IMPROVEMENT IN PERFORMANCE PARAMETERS USING HEURISTIC ALGORITHM FOR VLSI CIRCUITS

Dissertation submitted in partial fulfillment of the requirements

for the award of the degree of

Master of Technology

in

VLSI Design

Submitted by:

SURBHI JINDAL

Roll No: 601461028

Under the supervision of

Mrs. Manu Bansal

Assistant Professor, ECED

Department of Electronics and Communication Engineering

Thapar University, Patiala, Punjab, INDIA

2016

DECLARATION & CERTIFICATE

I hereby declare that the work presented in the dissertation entitled "**Improvement in Performance Parameters using Heuristic Algorithm for VLSI Circuits**" in partial fulfillment of the requirement for the award of the degree of the Master of Technology in 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.

Surbhi Jindal

Date: 11.07.2016

SURBHI JINDAL
Roll No: 601461028

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

Manu Bansal

Mrs. Manu Bansal
Assistant Professor
Electronics and Communication Department
Thapar University, Patiala

Countersigned by:

Professor & Head
ECED, Thapar University
Patiala -147004

Dean of Academic Affairs
Thapar University
Patiala - 147004

ACKNOWLEDGEMENT

I take this opportunity to express my profound sense of gratitude and respect to all those who helped me throughout 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 her valuable guidance and support throughout the dissertation work.

I convey my sincere thanks to Head of the Department, **Dr. Sanjay Sharma,** PG Coordinator, VLSI Design, **Dr. Anil Arora**, as well as Lecturer, **Mr. Gagandeep Singh**, ECED, entire faculty and staff of Electronics and Communication Engineering Department for their encouragement and cooperation.

I am also thankful to my friends, especially **Rupinder Kaur** and **Raj Shah**, for being very supportive throughout the work.

My greatest thanks go 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.

SURBHI JINDAL

ABSTRACT

A Binary Decision Diagram (BDD) is an effective data structure based on recursive Shannon Expansion extensively used to obtain and implement any Boolean function and obtain a canonical representation and an efficient variable ordering for getting an optimized version. Ordering of BDDs plays a significant role in the total node count and hence, the total used area, and with that, the average computation time and storage requirement. BDDs have been broadly used in Computer Aided Design for the optimum logic synthesis and also in formal verification and testing of Digital Circuits.

Genetic Algorithm based approach, based on the process of natural evolution, is one of the well known approaches in Evolutionary Computation Algorithms. This approach is dependent on the performance of the employed Crossover technique using three crossover operators – order, cycle and partially mapped. It plays an important role in optimization of shared ordered BDDs and yields highly improved results in comparison with the already existing Sifting, Window and Random algorithms. The results obtained have been further improved by using the Hybrid Genetic technique by hybridizing the optimized Genetic approach with Branch and Bound algorithm.

The optimization of power consumption holds an equal importance in the performance metrics now. Depending on the switching activity of a node in a CMOS digital circuit, the overall dynamic power dissipation gets varied. The estimated power dissipation of a BDD mapped circuit is based on the switching activity and fan out (corresponding to capacitive load) of its nodes.

The efficiency of the proposed algorithm has been tested on International Workshop on Logic Synthesis (IWLS), IWLS'93 combinational benchmark circuits. Experimental results show that the proposed approach significantly outperforms the already existing Sifting, Window and Random algorithms in terms of substantial reduction in node count and hence, area, in a limited CPU time as well as the probabilistic power.

TABLE OF CONTENTS

TABLE OF CONTENTS (…cont.)

LIST OF ACRONYMS

ACRONYM	MEANING
VLSI	Very Large Scale Integration
CAD	Computer Aided Design
IWLS	International Workshop on Logic Synthesis
BDT	Binary Decision Tree
BDD	Binary Decision Diagram
DAG	Directed Acyclic Graph
OBDD	Ordered Binary Decision Diagram
ROBDD	Reduced Ordered Binary Decision Diagram
HA	Heuristic Algorithm
GA	Genetic Algorithm
HGA	Hybridized Genetic Algorithm
PMX	Partially Mapped Crossover

LIST OF FIGURES

LIST OF TABLES

Chapter 1. Introduction

1.1. BINARY DECISION DIAGRAM

A Binary Decision Diagram (BDD) is a data structure widely used for the compact representation of Boolean functions [1]. For representation of Boolean functions, the use of BDDs was first proposed in [2] and developed to the form of the Reduced Ordered BDDs (ROBDDs) [3]. The variable ordering employed for a BDD has a significant effect on the total node count [3] and hence, the overall area. The technique of BDD has come out to be one of the most effective styles of Boolean function representation and implementation [4]. BDDs can be directly transformed into circuits by substituting every node of the underlying graph with a multiplexer [5]. The reason for this transformation lies in the Shannon decomposition [6].

The concept of Boolean function that defines a digital circuit can be represented as a BDD which is a directed acyclic graph (DAG) [7] with a compact data structure [8] as per a particular order and satisfying a defined set of properties. It is usually based on the recursive Shannon Expansion [8] for the switching function f, on which the graph based representation of BDD relies, based on the decomposition of f [8] around each variable y_i as is described by:

$$f = y_i'.f\big|_{(x=0)} + y_i.f'\big|_{(x=1)} \tag{1}$$

Where x is a finite set of Boolean variables, $x = \{x_1, x_2 \ldots x_n\}$ and y_i', f' denote the complements of y_i, f respectively.

Therefore, a BDD is represented by a binary tree consisting of non-terminal nodes (decision nodes) and terminal nodes (0-Terminal and 1-Terminal) which are connected by the directed edges. The convention for the terminal-0 directed edge is a dotted line and for a terminal-1 directed edge is a solid line. The term DAG is given to BDD as there is no way to come back from the same edge.

Each decision node has two child nodes – Low child (representing assignment of 0) and High child (representing assignment of 1). A BDD for the Boolean function ‘f = ac + bc’ has been shown in Fig 1.1.

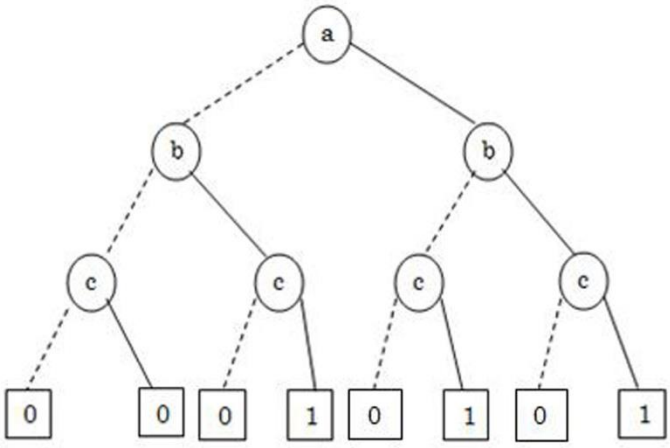

Fig. 1.1. BDD for Boolean function f = ac + bc

A BDD can be implemented either in canonical form, ‘ordered in any order’ form (OBDD) or in reduced ordered form (ROBDD).

1.1.1. Ordered Binary Decision Diagrams

The size of a BDD strongly depends on the order of input variables. It may happen that the size of a BDD is getting increased exponentially with the circuit size using one variable ordering, and in a linear fashion using another variable ordering.

An Ordered Binary Decision Diagram (OBDD) is a BDD obtained by making all the different variables appear in the same sequence on all the paths starting from the root node of the BDD. The choice of variables for this ordering is a crucial step as it will determine the overall size and all the specifications such as area, speed and power associated with it. Fig. 1.1 is an OBDD as it encounters same set of variables in the same order (or sequence) on all the paths, when visited from the start node of the BDD.

It can be seen that there are many redundant paths which can be omitted, hence leading to the overall reduction in the size of the OBDD. This leads to the concept of ROBDDs.

1.1.2. Reduced Ordered Binary Decision Diagrams

An OBDD gets converted into a Reduced Ordered Binary Decision Diagram (ROBDD) when all duplicate terminals and redundant nodes are eliminated, identical nodes are shared and all duplicate nodes, if any, are merged together [3]. Fig. 1.2 shows the equivalent ROBDD for the OBDD shown in Fig. 1.1. ROBDD is almost always referred by BDD in the literature. The major advantage offered by the ROBDD is the canonicity besides being ordered.

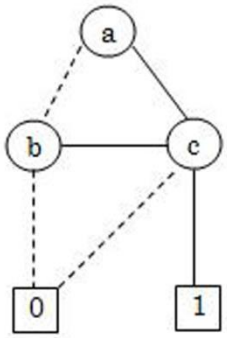

Fig. 1.2. ROBDD for Boolean function f = ac + bc

This reduction to obtain ROBDD is based upon some rules. These are as follows:

- *Eliminate Duplicate Nodes:*
  If number of terminal-0 nodes is more than 1, then direct all edges to 0-node to just one of them. Similarly do for terminal-1 nodes.
- *Eliminate Redundant Nodes:*
  Remove the node with both of its directed edges pointing to the same node.
- *Merge Duplicate Nodes:*
  Nodes have to be unique. They must not repeat.

The reduction rules are shown in Fig. 1.3.

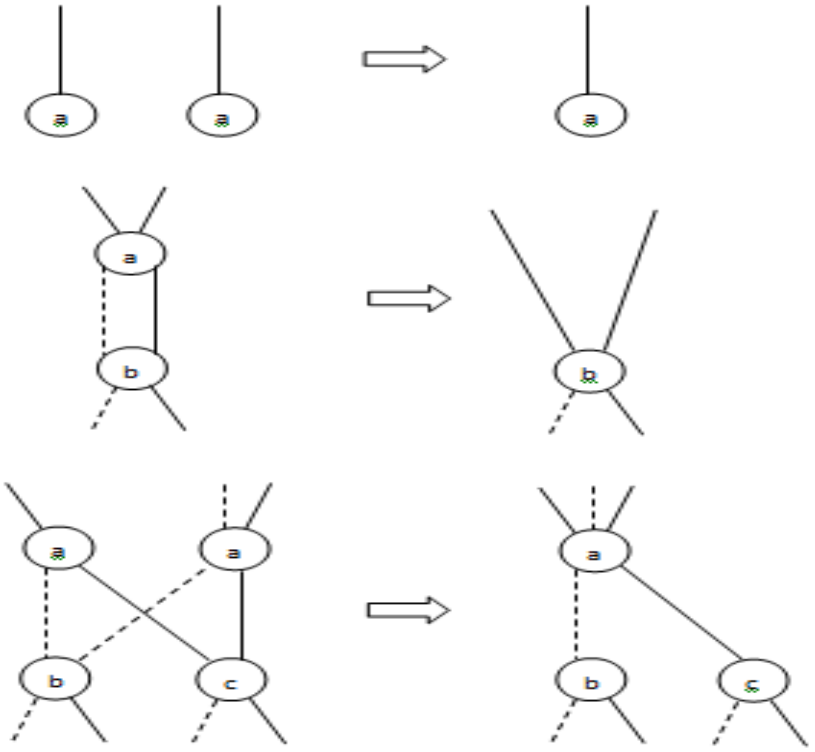

Fig. 1.3. Reduction rules for BDD reduction

1.1.3. Illustration of Binary Decision Diagram

For the Boolean function, firstly, we start the process by making the Binary Decision Tree (BDT) showing the complete design and results view. The BDT for the Boolean function f = ac + bc has been shown in Fig. 1.1. As it can be seen that there are 2^n terminal nodes in BDD which is the same as the 2^n terminal nodes in a truth table, it can be concluded that BDD has not offered much advantage till now. So, we move on to making the primitive BDD as shown in Fig. 1.4.

