DELAY MINIMIZATION IN PASS TRANSISTOR LOGIC USE OF BINARY DECISION DIAGRAM

P.W.C. PRASAD
Faculty of Information Science and Technology,
Multimedia University,
Melaka, Malaysia.
m2160062@mmu.edu.my

M. RASEEN
College of Information Technology,
United Arab Emirates University, Al Ain, UAE
Mohamed.raseen@uaeu.ac.ae

S. SASIKUMARAN
Faculty of Science and Information Technology,
AL-Zaytoonah University,
Amman, Jordan
sasi_kumaran2002@yahoo.co.in

ABSTRACT: Pass Transistor Logic (PTL) offer great promise than Static CMOS for the development of chips that can operate in high speeds. Since the delay in a pass-transistor chain is proportional to its length, it is required to minimize the longest evaluation time. Minimization of longest evaluation time can improve the performance of the circuit, and have a strong influence on the quality of the final implementation. Our approach is based on the use of Binary Decision diagram (BDD) to minimize the longest evaluation time. We perform a novel static variable ordering techniques which is based on the complexity of each function. Benchmark results show an encouraging approach towards minimizing the longest evaluation time.

Key Words: Pass Transistor Logic, Evaluation time, Binary Decision Diagram

1. INTRODUCTION

Pass transistor logic offers a good area/power-delay trade-off alternative to static CMOS circuits in today's technologies. It may continue to do so even when leakage power becomes dominant in sub-100 nano-meter era due to smaller area implementations as compared to the corresponding static CMOS implementations [1]. One of the main problems with the pass transistor network is the presence of long paths: the delay of a chain of *n* pass transistors is proportional to n^2. The path length can be reduced by inserting buffers, but this increases area [2].

In general the evaluation time is proportional to the path length of BDD. Therefore minimization of path length can be achieved by minimizing the evaluation time of logic function [3]. The minimization of average path length (APL) proposed in [4], [5] reduces the average evaluation time of logic functions. The minimization of the average path length leads to circuits with a smaller depth on the paths from Root to Terminal nodes. By this, the circuit is optimized for speed on the one hand and on the other hand the number of very long paths is reduced [6]. The minimization of Longest path length (LPL) of BDD can reduce the longest evaluation time which is more important for PTL, since the minimization of the longest evaluation time will improve the performance of the circuit [3].

During the last two decades BDDs have achieved great popularity as a method for representing Boolean functions. The BDDs advantages as canonical representations were observed and recognized by Bryant [7], [8]. The success of this kind of representation has attracted many researchers in the area of synthesis and verification of VLSI and CAD systems [9], [10] and BDDs became very popular data structures, since they allow efficient representation of most of today's practical Boolean functions. The efficiency of BDDs is directly related to the size of their graph representations [11]. It appears that the node count is a good measurement for the complexity of the BDD. It is known that the complexity of BDDs is very sensitive to the ordering of the input variables [12].

Determining an optimal variable ordering is an NP-hard problem [13]. In the past many heuristic approaches have been proposed, that are based on Static variable ordering (SVO) algorithms [14], [15] that operate on a given circuit net-list and exploit topologies in order to find a good variable ordering, and Dynamic variable ordering (DVO) algorithms [16], [17], which assume that SVO has been successfully applied and operate on the resultant ROBDD to reduce its size through the exchange of variables [18]. A good ordering can lead to a smaller BDD and faster runtime, whereas a bad ordering can lead to an exponential growth in the size of BDD and hence can exceed the available memory. But most of these methods cannot guarantee an optimal result and experimental studies have shown that they are often up to a factor of two away from the best known solution [12]. In the current implementation, we perform a novel static variable ordering algorithm based on the manipulation of the Boolean function, which will reduce the longest evaluation time of BDD. The remaining of this paper is divided as follows: in the second section, necessary terminology and definitions are given. The third section explains the proposed method, and in the fourth section experimental results are given and interpreted. We conclude our paper with future developments of this research work.

