Indian Journal of Science and Technology, *Vol 9(48), DOI: 10.17485/ijst/2016/v9i48/93703, December 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

A Novel and Efficient Variable Ordering and Minimization Algorithm based on Evolutionary Computation

Surbhi Jindal* **and Manu Bansal**

Department of Electronics and Communication Engineering, Thapar University, Patiala - 147004, Punjab, India; ersurbhijindal@gmail.com, mbansal@thapar.edu

Abstract

Objectives: Considering the rapidly increasing scales of on-chip integration, the objective of this research is to propose an improved and efficient genetic algorithm for area optimisation in limited computational time. **Methods/Statistical Analysis**: A novel and efficient Evolutionary Algorithm, with three crossover operators – order, cycle and partially mapped, has been employed to obtain an efficient variable ordering of Binary Decision Diagram (BDD) as it plays crucial role in total node count and hence, the total used area, average computation time and storage requirement. The efficiency of proposed algorithm has been tested on International Workshop on Logic Synthesis (IWLS), IWLS'93 combinational benchmark circuits. **Findings:** It has been found, for node count reduction, that the proposed Genetic approach using Order Crossover, gives an average reduction of 25.92% with a maximum value of 56.19%; using Cycle crossover, achieves an average reduction of 26.43% with a maximum value of 59.79%; whereas, using PMX crossover, leads to the best possible reduction with an average value of 35.26% with a maximum value of 84.54% as compared to average value of24.81%, 24.19% and 13.77%in case of already existing Window, Sifting and Random algorithms respectively. In terms of CPU time, 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. The proposed algorithm is able to yield higher area optimisation in a limited CPU time. Depending on the priority of the application based on area reduction and time dissipation, either of the algorithms and either of the crossovers can be employed. **Application/Improvements:** The algorithm is able to give efficiently optimised results for about 90% of the benchmark circuits; hence, these can be employed for Multi-Input- Multi-Output Systems (MIMO systems) in VLSI.

Keywords: Binary Decision Diagram (BDD), Cycle Crossover, Genetic Algorithm, Order Crossover, Optimisation, Partially Mapped Crossover

1. Introduction

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. In this paper, an evolutionary algorithm, i.e., Genetic Algorithm has been presented with three different crossover operators – order, cycle and partially mapped (PMX) for the minimization of BDDs and its effectiveness as compared to the existing Sifting Window and Random algorithms.

BDDs have been broadly used in Computer Aided Design for the optimum logic synthesis and also in formal verification and testing of Digital Circuits. A

*Author for correspondence

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 have been extensively used in logic synthesis, logic verification, optimisation, fault simulation and test-pattern generation for Digital systems[5,6]. 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 because of the fact that the BDDs can be directly transformed into circuits by substituting every node of the underlying graph with a multiplexer[7]. The reason for this transformation lies in the Shannon decomposition[8].

Many heuristics[9–12] and algorithms[13–23] have been worked upon to determine the close-to-optimal solution for the optimisation of BDDs by improving the variable ordering of the input variables- be it by using static variable ordering[10,11] which is dependent upon circuit topology, to the dynamic variable ordering using optimisation or evolutionary techniques[13–23]. 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[24], and to determine an optimal variable ordering is an NP-hard problem[25]. 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[16], dynamic variable ordering[16]and evolutionary algorithms[18,26]. 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[8]. To overcome the drawbacks, several approaches have been suggested. In this paper, the effectiveness of Evolutionary Approach has been presented using the three crossover operators – order, cycle and PMX in comparison with the Sifting, Window and Random algorithms.

In this paper, the proposed algorithm has been found out to be having better ability to give optimisation in terms of node counts and hence, area in comparatively improved CPU time.

The paper is organized as follows: Section 2 reviews the BDD representation for the VLSI digital circuits. Section 3 describes the Genetic Algorithm for achieving variable ordering for BDDs. Section 4 presents the description of Crossover operators employed in the research work. Problem Statement and the formulation of the issues encountered when dealing with optimising BDDs along with the proposed algorithmic approach have been discussed in Section 5. Section 6 presents the Experimental Results and discussion. Finally, Section 6 presents some concluding remarks and directions of the future work.

