IMPLEMENTATION OF EVOLUTIONARY ALGORITHM WITH EFFECTIVE USE OF CROSSOVER OPERATORS FOR BDD MAPPED CIRCUITS

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

Master of Technology

In

VLSI Design

Submitted by

Rohit Kumar Sharma

Roll. No. 601361021

Under the supervision of

Mrs. Manu Bansal

Assistant Professor, ECED

Department of Electronics & Communication Engineering

Thapar University, Patiala

INDIA

June, 2015

CERTIFICATE

I hereby declare that the work which is being presented in the dissertation entitled, **"Implementation of Evolutionary Algorithm with Effective Use of Crossover operators for BDD Mapped Circuits"** in partial fulfilment 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.

RKSharma

Date: 14/07/15

Rohit Kumar Sharma
Roll.No. 601361021

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

MBansal

Mrs. Manu Bansal
Assistant Professor
ECE Department
Thapar University

Counter signed by:

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

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. Amit Kumar Kohli, 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.

Rohit Kumar Sharma

ABSTRACT

Binary Decision Diagrams are Data structure for representation and manipulation of Boolean functions. It is a canonical representation based on recursive Shannon expansion. The size of the BDD is hugely depends upon the order of the input variables. In BDD's calculations, the major problem deals with calculating the fine order of variables. Since in VLSI Design, area of the logic circuit should be minimal as much as possible to reduce the chip size. There are many approaches such as Static, Dynamic and Heuristic techniques exist for obtaining the optimum order of variables in BDDs. Basic genetic algorithm is an efficient technique to find solutions of such problem. Crossover operators used in genetic algorithm also play significant role in finding good result. So, a crossover operator used for a problem should choose wisely. The main idea of this article is based on the efficient use of various crossover operators in Genetic and Hybrid Genetic algorithm for optimizing the variable order in BDDs.

The applied Hybridized Genetic Algorithm is combination of the Genetic Algorithm and the Branch and Bound Algorithm whose main purpose is to search for the best solution of an optimization problem efficiently. Genetic Algorithm is a computational model which is inspired by evolution. When there is no mathematical analysis of a problem is available then GA is very useful technique. Branch and Bound technique set the upper and lower limit of solutions of a given problem. The proposed technique is applied on the LGSynth93 benchmark circuits with an aim to reduce the node count. Hence, as a result, the proposed methods provide optimal results for most of the benchmark circuits. Up to 82.02% reduction in area is found in 8 bit adder circuit. Comparison tables have been presented to show the effectiveness of the proposed algorithm as compared to other previously implemented state-of-art techniques for variable ordering of BDDs.

List of Contents

List of Abbreviations

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

List of Figures

List of Tables

Chapter 1
Introduction

Within the last 10-13 years, for representation and manipulation of Boolean functions Binary decision diagrams (BDDs) have become the state-of-art data structure in the designing tools of VLSI. Especially in the field of verification and synthesis, the use of BDDs in the commercial tools is growing day by day.

The advantage of the BDDs is that the data structure is generally accepted as providing a good compromise between conciseness of representation and efficiency of manipulation. With increasing number of applications, in CAD and also in non CAD areas, there are many methods and techniques are evolving at great pace to improve the handling of BDDs.

1.1 Binary Decision Diagram

For representation and manipulation of Boolean functions, in 1959, Lee introduced Binary Decision Diagrams (BDDs) as a data structure for Boolean functions. Later Akers popularized this data structure by showing how BDD can be handled efficiently. The importance of the Binary Decision Diagrams increased when Braynt, in 1985/86, introduced the concept of Ordered Binary Decision diagrams and demonstrated that BDDs in this restricted form, within the following years the importance of Binary Decision Diagrams for VLSI CAD was realized by several groups and increase number of BDD algorithms and successful applications were reported. In 1990 the first BDD packages, SW tools for the representation and manipulation of BDDs, become available. Today, BDDs have become the state–of-art data structure in VLSI CAD generally accepted as providing a good compromise between conciseness of representation and efficiency of manipulation.

A Binary Decision Diagrams is a well define data structure that is used to represent a Boolean function [1]. On a more abstract level, BDDs can be considered as a compressed representation of sets or relations. BDD is called a directed acyclic graph because there is no way to loop back to the same edge.

The properties of a BDD are:

1. A vertex in graph V is either a non-terminal or a terminal vertex.

2. Each non-terminal vertex (v) is labelled with a variable from Xn and has two successors in V, denoted by low (v), high (v).
3. Each terminal vertex "v" is labelled with value (v) belongs to T and has no successors.

The edge between v and low (v) is also known as low edge of v and edge between v to high (v) known as high-edge [2]. As terminal set in general T = {0, 1}. The size of the BDD is defined by the number of nodes present in the binary decision diagram.

A simple BDD of a random function F(a,b,c) = (a+b+c) is shown in figure 1.1.

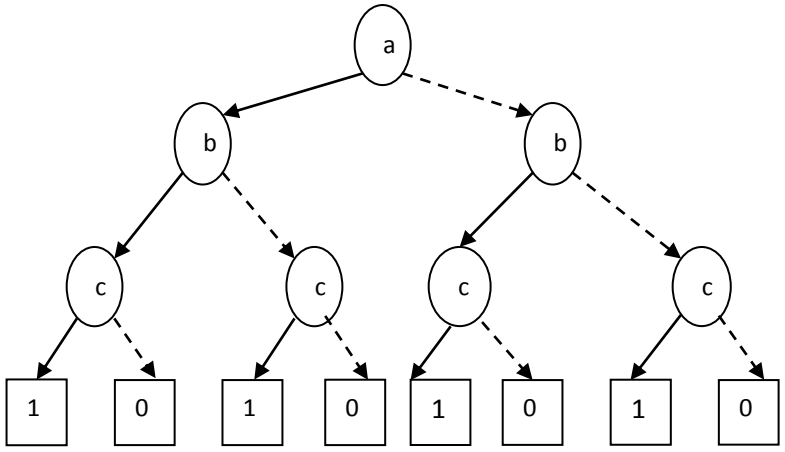


Figure 1.1 BDD for F(a,b,c) = (a+b+c)

1.1.1 OBDD

If in the BDD, at one level only one type of variable present i.e .if all different variables in the Boolean function are present at the same level in BDD then this type of BDD is called Ordered BDD. Every variable encountered at most one if we go from root node to the terminal node and the order of the variables remain same throughout the graph [2, 4]. Ordered BDD was proposed by Bryant in 1986. Bryant showed that number of nodes is different for same Boolean function if order of the variables is different. After that so many techniques were evolved for finding good variable order to reduce the size of BDD. Thus, OBDD is a restricted decision graph in which the ordering of variables is consistent on all paths of the graph. The order selected for decision variables can significantly affect the number of nodes required in

a BDD. The BDD in figure 1.2 uses variable order "a < b < c < d", while the BDD in figure 1.3 represents the same function, with variable order "a < c < d < b". Both figures are representing the same Boolean function but due to the different variable ordering, the number of node is different. That is, the diagram is unique for a given logic function and for a given variable order.

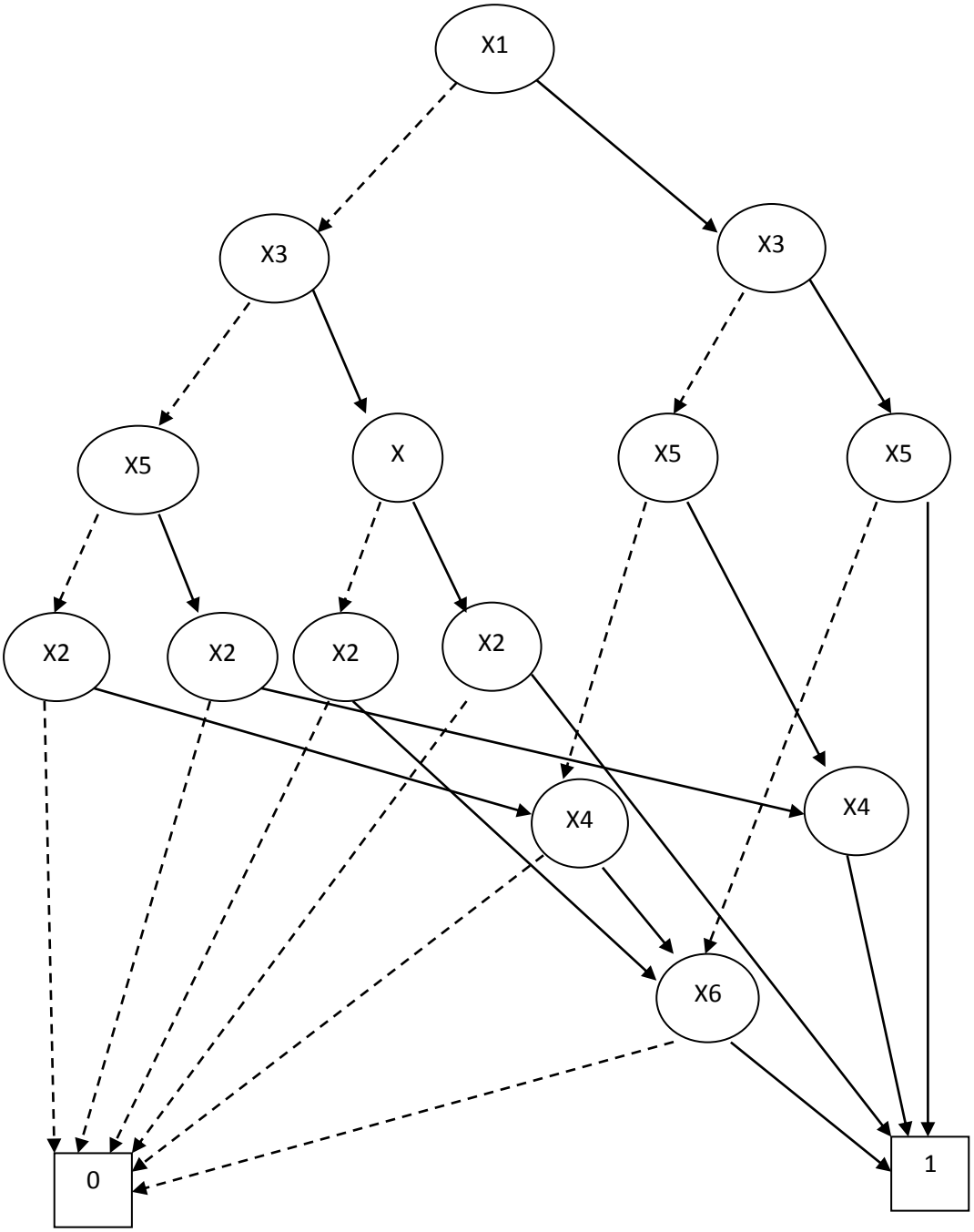


Figure 1.2 F = (x1+x2).(x3+x4).(x5+x6) with order [X1,X3,X5,X2,X4,X6]