2. PRELIMINARIES

Basic definitions for BDDs and PTLs are given in [3], [7], [11], [19]. In the following we review some of these definitions.

Definition 1: *A BDD* is a directed acyclic graph (DAG). The graph has two sink nodes labeled 0 and 1 representing the Boolean functions 0 and 1. Each non-sink node is labeled with a Boolean variables *v* and has two out-edges labeled 1 (or *then*) and 0 (or *else*). Each non-sink node represents the Boolean function corresponding to its 1 edge if *v*=1, or the Boolean function corresponding to its 0 edge if v=0.

Definition 2: An *OBDD* is a BDD in which each variable is encountered no more than once in any path and always in the same order along each path.

Definition 3: An *ROBDD* is a BDD with the following properties.

(i) There are no redundant nodes in which both of the two edges leaving the node point to the same next node present within the graph. If such a node exists it is removed and the incoming edges redirected to the following node.
(ii) If two nodes point to two identical sub-graphs (i.e. Isomorphic sub-graphs) then one sub−graph will be removed and the remaining one will be shared by the two nodes.

Definition 4: In a BDD, a sequence of edge and nodes leading from the root node to a terminal node is a *Path*. The number of non-terminal nodes on the path is the *Path Length*.

Definition 5: *The Longest Path Length (LPL)* of a BDD denoted by LPL (BDD), is the *Length of the Longest Path Length*.

3. PROPOSED METHOD

The proposed method is a static variable ordering technique [20], which uses the input Boolean expression to find the variable order which is used to minimize the longest evaluation time. It is based on the single level Boolean function; hence if a Boolean function is multilevel then it is converted to single level function prior to applying the proposed method. MVSIS (Multi Valued Logic Synthesis Tool) Version 1.0 is used for the conversion of functions from multilevel to single level. The proposed method selects the next variable in order based on the complexity of sub-functions derived by assigning logic 1 and logic 0 for that variable. The variable that produces sub-functions with minimum complexity is given priority over other variables. The complexity of the sub-functions is evaluated based on the number of variables (*NV*), number of product terms (*NPT*) and the number of variable occurrences (*NVO*). Using this method we can produce the BDD graph with shortest possible paths among each node including the terminal nodes, which will eventually, reduces the longest evaluation time. The complete steps of the proposed method are explained in the following algorithm:

Step1: The number of inputs (*N*) of the Boolean function is recorded. The input variables are named as $X_1, X_2, X_3 \ldots X_N$;
Step 2: Set the variable counter (*M*) to 1;
Step 3: Substitute logic 0 for variable X_M in the input function;
Step 4: Simplify the resulting function using McCluskey simplification method;
Step 5: Record the number of variables (*NV*), the number of product terms (*NPT*) and the number of variable occurrences (*NVO*);
Step 6: Substitute logic 1 for variable X_M in the input function and repeat steps 4 and 5;
Step 7: Record the total of *NV*, *NPT* and *NVO* obtained in steps 5 and 6;
Step 8: Repeat steps 3 to 7 for $M = 2$ to N;
Step 9: The variable that produced the least *NV* is selected as next variable in the order;
Step 10: If two or more variables have the same *NV* then selection is based on the least *NPT*;
Step 11: If two or more variables have the same *NPT* then selection is based on the least *NVO*;
Step 12: If still there are two or more variables that have equal *NV*, *NPT* and *NVO*, then the first of these variables is selected;
Step 13: The selected variable is then substituted for 0 and 1 in the input function and two sub-expressions are obtained;
Step 14: Steps 1 to 12 are repeated for the two new sub-expressions till we get the second variable of the order;
Step 15: The above steps are repeated for 2, 4, 8 ….. Sub-expressions until we find all the variables of the order;

For each the iterations described above, we note that the number of expressions increases but complexity of the expressions decreases. This decrease in complexity reduces the procedure to find the variables that come next in the variable order. The obtained variable order is used to build the BDD and computer the LPL using CUDD. The following example illustrates the proposed algorithm.

