IMPLEMENTATION OF EVOLUTIONARY ALGORITHMS FOR BDD MAPPED CIRCUITS TO IMPROVE PERFORMANCE PARAMETERS

Dissertation submitted in partial fulfillment of the requirements
for the award of the degree of

Master of Technology

In

VLSI Design

Submitted by

Rupinder Kaur

Roll. No. 601261023

Under the supervision of

Mrs. Manu Bansal

Assistant Professor, ECED

Department of Electronics & Communication Engineering

Thapar University, Patiala

INDIA

July, 2014

CERTIFICATE

I hereby declare that the work which is being presented in the dissertation entitled, **"Implementation of Evolutionary Algorithms for BDD Mapped Circuits to improve the Performance Parameters"** in partial fulfillment of the requirement for the award of degree of Master of Technology (VLSI Design) at the department of Electronics and Communication Engineering, Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Mrs. Manu Bansal, Assistant Professor, ECED.

The matter presented in this dissertation has not been submitted in any other University/Institute for the award of any other degree.

Rupinder Kaur.

Date: 17 July 2014

Rupinder Kaur
Roll.No.601261023

It is certified that the above statement made by the student is correct to the best of my knowledge and belief.

MBansal 17/7/14

Mrs. Manu Bansal
Assistant Professor
ECE Department
Thapar University

Counter signed by:

Dr. Sanjay Sharma
Professor & Head
ECED, Thapar University
Patiala-147004

Dr. S. K. Mahopatra
Dean of Academic Affairs
Thapar University
Patiala-147004

ACKNOWLEDGEMENTS

I take this opportunity to express my profound sense of gratitude and respect to all those who helped me through the duration of this dissertation. I acknowledge with gratitude and humility my indebtedness to **Mrs. Manu Bansal, Assistant Professor,** Department of Electronics and Communication Engineering, Thapar University, Patiala, under whose guidance I had the privilege to complete this dissertation. I wish to express my deep gratitude towards her for providing individual guidance and support throughout the dissertation work.

I convey my sincere thanks to **Head of the Department, Dr. Sanjay Sharma** as well as **PG Coordinator, Dr. Kulbir Singh, Associate Professor, ECED**, entire faculty and staff of Electronics and Communication Engineering Department for their encouragement and cooperation.

My greatest thanks are to all who wished me success especially my family. Above all I render my gratitude to the Almighty who bestowed ability and strength in me to complete this work.

Rupinder Kaur

This work is dedicated to my father, mother and my brother who shaped much of the way I think, instilled my confidence and technical desire, and unknowingly coined many of phrases by which I live my life.

ABSTRACT

Binary Decision Diagram is the data structure to implement and store any Boolean function. It is a canonical representation and is very similar to binary decision tree but it is a graph instead of a tree. It is a canonical representation based on recursive Shannon expansion. The ordering of BDDs is essential to reduce the number of nodes, hence minimize the area. The power dissipation and switching activity can also be optimized by suitable variable ordering. Several algorithms can be used for optimizing these performance parameters.

Genetic Algorithms are a family of computational models inspired by evolution. GA is useful when no mathematical analysis is available. The performance of genetic algorithm depends, to a great extent, on the performance of the crossover operator used. Crossover is the primary method of optimization in the genetic algorithm, and works by combining sub-placements from two different parent configurations to generate a new placement.

In Hybridized Genetic Algorithm, The Genetic Algorithm has been embedded with Branch and Bound Algorithm to intelligently search for the best solution of an optimization problem. The objective of Hybridized Genetic Algorithm is to find an optimal ordering of BDDs with reduced node count and computation time.

The switching activity of a circuit node in a CMOS digital circuit directly contributes to the overall dynamic power dissipation. Temporal correlation of the occurring input signals can have a significant effect on the switching activity and hence the power consumption. The power dissipation estimate for a mapped BDD node is based on its switching activity and its fan out (corresponding to the capacitive load).

List of Contents

List of Abbreviations

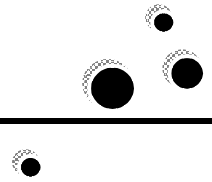
ABBREVIATION	MEANING
BDD	Binary Decision Diagram
DAG	Propositional Directed Acyclic Graph
ROBDD	Reduced Ordered Binary Decision Diagram
CAD	Computer Aided Design
GA	Genetic Algorithm
HGA	Hybridized Genetic Algorithm

List of Figures

8.5	Comparison of the power calculated Using Probabilistic Technique and Existing Technique in Genetic Algorithm for Multi-input Adder Circuits
8.6	Comparison of the power calculated Using Probabilistic Technique and Existing Technique in Hybridized Genetic Algorithm for Multi-input Adder Circuits

List of Tables

Chapter-1
Introduction



1.1 Binary Decision Diagram

A binary decision diagram (BDD) or branching program [1] is a data structure that is used to represent a Boolean function. On a more abstract level, BDDs can be considered as a compressed representation of sets or relations.

A Boolean function can be represented as a rooted, directed, acyclic graph, which consists of several decision nodes and terminal nodes. There are two types of terminal nodes called 0-terminal and 1-terminal.The nodes are joined by the direction edges. BDD is called a Directed Acyclic Graph because there is no way to loop back to the same edge. Each decision node N is labelled by Boolean variable V_N and has two child nodes called low child and high child. The direction edge from node V_N to a low (or high) child represents an assignment of V_N to 0 (resp. 1). Such a BDD is called 'ordered' if different variables appear in the same order on all paths from the root. A BDD is said to be 'reduced' if the following two rules have been applied to its graph:

- Merge any isomorphic sub graphs.
- Eliminate any node whose two children are isomorphic.

1.1.1 Reduced Ordered Binary Decision Diagram

In popular usage, the term BDD almost always refers to Reduced Ordered Binary Decision Diagram (ROBDD) [14] in the literature, used when the ordering and reduction aspects need to be emphasized). The advantage of an ROBDD is that it is canonical (unique) for a particular function and variable order [1]. A path from the root node to the 1-terminal represents a (possibly partial) variable assignment for which the represented Boolean function is true. As the path descends to a low (or high) child from a node, then that node's variable is assigned to 0 (resp. 1).

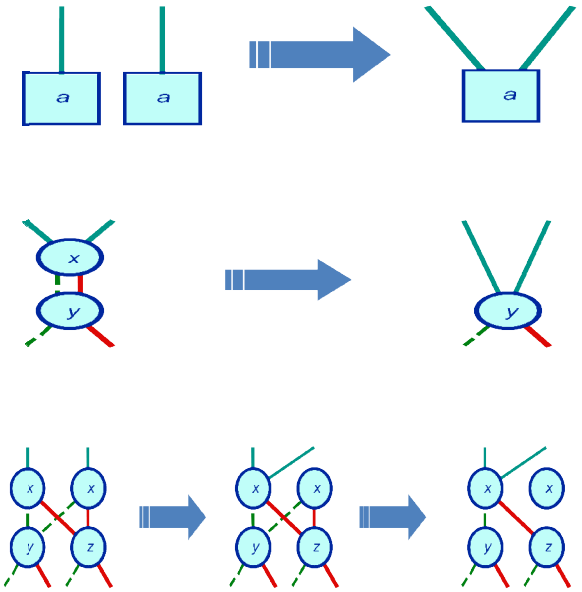


Figure 1.1 Reduction rules for BDD.

1.1.2 Example of BDD

Firstly we will consider a Binary Decision Tree representing the Boolean function $f=x_1x_2+x_3x_4$ as seen in figure 1.2.

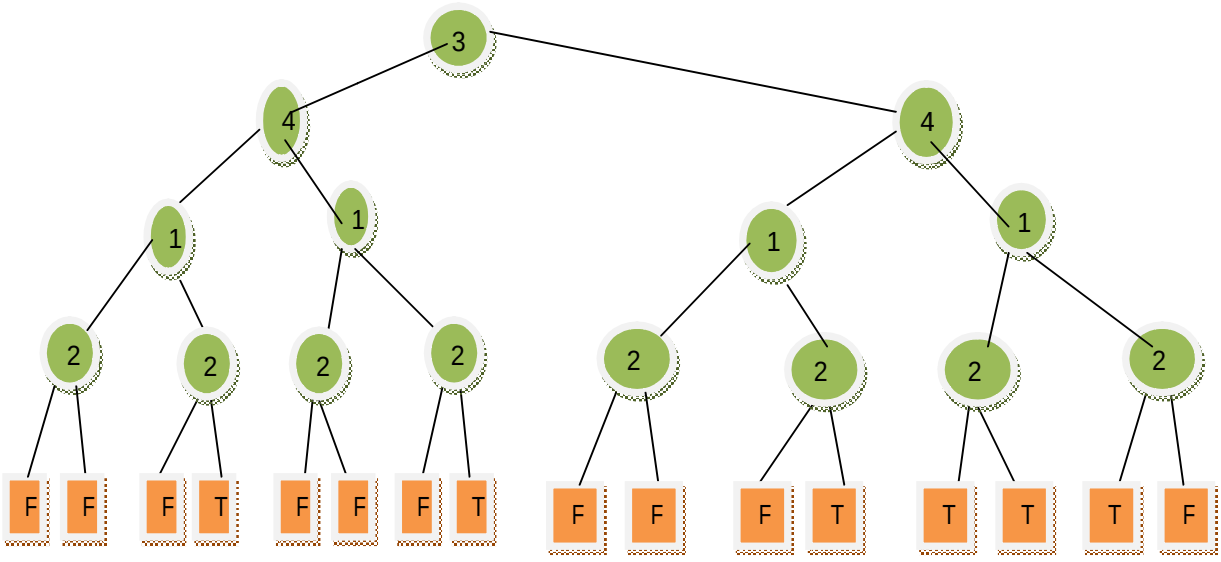


Figure 1.2 Binary Decision Tree of the Boolean function $F=X_1X_2+X_3X_4$ [2]

It is not a concise representation as the number of nodes for n variables are 2^{n+1}-1.Therefore, we use the BDDs which is the compressed form based on the Shannon expansion as seen in figure 1.3.

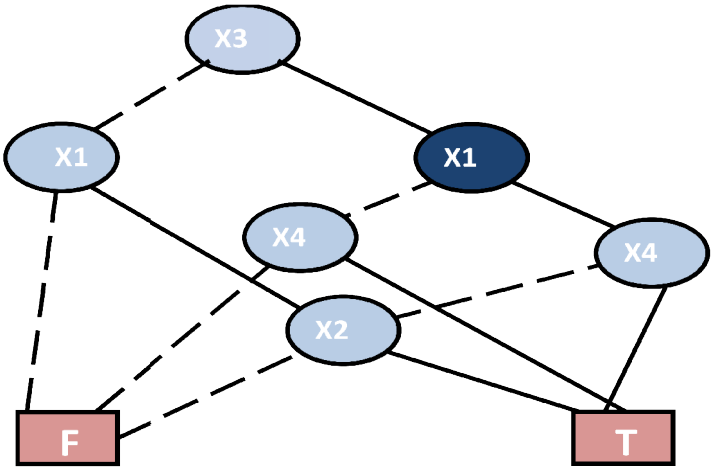


Figure 1.3 Binary Decision Diagram for the function F=x1x2+x3x4.

Ordered binary decision diagram (OBDD), was proposed by Bryant. The OBDD [2] is a DAG in which different variables appear in same order on all paths from the root.If all the equivalent and isomorphic nodes are merged and all the redundant nodes are removed from the OBDD, the resultant BDD is called Reduced ordered BDD. One such structure for the function is shown in figure 1.4.

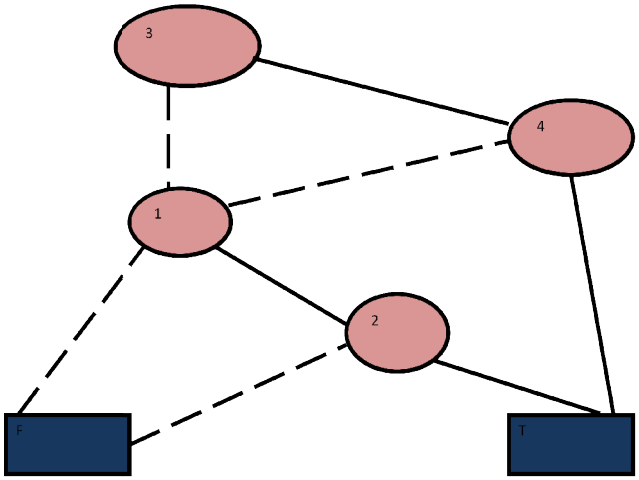


Figure 1.4 Reduced Ordered BDD of the same Boolean function [2]

1.1.3 Applications of BDDs

1. BDDs are extensively used in CAD software to synthesize circuits (logic synthesis) and in formal verification and testing.

2. There are several lesser known applications of BDD, including Fault tree analysis, Bayesian Reasoning, Product Configuration, and Private information retrieval .