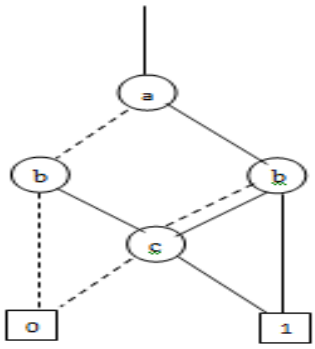

Fig. 1.4. Primitive BDD for the Boolean function f = ac + bc

The ROBDD can be extracted by using the reduction rules on this OBDD of Fig. 1.4. The resultant ROBDD has been shown in Fig. 1.2. The reduction in the overall size can be easily seen from here. The total node count has been reduced from 15 to 5.

Another example showing the truth table, OBDD and the corresponding ROBDD for the Boolean function f = ab + c has been shown in Fig. 1.5.

a	b	c	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

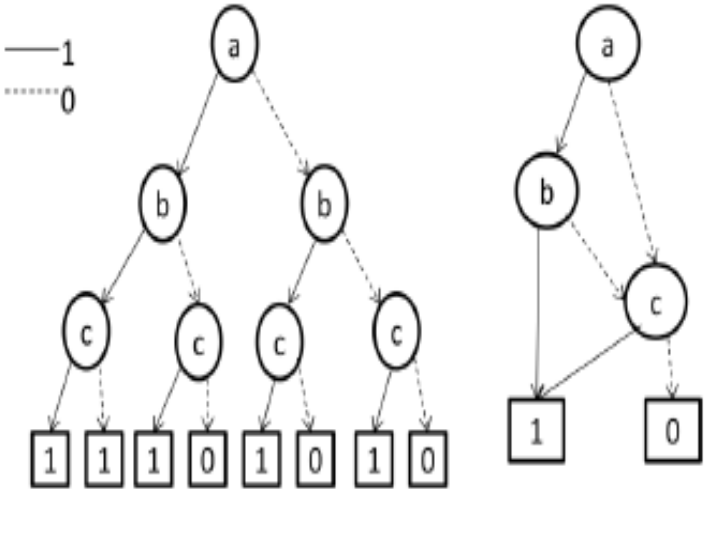

Fig. 1.5. Truth table, OBDD and ROBDD for the Boolean function, f = ab + c

1.1.4. Applications of Binary Decision Diagrams

BDDs have been extensively used in:

- Compact representation of Boolean functions.
- Test for absence of redundant variables.
- Test for semantic equivalence because an ROBDD is the canonical form for a function.
- Test for Implication.
- Test for Satisfiability.
- In logic synthesis, logic verification, optimization, fault simulation and test-pattern *generation* for Digital systems [9, 10].
- The theory behind this concept is that synthesis tools make use of BDDs to solve many of the problems occurring in VLSI Computer-Aided Design.

Every BDD (either reduced or any ordered) can be directly implemented in hardware by replacing every node in the BDD by a 2:1 multiplexer. The advantage of this becomes clear when it is seen that each multiplexer can be directly implemented by a 4-LUT in a FPGA.

1.1.5. Significance of Variable Ordering

The size of a BDD strongly depends on the order of input variables. Moreover, a function can have multiple ROBDDs [11] depending on the ordering of variables. It may happen that the size of a BDD is getting increased exponentially with the circuit size using one variable ordering, and in a linear fashion using another variable ordering. Since the memory requirement, area and evaluation time increase with the increasing size of BDD, it is desirable to keep the size of the BDD as small as possible. Actually, the size of the BDD most of the times determines the performance of all the specifications linked to it – be it area, speed, memory or power.

1.2. NEED OF OPTIMIZATION

Nowadays, as the chip dimensions are reducing, there is a significant need to be able to integrate more and more components and functionality onto a single chip. This requires more efficient usage of available chip area with considerable consideration to the power dissipations.

The optimization is of utmost importance in today's era of giant scales of integration – be it single-level optimization or multi-level optimization. In today's world, optimization forms the future in spite of the fact that multi-objective optimization is not always possible with all the objectives achieved simultaneously. Sometimes there is a need to prioritize the objectives or the other times, just take the weighted aggregate of the considerable objectives. So, there is a need to develop mathematical and computational optimization methods to improve the efficiency of the processes.

1.3. USAGE OF ALGORITHMS FOR OPTIMIZATION

Many heuristics [12-15] and algorithms [11, 16-25] have been worked upon to determine the close-to-optimal solution for the optimization of BDDs by improving the variable ordering of the input variables – be it by using static variable ordering [13, 14] which is dependent upon circuit topology, to the dynamic variable ordering using optimization or evolutionary techniques [11, 16-25]. Traversing BDDs via algorithms through all the nodes and edges of the ordered directed graph takes polynomial time in the current size of the graph. However, creation of new BDDs might lead to a major increase in the number of nodes in the BDD depending on the position of nodes in the graph, thereby, leading to exponential memory and run time necessity. The choice of BDD variable order is very crucial [26], and to determine an optimal variable ordering is an NP-hard problem [27]. There are a large number of algorithms being used for variable ordering in BDDs. Basically, there are three broad categories for these algorithms – static variable ordering [11], dynamic variable ordering [11] and evolutionary algorithms [20, 28]. In the last years, several methods have been proposed with good quality outcomes, such as, genetic algorithms or evolutionary algorithms, simulated annealing etc. But they were applicable only to small functions and had bad runtime behaviour [6].

1.4. ORGANIZATION OF THE DISSERTATION

The chapters in the dissertation have been organized as follows:

1.4.1. ***Chapter 1*** gives the basic introduction of Binary Decision Diagrams along with their applications and significance. It also includes the need of the optimization and the usage of algorithms to achieve it. The organization of the dissertation has been described.

1.4.2. ***Chapter 2*** contains the literature review that includes the study done from the research articles related to BDDs and Genetic Algorithms and their contribution to the optimization along with the role of power optimization.

1.4.3. ***Chapter 3*** contains the heuristic algorithms – Genetic Algorithm and Hybridized Genetic Algorithm with the crossover operators and describes their working principles and algorithmic flows.

1.4.4. ***Chapter 4*** is all about the concept of power dissipation as per the switching activity and its importance. Low Power estimation has been described for the BDD Mapped Circuits using Probabilistic Power estimation technique.

1.4.5. ***Chapter 5*** discusses the problem formulation.

1.4.6. ***Chapter 6*** provides information about adopted methodology for the research and the analysis.

1.4.7. ***Chapter 7*** presents the results of node count, computation time and power from the proposed approaches when tested on IWLS'93 benchmark circuits.

1.4.8. ***Chapter 8*** presents the graphical analysis of results and discusses the conclusion and the future scope of the research work.

Chapter 2. Literature Review

For this research work, many papers and research articles in the field of BDDs, Variable Ordering, Algorithmic approaches, Genetic and Hybridized Genetic Algorithms for the optimization of BDDs were studied. A succinct review on the basis of studies of these papers has been presented as follows:

2.1. C. Y. LEE, (1959) [2]

This paper discusses the advantage of using binary-decision representation than the ordinary Boolean expression representation along with giving the concept of transforming binary decision programs to switching circuits and also presents their computation.

2.2. S. B. AKERS, (1978) [1]

This paper introduces BDD as a major contribution in getting a complete and concise analysis, description, implementation and testing of both the combinational as well as sequential digital functions along with presenting their advantage of easy interconnecting for defining still larger functions. The concept was clarified further with the example of carry look-ahead adder.

2.3. R. E. BRYANT, (1986) [3]

This paper presents the new data structures for manipulation of Boolean functions using the canonical representations. The efficiency of the canonical structures has been discussed along with the dependency of the function on its ordering. Any graph can be reduced to its canonical form in linear time and helps in saving storage requirement as well as the computation time required for the solution. Algorithms and experimental results have been presented for the logical verification.

2.4. S. J. FRIEDMAN AND K. J. SUPOWIT, (1987) [18]

The authors presented dynamic variable ordering and constructed Ordered BDDs for the benchmark circuits. The size of BDDs is largely influenced by the ordering of variables. Hence to achieve optimization, it becomes necessary to reduce the size of BDDs, for example, in case of differential cascode voltage switch (DCVS) trees. The importance of the optimum choice of the ordering of variables to achieve compact represesntation is discussed.

2.5. N. ISHIURA, H. SAWADA AND S. YAJIMA, (1991) [21]

The exact minimization method and gradual improvement methods employed by exchanging the m adjacent variables, for minimizing BDDs have been presented in this paper. The use of BDD representation for intermediate functions and the concept of pruning was introduced to optimize the computation cost.

2.6. P. Y. CHUNG, I. M. HAJJ AND J.H. PATEL, (1993) [13]

The authors gave the variable ordering guidelines on the basis of which they implemented Distance-based Variable-ordering technique and reached to the conclusion that though not much consistent, are still able to reduce OBDDs for some circuits. The implementation was done on ISCAS and MCNC benchmark circuits. The result was that it is better to try a set of fast heuristics on a specific circuit, rather than try only a single one, and choose the best for the circuit accordingly.

2.7. I. WEGENER, (1995) [29]

This paper discussed the concept of simulated annealing as well as genetic algorithms separately for variable ordering of BDDs. In genetic approach, the PMX, sifting, mutation were applied. The results obtained with genetic approach resulted in good variable ordering but consumed large computation times.

2.8. R. DRECHSLER, M. KERTTU, P. AND M. THORNTON, (2002) [5]

In this paper, the technique for optimizing power consumption by reducing switching activity of the nodes of the BDD has been presented. This is because in CMOS based digital circuits, the power dissipation is directly dependant on the switching activity. Therefore, by approximating the switching activity of nodes of the circuits by internal probabilities of switching, dynamic power dissipation can be estimated. The power dissipation factor plays a significant role in today's world of shrinking dimensions, especially for portable or hand-held devices. Experimental results for MCNC benchmark circuits have been analysed in this paper.

2.9. S. S. LIN AND C. J. WEI, (2005) [30]

In this paper, optimal ordering of BDDs has been achieved for almost all the LGSynth 91 benchmark circuits with up to 500 variables, in contrast to only up to 64 variables with the then existing techniques. Analyses and results have been presented for related work.

2.10. W. MINGQUAN AND Y. HAIBIN, (2005) [31]

Tabu Search has been implemented with the Genetic Algorithm based approach for the BDD minimization, using variable swapping and uniform distribution to generate the neighbouring solution set. Experimental results have been demonstrated to tell the efficiency of the proposed methodology. A comparison has been made with the various existing techniques.

2.11. W. LENDERS AND C. BAIER, (2005) [8]

This paper summarizes the genetic based approach to obtain the optimum variable ordering for the BDDs. The effect of various crossovers and the influence of mutation was presented. The mutation explained gave the major advantage of the ability of the algorithm to revert back to the original ordering in case it was better than the newly generated solutions. The main result obtained in this paper was that the hybrid of genetic with the reordering algorithms leads to the achievement of a good balance between the solution quality and speed. The presentation opened the gates for group sifting algorithms rather than single variable sifting, used till that time.

2.12. P. W. C. PRASAD, A. ASSI, A. HARB AND V.C. PRASAD, (2008) [27]

This paper proposes a new improved variable ordering method to further reduce the Reduced Ordered BDDs. The Boolean function has been formed in the unidirectional graph topology and optimized for three parameters – total number of nodes in all paths, number of paths, and maximum number of nodes among all paths. The shortest path between two variables as well as the sums of the shortest paths to all the other variables was also employed in higher levels of the proposition. The method has been found quite effective in finding the optimal variable ordering for many benchmark circuits.

2.13. F. TOWHIDI, A.H. LASHKARI AND R.S. HOSSEINI, (2009) [32]

The BDDs, OBDDs and ROBDDs have been introduced in this paper. The applications on the BDDs have been reviewed.

2.14. O. BRUDARU, R. EBENDT AND I. FURDU, (2010) [33]

The authors were able to develop a double hybridized optimized algorithm for variable ordering for parallel or distributed implementations by implementing the use of embryos instead of full length chromosomes with appropriate growing strategies and fitness definition and by incorporating sifting into GA by a hyper mutation operator with due consideration to the adjustment of probabilities of mutation and crossover operators, plus the embedded branch and bound technique. The results obtained offered good quality of the solutions obtained.