2. Binary Decision Diagram (BDD) Representation for Digital Circuits

The concept of Boolean function that defines a digital circuit can be represented as a Binary Decision Diagram (BDD) which is a directed acyclic graph[27] with a compact data structure[28] as per a particular order and satisfying a defined set of properties. It is usually based on the recursive Shannon Expansion[28] for the switching function f, on which the graph based representation of BDD relies, based on the decomposition of f[28]around each variable y_i as is described by:

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

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.

A BDD can be implemented either in canonical form, 'ordered in any order' form (OBDD) or in reduced ordered form (ROBDD). An OBDD gets converted into an ROBDD when all duplicate terminals and redundant nodes are eliminated, identical nodes are shared and all duplicate nodes, if any, are merged together[3]. A BDD is shown for a function f = ac + bc in Figure 1.

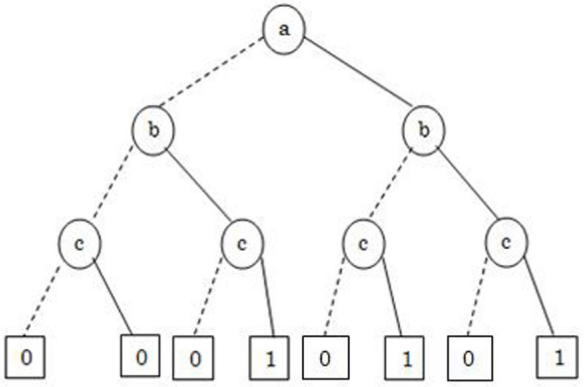

Figure 1. BDD for the function f = ac + bc.

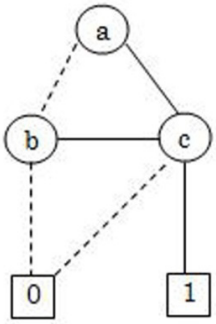

Figure 2. Reduced BDD for the function f = ac + bc.

Figure 2 shows its equivalent Reduced BDD (RBDD). The reduction in the overall size can be easily seen from here. The total node count has been reduced from 15 to 5.

The size of a BDD strongly depends on the order of input variables. Moreover, a function can have multiple ROBDDs[16] 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. Hence a number of heuristic methods and algorithms have been developed till now, to determine the optimal ordering for input variables.

3. Genetic Algorithm for Optimisation of BDDs

Four well-known paradigms for evolutionary algorithms are Genetic Algorithms (GAs)[29], Evolutionary Programming (EP), Evolution Strategies (ES), and Genetic Programming (GP)[30]. Natural evolution is the motivation behind the basis of such algorithms.

Based on the initial population size[31,32], 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 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[24] employed by any genetic algorithm[33,34] are:

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

The basic flow of Genetic Algorithm is as shown in Table 1.

Table 1. Basic genetic algorithm

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

4. Crossover Employed in Genetic Algorithm

Crossover is the processes of making the parents combine their genes to produce offspring. Various techniques for Crossovers have been proposed and accepted till now, out

of which, improved Order Crossover (using concept of selective positions or using cut points), Cycle Crossover, and Partially Matched Crossovers (PMX)[28] have been employed in this research.

- In case of 'Order Crossover', a component of one parent is mapped to a component of the other parent. From the replaced part on, the remaining is filled up by the left over genes, whereas the already present genes are excluded keeping the order intact.
- 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.
- 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.

5. Issue Formulation and Proposed Improved Genetic Algorithm

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. 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. 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[3,26]. However, the most tedious job in case of Ordered BDDs[16] 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 optimised results in a limited CPU time. Optimisation of variable ordering of BDDs using algorithms has long been an important optimisation methodology in Computer Aided Design (CAD/CAM) applications[35].

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[11]. The drawback of using Static techniques is that it requires a prior knowledge of the function's behaviour, like effect of each input variable on the function, which may not be always known[24]. 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[13]. 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[8,13,29] algorithm.