1.1.4 Variable Ordering

The size of the BDD is determined both by the function being represented and the chosen ordering of the variables [3]. There exist Boolean functions $f(x_1,\ldots..x_n)$ for which depending upon the ordering of the variables we would end up getting a graph whose number of nodes would be linear (in n) at the best and exponential at the worst case (e.g., a ripple carry adder). Let us consider the Boolean function $f(x_1\ldots x2n) = x_1x_2+x_3x_4+\ldots\ldots\ldots x_{2n-1}x_{2n}$. Using the variable ordering x1<x3<……<x2n-1<x2<x4<x2n, the BDD needs 2^{n+1} nodes to represent the function. Using the ordering, x1<x2<x3<x4……..<x2n, the BDD consists of 2n+2 nodes.

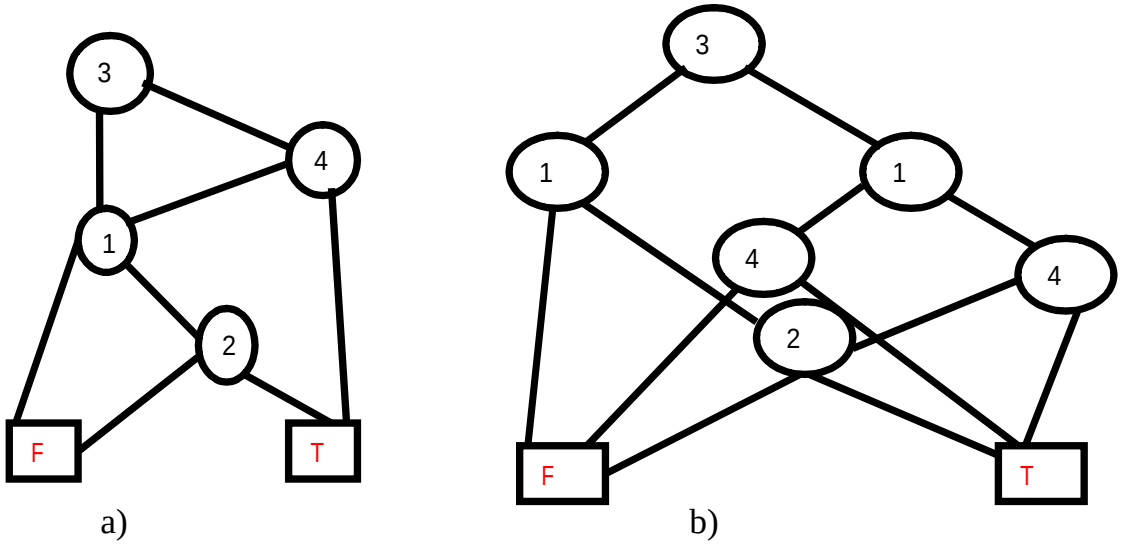
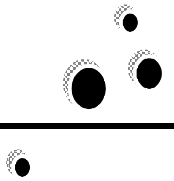


Figure 1.5 a) BDD (xlx2 + x3x4(2, 1, 4, 3)) (4 nodes) and b) BDD (xlx2 + x3x4(2, 4, 1, 3)) (6 nodes) [2]

Chapter-2

Literature Review



Many papers on the recent researches and developments in the field of Genetic and Hybridized Genetic Algorithms for the optimization of BDDs were studied. A succinct review based on the study of these papers is as follows:

1. Randal E. Bryant (1986) [1]

In this paper Binary Decision Diagram has been represented as a new data structure for representing Boolean functions. An associated set of manipulation algorithms have also been presented. Functions have been represented by directed, acyclic graphs with restrictions on the ordering of decision variables in the graph. Although a function requires, in the worst case, a graph of size exponential in the number of arguments, many of the functions encountered in typical applications have a more reasonable representation. Experimental results have been presented from applying the algorithms to problems in logic design verification.

2. Steven J. Friedman, Kenneth J. Supowit (1990) [2]

In this paper, an algorithm has been presented to find ordering of binary decision diagrams leading to the most compact representation. The ordered binary decision diagram is a canonical representation for Boolean functions, presented by Bryant as a compact representation for a broad class of interesting functions derived in circuits. However, the size of the diagram is very sensitive to the choice of ordering on the variables; hence, for some applications, such as differential cascode voltage switch (DCVS) trees, it becomes extremely important to find the ordering leading to the most compact representation.

3. Nagisa Ishiura, Hiroshi Sawada, Shuzo Yajima(1991) [3]

In this paper the exact minimization algorithm and gradual improvement methods of minimizing binary decision diagrams (BDD’s) have been presented. In the exact

minimization algorithm the optimum order is searched by the exchanges *o!* variables of BDDs based on the framework of Friedman's algorithm. The use of BDD representation of a given function and intermediate functions makes it possible to introduce pruning into the method, which drastically reduces the computation cost.

4. N. Zhuang, M.S.T. Benten, P.Y.K. Cheung (1996) [4]

In this paper a new algorithm for variable ordering of binary decision diagram (BDD) has been presented. The algorithm is based on a novel formulation of the Genetic Algorithm (GA) employing three dynamic GA parameters: population size, mutation rate and stop criteria. Test results using LGSynth93 benchmark circuits show that the new algorithm offers considerable improvements on large circuits when compared with previously published results.

5. Rolf Drechsler, Mikael Kerttu, Per Lindgren, and Mitchell Thornton (2001) [6]

In this paper technique for minimizing the overall sum of switching probabilities and hence power dissipation has been presented. The minimization of power consumption is an important design constraint for circuits used in portable devices. The switching activity of a circuit node in a CMOS digital circuit directly contributes to overall power dissipation. By approximating the switching activity of circuit nodes as internal switching probabilities in Binary Decision Diagrams (BDDs), it is possible to estimate the dynamic power dissipation characteristic of circuits resulting from a structural mapping of a BDD. Experimental results on a set of MCNC benchmarks are given for this technique.

6. Orna Grumberg, Shlomi Livne, Shaul Markovitch(2003) [8]

In this paper an approach to find optimal order for a BDD has been proposed in which the variable ordering algorithm gains "ordering experience" from training models and uses the learned knowledge for finding good orders. The size and complexity of software and hardware systems have significantly increased in the past years. As a result, it is harder to guarantee their correct behavior. One of the most successful methods for automated verification of finite-state systems is model checking. Most of the current model-checking systems use binary decision diagrams (BDDs) for the representation of the tested model and in the verification process of its properties. Generally, BDDs allow a canonical compact representation of a Boolean function (given an order of its variables).The more compact the BDD is, the better performance one gets from the verifier. The method

discussed in paper is based on offline learning of pair precedence classifiers from training models, that is, learning which variable pair permutation is more likely to lead to a good order. For each training model, a number of training sequences are evaluated. Every training model variable pair permutation is then tagged based on its performance on the evaluated orders. The tagged permutations are then passed through a feature extractor and are given as examples to a classifier creation algorithm. Given a model for which an order is requested, the ordering algorithm consults each precedence classifier and constructs a pair precedence table which is used to create the order. The algorithm was integrated with SMV, which is one of the most widely used verification systems.

7. Shun-Shii Lin,Chun-Jen Wei(2005) [9]

In this paper a new approach for optimal ordering of BDDs has been proposed that successfully finds the optimal variable orderings for almost all reduced ordered binary decision diagrams (BDDs) in the LGSynth91 benchmark circuits with up to 500 variables. All previously known approaches could solve only functions with less than 64 variables. The progress of the new approach is attributable to the concept of randomized algorithms, which significantly reduce the influence of the initial variable ordering on the minimization performance. The performance of the proposed approach has been illustrated through its application on LGSynth91 benchmark circuits. In addition to providing a feasible BDD minimization algorithm, statistical results and analyses have also been presented that could be helpful for related research.

8. Wang Mingquan,Yu Haibin(2005) [10]

In this paper a BDD minimization algorithm based on a genetic tabu hybrid strategy has been proposed which combines the global space search of genetic algorithm with the local neighborhood search and the gradual global optimizing of tabu search. The variables swap and the uniform distribution have been utilized to generate neighborhood solutions in the search space. Experiment results have been given to demonstrate the efficiency of the presented approach. A comparison to other minimization algorithms has been reported.

9. P.W. C. Prasad, A. Assi, A. Harb and V.C. Prasad (2006) [14]

In this paper an improved variable ordering method to obtain the minimum number of nodes in Reduced Ordered Binary Decision Diagrams (ROBDD) has been proposed. The

graph topology to find the best variable ordering has been used in this method. The input Boolean function has been converted to a unidirectional graph. Three levels of graph parameters have been used to increase the probability of having a good variable ordering with the initial level using the total number of nodes (NN) in all the paths, the total number of paths (NP) and the maximum number of nodes among all paths (MNNAP). Two extra parameters have been used in the second and third levels: The shortest path among two variables (SP) and the sum of shortest path from one variable to all the other variables (SSP). A permutation of the graph parameters has been performed at each level for each variable order and the number of nodes has been recorded. The proposed method is found to be effective in finding the variable ordering for the majority of benchmark circuits.

10. Piotr Porwik, Piotr Zaczkowski, Krzysztof Wrobel (2006) [13]

In this paper a spectral method for the optimal variable ordering of binary decision diagram has been presented. The complexity in finding the best order has been improved with this method. Results have been presented with effective optimization of the number of nodes and computation time of the BDD than the linear sifting algorithm.

11. Rolf Drechsler, Gorschwin Fey (2006) [12]

In this paper, an algorithm to minimize the number of paths in BDDs has been presented. The complexity of circuit and systems design increases rapidly. Therefore, a main focus of research in the area of electronic-design automation is efficient algorithms and data structures. Among these, binary decision diagrams (BDDs) have been used in a wide variety of applications and were intensively studied from a theoretical point of view. But mostly, when complexity issues were considered, only the number of nodes in a BDD has been analyzed. In this paper minimizing the number of paths in BDDs from a theoretical and a practical point of view has been proposed. Experimental results show the efficiency of the algorithm.

12. F. Towhidi, A.H. Lashkari, R.S. Hosseini (2009) [18]

In this paper the author has introduced BDD (binary decision diagram) and its variants OBDD (ordered binary decision diagram) and ROBDD (reduced ordered binary decision diagram). Then a review has been made on applications of BDD.

13. Awinash Kumar, Ajit Kumar, Soumik choudhary and P.Y. Yarde (2010) [20]

For determining the optimal ordering, use of "Genetic algorithm (GA)" is presented in this paper. In this paper a generalized method for the selection of optimal ordering of basic events using GA has been proposed, which is not based on any heuristic previously given. Main idea in the application of GA in BDD size optimization is to define population size and representation of ordered set of variable as chromosome.

14. Octav Brudaru, Rüdiger Ebendt, Iulian Furdu(2010) [23]

In this paper a new double hybridized genetic algorithm for optimizing the variable order in Reduced Ordered Binary Decision Diagrams has been presented. The first hybridization adopts embryonic chromosomes as prefixes of variable orders instead of complete variable orders and the branch & bound technique is embedded with the basic genetic algorithm. The second hybridization is done with the existing sifting algorithm, known as one of the most effective heuristic for this problem, which is incorporated as a hypermutation operator.

15. Thangavel Bhuvaneswari,Vishnuvajjula Prasad, Ajay Kumar Singh, Chinnaiyan Senthilpari (2011) [25]

In this paper, reversed BDD-based pass transistor logic (PTL) logic synthesis has been presented for low-power and high-performance circuits without exploiting the canonical property of BDDs. Binary decision diagrams (BDDs) are the most frequently used data structure for the representation and handling of Boolean functions because of their excellent time and space efficiencies. The procedure of the reversed BDD transformation into PTL has been achieved by a one-to-one correspondence with the BDD node and PTL cell. Layouts have been generated for the benchmark circuits and simulated in terms of power dissipation, propagation delay and area. The reversed BDD technique has been witnessed to perform better in terms of area, delay and power dissipation due to the regularity, a reduced critical path, less interconnection wires, a multiplexer-based construction of PTL circuits, and less switching activities.

16. Saurabh Chaudhury, Anirban Dutta (2011) [24]

In this paper, authors have proposed two algorithms: a genetic algorithm and a branch and bound algorithm to find an optimal input variable order. The node reordering was taken care of by the standard BDD package buddy-2.4. They have evaluated the performances

of the proposed algorithms by running an exhaustive search program. Experimental results showed a substantial saving in area and power. They also compared their techniques with other state-of-art techniques of variable ordering for OBDDs and found to give superior results.

17. David Bergman, Andre A. Cire, Willem-Jan Van Hoeve, J. N. Hooker (2012) [28]

In this paper the idea of obtaining bounds on the optimal value of an optimization problem from a discrete relaxation based on binary decision diagrams (BDDs) has been presented. The formation of a BDD that represents a relaxation of an optimization problem with binary variables has been shown. Also how to obtain a bound for any separable objective function by solving a shortest (or longest) path problem in the BDD has been proposed.

