical or recursive analytical solutions, this approach relies on sampling instead to compute the moments according to (10) [9]. Similar to the SY method, it estimates the 1st two moments of $\ln(Y)$. It does not pertain to a closed-form solution, but it maintains the lognormal assumption. Thus, the recursive analytical approach in SY and this

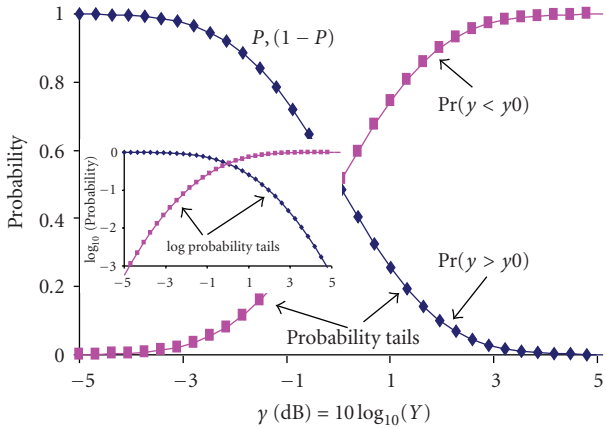


FIGURE 8: To highlight the tail region modeling, we rely on a tail log probability plot (small graph). It is derived from the density function and its complement.

approximation do not always converge as we will see in Section 3.5, especially that the SY assumptions is held exact when $n = 2$

$$Y \approx e^{Z_{\text{lmm}}},$$
$$\mu_{Z_{\text{lmm}}} = E[\ln(Y)], \tag{14}$$
$$\sigma_{Z_{\text{lmm}}} = \sqrt{\left(\left[E\left[(\ln Y)^2\right] - E[\ln(Y)]^2\right]\right)}.$$

3.4. CDF Matching. This is the closest possible to the true sum. Unlike the previous approximations, it does not rely on the lognormal approximation. We build a piece-wise linear probability density function from the Monte Carlo samples. Our goal is to demonstrate the difference between this approach and other lognormal approximations, when it comes to tail regions modeling and the resultant probability of fail estimations. Key features of the problem are

(i) the piece-wise linear fit for the density function of $\ln(Y)$ is non-Gaussian.

  (a) $g_{\text{pwl}}(\cdot)$ is defined by pairs (P_j, Y_{0j}), such that $\text{Prob}(Y > Y_{0j}) = P_j$, hence $\text{Prob}(\ln(Y) > \ln(Y_{0j})) = P_j$.

  (b) The pwl function can be sparse in the center of the distribution and more dense in the tails to adequately model the low fail probability region. In an extreme fashion, the tail probabilities can be recomputed from tail samples to avoid interpolation errors.

(ii) generating the Y samples is cheap, so is the I_{sum} sample once the function or even tables of $(I, \delta V_T)$ are available. Interpolation, bootstrapping, and other techniques can reduce the number of real simulations needed and still enable good confidence in the density function. After all, the previous approaches do rely on the availability of a closed form function for $I(\delta V_T)$. The number of samples is inversely proportional to the tail probability of interest; for example, if we are looking for accurate probabilities in the range of $1e-4$, then we need to have replications of samples that are larger than $1e4$. Replications add to the confidence in the expected tail probability. After all the interest is in the CDF tails mainly,

(iii) finally, this model can accommodate any complex nonlinear function $(I = f(\delta V_T))$; even if it is different from the exponential approximation above,

(iv) most importantly, once the distribution of I_{sum} is available, V_T distribution is derived by reverse fitting $(f^{-1}(I_{\text{sum}}))$ samples.

Note that for purposes of comparison, in this study all methods share the same exponential current model.

3.5. Theoretical Experiments. In this section, we compare the different methods ability to model Y (Figure 4). We study both the moment matching abilities as well as the upper and lower CDF tails. The study is performed over different combinations of n and σ_z (the standard deviation of Z_i):

$$(\sigma_z = 0.5, 1, 2, 3, 4) \times (n = 2, 8, 16, 64) \tag{15}$$

σ_z range corresponds to the range 2 dB–16 dB. Recall that often 4 dB is considered as a critical threshold for how accurate FW methods can model the distribution. Also as demonstrated in [6] standard deviation of the threshold voltage of scaled devices can exceed 4–8 dB (this is based on the leakage coefficient in (1)).

For each (n, σ) combination, multiple replications of 1 million samples are studied; this enables good estimates for the low fail probabilities. Figure 5 plots the mean and standard deviation of $\ln(Y)$ obtained from the different approximations. FW method falls behind for larger σ_z (>1 or 4 dB). Recall that this method was intended to match the moments of Y and not $\ln(Y)$. *We note that the FW estimates do underestimate the mean and overestimate the variance of* $\ln(Y)$ *for larger dB values.* Finally, the SW and LMM methods do match the CDF well, yet they do differ from each other a bit for larger sigma values, given that the former is a recursive approximation.

Figure 6 illustrates histogram (pdf plot) of $\ln(Y)$; the FW method does not model the body of the (Z or δVt) distribution well compared to other methods; this was also indicated in [6] for small n. The trend is even more obvious as the number of summands (n) increases. However, to obtain a complete picture, there is a need to study the lognorm matching and the tail regions which we will cover next.

Figure 7 plots the mean and standard deviation of Y obtained from the different approximations. SY and LMM methods that match moments of $\ln(Y)$ falls behind for larger σ_z (>1 or 4 dB). This is true for large n (>2), and we note that SY tends to particularly underestimate the variance of the sum of lognorms; this effect is most visible when the variables are identical and uncorrelated as is the case in this study. To study the tail regions, we rely on a "tail log plot"

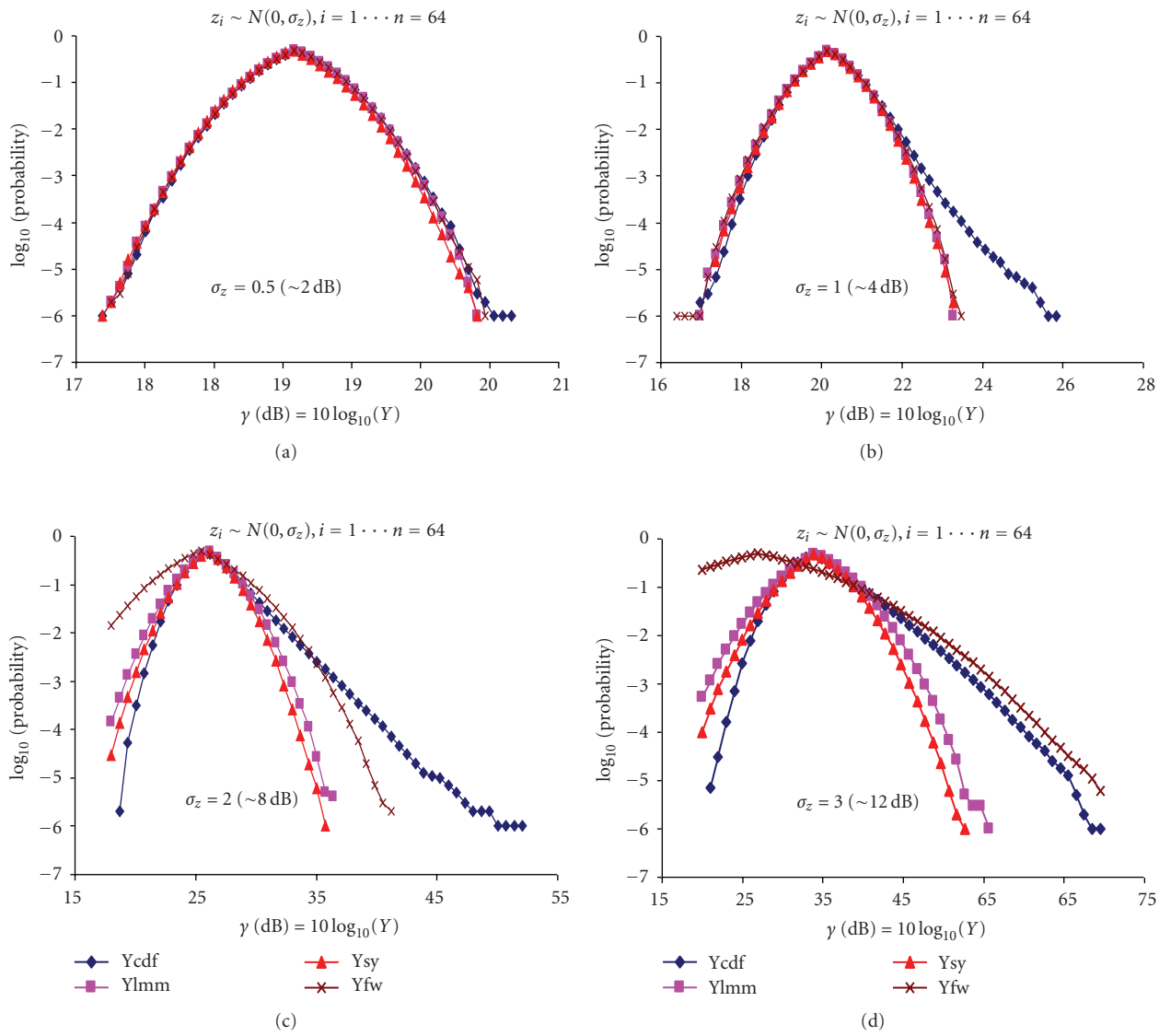


FIGURE 9: Log probability tail plot as function of σ_z; $n = 64$. For critical σ range (4–8 dB) SY, FW and LMM methods miss the right tail.

as illustrated in Figure 8. Without loss of generality we set the x-axis in these plots to be in dB; a small shift can mean large change in Y values. The plot is derived from the density function ($P(y > y0)$) and its complement ($1 - P$). It is such tail probabilities that are linked to the fail probabilities of a design. *Note that for the case of leakages, the right-side tail is critical for the fails (larger Y values correlate with larger leakage values in real applications).*

Figure 9 illustrates the tail probability plot as function of σ_z for $n = 64$. We get good match for all the methods for small σ. For critical σ values (4–8 dB), we note that SY and FW do miss the right tail modeling. As σ increases, FW tries to catch towards modeling the right tail by missing on the left tail model. Figure 10 illustrates the tail probability plots as function of n for $\sigma_z = 2$ (8 dB). SY and LMM methods have larger errors in modeling the right tail. FW error increases with increasing n.

4. Case Study: Leak-Down Time Comparisons

In this section, we extend the analysis to study the impact of the different modeling schemes, compared to the CDF matching method, on the probability of fail estimations. Figure 11 illustrates a summary of δV_{Teqv} distribution approximations based on the relation between δV_{Teqv} and Y in (2); except for the CDF matching method, δV_{Teqv} is modeled as Gaussian random variable $\sim N(\mu_{\text{eqv}}, \sigma_{\text{eqv}})$. The distributions are then used to analyze the leak-down of RBL in the circuit of Figure 12. The time for the bitline to discharge to 50% of the rail under extreme noisy corner voltage is then estimated.

Figures 13, 14, and 15 illustrate the normalized time-to-leak distributions for the case of 16 cells/bitline. s_0 represents a lower limit on the threshold voltage standard deviation of a 45 nm cell device; its value is set to the equivalent of $\sigma_z = 0.6$

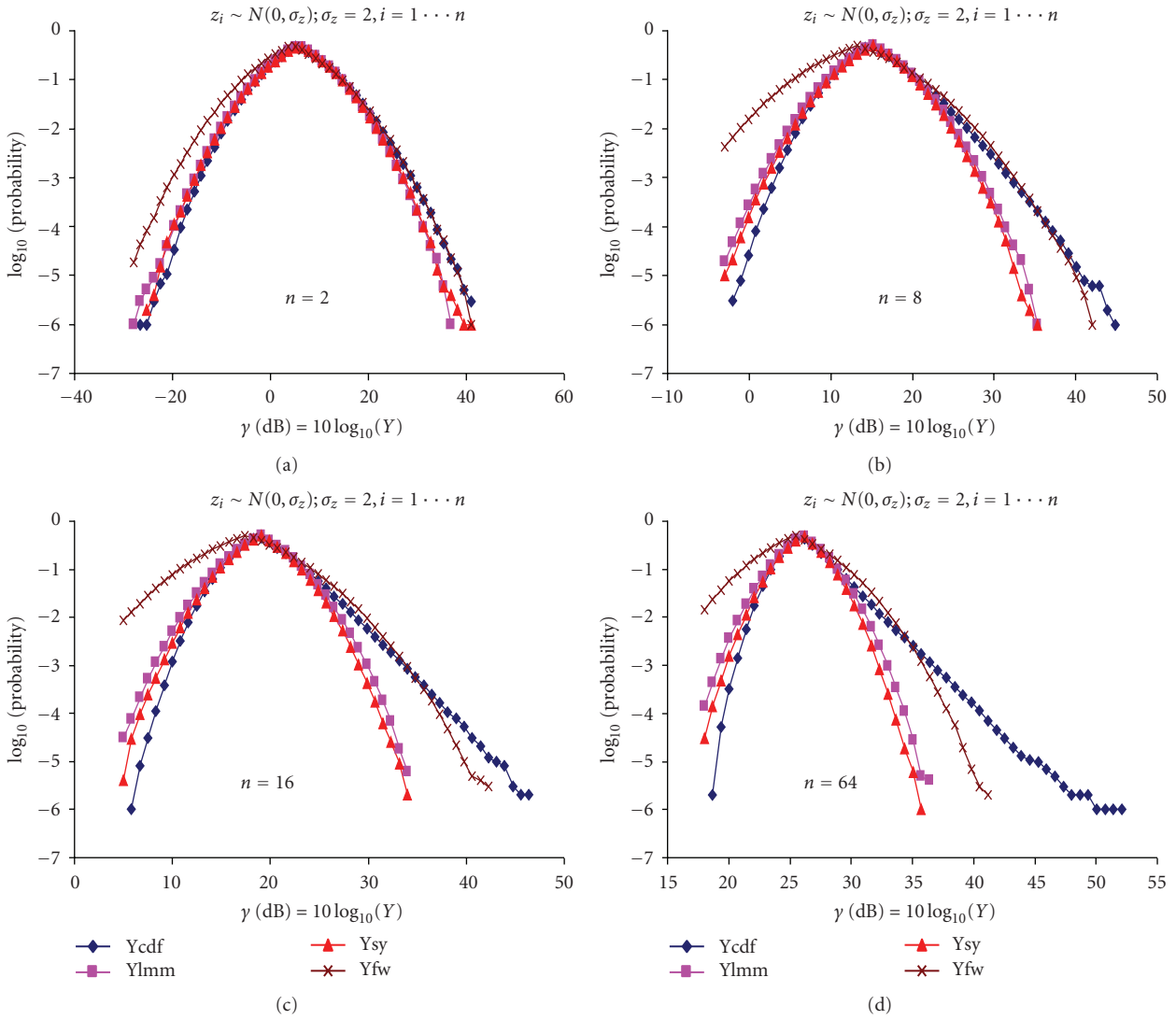


FIGURE 10: Log probability tail plot as function of n, $\sigma_z = 2$ (8 dB). SY and LMM methods show large error in the right tail model. FW error increases as the number of summands increases.

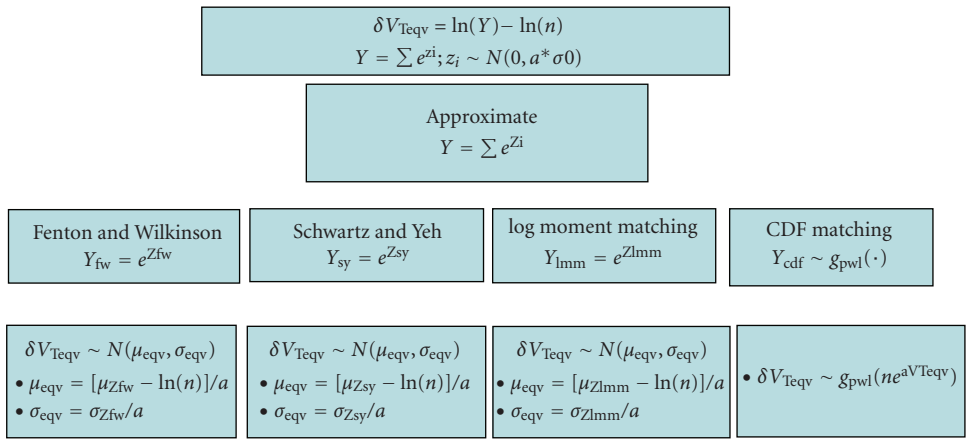


FIGURE 11: Based on Figure 4, distributions for the δV_{Teqv} can be derived from the Y approximations.

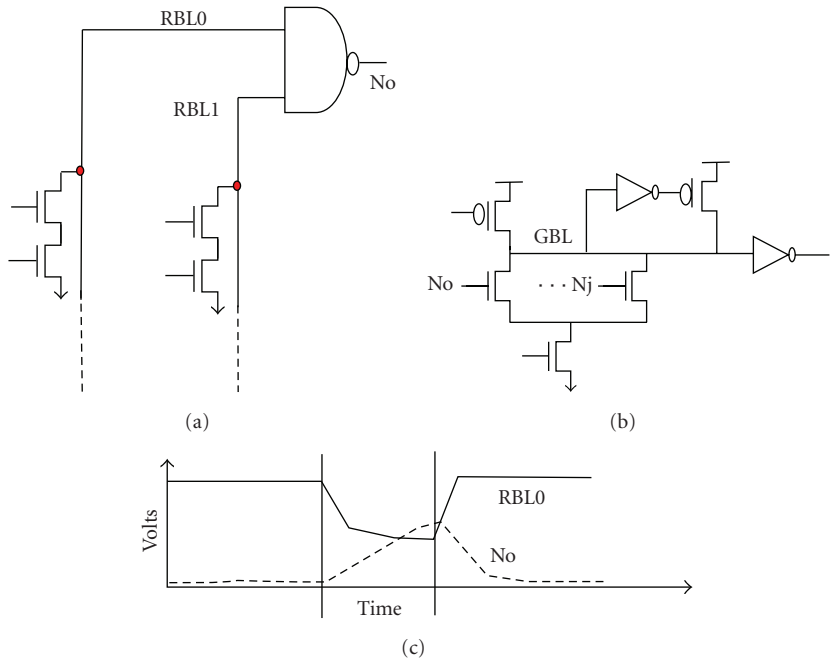


Figure 12: Due to leakages, the RBL may falsely discharge causing false evaluate of node N_0.