Although this algorithm decreases the number of ordering permutations from n1 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[29]

Many other methods have been using genetic algorithm[8,18,28,29,32] and different heuristics[10–12] to generate new orderings. The Genetic algorithm based approach used genetic operations like crossover and mutation[18] 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[24].

Considering the drawbacks, in this paper, an improved Genetic Algorithm approach for optimising 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.

The proposed algorithm for the Genetic approach is as shown in Figure 3.

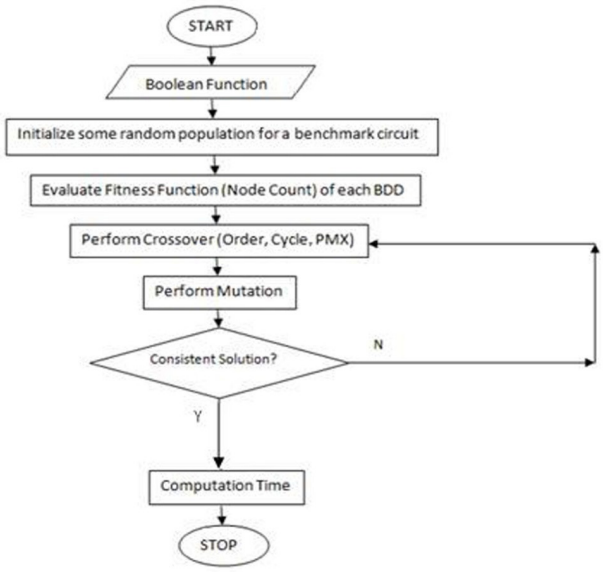

Figure 3. Flowchart for the proposed genetic approach.

A new random population of BDDs for every benchmark circuit was initialized. After that, individual fitness i.e. node count for each BDD was calculated. And then the application of Crossover Operators (Order/PMX/Cycle) i.e., combining features from two different parent individuals to generate a new placement was carried out. Then Mutation i.e., modifying characteristics of existing solution to generate a new one with number of mutations not more than [(number of elements)/2 + 1], was applied. Then, the obtained solution, if found to be consistent, was led to the calculation of computation time. In case of inconsistent solution, the iteration was repeated till consistent solution was obtained.

6. Experimental Results and Discussion

The genetic algorithm using three types of crossover operators has been implemented with C++ codes and simulated using BDD package Buddy 2.4[36] on Ubuntu 14.04. Simulation Results i.e., node count and corresponding computation times for IWLS'93 combinational benchmark circuits have been presented in the tables. 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

Table 2. 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 3. 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 4. 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

area optimisation results for different IWLS'93 benchmark circuits for Window algorithms have been shown in Table 2; for Sifting and Random algorithms, results are shown in Table 3; and shown in Table 4 are the results of the proposed Genetic approach. Table 5 lists the Computation Times taken to obtain the optimum results using the respective Genetic operators of the proposed approach.

Table 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

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.

From Table 2, 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 3 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 4, it can be found that the proposed 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%.

Hence, the proposed approach has been able to yield best optimisation results in terms of node count when compared to the Sifting, Window and Random algorithms, with PMX yielding the best reduction out of all the three crossover techniques.

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 but at the same time, is able to result in best optimisation in node count.

The graphs comparing the results in terms of node count have been shown in Figure 4, Figure 5, Figure 6 and Figure 7. The results of Computation times have been plotted in Figure 8.

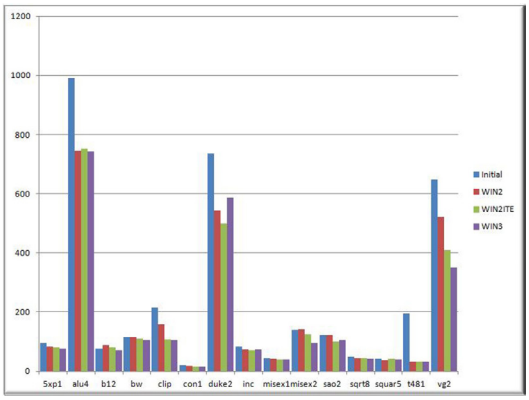

Figure 4. Node count comparison against initial values for window algorithms for IWLS'93 benchmarks.