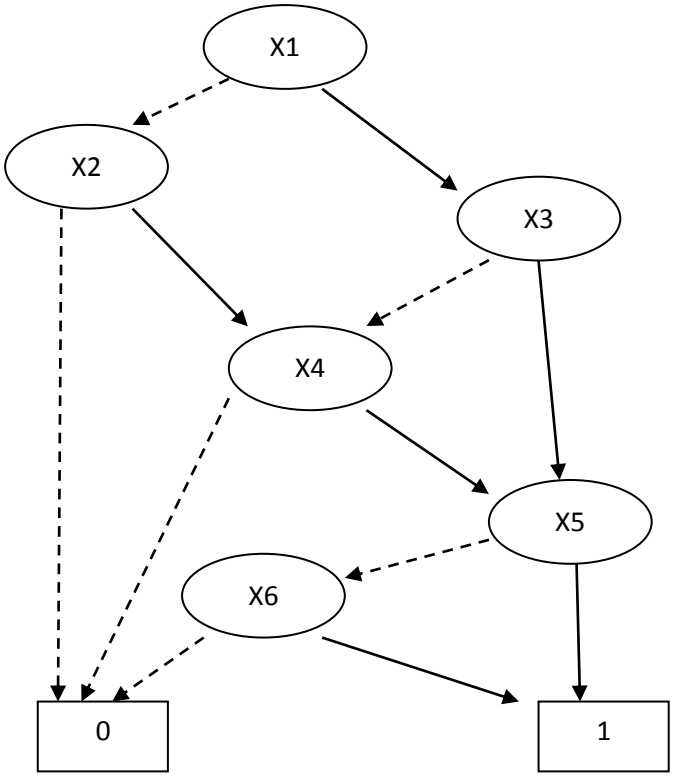


Figure 1.3 F = (x1+x2).(x3+x4).(x5+x6) with order [X1,X2,X3,X4,X5,X6]

1.1.2 ROBDD

The size of the OBDD is further reduced if it contains the redundant nodes or identical node. If all identical nodes are shared and all redundant nodes are eliminated, the OBDD is said to be reduced (an ROBDD) [2]. There are some restrictions which are imposing on the BDD to represent it in ROBDD form. After applying restrictions, canonical form of the data structure remains in the graph. A BDD is said to be reduced, if it contains no isomorphic sub graphs, and every vertex has distinct children. Reduced, ordered BDDs are canonical representations of Boolean functions: for a fixed variable order, two functions are equivalent if and only if their BDDs are identical [5]. Redundant nodes in the graph, i.e. nodes not needed to represent a Boolean function, can be eliminated. The main advantage of the ROBDD is that it requires much less memory as compared to other techniques to represent Boolean functions such as Truth tables or Karnaugh maps etc., and it is a faster algorithm for their manipulation.

An OBDD is converted to an ROBDD using the following reduction rules:

1. Merging equivalent nodes
2. Merging isomorphic nodes
3. Eliminating redundant tests

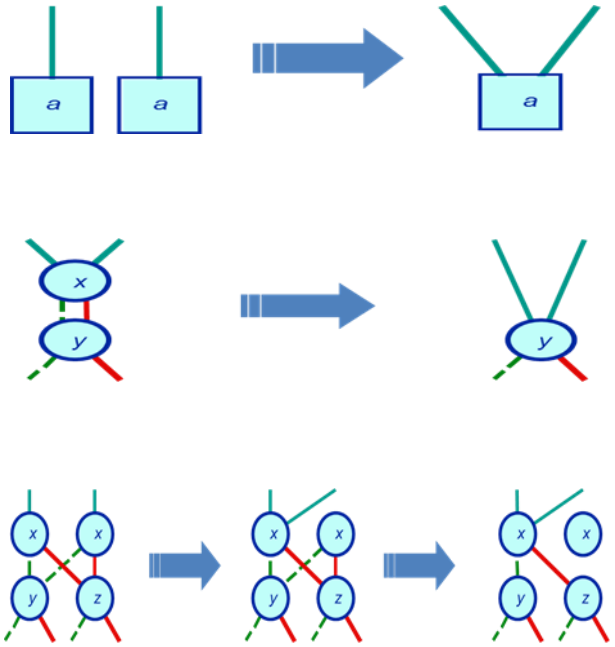


Figure 1.4 Reduction Rules

1.1.3 Applications of BDD:

Binary Decision Diagrams (BDDs) have achieved vast popularity as an efficient data structure for representing Boolean functions in solving and manipulating most of the combinational problems which arise in synthesis and verification of digital systems [1]. The two important properties of BDDs which form the basis for applications of BDDs in different areas are:

1. Since BDDs occupy canonical property. This property is useful when verifying the equivalence between two given logic circuits. According to this, two circuits are equivalent if and only if their BDDs are identical for a specific variable ordering.

2. BDDs can be used as effective structures for the representation of large combinatorial sets.

1.1.3 (a) Functional Synthesis: BDDs as MUX Circuits

Boolean functions, represented by BDDs can be realized using multiplexer based design styles such as Pass Transistor Logic. Pass Transistor Logic uses only two transistors to realize a multiplexer a wired OR of 2 MOS transistors. The advantages of Pass Transistor Logic circuits as an alternative to static CMOS design are minimized circuit area, low power dissipation and circuit speed is high. Each node in the corresponding BDD for a Boolean function represents a 2x1 MUX. In 2x1 multiplexer when select line ‘s’ is set to zero then output ‘y’ is equal to ‘a’ and when ‘s’ is set to one then output is ‘b’ which is shown in figure 1.5 and is expressed logically as y = s’a + sb. The MUX representation is obtained by back propagation from terminal nodes to the root nodes of the BDD, as shown in Fig 1.6. Hence, lesser node-count is desirable in order to reduce the net area for realization and fabrication of a digital circuit.

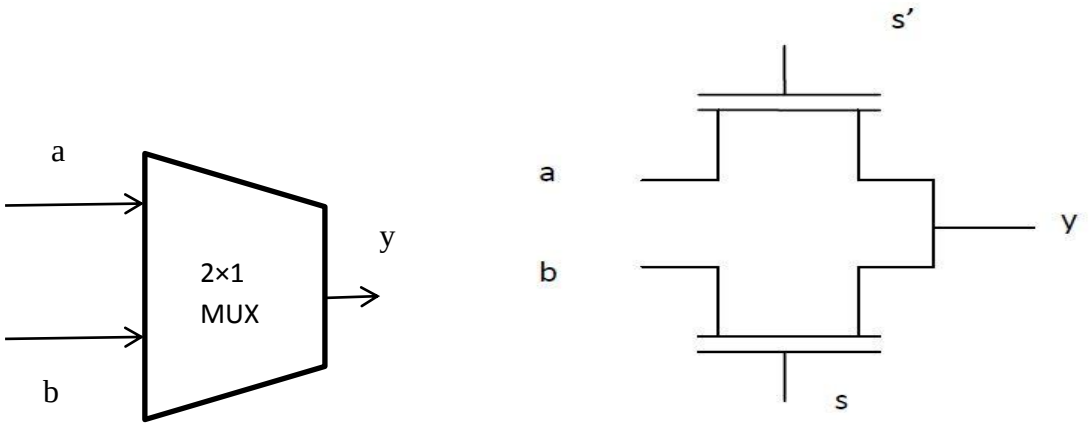


Figure 1.5 2x1 Multiplexer and corresponding transistor realization

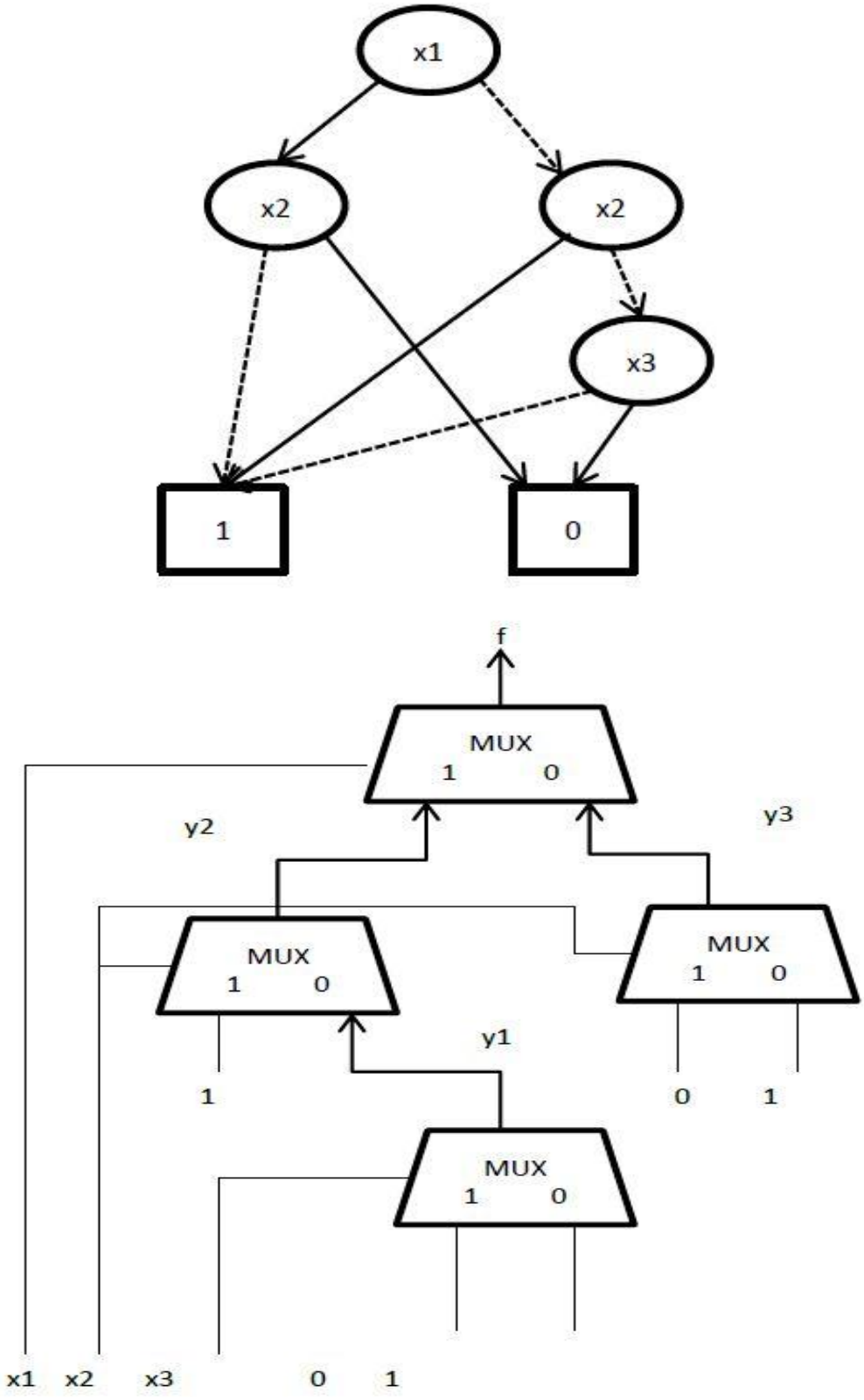
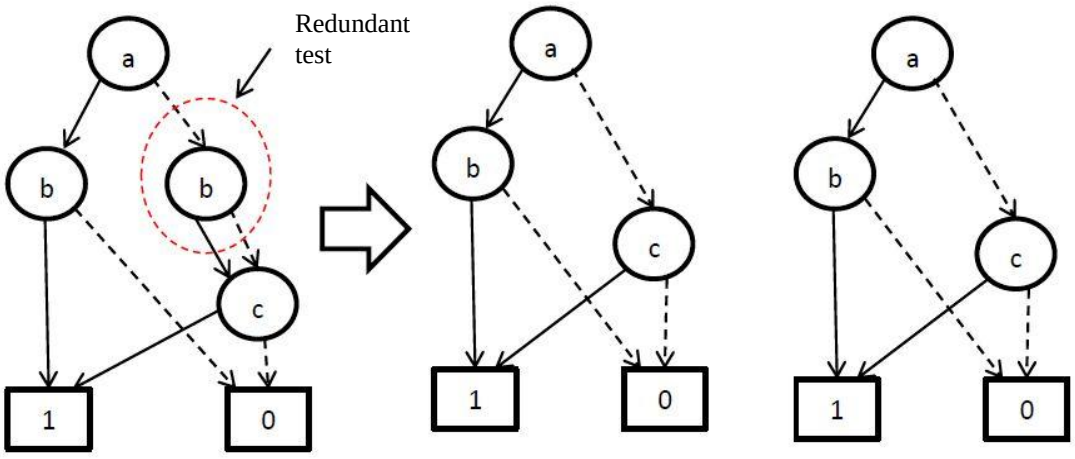