18. Harsh Arora, Arindam Banerjee, Rahul Rupchand Jidge (2014) [29]

Logic synthesis is used in VLSI Design to convert technology independent high level description of a complex electronic circuit into optimized gate level netlist. In Boolean algebraic factorisation, a logic expression is considered as polynomials. The conventional methods such as truth table, K-map, SOP/POS forms yield satisfactory results for the Boolean functions comprises of AND/OR expressions. But, optimality of Boolean factorisation for AND/OR/XOR intensive functions is degraded. In this paper, a minimisation algorithm employing data structure to form the basis for synthesis engine has been presented. A heuristic approach has been proposed to derive proper ordering of input variables for BDD tree with minimum computation to reduce the space complexity of the circuit, thereby enhancing the performance.

Chapter-3
Evolutionary Algorithms

In artificial intelligence, an evolutionary algorithm (EA) is a subset of evolutionary computation, a generic population-based meta heuristic optimization algorithm. An EA uses mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection. Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness function determines the environment within which the solutions "live". Some of the evolutionary algorithms are discussed below.

3.1 Genetic Algorithm (GA)

Genetic Algorithms GA, [4] are a family of excellent optimization techniques based on principle of natural selection and human genetics. Genetic Algorithm is based on Darwinian Theory of survival of the fittest. The objective of GA is to find an optimal solution to a problem. GAs work by evolving a population of individuals over a number of generations .A fitness value is assigned to each individual depending on the application. For each generation, individuals are selected from existing population for reproduction, the individuals are crossed to generate new individuals, and the new individuals are mutated to introduce new characteristics. The solution with best fitness value will be the most optimum solution. The formulation of Genetic Algorithm for any problem involves the careful and efficient encoding of the solutions to form chromosomes, crossover and mutation operators and a cost function measuring the fitness of the chromosomes in a population. GAs use two basic processes from evolution: inheritance, or passing of features from one generation to other, and competition, or the survival of the fittest, which results in weeding out bad features from individuals in the population. This process is continued for a large number of iterations to obtain a best-fit (near-optimum) solution.

3.1.1 Basic Working Principle

The major steps involved are the generation of a population of solutions, finding the objective function and fitness function and the application of genetic operators-

1. Formulate initial population
2. Randomly initialize population
3. Repeat
4. Evaluate objective function
5. Find fitness function
6. Apply genetic operators
7. Reproduction
8. Crossover
9. Mutation
10. Until stopping criteria

GA begins with a set of solutions, represented by chromosomes, called the population. This population represents the complete search space of all feasible solutions to a given problem. These solutions from one population are taken and used to form a new population of offspring which are better than their parents. The selection of parents is done according to their fitness function and more fit solutions are given more chance to reproduce. The whole process is repeated until some termination condition is achieved which may be the maximum number of generations or improvement of the best solution.

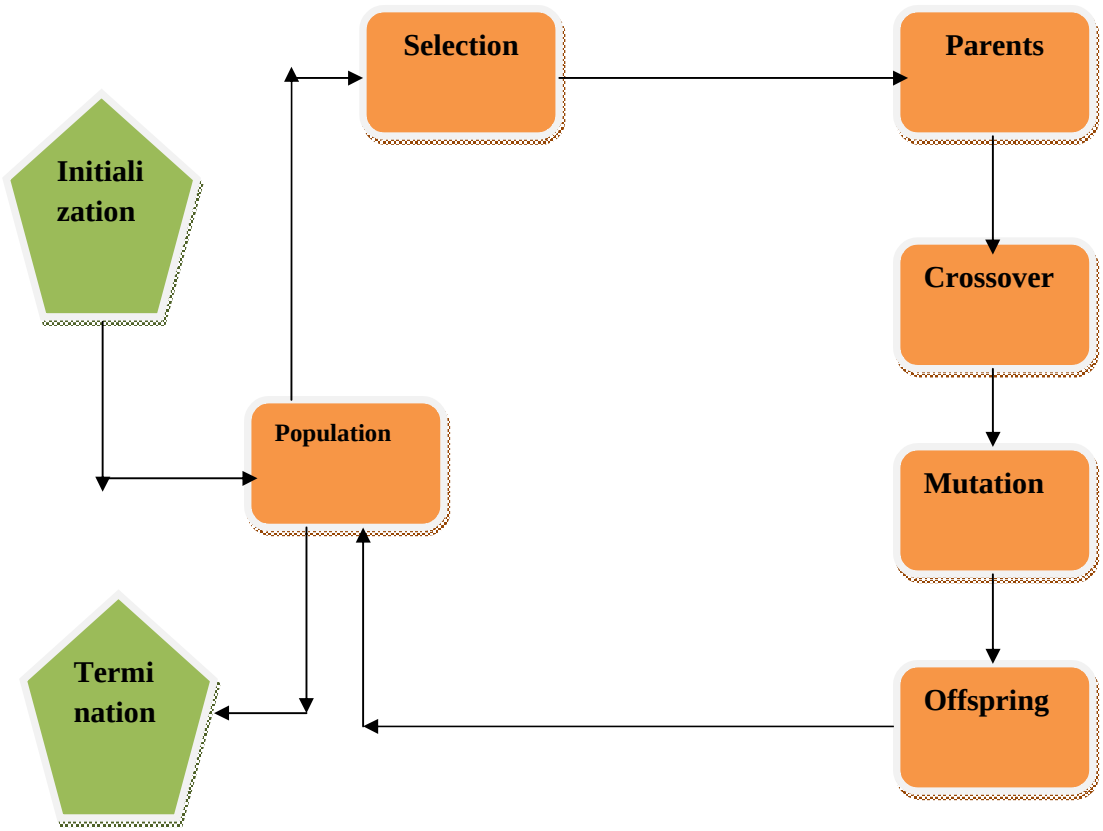


Figure 3.1 Operation Flow of Genetic Algorithm [4]

3.1.2 Flowchart for BDD Minimization Using Genetic Algorithm

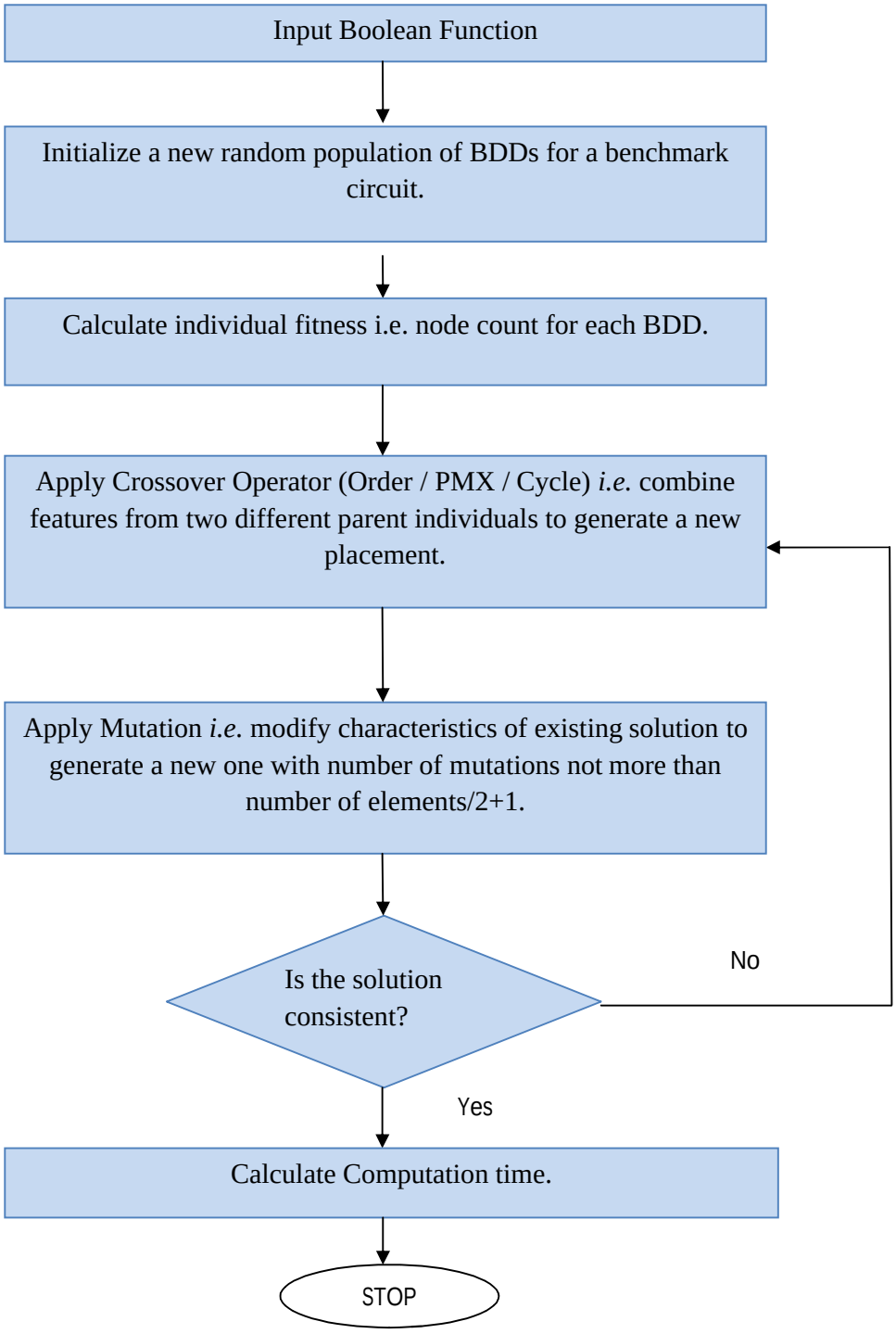


Figure 3.2 Flow-chart of the Genetic Algorithm based technique used for BDD Variable Ordering [24]

3.1.3 Genetic Operators

There are two types of operators that are applied in the genetic algorithm to form a new solution (individual) from the existing solutions (parents):-

3.1.3.1. Crossover Operator

It is performed between two selected individuals, called parents, by exchanging parts of their features (i.e. encodings) to form two new individual, called offspring.

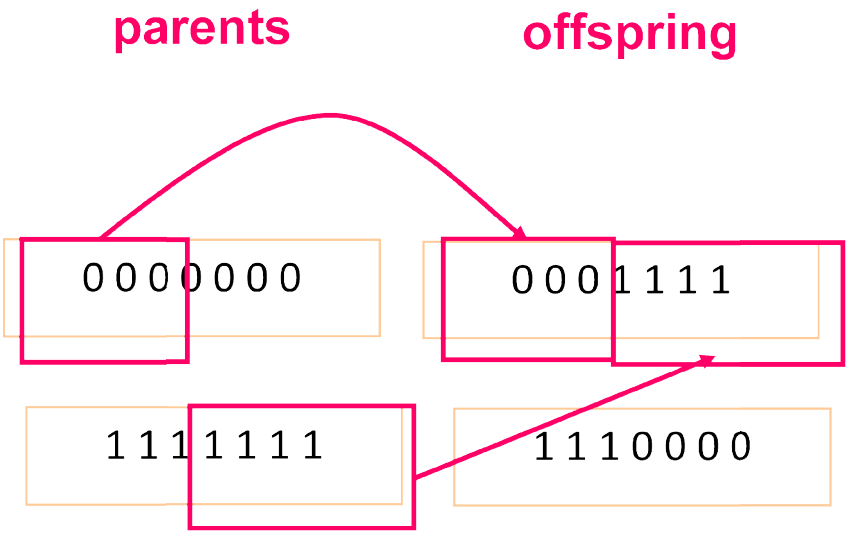


Figure 3.3 Crossover Operator [30]

3.1.3.2. Mutation operator

It brings variety into population by selecting a chromosome randomly from the population and modifies the chromosome at a point depending upon the value of a generated random number.

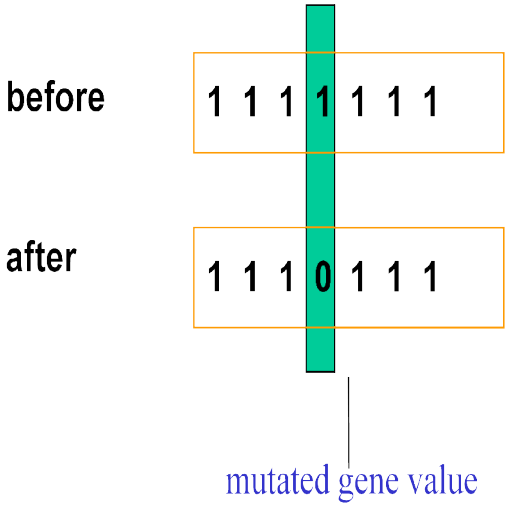


Figure 3.4 Mutation Operator [30]

3.1.4 Application Areas of Genetic Algorithm

GA is useful and efficient when:

- The search space is large, complex or poorly understood
- Domain knowledge is scarce or expert knowledge is difficult to encode to narrow the search space
- No mathematical analysis is available
- Traditional search methods fail

3.2 Hybridized Genetic Algorithm (HGA)

An Evolutionary algorithm can be hybridized with other evolutionary or dynamic algorithms to find optimized results. In Hybridized Genetic Algorithm [23], The Genetic Algorithm is embedded with Branch and Bound Algorithm to intelligently search for the best solution of an optimization problem. The objective of Hybridized Genetic Algorithm is to find an optimal ordering of BDDs with reduced node count and computation time. It works by finding a best individual i.e. BDD ordering from a population of individuals evolved over many generations. A fitness value *i.e.* in terms of node count is assigned to each individual *i.e.* BDD order. A population of random individuals is initialized .Parents are selected based on their fitness values. Branch and Bound Algorithm discards fruitless solutions by using upper and lower bounds on fitness value. Crossover and mutation operations are performed on them to generate offspring with a better fitness value i.e. node count. The process is continued till an optimum solution i.e. BDD with least node count is found.