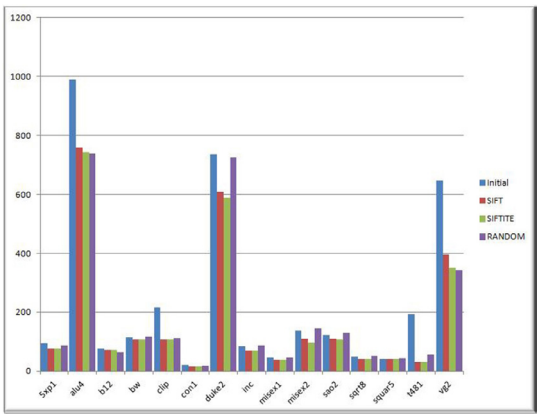

Figure 5. Node count comparison against initial values for sifting, random algorithms for IWLS'93 benchmarks.

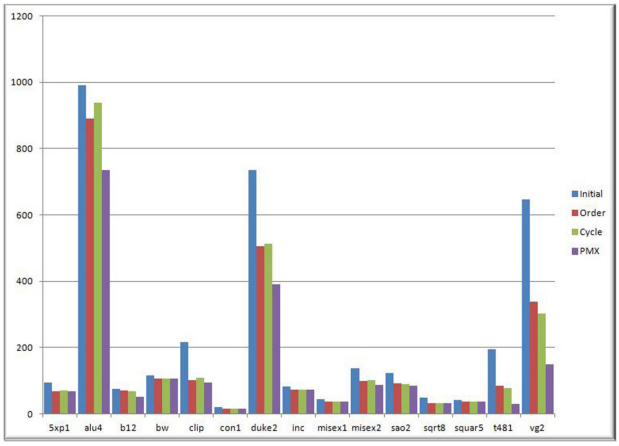

Figure 6. Node count comparison for proposed genetic (order, cycle, PMX) for IWLS'93 benchmarks.

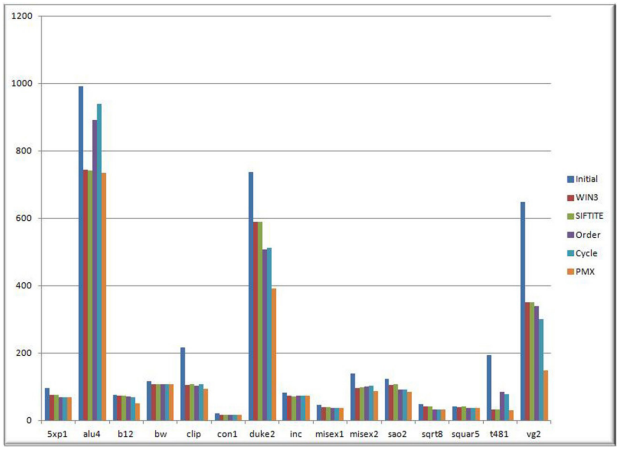

Figure 7. Node count comparison of best from sifting, window, proposed genetic for IWLS'93 benchmarks.

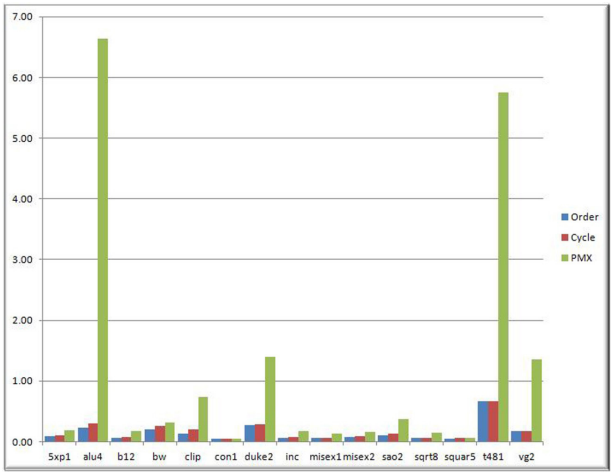

Figure 8. Computation time (seconds) comparison for proposed genetic (order, cycle, PMX) for IWLS'93 benchmarks.