Figure 1.6 ROBDD and corresponding MUX representation for Boolean function,F = x1x2'+ x1' (x2'x3'+ x2)

1.1.3 (b) Functional Verification of Logic Circuits

Binary decision diagrams (BDDs) were originally invented for hardware verification to efficiently store a large number of states that share many commonalities. The canonical property of BDDs makes them an efficient alternate approach for logic circuit comparison. The basic aim for hardware verification is to compare a new design to a known good design. Two logic circuits are equivalent if and only if their compact representations in the form of BDDs are identical provided the same variable ordering is used in both representations as illustrated in Fig. 1.7. Fig. 1.7 (a) shows the BDD for Boolean function F1 = ab+a‟c+bc. On applying the reduction rules, i.e. eliminating redundant test, ROBDD for function F1 is obtained with variable order a < b < c as shown in Fig.1.7 (b). ROBDD for function F2 = ab+a‟c with order a < b < c is shown in Fig. 1.7 (c). Both figures being identical show that the two Boolean functions F1 and F2 are identical.



(a) **(b)** **(c)**

Figure 1.7 Equivalence Verification of functions F1 = ab+a'c+bc and F2 = ab+a'c

1.2 Evolution Algorithm

In the past, to find solutions of the problems which are not efficiently solved by the mathematical tool, researchers apply a natural evolution process on the problems to find

optimum results. It is also a part of artificial intelligence. It is also called evolutionary computation. Evolution algorithm is a meta-heuristic optimization algorithm which is based upon generic population. The main concepts of Evolution Algorithm are inspired with biological evolution, such as selection, reproduction, mutation and recombination [8]. Candidates from the solutions of the optimization problem play the role of individuals in a population, and the fitness function determines the environment within which the solutions live. Genetic algorithm is one type of the Evolutionary Algorithm. An Evolutionary algorithm can be hybridized with other evolutionary or dynamic algorithms to find optimized results. The objective of Hybridized Genetic Algorithm is to find an optimal ordering of BDDs with reduced node count.

1.3 Simple Genetic Algorithm

Genetic Algorithms (GA's) are very powerful tools for optimization problems. Due to its properties like robustness and flexibility, GA becomes obvious choice for the user for finding solutions of the optimization problem when there is large search area or multiple objectives are present. It produces sub-optimal solutions at much faster rate than mathematical methods.

Genetic algorithm involves a process of creating a new population (i.e. set of possible solutions) from previous population with the help of genetic inspired operators of selection, crossover, mutation and inversion. Population is set of chromosomes. The strings of '1's and '0's is the simple form of representing a population. The combination of genes form a chromosome. Every chromosome made up of genes (e.g., bit), each gene being an instance of a particular allele. Chromosomes are chosen wisely from the population for generating next population. Selection operator is used for this purpose. Selected chromosomes act as parents of next chromosomes. Selection is based upon the fitness of individual chromosome. Fitness function is defined for every problem before genetic algorithm is applied. More fitness value of chromosome means better the solution or next chromosome. The chromosome generate from its parents by applying the crossover operator on the parents. Crossover operator simply exchanges the genes of parents and form a new chromosome by mixing them. To give new properties to the new chromosome, mutation is performed on it. Mutation simply means that changing a gene with some probability.

The major advantage of the evolution process is that it uses parallel search method rather than work on one species at a time to find the optimal solutions of the given

problem. The rules of these techniques are very simple to follow, new species are produce by means of random variation in mutation, recombination, and other operators. The basic thing behind obtaining the optimal solutions from these techniques is that only fit chromosomes are remained in population, so with every new population we move towards better solutions.

1.3.1 GA Operators

The simplest form of genetic algorithm involves three types of operators: selection, crossover and mutation [8].

1. **Selection:** The purpose of this operator is to select chromosomes from population as parents for reproduction i.e. generating new population. A good selection operator selects chromosome which has high fitness value.
2. **Crossover:** This operator exchanges the genes of two parent to form a new chromosome by randomly choosing a locus and exchanges the subsequence before and after that locus between selected chromosomes to create offspring. For example, the strings "10000100" and "11111111" are two parent chromosome. Let they are crossed over after the third locus. The two offspring "10011111" and "11100100" are generated by simply using this.
3. **Mutation:** This operator adds new property by randomly--flipping some of the bits in a chromosome. For example, the string "00000100" mutated at its second--position to form "01000100". Mutation can be done at each bit position in a string with some defined probability; usually this probability keeps very small.

1.4 Branch and Bound:

Branch and Bound is an excellent optimization technique for multi-objective problems. Branch and Bound (BB) is basically a type of traditional greedy approach technique [11]. It is a search tree graph, in which each node except the root node corresponds to the sub problem of the original problem while root node represents the original problem. This technique gives large number of feasible solutions but in finite number. Let us take an example, Let 'v' be a node of the search tree, the children of 'v' are sub problems derived from 'v' by imposing a new constraint for every sub problem. These descendants i.e. sub-problems of 'v' satisfy the same constraints as 'v'. In each iteration of a Branch and Bound algorithm, using some selection strategy one node is chosen for exploration from the set of live nodes corresponding

to unexplored feasible sub problems. The strategy behind exploring is branching, which is performed by constructing two or more children of the node through the addition of constraints to the sub-problem of the node. In this way the subspace is subdivided into smaller subspaces.

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:

SHELDON B. AKERS (1978) [1]

This paper describes the usefulness of binary decision diagram for defining, analysing, testing, and implementing large digital functions and systems. There are number of examples given in the paper for a number of basic combinational and sequential devices. Techniques are then outlined for using the diagrams to analyse the functions involved, for test generation, and for obtaining various implementations. In this paper author describe in detail how test generation can perform using BDDs.

Randal E. Bryant (1986) [2]

In this paper, author describes how a Boolean function can be represented in a new data structure using directed, acyclic graphs. Restrictions are imposed on the ordering of decision variables of the graph. In this paper Binary Decision Diagram has been represented as a new data structure for representing Boolean functions. The paper also has presented some manipulation algorithms to manipulate the graphs. The algorithms have been applied on problems in logic design verification and their experimental results also presented in the paper.

Shin-ichi MINATO, Nagisa ISHIURA and Shuzo YAJIMA (1990) [3]

This paper has described a technique for more efficient Boolean function manipulation that uses Shared Binary Decision Diagrams (SBDD's). The edges in the graph are attributed edges. Ordering algorithm has implemented on the SBDD for input variables. The paper has also shown method of handling don't care. Experimental results produced by the implementation of the Boolean function manipulator.

Steven J. Friedman, Kenneth J. Supowit (1990) [4]

In this paper, author has given an algorithm to solve the problem of finding the fine ordering of binary decision diagrams leading to the most compact representation.

Since, the size of the binary decision diagrams is very much dependent on the ordering of the decision variables. For some applications, like differential cascode voltage switch trees, it becomes extremely important to find the ordering leading to the most compact representation.

N. Zhuang, M.S.T. Benten, P.Y.K. Cheung (1996) [7]

The author has given a new algorithm which is based on a novel formulation of the Genetic algorithm for variable ordering of a binary decision diagram. The proposed algorithm has three major dynamic parameters: population size, mutation rate and stop criteria. The proposed algorithm has applied on the benchmark circuits and the experimental results have compared with other previously published results which shows that this proposed algorithm has better efficiency.

Octav Brudaru, Rüdiger Ebendt, Iulian Furdu(2010) [11]

In this paper optimization of "Reduced Ordered Binary Decision Diagrams" is done using a new double hybridized genetic algorithm. In the process, first hybridization has adopted embryonic chromosomes as prefixes of variable orders. The hybridization achieved by merging branch & bound technique with the basic genetic algorithm. Then second hybridization is performed with the sifting algorithm.

Rolf Drechsler, Gorschwin Fey (2006) [12]

In this paper, an algorithm has presented which minimize the number of paths in BDDs. The complexity of circuit and systems design increases rapidly. Authors have described that not only number of nodes effect the performance of BDDs number of paths also effect the performance. So they have proposed a method theoretically how to decrease number of paths in BDDs and also implemented it practically. Experimental results of this paper show the efficiency of the implemented approach.

Orna Grumberg, Shlomi Livne, Shaul Markovitch (2003) [13]

In this paper, an approach has been proposed which finds the optimal order for a BDD. The basic idea behind this approach is that variable ordering algorithm gains experience from the training samples and the knowledge it gained from that experience is used for finding better orders. BDDs are used at a great extent in verification process and for representing a tested model in current scenario of model checking systems. Thus more compact size the BDD, better the performance we get

from the model verifier. The method which is discussed in this paper is based on the experienced learning of pair-precedence classifiers from training models. It leads to a good order. For each training model, evaluation is performed on the number of training sequences. The implemented algorithm was integrated with SMV, which is most widely used in verification systems. The experimental results show significant improvement in reducing the size of BDD. But the disadvantage of this technique is that it consumes a lot of memory of the system.