Example: Consider the Boolean function (1) with 4 variables x_1, x_2, x_3 and x_4.

$$F = x_1 \cdot x_2 \cdot \overline{x_4} + x_2 \cdot \overline{x_3} \cdot x_4 + x_1 \cdot x_3 \quad (1)$$

Substituting logic 0 and logic 1 for the four variables x_1, x_2, x_3 and x_4 we get the following sub-functions:

$$x_1 = 0 \Rightarrow x_2 \cdot \overline{x_3} \cdot x_4 \quad (2)$$

$$x_1 = 1 \Rightarrow x_2 \cdot \overline{x_4} + x_2 \cdot \overline{x_3} \cdot x_4 + x_3 \Rightarrow x_2 + x_3 \quad (3)$$

$$x_2 = 0 \Rightarrow x_1 \cdot x_3 \quad (4)$$

$$x_2 = 1 \Rightarrow x_1 \cdot \overline{x_4} + \overline{x_3} \cdot x_4 + x_1 \cdot x_3 \quad (5)$$

$$x_3 = 0 \Rightarrow x_1 \cdot x_2 \cdot \overline{x_4} + x_2 \cdot x_4 \Rightarrow x_1 \cdot x_2 + x_2 \cdot x_4 \quad (6)$$

$$x_3 = 1 \Rightarrow x_1 \cdot x_2 \cdot \overline{x_4} + x_1 \Rightarrow x_1 \quad (7)$$

$$x_4 = 0 \Rightarrow x_1 \cdot x_2 + x_1 \cdot x_3 \quad (8)$$

$$x_4 = 1 \Rightarrow x_2 \cdot \overline{x_3} + x_1 \cdot x_3 \quad (9)$$

The parameters NV, NPT and NVO for the above six sub-functions are shown the table 1.

Table 1.

Variable	NV			NPT			NVO		
	Logic 0	Logic 1	Total	Logic 0	Logic 1	Total	Logic 0	Logic 1	Total
X_1	3	2	5	1	2	3	3	2	5
X_2	2	3	5	1	3	4	2	6	8
X_3	3	1	4	2	1	3	4	1	5
X_4	3	3	6	2	2	4	4	4	8

The main title (on the first page) should begin 1-3/8 inches (3.49 cm) from the top edge of the page, centered, and in Times 14-point, boldface type. Capitalize the first letter of nouns, pronouns, verbs, adjectives, and adverbs; do not capitalize articles, coordinate conjunctions, or prepositions (unless the title begins with such a word). Leave two 12-point blank lines after the title.

From table 1, variable x_3 is selected as the first variable of the order since it has the least total of *NV* compared to other variables. Two new sub-expressions (5) and (6) are obtained from substituting $x_3 = 0$ and $x_3 = 1$ respectively. From expressions (5) and (6) we re-start the procedure to find the second variable of the order. Substituting logic 0 and logic 1 for the three variables x_1, x_2 and x_4 in the expressions (5) and (6) we obtain the following sub-functions:

$$x_3 = 0,\ x_1 = 0 \Rightarrow x_2 \cdot x_4 \qquad x_3 = 0,\ x_1 = 1 \Rightarrow x_2$$

$$x_3 = 1,\ x_1 = 0 \Rightarrow 0 \qquad x_3 = 1,\ x_1 = 1 \Rightarrow 1$$

$$x_3 = 0,\ x_2 = 0 \Rightarrow 0 \qquad x_3 = 0,\ x_2 = 1 \Rightarrow x_1 + x_4$$

$$x_3 = 1,\ x_2 = 0 \Rightarrow x_1 \qquad x_3 = 1,\ x_2 = 1 \Rightarrow x_1$$

$$x_3 = 0,\ x_4 = 0 \Rightarrow \cdot x_1 \cdot x_2 \qquad x_3 = 0,\ x_4 = 1 \Rightarrow x_2$$