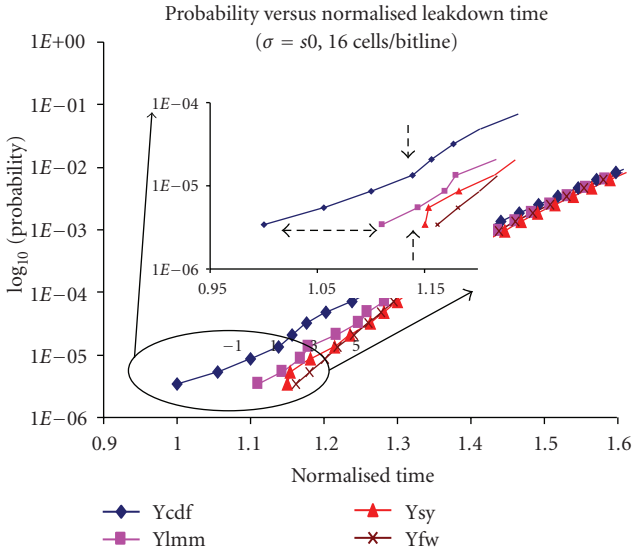


Figure 13: Normalized time-to-leak probability for $\sigma_{VT} \sim s_0 \sim$ 2.5 dB.

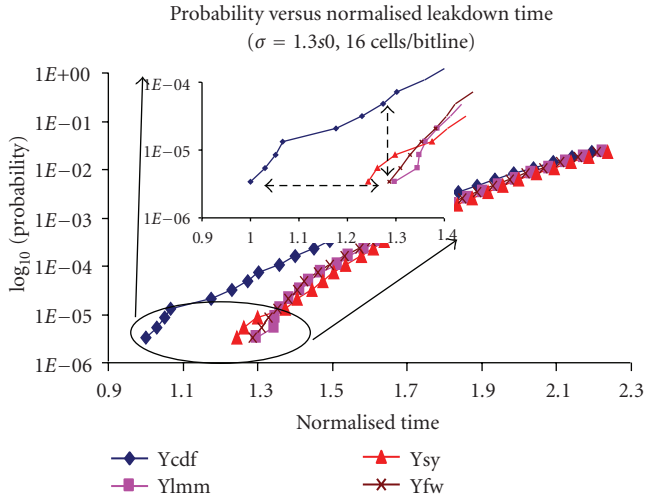


Figure 14: Normalized time-to-leak probability plot for $\sigma_{VT} \sim 1.3s_0$.

(or 2.5 dB). This is conservative especially that as technology scales we expect more variability and additional sources of variability like the random telegraphic noise can add to the V_T variation; cases for standard deviation of 1.3s0 and 1.6s0 are also plotted. SY, FW, and LMM methods overestimate the time-to-leak from 10% for s_0 to close to 100% for $1.6s_0$ (see horizontal arrow in the figures; this corresponds to system yield around 4.5 sigma). More importantly, this leads to underestimating the probability of the number of elements failing at a given leak-time. Note that time-to-leak values can be critical relative to operating frequencies and accurate prediction is needed for robust designs. Thus, we are interested in computing the ratio of the probability of fails (vertical arrows in figures) for predicted time-to-leak values. Figure 16 summarizes the ratio of the true (cdf) probability of fail compared to that of the other methods (SY, FW, and LMM) at their 4.5 sigma yield leak-time. Each experiment is based on the average of 25×1 million replications. This is done for the case of 16 and 64 cells/bitline and at increments of $0.1 * s_0$. We note that the SY, FW, and LMM methods underestimate the probability of fail 10×–147×.

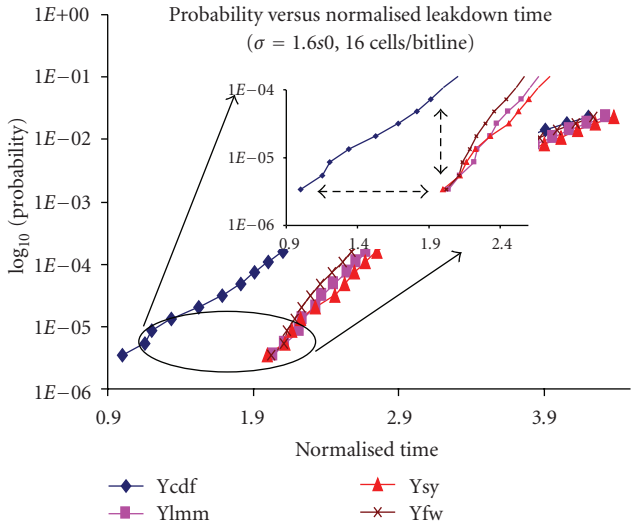


FIGURE 15: Normalized time-to-leak probability plot for $\sigma_{VT} \sim 1.3s_0$.

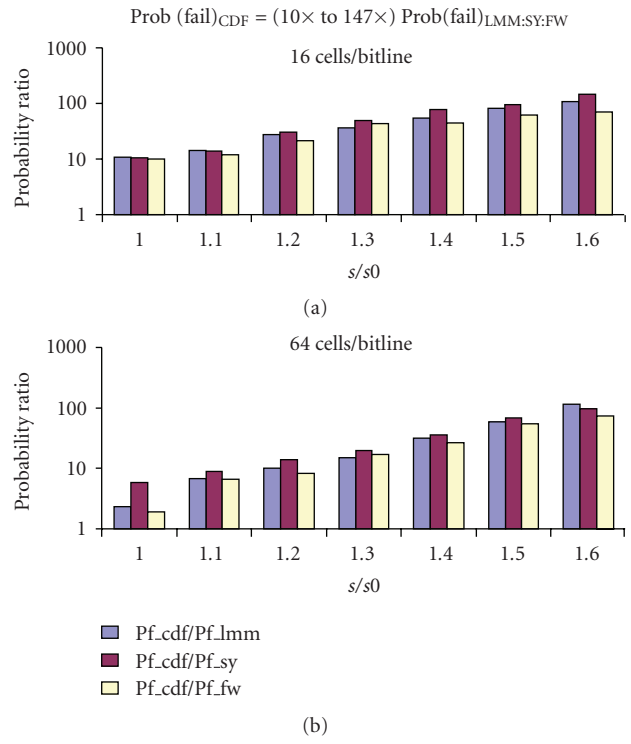


FIGURE 16: Probability of fail ratio is underestimated in SY, FW, and LMM methods.

5. Conclusions

We study the ability of different sums of lognormal approximation to emulate the leakage of multiple leaky devices by a single equivalent device. With the goal of rare event estimation, tail distributions are examined closely. Modeling the tail probability by CDF matching is found to be critical compared to the Fenton-Wilkinson and Schwartz-Yeh methods that are found to underestimate the sum of lognorms and hence overestimate the fail probability by 10×–147×; this trend is expected to increase as the variability increases with scaling technology.

References

[1] R. Joshi, A. Pelella, A. Tuminaro, Y. Chan, and R. Kanj, "The dawn of predictive chip yield design: along and beyond the memory lane," *IEEE Design and Test of Computers*, vol. 27, no. 6, pp. 36–45, 2010.

[2] E. Acar, K. Agarwal, and S. R. Nassif, "Characterization of total chip leakage using inverse (reciprocal) gamma distribution," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '06)*, pp. 3029–3032, May 2006.

[3] R. Kanj, R. Joshi, and S. Nassif, "Mixture importance sampling and its application to the analysis of SRAM designs in the presence of rare failure events," in *Proceedings of the 43rd Annual Design Automation Conference (DAC '06)*, pp. 69–72, 2006.

[4] A. Asenov, "Random dopant induced threshold voltage lowering and fluctuations in sub-0.1 fjim MOSFET's: a 3-D "atomistic" simulation study," *IEEE Transactions on Electron Devices*, vol. 45, no. 12, pp. 2505–2513, 1998.

[5] J. Gu, S. S. Sapatnekar, and C. Kim, "Width-dependent statistical leakage modeling for random dopant induced threshold voltage shift," in *Proceedings of the 44th ACM/IEEE Design Automation Conference (DAC '07)*, pp. 87–92, June 2007.

[6] S. H. Rasouli, K. Endo, and K. Banerjee, "Variability analysis of FinFET-based devices and circuits considering electrical confinement and width quantization," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '09)*, pp. 505–512, November 2009.

[7] L. F. Fenton, "The sum of log-normal probability distibutions in scattered transmission systems," *IRE Transactions on Communications Systems*, vol. 8, pp. 57–67, 1960.

[8] S. C. Schwartz and Y. S. Yeh, "On the distribution function and moments of power sums with log-normal components," *The Bell System Technical Journal*, vol. 61, no. 7, pp. 1441–1462, 1982.

[9] R. Kanj, R. Joshi, and S. Nassif, "Statistical leakage modeling for accurate yield analysis: the CDF matching method and its alternatives," in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED '10)*, pp. 337–342, Austin, Tex, USA, 2010.

[10] N. C. Beaulieu, A. A. Abu-Dayya, and P. J. McLane, "Estimating the distribution of a sum of independent lognormal random variables," *IEEE Transactions on Communications*, vol. 43, no. 12, pp. 2869–2873, 1995.

[11] A. Papoulis, *Probability, Random Variables and Stochastic Processes*, McGraw Hill, New York, NY, USA, 1991.

[12] P. Cardieri and T. S. Rappaport, "Statistics of the sum of log-normal variables in wireless communications," in *Proceedings of the 51st Vehicular Technology Conference (VTC '00)*, pp. 1823–1827, May 2000.

[13] C. L. Ho, "Calculating the mean and variance of power sums with two log-normal components," *IEEE Transactions on Vehicular Technology*, vol. 44, no. 4, pp. 756–762, 1995.

Hindawi Publishing Corporation
VLSI Design
Volume 2011, Article ID 503025, 10 pages
doi:10.1155/2011/503025

Review Article

Shedding Physical Synthesis Area Bloat

Ying Zhou,[1] Charles J. Alpert,[1] Zhuo Li,[1] Cliff Sze,[1] and Louise H. Trevillyan[2]

[1] *IBM Austin Research Laboratory, Austin, TX 78758, USA*
[2] *IBM Waston Research Laboratory, Yorktown Heights, NY 10598, USA*

Correspondence should be addressed to Ying Zhou, nancyzhou8@gmail.com

Received 1 October 2010; Accepted 11 December 2010

Academic Editor: Shiyan Hu

Area bloat in physical synthesis not only increases power dissipation, but also creates congestion problems, forces designers to enlarge the die area, rerun the whole design flow, and postpone the design deadline. As a result, it is vital for physical synthesis tools to achieve timing closure and low power consumption with intelligent area control. The major sources of area increase in a typical physical synthesis flow are from buffer insertion and gate sizing, both of which have been discussed extensively in the last two decades, where the main focus is individual optimized algorithm. However, building a practical physical synthesis flow with buffering and gate sizing to achieve the best timing/area/runtime is rarely discussed in any previous literatures. In this paper, we present two simple yet efficient buffering and gate sizing techniques and achieve a physical synthesis flow with much smaller area bloat. Compared to a traditional timing-driven flow, our work achieves 12% logic area growth reduction, 5.8% total area reduction, 10.1% wirelength reduction, and 770 ps worst slack improvement on average on 20 industrial designs in 65 nm and 45 nm.

1. Introduction

Physical synthesis is the critical part of modern VLSI design methodologies. It refers to the process of placing the logic netlist of the design as well as sizing/adding/removing/logic changing cells, concurrently optimizing multiple objectives under given constraints, where objectives and constraints are choices among area, power, timing, and routability depending on design characteristics. In the last decade, timing closure has been the main focus of physical synthesis flow [1], partially due to interconnect delay dominance over gate delay from technology mitigation.

So why do we suddenly care about area bloat? In 65 and 45 nm technologies, design companies tend to pack more logic functionalities into a small-sized die to balance the expensive fabrication cost, while they also want to keep low power budget to maintain their competitive margin. Such requirement could break the traditional timing-driven flow which tends to consider area as merely a constraint and overdesigns the chip.

Area bloat could cause several problems.

(1) More power consumption: area is the first-order estimation of the power, especially for dynamic power. For leakage power, smaller area device tends to have less leakage even in the gate library family with same threshold voltage (Vt).

(2) Congestion problems: there are many causes for congestion problems, such as inferior floorplan and bad placement. Area bloat is one significant cause, which creates high logic gate density in certain regions, and there are no enough tracks to route all nets.

(3) Timing problems: when the chip area has been fully occupied, there is no extra space for optimizations to further improve timing, or new buffers and resized gates are moved a long distance during legalization and big timing degradation happens.

Each problem described above or their combination could cause designers to increase die size, restart floorplanning and complete physical synthesis flow, which in turn lengthen the total time-to-market.

One example of area bloat causing congestion problems is shown below. For an industrial 45 nm design with 102 K input gates, we first run a traditional timing-driven flow and a global router with 5% detour length control to

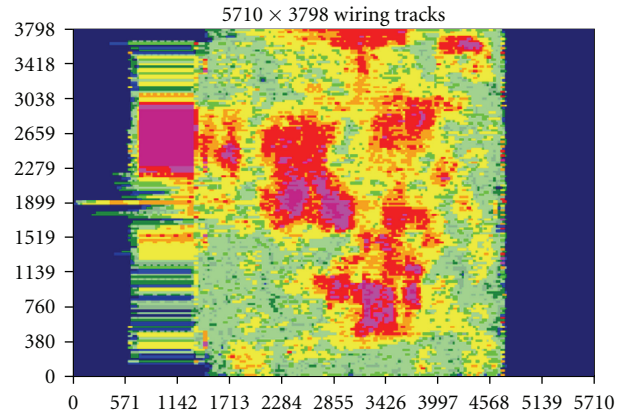


Figure 1: Horizontal congestion from timing driven physical synthesis flow.

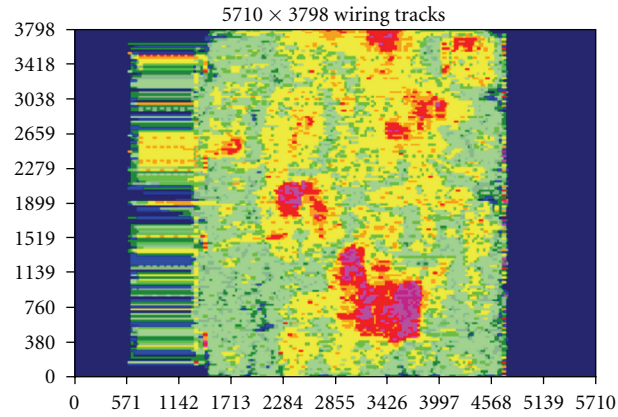


Figure 2: Horizontal congestion from area efficient physical synthesis flow.

measure the congestion. The congestion picture is shown in Figure 1 and the pink area is most congested one. The average congestion metric (measured by taking the worse 20% congested nets and averaging the congestion number of all routing tiles these nets pass through) is 94%, with 5535 nets pass through 100% routing congested tiles. Then with the techniques later discussed in this paper, the area is reduced by 8%, and the congestion picture is shown in Figure 2. The average congestion metric decreases to 89% with only 2856 nets passing through 100% routing congested tiles. The output netlist with the new technique can actually be routed with some further cell spreading techniques, where the original design has even no free space to be spread. Therefore, it is important to achieve a minarea physical synthesis flow.

The major sources of the area bloat from the physical synthesis are buffering and gate sizing. Even though there are lots of existing literatures on these problems, there are still practical constraints that existing approaches do not model correctly.

1.1. Buffer Insertion. Buffer (repeater) insertion, is a popular technique to reduce interconnect delay and fix electrical violations. Perhaps the most well-known buffer insertion algorithm is Van Ginneken's classic dynamic programming algorithm [2], which has led to several improvements for the kernel algorithm, such as speedup [3], resource control [4, 5], and slew constrained buffering [6]. Other extensions or related work include buffer tree construction [7, 8], buffering with more accurate delay models [9], buffering for noise reduction [10], and simultaneous wire sizing and layer assignment [11].

Most of these literatures focus on single algorithm for a particular problem. However, creating an area efficient flow based upon these existing techniques and the way to handle all practical constraints are rarely discussed, which have big impact to the design area at the end of the flow. To list a few, we mention the following.

(i) Should one use slew constrained buffering or timing driven buffering (they have different area and timing results)?

(ii) How to set the input slew and slew constraint for buffering algorithms, which has the big impact on the area?

(iii) Can one still use Elmore delay and linear gate delay model in buffer insertion for modern designs and any area impact?

(iv) How to handle rising and falling transitions inside the algorithm?

The impact of slew constraint on buffer area is shown in Figure 3. The experiment is done at a 5 mm long line on a 2X wide/thick layers in 45 nm technology where buffers are placed at the max distance to meet the specified slew constraint in a repeated pattern. As slew constraint becomes smaller, the distance between buffers is smaller, which results in more buffers and bigger buffer area. On another hand, the signal delay per mm, the sum of buffer delay and wire delay divided by the slew reach length, is measured and shown in the same figure. One can also see that by adjusting the slew goal, one can achieve the optimal delay for a buffered wire without performing timing-driven buffering.

The impact of input slew and rising/falling inputs is illustrated in Figure 4, where we choose a buffer from a 45 nm node buffer and show the relationship between the delay and capacitance under different input slew values and rising/falling input transitions. It is clear that the delay is also quite sensitive to the input slew as well as the rising and falling input directions, and the difference could be 10% to 15%.

1.2. Gate Sizing. Gate sizing has also been extensively studied in the literature. Most early work assumes the library is continuous, models the sizing problem as a convex programming with closed form solution [12] or Lagrangian relaxation [13]. These works ignore the fact that most cell library-based designs have discretized gate sizes. Later, some rounding techniques have been proposed to map the continuous solutions to the discrete grid [14]. More recently,

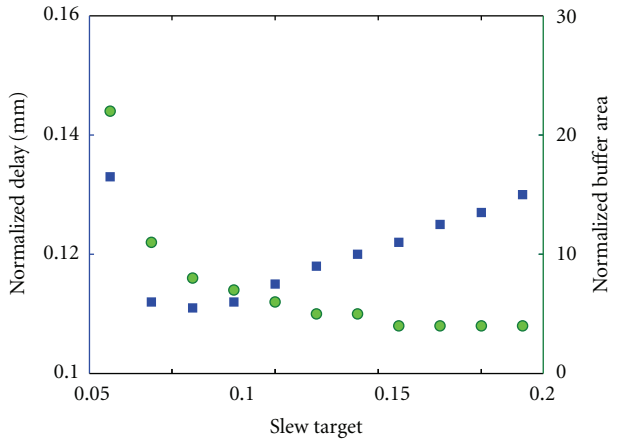


Figure 3: Slew constrait (ns) versus buffer area relationship is shown in green dots. Slew constraint (ns) versus signal delay per mm relationship is shown in blue squares.