7. Conclusions

The proposed algorithm for area optimisation i.e., reduction in code count, is based on the improved crossover techniques in Genetic Algorithm. It has been observed that the proposed methodology is a good improvement over the previous methods like Sifting, Window and Random algorithms in terms of minimization of the number of nodes, and hence, area when tested on many circuits. From the results it is evident that PMX is yielding the best node count but on the cost of increased computation times than the other two crossovers while the Order crossover and Cycle crossover are able to provide the node count results in a much lesser computation time. So, there has to be a trade-off between the choices. Depending on the priority of the application based on area or time, either of the crossovers can be employed. The results obtained here can be used as a good initial solution as inputs to the other optimisation algorithms in order to achieve multi-objective optimisation along with consideration to timing and switching activity concerns.

8. Acknowledgements

The research work was supported by Thapar University. The authors would like to thank Mr. Gagandeep Singh Dhingra (Thapar University, Punjab, India) for sharing their precious knowledge during this research, Ms. Rupinder Kaur (Chitkara University, Punjab, India), Mr. Raj Shah (Thapar University, Punjab, India) and Ms. Chandni Dodiya (VIT University, Chennai, India) for their valuable suggestions and motivation. Special thanks to the editors and reviewers for their valuable comments.

9. References

1. Akers SB. Binary decision diagrams. IEEE Trans Comput. 1978 Jun; 27(6):509–16.
2. Lee CY. Representation of switching circuits by binary-decision programs. The Bell System Technical Journal. 1959 Jul; 38(4):985–99.
3. Bryant RE. Graph-based algorithms for Boolean function manipulation. IEEE Trans Comput. 1986 Aug; 35(8): 677–91.
4. Cabodi G. Improving the efficiency of BDD-based operators by means of partitioning. IEEE Transactions on