Shun-Shii Lin,Chun-Jen Wei (2005) [14]

In this paper a new approach for optimal ordering of BDDs has been proposed. This approach has successfully found the optimal variable orderings for almost all reduced ordered binary decision diagrams (ROBDDs) in the LGSynth91 benchmark circuits with up to 500 variables. Before this approach, previous approaches were efficiently solved only those functions which have less than 64 variables. The performance and efficiency of the proposed approach in this paper has illustrated through its implementation on LGSynth91 benchmark circuits. In the end of this paper statistical results and analyses have presented that can be helpful for other people in related research.

Wolfgang Lenders , Christel Baier (2005) [15]

Finding a variable order of the input variables which decrease the size of a given BDD is the big deal. This is the major problem with BDD-based calculations. In this paper, author has discussed the use of genetic algorithms to improve the variable ordering of a given BDD. Author has presented a new crossover technique that turned out to be very useful when genetic is combined with sifting as hybridization technique. Secondly, author also describe of a distance graph approach which can serve as formal frame work and help in the evaluation of fitness function.

Piotr Porwik, Piotr Zaczkowski, Krzysztof Wrobel (2006) [16]

In this paper, for the optimal variable ordering of binary decision diagram a spectral method has presented. With this approach, author improved the complexity in finding the best solution. Results of this paper presented with applied approach have shown the minimization of the number of nodes of a given BDD and decrease in computation time of the executed program than the linear sifting algorithm.

Awinash Kumar, Ajit Kumar, Soumik choudhary and P.Y. Yarde (2010) [17]

In this paper, genetic algorithm has used for determining the optimal ordering of BDDs. The difference in the approach of this paper and previously used heuristic techniques is that in this paper a generalized method for the selection of optimal ordering in GA is used. Authors have given more importance to the selection of population size and representation of order set of chromosomes.

Saurabh Chaudhury, Anirban Dutta (2011) [18]

In this paper, authors have used two algorithms: a genetic algorithm and a branch and bound algorithm. These algorithms are used to find an better input variable order of BDDs. They used BDD package buddy-2.4 which taken care of node reordering by the standard. Experimental results of this article showed a tremendous saving in area and power of logic circuit. They also compared their techniques with other techniques of variable ordering for BDDs and found that it has given superior results.

Rasoul Faraji, Hamid Reza Naji (2014) [19]

In this article, authors have proposed a new architecture to implement genetic algorithm on the reconfigurable embedded system hardware. The main idea of implementing new architecture is to enhance the speed of the algorithm to find optimal solutions. Hardware implementation in this article has used several features of the field programmable array like reuse of resourcing and parallel processing. Authors have introduced a new crossover operator DSO-crossover. The experimental results show that the speed of the algorithm with this new architecture is much better as compared to simple implementation of the algorithm.

Christoph Scholl, Bernd Becker, Andreas Brogle (2001) [20]

In this paper author has given a solution of the problem of multiple variable order. Moreover, author has transformed Reduced Ordered Binary Decision Diagrams (ROBDDs) with different variable orders into a good common variable order using dynamic variable ordering techniques. ROBDD has been used in fault simulation, logic design verification, test generation, and logic synthesis. Paper has also discussed how to reduce the nodes of the ROBDD to increase its performance.

Thangavel Bhuvaneswari (2011) [29]

In this paper, reversed BDD-based pass transistor logic synthesis has been presented for low-power and high-performance circuits without exploiting the canonical property of BDDs. Binary decision diagrams 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.

Chapter 3
Proposed Work

The main purpose of this article is to reduce the node count of BDD mapped circuits in least computation time as much as possible. The whole basic process of optimizing the Binary decision diagram is shown in the following flowchart in figure 3.1.

Buddy package is used to reorder the variables by itself. At first, the output given by Espresso tool i.e. the node equations is given to buddy package. Buddy package reorders the variables and then node count is performed. After that, proposed technique is applied on the various solutions which are generated by buddy package. The proposed technique gives further optimized solutions which transferred to buddy package and this package gives order of variables and node count. This process runs continuously until the consistence solutions are obtained.

The proposed technique is the efficient use of various crossover operators in Genetic and Hybrid genetic algorithms. First, genetic algorithm is applied and its results are checked. Results show the improvement in node count. Hybrid genetic algorithm is used after genetic algorithm whose results show more improvement in the node count.

An Evolutionary algorithm can be hybridized by embedded it with other evolutionary or dynamic algorithms. The basic purpose behind this is to find more optimum solution and increase the performance of the program. In the applied hybridized Genetic Algorithm, 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.

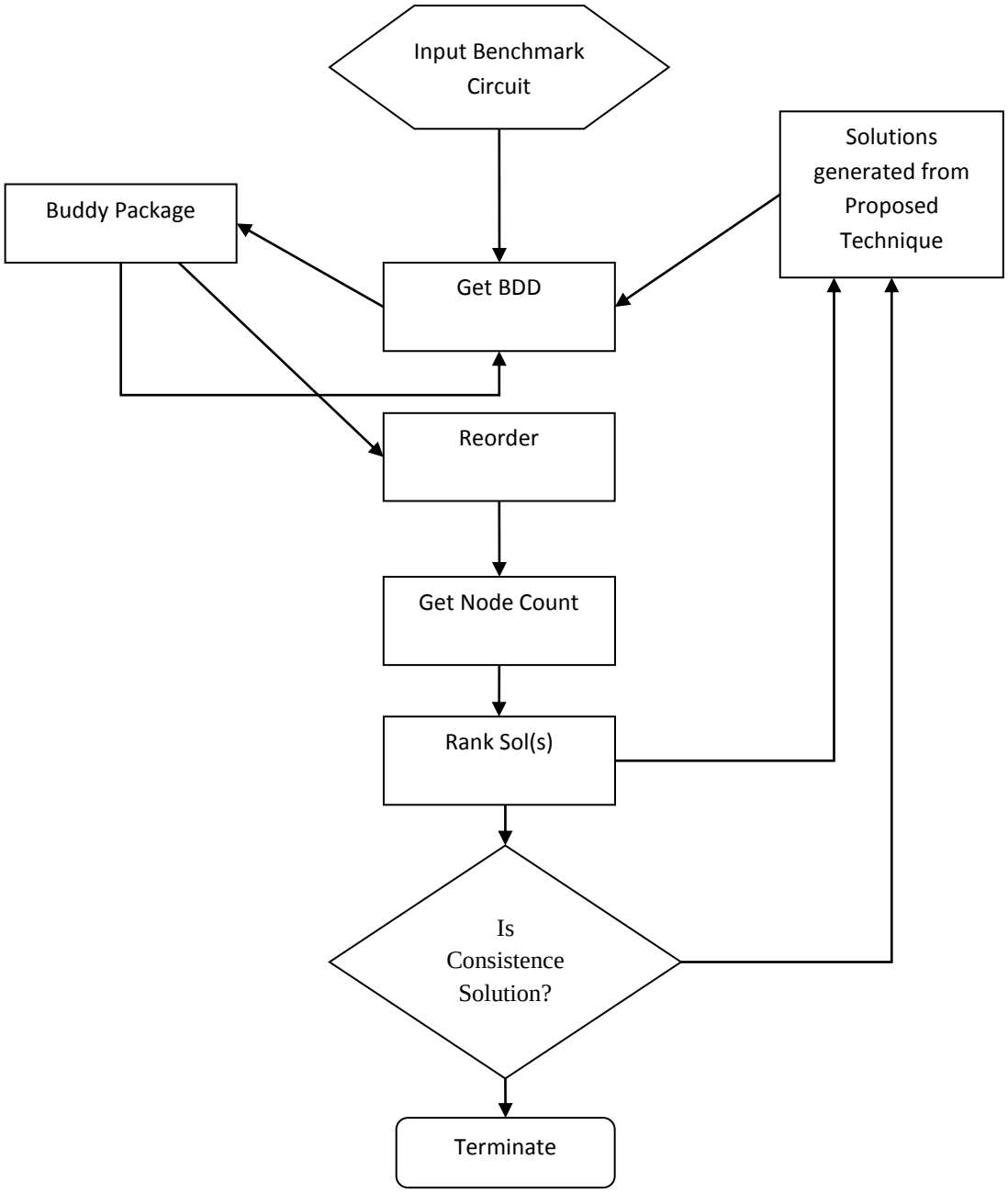


Figure 3.1 Flow Chart of whole Process

3.1 Basic Steps of Genetic Algorithm

Given a clearly defined problem to be solved and a bit string representation for candidate solutions, a simple GA works as follows:

1. Formulate initial population: generate ‘n’ chromosome
2. Randomly initialize population.
3. Find fitness function.
4. Selection: Select two chromosomes from population for reproduction

5. Apply crossover operator on parents. .
6. Mutation: change the property of new chromosome.
7. Repeat from step 3 to step 6 until ‘n’ new chromosomes generated.
8. Replace previous population with new population.
9. Go to step 2
10. Repeat above steps until criteria match.

Genetic algorithm starts with creating a initial population. Population is a set of possible solutions of the problem. Solutions are also called chromosomes. All feasible solutions of a given problem are represented in this complete search space of population. User has to define a fitness function for each problem. Fitness function is used to calculate the quality of each solution. In the population, fitness value of every chromosome is calculated. Only those chromosomes are selected as parents of next generation which are fit enough. Chromosomes with low fitness value are weeded out from solution. New chromosomes are generated by applying crossover operator on selected parents. New chromosomes have properties which are matched with their parent chromosomes. To increase the property of chromosome, mutation is performed on each chromosome. This process is continued until a new population set is build. Single iteration is performed when new population is completely generated. This iteration is repeated until consistence solutions get. During writing a program, there are a number of details to fill in it, such as the size of the population, the probabilities of crossover and mutation. The success of the algorithm often depends greatly on these details.