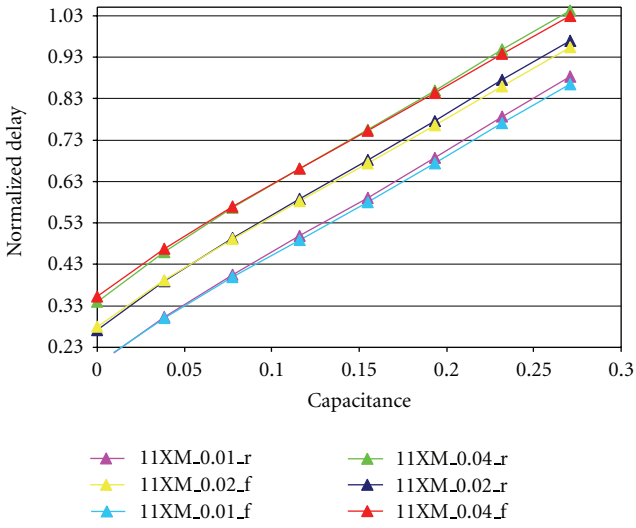


Figure 4: Delay versus cap for a buffer in 45 nm node. Three different input slew values, 10, 20, and 40 ps, are used here. 11XM_0.2_r refers to 11X driving strength buffer, 20 ps input slew and the rising inputs.

Liu and Hu [15] proposed to use dynamic-programming style algorithms for solving discretized gate sizing problem directly. However, the slew impact is still ignored when propagating the solutions in the dynamic programming. Also all previous work tends to optimize the worst critical path, where the sum of negative slacks (referring to Figure of Merit(FOM)) are always ignored, which is also an important factor to measure design quality, especially when the worst critical path stuck during the optimization with either logic structure problems or wrong timing assertions.

The following figures show the relationship between delay and area for one complete buffer library in 65 nm node (Figure 5) and one in 45 nm node (Figure 6). The delay is normalized and the buffer area is measured by its width. The capacitance load is the sum of the capacitance of a typical

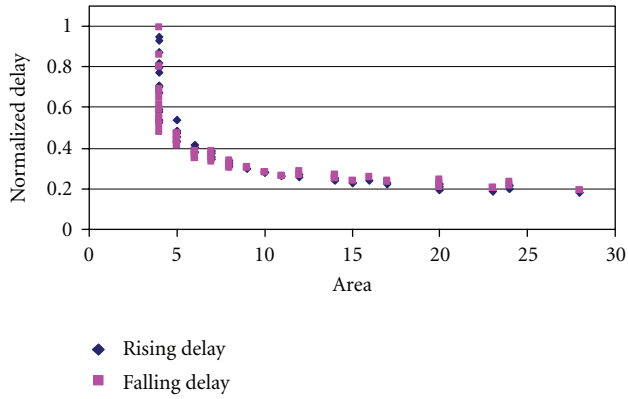


Figure 5: Delay versus area for a buffer library in 65 nm node.

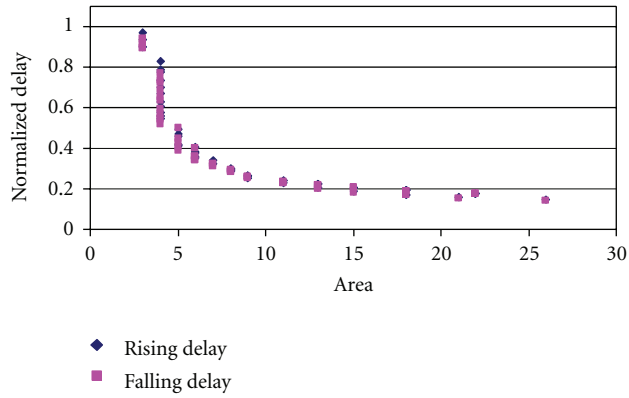


Figure 6: Delay versus area for a buffer library in 45 nm node.

interconnect length plus the capacitance of a typical buffer for the corresponding technology. The input slew is set at 200 ps for 65 nm node and 40 ps for 45 nm node. Both rising and falling delay values are shown in the figures. Note that not only the library is discrete (the area of a gate is generally measured by its width in the standard cell methodologies; since the vertical track is generally fixed) there are also many buffers with the same area (or footprint) but totally different delay values. It is caused by different N/P transistor strength for rising/falling balance or the choice of transistor sizes in the first and second inverters. Therefore, all assumptions such as "convex" and "continuous" do not work for cell based designs, and even rounding approach will meet problems when many gates share the same area. Also, as shown in Figure 4, the delay is sensitive to the input slew too. Such relationship between delay and area is also found in other logic gate library, such as AND/NAND/OR/XOR gates.

1.3. Our Contribution. In this work, we present practical guide and experience of how to put an efficient incremental optimization step inside a physical flow with good area/timing tradeoff. Papers in this category are rarely seen. For a designer or a new EDA startup, we hope this work can provide a quick guide without looking through 100 papers on buffering and gate sizing. Our contributions in this paper are as follows:

(i) an area efficient iterative slew-tighten approach for slew-driven buffering and gate sizing (iterative EVE);

(ii) a simple area efficient timing-driven gate sizing method for cell library designs;

(iii) a new area efficient optimization flow with practical buffering and gate sizing techniques to handle modern design constraints.

2. Overview of Existing Physical Synthesis Flow

A timing-driven physical synthesis flow described in [16] includes the following steps: (1) initial placement and optimization, (2) timing-driven placement and optimization, (3) timing-driven detailed placement, (4) clock insertion and optimization, and (5) routing and post routing optimization; A simple diagram is shown in Figure 7(a) with the placement and optimization part, where refine step refers to optimization at the finer level.

Further, optimization can also be broken into 3 steps shown in Figure 7(b): (1) electrical correction, (2) critical path optimization, and (3) histogram compression.

The purpose of electrical correction is to fix the capacitance and slew violations with buffering and gate sizing. In general, one wants to first perform electrical correction in order to get the design in a reasonable state for the subsequent optimizations. In [17], an electrical violation eliminator (EVE) technique is used to fix electrical violations through fast slew-based gate sizing and buffering.

Then more accurate but slower timing-based buffering approach and gate sizing method are applied in the critical path optimization and histogram compression stages for the rest of the critical paths. During critical path optimization, a small subset of the most critical paths are examined and optimization is performed to improve the timing on those paths. Sometimes, critical path optimization still can not close on timing, and physical synthesis could return its best so far solution. The purpose of the histogram compression phase is to perform optimization on these less critical, but still negative paths.

This flow has the speed and timing performance advantage as shown in [16]. However, it has several drawbacks to cause the area bloat. In the following sections, we describe two main techniques to shed the area bloat for this flow, though the merit of techniques may benefit other flows too.

3. Iterative EVE

The concept of original EVE technique is to combine slew-based gate sizing and slew constrained min-cost buffering [6], and process gates from primary output (or latch inputs) to primary inputs (or latch outputs) in the combinational circuits. If a net has no violations, size the source gate down to save area and reduce load on inputs. If a net has violations, size the source up; if the biggest size cannot fix the slew violation, perform buffering on the net. This approach has several advantages: (1) combining buffering and gate sizing seamlessly, (2) high efficiency, and (3) no reconvergence problem (all decisions are local for electrical corrections).

EVE is motivated by the fact that most nets in modern designs either (1) have only electrical violations but without timing violations, or (2) have positive slacks after electrical violations have been fixed.

More importantly, slew constrained min-cost buffering is much faster than timing-driven buffering because slew constraints can usually be handled locally.

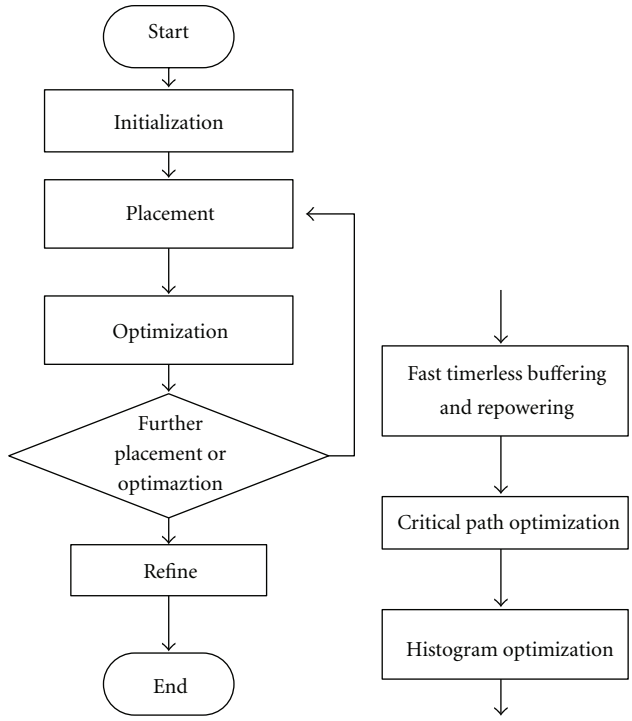
However, as explained in [6], the buffering algorithm is very sensitive to the input slew and the slew target at the sinks. Buffering with a tight slew target (e.g., the slew target is set to be the saturation slew in a long optimal buffered-line [12]) can usually achieve similar timing performance as timing-driven buffering, however, surplus buffer insertion will be done for nets with positive or near-positive slacks and loose original slew targets. On the contrary, using a relaxed slew target can save area but it unacceptably sacrifices performance, and more nets need timing driven buffering afterwards. It also slows down the runtime because timing verification has to be done for each new solution to make sure no timing degradation occurs. Similar situation happens to gate sizing operations, though we use buffering as the main example for the rest of the section. In the following, we propose a new iterative method which gradually tightens slew target.

Instead of starting with a tight slew target, the initial slew target is set based on the operating frequency of the design, which could be the tightest slew constraint for all data phases excluding scan. Sometimes, the initial slew constraint is provided by the designers. Comparing to the initial slew target, each net may have its own slew constraint (e.g., from design rules), and the tighter one is used during optimization for each net. In the first iteration, most of the nets may end up using its own slew constraint, but later in the process, the global slew target gets tighter and eventually overrides the local ones to guide timing optimization.

For the input slew, we start with the saturation slew along a long optimal buffer chain. This is applicable for nets which need the most aggressive buffering. For the other nets, this is a conservative assumption and results in better area since the saturation slew is usually smaller than the initial slew target.

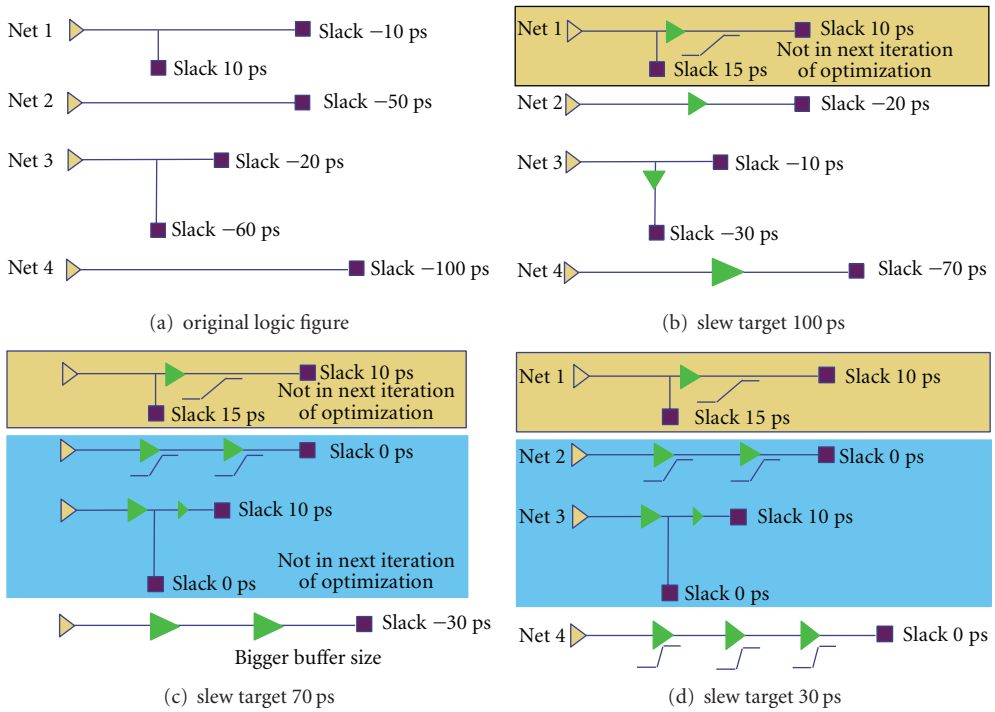
With these settings, we run EVE and follow with a full static timing analysis. Then, the slew target is reduced by a given percentage (say 10% or 20%) and input slew is updated by taking a set of worst paths and averaging their input slew. EVE is run again for all negative nets with the right to left order. In all iterations, buffers on negative paths are completely ripped up and rebuilt. The process is repeated until the slew target is smaller than the saturation slew of an optimal buffered chain. Our experiments generally converge in 3 to 5 iterations for 65 nm and 45 nm technology, and the overhead due to static timing analysis is acceptable. This approach is significantly faster than the traditional timing-driven buffering approach for all nets, which returns similar area results.

An exemplary circuit is shown in Figures 8(a) to 8(d). In this simple example, there are 4 nets and the initial design structure in Figure 8(a) could be an optimized placement



(a) a simple physical synthesis flow (b) detail steps of optimization

FIGURE 7: Flow diagram.



(a) original logic figure (b) slew target 100 ps

(c) slew target 70 ps (d) slew target 30 ps

FIGURE 8: A simple example for iterative EVE.

from a commercial tool, a random placement or a custom placement. In Figure 8(a), there is no repeaters. The slack of all sinks is negative as shown in figure.

Figure 8(b) represents the structure after the first iteration of EVE which generates the first optimized design structure. This iteration uses a global slew target of 100 ps, and a buffer has to be inserted in each net in order to meet the slew target. These buffers may have different sizes in a buffer library. After the first iteration, the slacks of all nets are shown in Figure 8(b). Since the slack of both sinks in the first net become positive, the first net is skipped in the future iterations.

After the first iteration, the slew target is down to 70 ps from 100 ps. Figure 8(c) shows the result of the second iteration of EVE. In this iteration, additional buffers have been inserted into the second, third nets to make their slack positive. They will then be skipped in future iteration.

In the third round, the slew target becomes 30 ps. The optimized design structure is shown in Figure 8(d). Since the first three nets were skipped for this iteration, additional buffer is only inserted in the forth net as shown in Figure 8(d). The final slack of the forth net is 0 ps. This is an ideal case where all nets now have positive slack, and further timing-driven optimization is not necessary.

While the example of Figures 8(a) to 8(d) illustrates only 4 nets and a total of eight inserted buffers, in real designs, the number of nets is typically in the thousands, with the insertion of as many as 500,000 buffers.

With this method, we can insert as fewer buffers as possible to meet timing requirement and the total area and wirelength is greatly reduced.

4. Area Efficient Timing-Driven Gate Sizing

Timing-driven gate sizing is used in the critical path optimization and histogram compression stage. It needs to be accurate, incremental, and harmless.

As discussed in the Section 1.2, the discrete nature of the cell library for standard cell-based designs, the slew impact and the FOM problems, make existing approaches such as convex programming either be unrealistic to use, or inaccurate. Also, previous approaches tend to find the sizing solution for the whole circuit, or a group of hundreds to thousands of gates with internal delay models, and then apply it. The scale of changes, combined with the model inaccuracy, may result in big rejection rate from the static timing analysis with slew propagation, and even some good partial solutions may be thrown away.

In our implementation, we choose to use a simple gate sizing approach. We first give an order of all boxes (could be based on slack or sensitivity), and then work on each single box at each time. After choosing the right power level, perform the incremental timing analysis to update the timing graph, and move on to the next box. It looks like quite naive, but is more accurate for the local change, and also tends to give the best FOM result since even a box is on the not-near-critical paths as long as the slack is still negative and can be improved, a better solution will be chosen. The whole process can also be iterated. One can speed up the process by simply limiting the update scope of static timing analysis engine when every size of the gate is evaluated.

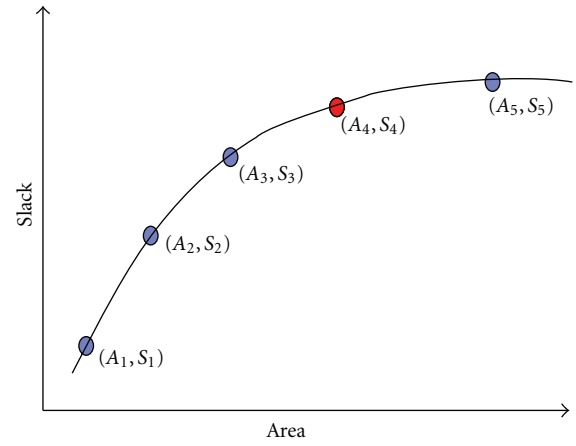


Figure 9: Area aware gate sizing.

We can resize boxes as long as the slack is improved, however, the slack only gets a little bit improvement with lots of area resource sometimes. To be area efficient, the minimum improvement threshold δ is defined as the minimum required slack improvement per unit area change. When a particular gate is resized, we only accept the new solution, if the slack improvement of new solution compared to the previous best solution is bigger than δ times the area difference. As shown in Figure 9, rather than choosing the best slack solution with area A_5, we pick the solution with much smaller area A_4, with acceptable timing degradation.

5. New Area Efficient Optimization Flow

The new flow assembled from iterative EVE is illustrated in Figure 10. The area aware gate sizing is the critical path and histogram stages, where the cost can be tuned for different design requirement.

6. Other Practical Techniques

In any physical synthesis flow, a timing-driven buffering tool plays an essential role. We implement the techniques described in [5] to control the buffer resources, but to make it more practical, several tunings need to be done.

Handling Rising and Falling Transitions. The buffering algorithm needs not only to handle polarities and inverters, also to distinguish the rising/falling signals during the bottom-up dynamic programming since the delay for both edges are noticable different (Figure 4).