3.2.1 Basic Working Principle

The major steps involved are the generation of a population of solutions, finding the objective function and fitness function, application of branch and bound algorithm and genetic operators-

1. Formulate initial population
2. Randomly initialize population
3. Repeat
4. Evaluate objective function
5. Find fitness function
6. Branch and Bound
7. Reproduction
8. Crossover
9. Mutation
10. Until stopping criteria

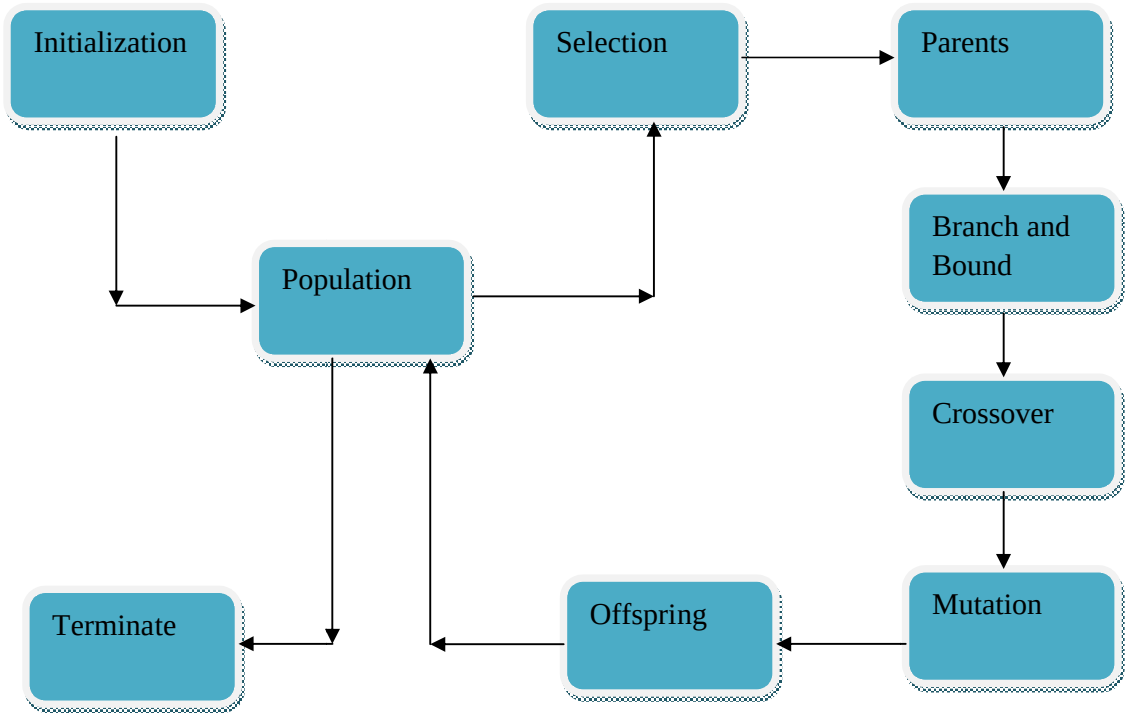


Figure 3.5 Operation Flow of Hybridized Genetic Algorithm.

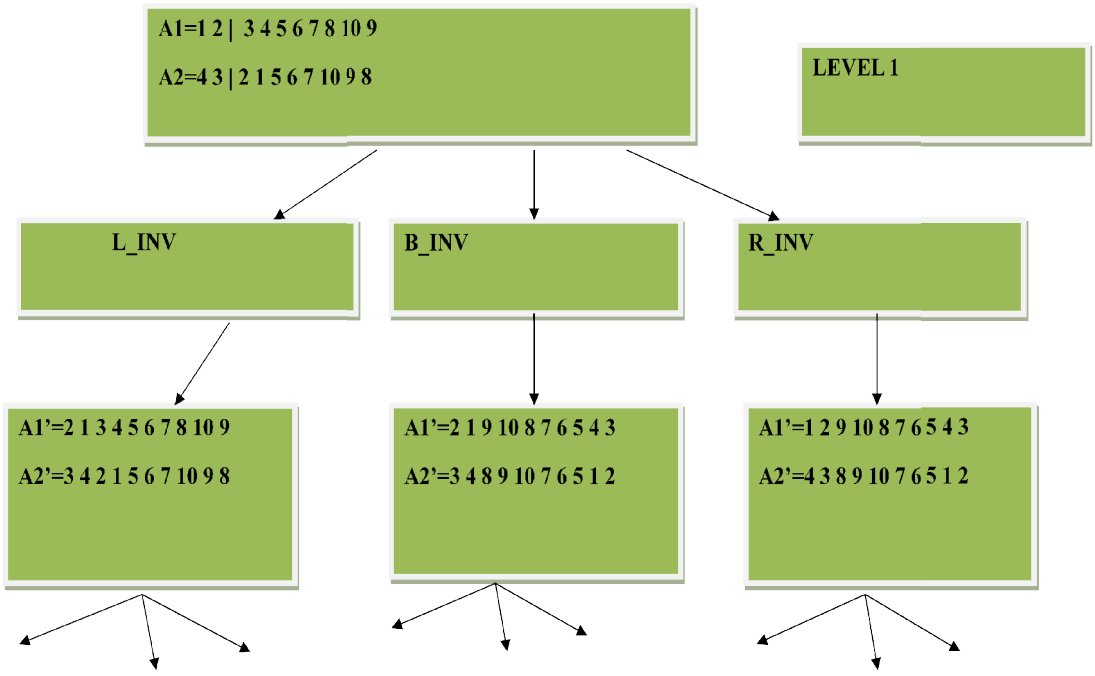


Figure 3.6 Branching scheme in Branch and Bound Algorithm [24].

3.2.2 Flowchart for BDD Minimization Using Hybridized Genetic Algorithm

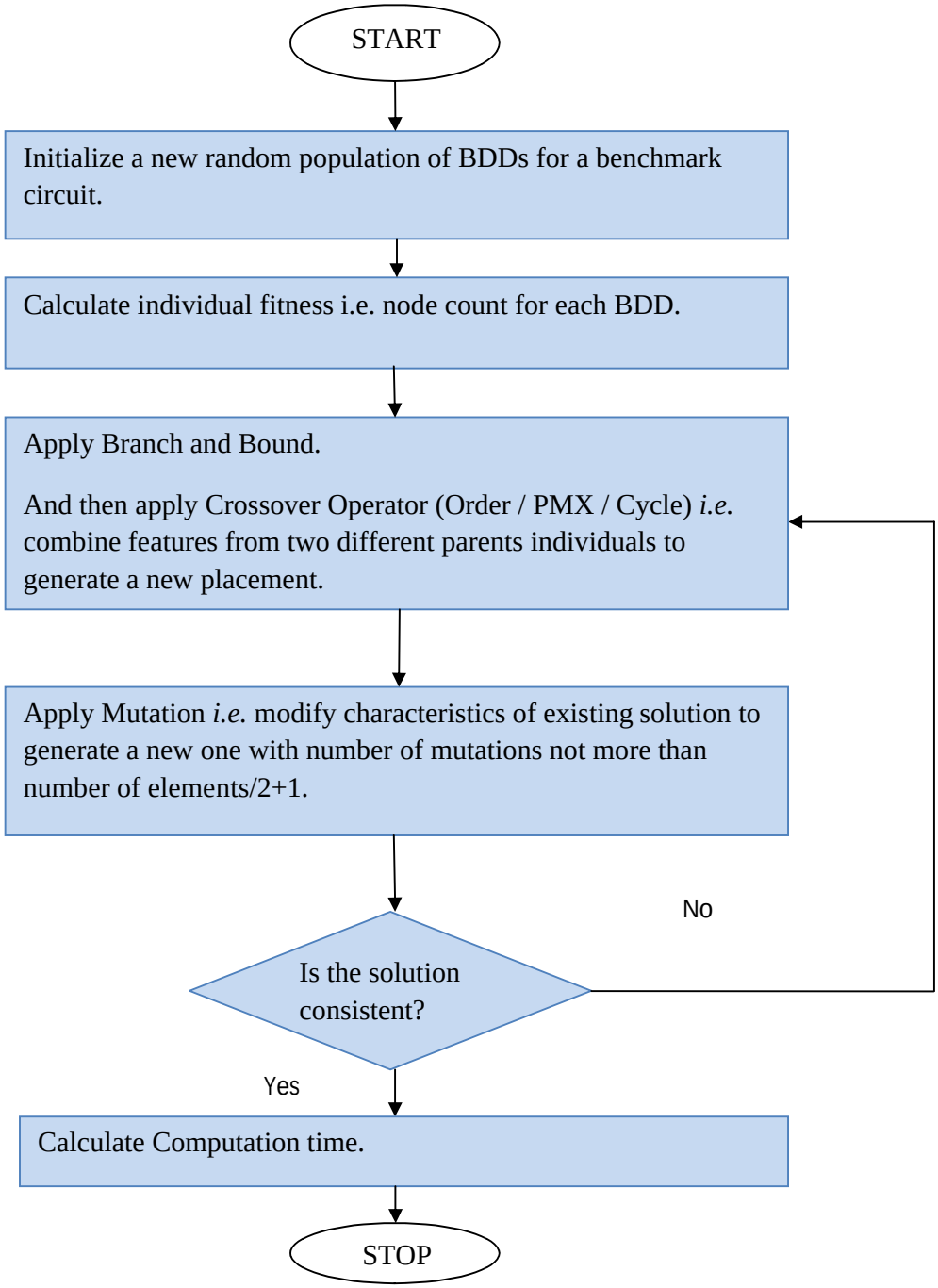


Figure 3.7 Flow-chart of the Hybridized Genetic Algorithm based technique used for BDD Variable Ordering [23].

Chapter-4
Crossover Operators

4.1 Types of Crossover Operators

The performance of the genetic and the hybridized genetic algorithm depends, to a great extent, on the performance of the crossover operator used [30]. As mentioned earlier, crossover is the primary method of optimization in these algorithms, and works by combining sub-placements from two different parent configurations to generate a new placement. There are three types of Crossover Operators namely Order Crossover, Cycle Crossover and Partially Mapped Crossover [30].

4.1.1 Order Crossover

This operator conveys a sub-placement from the first parent without any changes, and then, to resolve conflicts, compresses the second parent by eliminating the cells conveyed by the first parent and shifting the rest of the cells to the left without changing their order. It then copies this compressed second part into the remaining part of the offspring array.

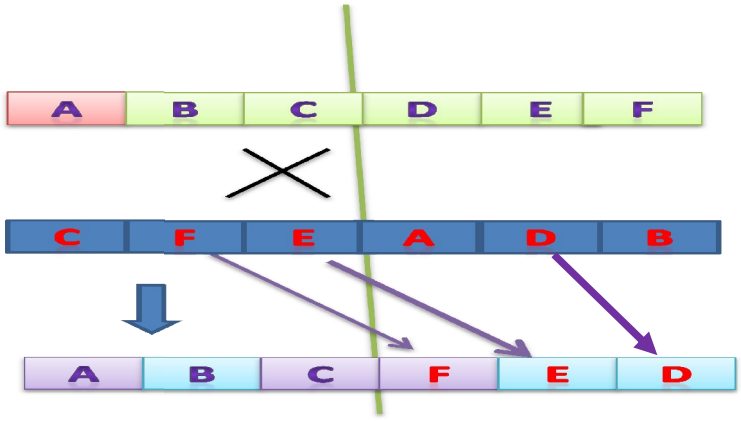


Figure 4.1 Order Crossover Operator.

In figure 4.1, there are two existing solutions, in terms of order, of a binary decision diagram. To apply order crossover, the crossover point is cut after three elements. The elements to the right of crossover point from first parent are copied to the corresponding

positions of the offspring i.e. A, B, C. The remaining positions in the offspring are filled by the elements of the second parent, in order, leaving those which have been already copied from first parent i.e. F, E, D.