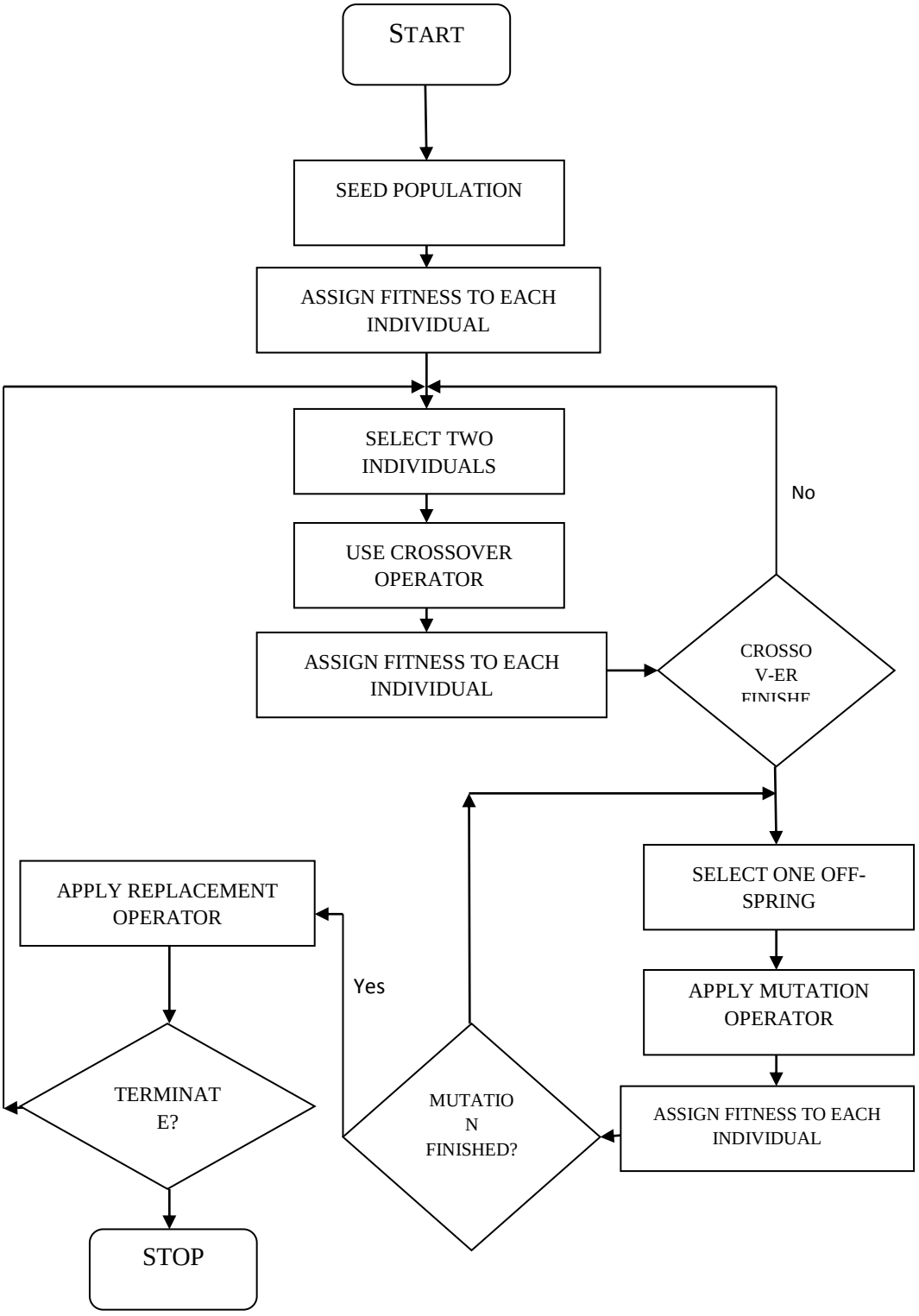


Figure 3.2 Flow Chart for Genetic Algorithm

3.2 Branch and Bound

The Branch and Bound algorithm which is used in the hybrid genetic algorithm for a minimization problem, consists of three main components. These components are:

1. A bounding function: A lower bound for the best solution value obtainable in the subspace.
2. A strategy for selecting the live solution.
3. A branching rule: To be applied if a subspace after investigation cannot be discarded.

A BB algorithm is the exact method which searches the complete set of solutions for finding a solution of a given problem. This approach is achieved by an iteration process which has basically three main components as discussed above. Branching is a splitting process .Through splitting procedure nodes are generated from the previous level by is breakdown the solution space of nodes into two or more sub spaces [11].

The next step of the BB technique is bounding. In this step, we have to compute only the lower bounds of the solution set by calculating the bounds for each of the solutions, within the given solution set. The lower bounds are calculated by setting the objective function of the proposed problem based on the fitness. The key idea of the Branch and Bound algorithm is implementing pruning by maintaining a global variable ‘w’ that records the minimum lower bound seen among all solutions present in the space which is already examined. The variable ‘w’ is shared among all the nodes of the tree. Pruning means if the lower bound for some tree nodes say X is greater than the lower bound for some other node Y in the same level, then X may be safely discarded from the search. Any node whose lower bound is greater than ‘w’ can be discarded. Otherwise, the bounding function for the subspace is calculated and compared with the current best solution.

3.3 Basic Working Principle of Hybrid Genetic Algorithm

The major steps involved in this algorithm are the generation of a population of solutions, finding the objective function and fitness function, application of branch and bound algorithm and genetic operators. The whole process consists of the following steps:

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

The hybrid genetic algorithm follows the same steps as genetic algorithm until branch and bound technique is applied. After initialising the population, fitness value of each chromosome is calculated. Then chromosomes are sorted according to their fitness value. Then branch and bound technique is applied on the set of chromosomes. Pruning is done on the search space. Fruitless chromosomes are discarded from the population. Then crossover operation is performed on the selected parents. New chromosomes are mutated to add new features in the chromosomes. This process is continued until new population of chromosomes is generated. When criteria is fulfilled the whole process stops running.

As discussed earlier, the crossover operator plays a significant role in the efficiency and performance of the algorithm. So, Crossover operator should choose wisely for each problem. The results are different for same problem with different crossover operator. In the proposed work, three different crossover operators are used in single program for efficient results. These crossover operators are Order crossover, Cycle crossover and Partially Mapped crossover. In simple genetic algorithm after selecting the parents by calculating the fitness value of initial population, three different crossover operators are applied on the same parents. This process is independent to each other. Then mutation is performed on all three off springs generated from parents. After that the fitness value of these off springs are calculated. The offspring with best fitness chose as the parent of next generation. Similarly, in the Hybrid Genetic algorithm after applying Branch and Bound on parents, crossover operators are applied on them. After mutation and calculating the fitness value, best chromosomes are chosen for next generation

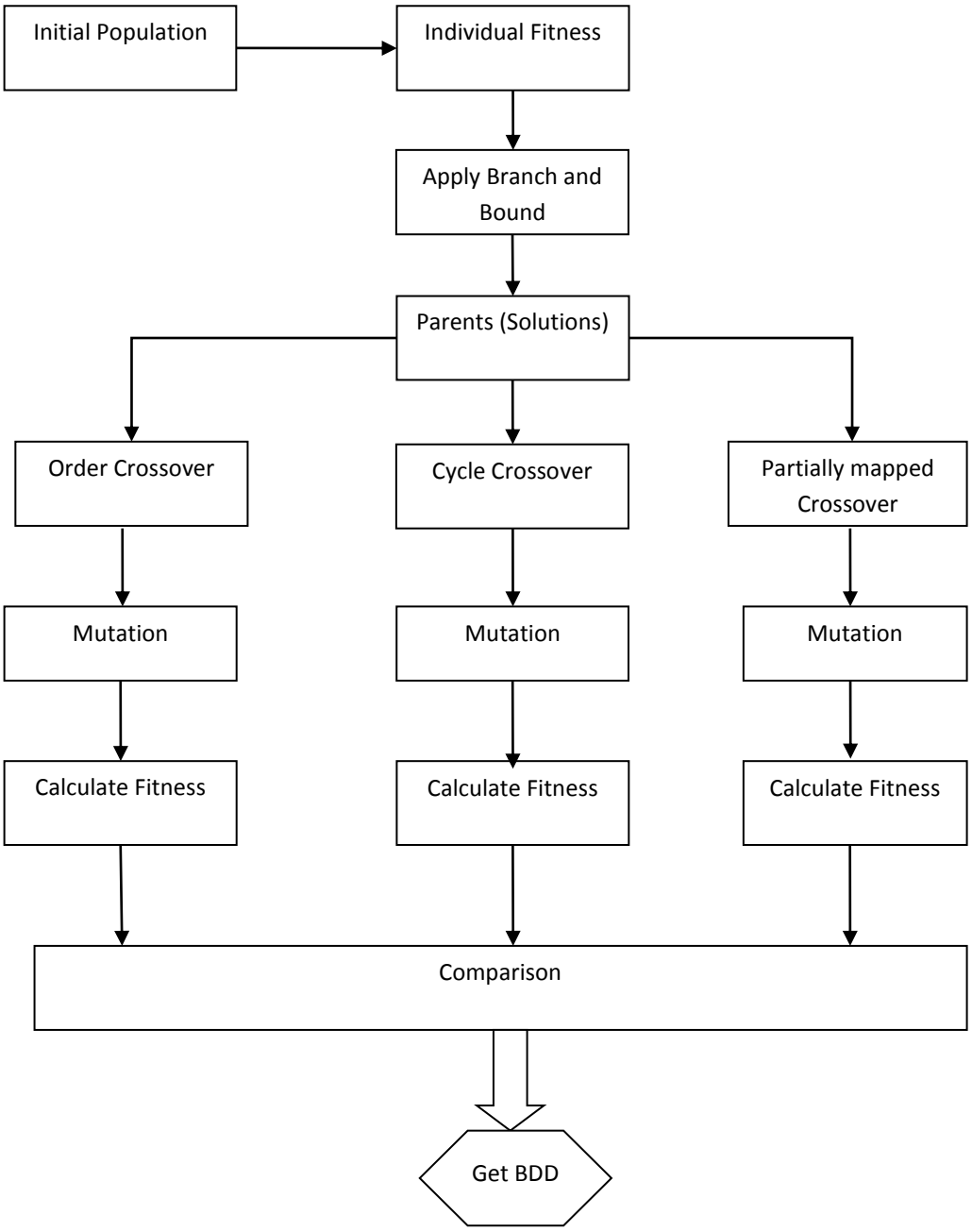


Figure 3.3 Proposed Technique with Hybrid Genetic Algorithm

3.4 Crossover Operator

The performance of the genetic and the hybridized genetic algorithm depends, to a great extent, on the performance of the crossover operator used. As mentioned earlier, crossover is the primary method of optimization in these algorithms, and works by exchanging the genes from two different parents to generate an offspring. There are three types of Crossover Operators used in the technique: Order Crossover, Cycle Crossover and Partially Mapped Crossover.

3.4.1 Order Crossover

This operator takes genes of first parent which present before the locus point without any change and copy them in the array of new chromosome [28]. Then, to resolve conflicts, it compresses the second parent by eliminating the cells or genes conveyed by the first parent and shifting the rest of the cells to the left without changing their order. After that, it copies this compressed second part into the remaining part of the offspring array.