Delay Modeling. Elmore delay model is too conservative and causes overbuffering where moment-based computation is too slow. We use scaled Elmore delay model (0.8 as factor) and linear gate delay models, and found the buffer locations are quite close to the solution from the moment-based wire

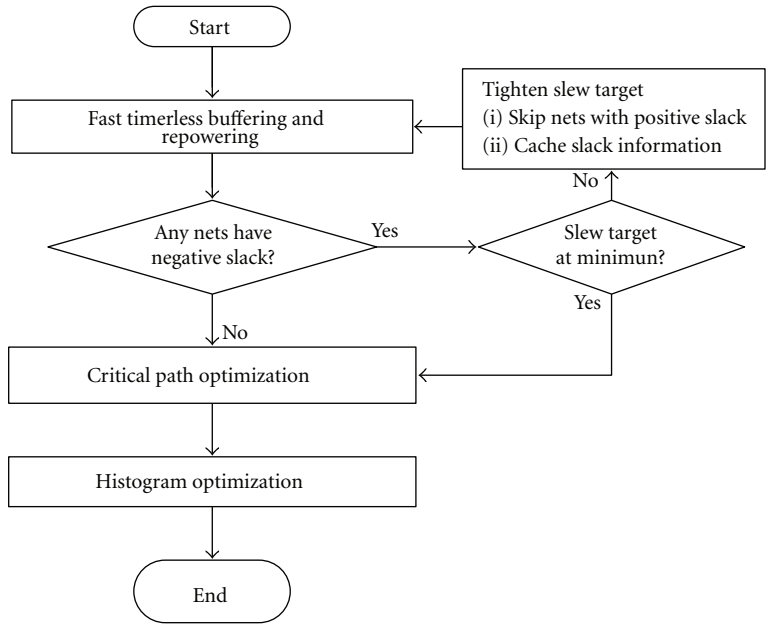


Figure 10: New optimization flow.

delay models and lookup tables-based gate delay models, while the runtime is much faster. We suspect the buffering is a global scaled operation.

Speed. We implement following techniques: range search tree pruning, convex and predictive pruning techniques to gain speedup from solution pruning; buffer solution storage technique in [3] to gains speedup by avoiding naive copying solution list during adding wire operations; layer assignment techniques in [18] to gain speedup by efficiently utilizing available high level metals for each technology.

7. Experiments

We implement the new optimization flow in C++ and embed in the flow shown in Figure 7(a). Twenty industrial designs (eighteen 65 nm and two 45 nm) are selected, ranging from 100,000 to 1 millon gates in the input netlist. All experiments are run on a 2.9GHz machine with Linux operation system.

7.1. Iterative EVE versus Single EVE. In this experiment, we compare our iterative EVE algorithm with the single EVE algorithm which uses the aggressive slew target to get the best timing. Both optimizations are performed after initial placement, which produce a huge slack and FOM improvement since no net (including high fanout nets) has been buffered (other than polarity inverters). We compare the worst slack (WSLK) improvement (the difference of before/after worst slack), FOM improvement, and area increase due to the optimization. Data for only 10 designs are shown in Table 1 to save the space. In summary, iterative EVE approach uses as less as 20% area of single EVE, while achieving similar timing quality. There is runtime overhead since we run multiple iterations, which is generally 2 to 4 times depending on the design.

7.2. Timing-Driven Gate sizing. In this section, experiment results with different minimum improvement threshold δ for timing-driven gate sizing are shown in Table 2. The timing-driven gate sizing is performed in "critical path optimization" stage after iterative EVE, which also explains the scale of the improvement compared to Table 1. The unit of δ is picoseconds per unit area change. When $\delta = 0$, it gives the best timing, and when $\delta > 0$, area cost is considered. From Table 2, the increased area becomes smaller when δ increases. It is interesting to see that when δ is big enough, the area starts to decrease and one can still achieve the worst slack and FOM improvement. Generally we do not use such aggressive minimum improvment threshold like $\delta = 0.05$ in our flow, area saving and final timing performance are considered at the same time. Based on our experimental results, $\delta = 0.005$ is a good choice.

7.3. Overall Flow Comparison. In this part, we put iterate EVE and area aware gate sizing (with $\delta = 0.005$) in the flow and compare to the baseline (single EVE and $\delta = 0$). Both flows go through complete physical synthesis, include 2 iterations of placement (the second one is timing-driven placement with net weight updated according to the timing information), optimizations, timing-driven buffering, detail placement, legalization and the refine part. Both flows also use the practical techniques mentioned in Section 5 too.

For all experiments, we compare area, worst slack (WSLK), FOM, wirelength (WL) and runtime at the end of physical synthesis and the results are shown in Table 3.

From Table 3, our new flow saves 5.8% total area compared to the baseline flow on average, and the maximum

Table 1: The QOR comparison for iterative EVE.

	Type (ns)	FOM Imp (ns)	WSLk Imp	Area Inc
test1	Single EVE	1150890.0	464.87	946003
	Iterative EVE	1153195.6	464.05	198297 (20.96%)
test2	Single EVE	3425630.0	718.20	677286
	Iterative EVE	3344172.7	593.24	189253 (27.94%)
test3	Single EVE	993808.0	125.45	369589
	Iterative EVE	1093409.7	148.62	151493 (40.98%)
test4	Single EVE	8564860.0	426.96	1221551
	Iterative EVE	8378022.8	395.96	490440 (40.14%)
test5	Single EVE	1416620.0	160.65	1611279
	Iterative EVE	1340335.3	167.00	331856 (20.60%)
test6	Single EVE	12550800.0	968.76	811444
	Iterative EVE	12767810.0	1027.0	271742 (33.49%)
test7	Single EVE	9480330.0	369.28	1200267
	Iterative EVE	7593774.2	366.07	454308 (37.85%)
test8	Single EVE	35511100.2	1918.0	1168543
	Iterative EVE	35530547.4	1928.0	414317 (35.45%)
test9	Single EVE	11674800.3	984.47	1103223
	Iterative EVE	11157401.0	1013.3	331287 (30.03%)
test10	Single EVE	66256.2	66.04	326920
	Iterative EVE	64255.5	65.47	93924 (28.73%)

Table 2: The QOR comparison for area-efficient gate sizing.

	δ	WSLK Imp	FOM Imp	Area Inc
test1	0	0.08041	2373.05	200062
	0.005	0.00993	1198.84	62733
	0.010	0.00791	629.81	7217
	0.030	0.00146	268.06	−10162
	0.050	0.00146	252.72	−10370
test2	0	0.03775	1167.43	68958
	0.005	0.00902	646.73	18254
	0.010	0.00813	307.5	1940
	0.030	0.00813	138.93	−1785
	0.050	0.00813	123.46	−1862
test3	0	0.02486	164.51	31861
	0.005	0.00209	91.42	6859
	0.010	0.00209	45.89	325
	0.030	0.00209	19.15	−2575
	0.050	0.00209	19.09	−2593
test4	0	0.06008	309.02	13329
	0.005	0.02325	241.25	3862
	0.010	0.00021	83.67	−120
	0.030	0.00021	39.39	−862
	0.050	0.00021	39.05	−870

area saving is 12.5%. Considering the area is the first-order estimation of the gate power, the number is significant. In addition to total area, we reduce the logic area growth by 12%. Logic area is defined as the amount of area which a physical synthesis tool can "optimize", which excludes fixed and nonsizable cells, such as memory, SRAM, and fixed macroblockages. The logic area growth is (the increased logic area/the initial logic area) × 100%.

As we mentioned before, area bloat also causes the routing and timing problems. The design shown in Figures 1 and 2 is actually the "test1" design in Table 3. As discussed before, the routability of this design is very sensitive to

Table 3: The QOR comparison of baseline and new flow.

Circuit	Type	Area	CPU	WSLK	FOM	WL
test1	base	1.30777	2812.7	−12.738	−115965	46175031
#gates: 102046	new	1.15874	4483.7	0.099	0	18100494
test2	base	20.2130	45229.6	−0.170	−2605	381929854
#gates: 717075	new	19.5043	52505.1	−0.288	−5669	383202138
test3	base	1.92936	5948.8	−0.026	−2	34700143
#gates: 110118	new	1.8862	6326.5	0.100	0	34175838
test4	base	10.9556	17771.0	−0.436	−928	121271648
#gates: 253815	new	10.7207	22537.3	−0.436	−203	116457562
test5	base	5.99029	28559.2	0.099	0	150358750
#gates: 663693	new	5.51185	31977.5	0.096	0	141428293
test6	base	8.29468	13441.5	−0.747	−37	54379592
#gates: 94220	new	8.15084	17410.4	−0.754	−36	50603208
test7	base	16.3154	18454.7	−0.089	−1	273579209
#gates: 367478	new	15.8426	22535.1	0.039	0	257814794
test8	base	7.46228	18598.5	0.012	0	136021272
#gates: 476495	new	7.12724	22208.1	0.013	0	142562718
test9	base	4.81608	14899.8	−0.512	−357	101211621
#gates: 168379	new	4.53251	17373.3	−0.375	−118	89450246
test10	base	5.54470	27398.3	−1.805	−27248	165713654
#gates: 347502	new	4.85059	24215.5	−0.032	−53	90467156
test11	base	9.97560	30120.5	−0.528	−696	151761936
#gates: 517459	new	9.27245	37599.5	−0.525	−726	144139156
test12	base	9.92720	28309.0	−0.347	−117	128984471
#gates: 517583	new	9.27470	36257.1	−0.346	−156	124913294
test13	base	6.04183	18874.4	0.087	0	157218211
#gates: 554423	new	5.82644	20980.1	0.098	0	154116834
test14	base	3.11515	6193.2	0.100	0	33563568
#gates: 142542	new	3.01044	6713.1	0.100	0	28902330
test15	base	9.95405	31284.2	−0.094	−1	269291914
#gates: 797963	new	9.24077	39238.8	−0.057	−22	255280710
test16	base	13.7727	57974.5	−0.743	−4498	404978253
#gates: 1066512	new	12.1618	71217.3	−0.646	−4048	397577302
test17	base	14.4742	30467.7	−0.253	−55	160451598
#gates: 424465	new	13.7434	37764.0	−0.486	−63	142016166
test18	base	5.15163	15858.4	−0.412	−30	158948356
#gates: 416142	new	4.79116	17207.0	−0.403	−29	137246466
test19	base	4.66864	9726.1	−0.200	−58	61616468
#gates: 246524	new	4.59064	11020.5	−0.200	−58	62023304
test20	base	7.00458	27110.2	−0.688	−332	139740251
#gates: 494645	new	6.41238	33254.7	−0.100	−54	130307063

the area. With our new flow, "test1" is not only routable, but also the timing is near to close (the slack threshold is 0.1 ns). Compared to the baseline flow, the wirelength is reduced by 60.8%, and the worst slack is improved by over 12 ns. "test10" shows the similar trend with 45.4% wirelength reduction and 1.8 ns worst slack improvement. On average, our new flow achieves 10.1% wirelength reduction. The worst slack is improved by 770 ps and FOM is improved by 42.9% on average. Out of 20 designs, our new flow gives better slack for 12 designs compared to the baseline flow (same for 2 designs), and gives better FOM for 10 designs (same for 5 designs).

The main reason for the timing and the wirelength improvement is (1) more free space to insert buffers at the desired location or size gate up without moving, (2) better timing before timing-driven placement will make placement move less objects and results in less wirelength increase, and (3) with more freespace, legalization tends to have minimal impact on the wirelength and timing.

8. Conclusion

As we see, timing, congestion, area and power problems are coupled together. By significantly reducing the area bloat, our techniques improve the timing, wirelength, congestion and power consumption as well. Compared to a traditional timing-driven flow, our work achieves 12% logic area growth reduction, 5.8% total area reduction, 10.1% wirelength reduction and 770 ps worst slack improvement on average on 20 industrial designs in 65 nm and 45 nm.

References

[1] L. Trevillyan, D. Kung, R. Puri, L. N. Reddy, and M. A. Kazda, "An integrated environment for technology closure of deep-submicron IC designs," *IEEE Design and Test of Computers*, vol. 21, no. 1, pp. 14–22, 2004.

[2] L. P. P. P. van Ginneken, "Buffer placement in distributed RC-tree networks for minimal Elmore delay," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '90)*, pp. 865–868, 1990.

[3] W. Shi and Z. Li, "A fast algorithm for optimal buffer insertion," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 6, pp. 879–891, 2005.

[4] J. Lillis, C. K. Cheng, and T. T. Y. Lin, "Optimal wire sizing and buffer insertion for low power and a generalized delay model," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 3, pp. 437–446, 1996.

[5] Z. Li, C. N. Sze, C. J. Alpert, J. Hu, and W. Shi, "Making fast buffer insertion even faster via approximation techniques," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '05)*, pp. 13–18, 2005.

[6] S. Hu, C. J. Alpert, J. Hu et al., "Fast algorithms for slew-constrained minimum cost buffering," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 11, pp. 2009–2022, 2007.

[7] X. Tang, R. Tian, H. Xiang, and D. F. Wong, "A new algorithm for routing tree construction with buffer insertion and wire sizing under obstacle constraints," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD '01)*, pp. 49–56, 2001.

[8] P. Cocchini, "Concurrent flip-flop and repeater insertion for high performance integrated circuits," in *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD '02)*, pp. 268–273, 2002.

[9] C. J. Alpert, A. Devgan, and S. T. Quay, "Buffer insertion with accurate gate and interconnect delay computation," in *Proceedings of the 36th Annual Design Automation Conference (DAC '99)*, pp. 479–484, 1999.

[10] C. J. Alpert, A. Devgan, and S. T. Quay, "Buffer insertion for noise and delay optimization," in *Proceedings of the 35th Design Automation Conference*, pp. 362–367, 1998.

[11] Z. Li, C. J. Alpert, S. Hu, T. Muhmud, S. T. Quay, and P. G. Villarrubia, "Fast interconnect synthesis with layer assignment," in *Proceedings of the ACM International Symposium on Physical Design (ISPD '08)*, pp. 71–77, 2008.

[12] C. Chu and D. F. Wong, "Closed form solution to simultaneous buffer insertion/sizing and wire sizing," *Proceedings of the ACM Transactions on Design Automation of Electronic Systems (TODAES '01)*, vol. 6, no. 3, pp. 343–371, 2001.

[13] C. P. Chen, C. C. N. Chu, and D. F. Wong, "Fast and exact simultaneous gate and wire sizing by Lagrangian relaxation," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '98)*, pp. 617–624, 1998.

[14] S. Hu, M. Ketkar, and J. Hu, "Gate sizing for cell library-based designs," in *Proceedings of the 44th ACM/IEEE Design Automation Conference (DAC '07)*, pp. 847–852, June 2007.

[15] Y. Liu and J. Hu, "A new algorithm for simultaneous gate sizing and threshold voltage assignment," in *Proceedings of the International Symposium on Physical Design (ISPD '09)*, pp. 27–34, 2009.

[16] C. J. Alpert, S. K. Karandikar, Z. Li et al., "Techniques for fast physical synthesis," *Proceedings of the IEEE*, vol. 95, no. 3, pp. 573–599, 2007.

[17] S. K. Karandikar, C. J. Alpert, M. C. Yildiz, P. Villarrubia, S. Quay, and T. Mahmud, "Fast electrical correction using resizing and buffering," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '07)*, pp. 553–558, 2007.

[18] Z. Li, C. J. Alpert, S. Hu, T. Muhmud, S. T. Quay, and P. G. Villarrubia, "Fast interconnect synthesis with layer assignment," in *Proceedings of the ACM International Symposium on Physical Design (ISPD '08)*, pp. 71–77, April 2008.

Hindawi Publishing Corporation
VLSI Design
Volume 2011, Article ID 948926, 7 pages
doi:10.1155/2011/948926

Review Article

Suitability of Various Low-Power Testing Techniques for IP Core-Based SoC: A Survey

Usha Mehta,[1] Kankar Dasgupta,[2] and Niranjan Devashrayee[1]

[1] *PG-VLSI Design Group, EC Department, Institute of Technology, Nirma University, Ahmedabad 382 481, India*
[2] *Space Application Centre, Indian Space Research Organization, Ahmedabad 380 015, India*

Correspondence should be addressed to Usha Mehta, usha.mehta@nirmauni.ac.in

Received 3 September 2010; Accepted 23 December 2010

Academic Editor: Yangdong Deng

Test power is the major issue for current generation VLSI testing. It has become the biggest concern for today's SoC. While reducing the design efforts, the modular design approach in SoC (i.e., use of IP cores in SoC) has further exaggerated the test power issue. It is not easy to select an effective low-power testing strategy from a large pool of diverse available techniques. To find the proper solutions for test power reduction strategy for IP core-based SoC, in this paper, starting from the terminology and models for power consumption during test, the state of the art in low-power testing is presented. The paper contains the detailed survey on various power reduction techniques proposed for all aspects of testing like external testing, Built-In Self-Test techniques, and the advances in DFT techniques emphasizing low power. Further, all the available low-power testing techniques are strongly analyzed for their suitability to IP core-based SoC.

1. Introduction

The power consumption has been a major challenge to both design and test engineers. The efforts to reduce the power consumption during normal function mode further exaggerated the power consumption problem during test. Generally, a circuit may consume 3–8 times power in the test mode than in the normal mode [1]. As a result, the semiconductor industry is looking for low-power testing techniques [2].

To reduce the cost and time to market, the modular design approach is largely adopted for SoC. The structure of such predesigned, ready-to-use intellectual property (IP) core is often hidden from the system integrator. So testing of such cores is even more daunting. So power reduction during testing of such cores puts many constraints on current low-power testing methodology. To develop the right testing strategy for such SoC, it is necessary to survey all the available low-power testing approaches and find out the suitable approach for such SoC.

The paper is organized as follows. Section 2 gives the reasons for very high-power consumption during test and its effects of such high-power consumption on IC. It also includes definitions of various terms related to test power and also explains the model for energy and power. Section 3 contains the various schemes for low-power testing. Section 4 discusses the suitability of each scheme with reference to IP core-based SoC. Section 5 concludes the survey and explores the future scope.

2. Low-Power Test

A high density system like ASIC or SoC always demands the nondestructive test which satisfies all the power constraints defined during design phase. On the other way, the current testing philosophy demands much more power consumption during test compared to power consumption during functional mode. This section describes the reasons and effects of such high-power consumption.

2.1. Reasons of High-Power Consumption during Test. There are several reasons for this increased test power. Out of them, the main reasons are as follows.

(i) The test efficiency has been shown to have a high correlation with the toggle rate; hence, in the test mode, the switching activity of all nodes is often several times higher than the activity during normal operations.

(ii) In an SoC, parallel testing is frequently employed to reduce the test application time, which may result in excessive energy and power dissipation.

(iii) The design-for-testability circuitry embedded in a circuit to reduce the test complexity is often idle during normal operations but may be intensively used in the test mode.

(iv) That successive functional input vectors applied to a given circuit during system mode have a significant correlation, while the correlation between consecutive test patterns can be very low. This can cause significantly larger switching activity and hence power dissipation in the circuit during test than that during its normal operation [3].

2.2. Effects of High-Power Dissipations. The most adverse effect of very high-power dissipation during test is the destruction of IC itself. In addition, to prevent the IC from destruction, the power dissipation during test can affect the cost, reliability, autonomy, performance-verification, and yield-related issues [4]. Some of the effects are as follows.