2.15. T. BHUVANESWARI, V. PRASAD, A. K. SINGH AND C. SENTHILPARI, (2011) [34]

BDD Pass Transistor Logic (PTL) logic synthesis has been analysed for high performance circuits, by utilizing a one-to-one correspondence between the BDD node and PTL cell. Layouts have been generated and simulated in terms of power consumption, area and delay for the benchmark circuits.

2.16. S. CHAUDHURY AND A. DUTTA, (2011) [20]

The authors were able to achieve an optimal trade-off between the area and power reduction by using the fitness function employed in terms of number of nodes in BDD and the Switching activity along with the leakage factor. The genetic algorithm was used for variable ordering which was affected mainly because node switching and leakage is dependent on the number of BDD nodes and its order.

2.17. C. L. VALENZUELA AND P. Y. WANG, (2013) [35]

The authors achieved the area optimization on soft modules for slicing floorplans when they considered their fitness function in proportion to the amount of dead space on the chip, adopted the Roulette Wheel Selection procedure, Mutation was based on Probability function defined for their problem, and overall cycle crossover was taken into algorithm.

2.18. M. D. B. SIDDIQUI AND M. BANSAL, (2013) [26]

This paper proposes a new improved technique for initial ordering to generate good initial population. The authors were able to achieve optimization in terms of node count, and hence, improvement in chip area.

2.19. S. REHAN AND M. BANSAL, (2013) [9]

In this paper, the various evolutionary algorithms were compared and were found better in terms of their performances and the results have been discussed for the multi-input adders and the benchmark circuits. The optimization was achieved in terms of area, by achieving the reduction in the node count for the BDDs.

2.20. J. JIANG, H. LIU, H. FENG AND F. MENG, (2014) [36]

This paper introduced Simulated Annealing technique in Genetic Algorithm for Directed Acyclic Graphs to decide the acceptability of the solution obtained by crossover or mutation operations performed in genetic algorithm, using the Boltzmann's probability function, in order to avoid premature local termination so that optimal global solution is attained.

2.21. R. KAUR AND M. BANSAL, (2014) [37]

Three types of crossover operators (order, cycle and PMX) were used to achieve optimization in terms of node count and computation time for multi-input adders. The comparison was made with the existing techniques.

2.22. K. SINGH, S. SINGH AND R. SINGH, (2014) [38]

The authors implemented Simulated Annealing for parent selection in genetic algorithm in order to achieve balanced bi-partitioning (of a VLSI circuit) in terms of size and connections while avoiding premature convergence to any local optimum. They were able to reduce the CPU time thereby increasing speed.

2.23. N. SATHYA AND A. MUTHUKUMARAVEL, (2015) [24]

This paper reviews the recent work done in the fields of algorithms like Genetic Algorithm, Ant Colony Algorithm, Tabu Search to locate the nearest optimized solution with respect to the travelling salesman problem. Heuristic approach as genetic algorithmic approach, besides being dependent on the crossover and mutation operators used, has been still observed to be able in finding good solutions for the problem. Combining the heuristic methods with genetic approach is a hopeful methodology.

Chapter 3. Heuristic Algorithms

In mathematical computation and optimization, and, in artificial intelligence, a 'Heuristic Algorithm (HA)' is an algorithm based technique which is designed to provide near optimal solutions to a problem in a lesser computation time when the classic methods go too slow. HA has been found to yield the approximate exact solutions when the classic methods fail to find any exact solution. The solution obtained from the HA may not be the best out of all possible solutions, but it is quite close to the best solution. HA offer great flexibility as they can be easily implemented either to generate new good seeds for some other optimization algorithms or importantly, for the NP-hard problems also that need to be handled in the applications of the real world. The category of HA involves local search, tabu search, simulated annealing, and genetic algorithms.

3.1. GENETIC ALGORITHM

The Genetic Algorithm (GA), invented by Prof. John Holland, in 1975, at University of Michigan, is a widely popular and an excellent optimization algorithm based on the natural process of evolution and is able to achieve good soutions much faster than the classic techniques available.

Four well-known approaches for evolutionary computation are genetic algorithms (GAs) [37], evolution strategies (ES), evolutionary programming (EP), and genetic programming (GP). Natural evolution is the motivation behind the basis of such algorithms.

Based on the initial population size [39, 40], the two parents are selected from the population and are subjected to produce offspring. The objective of these algorithms is to find the optimal solution to the given problem. Since, these are heuristic techniques, so the result obtained with these algorithms may not be the best one. But they are found to give very good near-optimum solutions.

The major advantages offered by the GAs are are follows:

- Adaptivity and Learning from experience
- Easier and Intrinsic Parallelism
- Efficiency for complex problems

3.1.1. Basic Principle & Algorithm, Flowchart

In the GA terminology, the 'population' refers to the set of solutions to the given problem and is evolved over a number of generations; 'chromosome' refers to each individual (or solution) in the population; 'Genes' refer to the individual characters (or symbols) in the chromosome. The 'population size' provides the information stored by the GA. In GA, iterations of the existing population are performed where every time a new generation is evolved in order to get better solutions for the problem. The 'evaluation function' defines the fitness (opposite of *cost*) of each candidate of the population.

Various individuals out of the population are evaluated for their fitness, which is area in terms of node count in our case. As per the fitness function defined for the given problem, the variables are arranged according to their fitness levels. According to the Darwin's theory of the Survival of the Fittest, the fittest individuals are selected to further carry on the evolution processes and disperse their good genes to get more converging solutions. The best chromosomes are allowed to perform operations like crossovers, mutation [20] and/or inversions [6] in order to generate further good offspring chromosomes. These chromosomes will be evaluated for their fitness levels and allowed to reproduce further only if they are better than the earlier generation. Hence, with this algorithmic approach, gradually, the worst solutions are eliminated and good solutions are accepted at every level of the iteration. The number of iterations is kept large enough to get a best-fit solution for the problem.

So, it is possible that the new individuals completely replace the old ones or the old ones reproduce with the new ones to produce optimal results as the population size is kept constant throughout. The number of iterations is decided either based on some kind of stopping criteria or till the time when no further improvements in the fitness values are achieved.

The two basic processes [26] employed by any genetic algorithm are:

- Inheritance (passing of features from one generation to the next)
- Competition (Survival of the Fittest as per Darwin's principle)

The basic principle of GA is as shown in Fig. 3.1.

1. Generate initial population
2. Initialize population (Either randomly or as per some algorithm)
3. Find fitness function
4. Evaluate each individual
5. Choose the best-fit ones (avoid repetition in crucial and high speed requiring cases)
6. Apply genetic operators (crossover, mutation, inversion)
7. Repeat 3 to 6 Until stopping criteria is met (or no more improvements are obtained)

Fig. 3.1. Basic principle of GA

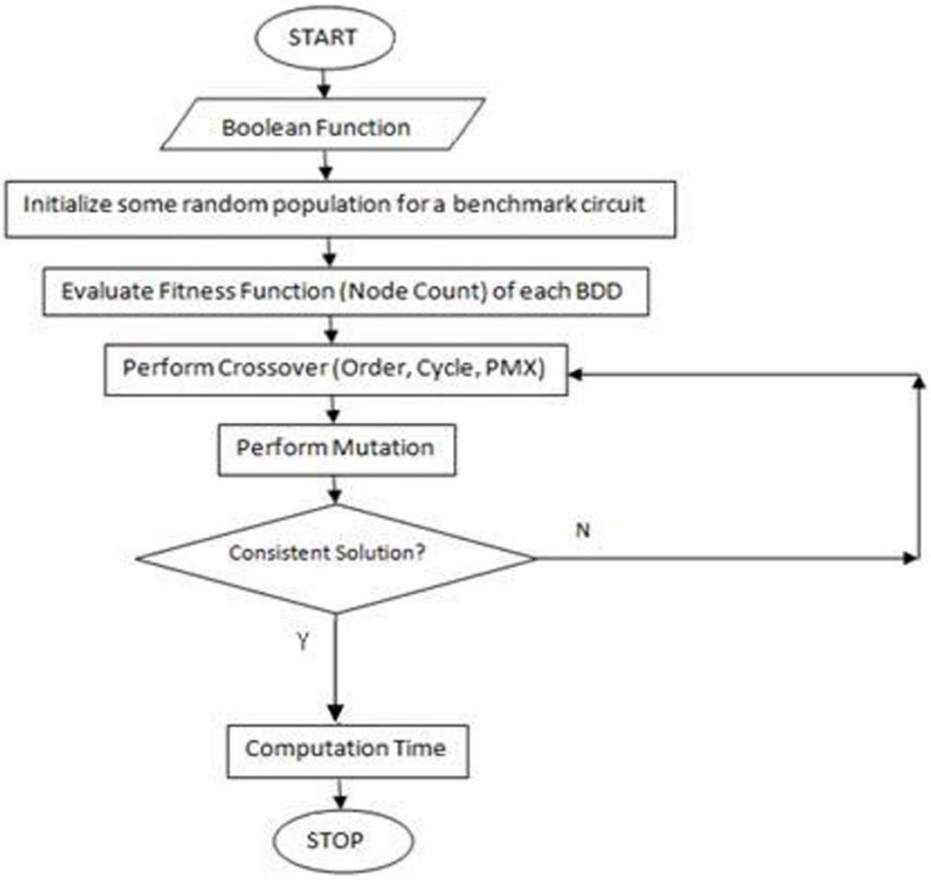

Fig. 3.2. Flowchart for BDD variable ordering using GA

3.1.2. Genetic Operators

There are many operators which can be used in a GA for forming a new solution out of the existing set of solutions. The technique by which new solution (or an individual), can be formed from the existing solutions (or parents), is identified by the name Operator. These are the operators which are applied to get the new solutions from the existing solutions. The main operators used on a wide scale when talking about genetic algorithms are:

- Selection Operator
- Crossover Operator
- Mutation Operator
- Inversion Operator

3.1.2.1. Selection Operator

In every generation, some individuals are picked up from the set of population to reproduce and produce offspring. The selection can be made at random or according to the fitness level criteria (preferred and used). According to the Darwin's theory of Survival of the Fittest, the individuals that are best fit are allowed to reproduce further. This is done so as to make sure that only the best genes are carried further to the next generations. In this way, the worst solutions get eliminated at each level of iteration.

3.1.2.2. Crossover Operator

This is the process of making the parents combine their genes to produce offspring. Various techniques for Crossovers have been proposed and accepted till now, out of which, Order Crossover (using concept of selective positions or using cut points), Cycle Crossover, Alternate Crossover, Partially Matched Crossovers (PMX) are the mostly used ones.

Another classification of crossover operators goes by the – one-point and two-point crossover operators where one or two chromosome positions are randomly selected between one and (N-1) where N is the chromosome length and the two parents are crossed at those points.

- In case of '**one-point crossover**', the first child is identical to the first parent up to the crossing point and identical to the second parent after the crossing point.

  e.g. Parent 1: **10010011** | 11101100

  Parent 2: 00110110 | **10010110**

  Offspring 1: **10010011** | **10010110**

  Offspring 2: 00110110 | 11101100

- In case of '**two-point Crossover**', two points are set on the parents. The first child is identical to the extremes of the two points of first parent and identical to middle of the two points of the other parent, whereas the other child has the corresponding swapped form.

  e.g. Parent 1: **1001** | 00111110 | **1101**

  Parent 2: 0011 | **01101001** | 0110

  Offspring 1: **1001** | **01101001** | **1101**

  Offspring 2: 0011 | 00111110 | 0110

- In case of '**Order Crossover**', a portion of one parent is mapped to a portion of the other parent. From the replaced portion on, the rest is filled up by the remaining genes, where already present genes are omitted and the order is preserved.

  e.g. Parent 1: 8 4 7 3 6 2 5 1 9 0

  Parent 2: 0 ~~1~~ ~~2~~ ~~3~~ 4 ~~5~~ ~~6~~ 7 8 9

  Offspring 1: 0 4 7 3 6 2 5 1 8 9

- In case of '**Cycle Crossover**', a cycle is found between the two parents. Genes from the first parent, which form the cycle, are copied as it as to the child. The remaining genes of the child are filled with the genes from second parent.

  e.g. Parent 1: 1 2 3 4 5 6 7

  Parent 2: **5 4 6 7 2 3 1**

  Cycle: 1-5-2-4-7-1 ⇨ Offspring 1: 1 2 **6** 4 5 **3** 7

- In case of '**Partially Mapped Crossover (PMX)**', a random sub set of a parent is chosen, that part is swapped with the corresponding part from the second parent, the cycle between the genes is established, and the corresponding genes are swapped according to that.

e.g.	Parent 1: 1 2 **3 4 5 6** 7	Now, cycle: 1-6-3, 7-4, 2-5
	Parent 2: 5 4 **6 7 2 1** 3	Child 1: 3 5 6 7 2 1 4
	After Swapping,	Child 2: 2 7 3 4 5 6 1
	Parent 1: 1 2 **6 7 2 1** 7	
	Parent 2: 5 4 **3 4 5 6** 3	

3.1.2.3. Mutation Operator

This operator modifies the properties of a solution by changing one bit of the solution randomly selected by the algorithm. The mutation probability should not be set too high, as doing this may lead to the search becoming a primitive random search resulting in the loss of even the good properties.