4.1.2 Cycle Crossover

In the offspring generated by the cycle crossover, every cell is in the same location as in one parent or the other. The basic idea behind cycle crossover is to avoid cell conflicts by finding non-overlapping sets of cells to pass from the two parents.

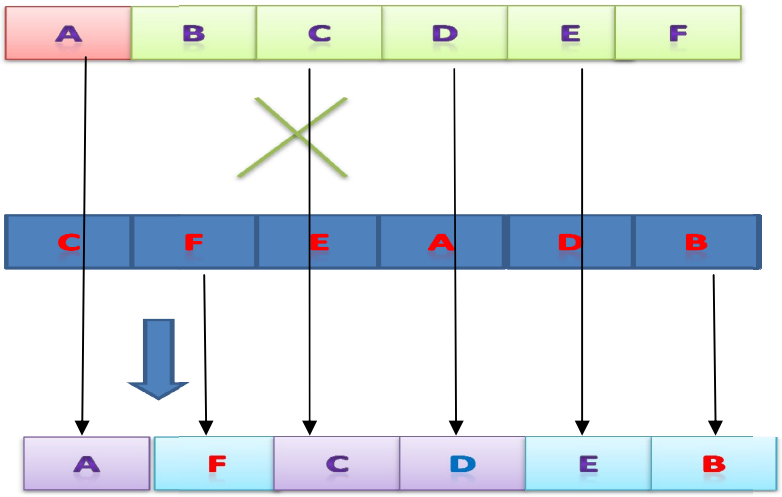


Figure 4.2 Cycle Crossover Operator.

In figure 4.2, there are two existing solutions, in terms of order, of a binary decision diagram. To apply Cycle crossover, the first element of first parent i.e. A is copied to same position in offspring. Therefore, the first element of second parent i.e. C cannot be copied to offspring. So C is copied from the third position of first parent to the same position of the offspring. Similarly, D and E are copied from the first parent to the corresponding positions of the offspring. This completes one cycle. The second cycle is then started from the second parent. The last element from second parent is copied to the same position of the offspring i.e. B occupies the last position of the offspring. Therefore, the last element of first parent i.e. F cannot be copied to offspring. So F is copied from the second position of second parent to the same position of the offspring. Since all the positions in the offspring have been occupied, therefore, the second cycle has also been completed.

4.1.3 Partially Mapped Crossover

A cell in the segment of the first parent and a cell in the same location in the second parent define which cells in the first parent have to be exchanged to generate the offspring.

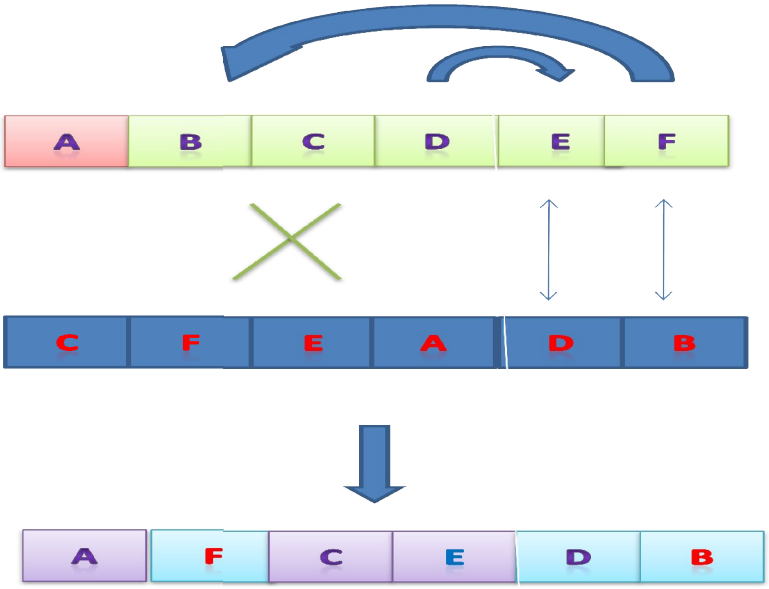
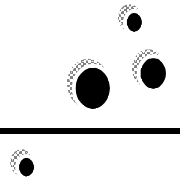


Figure 4.3 PMX Crossover Operator.

In figure 4.3, there are two existing solutions, in terms of order, of a binary decision diagram. To apply Partially Mapped Crossover, the crossover point is cut after four elements. The elements to the left of the crossover point of first parent are partially mapped with corresponding elements of the second parent i.e. E is partially mapped with D and F is partially mapped with B. Therefore, in the first parent E is exchanged with D and F is exchanged with B to form a new individual i.e. A, F, C, E, D, B.

Chapter-5
Optimization of Power



In VLSI Design, the silicon area and the expected performance are accurately estimated before the circuit goes into fabrication. However, with the requirements of low power, the designers are faced with the added burden of qualifying their designs with respect to power dissipation. Hence, work has started in earnest to accurately estimate power dissipation at different levels. Earlier, the major factors of the VLSI designer were area, cost, performance and reliability, power considerations were only of secondary importance. But now, this has been changed and power is being given equal importance in comparison to area and performance. Lots of factors have contributed to this trend. This is mainly due to the remarkable and huge success of personal computing devices which demands very high speed computations and complex functionality with low power consumption.

5.1 Reduction of power at different Abstraction Levels

Limits discussed so far have been fundamental since they do not depend on the technology or the choice of power supply voltage. However, there a number of obstacles or technological limitations are in the way to approaching these limits in practical circuits and ways to reduce the effect of these various limitations could be found at all levels of digital design ranging from device to system. Battery powered systems such as laptop, computers, electronics devices etc. the need of these systems arise from the need to extend battery life. Low power design is not only needed for portable applications but also to reduce the power of high performance systems with large integration density and improved speed of operation.

Emerging of portable computing and communication equipment such as laptop, palmtops, cell-phones, etc. growth rate of this portable equipment are very high. Following are different levels of abstraction:

- **System level**- The system is described in terms of a set of hardware, software and memory components and interacting algorithms they perform to provide certain functionality.
- **Behavioral or architectural level**- The individual components such as ASICs, data paths, software processes are described in the terms of their algorithmic behavior, typically in a hardware description languages, such as behavioral VHDL, or a high-level programming language such as C. Typically, there is no timing, but only causal information is available on this level and the interface protocols controlling the communication between the interacting processes has been already fixed.
- **Register-transfer level**- In this, hardware is described in terms of arithmetic modules, registers and multiplexers and interconnects to steer data flow. RT level representations describes the basic building blocks of a system and their interconnects on clock-cycle level; it specifies the "micro-architecture" of the system under consideration.
  - **Gate-level**- A circuit is described in terms of a net list of gates or a set of Boolean equations reflecting its functionality.
  - **Transistor level**- A circuit is described in the terms of its transistor network structure.
  - **Physical level**- A circuit is described in terms of its mask layout as it will be used for fabrication. The basic building blocks on this level are polygons which reflect different materials used for fabrication such as metal, polysilicon, oxide etc.

At system level, the first decision to be made on system level is that of algorithm selection, i.e. selecting, from a set of given algorithms, one that provides the required functionality of the system under design while best meeting design constraints. The power consumption induced by an algorithm depends on its characteristics in terms of overall complexity, complexity of the basic operations, communication requirements etc. Although the selected algorithm itself can later be optimized, subsequent optimizations are typically of a local scope and will not be able to compensate the differences in power consumption across different algorithms, which can be up to several orders of magnitude. One of the most important factors to be taken into consideration during algorithm selection is that of memory access and management.

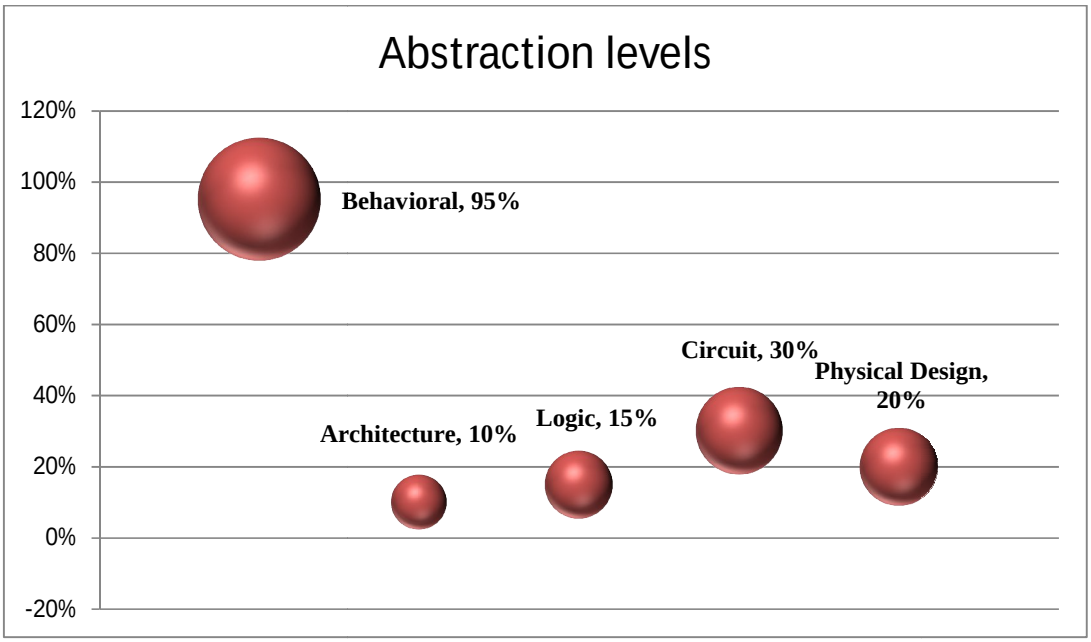


Figure 5.1 Different abstraction levels to reduce power dissipation.

Behavioral level power optimization is performed during behavioral synthesis, i.e. when mapping an algorithm onto a hardware register-transfer level description. We can distinguish between optimizations which operate on the behavioral description alone and those which optimize the mapping of operations onto hardware units. Optimizations in register-transfer level operate by structurally transforming RT level net lists. There is a general consensus that design decisions on higher levels of abstraction have the most significant impact on the overall structure and quality of the resulting design. For instance, choosing different partitioning of functionality on design components on the system level can result in vastly different requirements on the characteristics of the individual components in terms of area, performance or power. Accordingly, optimizing transformations on higher levels of abstraction have the largest impact on the power consumption of the design as a whole. It is therefore desirable to take power into account as a cost function as early as possible in the design flow, i.e. on system and behavioral level, in the design flow, since in the later stages optimizations will typically only be of a very local scope, and hence of limited effectiveness.

5.2 Types of Power Dissipation

It has been noted that the power dissipation in CMOS circuits is mainly due to dynamic (switching and short circuit) current and the sub threshold leakage current. With the scaling down of device sizes and transistor threshold, the sub-threshold current will

increase and can become a sizable component of total power dissipation. However, in the current-day technology, the dominant component power is still the switching component (about 80% of the total power dissipation) as shown in the figure. Therefore, we will devote a considerable effort in estimation of switching power in VLSI circuits.

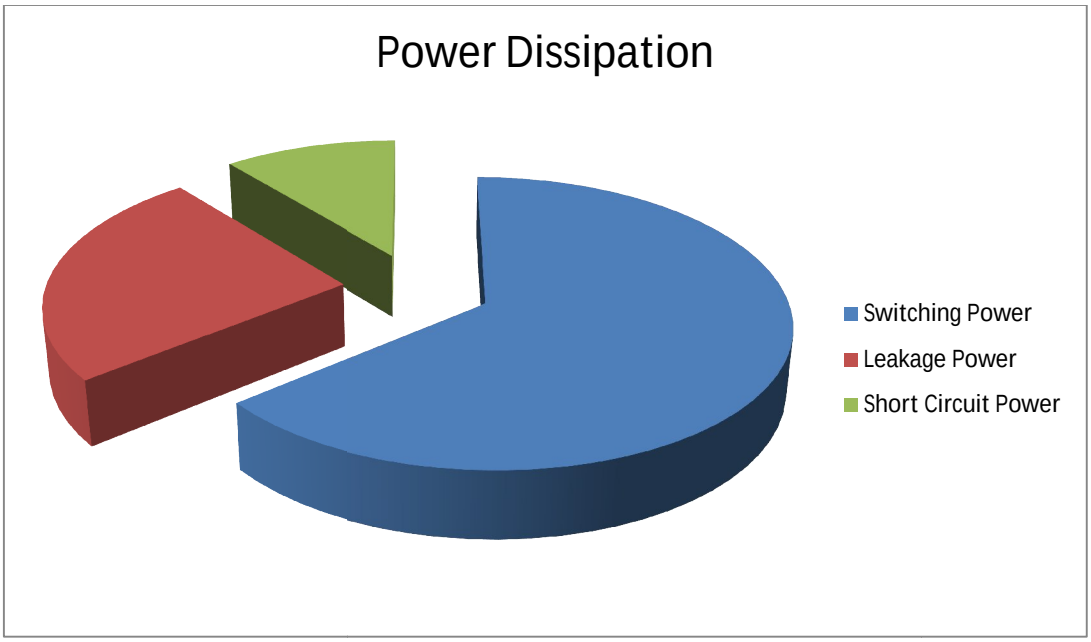


Figure 5.2 Types of power dissipations.