(i) The growing need of at-speed testing can be constrained because of the high-power dissipation. So stuck at faults can be tested without any effect, but the testing of the delay fault will become difficult.

(ii) During functional testing of the die just after wafer etching, the unpackaged bare die has very little provision for power or heat dissipation. This might be a problem for applications based on multichip module technology, for example, in which designers cannot realize the potential advantages in circuit density and performance without access to fully tested bare dies [5].

(iii) Circuit can be failed because of erosion of conductors caused by electromigration.

(iv) The online BIST in battery-operated remotes and portable systems consumes very high power for testing only. Remote system operation occurs mostly in standby mode with almost no power consumption, interrupted by periodic self-tests. Hence, power savings during test mode directly prolong battery lifetime.

(v) The elevated temperature and excessive current density severely decrease circuit reliability.

(vi) Because of excessive power dissipation, the simple low-cost plastic packages used in consumer electronics cannot be used, but expensive packages which can remove the excessive heat are forced to be used.

2.3. Definitions and Models of Energy and Power. Power consumption in CMOS circuits can be classified into static and dynamic. Static power dissipation is due to leakage current or other current drawn continuously from the power supply. Dynamic dissipation is due to (i) short circuit current and (ii) charging and discharging of load capacitance during output switching. For the current CMOS technology, dynamic power is the dominant source of power consumption.

A good approximation of the energy consumed during one clock period is

$$Ei = \frac{1}{2} * S_i * F_i * C_o * V_{DD}^2, \tag{1}$$

where S_i is the number of switching during the period, F_i is the fanout of the node, and C_o is the minimum size parasitic capacitance of the circuit. The fanout of the nodes is defined by circuit topology, and the switching can be estimated by a logic simulator (note that in a CMOS circuit, the number of switching is calculated from the moment the input vector is changed until the moment the internal nodes reach the new stable state, including the hazard switching). The product S_iF_i is named Weighted Switching Activity (WSA) of node i and represents the only variable part in the energy consumed at node i during test application. So the energy consumed in the circuit after application of a pair of successive input vectors (V_{k-1}, V_k) can be expressed by

$$E_{V_K} = \frac{1}{2} * F_i * C_o * V_{DD}^2 * \sum_i S(i,k), \tag{2}$$

where i ranges all the nodes of the circuits and $S(i,k)$ number of switching provoked, by V_k at node i. Consider now a pseudorandom test sequence of L vectors which is the test length. The total energy consumed in the circuit is

$$E_{\text{total}} = \frac{1}{2} * F_i * C_o * V_{DD}^2 * \sum_L \sum_i S(i,k). \tag{3}$$

It should be noted that energy is the total switching activity generated during test application and has impact on the battery lifetime during powerup or periodic self-test of battery-operated devices. Therefore, we can express the instantaneous power consumed in the circuit after application of vectors (V_{k-1}, V_k) as

$$P_{\text{inst}}(V_k) = \frac{E_{V_k}}{T}. \tag{4}$$

The peak power consumption corresponds to the maximum of the instantaneous power consumed during the test session. It therefore corresponds to the highest energy consumed during one clock period, divided by T. More formally, it can be expressed by

$$P_{\text{peak}} = \max_k[P_{\text{inst}}(V_k)] = \frac{\max_k(E_{V_k})}{T}. \tag{5}$$

Finally, the average power consumed during the test session is the total energy divided by the test time and is given as follows:

$$P_{\text{avg}} = \frac{E_{\text{total}}}{L \cdot T}. \tag{6}$$

Elevated average power adds to the thermal load that must be vented away from the device under test. It may cause

structural damage to the silicon (hot spots), to bonding wires, or to the package.

According to the above expressions of the power and energy consumption and assuming a given CMOS technology and supply voltage for the circuit design, the number of switching of a node i in the circuit is the only parameter that has impact on the energy, the peak power, and the average power consumption. Similarly, the clock frequency used during testing has impact on the peak power, and the average power. Finally, the test length, the number of test patterns applied to the CUT, has impact only on the total energy consumption. Consequently, when deriving a solution for power and/or energy minimization during test, a designer or a test engineer has to have these relationships in mind.

From the viewpoint of scan test, test power can be divided into shift power and capture power, corresponding to shift mode and capture mode, respectively. In shift mode, many clock pulses are applied to load a test vector and unload a test response. Therefore, average shift power dominates heat dissipation during scan shift. Excessive peak shift power may cause scan chain failures, resulting in yield loss. In capture mode, where only one or two clock pulses are needed, the contribution towards test heat is negligible.

3. Low-Power Testing Schemes

During the last two decades, the number of power reduction techniques for testing have evolved. These techniques either explore the ATPG and deal with the test vectors to be used with external testing or explore the internal structure of design using BIST or DFT. So existing low-power testing scheme is divided into the following two categories.

(1) Low-Power Testing Techniques for External Testing using ATE, ATPG, and so forth.

(2) Low-Power Testing Techniques for Internal Testing using BIST, DFT, and so forth.

3.1. Low-Power Testing Techniques for External Testing. The following are the classifications of low-power techniques for external testing.

3.1.1. Low-Power ATPG Algorithms. This category contains various techniques adopted to reduce the power consumption during external testing by ATE. These methods depend on the number of transitions in test data set. The current research in this field focuses on ATPG algorithm which not only gives maximum fault coverage but also ensures the maximum fault coverage at lowest possible power dissipation. Reference [6] proposed a heuristic method to generate test sequences which create worst-case power droop by accumulating the high- and low-frequency effects using a dynamically constrained version of the classical D-algorithm for test generation. A novel scan chain division algorithm [7] analyzes the signal dependencies and creates the circuit partitions such that both shift and capture power can be reduced when using the existing ATPG flows. Reference [8] presents a low capture power ATPG and a power-aware test compaction method. This ATPG lowers the growth of test pattern count compared to the detection number n. The peak power becomes smaller as the detection number n increases. The test compaction algorithm further reduces the number of test patterns as well as the average capture power.

3.1.2. Input Control. Here the idea is to identify an input control pattern such that, by applying that pattern to the primary inputs of the circuit during the scan operation, the switching activity in the combinational part can be minimized or even eliminated. The basic idea of input control technique with existing vector- or latch-ordering techniques that reduces the power consumption has been covered in [9]. In the same area, [10] presented a technique of gating partial set of scan cells. The subset of scan cells is selected to give maximum reduction in test power within a given area constraint. An alternate formulation of the problem is to treat maximum permitted test power and area overhead as constraints and achieve a test power that is within these limits using the fewest number of gated scan cells, thereby leading to least impact in area overhead. The area overhead is predictable and closely corresponds to the average power reduction.

3.1.3. Ordering Techniques. The researches have widely explored the test vector reordering techniques to reduce the switching power. Hamming distance based reordering is described in survey paper [11]. Girard's approach of vector ordering is enhanced in [12]. In [13], another method based on artificial intelligence is proposed to order the test vectors in an optimal manner to minimize switching activity during testing.

3.1.4. Exploring the Do Not Care Bit. ATPG-generated uncompacted test data contains a large number of do not care bits. [14] proposed an automatic test pattern generation (ATPG) scheme for low-power launch-off-capture (LOC) transition test. The authors in [15] have used a Genetic Algorithm-based heuristic to fill the do not cares. This approach produces an average percentage improvement in dynamic power and leakage power over 0-fill, 1-fill, and Minimum transition fill (MT-fill) algorithms for do not care filling. [16] proposed segment-based X-filling to reduce test power and keep the defect coverage. The scan chain configuration tries to cluster the scan flip-flops with common successors into one scan chain, in order to distribute the specified bits per pattern over a minimum number of chains. Based on the operation of a state machine, [17] elucidates a comprehensive frame for probability-based primary-input-dominated X-filling methods to minimize the total weighted switching activity (WSA) during the scan capture operation. The authors in [18] describe the effect of do not care filling of the patterns generated via automated test pattern generators, to make the patterns consume lesser power. It presents a tradeoff in the dynamic and static power consumption.

3.2. Low-Power Testing Techniques for Internal Testing. The following are the classifications of low-power techniques for internal testing.

3.2.1. By LFSR Architecture. The Built-In Self-Test (BIST) architecture contains two major components: test pattern generator and response checker [19]. Both of these components use Linear Feedback Shift Register (LFSR). The LFSR can be designed to reduce the power consumption during test in the following ways.

3.2.2. By Reducing the Transitions. These methods reduce the transitions between successive patterns generated by LFSR as well as between the successive bits in a given pattern. A dual-speed LFSR scheme [20] is based on two different speed LFSRs to decrease the circuit's overall internal activity. Its objective is to decrease the circuit's overall internal activity by connecting inputs that have elevated transition densities to the slow-speed LFSR. This strategy significantly reduces average power and energy consumption without decreasing fault coverage. Cellular automata-based test pattern generation to reduce power is described in [21]. In [22], the LFSR is modified by adding weight sets to tune the pseudorandom vectors signal probabilities and thereby decrease energy consumption and increase fault coverage. The LP-TPG [23] inserts intermediate patterns between the random patterns to reduce the transitional activities of primary inputs which eventually reduces the switching activities inside the circuit under test, and hence, power consumption. A polynomial-time algorithm that converts the test pattern generation problem into combinatorial problem called Minimum Set Covering Solutions is proposed in [24]. A new low-power BIST TPG scheme [25] uses a transition monitoring window (TMW) that is comprised of a TMW block and an MUX. The proposed technique represses transitions of patterns using the k-value which is a standard that is obtained from the distribution of TMW to observe over transitive patterns causing high-power dissipation in a scan chain. In [26], a TPG based on Read-Only Memory (ROM) is carefully designed to store the test vectors with minimum area over the conventional ROM. This reduces the number of CMOS transistors significantly when compared to that of LFSR/Counter TPG. An approach to reconfigure the CUT's partial-acting-inputs into a short ring counter (RC) and keep the CUT's partial-freezing-inputs unchanged during testing is proposed in [27]. A low hardware overhead test pattern generator (TPG) for scan-based Built-In Self-Test (BIST) that can reduce switching activity in circuits under tests (CUTs) during BIST is presented in [28]. It also achieves very high fault coverage with reasonable lengths of test sequences. The proposed BIST TPG decreases transitions that occur at scan inputs during scan shift operations and hence reduces switching activity in the CUT. In LT-LFSR [29], transitions in LFSR are reduced in two dimensions: (1) between consecutive patterns and (2) between consecutive bits. The proposed architecture increases the correlation among the patterns generated by LT-LFSR with negligible impact on test length. An efficient algorithm to synthesize a built-in TPG from low-power deterministic test patterns without inserting any redundancy test vectors is presented in [30]. The structure of TPG is based on the nonuniform cellular automata (CA). And the algorithm is based on the nearest neighborhood model, which can find an optimal nonuniform CA topology to generate given low-power test patterns. A low-power dynamic LFSR (LDLFSR) circuit [31] achieves comparable performance with less power consumption. Typical LFSR, a DFLSR[I], and a LDLFSR are compared on randomness property and inviolability property. Multilayer perceptron neural networks are used to test this LFSRs' inviolability property.

3.2.3. By Generating the Useful Vectors Only. A significant amount of energy is wasted in the LFSR and in the CUT by useless patterns that do not contribute to fault dropping. LFSR tuning modifies the state transitions of the LFSR such that only the useful vectors are generated according to a desired sequence [32]. To reduce such energy consumption, a mapping logic is designed in [33] which modifies the state transitions of the LFSR such that only the useful vectors are generated according to a desired sequence.

3.2.4. By Filtering Unnecessary Vectors. There are some nondetecting sequences generated by LFSR. By inhibiting such vectors during testing, over all switching can be reduced. A test-vector-inhibiting technique to filter out some nondetecting subsequences of a pseudorandom test set generated by an LFSR is proposed in [34]. The authors use a decoding logic to store the first and last vectors of the nondetecting subsequences. This work was extended in [35] by the filtering action to all the nondetecting subsequences. A pattern-filtering technique is combined with Hertwig and Wunderlich's technique to avoid scan-path activity during scan shifting in [36]. Hatami et al. [37] proposed a scan cell architecture that decreases power consumption and the total consumed energy. In the method which is based on the data compression, the test vector set is divided into two repeated and unrepeated partitions. The repeated part, which is common among some of the vectors, is not changed during the new scan path, where new test vector will be filled. As a result, the test vector is applied to the circuit under test in a fewer number of clock cycles, leading to a lower switching activity in the scan path during test mode.

3.2.5. By Partitioning Circuit. The circuit is strategically partitioned into subcircuits to achieve the parallel testing. An efficient scan partitioning technique reduces average and peak power in the scan chain during shift and functional cycles. A low-power BIST strategy based on circuit partitioning is described in [38]. This strategy partitions the original circuit into two structural subcircuits so that two different BIST sessions can successively test each subcircuit. To address the power in the scan chain, in [39], an efficient scan partitioning technique that reduces both average and peak power in the scan chain during shift and functional cycles is proposed. In [40], the authors proposed a novel low-power virtual test partitioning technique, where faults in the

glue logic between subcircuits can be detected by patterns with low-power dissipation that are applied at the entire circuit level, while the patterns with high-power dissipation can be applied within a partitioned subcircuit without loss of fault coverage. Experimental results show that the proposed technique is very effective in reducing test power.

3.2.6. By Separate Testing Strategy for Memory. Various transition reduction techniques for memory testing by reordering read and write access are available in the literature. A row bank-based precharge technique based on the divided wordline (DWL) architecture is proposed in [41]. In low-power test mode, instead of precharging the entire memory array, only the current accessed row bank is precharged. This will result in significant power saving for the precharge circuitry. With the ever-increasing number of memories embedded in a system on chip (SoC), power dissipation due to memory test has become a serious concern. In [42], the authors proposed a novel low-power memory BIST. Its effectiveness is evaluated on memories in 130 and 90 nm technologies. A significant power reduction can be achieved with virtually zero hardware overhead.

3.2.7. Low-Power Design-for-Test Techniques. In this category, some extra hardware is added to design for reducing the power consumption during test. Clock partitioning and clock freezing [43] and use of J-scan instead of traditional MUX scan [44] are the examples of such methods. This approach adds to or modifies the on-chip design hardware to reduce the power consumption during test and hence may be called Design for Low-Power Test (DFLPT).

4. Low-Power Testing Techniques Emphasizing IP Core-Based SoC

With the emergence of core-based SoC design, BIST already coming as a part of IP core presents one of the most favorable testing methods because it allows preservation of a design's intellectual property [45]. Such BISTs are most suitable to test the IP core in standalone mode, but, when the IP core is integrated with other blocks to form a complete system, they might not be suitable.

Now let us think about adding some low-power schemes at the time of system integration. The structure of IP cores are often hidden from system integrator. So neither any modification to its internal scan chain nor any DFT insertion is possible for IP cores. Further, any testing tools like Automatic Test Pattern Generator (ATPG) or fault simulation cannot be applied to it. Such cores are coming with ready to use test data. This test data is used to test the core when it is in isolation as well as when it is as a part of system after being integrated to system. It is usually assumed that the core is directly accessible, and it becomes the task of the system integrator to ensure that the logic surrounding the core allows the test stimuli to be applied and the produced responses to be transported for evaluation. So the only option remains to reduce power is schemes applicable to readymade test data.

Based on the above discussion, first of all, let us list the characteristics of the power reduction technique suitable to IP core-based Soc and then compare each of the available techniques with our ideal model.

4.1. Characteristics of Power Reduction Scheme Suitable to IP Core-Based SoC.

(i) It should not demand the knowledge of internal structure of design.

(ii) It should not make any modification in internal design.

(iii) But it can add the hardware as per requirement without modifying the available I/O pin configuration.

(iv) It should deal with readymade test sequence rather than test architecture.

(v) It should not be dependant on testing tools like ATPG or fault simulation which deals with the netlist of design.

Now comparing the available techniques with above characteristics.

4.2. Modification in LFSR. The implementation of this method

(i) deals with test sequence rather than test architecture,

(ii) requires the knowledge of internal details of design,

(iii) requires the additional hardware to modify test pattern sequence, and

(iv) requires modification in internal structure.

4.3. Partitioning the Circuit. The implementation of this method

(i) deals with test architecture rather than test sequence,

(ii) requires the well-defined internal hierarchical structure of design,

(iii) requires the knowledge of internal details of design,

(iv) requires the additional hardware to modify test pattern sequence, and

(v) requires modification in internal structure.

4.4. Separate Testing Strategy for Memory. The implementation of this method

(i) can be applied for memory when and where it is required, and

(ii) is not applicable to functional blocks.

4.5. Improved ATPG Algorithms. The implementation of this method

(i) deals with generation of new test set rather than available test sequence or test architecture,

(ii) requires the netlist of the design. It cannot be directly applicable to hard core, and

(iii) requires the knowledge of internal details of design.

4.6. Input Control. The implementation of this method

(i) deals with test architecture rather than test sequence,

(ii) requires the knowledge of internal details of design,

(iii) requires the additional hardware to modify test pattern sequence, and

(iv) requires modification in internal structure.

4.7. Ordering Technique. The implementation of this method

(i) deals with test sequence rather than test architecture,

(ii) requires the well-defined test sequence, that is, test data set,

(iii) does not require the knowledge of internal details of design,

(iv) requires the additional hardware to reorder test pattern sequence, and

(v) does not require modification in internal structure.

4.8. Exploring Do Not Care Bits. The implementation of this method

(i) deals with test data bit sequence rather than test vector sequence or test architecture,

(ii) requires the well-defined test sequence, that is, test data set,

(iii) does not require the knowledge of internal details of design.

(iv) does not require any additional hardware,

(v) does not require modification in internal structure.

Out of the above-mentioned categories, except "ordering techniques" and "exploring do not care bits" methods, all other methods require the internal details of the design under test. Hence, in context of IP core-based SoC, only these two categories are suitable.

5. Conclusion

This survey paper on low-power testing techniques suitable to IP core-based SoC starts with the reasons and effects of high-power consumption during test, including energy and power model. Very advanced techniques available for power reduction during test are described in detail. The issues related to test power reduction in case of IP core-based SoC are discussed, and characteristics of ideal scheme suitable to IP core-based SoC is defined. Based on that, each available category of power reduction is compared with this ideal model. It is concluded that "ordering techniques" and "exploring do not care bits" methods are the best suited to IP core-based SoC. The research can start with improvement in these schemes in terms of power reduction and then further optimizing them with other important test parameters like test application time, on-chip area overhead, test data compression, and so forth.