$$x_3 = 1,\ x_4 = 0 \Rightarrow x_1 \qquad x_3 = 1,\ x_4 = 1 \Rightarrow x_1$$

Table 2 indicated the values of *NV*, *NPT* and *NVO* for the twelve sub functions. From table 2 the variable x_1 is selected as the next variable since it has the least number of total *NV*. It should be noted here that, so far in this example there was no a need to check the parameters *NPT* and *NVO*, since *NV* was always the least. The first two variables are selected, hence four sub- expressions are obtained from the substitution set (x_3, x_1) = (0,0), (0,1), (1,0) and (1,1). The 4 sub-functions can be used to find the third variable of the order. Performing the same procedure describe above one can find all the variables of the order. In our example, after doing all the steps we obtain the variable order x_3, x_1, x_2, and x_4. This variable order is used to build the BDD and find the LPL.

Table 2.

Variables		NV			NPT			NVO		
		Logic 0	Logic 1	Total	Logic 0	Logic 1	Total	Logic 0	Logic 1	Total
X_3=0	X_1	2	1	3	1	1	2	2	1	3
X_3=1		0	0		0	0		0	0	
X_3=0	X_2	0	2	4	0	2	4	0	2	4
X_3=1		1	1		1	1		1	1	
X_3=0	X_4	2	1	5	1	1	4	2	1	5
X_3=1		1	1		1	1		1	1	

4. EXPERIMENTAL RESULTS

To experimentally evaluate the algorithm presented in this paper, we assembled two set of results: the first set derives directly from the 3 variable ordering techniques (Symmetric Sifting, Swapping and Window Permutation) in CUDD package and the second set from the proposed method. Both set of results were observed using the Colorado University Decision Diagram (CUDD) package [21] on a Pentium IV machine with 512 MB RAMs. Table 3 summarizes the results for selected ISCAS benchmark circuits [22], [23], [24]. In Table 3 the first column shows a list of selected ISCAS benchmark circuits we have used to demonstrate the performance of the proposed method. Columns 2, 4 and 6 illustrate the results obtained for three CUDD variable reordering methods, namely the swapping, symmetric sifting and window permutation in terms of number of nodes and Columns 3, 5 and 7 shows the results for the same in terms on LPL. The results shown in column 8 and 9 are from the implementation of our proposed method in terms of number of nodes and LPL respectively. The Gain factors of the proposed method against the three CUDD methods are denoting in Column 10 to 12.

The obtained results indicate the efficiency of the proposed method compared to other CUDD methods in term of minimization of Longest Path Length and Number of nodes. In general, the obtained results in Table 3 indicate that the number of nodes decreases in more than 90% of the benchmarks compared to the Swapping and Window Permutation and almost 50% compared to the Symmetric sifts reordering method.

The circuit's i1, i6, X2, apex4, B9, pm1, clip, B12, mux and cm150a produces maximum gain by proposed algorithm compared to all the three CUDD methods. The benchmarks i7, X4, cc, squar5, misex2 and cm151a achieved better gain than two of the CUDD methods. The circuits alu2, decod, 5xp1, con1, cm85a, cm162a and cm163a produces equally good results as CUDD methods.

Table 3.