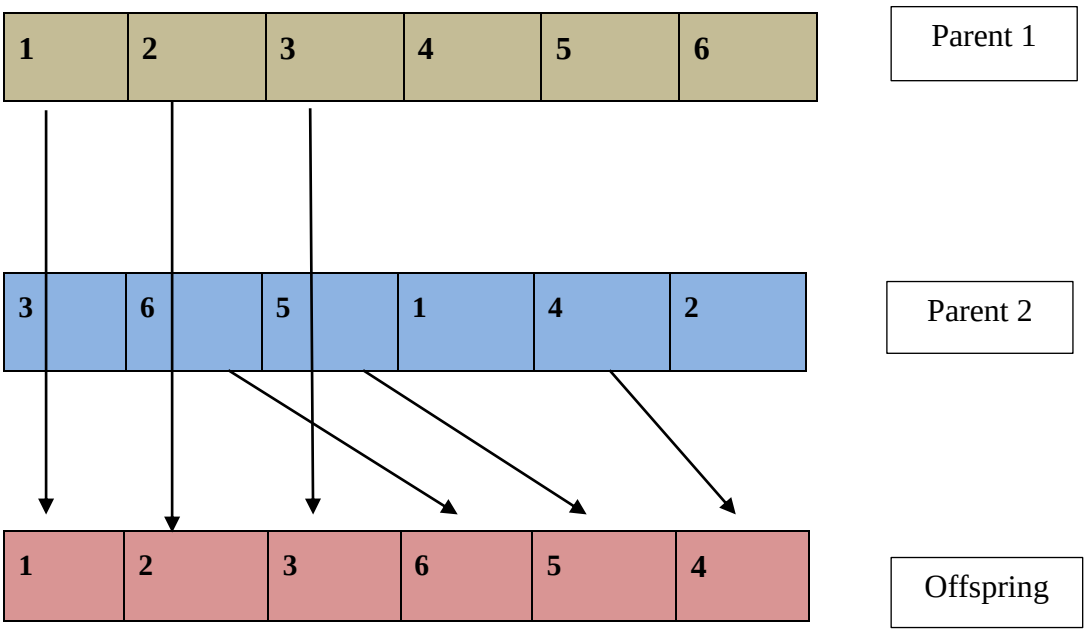


Figure 3.4 Order Crossover Operator

In figure 3.4, 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. 1, 2, 3. 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. 7, 8, 5.

3.4.2 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.

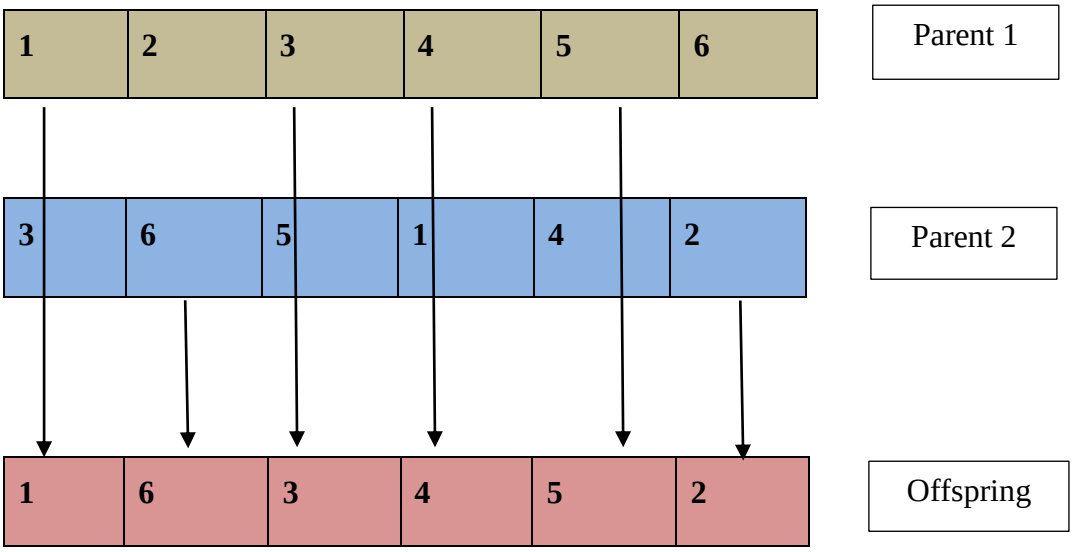


Figure 3.5 Cycle Crossover Operator

In figure 3.5, 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. 1 is copied to same position in offspring. Therefore, the first element of second parent i.e. 3 cannot be copied to offspring. So 3 is copied from the third position of first parent to the same position of the offspring. Similarly, 4 and 5 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. 2 occupies the last position of the offspring. Therefore, the last element of first parent i.e. 6 cannot be copied to offspring. So 6 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.

3.4.3 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.

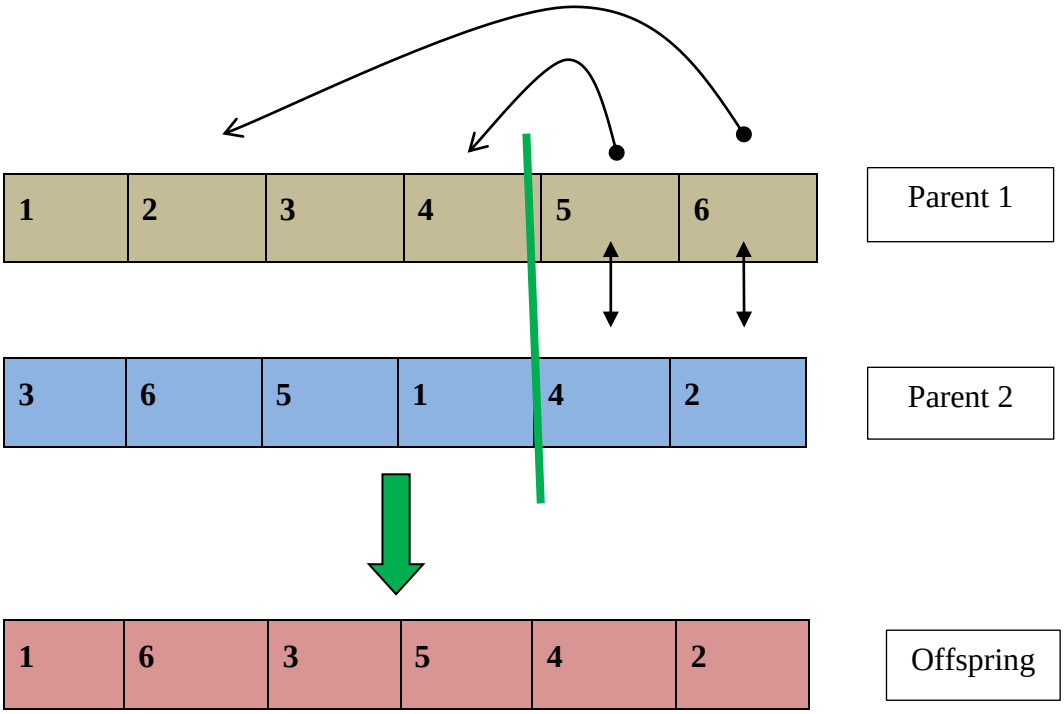


Figure 3.6 PMX Crossover Operator

In figure 3.6, 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. 5 is partially mapped with 4 and 6 is partially mapped with 2. Therefore, in the first parent 5 is exchanged with 4 and 6 is exchanged with 2 to form a new individual i.e. 1, 6, 3, 5, 4, 2.

3.5 Mutation

To increase the performance or fitness value mutation operation is used to introduce variety in offspring [28]. In other words, mutation is a process in which genetic information of the offspring is disturbed randomly. This is done to prevent all

chromosomes in a search space to fall into a local-optimum of solved problem. The type of encoding we used for representing a chromosome (i.e, the type of genes) determines how to perform mutation method. For example, if we are using binary encoding for representing chromosome then mutation of the selected bits can be perform by simply complementing them. If we are using integers to represent genes in this algorithm, then mutation operation can be performed in a similar way. It means genes are complimented and replaced with their corresponding integer compliments as shown in figure 3.7. To avoid the violation of constraints other genes can be modified. For example, if m is the total number of variables and n is any integer then

Complement of integer ‘n’ = m – n + 1

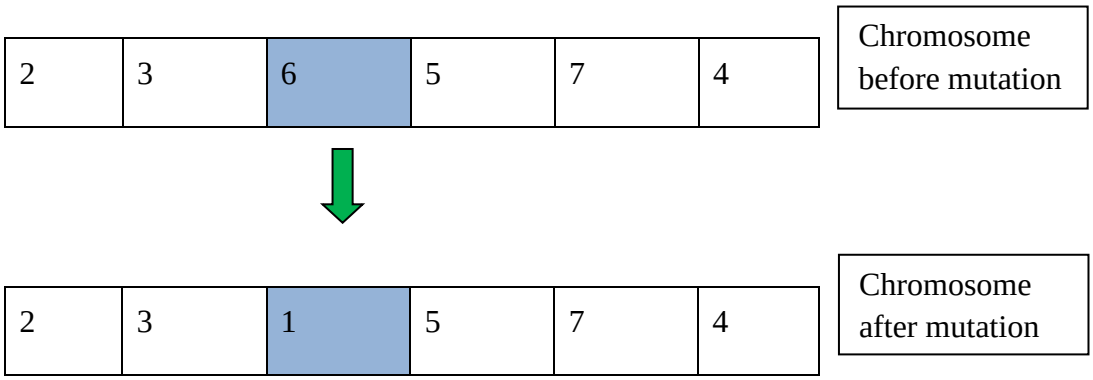


Figure 3.7 Mutation Operation

Chapter 4

Analysis and Methodology

- 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 14.04.
- The available approaches for BDD ordering and node minimization are compared along with the proposed techniques for BDD minimization on the set of Boolean functions.
- The proposed approaches give minimal nodes for every function when compared with all other approaches.
- Since a node in BDD is also consider as a multiplexer. The switching activity of node plays major role in power dissipation. Low power is a significant parameter in VLSI Circuits. So the as the number of nodes decrease power dissipation will also decrease. So optimization of BDD also helps in achieving low power VLSI circuits.
- The simulation results for various LGSynth93 benchmark circuits and Multi-input Adder circuits have been displayed in Tables 5.1, 5.2, 5.3, 5.4, and 5.5.
- The proposed algorithms and approaches provide the optimum variable order with minimum number of nodes, significant low computation time and low power dissipation in minimum iterations and number of runs for every function.

Chapter 5

Simulation Results

5.1 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 14.04. Simulation results *i.e.* node count and computation time for LGSynth93 Benchmark Circuits and Multi-input Adder Circuits have been presented in Tables 5.1, 5.2, 5.3, 5.4, 5.5 and 5.6.

Table 5.1 Comparison of Node Count for various Dynamic algorithms with Genetic Algorithm with Proposed technique for Benchmark Circuits.