References

[1] C. P. Ravikumar, M. Hirech, and X. Wen, "Test strategies for low power devices," in *Proceedings of the Design, Automation and Test in Europe (DATE '08)*, pp. 728–733, March 2008.

[2] Y. Bonhomme et al., "Test power: a big issue in large SoC designs," in *Proceedings of The 1st IEEE Workshop on Electronic Design, Test And Applications*, pp. 447–449, 2002.

[3] S. Wang and S. K. Gupta, "DS-LFSR: a new BIST TPG for low heat dissipation," in *Proceedings of the 1997 IEEE International Test Conference (ITC '97)*, pp. 848–857, November 1997.

[4] M. B. M. Abramovici and A. Friedman, *Digital Systems Testing and Testable Design*, IEEE Press, Piscataway, NJ, USA, 1990.

[5] P. Parkar, "Bare die test," in *Proceedings of IEEE Multi-Chip Module Conference*, pp. 24–27, 1992.

[6] I. Polian, A. Czutro, S. Kundu, and B. Becker, "Power droop testing," in *Proceedings of the 24th International Conference on Computer Design (ICCD '06)*, pp. 243–250, October 2006.

[7] H. Fai and N. Nicolici, "Automated scan chain division for reducing shift and capture power during broadside at-speed test," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 11, pp. 2092–2097, 2008.

[8] S. J. Wang, K. L. Fu, and K. S. M. Li, "Low peak power ATPG for n-detection test," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '09)*, pp. 1993–1996, May 2009.

[9] P. Girard, "Survey of low-power testing of VLSI circuits," *IEEE Design and Test of Computers*, vol. 19, no. 3, pp. 80–90, 2002.

[10] M. ElShoukry, C. P. Ravikumar, and M. Tehranipoor, "Partial gating optimization for power reduction during test application," in *Proceedings of the 14th Asian Test Symposium (ATS '05)*, pp. 242–245, ind, December 2005.

[11] P. Girard, C. Landrault, S. Pravossoudovitch, and D. Severac, "Reducing power consumption during test application by test vector ordering," in *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems (ISCAS '98)*, pp. 296–299, June 1998.

[12] K. Paramasivam, K. Gunavathi, and P. Sathishkumar, "Algorithm for low power combinational circuit testing," in *Proceedings of IEEE Region 10 Conference (TENCON '04)*, pp. D336–D339, November 2004.

[13] S. Roy, I. S. Gupta, and A. Pal, "Artificial intelligence approach to test vector reordering for dynamic power reduction during VLSI testing," in *Proceedings of the IEEE Region 10 Conference (TENCON '08)*, pp. 1–6, November 2008.

[14] S. J. Wang, Y. T. Chen, and K. S. M. Li, "Low capture power test generation for launch-off-capture transition test based on don't-care filling," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '07)*, pp. 3683–3686, May 2007.

[15] S. Kundu and S. Chattopadhyay, "Efficient dont care filling for power reduction during testing," in *Proceedings of IEEE International Conference on Advances in Recent Technologies In Communication and Computing*, pp. 319–323, 2009.

[16] Z. Chen, "Scan chain configuration based X filling for lowpower and high quality testing," *IET Journal on Computers and Digital Techniques*, vol. 4, pp. 1–13, 2009.

[17] J.-L. Yang and Q. Xu, "State-sensitive X-filling scheme for scan capture power reduction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 7, pp. 1338–1343, 2008.

[18] T. KR. Maiti and S. Chattopadhyay, "Don't care filling for power minimization in VLSI circuit testing," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '08)*, pp. 2637–2640, usa, May 2008.

[19] M. Abramovici et al., *Digital Systems Testing and Testable Design*, Jacoba, 1997.

[20] K. Gunavathi, K. Paramasivam, P. Subashini Lavanya, and M. Umamageswaran, "A novel BIST TPG for testing of VLSI circuits," in *Proceedings of the 1st International Conference on Industrial and Information Systems (ICIIS '06)*, pp. 109–114, August 2006.

[21] F. Corno, M. Rebaudengo, M. S. Reorda, G. Squillero, and M. Violante, "Low power BIST via non-linear hybrid cellular automata," in *Proceedings of the 18th IEEE VLSI Test Symposium (VTS '00)*, pp. 29–34, May 2000.

[22] X. Zhang, K. Roy, and S. Bhawmik, "POWERTEST: a tool for energy conscious weighted random pattern testing," in *Proceedings of the 12th International Conference on VLSI Design*, pp. 416–422, January 1999.

[23] N. Ahmed, M. H. Tehranipour, and M. Nourani, "Low power pattern generation for BIST architecture," in *Proceedings of the IEEE International Symposium on Cirquits and Systems*, pp. II689–II692, May 2004.

[24] H. Kiliç and L. Öktem, "Low-power test pattern generator design for BIST via non-uniform cellular automata," in *Proceedings of the IEEE VLSI-TSA International Symposium on VLSI Design, Automation and Test,(VLSI-TSA-DAT '05)*, vol. 2005, pp. 212–215, 2005.

[25] Y. Kim, M. H. Yang, Y. Lee, and S. Kang, "A new low power test pattern generator using a transition monitoring window based on BIST architecture," in *Proceedings of the 14th Asian Test Symposium (ATS '05)*, pp. 230–235, December 2005.

[26] K. Gunavathi, K. Paramasivam, P. Subashini Lavanya, and M. Umamageswaran, "A novel BIST TPG for testing of VLSI circuits," in *Proceedings of the 1st International Conference on Industrial and Information Systems (ICIIS '06)*, pp. 109–114, 2006.

[27] B. Zhou, Y.-Z. Ye, Z.-L. Li, X.-C. Wu, and R. Ke, "A new low power test pattern generator using a variable-length ring counter," in *Proceedings of the 10th International Symposium on Quality Electronic Design (ISQED '09)*, pp. 248–252, 2009.

[28] S. Wang, "A BIST TPG for low power dissipation and high fault coverage," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 7, pp. 777–789, 2007.

[29] M. Nourani, M. Tehranipoor, and N. Ahmed, "Low-transition test pattern generation for BIST-based applications," *IEEE Transactions on Computers*, vol. 57, no. 3, pp. 303–315, 2008.

[30] C. Bei, L. Xiao, and Y. Wang, "A low power deterministic test pattern generator for BIST based on cellular automata," in *Proceedings of the 4th IEEE International Symposium on Electronic Design, Test and Applications (DELTA '08)*, pp. 266–269, January 2008.

[31] L.-G. Hou, X.-H. Peng, and W.-C. Wu, "A low power dynamic pseudo random bit generator for test pattern generation," in *Proceedings of the International Conference on Solid-State and Integrated Circuits Technology Proceedings (ICSICT '08)*, pp. 2079–2082, 2008.

[32] P. Girard, L. Guiller, C. Landrault et al., "Low-energy BIST design: impact of the LFSR TPG parameters on the weighted switching activity," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '99)*, June 1999.

[33] B. Bhargab et al., "Low energy BIST design for scan-based logic circuits," in *Proceedings of 16th International Conference On VLSI Design(VLSID '03)*, pp. 546–551, 2003.

[34] P. Girard, L. Guiller, C. Landrault, and S. Pravossoudovitch, "Test vector inhibiting technique for low energy BIST design," in *Proceedings of the 17th IEEE VLSI Test Symposium (VTS '99)*, pp. 407–412, April 1999.

[35] S Manich et al., "Low power BIST by filtering non detecting vectors," in *Proceedings of the IEEE European Test Workshop*, pp. 317–319.

[36] Gerstendoerfer and Wunderlich, "Minimized power consumption for scan-based BIST," in *Proceedings of the IEEE International Test Conference (ITC '99)*, pp. 77–84, 1999.

[37] S. Hatami, M. Alisafaee, E. Atoofian, Z. Navabi, and A. Afzali-Kusha, "A low-power scan-path architecture," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '05)*, pp. 5278–5281, May 2005.

[38] P. Girard, L. Guiller, C. Landrault, and S. Pravossoudovitch, "Circuit partitioning for low power BIST design with minimized peak power consumption," *Proceedings of the Asian Test Symposium*, pp. 317–319, 1999.

[39] S. Bhunia, "Power reduction in test-per-scan BIST with supply gating and efficient scan partitioning," in *Proceedings of the 6th International Symposium On Quality of Electronic Design*, pp. 453–458, 2005.

[40] Q. Xu, D. Hu, and D. Xiang, "Pattern-directed circuit virtual partitioning for test power reduction," in *Proceedings of the IEEE International Test Conference (ITC '07)*, pp. 1–10, October 2007.

[41] S.-K. Lu, Y.-C. Hsiao, C.-H. Liu, and C.-L. Yang, "Low-power built-in self-test techniques for embedded SRAMs," *VLSI Design*, vol. 2007, no. 2, pp. 1–7, 2007.

[42] Y. Wu and A. Ivanov, "Low power SoC memory BIST," in *Proceedings of the 21st IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 197–205, October 2006.

[43] YP. Xiaoming and M. Abramovici, "Sequential circuit ATPG using combination alalgorithms," *IEEE Transactions On Computer Aided Design of Integrated Circuits and Systems*, vol. 24, pp. 1294–1310, 2005.

[44] G. Tavazza and E. Cervi, "Jump scan: a DFT technique for low power testing," in *Proceedings of 23rd IEEE VLSI Test Symposium*, pp. 277–282, 2005.

[45] H. J. Wunderlich, "BIST for systems-on-a-chip," *Integration: The VLSI Journal*, vol. 26, no. 1-2, pp. 55–78, 1998.

Hindawi Publishing Corporation
VLSI Design
Volume 2011, Article ID 530851, 10 pages
doi:10.1155/2011/530851

Research Article

Buffer Planning for IP Placement Using Sliced-LFF

Ou He,[1] Sheqin Dong,[1] Jinian Bian,[1] and Satoshi Goto[2]

[1] *Tsinghua National Laboratory for Information Science & Technology, Tsinghua University, Beijing 100084, China*
[2] *Information, Production and Systems (IPS), Waseda University, Kitakyushu-shi 808-0135, Japan*

Correspondence should be addressed to Ou He, ho06@mails.tsinghua.edu.cn

Received 14 November 2010; Accepted 11 December 2010

Academic Editor: Shiyan Hu

IP cores are widely used in modern SOC designs. Hierarchical design has been employed for the growing design complexity, which stimulates the need for fixed-outline floorplanning. Meanwhile, buffer insertion is usually adopted to meet the timing requirement. In this paper, buffer insertion is considered with a fixed-outline constraint using Less Flexibility First (LFF) algorithm. Compared with Simulated Annealing (SA), our work is able to distinguish geometric differences between two floorplan candidates, even if they have the same topological structure. This is helpful to get a better result for buffer planning since buffer insertion is quite sensitive to a geometric change. We also extend the previous LFF to a more robust version called Sliced-LFF to improve buffer planning. Moreover, a 2-staged LFF framework and a post-greedy procedure are introduced based on our net-classing strategy and finally achieve a significant improvement on the success rate of buffer insertion (40.7% and 37.1% in different feature sizes). Moreover, our work is much faster than SA, since it is deterministic without iterations.

1. Introduction

IP cores, which are modeled as quantities of small, hard, and mixed-size macros [1], will dominate modern SOC designs. Hierarchical floorplanning then becomes essential. Different from traditional outline-free floorplanning, the fixed-outline constraint, which enables hierarchical design [2], will play a fundamental role in modern SOC designs.

Besides, in deep submicron technology, interconnect delay determines the chip performance. Buffer insertion is helpful to reduce the delay. However in SOC designs, buffers cannot be inserted into hard IP modules, because they consume silicon recourses as well, which will cause a redesign of hard IPs. As a result, they could only be inserted in the interspace between modules. So, it is better to consider buffer insertion together with IP module placement. In this paper, we integrate buffer planning into floorplanning stage, which places IP modules and reserves silicon space for buffers with a fixed-outline constraint.

Many previous works have been addressed in floorplanning stage considering buffer insertion. Cong et al. [3] first gave the concept of "Feasible Region" (FR) to calculate all the feasible spots to insert a buffer and meet the timing requirement of the net. Sarkar et al. [4] improved the notion "FR" into the "Independent Feasible Region" (IFR). IFR means a region where a buffer can be placed as long as the other buffers are inserted successfully in their IFRs. Using the net flow method, Tang and Wong [5] proposed an optimal algorithm to allocate buffers into buffer blocks with the assumption that every net only needed one buffer. Dragan et al. [6] put buffers into an existing block, which was implemented using the multicommodity flow-based theory. Sham et al. [7] released a routability-driven floorplanning system, which was able to estimate needs and resources of buffers with the consideration of routing congestions. Rafiq et al. [8] proposed a floorplanner for bus-based microprocessors with buffer and channel insertion. Alpert et al. [9] solved buffer planning problem using tile graph and dynamic programming. They assumed that buffers could be inserted into the IP modules, which was relied on the future design methodology of IP modules. Chen et al. [10] proposed a post-buffer planning algorithm based on deadspace redistribution. Deadspace means the interspace among the placed modules. Jiang et al. [11] implemented floorplanning and buffer block planning simultaneously. In each iteration of SA, they constructed a routing tree for each net, allocated buffers, and introduced buffer blocks into the intermediate floorplan. Ma et al. [12] solved buffer planning

as an integral part of floorplanning with the consideration of routing congestions. This work was also based on Simulated Annealing. Dong et al. [13] resized the circuit modules to insert more buffers. But this method is not practical for hard IP cores in SOC design, because their dimensions are always fixed and cannot be resized. Both buffer and interlayer via planning were considered by He et al. [14] for 3D ICs.

Most of the aforementioned works adopted either of two different strategies to do buffer planning. One is to take the problem as a postprocess after floorplanning. In this case, a seed floorplan is generated at first. Then, based on the existing floorplan, we reshape deadspace for buffers with a fixed topological structure, such as [6, 10]. However, all the works are finished in a given topological structure; so the solution space is limited. The other is to consider buffer planning with floorplanning. This methodology evaluates different topological structures and finally picks the best one to improve the number of inserted buffers, such as [12]. However, this method focuses on topological structures and fails to distribute deadspace effectively. More details will be discussed in Section 4.6. Besides, most of SA-based previous works did not consider the fixed outline constraint.

Section 4.6 explains the fact that buffer insertion issue is sensitive to geometric structures. That means that we should consider not only different topological structures but also the deadspace distribution with a fixed topology. Both of them will change the geometric structure of a floorplan. Unlike the other previous works, [11] could consider these two at the same time, but it was time-consuming using an SA framework.

In this paper, another algorithm framework named Less Flexibility First (LFF) [15] is adopted to handle buffer insertion issue. It works in a fixed-outline constraint unlike the aforementioned previous works. Compared with SA, LFF can handle buffer insertion and fixed-outline constraint together and run much faster. In LFF, different packing orders correspond to different topological structures. Thus, we can implement topological optimization by choosing the order of packing. In addition, we can also reshape the deadspace for buffers by inserting some fictitious modules in IP placement.

Moreover, LFF can work together with SA framework. Results of our LFF buffer planning can also be used to provide an initial solution for SA to improve the efficiency of SA iterations. Meanwhile, Half Perimeter Wirelength (HPWL) is still used to optimize timing in this paper. This is because most previous works also adopted wirelength (or weighted wirelength) for timing optimization. Moreover, HPWL is easy and clear to compare with previous works.

Detailed contributions in this paper are listed as follows.

(i) A new version for LFF called Sliced-LFF is introduced to represent the deadspace and enhance the robustness of the packing process. This work will facilitate buffer planning as well, since more information about the deadspace could be provided by our slice list for buffers.

(ii) Buffer insertion is integrated into LFF process. After that, topological and geometric information is both handled in our algorithm for buffer optimization.

(iii) Net-classing strategy is proposed for further optimization.

(iv) A post-floorplanning method is adopted to greedily insert more buffers.

Experimental results show that a significant improvement on the success rate of buffer insertion (40.7% and 37.1%) is achieved. In addition, more interconnections (14.3% and 15.5%), which violate the delay constraints, are corrected by our buffer planning method.

The rest of this paper is organized as follows. Section 2 formulates the fixed-outline floorplanning as well as buffer insertion. In Section 3, Sliced-LFF method is introduced. Section 4 shows how to integrate buffer planning into LFF. Section 5 discusses the net-classing strategy and our post greedy method while Section 6 shows the experiment results. At last, conclusions and future work are given in Section 7.

2. Problem Formulation

The input of the algorithm is a set of N rectangular IP modules $M = \{m_1, m_2, \ldots, m_N\}$ and their areas, locations of I/O pins, netlist, and delay constraints of the nets. Moreover, the width W and the height H of the chip are also given as the fixed-outline constraint.

The output is the coordinates of the bottom-left corners of the modules and their orientations. Meanwhile, deadspace is reserved for buffers on the resultant floorplan. Assumptions in this paper are listed as follows.

(i) All the modules are hard modules. They have fixed sizes and aspect ratios.

(ii) All the buffers can only be inserted in the deadspace among the modules instead of the inside of modules.

(iii) All nets are 2-pin, and multipin nets are split to 2-pin nets, which is the same as previous works [3, 12].

(iv) All the modules can be rotated or mirrored during the packing process.

The goal of our algorithm is to find a feasible packing in the fixed outline, which is able to reserve deadspace for the required buffers as many as possible.

3. Sliced LFF Algorithim

3.1. Original LFF Algorithm. LFF is a published floorplanning algorithm [15], which works with the fixed-outline constraint. A typical LFF packing process is demonstrated in Figure 1.