Benchmark Circuits	CUDD Methods						Proposed Method		LPL Gain Factor over CUDD methods		
	Swapping		Symmetric Sift		Window Permutation						
	Nodes	LPL	Nodes	LPL	Nodes	LPL	Nodes	LPL	Swapping	Symmetric Sift	Window Permutation
i1	72	57	71	56	77	57	70	55	1.04	1.02	1.04
i6	433	259	408	240	440	240	418	236	1.10	1.02	1.02
i7	535	281	510	271	666	303	535	273	1.03	0.99	1.11
X4	895	541	690	532	1199	554	720	532	1.02	1.00	1.04
X2	60	34	53	33	68	39	51	31	1.10	1.06	1.26
Apex4	1452	171	1410	171	1583	173	1384	168	1.02	1.02	1.03
Alu2	189	36	179	36	249	37	182	36	1.00	1.00	1.03
B9	245	150	196	150	275	153	190	147	1.02	1.02	1.04
cc	108	74	95	69	108	71	104	69	1.07	1.00	1.03
decod	96	80	96	80	96	80	96	80	1.00	1.00	1.00
Misex2	176	134	174	132	178	136	174	132	1.02	1.00	1.03
Pm1	78	63	77	63	81	63	74	61	1.03	1.03	1.03
5xp1	88	49	79	49	89	49	100	49	1.00	1.00	1.00
Con1	18	10	16	9	18	10	21	10	1.00	0.90	1.00
Misex1	68	34	66	36	68	34	66	34	1.00	1.06	1.00
F51m	66	36	61	34	68	36	79	36	1.00	0.94	1.00
Z4ml	36	22	33	22	45	22	32	21	1.05	1.05	1.05
Sao2	113	40	117	40	171	40	121	40	1.00	1.00	1.00
Cm85a	41	29	41	29	41	29	41	29	1.00	1.00	1.00
Squar5	54	34	52	32	57	34	53	32	1.06	1.00	1.06
clip	152	45	116	43	202	45	112	42	1.07	1.02	1.07
Cm151a	41	14	34	10	558	24	36	10	1.40	1.00	2.40
B12	82	54	77	53	97	54	83	51	1.06	1.04	1.06
mux	36	7	33	6	2327	18	33	5	1.40	1.20	3.60
cm150a	78	9	33	6	332	20	32	5	1.80	1.20	4.00
inc	223	106	220	106	237	108	247	107	0.99	0.99	1.01
cm162a	49	37	48	37	62	39	54	37	1.00	1.00	1.05
cm163a	46	31	42	31	55	34	42	31	1.00	1.00	1.10

5. CONCLUSIONS AND FUTURE WORK

A new algorithm for minimizing the Evaluation time in BDD has been developed. The algorithm has been implemented using ISCAS benchmark circuits and the results have been compared with the three CUDD reordering methods which show the applicability of Longest Path Length of BDD to Pass Transistor Logic circuits. From the results obtained so far, it is quite clear that the minimization of the Longest Evaluation tine of BDD is the most important factor in determining the quality of the final implementation. Our future work and developments will be concentrated on investigating the LPL minimization for larger Scale benchmark circuits.

REFERENCES

1. R. S. Shelar and S. S. Sapatnekar, Recursive Bipartitioning of BDD's for Performance Driven Synthesis of Pass Transistor Logic, *Proceedings of IEEE/ACM ICCAD*, Nov. 2001, pp. 449 - 452
2. V. Bertacco, S. Minato, P. Verplaetse, L. Benini, and G. De Micheli: "Decision Diagrams and Pass Transistor Logic Synthesis", *Stanford University CSL Technical Report*, No. CSL-TR-97-748, Dec. 1997.
3. S. Nagayama and T. Sasao, "On the minimization of longest path length for decision diagrams," International Workshop on Logic and Synthesis (IWLS-2004), June 2-4, Temecula, California, U.S.A., pp. 28-35.
4. R. Ebendt, S. Hoehne, W. Guenther, and R. Drechsler, "Minimization of the expected path length in BDDs based on local changes," *Asia and South Pacific Design Automation Conference (ASP-DAC'2004)*, pp. 866-871, Yokohama, Japan, Jan. 2004.
5. S. Nagayama and T. Sasao, "Code generation for embedded systems using heterogeneous MDDs," *the 12th workshop on Synthesis And System Integration of Mixed Information technologies*

(SASIMI 2003), pp. 258-264, Hiroshima, Japan, April 3-4, 2003.