This is the operator used to abruptly change the values of one or more variables by swapping any pair of bits (in case of binary data). This is required to introduce a change in species in case the offspring produced in the algorithm are of same type and therefore, are unable to examine the entire solution set possible for the given problem.

In a binary-coded GA, mutation may be done by flipping a bit, while in case of a non-binary coded GA, mutation is achieved by randomly generating a new character in a specified position.

The mutation probability i.e. the probability of mutating each gene has to be selected such that it is neither too high nor too low, as its higher value leads to increased random perturbation leading to lost resemblance of offspring with their parents; and its lower value leads to skipping of certain gene values that might be beneficial.

The number of mutations in this work was limited to number of elements/2 + 1.

3.1.2.4. Inversion Operator

This operator takes a random segment in a solution string and invert it end for end in such a way such that it does not modify the actual solution as represented by the original gene. Instead, it only changes the representation of the solution. Inversion simply refers to the rearrangement of the genes just for representation purpose so that only group formation is changed. This operator may hold significance in case when a large schemata contains large number of don't cares at various separated positions, in that case, inversion operator may be used to switch to a representation where these don't cares will be represented together.

3.1.3. Applications of Genetic Algorithms

The application of GAs is efficient when:

- The search space is large, complex and NP-hard or NP-complete.
- Mathematical analysis is not available.
- Traditional or classic search methods fail to get the solution.
- Near optimal solution is to be obtained in lesser time.

3.2. HYBRIDIZED GENETIC ALGORITHM

As the heuristic algorithms carry the ability to be able to be combined with other dynamic or heuristic or evolutionary methods [41], in order to further optimize and enhance the solutions, hence, the GA implemented has been next combined with the Branch and Bound Algorithm to form the Hybridized Genetic Algorithm (HGA). Branch and Bound works the same way as GA but in addition to GA, this Branch and Bound works a little more by putting a lower bound and an upper bound on the evaluated fitness values. This helps to discard the highly fluctuated values in order to make the solution converge to the best near optimum solution in a more efficient manner and in a lesser computation time.

3.2.1. Principle & Algorithm, Flowchart

The basic algorithm principle and the algorithmic flow for the approach have been shown in Fig. 3.

1. Generate initial population
2. Initialize population (Either randomly or as per some algorithm)
3. Find fitness function
4. Evaluate each individual
5. Choose the best-fit ones (avoid repetition in crucial and high speed requiring cases)
6. Apply Branch and Bound Algorithm
7. Apply genetic operators (crossover, mutation, inversion)
8. Repeat 3 to 6 Until stopping criteria is met (or no more improvements are obtained)

Fig. 3.3. Working principle of HGA

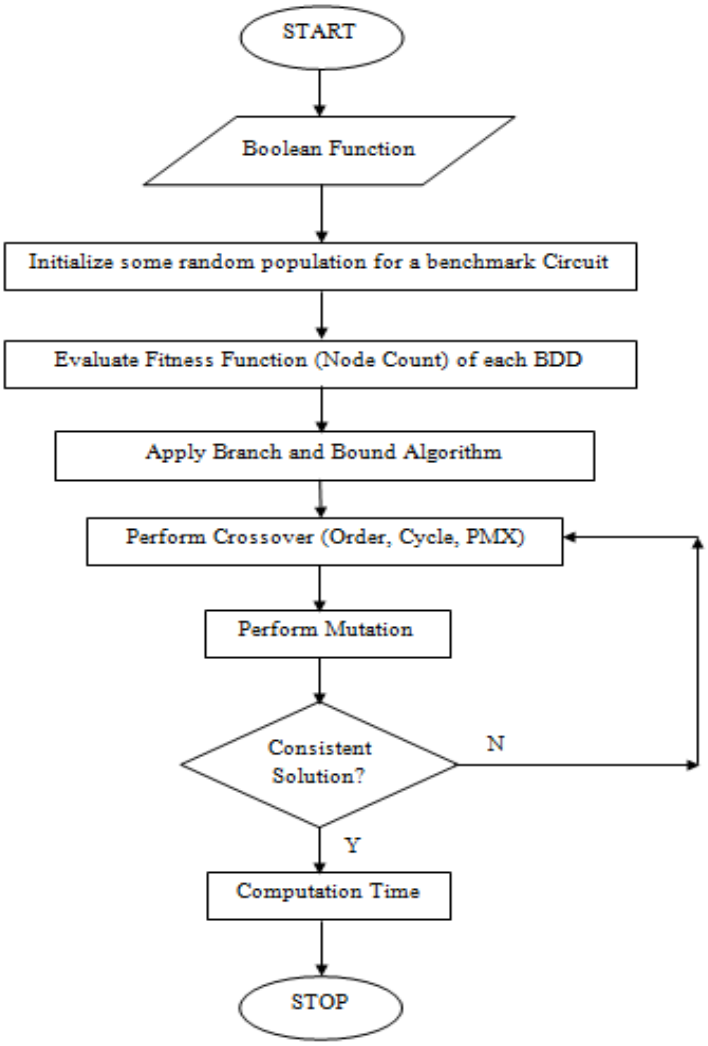

Fig. 3.4. Flowchart for BDD variable ordering using HGA

3.2.2. Branch and Bound Algorithm

Branch and Bound algorithm, proposed by A. H. Land and A. G. Doig, in 1960, is a systematic optimization algorithm designed for combinational and discrete, and also for real valued problems. The algorithm moves by exploring all the branches of the root node of the tree. This refers to the subsets of the possible solution set. Every branch is checked for the estimated lower and upper bounds obtained for the optimum solution one by one and the new candidate solution obtained is discarded if it is not in the defined bound. The process is repeated for all the nodes in the BDD till all the branches are covered and the stopping criterion for the algorithm is met. So, the working of the complete algorithm is determined by efficiency in finding the correct estimation of bounds.

This algorithm works in accordance with the two principles:

- **Branching:** Refers to splitting of the total search space into sub divisions.
- **Bounding:** Applying the lower or/and upper bound on the solution set.

Algorithm for the Branch and Bound technique goes as shown in Fig. 3.5.

1. Select a node to be processed either randomly or through some other heuristic method.
2. Apply Branching on the selected node either in Depth First Search (DFS) manner or a Breadth First Search (BFS) manner.
3. Apply Optimum Lower Bound.
4. Apply Optimum Upper Bound.
5. Repeat till all nodes have been covered or the stopping criteria have been met.

Fig. 3.5. Principle of Branch and Bound algorithm

The bounds are to be estimated very carefully so that the range of the space search is a valid and optimized range. Higher Upper bound and Lesser Lower bound may lead to the entrance of far from optimized values in the space range. This may lead to increased computational times and reduced speeds. Hence, the efficiency of this algorithm is largely dependent on the optimum bounds.

Chapter 4. Power Optimization using Probabilistic Technique

The power consumed has also become a significant factor in the design now. Earlier, the major parameters to the overall performance were cost, area and reliability. But now, this trend has been changed and power has been gaining equal importance. The importance of optimization of power has been growing due to the increased usage of battery powered embedded systems [5]. It has been found that the switching activity of a node in the CMOS based digital circuits directly contributes to the overall dynamic power dissipation [5]. Temporal correlation of the recurring input signals can leave a significant effect on the switching activity [23], and therefore, the power consumption. The power dissipation for every node of the BDD has been computed by the estimated switching activity and the fan out of the node [5], for the capacitive load. The ordering of variables in the BDD not only affects the overall area, but also the overall power consumed by the system. The power dissipation has been minimised by minimising the sum of all internal switching activities at each node of the BDD. The fan out of each BDD node determines the total stages of active buffers, and hence, the number of BDD edges pointing to the corresponding node of the BDD. The modelling of power dissipation and switching activity has been done according to [5], while ignoring the leakage and buffering effects.

4.1. IMPORTANCE OF POWER OPTIMIZATION

In today's era, power optimization is one of the most important considerations in the lifecycle of a project. The importance of power consumption can be visible from the hand-held devices which demand more and more features in limited area and desire battery power to be as large and efficient as possible so that device heating is minimum and battery backup is maximum. Other technical media such as medical equipment, measurement, test, data, remote sensing, media, wireless stations, etc – are all sensitive to power consumption. Hence, this factor can't be ignored or left undealt with.

4.2. SWITCHING ACTIVITY ESTIMATION

Switching activity estimation is a very important aspect of the system. To calculate the switching activity of the nodes of the BDDs, certain assumptions have been made. These are listed as follows:

- Input Signals modelled as Strict-Sense Stationary (SSS), thus, signifying that all switching happens simultaneously.
- Input Signals modelled as mean-ergodic with zero delay, means probability distribution and switching values of the signals don't vary over time.
- Input Signals are mutually independent i.e. they are spatially uncorrelated.
- Spatial correlation between the Shannon cofactors for the function under consideration is assumed to be nil to avoid the high computational complexities required to find exact results.

Now, for the calculation of the switching activity, [5] has given the formula for bottom - up approach for finding switching activity for each node of BDD, as:

$$
\begin{aligned}
a(f) = \Bigg(& \frac{P(f0) + P(f1) - 2P(f0)(f1)}{1 - \big(a(f0) + a(f1) - a(f0)a(f1)\big)} \\
& - \frac{\frac{1}{2}\big(a(f0) + a(f1) - a\,(f0)a(f1)\big)}{1 - \big(a(f0) + a(f1) - a(f0)a(f1)\big)} \Bigg) \\
& \times a(v)\big(1 - a(f0)\big)\big(1 - a(f1)\big) + \frac{1 - P(v) - \frac{a(v)}{2}}{1 - a(v)} \\
& \times a(f0)\big(1 - a(v)\big)\big(1 - a(f1)\big) + \frac{P(v) - \frac{a(v)}{2}}{1 - a(v)} \\
& \times a(f1)\big(1 - a(v)\big)\big(1 - a(f0)\big) + \frac{1}{2}a(v)a(f0)\big(1 - a(f1)\big) \\
& + \frac{1}{2}a(v)a(f1)\big(1 - a(f0)\big) + a(f0)a(f1)\big(1 - a(v)\big) \\
& + \frac{1}{2}a(v)a(f0)\big(a(f1)\big)
\end{aligned}
\tag{2}
$$

In (2),

P(f): output probability of f that f is 1

a(f): activity of f, i.e. the change of value of f from one cycle to another

v: input variable

f0: low cofactor

f1: high cofactor

The main requirements when dealing with low power estimations for BDD mapped circuits was that there is a necessity of a precise, accurate and computationally efficient method for switching activity estimation.

4.3. LOW POWER ESTIMATION FOR BDD MAPPED CIRCUITS

The power consumption or the power dissipation in every node of the BDD is calculated by the switching activity (estimated value) and the fan out of the node (with respect to the capacitive load).