5.3 Probabilistic technique for Switching Power Optimization

The importance of low-power optimization is growing due to the increased use of battery-powered embedded systems. In order to optimize for low power dissipation, statistical information about the behavior of the system can be exploited. The switching activity of a circuit node in a CMOS digital circuit directly contributes to the overall dynamic power dissipation [7]. Temporal correlation of the occurring input signals can have a significant effect on the switching activity and hence the power consumption. The power dissipation estimate for a mapped BDD node is based on its switching activity and its fan out (corresponding to the capacitive load).

5.3.1 Switching activity estimation

To calculate the switching activity, assumption is made that the input signals are mutually independent (spatially uncorrelated) and that the signals can be modeled as strict-sense stationary (SSS) and mean-ergodic with zero delay which means that all switching is

carried out simultaneously, and signal probability values and switching activity do not vary over time.

P (f) denotes the probability that f is 1 (the output probability of f), and a (f) denotes the activity for f (the probability that f will change in value from one cycle to the next).

In order to devise an improved low-power synthesis method for BDD mapped circuits [6], an accurate and computationally efficient switching activity estimation method is needed that is able to utilize temporal correlation. To avoid the high computational complexity of an exact method, it is assumed that there is no spatial correlation between the Shannon cofactors of the function of interest. The approximation technique provides the exact result for the case where the cofactors are spatially uncorrelated. In the case where cofactors are positively correlated an overestimate is obtained, since the switching of a top variable is less prone to cause a true switching of the node's output. The opposite holds for negatively correlated cofactors. This observation allows for the application of Theorem 3.1 from [5]. The formula for the estimation of switching activity is derived using the multiplexer-based circuit model [7]:

a(f)=[[((P(f0)+P(f1)-2P(f0)P(f1))/(1-(a(f0)+a(f1)-a(f0)a(f1)))) - (((1/2)(a(f0)+a(f1)-a(f0)a(f1)))/(1-(a(f0)+a(f1)-a(f0)a(f1)))]a(v)(1-a(f0)(1-a(f1))] + [((1-P(v)-(a(v)/2))/(1-a(v)))a(f0)(1-a(v))(1-a(f1))] + [((P(v)-(a(v)/2))/(1-a(v)))a(f1)(1-a(v))(1-a(f0))] + ((1/2)a(v)a(f0)(1-a(f1)))+((1/2)a(v)a(f1)(1-a(f0))) + (a(f1)a(f0)(1-a(v))) + ((1/2)a(v)a(f0)a(f1)) **(1)**

In (1), v is the input variable, f0 is the low cofactor, and f1 is the high cofactor. This formula is used recursively in a bottom-up approach to calculate the activity for each node in the BDD.

5.3.2 Low-power Estimation

The power dissipation of each node of the binary decision diagram is computed by the estimated switching activity and the fan out of the node.

5.3.2.1 BDD mapped circuits

A BDD is directly mapped to a multiplexer-based circuit and the resulting circuit is obtained by replacing BDD vertices with small sub circuits and BDD edges with wires. The diagram size (and therefore the circuit complexity) is sensitive to the ordering of the

function variables. The complexity varies from linear to exponential under different orderings for some functions. The power dissipation of each node is computed by the estimated switching activity and the fan out of the node. The variable order of the underlying BDD influences not only the area (number of nodes) but also the internal switching activity. The technique is also used with BDDs using complemented edges. The use of complemented edges is shown to both reduce BDD complexity and improve performance of operations.

1. The output probability, P (f^{-}), of f^{-} is equal to 1-P (f).

2. The switching activity, a (f^{-}), of f^{-} is equal to a (f).

These properties are used to compute local switching probabilities during variable exchange operations on BDDs with complemented edges. A cost model based on the total circuit switching activity under a given set of dependent-variable output probabilities is defined. The dependent variables are denoted as support variables. The sum of all internal switching activities at each BDD vertex is minimized. Each BDD node is then mapped into a multiplexer-based circuit. The number of stages of active buffers is determined by the fan out of each BDD node, which is equivalent to the number of BDD edges pointing to the node. The power dissipation for the mapped node PD_n is estimated using the relationship in (2) [7]:

PDn = a (n) x driver (fan out (n)) + leakage (n) **(2)**

In (2), the power dissipation due to leakage is ignored as its contribution to total power dissipation is very less; therefore, leakage (n) is ignored.

Chapter-6
Analysis and Methodologies

- The Genetic Algorithm and the Hybridized Genetic Algorithm, using three types of Crossover Operators, are implemented with C++ codes and simulated using BUDDY 2.4 package on Ubuntu 12.04.
- The available approaches for BDD ordering and node minimization are compared along with the proposed three versions of crossover operator of the Genetic and Hybridized Genetic Algorithm, namely, order, cycle and PMX crossover for BDD minimization on the set of Boolean functions.
- The proposed approaches give minimal nodes for every function when compared with all other approaches.
- The power dissipation is calculated with the help of estimation of switching activity using Probabilistic Technique in Genetic and Hybridized Genetic Algorithms.
- The simulation results for various Multi-input Adder circuits have been displayed in Tables 7.1, 7.2, 7.3, 7.4, 7.6 and 7.7.
- The proposed algorithms and approaches provide the optimum variable order with minimum number of nodes, less computation time and low power dissipation in minimum iterations and number of runs for every function.

Chapter-7
Simulation Results

7.1 Estimation of Node Count and Computation Time

The Proposed Genetic Algorithm and Hybridized Genetic Algorithm, using three types of crossover operators, are implemented with C++ codes and simulated using BUDDY 2.4 package on Ubuntu 12.04. Simulation results *i.e.* node count and computation time for Multi-input Adder Circuits have been presented in Tables 7.1, 7.2, 7.3 and 7.4.

Table 7.1 Comparison of Node Count for Dynamic and Genetic algorithms for Multi-Input Adders Circuits.

Bench mark Circuits	I/P	O/P	Initial Node Count	WIN 2	WIN 2ite	WIN 3	SIFT	RANDOM	Proposed GA with Crossover		
									ORDER	CYCLE	PMX
1-adder	3	2	**8**	8	8	8	8	8	**8**	**8**	**8**
2-adder	5	3	**17**	17	17	17	17	17	**17**	**17**	**17**
3-adder	7	4	**32**	32	34	32	26	34	**25**	**32**	**26**
4-adder	9	5	**63**	65	63	46	46	61	**35**	**45**	**35**
5-adder	11	6	**126**	128	126	61	55	78	**49**	**82**	**44**
6-adder	13	7	**253**	255	253	85	85	93	**79**	**132**	**58**
7-adder	15	8	**508**	510	508	94	94	264	**135**	**165**	**72**
8-adder	17	9	**1027**	1001	1001	87	283	845	**558**	**396**	**267**

Table 7.2 Comparison of Computation Time for Different Crossover Operators in Genetic algorithm for Multi-input Adder Circuits.

Benchmark Circuits	Inputs	Outputs	Proposed GA with crossover		
			Order(sec)	Cycle (sec)	PMX (sec)
1-adder	3	2	**0.01**	**0.01**	**0.01**
2-adder	5	3	**0.02**	**0.02**	**0.03**
3-adder	7	4	**0.1**	**0.17**	**0.22**
4-adder	9	5	**0.2**	**0.31**	**0.52**
5-adder	11	6	**0.42**	**0.62**	**1.78**
6-adder	13	7	**1.11**	**1.47**	**5.94**
7-adder	15	8	**2.67**	**3.86**	**8.38**
8-adder	17	9	**3.91**	**3.42**	**13.5**

Table 7.3 Comparison of Node Count for Dynamic and Hybridized Genetic algorithms for Multi-Input Adders Circuits.

Benchmark Circuits	Inputs	Outputs	Initial Node Count	WIN2	WIN2 ite	WIN 3	SIFT	RANDOM	Proposed GA with Crossover		
									ORDER	CYCLE	PMX
1-adder	3	2	**8**	8	8	8	8	8	**8**	**8**	**8**
2-adder	5	3	**17**	17	17	17	17	17	**17**	**17**	**17**
3-adder	7	4	**32**	32	34	32	26	34	**26**	**26**	**26**
4-adder	9	5	**63**	65	63	46	46	61	**35**	**54**	**35**
5-adder	11	6	**126**	128	126	61	55	78	**58**	**86**	**50**
6-adder	13	7	**253**	255	253	85	85	93	**96**	**160**	**69**
7-adder	15	8	**508**	510	508	94	94	264	**154**	**229**	**92**
8-adder	17	9	**1027**	1001	1001	87	283	845	**262**	**321**	**183**

Table 7.4 Comparison of Computation Time for Different Crossover Operators in Hybridized Genetic algorithm for Multi-input Adder Circuits.

Benchmark Circuits	Inputs	Outputs	Proposed GA with crossover		
			Order(sec)	Cycle (sec)	PMX (sec)
1-adder	3	2	**0.01**	**0.02**	**0.03**
2-adder	5	3	**0.02**	**0.02**	**0.03**
3-adder	7	4	**0.05**	**0.06**	**0.1**
4-adder	9	5	**0.21**	**0.13**	**0.38**
5-adder	11	6	**0.73**	**0.38**	**1.11**
6-adder	13	7	**1.57**	**1.17**	**2.8**
7-adder	15	8	**3.2**	**2.29**	**4.73**
8-adder	17	9	**4.79**	**4.13**	**10.99**

7.2 Estimation of Power Dissipation

In BDD, each node is represented as a single 2x1 multiplexer. After synthesization of a 2x1 multiplexer in Synopsys tool, the power calculated is 762.3125 nW as shown in Table 7.9

Table 7.5 Report for Power calculation using Synopsys tool

****

Report : power -

analysis_effort low

Design : mux_2 Version:

Y-2006.06-SP4

Date : Thu May 29 12:18:01 2014

****

Library(s) Used:

fsa0a_c_generic_core_tt1p8v25c(File:/cad/DigitalFDKs/faraday180nm/core180nm/fsa0a_c/

2009Q2v2.0/GENERIC_CORE/FrontEnd/synopsys/fsa0a_c_generic_core_tt1p8v25c.db)
Operating Conditions: TCCOM Library: fsa0a_c_generic_core_tt1p8v25c

Wire Load Model Mode: enclosed
Design Wire Load Model Library

---

mux_2 G5K fsa0a_c_generic_core_tt1p8v25c
Global Operating Voltage = 1.8 Power-specific
unit information:

Voltage Units = 1V
Capacitance Units =
1.000000pf Time Units = 1ns

Dynamic Power Units = 1mW (derived from V,C,T units)
Leakage Power Units = Unit less

Cell Internal Power = 765.1176 nW (87%)

Net Switching Power = 97.1948 nW (13%)

Total Dynamic Power = 762.3125 nW (100%)

***** End Of Report **

Total power dissipation of the circuit can be calculated by multiplying the power calculated for a BDD (represented as 2x1 Multiplexer) with the node count. It can be represented as below

Total power = node count * 762.3125nw (1)

Results have been calculated for Multi-input Adder circuits using different Crossover Operators in Genetic and Hybridized Genetic Algorithms. Table 7.6 performs comparison of the Power calculated using Probabilistic techniques [7] with the already existing algorithm (values calculated using Synopsys tool).The results indicate that in more than 90% cases, Genetic and Hybridized Genetic algorithm based approach using probabilistic analysis has shown lesser power dissipation.

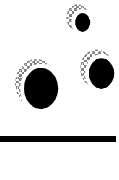
Table 7.6 Comparison of the Power calculated using Probabilistic technique and existing technique (values calculated using Synopsys tool) in Genetic Algorithm for Multi input Adder Circuits.

Bench-mark Circuits	I/P	O/P	Power for Order Crossover Operator(in mW)		Power for Cycle Crossover Operator(in mW)		Power for PMX Crossover Operator(in mW)	
			Proposed Probabilistic Technique	Existing MUX based Technique	Proposed Probabilistic Technique	Existing MUX based Technique	Proposed Probabilistic Technique	Existing MUX based Technique
1-adder	3	2	**0.0001198**	0.006098	**0.0001198**	0.006098	**0.0001198**	0.006098
2-adder	5	3	**0.003361**	0.012959	**0.003361**	0.012959	**0.003361**	0.012959
3-adder	7	4	**0.01661**	0.019057	**0.00463**	0.024394	**0.004661**	0.019820
4-adder	9	5	**0.002301**	0.026680	**0.002387**	0.034304	**0.003211**	0.026680
5-adder	11	6	**0.01611**	0.037353	**0.027512**	0.062509	**0.0254987**	0.033541
6-adder	13	7	**0.021876**	0.060222	**0.000134**	0.100625	**0.000231**	0.044214
7-adder	15	8	**0.006341**	0.102912	**0.00812**	0.125781	**0.001223**	0.054886
8-adder	17	9	**0.009543**	0.425370	**0.01231**	0.301875	**0.000124**	0.203537

Table 7.7 Comparison of the Power calculated using Probabilistic technique and the existing technique (values calculated using Synopsys tool) in Hybridized Genetic Algorithm for Multi-input Adder Circuits.