In Figure 1, modules are placed from four corners of the fixed-outline gradually to the center one by one. To place one module (or called a *packing step*), the most suitable candidate from the queue of unplaced modules will be selected and placed on the most suitable corner. "Suitable" is quantified by definitions of "flexibilities". Two essential flexibilities are defined by modules' sizes (R_{if}) and their interconnections (FC_i) to ensure that modules with larger sizes or with more interconnections to the placed modules will have higher

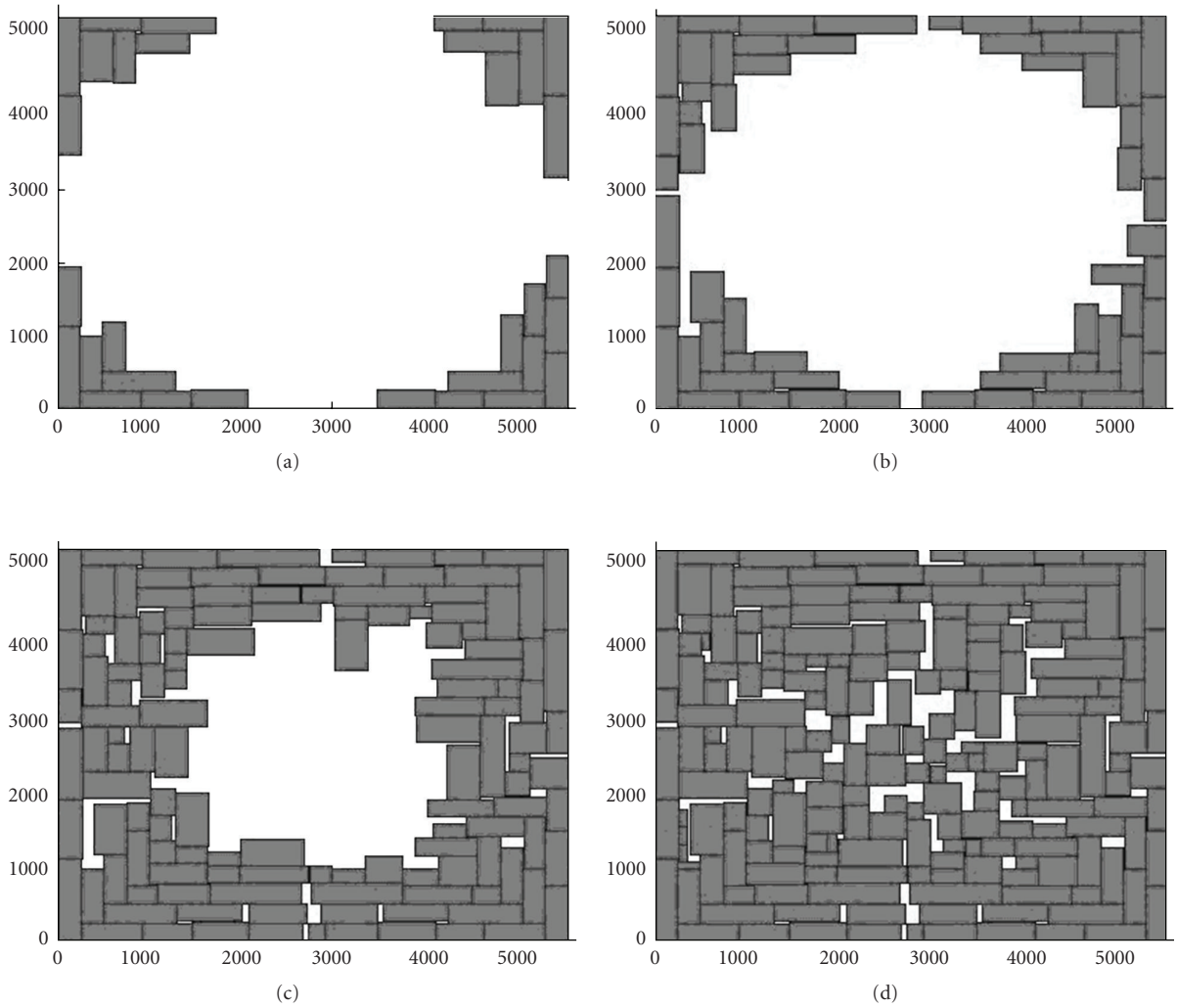


Figure 1: A typical LFF packing process.

priorities to be placed. More specifically, the suitability (or called *Fitness Value*, FV) of unplaced Module i is defined by the weighed sum of R_{if} and FC_i as in (1):

$$\mathrm{FV}_i = w_1 \cdot R_{if} + w_2 \cdot \mathrm{FC}_i, \tag{1}$$

where $R_{if} = r_1 \times (1 - a_i/(W \times H)) + r_2 \times (1 - \max(w_i, h_i)/\min(W, H))$,

$$\mathrm{FC}_i = 1 - \sum_{j=0, j \neq i}^{N} \left(\frac{W_{ij}}{W_{\mathrm{net}}} \right). \tag{2}$$

In (1), a_i, w_i and h_i are the area, width, and height of Module i. W and H are the width and height of the fixed outline. r_1, r_2, w_1, and w_2 are weight factors. W_{ij} denotes the number of nets between Module i and placed Module j while W_{net} means the total number of nets.

3.2. Sliced-LFF Algorithm. Original LFF algorithm has to store two lists. One is for all the unplaced modules, and the other is for all the corners on the current floorplan. In each packing step, the most suitable module on the most suitable corner will be chosen and placed. Therefore, LFF evaluates coordinates of the corners, modules' sizes, and their interconnections. But it has no consideration on the shape of deadspace. Unfortunately, in some specific cases, ignoring this information will cause a packing error. Figure 2(a) gives an example.

In Figure 2(a), Module e is going to be placed and there are 12 corners in the current floorplan, which are recorded in the LFF process. However, we find that Module e cannot be placed on any of these 12 corners. In this case, it will finally fail to pack Module e.

Nevertheless, we know that Module e can be placed like Figure 2(b). A further observation shows that LFF packing will become more robust if it does not store all the corners but catches the shape of deadspace. In this paper, an effective model called HSlice and VSlice structures is introduced to partition the deadspace. As shown in Figure 3, the whole deadspace is cut into five slices horizontally and

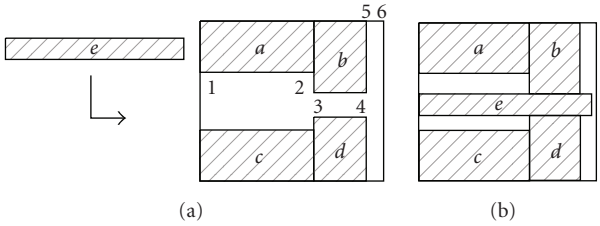


Figure 2: A Packing error in original LFF.

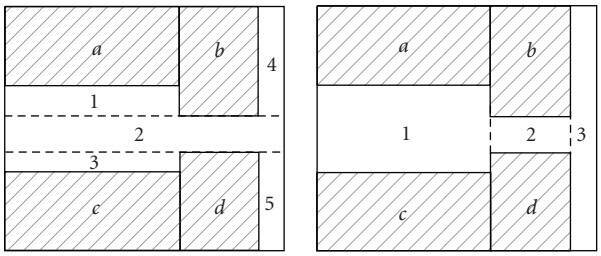


Figure 3: HSlice and VSlice structures.

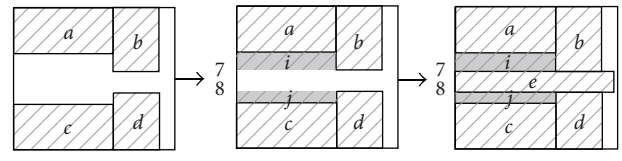


Figure 4: A Slice-based packing with fictitious modules.

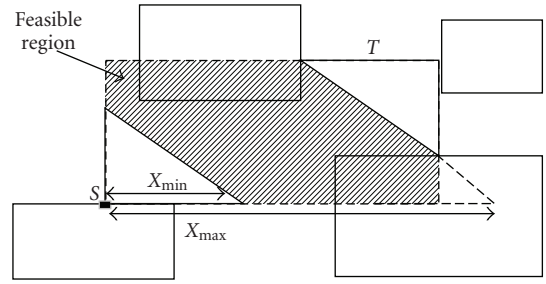


Figure 5: 2-D feasible region.

three vertically. With their help, we can take more advantages of deadspace and finally correct the error in Figure 2(a), as shown in Figure 4. In our implementation, the list of HSlice and VSlice is stored instead of the list of all the corners.

Another observation is that all the corners can be divided into two categories, depending on whether the two edges of this corner belong to different modules or not. If they do, the corner is named 90-degree corners, for example, Corners 1, 2, 5 and 6 in Figure 2(a); and if not, it is called 270-degree corners, for example, Corners 3 and 4. Furthermore, we find that higher area utilization will be achieved if modules are only arranged on 90-degree corners.

After HSlice and VSlice structures are obtained, coordinates of all 90-degree corners will be calculated by the following criterion using HSlice and VSlice.

A vertex will be a 90-degree corner if and only if

(i) this vertex belongs to one HSlice and one VSlice at the same time;

(ii) the direction of this vertex (i.e., top-left, top-right, bottom-left and bottom-right) should be the same in both HSlice and VSlice.

After the coordinates of all the corners are calculated, Module *e* will be successfully placed on a newly generated Corner 7 after defining HSlice 1 and 3 as two fictitious Module *i* and *j*, as shown in Figure 4. Furthermore, HSlice and VSlice structures will provide more information of the deadspace and facilitate deadspace planning for buffers.

4. Buffer Insertion

In order to integrate buffer insertion issue into LFF packing, we first adopt two published models on buffer insertion: FR and IFR. Based on these two models, some detailed techniques are developed for this integration. Then, how to implement the buffer insertion process in LFF will be summarized in Section 4.5. In Section 4.6, we will illustrate how the geometric structure impacts the success rate of buffer insertion.

4.1. The Notion of FR and IFR. Buffer insertion spots have their own constraints to meet the timing requirement. Basic studies on these constraints are listed as follows.

In this paper, we adopt 2D Feasible Region (FR) and Independent Feasible Region (IFR) to calculate buffer insertion spots. 2-D FR is a polygon consisting of all the feasible spots for a buffer of a net. IFR of a buffer means a region where this buffer can be placed once the other buffers of the same net have been inserted successfully in their IFRs. Figure 5 shows the 2-D Feasible Region of a 2-pin net.

For each net, inserting more buffers may not always improve the interconnect delay. There is a tradeoff between wire delay and buffer delay. That means that there is an optimal number for the buffers, which will minimize the interconnect delay of this net. We assume that this number is N and the delay with optimal number of buffers is T^N_{opt}. In our implementation, we let N vary from zero to a threshold (e.g., 30) for each net. Then, we calculate the interconnect delay with each N and record the smallest one as T^N_{opt}.

Equations to calculate the delay with N buffers can be referred in [16]. We omit them for the conciseness of this paper, since they are pretty complicated and hard to explain.

After that, all the nets can be divided into three groups and their definitions are discussed as follows.

Definition 1. Suppose that the Elmore delay and target delay of a 2-pin net are T^N_{elmore} and T^N_{tgt}, respectively. The optimal

delay with best buffer insertion is T^N_{opt}. After that, all the nets could be divided into 3 classes.

(i) Successful Nets (SNs). If either $T^N_{\text{elmore}} \le T^N_{\text{tgt}}$ or $T^N_{\text{elmore}} > T^N_{\text{tgt}} \ge T^N_{\text{opt}}$ is satisfied by successful buffer insertion, the target delay for the net will be achieved and the net is called Successful Net.

(ii) Failed Nets (FNs). If $T^N_{\text{elmore}} > T^N_{\text{tgt}} \ge T^N_{\text{opt}}$ and the buffer insertion is failed (the entire FR has been taken up by IP modules), the target delay for the net will not be achieved and the net is named Failed Nets.

(iii) Invalid Nets (INs). If $T^N_{\text{tgt}} < T^N_{\text{opt}}$, this net will not meet the target delay even by buffer insertion and the net is defined as Invalid Nets.

The goal of buffer insertion is to increase the number of SN but to decrease the number of IN and FN.

4.2. Flexibilities' Definition for Buffer Insertion. Since LFF packing process runs with the guidance of the flexibility, if we intend to consider buffer insertion issue in LFF, definitions of flexibilities for buffer insertion will be necessary. Since IN and FN should be treated differently and accordingly, we should define two flexibilities in LFF packing.

Definition 2. As listed in Section 4.1, when packing Module i, three classes of nets are generated: SN, FN, and IN. Moreover, their amounts are denoted by n^i_{SN}, n^i_{FN}, and n^i_{IN}. Meanwhile, the number of nets that are related to Module i is denoted by n^i_{NET}.

(1) Flexibility for FN of Module i (F^i_{FN}) is as follows:

$$F^i_{\text{FN}} = \frac{n^i_{\text{FN}}}{n^i_{\text{NET}}}, \tag{3}$$

where n^i_{FN} of Module i has two origins: one is the nets of Module i whose IFRs are blocked by other placed modules and the other is the nets of placed modules whose IFRs are blocked by Module i.

(2) Flexibility for IN of Module i (F^i_{IN}) is as follows:

$$F^i_{\text{IN}} = \frac{n^i_{\text{IN}}}{n^i_{\text{NET}}}, \tag{4}$$

By far, the cost function for Module i used in LFF can be extended from (1) to (5):

$$\text{FV}_i = w_1 \cdot R_{if} + w_2 \cdot \text{FC}_i + w_3 \cdot F^i_{\text{FN}} + w_4 \cdot F^i_{\text{IN}}, \tag{5}$$

4.3. True-Deadspace Based on the Guidance of FV. Figure 6 shows two packing candidates. Obviously, FV_c of placing Module c at Corner 1 will be less than Corner 2. because placing Module c at Corners 2 will block three buffers. As a result, Corner 1 has less flexibility. According to the guidance of FV_c, Module c will be placed at Corner 1 instead of Corner 2. Thus, the three buffers will be protected in the deadspace between Module a and c. This kind of deadspace is called *True-Deadspace*, corresponding to *Pseudo-Deadspace* to be defined in Section 4.4.

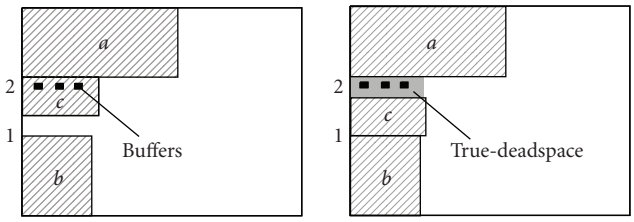


Figure 6: True-deadspace based on the guidance of FV.

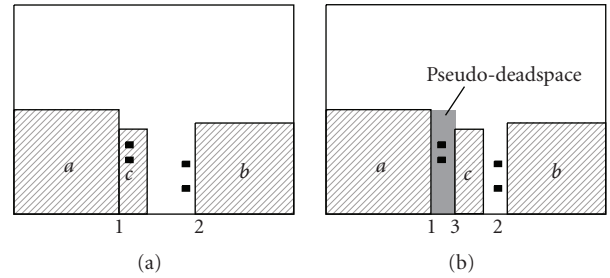


Figure 7: The Function of pseudo-deadspace.

4.4. Generating Pseudo-Deadspace. Figure 7 illustrates one fact that only using FV to guide the placement of modules in LFF Algorithm is not enough.

Notice that a corner should be always needed before placing a module in LFF. Thus, in Figure 7(a), Module *cc* should be placed at neither Corner 1 nor 2. However, both Corner 1 and 2 are worse than the placement in Figure 7(b).

LFF method only allows modules to be placed at one existing corner. Therefore, in order to generate such a placement as Figure 7(b), Pseudo-Deadspace is introduced and placed at Corner 1 as a fictitious module. With its help, Corner 3 is newly created and Module c could be placed at Corner 3. After that, the better placement in Figure 7(b) is finally achieved.

4.5. Buffer Insertion Implementation. The methods in Sections 4.3 and 4.4 have done a good allocation of deadspace for buffer insertion. Then in LFF packing process, buffers are inserted simultaneously with packing the modules. If one module is placed, its buffers related to this module are also inserted in one spot of their IFRs. However, how to select a suitable buffer spot in its IFR also needs discussions.

In the algorithm, *Pseudo-Deadspace* is treated as a fictitious module. Therefore, if buffers are inserted far from modules, the *Pseudo-Deadspace* will be huge, which will reduce the area utilization of the chip. As shown in Figure 8, Buffers 1 and 2 are far from Module a and Huge *Pseudo-Deadspace* is generated

Since buffers should be inserted close to modules to reduce the size of Pseudo-Deadspace, we use *Laplacian Operator* that is widely used in Digital Image Processing for edge detection of placed modules and finally insert buffers along the edges of modules.

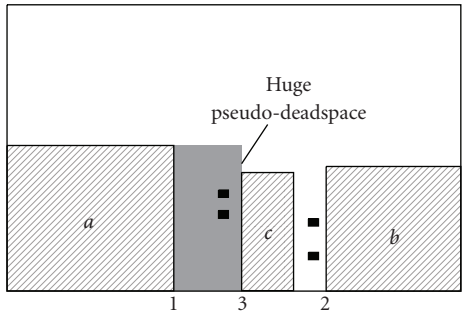


Figure 8: Huge pseudo-deadspace.

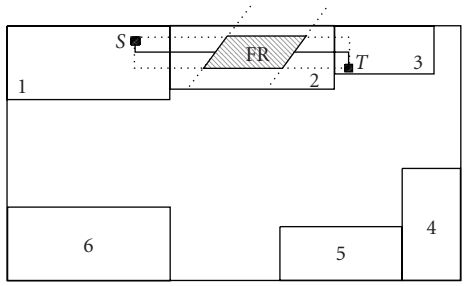


Figure 9: The noncausality during module placement.

4.6. A Comparison between SA and LFF. SA is usually based on topological representations of a floorplan. However, besides different topologies, we notice that buffer insertion is also sensitive to different geometric structures with the same topology. For example, in Figure 7(a), the placement will block two buffers while the one in Figure 7(b) blocks none. Considering that both of them have the same topological structure, it means that topological representation in SA cannot distinguish the difference between Figures 7(a) and 7(b). Some critical geometric information for buffer insertion will be lost.

On the contrary, LFF-based algorithm can tell the geometric difference within the same topology, like Figures 7(a) and 7(b). The packing order of LFF can also be optimized to decide a suitable topological structure among the modules. Thus, both geometric and topological information will be considered for buffer insertion in LFF, which is the motivation of our work.

5. Improved Buffer Insertion

LFF has no backtracks in its packing process. Therefore, further studies of LFF packing process will come up with a problem called *Noncausality*. To alleviate the impact of this problem, a net-classing strategy is put forward according to different characteristics of the nets. Moreover, a 2-staged LFF method and a post greedy algorithm are developed based on this net-classing strategy.

5.1. The Challenge in LFF Packing Process. The process of the LFF algorithm is to place the modules one by one. When considering buffer insertion issue, one problem will come up, as illustrated in Figure 9.

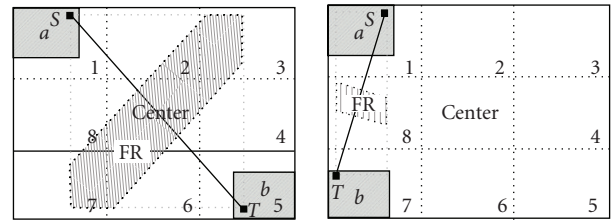


Figure 10: Two classes of nets.

Table 1: Characteristics of two classes of nets.

	Amount	Buffers needed per net	FR's vitality
CCN	Fewer	More	Stronger
CNN	More	Fewer	Weaker

In Figure 9, Modules 1, 2, and 3 are placed in order. The FR between Module 1 and 3 is occupied by Module 2. However, it is impossible to consider this case when placing Module 2, because Module 3 has not been placed at that time and this FR between Module 1 and Module 3 cannot be calculated in advance. This is called *Noncausality*.