A BDD carries the capability to be directly transformed into a multiplexer based circuit. Moreover transforming a BDD into a circuit is also very easy. It involves replacing the BDD nodes by small sub circuits of the system and the directed edges of the BDD by the connecting wires. Hence, all the parameters of the circuit can be smartly tackled with the concept and use of BDDs. As the variable ordering in BDD was found to be highly affecting the node count, at the same time, it also determines the level of the internal switching.

The power consumption in each node is a function of the fan-out of node and the internal switching activity. The technique mentioned to in [5] helped to analyse that the use of complemented edges for the BDDs have resulted in optimization of the overall size and performance of the operations.

The following observations were made:

- Output Probability P(f'), for the complemented edge f', is given by: 1-P[f].
- Switching Activity a(f'), for the complemented edge f', is given by: a[f].

Implementing variable exchanges while working with BDDs with complemented edges became more efficient with the use of the above two properties to deal with the local switching probabilities.

To reduce the overall power consumption or power dissipation of the BDD, the technique of minimizing the sum of all the switching happening at every node of the BDD, was implemented in [5].

In order to model the power dissipation for the BDD mapped circuits, a cost model is developed which is based on the overall switching activity for a predefined matrix of the output probabilities of the support variables (dependent variables). Each BDD node is mapped into a multiplexer based circuit after minimizing the sum of all the internal switching activities acting at each and every BDD node of the system. The number of BDD edges pointing to the node (or the fan-out of the node) determines the required number of the stages of active buffers.

To compute the power dissipation at the mapped node k, the following estimation equation was used:

$$PD_k = a(k) \times driver\left(fanout(k)\right) + leakage(k) \tag{3}$$

This equation ignores the power dissipation due to leakages, which are process dependent, and buffering of input signals and also, the effect of parasitic capacitances arising because of routing. In this case the effect of leakages was not taken into account because its relative contribution is negligible in front of total power dissipation occurring.

Chapter 5. Issue Formulation

Nowadays, as the chip dimensions are reducing, there is a significant need to be able to integrate more and more components and functionality onto a single chip with considerable importance to the power dissipated. This requires more efficient usage of available chip area. When expressing the Boolean functionality in terms of BDDs, the main objective is the variable ordering of BDDs as it largely influences the overall node count [3] and hence, the area, and the internal switching activity and hence, the power dissipation. For a wide range of Boolean functions, there are polynomial-sized BDDs for "good" variable orderings, while the size of BDDs grows exponentially under "bad" variable ordering. Unfortunately, improving the variable ordering of BDDs is NP-Complete and finding the best order is NP-hard [3, 28]. However, the most tedious job in case of Ordered BDDs [14] is to find an optimal variable order.

Moreover, in today's era of fast computing, there is an urgent need for those methodologies which are able to compute the optimized results in a limited CPU time.

Many variable ordering methods have been proposed in the last decades. These methods include ***static*** and ***dynamic*** techniques. ***Static techniques*** are applied before constructing the BDD to generate an order, dependent upon the implemented function as in [14]. The drawback of using Static techniques is that it requires a prior knowledge of the function's behavior, like effect of each input variable on the function, which may not be always known [26]. Whereas, the ***Dynamic techniques*** are focused on generating new variable orders to improve the size of the already constructed BDD.

One out of such techniques is Rudell's **sifting algorithm** [16]. It is based on the ordering achieved by swapping the variables. It first selects one variable, moves it to every possible position and finds the number of nodes. Then the position, which gives the minimum number of nodes, is fixed for that variable. Now, the next variable is picked and moved to every possible position except the positions already been fixed.

The position of the new variable is again fixed considering the minimum number of nodes. The process is repeated till all variables are sifted according to new optimal solution. The final ordering by fixing all variables is the improved ordering achieved by the sifting [6, 16, 37] algorithm.

Although this algorithm decreases the number of ordering permutations from n! to n2, in many cases the size of resulting BDD is far from optimal. Another technique is Window permutation, which has been found out to be fast, but rather a weak minimization heuristic [37].

Many other methods have been using genetic algorithm [6, 20, 8, 37, 40] and different heuristics [13-15] to generate new orderings. The Genetic algorithm based approach used genetic operations like crossover and mutation [20] in order to generate new variable ordering based on some fitness criteria. Genetic algorithm based approaches have been found to be yielding good run times but when compared to other techniques, node reduction result hasn't found much improvement. Simulated annealing based approach has been found to give better area results but on the cost of long run times [26].

Considering the drawbacks, in this paper, an improved Genetic Algorithm approach for optimizing the node count and hence, area has been presented and the simulation results have shown significant minimization in the node count, in a limited CPU time.

Chapter 6. Methodology and Analysis

- The Genetic Algorithm and the Hybridized Genetic Algorithm using three types of crossover operators have been implemented with C++ codes and simulated using BDD package Buddy 2.4 [42] on Ubuntu 14.04.

- The optimization in terms of node count (area), computation time and power dissipation has been achieved.

- Simulation Results i.e. node count, corresponding computation times and power dissipation for IWLS'93 combinational benchmark circuits have been presented.

- The Probabilistic Power Dissipation has been estimated and computed with the help of switching activity on the BDD nodes for both the Genetic Algorithm and the Hybridized Genetic Algorithm.

- The comparison has been made with the existing Window (WIN2, WIN2ITE, WIN3), Sifting (SIFT, SIFTITE) and Random algorithms that have been included in the Buddy 2.4.

- The simulation results for different IWLS'93 benchmark circuits for genetic and hybridized genetic algorithms have been shown in Tables 7.1, 7.2, 7.3, 7.4, 7.5, 7.6, 7.7 and 7.8 and the graphical analysis has been shown in Fig. 8.1, 8.2, 8.3, 8.4, 8.5 and 8.6.

- The proposed approaches in both the cases have been observed to provide better node count reduction, in lesser computation times and lower power dissipation for the circuits in minimum number of iterations for a large number of runs.

Chapter 7. Simulation Results

In the tables, ‘Ben Ckt’ refers to the ‘Benchmark Circuits’ of IWLS’93 on which the algorithms have been tested; #i and #o indicate the number of inputs and outputs respectively, for the corresponding benchmark circuits; and, NC refers to the Node Count.

7.1. ESTIMATION OF NODE COUNT

The simulation results for node counts have been presented in Tables 7.1, 7.2, 7.3 and 7.4.

Table 7.1 Node count comparison against initial values for window algorithms for IWLS'93 benchmarks

Ben Ckt	#i - #o	Initial NC	WIN2		WIN2ITE		WIN3	
			NC	%age reduction in NC by WIN2	NC	%age reduction in NC by WIN2ITE	NC	%age reduction in NC by WIN3
5xp1	7 - 10	95	83	12.6316	81	14.7368	75	21.0526
alu4	14 - 8	990	746	24.6465	752	24.0404	743	24.9495
b12	15 - 9	76	87	-14.4740	80	-5.2632	72	5.2632
Bw	5 - 28	115	115	0.0000	110	4.3478	106	7.8261
clip	9 - 5	215	158	26.5116	108	49.7674	105	51.1628
con1	8 - 2	20	17	15.0000	15	25.0000	15	25.0000
duke2	22 - 29	736	543	26.2228	500	32.0652	588	20.1087
Inc	7 - 9	83	74	10.8434	72	13.2530	74	10.8434
misex1	8 - 7	45	41	8.8889	40	11.1111	39	13.3333
misex2	25 - 18	138	141	-2.1739	125	9.4203	95	31.1594
sao2	10 - 4	123	122	0.8130	100	18.6992	105	14.6341
sqrt8	8 - 4	48	43	10.4167	43	10.4167	42	12.5000
squar5	5 - 8	41	37	9.7561	42	-2.4390	39	4.8780
t481	16 - 1	194	32	83.5052	32	83.5052	32	83.5052
vg2	25 - 8	647	521	19.4745	410	36.6306	350	45.9042
Average Reduction %age:			**15.4708**		**21.6861**		**24.8080**	

Table 7.2 Node count comparison against initial values for sifting, random algorithms for IWLS'93 benchmarks

Ben Ckt	#i - #o	Initial NC	SIFT		SIFTITE		RANDOM	
			NC	%age reduction in NC by SIFT	NC	%age reduction in NC by SIFTITE	NC	%age reduction in NC by RANDOM
5xp1	7 - 10	95	76	20.0000	76	20.0000	86	9.4737
alu4	14 - 8	990	758	23.4343	742	25.0505	737	25.5556
b12	15 - 9	76	72	5.2632	72	5.2632	63	17.1053
bw	5 - 28	115	108	6.0870	108	6.0870	116	-0.8696
clip	9 - 5	215	106	50.6977	106	50.6977	112	47.9070
con1	8 - 2	20	16	20.0000	16	20.0000	18	10.0000
duke2	22 - 29	736	609	17.2554	589	19.9728	725	1.4946
inc	7 - 9	83	70	15.6627	70	15.6627	86	-3.6145
misex1	8 - 7	45	39	13.3333	39	13.3333	45	0.0000
misex2	25 - 18	138	110	20.2899	98	28.9855	144	-4.3478
sao2	10 - 4	123	109	11.3821	106	13.8211	130	-5.6911
sqrt8	8 - 4	48	41	14.5833	41	14.5833	50	-4.1667
squar5	5 - 8	41	42	-2.4390	41	0.0000	43	-4.8780
t481	16 - 1	194	32	83.5052	32	83.5052	55	71.6495
vg2	25 - 8	647	395	38.9490	350	45.9042	343	46.9861
Average Reduction %age:			**22.5336**		**24.1911**		**13.7736**	

Table 7.3 Node count comparison for proposed genetic (order, cycle, PMX) for IWLS'93 benchmarks

Ben Ckt	#i - #o	Initial NC	Order		Cycle		PMX	
			NC	%age reduction in NC by Order	NC	%age reduction in NC by Cycle	NC	%age reduction in NC by PMX
5xp1	7 - 10	95	68	28.4211	69	27.3684	68	28.4211
alu4	14 - 8	990	891	10.0000	939	5.1515	734	25.8586
b12	15 - 9	76	70	7.8947	68	10.5263	50	34.2105
bw	5 - 28	115	106	7.8261	106	7.8261	106	7.8261
clip	9 - 5	215	102	52.5581	108	49.7674	93	56.7442
con1	8 - 2	20	16	20.0000	15	25.0000	15	25.0000
duke2	22 - 29	736	506	31.2500	512	30.4348	390	47.0109
inc	7 - 9	83	72	13.2530	72	13.2530	72	13.2530
misex1	8 - 7	45	36	20.0000	36	20.0000	36	20.0000
misex2	25 - 18	138	100	27.5362	102	26.0870	87	36.9565
sao2	10 - 4	123	92	25.2033	90	26.8293	85	30.8943
sqrt8	8 - 4	48	33	31.2500	33	31.2500	33	31.2500
squar5	5 - 8	41	37	9.7561	37	9.7561	37	9.7561
t481	16 - 1	194	85	56.1856	78	59.7938	30	84.5361
vg2	25 - 8	647	339	47.6043	301	53.4776	148	77.1252
Average Reduction %age:			**25.9159**		**26.4348**		**35.2562**	

Table 7.4 Node count comparison for proposed hybridized genetic (order, cycle, PMX) for IWLS'93 benchmarks

Ben Ckt	#i - #o	Initial NC	Order		Cycle		PMX	
			NC	%age reduction in NC by Order	NC	%age reduction in NC by Cycle	NC	%age reduction in NC by PMX
5xp1	7 - 10	95	68	28.42105	68	28.42105	68	28.42105
alu4	14 - 8	990	804	18.78788	860	13.13131	746	24.64646
b12	15 - 9	76	65	14.47368	66	13.15789	45	40.78947
bw	5 - 28	115	106	7.826087	106	7.826087	100	13.04348
clip	9 - 5	215	96	55.34884	106	50.69767	93	56.74419
con1	8 - 2	20	15	25	15	25	15	25
duke2	22 - 29	736	444	39.67391	546	25.81522	400	45.65217
inc	7 - 9	83	72	13.25301	73	12.04819	71	14.45783
misex1	8 - 7	45	36	20	37	17.77778	35	22.22222
misex2	25 - 18	138	94	31.88406	99	28.26087	87	36.95652
sao2	10 - 4	123	90	26.82927	90	26.82927	85	30.89431
sqrt8	8 - 4	48	33	31.25	33	31.25	33	31.25
squar5	5 - 8	41	37	9.756098	37	9.756098	37	9.756098
t481	16 - 1	194	32	83.50515	77	60.30928	30	84.53608
vg2	25 - 8	647	265	59.04173	300	53.63215	146	77.43431
AVERAGE REDUCTION %:			**31.00338**		**26.92752**		**36.12028**	

7.2. ESTIMATION OF COMPUTATION TIME

The simulation results for computation times have been presented in Tables 7.5 and 7.6.