Computer-Aided Design of Integrated Circuits and Systems. 1999 May; 18(5):545–56.
5. Rehan S, Bansal M. Performance comparison among different evolutionary algorithms in terms of node count reduction in BDDs. International Journal of VLSI and Embedded Systems. 2013 Jul; 4:1–6.
6. Minato SI. Binary decision diagrams and applications for VLSI CAD. Kluwer Academic Publishers; 1996.
7. Drechsler R, Kerttu M, Lindgren P, Thornton M. Low power optimisation techniques for BDD mapped circuits using temporal correlation. Can J Elec Comput Eng. 2002 Oct; 27(4):1–6.
8. Furdu I, Patrut B. Genetic algorithm for ordered decision diagrams optimisation. Proceedings of ICMI; Bacau. 2006 Sep. p. 1–8.
9. Aloul FA, Markov IL, Sakallah KA. MINCE: A static global variable-ordering heuristic for SAT search and BDD manipulation. J Univers Comput Sci. 2004; 10(12):1562–96.
10. Chung PY, Hajj IM, Patel JH. Efficient variable ordering heuristics for shared ROBDD. Proceedings of the IEEE Inte rnational Symposium on Circuits and Systems; Chicago, IL. 1993 May. p. 1690–3.
11. Fujita M, Fujisawa H, Matsunaga Y. Variable ordering algorithms for ordered binary decision diagrams and their evaluation. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 1993 Jan; 12(1):6–12.
12. Fujji H, Ootomo G, Hori C. Interleaving based variable ordering methods for ordered binary decision diagrams. IEEE/ACM International Conference on Computer-Aided Design. Santa Clara, CA, USA. 1993. p. 38–41.
13. Rudell R. Dynamic variable ordering for ordered binary decision diagrams. IEEE/ACM International Conference on Computer-Aided Design11; Santa Clara, CA, USA. p. 42–7.
14. Meinel C, Somenzi F, Theobald T. Linear sifting of decision diagrams. IEEE Proceedings of the 24th ACM/IEEE Design Automation Conference; CA, USA. 1997 Jun. 1993. p. 202–7.
15. Friedman SJ, Supowit KJ. Finding the optimal variable ordering for binary decision diagrams. Proceedings of the 24th ACM/IEEE Design Automation Conference; New York, USA. 1987. p. 348–56.
16. Grumberg O, Livne S, Markovitch S. Learning to order BDD variables in verification. J Artif Intell Res. 2003; 18:83–116.
17. Zhuang N, Benten MST, Cheung PYK. Improved variable ordering of BDDs with novel genetic algorithm. Proceedings of the IEEE International Symposium on Circ uits and Systems; Atlanta, GA. 1996. p. 414–7.
18. Chaudhury S, Dutta A. Algorithmic optimisation of BDDs and performance evaluation for multi-level logic circuits with area and power trade-offs. Circuits and Systems. 2011 Jul; 2(3):217–24.
19. Ishiura N, Sawada H, Yajima S. Minimization of binary decision diagrams based on exchanges of variables. IEEE International Conference on Computer-Aided Design; Santa Clara, CA, USA. 1991. p. 472–5.
20. Fey G, Drechsler R. Minimizing the number of paths in BDDs. Proceedings of the 15th Symposium on Integrated Circuits and Systems Design; 2002. p. 359–64.
21. Kerttu M, Lindgren P, Thornton M, Drechsler R. Switching activity estimation of finite state machines for low power synthesis. IEEE International Symposium on Circuits and Systems; Sweden. 2002. p. 65–8.
22. Sathya N, Muthukumaravel A. A review of the optimisation algorithms on travelling salesman problem. Indian J Sci Technol. 2015 Nov; 8(29):1–4.
23. Takapoo M, Ghaznavi-Ghoushchi MB. IDGBDD: The novel use of ID3 to improve genetic algorithm in BDD reordering. International Conference on Electrical Engineering/Electronics Computer Telecommunications and Information Technology; Chiang Mai. 2010. p. 117–21.
24. Siddiqui MDB, Bansal M. BDD ordering: A method to minimize BDD size by using improved initial order. International Journal of VLSI and Embedded Systems. 2013 Jun; 4(3):1–4.
25. Prasad PWC, Assi A, Harb A, Prasad VC. Binary decision diagrams: An improved variable ordering using graph representation of Boolean functions. International Journal of Computer, Electrical, Automation, Control and Information Engineering. 2008; 2(2):1–7.
26. Sharma G. Algorithmic reduction and optimisation of logic circuit in area and power tradeoffs' with the Help of BDD. International Journal of Engineering and Computer Science. 2014 May; 3(5):6132–9.
27. Roeva O, Fidanova S, Paprzycki M. Influence of the population size on the genetic algorithm performance in case of cultivation process modeling. Federated Conference on Computer Science and Information Systems; 2013 Sep 8-11. p. 371–6.
28. Lenders W, Baier C. Genetic algorithms for the variable ordering problem of binary decision diagrams. Proceedings of the 8th International Conference on Foundations of Genetic Algorithms; Springer. 2005. p. 1–20.
29. Kaur R, Bansal M. BDD ordering and minimization using various crossover operators in genetic algorithm. International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering. 2014 Mar; 2(3):1–4.
30. Gotshall S, Rylander B. Optimal population Size and the genetic algorithm. USA. 2002. p. 1–5.
31. Somenzi F. Binary decision diagrams. 2006; 27(6):509–16.

32. Esmin AAA, Matwin S. HPSOM: A hybrid particle swarm optimisation algorithm with genetic mutation. International Journal of Innovative Computing, Information and Control. 2013 May; 9(5):1919–34.
33. Kaghed HN, Al–Shamery SE, Al-Khuzaie FEK. Multiple sequence alignment based on developed genetic algorithm. Indian J Sci Technol. 2016 Jan; 9(2):1–7.
34. MarSadeghi, Gholami M. Genetic algorithm optimisation methodology for PWM inverters of intelligent universal transformer for the advanced distribution automation of future. Indian J Sci Technol. 2012 Feb; 5(2):1–6.
35. Simolowo OE, Okonkwo FC, Kehinde OO. CAD/CAM applications: Status and impact in Nigerian industrial sector. Indian J Sci Technol. 2010 Jun; 3(6):1–5.
36. Lind-Nielsen J. BuDDy: A binary decision diagram package; 2011. p. 1–36.