LGSynth93 Benchmark Circuits	#i	#o	Initial node count	WIN2 node count	WIN2ite node count	WIN3 node count	SIFT node count	Proposed Method node count (Genetic Algorithm)	Up to % reduction w.r.t initial node count
5xp1	7	10	88	87	82	82	82	69	21.59
b12	15	9	91	86	86	65	65	59	35.16
Bw	5	28	118	113	112	106	106	96	18.64
Clip	9	5	254	178	108	105	105	89	64.96
con1	7	2	18	18	18	18	18	15	16.66
misex1	8	7	47	43	42	41	41	39	20.51
Z5xp1	7	10	69	68	68	68	68	60	13.04
sqrt8	8	4	42	40	39	37	37	35	16.66
Inc	7	9	73	75	69	68	68	58	20.54

Table 5.2 Comparison of Node Count of Multi-input adders for various Dynamic algorithms with Proposed approach of Genetic algorithm.

Benchmark circuits	#i	#o	Initial node count	WIN2 node count	WIN2ite node count	WIN3 node count	SIFT node count	Proposed Method node count (Genetic Algorithm)	Up to % reduction w.r.t initial node count
adder-1	3	2	8	8	8	8	8	8	0
adder-2	5	3	17	17	17	17	17	17	0
adder-4	9	5	63	65	63	46	46	35	44.44
adder-8	17	9	1027	1001	1001	987	283	265	74.19

Table 5.3 Comparison of Node Count of Benchmark Circuits for various Dynamic algorithms with Proposed approach of Hybrid Genetic Algorithm.

LGSynth93 Benchmark Circuits	#i	#o	Initial node count	WIN2 node count	WIN2ite node count	WIN3 node count	SIFT node count	Proposed Method node count (Hybrid Genetic Algorithm)	Up to % reduction w.r.t initial node count
5xp1	7	10	88	87	82	82	82	63	28.41
b12	15	9	91	86	86	65	65	59	35.16
Bw	5	28	118	113	112	106	106	94	20.33
Clip	9	5	254	178	108	105	105	89	64.96
con1	7	2	18	18	18	18	18	15	16.66
misex1	8	7	47	43	42	41	41	36	23.40
Z5xp1	7	10	69	68	68	68	68	61	11.59
sqrt8	8	4	42	40	39	37	37	33	21.42
Inc	7	9	73	75	69	68	68	56	23.28

Table 5.4 Comparison of Node Count of Multi-input Adders for various Dynamic algorithms with Proposed approach of Hybrid Genetic Algorithm

Benchmark circuits	#i	#o	Initial node count	WIN2 node count	WIN2ite node count	WIN3 node count	SIFT node count	Proposed Method node count (Hybrid Genetic Algorithm)	Up to % reduction w.r.t initial node count
adder-1	3	2	8	8	8	8	8	8	0
adder-2	5	3	17	17	17	17	17	16	5.8
adder-4	9	5	63	65	63	46	46	32	49.20
adder-8	17	9	1027	1001	1001	987	283	184	82.08

Table 5.5 Comparison of Computation Time of both Proposed techniques for Benchmark Circuits.

LGSynth93 Benchmark Circuits	#i	#o	CPU time(sec) Genetic Algorithm	CPU Time(sec) Hybrid Genetic Algorithm
5xp1	7	10	0.3	0.32
b12	15	9	12.01	13.1
Bw	5	28	0.04	0.054
clip	9	5	4.3	4.4
con1	7	2	2.57	2.59
misex1	8	7	2.87	3.6
Z5xp1	7	10	3.4	3.9
sqrt8	8	4	10.1	11.2
inc	7	9	2.9	3.4

Table 5.6 Comparison of Computation Time of both Proposed techniques for Multi-input Adders.

Benchmark circuits	#i	#o	CPU time(sec) Genetic Algorithm	CPU Time(sec) (Hybrid Genetic Algorithm)
adder-1	3	2	0.017	0.02
adder-2	5	3	0.03	0.035
adder-4	9	5	0.25	0.31
adder-8	17	9	14.3	15.7

5.2 Graphical analysis of Results

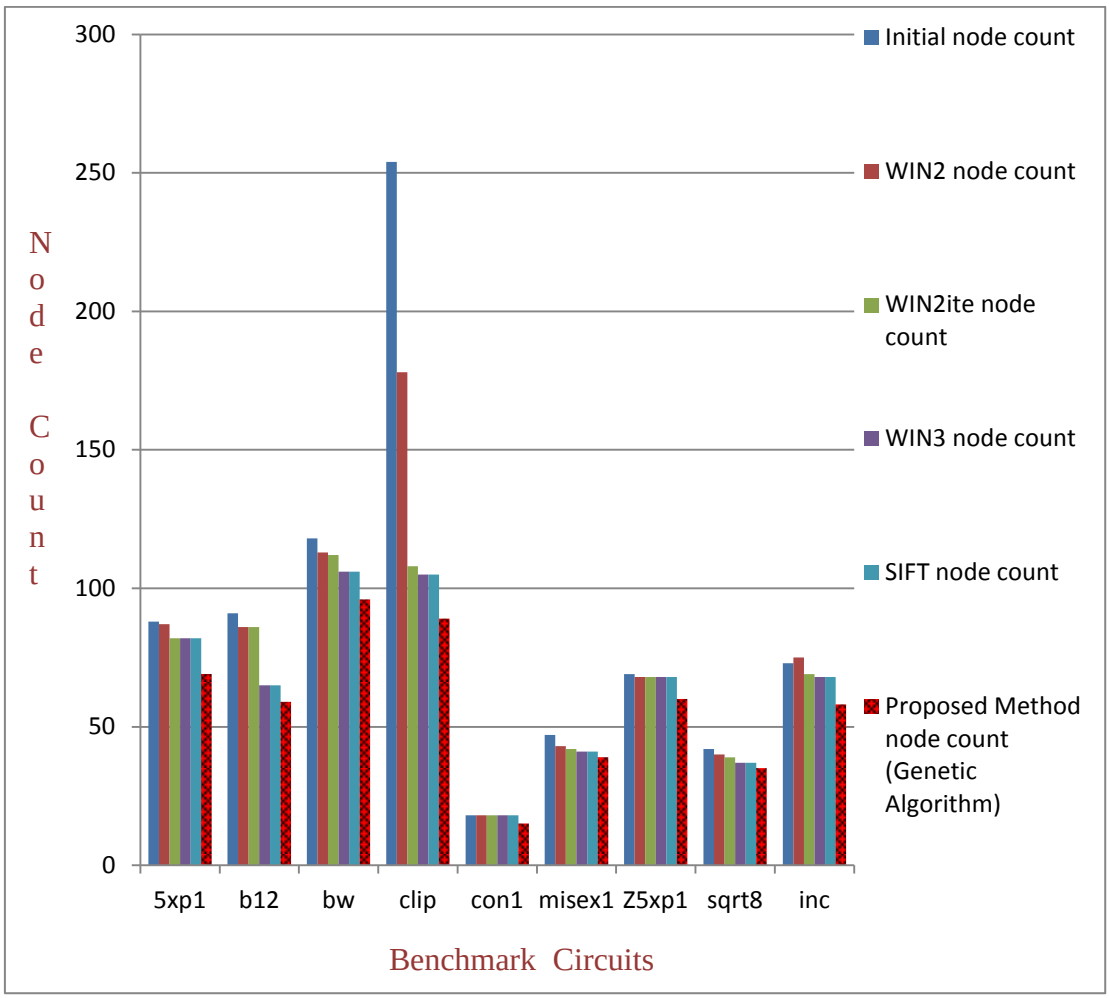


Figure 5.1 Comparison of Node Count of Benchmark Circuits for various Dynamic algorithms with Proposed approach of Genetic Algorithm

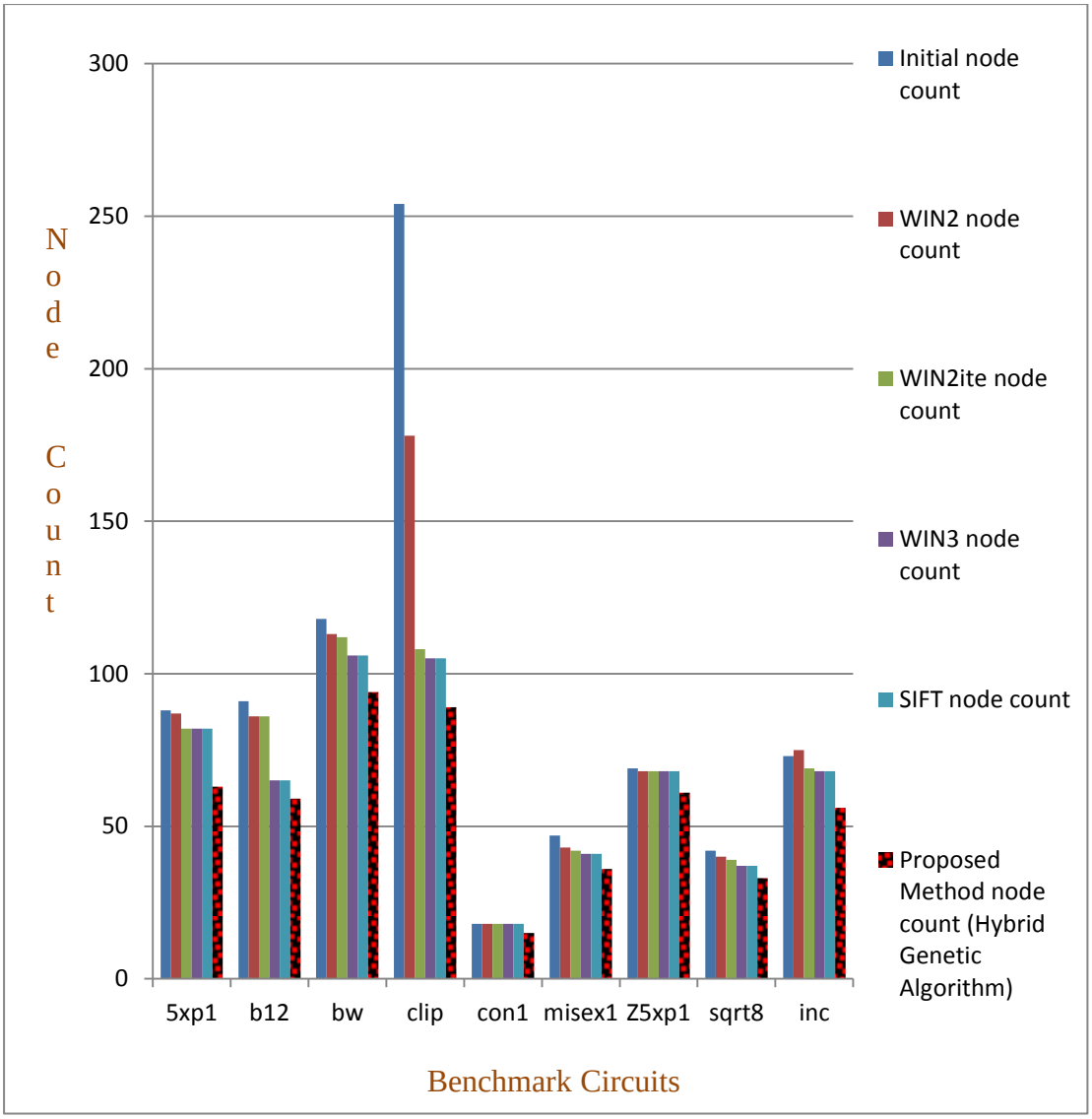


Figure 5.2 Comparison of Node Count of Benchmark Circuits for various Dynamic algorithms with Proposed approach of Hybrid Genetic Algorithm