Bench-mark Circuits	I/P	O/P	Power for Order Crossover Operator(in mW)		Power for Cycle Crossover Operator(in mW)		Power for PMX Crossover Operator(in mW)	
			Proposed Probabilistic Technique	Existing Technique	Proposed Probabilistic Technique	Existing Technique	Proposed Probabilistic Technique	Existing Technique
1-adder	3	2	**1.57471**	0.006098	**1.57471**	0.006098	**1.57471**	0.006098
2-adder	5	3	**0.0062263**	0.012959	**0.0062263**	0.012959	**0.0062263**	0.012959
3-adder	7	4	**0.012530**	0.019820	**0.012530**	0.019820	**0.012530**	0.019820
4-adder	9	5	**0.0004063**	0.026680	**0.000006384**	0.041164	**0.000006384**	0.026680
5-adder	11	6	**0.00662216**	0.044214	**0.005556**	0.065558	0.0036543	0.038115
6-adder	13	7	**0.0265436**	0.073182	0.0000194	0.121970	0.00000343	0.052599
7-adder	15	8	**0.00001031**	0.117396	0.0000465	0.174569	0.0000239	0.070132
8-adder	17	9	**0.00000196**	0.199725	0.00003613	0.244702	0.00000269	0.139503

Chapter-8
Conclusions and Future Scope



8.1 Conclusions

- The proposed algorithms and approaches provide the optimum variable order with minimum number of nodes in minimum iterations and number of runs for every function.
- It is observed that in comparison with other existing algorithms, the proposed Genetic Algorithm gives minimum node count. Also, when the operators are compared, Partially Mapped Crossover gives minimum nodes, while the Cycle Crossover gives minimum computation time for most of the circuits. The order crossover also gives optimum results. In multi-input adder circuits, as number of inputs are increasing, the proposed genetic algorithm with PMX crossover gives best reduction in node count, where as with cycle crossover it gives best reduction in computation time. From Table 7.1, for 1-adder, with crossover reduction in node count is about 0%, for 2-adder is about 0%, for 3-adder is about 18.75%, for 4-adder is about 44.44%, for 5-adder is about 65.07%, for 6-adder is about 77.07%, for 7-adder is about 85.82% and for 8-adder is about 74%. Therefore, the proposed Genetic algorithm is suitable for multi-input multi-output (MIMO) VLSI circuits.
- It is also observed that in comparison with other existing Dynamic algorithms, the proposed Hybridized Genetic Algorithm also gives minimum node count. Also, when the operators are compared, Partially Mapped Crossover gives minimum nodes, while the Cycle Crossover gives minimum computation time for most of the circuits. The order crossover also gives optimum results. From Table 7.3, for 1-adder, with crossover reduction in node count is about 0%, for 2-adder is about 0%, for 3-adder is about 18.75%, for 4-adder is about 44.44%, for 5-adder is about 60.31%, for 6-adder is about 72.72%, for 7-adder is about 81.88% and for 8-adder is about 82.18%.Therefore,the proposed Hybridized Genetic algorithm is suitable for multi-input multi-output (MIMO) VLSI circuits.

- From tables 7.2 and 7.4, it is observed that the computation time for Order and Cycle Crossover Operators is less than that of PMX Crossover Operator in case of Genetic and Hybridized Genetic Algorithms for more than 70% of multi-input benchmark circuits.
- In case of the estimation of power dissipation, it is observed that that in more than 90% cases, Genetic and Hybridized algorithm based approach using probabilistic analysis has shown lesser power dissipation than the Multiplexer based existing technique. From tables 7.6 and 7.7, it can be concluded that the power dissipation in case of PMX and Order Crossover Operators is lesser than Cycle Crossover Operator in case of both Genetic and Hybridized Algorithms for 80% benchmark circuits.
- Therefore, it is concluded that the Evolutionary algorithms namely Genetic Algorithm and Hybridized Genetic Algorithm are suited for the optimization of area, speed and power in VLSI circuits.

8.2 Future Work

- Various other evolutionary algorithms can also be implemented on the benchmark circuits for the optimization of area, power and speed in VLSI Design.
- The other dynamic and evolutionary algorithms can be used to implement BDD based probabilistic technique for less power dissipation.
- Apart from achieving low area, low power and high speed, memory requirements can be looked upon to achieve better performance.

8.3 Analysis of Results with Histograms

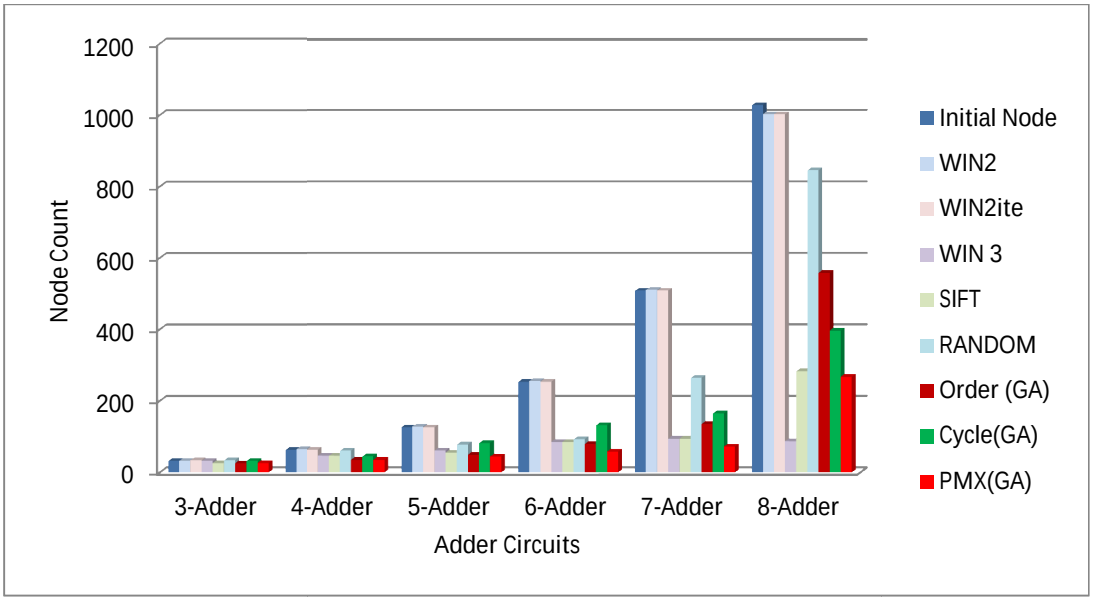


Figure 8.1 Comparison of Node Count for multi-input adder circuits using Dynamic Algorithms and Various Crossover Operators in Genetic Algorithm.

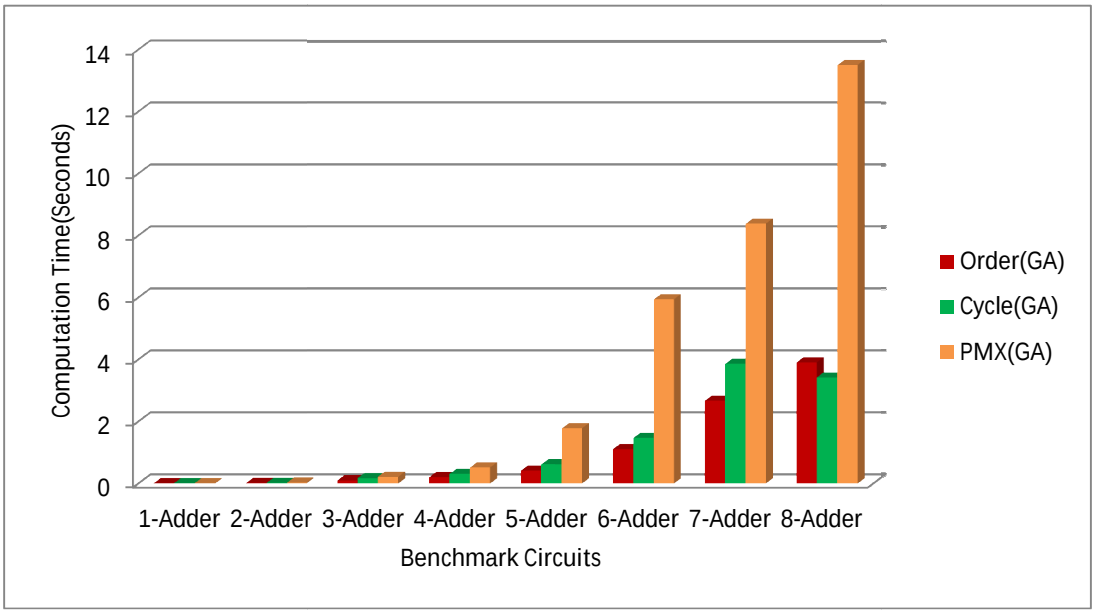


Figure 8.2 Comparison of Computation time for multi-input adder circuits using different Crossover Operators in Genetic Algorithm.

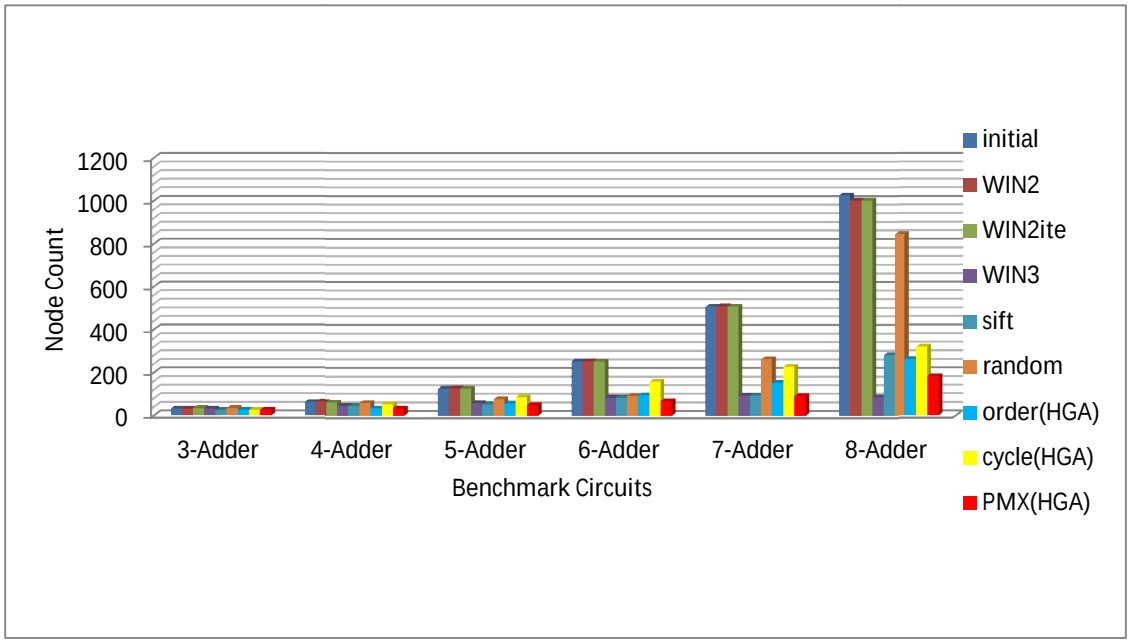


Figure 8.3 Comparison of Node Count for multi-input adder circuits using Dynamic Algorithms and Various Crossover Operators in Hybridized Genetic Algorithm.