Table 7.5 Computation time (seconds) comparison for proposed genetic (order, cycle, PMX) for IWLS'93 benchmarks

Ben Ckt	#i - #o	Computation Time (seconds)		
		Order	Cycle	PMX
5xp1	7 - 10	0.09	0.11	0.19
alu4	14 - 8	0.22	0.30	6.63
b12	15 - 9	0.06	0.07	0.18
bw	5 - 28	0.20	0.26	0.31
clip	9 - 5	0.13	0.21	0.74
con1	8 - 2	0.04	0.04	0.05
duke2	22 - 29	0.27	0.29	1.40
inc	7 - 9	0.07	0.08	0.17
misex1	8 - 7	0.06	0.06	0.13
misex2	25 - 18	0.08	0.08	0.16
sao2	10 - 4	0.10	0.13	0.37
sqrt8	8 - 4	0.06	0.06	0.14
squar5	5 - 8	0.05	0.06	0.06
t481	16 - 1	0.66	0.66	5.75
vg2	25 - 8	0.18	0.18	1.36
Average Computation Time:		**0.15**	**0.17**	**1.18**

Table 7.6 Computation time (seconds) comparison for proposed hybridized genetic (order, cycle, PMX) for IWLS'93 benchmarks

Ben Ckt	#i - #o	Computation Time		
		Order	Cycle	PMX
5xp1	7 - 10	0.0768	0.0397	0.0949
alu4	14 - 8	0.8112	0.2124	4.3471
b12	15 - 9	0.0378	0.0234	0.0842
bw	5 - 28	0.13	0.0917	0.1675
clip	9 - 5	0.2049	0.1339	0.4528
con1	8 - 2	0.0127	0.0298	0.0175
duke2	22 - 29	0.3035	0.3121	0.7992
inc	7 - 9	0.0269	0.0299	0.0789
misex1	8 - 7	0.034	0.0207	0.0499
misex2	25 - 18	0.0377	0.0299	0.0726
sao2	10 - 4	0.1019	0.0708	0.2332
sqrt8	8 - 4	0.0358	0.0301	0.0557
squar5	5 - 8	0.0198	0.0169	0.0258
t481	16 - 1	0.1596	0.1306	2.2718
vg2	25 - 8	0.1749	0.165	0.6999
Average Computation Time:		**0.14**	**0.09**	**0.63**

7.3. ESTIMATION OF PROBABILISTIC POWER DISSIPATION

The simulation results for probabilistic power dissipation have been presented in Tables 7.7 and 7.8.

Table 7.7 Probabilistic Power comparison for proposed genetic (order, cycle, PMX) for IWLS'93 benchmarks

Ben Ckt	#i - #o	Probabilistic Power		
		Order	Cycle	PMX
5xp1	7 - 10	0.0014	0.0056	1.1059
b12	15 - 9	0.0048	0.0012	7.3275
bw	5 - 28	0.0032	0.0063	0.0063
clip	9 - 5	0.0005	0.0002	0.0045
con1	8 - 2	0.0017	0.0033	0.0117
duke2	22 - 29	3.507	1.241	6.352
inc	7 - 9	0.0124	0.0371	0.0371
misex1	8 - 7	1.2005	1.2922	1.2922
misex2	25 - 18	0.0001	0.0001	0.0012
sao2	10 - 4	0.0001	0.0003	0.0007
sqrt8	8 - 4	4.584	2.5146	1.2922
squar5	5 - 8	1.2922	1.2922	1.2922
t481	16 - 1	0.0001	0.0002	2.1315

Table 7.8 Probabilistic Power comparison for proposed hybridized genetic (order, cycle, PMX) for IWLS'93 benchmarks

Ben Ckt	#i - #o	Probabilistic Power		
		Order	Cycle	PMX
5xp1	7 - 10	0.0011	0.0045	1.0091
b12	15 - 9	0.0037	0.0010	6.3275
bw	5 - 28	0.0021	0.0059	0.0059
clip	9 - 5	0.0003	0.0002	0.0031
con1	8 - 2	0.0012	0.0031	0.0102
duke2	22 - 29	2.6407	0.867	6.112
inc	7 - 9	0.0111	0.0289	0.0289
misex1	8 - 7	1.1192	1.1562	1.1562
misex2	25 - 18	0.0001	0.0001	0.0002
sao2	10 - 4	0.0001	0.0003	0.0006
sqrt8	8 - 4	4.002	2.1145	2.1145
squar5	5 - 8	1.1562	1.1562	1.1562
t481	16 - 1	0.0001	0.0001	1.8975

Chapter 8. Concluding Remarks and Future Scope

8.1. CONCLUSION

- The proposed approaches – the Genetic Algorithm and the Hybridized Genetic Algorithm, both have been found to be efficient in providing an optimized variable ordering in least number of iterations carried over a large number of runs.

- The proposed approaches have resulted in minimum node count when compared to the other existing Window, Sifting and Random algorithms, thus leading to reduced area requirements and thus, able to achieve optimization in area.

- From Table 7.1, it has been observed that out of Window algorithms, WIN3 algorithm resulted in the best node count reduction, with an average value of about 24.81%. Similarly, Table 2 has shown that SIFTITE is able to achieve an average node count reduction of about 24.19%, which is best out of Sifting and Random algorithms. From the results in Table 3, it can be found that the proposed Genetic approach using Order Crossover, results in an average reduction of about 25.92% with a maximum value of about 56.19%; using Cycle crossover, achieves an average reduction of about 26.43% with a maximum value of about 59.79%; whereas, using PMX crossover, leads to the best possible reduction with an average value of about 35.26% with a maximum value of 84.54%.

- From the results in Table 4, it can be found that the proposed Hybridized Genetic approach using Order Crossover, results in an average reduction of about 31% with a maximum value of about 83.51%; using Cycle crossover, achieves an average reduction of about 26.93% with a maximum value of about 60.31%; whereas, using PMX crossover, leads to the best possible reduction with an average value of about 36.12% with a maximum value of 84.54%.

- Both the proposed algorithms – Genetic Algorithm and Hybridized Genetic Algorithm, have been able to give efficiently optimized results for about 90% of the benchmark circuits; hence, these can be employed for Multi-Input-Multi-Output Systems (MIMO systems) in VLSI.

- From Table 7.5, for the proposed Genetic Algorithm, the best computation time of an average value of 0.15s, has been observed in case of Order Crossover though the Cycle crossover also gives average value of about 0.17s as compared to PMX, which takes a little longer, about 1.18s, on an average.

- From Table 7.6, for the proposed Hybridized Genetic Algorithm, the best computation time of an average value of 0.14s, has been observed in case of Order Crossover though the Cycle crossover also gives average value of about 0.09s as compared to PMX, which takes a little longer, about 0.63s, on an average.

- For the estimation of the power dissipation, it can be concluded from Tables 7.7 and 7.8, that the PMX operator consumes the highest power than the other two – order and cycle crossovers. This is true for both of the Genetic approach as well as the Hybridized Genetic approach.

- The proposed Hybridized Genetic Algorithm has been able to yield even better reduction in node count, lesser computation times and lesser power consumptions when compared to the proposed Genetic Approach.

- From the results it is evident that there needs to be a trade-off between the choices area, speed and power. Depending on the priority of the application based on area, time or power, either of the algorithms and either of the crossovers can be employed.

8.2. ANALYSIS OF RESULTS USING GRAPHS

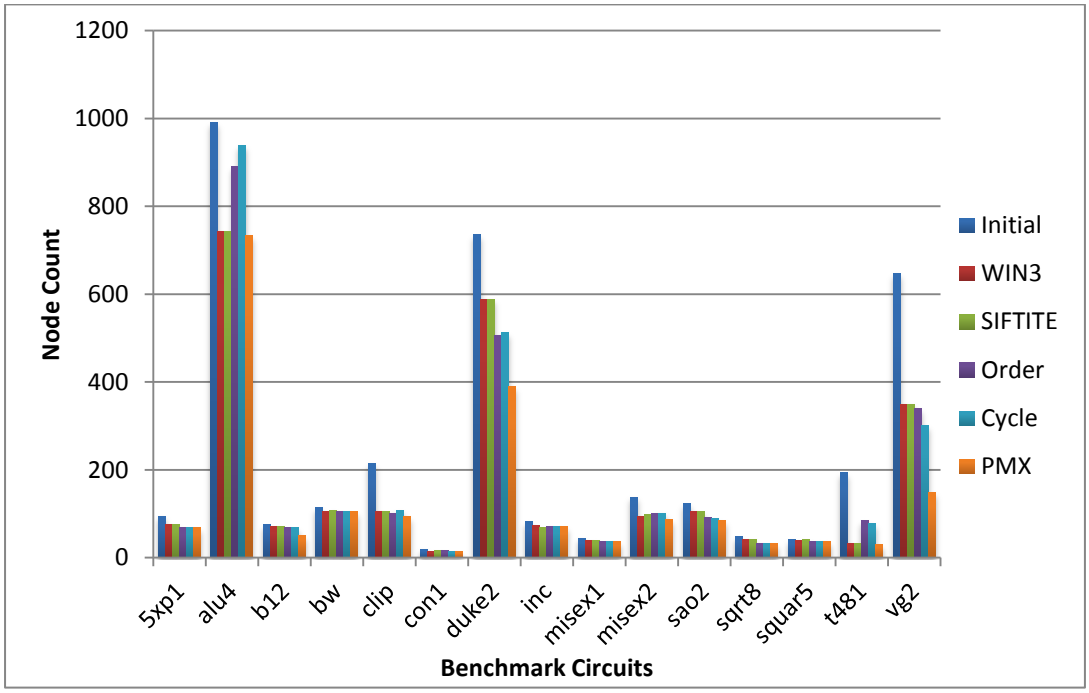

Fig. 8.1. Node count comparison of best from sifting, window, proposed genetic for IWLS'93 benchmarks

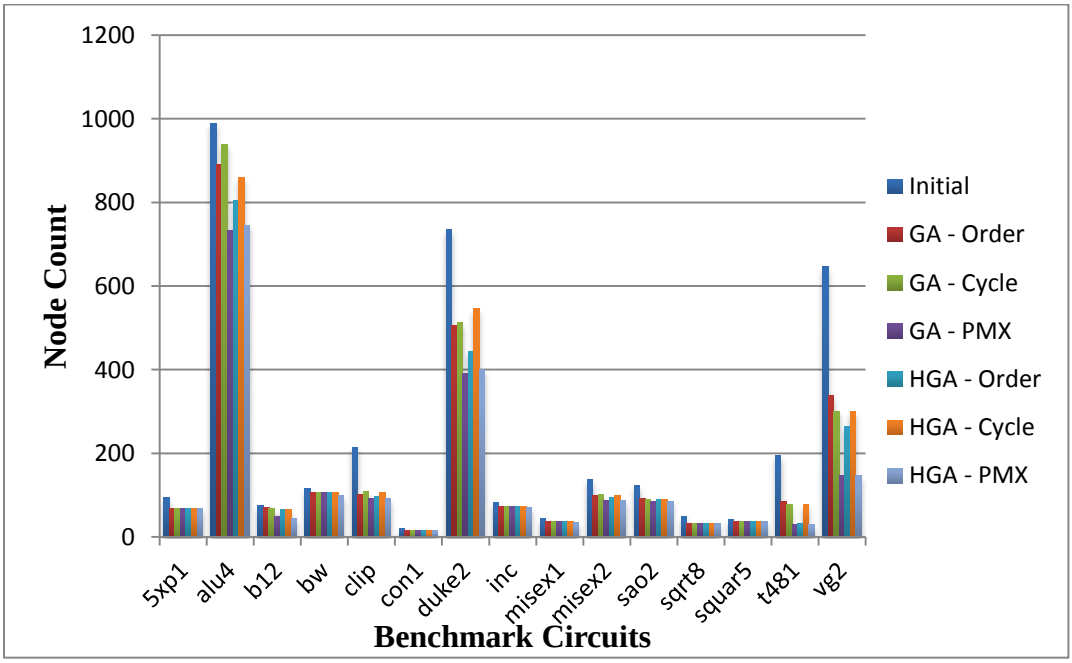

Fig. 8.2. Node count comparison of proposed genetic and proposed hybridized genetic for IWLS'93 benchmarks