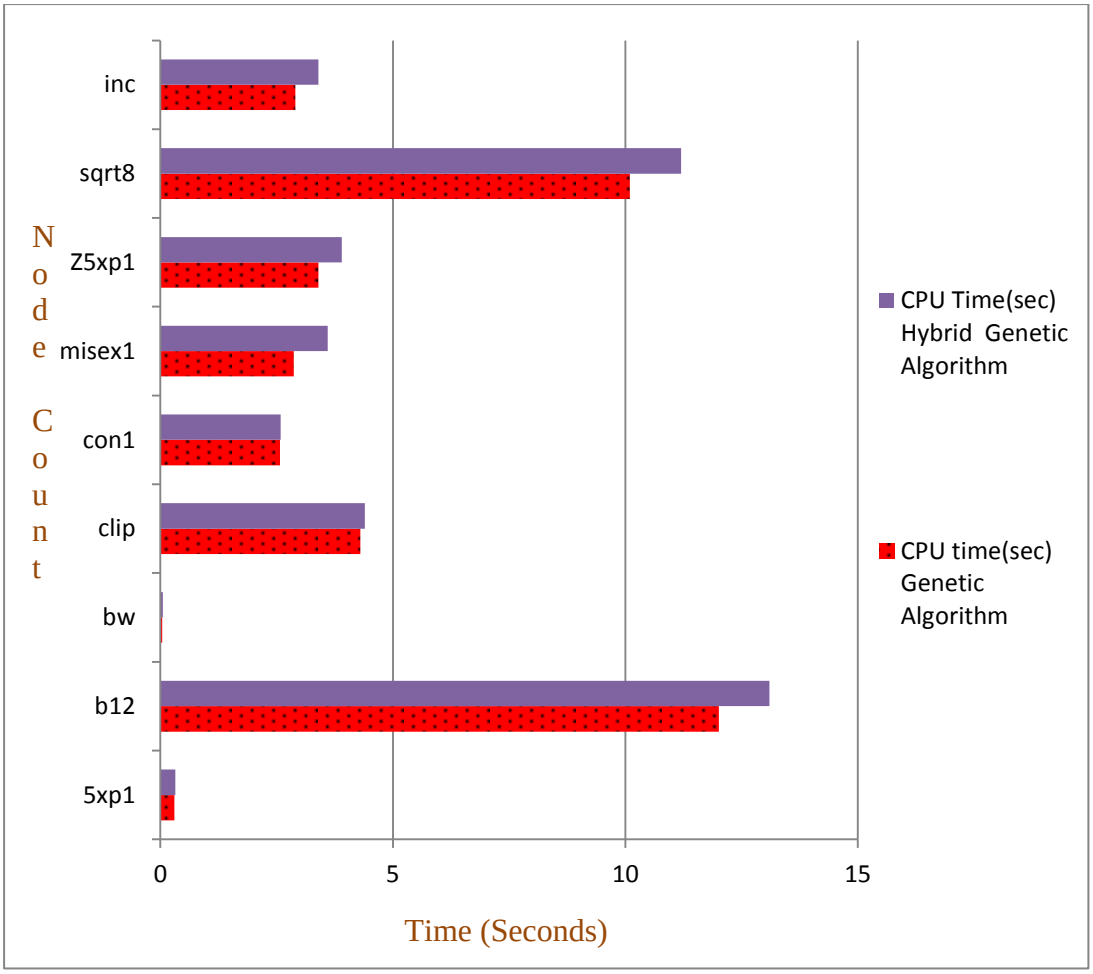


Figure 5.3 Comparison of Computation Time of both Proposed techniques for Benchmark Circuits.

Chapter 6
Conclusion and Future Work

6.1 Conclusions

1. 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.
2. It is observed that in comparison with other existing algorithms, the proposed approach with Hybrid Genetic Algorithm gives minimum node count. From Table 5.3, for Z5xp1, reduction in node count is about 11.5%, for clip is about 64.96%, for 5xp1 is about 28.41%, and for con1 is about 16.66%. The node count in multi-input adders has also reduced. For 8-adder reduction in node is about 82.09%. The reduction in node count is also present in other circuits as we can see in Table 5.3 and 5.4.
3. It is also observed that in comparison with other existing Dynamic algorithms, the proposed approach with Genetic Algorithm also gives reduction in node count. From Table 5.2, for 1-adder and 2-adder, with crossover reduction in node count is about 0%, for 4-adder is about 44.44%, and for 8-adder is about 74.19%. From Table 5.1, for Z5xp1, reduction in node count is about 13.04%, for clip is about 64.96%, for 5xp1 is about 21.59%, and for con1 is about 16.66%. The node count in multi-input adders has also reduced. For 8-adder reduction in node is about 82.09%. Therefore, the proposed approach with Genetic algorithm is also suitable for MIMO VLSI circuits.
4. From tables 5.5 and 5.6, it is observed that the computation time for both approaches is not much as compared to conventional methods. Computation time increases with increase in number of inputs and nodes. Genetic approach is taking less computation time than Hybrid Genetic approach.
5. 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.

6.2 Future Work

1. Various other existing heuristic algorithms can be applied to BDD mapped circuits. Evolution algorithms can be mix together for better efficiency. Other various dynamic algorithms can also be combined with heuristic algorithm.
2. The main problem with evolution algorithms is the use of large memory for storage of population. An approach can be developed in which algorithm use its resources again and again then this problem can be reduce.
3. Power can be optimized by using power optimization techniques along with Heuristic algorithms.

List of Publication

1. "**Effective Use of Various Crossover Operators in Hybrid Genetic Algorithm for Optimization of BDD Mapped Circuits**" in International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 5, Issue-6.

References

[1] Sheldon B. Akers “Binary Decision Diagrams”, *IEEE Transactions on Computers*, vol c-27, no.6, 1978.

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

[3] Shin-ichi MINATO, Nagisa ISHIURA and Shuzo YAJIMA, “Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation”, *IEEE Design Automation Conference, 27th ACM/IEEE,* pp. 52-57, 1990.

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

[5] 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.

[6] Nagisa Ishiura, Hiroshi Sawada, Shuzo Yajima, “Minimization of Binary Decision Diagrams Based on Exchanges of Variables”, *IEEE International Conference on Computer-Aided Design*, pp. 472 – 475, 1991.

[7] 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.

[8] 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.

[9] Alpana Agarwal and Manu Bansal, “Ordering and Reduction of BDDs for multi-input adders Using Evolutionary Algorithm”, *International Conference on Advanced Electronic Systems (ICAES)* , 2013.

[10] 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.

[11] 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*, pp. 167-173, 2010.

[12] Rolf Drechsler, Mikael Kerttu, Per Lindgren, Mitchell Thornton, “Low-power optimization techniques for BDD mapped circuits using temporal correlation”, *Can. J. Elect. Computer Eng.,* vol. 27, no. 4, 2002.

[13] Orna Grumberg, Shlomi Livne, “Learning to Order BDD Variables in Verification”, *Journal of Artificial Intelligence Research,* pp. 83-116, 2003.

[14] Shun-Shii Lin,Chun-Jen Wei, “A new approach for minimization of binary decision diagrams” , *Can. J. Elect. Computer Eng.*, vol. 30, no. 4, 2005.

[15] Wolfgang Lenders, Christel Baier, “Genetic Algorithms for the Variable Ordering Problem of Binary Decision Diagrams” *Springer Berlin Heidelberg*, pp. 1-20, 2005.

[16] 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.

[17] 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.

[18] 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*, vol. 2, no. 3, pp. 217-224, 2011.

[19] Rasoul Faraji, Hamid Reza Naji, “An efficient crossover architecture for hardware parallel implementation of genetic algorithm”, *ELSEVIER Neurocomputing*, vol. 128, *pp. 316–327*, 2014.

[20] Christoph Scholl, Bernd Becker, Andreas Brogle, “The multiple variable order problem for binary decision diagrams: theory and practical application” *Proceedings of the 2001 Asia and South Pacific Design Automation Conference*, pp. 85-90, 2001.

[21] Wang Mingquan,Yu Haibin, "BDD Minimization Based on Genetic Tabu Hybrid Strategy", *IEEE International Conference*, vol. 2, pp. 948-952, 2005.

[22] R. Ebendt, F. Gorschwin, R. Drechsler, "Advanced BDD Minimization", *Springer*, pp. 145-195, 2005.

[23] 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, pp. 1-7,2006.

[24] Du Su-guo, Sun Yan, "A Novel Ordering Method of Binary Decision Diagram", *International Conference on Management Science & Engineering*, vol. 5, no. 4, pp. 146-153, 2007.

[25] Raheleh Sadat Hosseini,Farnaz Towhidi,Arash Habibi Lashkari, "Binary Decision Diagram (BDD)", *International Conference on Future Computer and Communication*, pp. 92-96, 2009.

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

[27] Misagh Takapoo, M.B Ghaznavi-Ghoushchi, "IDGBDD: The novel use of ID3 to improve Genetic algorithm in BDD reordering", *International Conference on Electrical, Electronic Computer Telecommunication and Information Technology*, pp. 19-21, 2010.

[28] Pinaki Mazumder, Elizabeth M. Rudnick, Genetic Algorithms for VLSI Design,Layout & Test Automation, *Low Price Edition, Pearson Education, Inc. DK*, 1999.

[29] 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*, vol. 7, pp. 113-118, 2011.

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

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

[32] Osnat Keren, "Reduction of the Average Path Length in Binary Decision Diagrams by Spectral Methods", *IEEE Transactions on Computers*, vol. 57, no. 4, 2008.

[33] 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*, vol. 15, pp. 183-190, 2012.

[34] Harsh Arora, Arindam Banerjee, Rahul Roopchand Jidge, "Heuristic Approach to Variable Ordering for Logic Synthesis Engine Design: Algorithmic Insight", *International Journal of Circuits and Architectural Design*, vol.1, pp. 47-51, 2014.

[35] 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, pp. 4-11, 2006.