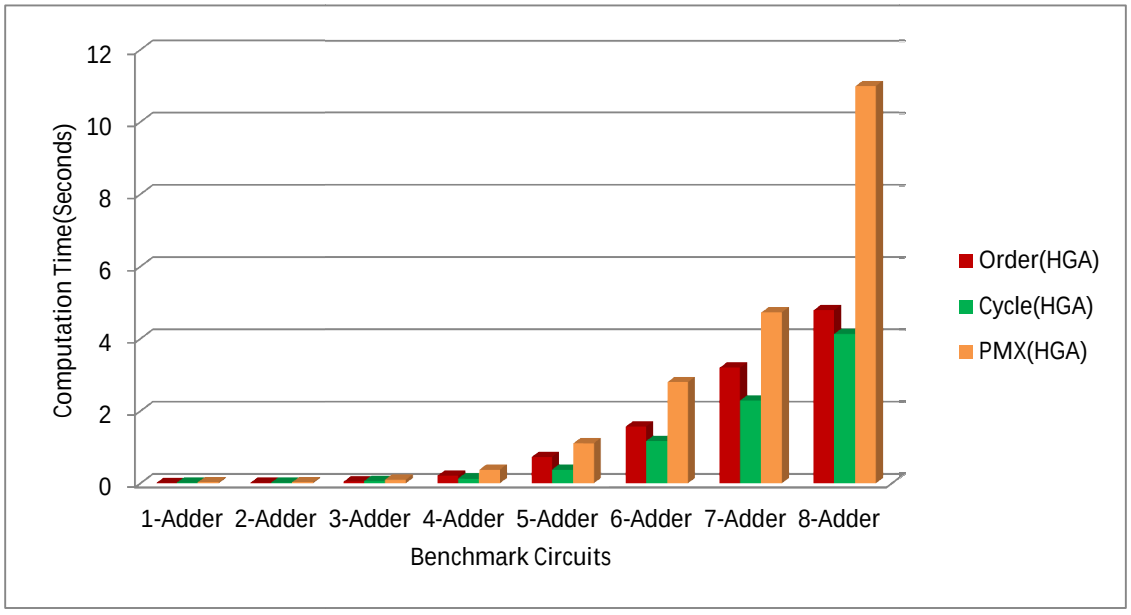


Figure 8.4 Comparison of Computation time for multi-input adder circuits using different Crossover Operators in Hybridized Genetic Algorithm.

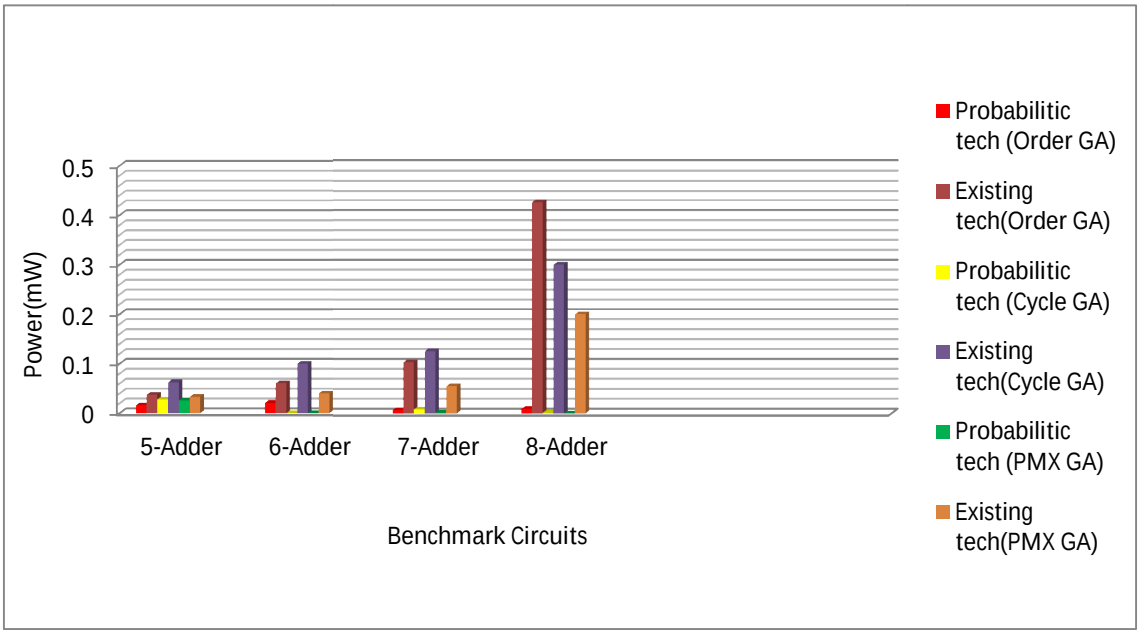


Figure 8.5 Comparison of the power calculated Using Probabilistic Technique and Existing Technique in Genetic Algorithm for Multi-input Adder Circuits.

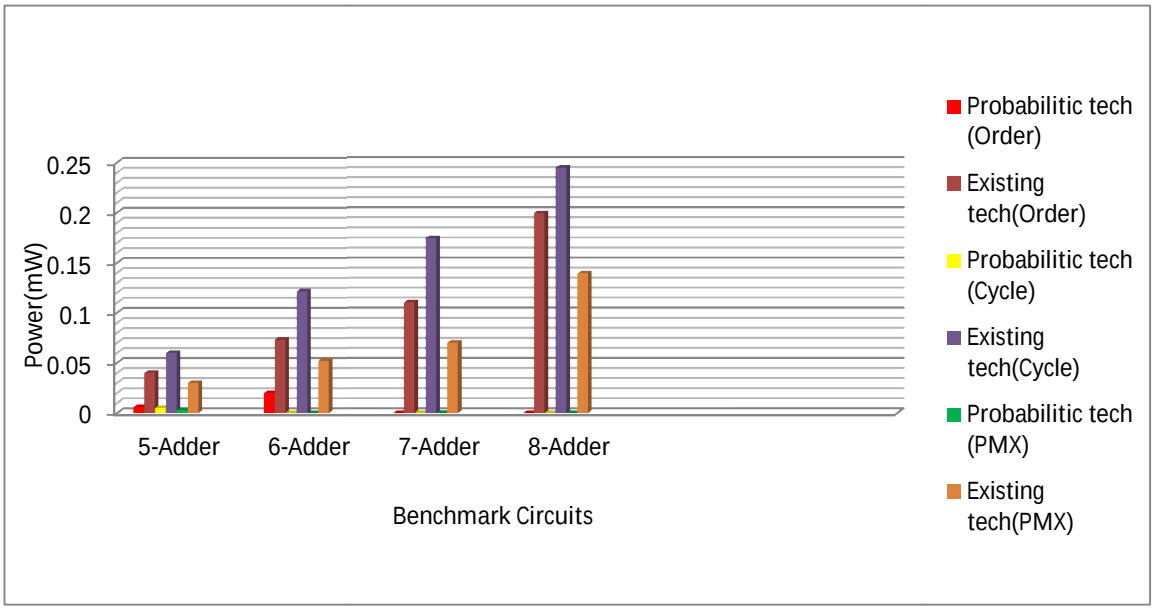


Figure 8.6 Comparison of the power calculated Using Probabilistic Technique and Existing Technique in Hybridized GA for Multi-input Adder Circuits.

LIST OF PUBLICATIONS

1. "**Ordering and Reduction of BDDs Using Various Crossover Operators in GA**" in 3rd National Conference on Advances in Metrology (Ad Met- 2014).The paper got the best paper award.

2. "**BDD Ordering and Minimization Using Various Crossover Operators in Genetic Algorithm**" in International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering, Vol. 2, Issue 3, pp. 1247-1250, March 2014.

References

[1] Randal E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation", *IEEE Transactions on Computers*, C-35(8), pp. 677–691, 1986.

[2] Steven J. Friedman, Kenneth J.Supowit, "Finding the optimal Variable ordering for Binary Decision Diagrams", *IEEE Transaction on Computers*, vol 39, no.3, 1990.

[3] Nagisa Ishiura, Hiroshi Sawada, Shuzo Yajima, "Minimization of Binary Decision Diagrams Based on Exchanges of Variables", *IEEE,* 1991.

[4] N. Zhuang, M.S.T. Benten, and P.Y.K Cheung, "Improved Variable Ordering of BDDs with Novel Genetic Algorithm," *IEEE International Symposium on Circuits and Systems*, vol. 3 pp. 12-15, 1996.

[5] K. Roy and S. Prasad, Low-Power CMOS VLSI Circuit Design, Hoboken, N.J.: Wiley Interscience, 2000.

[6] P. Lindgren, M. Kerttu, M. Thornton, and R. Drechsler, "Low power optimization technique for BDD mapped circuits" , *Proc. ASP Design Automation Conf.*, pp. 615–621, 2001.

[7] Rolf Drechsler, Mikael Kerttu, Per Lindgren, Mitchell Thornton, "Low-power optimization techniques for BDD mapped circuits using temporal correlation", *Can. J. Elect. Comput. Eng.*, Vol. 27, No. 4, 2002.

[8] Shaul Markovitch, Orna Grumberg, Shlomi Livne, "Learning to Order BDD Variables in Verification", *Journal of Artificial Intelligence Research 18*, pp. 83 -116, 2003.

[9] Shun-Shii Lin,Chun-Jen Wei, "A new approach for minimization of binary decision diagrams" , *Can. J. Elect. Comput. Eng*., Vol. 30, No. 4, 2005.

[10] Wang Mingquan,Yu Haibin, "BDD Minimization Based on Genetic Tabu Hybrid Strategy",*IEEE*, 2005.

[11] R. Ebendt, F. Gorschwin, R. Drechsler, "Advanced BDD Minimization", *Springer*, 2005.

[12] Gorschwin Fey, Rolf Drechsler, “Minimizing the Number of Paths in BDDs: Theory and Algorithm”, *IEEE Transactions on computer-aided design of integrated circuits and systems*, Vol 25, No.1, 2006.

[13] Piotr Porwik, Krzysztof Wrobel, Piotr Zaczkowski, “Some practical remarks about Binary Decision Diagram size reduction” , *IEICE Electronics Express*, Vol. 3,No.3 pp. 51-57,2006.

[14] P. W. C. Prasad, A. Assi, A. Harb, V. C. Prasad, “Binary Decision Diagrams: An Improved Variable Ordering using Graph Representation of Boolean Functions,” *International Journal of Computer Science*, Vol. 1, No. 1 pp. 1-7,2006.

[15] Du Su-guo, Sun Yan, “A Novel Ordering Method of Binary Decision Diagram”, 14th International Conference on Management Science & Engineering, 2007.

[16] Osnat Keren, “Reduction of the Average Path Length in Binary Decision Diagrams by Spectral Methods”, *IEEE Transactions on Computers*, Vol. 57, No. 4, 2008.

[17] Raheleh Sadat Hosseini,Farnaz Towhidi,Arash Habibi Lashkari, “Binary Decision Diagram (BDD)” ,*International Conference on Future Computer and Communication*,2009.

[18] F. Towhidi, A.H. Lashkari, R.S. Hosseini, “Binary Decision Diagram (BDD)”, *International Conference on Future Computer and Communication*, pp. 496-499, 2009.

[19] Hossein Moeinzadeh, Mehdi Mohammadi, Hossein Pazhoumand-dar, Arman Mehrbakhs,Navid Kheibar, Nasser Mozayani, “Evolutionary Reduced Ordered Binary Decision Diagram”, *Third Asia International Conference on Modelling & Simulation* ,pp. 142-145 ,2009.

[20] Awinash Kumar, Ajit Kumar, Soumik choudhary and P.Y. Yarde, “Optimization of Binary Decision Diagram Using Genetic Algorithm” , *2nd International Conference on Reliability, Safety and Hazard (ICRESH),* pp. 168-175, 2010.

[21] Saurabh Chaudhury, Anirban Dutta, “Genetic algorithm based variable ordering of BDDs for multilevel logic optimization with area power tradeoffs”, *17th IEEE International Conference on Electronics, Circuits, and Systems (ICECS),* pp. 627-630, 2010.

[22] Misagh Takapoo, M.B Ghaznavi-Ghoushchi, “IDGBDD: The novel use of ID3 to improve Genetic algorithm in BDD reordering”, *International Conference on*

Electrical Engineering / Electronic Computer Telecommunication and Information Technology, pp. 19-21,2010.

[23] Octav Brudaru, Rüdiger Ebendt, Iulian Furdu, "Optimizing Variable Ordering of BDDs with Double Hybridized Embryonic Genetic Algorithm" , *12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*,2010.

[24] Saurabh Chaudhury, Anirban Dutta, "Algorithmic Optimization of BDDs and Performance Evaluation for Multi-level Logic Circuits with Area and Power Trade-offs", *Circuits and systems*, pp. 217-224, 2011.

[25] Thangavel Bhuvaneswari, Vishnuvajjula Prasad, Ajay Kumar Singh, Chinnaiyan Senthilpari, "Performance Analysis of Reversed Binary Decision Diagram Pass Transistor Logic Synthesis", *International Journal of Circuit Theory and Applications*, 2011.

[26] T. Tsuchiya, "A BDD approach to reliability optimal module allocation in networks", *2012 IEEE 18th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pp. 121-126, 2012.

[27] Tom V Mathew, "Genetic Algorithm," Report submitted at IIT Bombay.

[28] David Bergman, Andre A. Cire, Willem-Jan van Hoeve, "Optimization Bounds from Binary Decision Diagrams", *INFORMS Journal on Computing Optimization Bounds from Binary Decision Diagrams*,2012.

[29] Harsh Arora, Arindam Banerjee, Rahul Roopchand Jidge, "Heuristic Approach to Variable Ordering for Logic Synthesis Engine Design: Algorithmic Insight", *Interna tional Journal of Circuits and Architectural Design*, Vol.1, 2014.

[30] Pinaki Mazumder, Elizabeth M. Rudnick, Genetic Algorithms for VLSI Design,Layout & Test Automation.