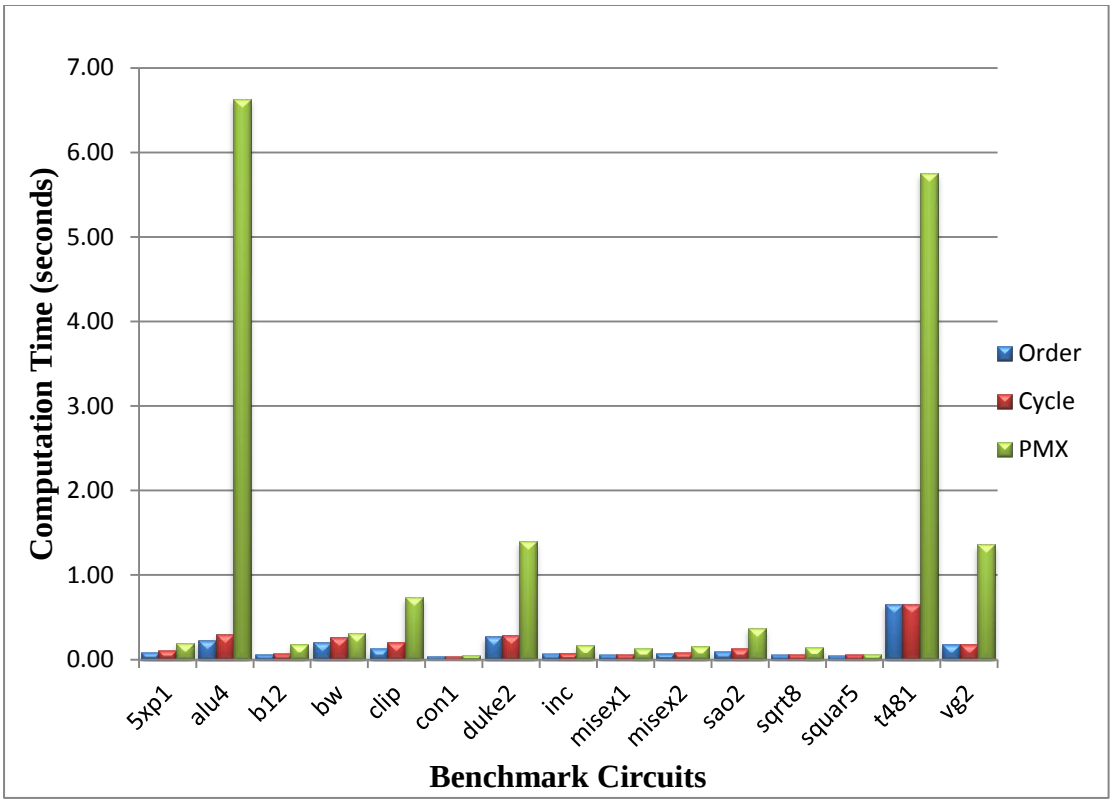

Fig. 8.3. Computation time (seconds) comparison for proposed genetic (order, cycle, PMX) for IWLS'93 benchmarks

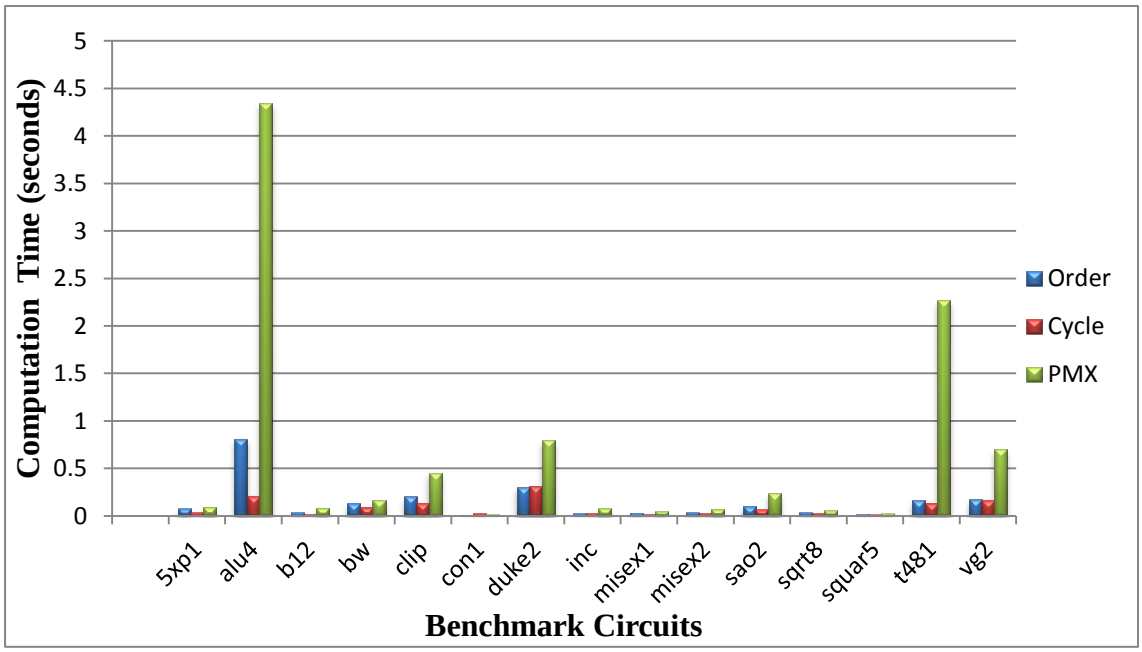

Fig. 8.4. Computation time (seconds) comparison for proposed hybridized genetic (order, cycle, PMX) for IWLS'93 benchmarks

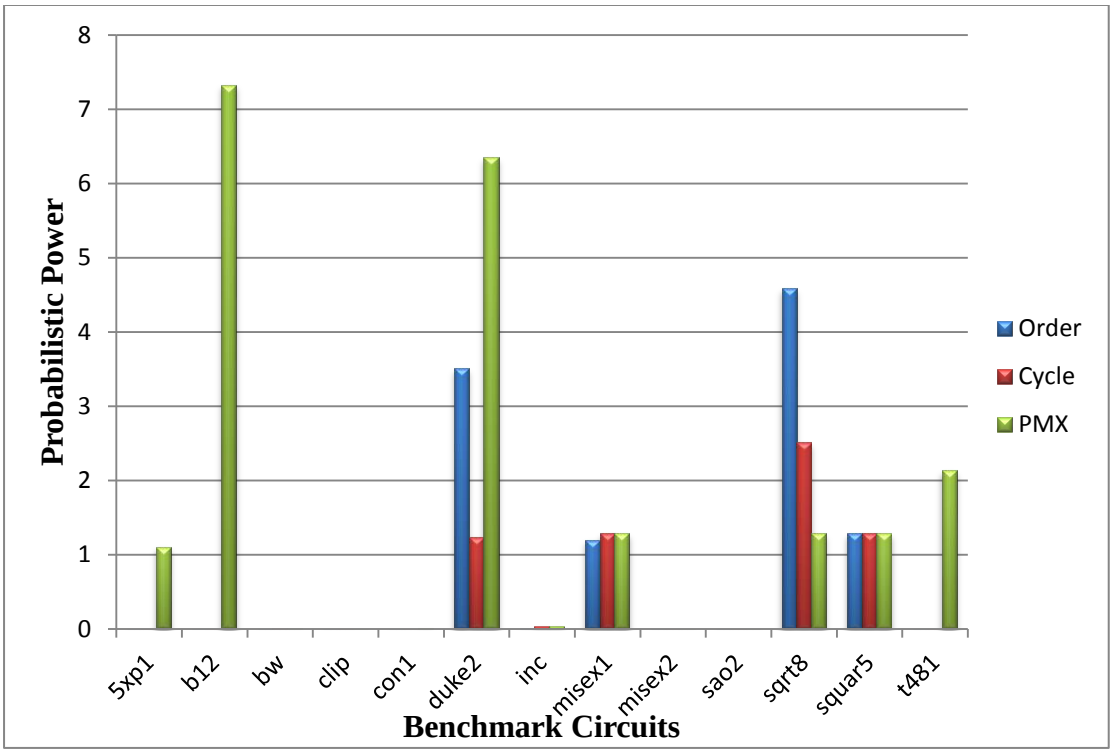

Fig. 8.5. Probabilistic Power comparison for proposed genetic (order, cycle, PMX) for IWLS'93 benchmarks

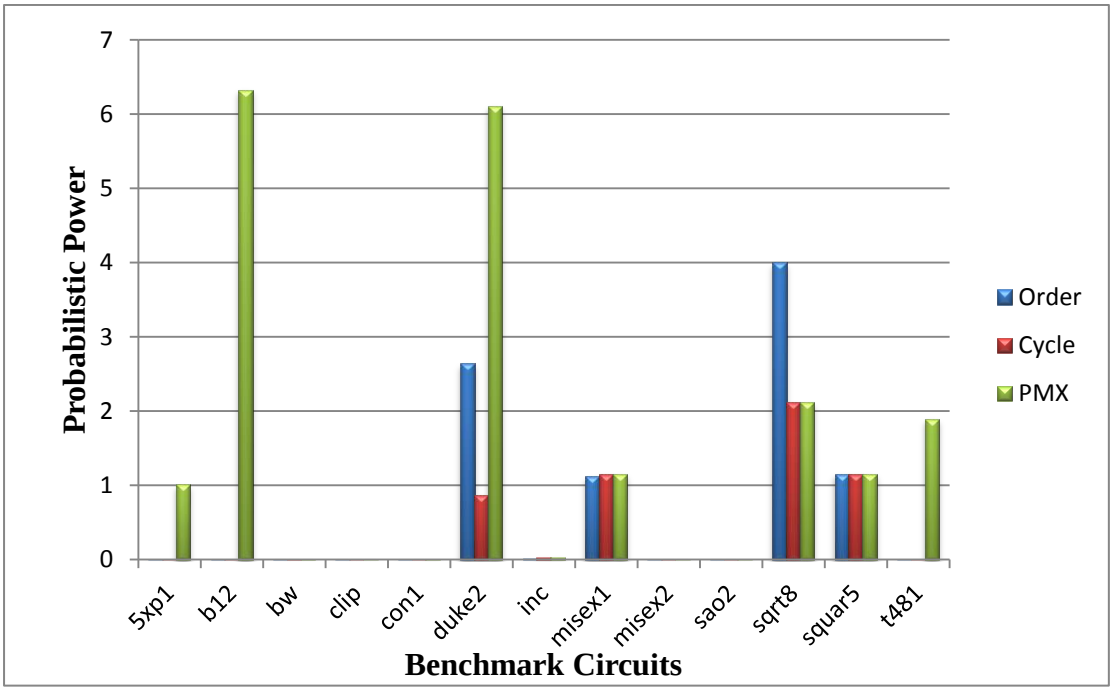

Fig. 8.6. Probabilistic Power comparison for proposed hybridized genetic (order, cycle, PMX) for IWLS'93 benchmarks

8.3. FUTURE SCOPE

- The results obtained can be used as a good initial solution as inputs to the other optimization algorithms in order to achieve multi-objective optimization.
- The proposed algorithms can be modified to work with consideration to timing and leakage concerns and parasitic capacitances.
- Besides achieving optimization in terms of area, speed and power, the approach can be extended and combined with other approaches to include the memory optimization also, thereby, enhancing the scope of performance.
- The proposed algorithms can be extended to the optimization of Sequential Circuits.