6. F. Görschwin, S. Junhao and R. Drechsler, "BDD Circuit Optimization for Path Delay Fault-Testability", proceedings on EUROMICRO Symposium on Digital System Design, pp. 168-172, 2004
7. R. E. Bryant, Graph–Based Algorithm for Boolean Function Manipulation, *IEEE Trans. Computers, Vol. 35*, 1986, 677-691.
8. R. E. Bryant, On the complexity of VLSI implementations and graph representations of Boolean functions with application to integer multiplication, *IEEE Trans. Computers, Vol. 40*, 1991, 203–213.
9. Y. Lu, J. Jain, E. Clarke and M. Fujita, Efficient Variable Ordering using a BDD Based Sampling, *Proc. 37th Design Automation Conf.*, pp. 687-692, 2000.
10. K. Priyank, VLSI Logic Test, Validation and Verification, Properties & Applications of Binary Decision Diagrams, *Lecture Notes*, Department of Electrical and Computer Engineering University of Utah, Salt Lake City, UT 84112, 1997.
11. S. B. Akers, Binary Decision Diagram, *IEEE Trans. Computers, Vol. 27*, 1978, 509-516.
12. H. Fujii, G. Ootomo, and C. Hori, "*Interleaving based variable ordering methods for ordered binary decision diagrams*," in Int'l Conf. on CAD, pp. 38-41, 1993.
13. E. Justin, and F. Brglez, "Design of Experiments and evaluation of BDD ordering Heuristics", *Inter. Journal on Software tools for Technology Transfer*, Vol. 3 , No.2 , pp. 193-206, 2001.
14. M. Fujita, H. Fujisawa, and N. Kawato, "Evaluation and Improvements of Boolean Comparison Method Based on Binary Decision Diagrams," *in Proceedings of the InternationalConference on Computer Aided Design (ICCAD)*, pp. 2-5, 1988.
15. S. Malik, A. Wang, R. Brayton, and A. Sangiovanni-Vincentelli, "Logic Verification Using Binary Decision Diagrams in a Logic Synthesis Environment," *in Proceedingsof the International Conference on Computer Aided Desig*n *(ICCAD)*, pp. 6-9, 1988.
16. S. Panda and F. Somenzi, "Who Are the Variables in Your Neighborhood,"*in Proceedings of the International Conference on Computer Aided Design (ICCAD)*, pp. 74-77, 1995.
17. F. Somenzi, "Efficient Manipulation of Decision Diagrams," *in International Journal on Software Tools for Technology Transfer (STTT)*, 3(2), pp. 171-181, 2001.
18. R. Rudell, "Dynamic Variable Ordering for Ordered Binary Decision Diagrams,"*in Proceedings of the International Conference on Computer Aided Design (ICCAD)*, pp. 42-47, 1993.
19. R. Drechsler and D. Sieling, Binary Decision Diagrams in Theory and Practice, *Springer-Verlag Trans.*, 2001, pp. 112-136.
20. P.W. C. Prasad , A.Assi, M. Raseen and A. Harb "BDD Minimization Based on Minimal Cumulative Sub-functions Complexity ", accepted for publications on International Conference on Research Trends in Science and Technology, Lebanon, March 2005.
21. F. Somenzi, CUDD: CU Decision Diagram Package. *ftp://vlsi.colorado.edu/ pub/.*, 2003.
22. S. Yang. Logic synthesis and optimization benchmarks user guide version 3.0. *Technical report*, Microelectronic Centre of North Caroline, Research Triangle Park, NC, January 1991.
23. M. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering," *IEEE Design and Test*, vol. 16, no. 3, pp. 72-80, July-Sept. 1999.
24. F. Brglez and H. Fujiwara, " A neutral netlist of 10 combinational circuits and a target translator in Fortran, In *Inter. Symposium on Circuit and Systems, Special Sess. On ATPG and Fault Simulation*, pp. 663-698, 1985.