If we can predict Module 3's location, the effect of *Noncausality* can be greatly handled. However, because of the complexity of the FRs, such a precise prediction is usually hard to implement. It means that completely avoiding the impact of *Noncausality* is not practical. It is a *chicken-egg* problem. However, we propose an effective method to alleviate this impact.

5.2. Net Classing Strategy. Fortunately, we find that the impact of *Noncausality* differs from two different classes of 2-pin nets. In one class, the line from source to sink has a slope close to +1 or −1 and these 2 pins are far from each other. Therefore, they have larger FRs than the other group of nets, as mentioned in Figure 10(a). Larger FRs mean stronger vitalities and it is easier to find a new spot candidate in such a large FR even if a placed module may occupy the original spot. Since the line from their source to sink usually crosses the center of the chip, they are called *Center-Crossing Nets* (CCNs) while the other class is named *Center-Noncrossing Nets* (CNNs). Table 1 shows their characteristics.

In Table 1, the number of CCN is usually smaller than CNN due to wirelength optimization. Meanwhile, CCN has longer distance from source to sink and will need more buffers than CNN per net. In this paper, a quantitative definition of CCN and CNN will be needed.

Definition 3 (*Center-Crossing* and *Center-Noncrossing Nets*). In Figure 10, the whole chip is cut into nine grids. In our implementation, nine is enough to define the difference between CCN and CNN. The middle one is marked as Center while the others are numbered with 1 to 8 clockwise around the Center.

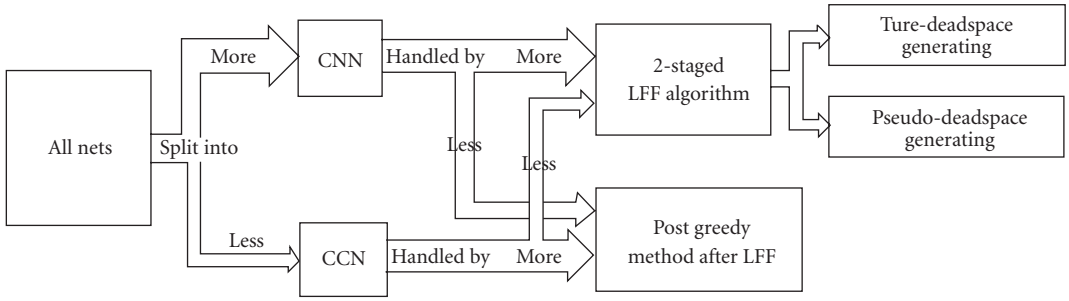


Figure 11: The Relation of All the Techniques in This Paper.

For a 2-pin net, suppose that its source and sink pins locate in *Region* N_S and *Region* N_T, respectively, if any of the three conditions

$$(N_S + 3) \bmod 8 = N_T,$$
$$(N_S + 5) \bmod 8 = N_T, \tag{6}$$
$$((N_S + 4) \bmod 8 = N_T) \wedge (N_S \bmod 2 = 1),$$

is met, the net will be defined as CCN, for example, the one in Figure 10(a). Otherwise, it is called as a CNN, for example, the one in Figure 10(b).

All nets of a circuit will be divided into these two classes at last. Based on the different impacts of *Noncausality* between CCN and CNN, a 2-staged LFF method and a postmethod are put forward.

5.3. Algorithm Framework: 2-Staged LFF. Figure 10 shows that the FRs of a CCN are much larger. So they have more buffer insertion candidates than CNN.

Meanwhile, since the modules are placed one after another, the chief objective in (5) could be different as the packing step continues. For example, at the beginning of packing, we focus mainly on the area utilization, wirelength and so on. When only a few modules are left unplaced, how to place them in the fixed-outline constraint and maximize the convergence of our algorithm becomes the key issue.

Similarly, in buffer planning process, the algorithm is segmented into two stages. In Stage 1, a more strict constraint is adopted to improve the success rate of buffer insertion mainly for CNNs, whose FRs are smaller and harder to insert buffers. In Stage 2, since fewer modules are left unplaced and the chief goal is shifted to improve the success rate of fixed-outline floorplanning, a more tolerant rule will be adopted for buffer insertion. Therefore, the success rate of buffer insertion will shrink at that time. However, buffer failures in this period mostly come from CCNs, which have larger FRs and are easier to find an alternative insertion spot. When the LFF ends, the post floorplanning procedure will be invoked to save these failed CCNs.

5.4. A Post Greedy Method after LFF Packing. Since 2-staged LFF in Stage 1 ensures a high success rate of CNNs and leaves many failed CCNs in Stage 2, a postmethod is developed. The goal of this post floorplanning method is to increase CCNs' success rate on buffer insertion since their buffers have stronger vitalities.

In this method, a queue of all failed buffers is set up. Buffers with smaller FR and fewer insertion candidates will be inserted in the foreside of the queue. Then we pick up one buffer from the queue and search a new spot in its IFR to insert it, and the next buffer follows. Using this greedy method, we save failed buffers as many as possible.

The experiments in Section 6.2 show that lots of failed nets will be saved by this greedy method.

5.5. The Relation of All Algorithm Details. Finally, all the techniques discussed in Sections 3, 4, and 5 are integrated in Figure 11. "More" and "Less" describe the importance of these techniques to the two classes of nets. For instance, more failed CNNs are saved by our 2-staged LFF procedure than the Post Greedy Method. In other words, the 2-staged LFF is more important than the greedy method for CNNs. Thus, the arrowhead from CNN to 2-staged LFF is thicker than the one from CNN to our Post Greedy Method.

Besides, a flow chart of our algorithm is shown in Figure 12. Our algorithm consists of two parts: 2-staged LFF and a Post Greedy Method. The threshold for the two stages of LFF is 20%. That means that Stage 2 will be started if 80% of circuit modules are successfully packed. The algorithm will return false if no feasible solutions could be found in each packing step.

6. Experimental Results

Our work is implemented using ANSI C and all the experiments are completed on a 1.6-GHz Intel PC with 1 GB memory. The weight factors w_1, w_2, w_3, and w_4 in (5) are set to 0.3, 0.3, 0.2, and 0.2, respectively.

6.1. Comparison Experiments. In this paper, we integrate buffer insertion issue into fixed-outline floorplanning. To test the result of our buffer planning algorithm, an SA-based outline-free floorplanning algorithm with buffer planning [12] is adopted as the comparison work. Experiments are finished to explain the comparison between our LFF packing algorithm (marked as F2) and the SA-based work (marked as F1) [12].

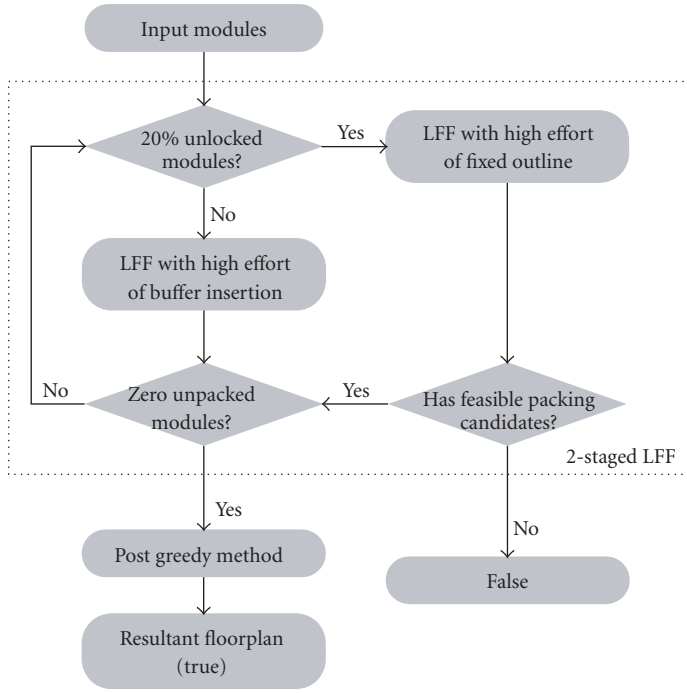


Figure 12: Flow chart of the algorithm.

We use six MCNC standard benchmarks and one generated benchmark M198. All the multipin nets are split into 2-pin nets just like [12]. All the power and ground interconnects are ignored. To generate the delay constraints, a floorplan is acquired by running a common SA-based floorplanner at first. Then for each net, we compute its T^N_{opt} and assign the target delay as $1.1 * T^N_{\text{opt}}$. All the pretreatments in this section are kept the same as the comparison floorplanner F1 [12].

The results are listed in Table 2(a) and 2(b) with different needs of buffers in different feature sizes. Table 2(b) does not include the first three benchmarks in Table 2(a), because they are not included in [12] either.

Our work achieves 40.7% and 37.1% improvement on buffer insertion rate, compared with the SA-based work [12]. As a result, the number of FN has been decreased while the number of SN has been increased. Meanwhile, our algorithm is more than 100 times faster than [12], since it is deterministic without any iterations or packing backtracks.

In Table 2(a) and 2(b), the fixed-outline constraint with larger area than F1 is required, because more buffers are inserted in our LFF method and they should consume more deadspace. However, F2 has similar wirelength as F1.

From Table 2(a), we can also find that the increase on the number of SN achieved by our work will be abated when the benchmark has fewer modules. For example, in Xerox of Table 2(a), there is even a decrease on SN compared with F1 [12]. This is because the solution space is more discrete when the floorplanner has fewer modules to operate. That means that too few modules may not provide enough floorplanning candidates to make better use of the deadspace. However, our floorplanner is usually invoked in hierarchical design. So, the number of modules in each hierarchy could totally be decided using a different threshold setting in the partitioning stage. We can choose a number, which is not quite small and easier for our buffer planning algorithm.

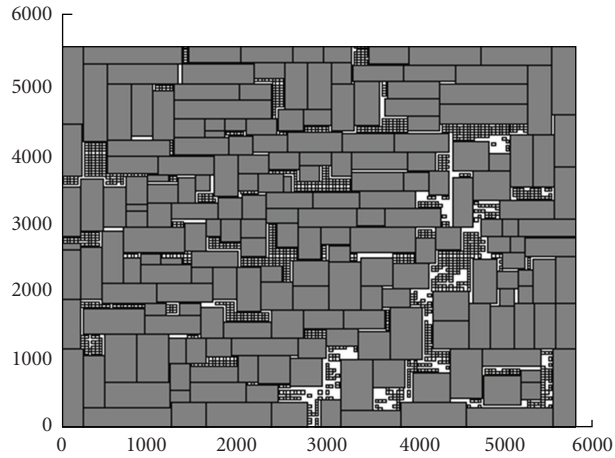


Figure 13: Floorplan of M198 with buffers.

At last, a practical floorplan for M198 is presented in Figure 13.

6.2. Experiments on Improved Buffer Insertion. In this section, detailed experiments on M198 are performed to explore the internal mechanism of LFF packing. It shows that the efficiency of basic methods in Section 3 and 2-staged LFF based on net-classing strategy in Section 4, as in Table 3.

Table 2

(a) The comparison between the SA-based and LFF-based floorplanner

	Area (mm^2)		Wirelength (mm)		#Inserted Buffer/#Buffer		#Successful Nets		#Failed Nets		Timing Slack per FN (ps)		Time(s)	
Methods	F1	F2	F1	F2	F1	F2	F1	F2	F1	F2	F1	F2	F1	F2
Apte	48.32	53.25	459.6	563.7	37/123	102/116	103	118	52	18	23.4	20.2	46	0.15
Xerox	86.20	92.00	671.5	783.8	114/203	247/265	388	349	39	15	32.1	28.9	109	0.3
Hp	39.56	40.56	216.1	196.0	60/121	60/87	145	160	58	17	19.1	18.8	35	0.2
Ami33	32.00	33.50	96.7	110.9	194/386	164/176	204	247	20	8	18.7	16.7	26	1.3
Ami49	172.2	158.8	1489.0	1462.8	277/545	435/447	330	448	161	30	43.0	41.2	760	3.7
Playout	460.6	470.0	12053	12970	613/1316	806/924	1564	1788	365	71	56.4	52.5	2330	46.6
M198	34.52	33.08	618.0	491.1	594/1232	741/833	1587	1813	536	157	17.8	14.9	2538	120.6
Average	+2.03%		+3.74%		+40.7%		+14.3%		−70.3%		−10.3%		×163	

(b) The Comparison between the SA-based and LFF-based Floorplanner with larger needs of buffers

	Area (mm^2)		Wirelength (mm)		#Inserted Buffer/#Buffer		#Successful Nets		#Failed Nets		Timing Slack per FN (ps)		Time(s)	
Methods	F1	F2	F1	F2	F1	F2	F1	F2	F1	F2	F1	F2	F1	F2
Ami33	33.50	34.00	99.1	104.3	236/409	339/354	196	227	32	11	14.3	13.8	396	1.62
Ami49	162.0	162.4	1507.0	1505.2	364/892	812/958	299	330	184	49	29.2	25.7	890	4.74
Playout	460.6	467.7	12363	12189	1240/2825	1597/1972	1230	1421	395	265	35.4	32.8	2920	52.5
M198	33.24	34.20	606.2	523.3	1329/3559	1677/2513	1240	1493	445	351	15.6	14.3	3975	138.3
Average	+1.54%		−1.42%		+37.1%		+15.5%		−48.3%		−8.6%		×129	

Table 3: Detailed results of M198 floorplanning.

	CCN		CNN	
	2-Staged LFF	Postmethod	2-Staged LFF	Postmethod
% of #Nets	7.4%		92.6%	
% of #Buffers	22.2%		77.8%	
#SN	62	134	1505	1704
#FN	76	4	220	21
#SN/(#FN+#SN)	44.9%	97.1% (+53.8%)	87.2%	98.8% (+11.6%)

(1) Characteristics of the 2 Classes of Nets. In Table 3, the percentages of CCNs and CNNs are 7.4% and 92.6%, since fewer CCNs are helpful to reduce the chip wirelength. However, CCN usually needs more buffers than CNN. The percentages of CCNs' and CNNs' buffer needs are 22.2% and 77.8%, which proves the description of the two classes of nets in Table 1.

(2) Collaboration of Methods in Figure 11. Moreover, Table 3 shows that the strategy listed in Section 4 is useful. In 2-staged LFF, CNNs are better settled and the ratio between SN and FN is 1505 : 220. The ratio is much higher than CCN (i.e., 62 : 76).

When LFF packing ends, the post greedy method is invoked. The ratio 62 : 76 has been increased to 134 : 4. Compared to CNN's increase (+11.6%), the post greedy method in Stage 2 is more important for CCN (+53.8%), which also proves the conclusion in Figure 11.

7. Conclusions and Future Work

In this paper, we develop a more robust version of LFF called Sliced-LFF and integrate buffer planning into LFF packing with a fixed-outline constraint. By using net-classing and 2-staged LFF strategy, the impact caused by *Noncausality* during module placement is greatly alleviated. A great improvement on the success rate of buffer insertion (40.7% and 37.1% in different feature sizes) is achieved, compared with an SA-based previous work.

Currently, all the nets are handled with the same priority. In practical designs, some nets are more important than the others, such as clock tree. The priority of the nets will be considered in future work.

References

[1] J. A. Roy, S. N. Adya, D. A. Papa, and I. L. Markov, "Min-cut floor-placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 7, pp. 1313–1326, 2006.

[2] S. N. Adya and I. L. Markov, "Fixed-outline floorplanning: enabling hierarchical design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 6, pp. 1120–1135, 2003.

[3] J. Cong, T. Kong, and D. Z. Pan, "Buffer block planning for interconnect-driven floorplanning," in *Proceedings of*

the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '99), pp. 358–363, November 1999.

[4] P. Sarkar, V. Sundararaman, and C. K. Koh, "Routability-driven repeater block planning for interconnect-centric floorplanning," in *Proceedings of the International Symposium on Physical Design (ISPD '00)*, pp. 186–191, April 2000.

[5] X. Tang and D. F. Wong, "Planning buffer locations by network flows," in *Proceedings of the International Symposium on Physical Design (ISPD '00)*, pp. 180–185, April 2000.

[6] F. F. Dragan, A. B. Kahng, I. Mandoiu, S. Muddu, and A. Zelikovsky, "Provably good global buffering using an available buffer block plan," in *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD '00)*, pp. 104–109.

[7] C. W. Sham and E. F. Y. Young, "Routability driven floorplanner with buffer block planning," in *Proceedings of the International Symposium on Physical Design (ISPD '02)*, pp. 50–55, April 2002.

[8] F. Rafiq, M. Chrzanowska-Jeske, H. H. Yang, and N. Sherwani, "Inegrated floorplanning with buffer/channel insertion for bus-based microprocessor designs," in *Proceedings of the International Symposium on Physical Design (ISPD '02)*, pp. 56–61, April 2002.

[9] C. J. Alpert, J. Hu, S. S. Sapatnekar, and P. G. Villarrubia, "A practical methodology for early buffer and wire resource allocation," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 189–194, June 2001.

[10] S. Chen, X. Hong, S. Dong, and Y. Ma, "A buffer planning algorithm based on dead space redistribution," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '03)*, pp. 435–438, Kitakyushu, Japan, January 2003.

[11] I. H.-R. Jiang, Y.-W. Chang, J.-Y. Jou, and K.-Y. Chao, "Simultaneous floorplanning and buffer block planning," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '03)*, pp. 431–434, Kitakyushu, Japan, January 2003.

[12] Y. Ma, X. Hong, S. Donag, S. Chen, C. K. Cheng, and J. Gu, "Buffer planning as ah integral part of floorplanning with consideration of routing congestion," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 4, pp. 609–620, 2005.

[13] Q. Dong, B. Yang, J. Li, and S. Nakatake, "Incremental buffer insertion and module resizing algorithm using geometric programming," in *Proceedings of the 19th ACM Great Lakes Symposium on VLSI (GLSVLSI '09)*, pp. 413–416, May 2009.

[14] X. He, S. Dong, Y. Ma, and X. Hong, "Simultaneous buffer and interlayer via planning for 3D floorplanning," in *Proceedings of the 10th International Symposium on Quality Electronic Design (ISQED '09)*, pp. 740–745, March 2009.

[15] S. Dong, X. Hong, Y. Wu, Y. Lin, and J. Gu, "VLSI block placement using less flexibility first principles," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '01)*, pp. 601–604, Yokohama, Japan, January 2001.

[16] C. Alpert and A. Devgan, "Wire segmenting for improved buffer insertion," in *Proceedings of the 34th Design Automation Conference*, pp. 588–593, June 1997.