References

[1]. S. B. Akers, "Binary Decision Diagrams," in *IEEE Transactions on Computers*, vol. C-27, no. 6, pp. 509-516, June 1978. doi: 10.1109/TC.1978.1675141

[2]. C. Y. Lee, "Representation of switching circuits by binary-decision programs," in *The Bell System Technical Journal*, vol. 38, no. 4, pp. 985-999, July 1959. doi: 10.1002/j.1538-7305.1959.tb01585.x

[3]. R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," in *IEEE Transactions on Computers*, vol. C-35, no. 8, pp. 677-691, Aug. 1986. doi: 10.1109/TC.1986.1676819

[4]. G. Cabodi, P. Camurati and S. Quer, "Improving the efficiency of BDD-based operators by means of partitioning," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 5, pp. 545-556, May 1999. doi: 10.1109/43.759068

[5]. R. Drechsler, M. Kerttu, P. Lindgren and M. Thornton, "Low power optimization techniques for BDD mapped circuits using temporal correlation," in *Canadian Journal of Electrical and Computer Engineering*, vol. 27, no. 4, Oct 2002.

[6]. I. Furdu and B. Patrut, "Genetic algorithm for ordered decision diagrams optimization," in *Proc. ICMI 45*, Bacau, 2006.

[7]. O. Roeva, S. Fidanova and M. Paprzycki, "Influence of the population size on the genetic algorithm performance in case of cultivation process modelling," *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on*, 2013, pp. 371-376.

[8]. W. Lenders and C. Baier, "Genetic algorithms for the variable ordering problem of binary decision diagrams," *Foundations of Genetic Algorithms, 2005 8th International Conference on*, 2005, pp. 1-20.

[9]. S. Rehan and M. Bansal, “Performance comparison among different evolutionary algorithms in terms of node count reduction in BDDs,” in *International Journal of VLSI and Embedded Systems*, vol. 4, Jul 2013.

[10]. S. I. Minato, *Binary decision diagrams and applications for VLSI CAD,* Vol. 342, 2012, Springer Science & Business Media.

[11]. O. Grumberg, S. Livne and S. Markovitch, “Learning to order BDD variables in verification,” in *Journal of Artificial Intelligence*, vol. 18, pp. 83-116, 2003.

[12]. F. A. Aloul, I. L. Markov and K. A. Sakallah, “MINCE: A static global variable-ordering heuristic for SAT search and BDD manipulation,” in *Journal of Universal Computer Science*, vol. 10, no. 12, pp. 1562-1596, 2004.

[13]. P. Y. Chung, I. M. Hajj and J. H. Patel, "Efficient variable ordering heuristics for shared ROBDD," *Circuits and Systems, 1993., ISCAS '93, 1993 IEEE International Symposium on*, Chicago, IL, 1993, pp. 1690-1693 vol.3. doi: 10.1109/ISCAS.1993.394067

[14]. M. Fujita, H. Fujisawa and Y. Matsunaga, "Variable ordering algorithms for ordered binary decision diagrams and their evaluation," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 1, pp. 6-12, Jan 1993. doi: 10.1109/43.184839

[15]. H. Fujii, G. Ootomo and C. Hori, "Interleaving based variable ordering methods for ordered binary decision diagrams," *Computer-Aided Design, 1993. ICCAD-93. Digest of Technical Papers, 1993 IEEE/ACM International Conference on*, Santa Clara, CA, USA, 1993, pp. 38-41. doi: 10.1109/ICCAD.1993.580028

[16]. R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," *Computer-Aided Design, 1993. ICCAD-93. Digest of Technical Papers, 1993 IEEE/ACM International Conference on*, Santa Clara, CA, USA, 1993, pp. 42-47. doi: 10.1109/ICCAD.1993.580029

[17]. C. Meinel, F. Somenzi and T. Theobald, "Linear Sifting Of Decision Diagrams," *Design Automation Conference, 1997. Proceedings of the 34th*, Anaheim, CA, USA, 1997, pp. 202-207. doi: 10.1109/DAC.1997.597144

[18]. S. J. Friedman and K. J. Supowit, "Finding the Optimal Variable Ordering for Binary Decision Diagrams," *Design Automation, 1987. 24th Conference on*, 1987, pp. 348-356. doi: 10.1109/DAC.1987.203267

[19]. N. Zhuang, M. S. T. Benten and P. Y. K. Cheung, "Improved variable ordering of BDDs with novel genetic algorithm," *Circuits and Systems, 1996. ISCAS '96., Connecting the World., 1996 IEEE International Symposium on*, Atlanta, GA, 1996, pp. 414-417 vol.3. doi: 10.1109/ISCAS.1996.541621

[20]. S. Chaudhury and A. Dutta, "Algorithmic optimization of BDDs and performance evaluation for multi-level logic Circuits with area and power trade-offs," in *Circuits and Systems*, vol. 2, no. 3, pp. 217-224, Jul 2011.

[21]. N. Ishiura, H. Sawada and S. Yajima, "Minimization of binary decision diagrams based on exchanges of variables," *Computer-Aided Design, 1991. ICCAD-91. Digest of Technical Papers, 1991 IEEE International Conference on*, Santa Clara, CA, USA, 1991, pp. 472-475. doi: 10.1109/ICCAD.1991.185307

[22]. G. Fey and R. Drechsler, "Minimizing the number of paths in BDDs," *Integrated Circuits and Systems Design, 2002. Proceedings. 15th Symposium on*, 2002, pp. 359-364. doi: 10.1109/SBCCI.2002.1137683

[23]. M. Kerttu, P. Lindgren, M. Thornton and R. Drechsler, "Switching activity estimation of finite state machines for low power synthesis," *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, 2002, pp. IV-65-IV-68 vol.4. doi: 10.1109/ISCAS.2002.1010389

[24]. N. Sathya and A. Muthukumaravel, "A review of the optimization algorithms on travelling salesman problem," in *Indian Journal of Science and Technology*, vol. 8, no. 29, Nov 2015.

[25]. M. Takapoo and M. B. Ghaznavi-Ghoushchi, "IDGBDD: The novel use of ID3 to improve Genetic algorithm in BDD reordering," *Electrical Engineering/Electronics Computer Telecommunications and Information Technology (ECTI-CON), 2010 International Conference on*, Chiang Mai, 2010, pp. 117-121.

[26]. M. D. B. Siddiqui and M. Bansal, “BDD ordering: A method to minimize BDD size by using improved initial order,” in *International Journal of VLSI and Embedded Systems*, vol.4, no. 3, Jun 2013.

[27]. P. W. C. Prasad, A. Assi, A. Harb and V. C. Prasad, “Binary decision diagrams: An improved variable ordering using graph representation of Boolean functions,” in *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 2, no. 2, 2008.

[28]. G. Sharma, “Algorithmic reduction and optimization of logic circuit in area and power tradeoffs’ with the Help of BDD,” in *International Journal of Engineering and Computer Science*, vol. 3, no. 5, pp. 6132-6139, May 2014.

[29]. I. Wegener, “Branching programs and BDDs,” in *Journal of Theoretical Computer Science*, vol. 141, no. 1-2, pp. 283 - 310, Apr 1995.

[30]. S. S. Lin and C. J. Wei, "A new approach for minimization of binary decision diagrams," in *Canadian Journal of Electrical and Computer Engineering*, vol. 30, no. 4, pp. 207-214, 2005. doi: 10.1109/CJECE.2005.1541753

[31]. W. Mingquan and Y. Haibin, "BDD minimization based on genetic tabu hybrid strategy," *2005 6th International Conference on ASIC*, Shanghai, 2005, pp. 948-952.doi: 10.1109/ICASIC.2005.1611465

[32]. F. Towhidi, A. Lashkari and R. S. Hosseini, "Binary Decision Diagram (BDD)," *Future Computer and Communication, 2009. ICFCC 2009. International Conference on*, Kuala Lumpar, 2009, pp. 496-499. doi: 10.1109/ICFCC.2009.31

[33]. O. Brudaru, R. Ebendt and I. Furdu, "Optimizing Variable Ordering of BDDs with Double Hybridized Embryonic Genetic Algorithm," *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2010 12th International Symposium on*, Timisoara, 2010, pp. 167-173. doi: 10.1109/SYNASC.2010.33

[34]. T. Bhuvaneswari, V. Prasad, A. K. Singh and C. Senthilpari, “Performance Analysis of Reversed Binary Decision Diagram Pass Transistor Logic Synthesis,” in *International Journal of Circuit Theory and Applications*, 2011.

[35]. C. L. Valenzuela and P. Y. Wang, "A Genetic Algorithm for VLSI Floorplanning," *Innovations In Intelligent Instrumentation, Optimization and Electrical Sciences, 2013 IJCA Proc. International Conference,* 2013.

[36]. J. Jiang, H. Liu, H. Feng and F. Meng, "Embedded Static Task Allocation and Scheduling based on Simulated Annealing and Genetic Algorithm," in *Journal of Computation Information Systems*, pp. 1465-14724, 2014.

[37]. R. Kaur and M. Bansal, "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, no. 3, Mar 2014.

[38]. K. Singh, S. Singh and R. Singh, "Hybrid Technique Based VLSI Design Automation," in *International Journal of Modern Communication Technologies and Research*, vol. 2, no. 10, pp. 2321-0850, Oct 2014.

[39]. F. Somenzi. *Binary Decision Diagrams*. Available: http://www.ecs.umass.edu/ece/labs/vlsicad/ece667/reading/somenzi99bdd.pdf

[40]. A. A. A. Esmin and S. Matwin, "HPSOM: A hybrid particle swarm optimization algorithm with genetic mutation," in *International Journal of Innovative Computing, Information and Control*, vol. 9, no. 5, pp. 1919-1934, May 2013.

[41]. S. Gotshall and B. Rylander. *Optimal population Size and the genetic algorithm*. Available:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.417.1552&rep=rep1&type=pdf

[42]. *BuDDy: A Binary Decision Diagram Package*. [Online]. Available: http://sourceforge.net/projects/buddy.

List of Publications

1. S. Jindal and M. Bansal, "A Novel and Efficient Variable Ordering and Minimization Algorithm Based on Evolutionary Computation," in *Indian Journal of Science and Technology*, 2016. (Communicated)

2. S. Jindal and M. Bansal, "A Proficient Ordering of Binary Decision Diagrams for Multi-objective Optimization of VLSI Circuits based on Evolutionary Algorithm," in *Journal of Computational and Applied Mathematics*, 2016. (Communicated)

3. S. Jindal and M. Bansal, "An Improved Approach for Variable Ordering and Optimisation of VLSI Circuits using Effective Genetic Algorithm," in *Defence Science Journal*, 2016. (Communicated)

601461028

ORIGINALITY REPORT

11%	7%	7%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	ijves.com Internet Source	3%
2	www.ablongman.com Internet Source	1%
3	P. W. C. Prasad. "Binary Decision Diagrams and neural networks", The Journal of Supercomputing, 03/16/2007 Publication	1%
4	www.ijecs.in Internet Source	1%
5	waset.org Internet Source	1%
6	Chaudhury, Saurabh, and Anirban Dutta. "Algorithmic Optimization of BDDs and Performance Evaluation for Multi-level Logic Circuits with Area and Power Trade-offs", Circuits and Systems, 2011. Publication	1%
7	Wang, Yong, Qiang Dou, Wei Peng, and Zheng	