Product Term Minimization in ROBDDs with Application to Reconfigurable SET Array Synthesis

Yi-Hang Chen, Yang Chen, and Juinn-Dar Huang

Department of Electronics Engineering and Institute of Electronics

National Chiao Tung University, Hsinchu, Taiwan.

carlok@adar.ee.nctu.edu.tw, vermilionchen@gmail.com, jdhuang@mail.nctu.edu.tw

***Abstract*—The power dissipation has become a crucial issue for most electronic circuit and system designs nowadays when fabrication processes exploit even deeper submicron technology. In particular, leakage power is becoming a dominant source of power consumption. In recent years, the reconfigurable single-electron transistor (SET) array has been proposed as an emerging circuit design style for continuing Moore's Law due to its ultra-low power consumption. Several automated synthesis techniques for area minimization have been developed for the reconfigurable SET array in the past few years. Nevertheless, most of those existing methods focus on variable and product term reordering during SET mapping. In fact, minimizing the number of product terms can greatly reduce the area as well, which has not been well addressed before. In this paper, we propose a dynamic shifting based variable ordering algorithm that can minimize the number of disjoint sum-of-product terms extracted from the given ROBDD. Experimental results show that the proposed method can achieve an area reduction of up to 49% as compared to current state-of-the-art techniques.**

I. INTRODUCTION

As manufacturing processes are constantly moving toward very deep submicron (VDSM) technology, the device feature size of CMOS technology is continuously scaling down. However, this trend also makes leakage power play a dominant role in system power dissipation [1]. To tackle the problem of leakage power, various emerging low-power devices have been developed in recent years. Among them, the single-electron transistor (SET) is regarded as one of the most promising devices since it can operate with only few electrons at room temperature [2][3][4]. However, a SET device is suffering from low transconductance and degraded output resistance since only few electrons are involved in a switching operation. Consequently, a SET-based circuit must be designed in conjunction with non-CMOS logic architectures. A binary decision diagram (BDD) [5] based logic structure has been proposed as a feasible approach for realizing logic functions using SETs [6]. Since BDD is an alternate representation of the truth table, any Boolean function can be implemented through a proper mapping onto a hexagonal nanowire network controlled by Schottky wrap-gates [7][8][9].

A BDD-based hexagonal nanowire network can be assembled using a set of node devices. Each node device has one entry branch and two exit branches; messenger electrons arrive via the entry branch and then leave through either the left exit branch or the right one depending on the control variable of the wrap-gate. An exit branch is a segment of SET-controlled nanowire, and has four operating modes in terms of its conductivity: *open*, *short*, *active-high*, and *active-low*. In a hexagonal network, each row is controlled by the same logic variable. Hence, with the help of node devices, the BDD representation of a given logic function can be easily mapped onto a hexagonal network according to its inherent structure. The first real BDD-based hexagonal nanowire network has already been demonstrated in [8]. Nevertheless, it is a custom implementation without reconfiguration capability, which has limited applications. In addition, it suffers from the low yield issue due to the high defect rate of nanowire segments.

A reconfigurable SET array architecture using wrap-gate tunable tunnel barriers has been proposed to deal with these variability and reliability issues [10]. The architecture also presents two fabrication constraints, *granularity constraint* and *symmetric fabric constraint*. Given a Boolean function, not only the mapping orders of variables and product terms can significantly affect the final implementation on a SET array, but the number of disjoint product terms extracted from the ROBDD also plays an important role. However, most of the existing methods focus on how to reorder variables and product terms during SET mapping, while only few of them discuss the effect by the number of product terms. The first automated synthesis method targeting the reconfigurable SET array for area minimization was proposed in [11]. It merely deals with the ordering of product terms. However, experimental results show that the ordering of variables actually has a bigger impact on the optimization outcome than the ordering of product terms. An enhanced version of [11] was later presented in [12], and it thus takes the ordering of variables into account as well.

However, neither of the above two algorithms is aware of the two mandatory fabrication constraints before proceeding into the actual mapping stage. Therefore, in most cases, the actual mapping solution is far away from what is expected in earlier optimization stages. In addition, another synthesis approach tries to map a given function in a ladder shape to minimize the number of required hexagons [13]. The approach focuses mainly on the number of hexagons, whereas the reduction in width is not that significant. Consequently, in our opinion, minimizing the width of the mapped SET array is more advantageous than minimizing the number of hexagons because it is obvious that the width can better estimate the area than the hexagon count. Another recent work proposed an area minimization synthesis flow for reconfigurable SET arrays, which takes two fabrication constraints into account in early stages [17]; it can achieve significant improvement as compared with other works.

However, none of the aforementioned works tries to reduce the number of product terms for better mapping results. There are lots of previous works dealing with ROBDD size minimization, but the target is usually to minimize the number of nodes rather than the number of extracted product terms [18][19][20]. Consequently, an algorithm that can minimize the number of product terms is demanded for the reconfigurable SET array synthesis flow. Recently, a synthesis flow has been proposed, featuring both product term minimization and architecture relaxation, for width minimization in reconfigurable SET array synthesis [16]. Basically, it utilizes the threshold network to reduce the number of product terms. Nevertheless, a reconfigurable SET array is constructed as BDD-based structure. In our opinion, minimizing the number of product terms over the ROBDD representation of target Boolean function can potentially increase the chances of hexagon and path sharing in the following logic mapping phase.

In this paper, we propose a dynamic shifting based variable ordering algorithm for ROBDDs that can minimize the number of product terms such that the SET array width can be further reduced. Moreover, we combine the proposed approach with the synthesis flow that we developed previously to form a complete synthesis framework. The key idea of the proposed algorithm is to minimize the number of product terms by finding a good ROBDD variable

ordering. Since various variable orderings lead to different ROBDD structures, our proposed algorithm can determine a proper variable ordering for product term minimization. The proposed method can dynamically identify the best unfixed variable that potentially achieves the minimum number of product terms with those fixed ones. Besides, the proposed algorithm has been integrated into an existing reconfigurable SET array synthesis flow to demonstrate how it can greatly reduce the SET array width after synthesis.

The rest of this paper is organized as follows. Section II describes our variable ordering method on ROBDDs to minimize the number of disjoint sum-of-product terms. Experimental results and analyses are then reported and discussed in Section III. Finally, concluding remarks are given in Section IV.

II. Proposed Algorithm

A. Key Attribute Computation

To explain our method clearly, we first define several necessary numeric attributes of ROBDD and provide a procedure that can compute them efficiently. The diagram size can be further reduced by adopting complement edges. As a result, each edge of an ROBDD can be a normal edge or complement edge. A complement edge is an edge indicates that the function of this edge is the complement of the function on its ending node. A path of an ROBDD from the root to the terminal node in which the total number of complement edges is even is denoted as a *one-path*; each one-path actually represents a product term of the output function. We further define the following attributes to record the intermediate values on nodes in an ROBDD. For nodes of ROBDD, $ep : V \rightarrow Z^+$ specifies the number of paths that have even number of complement edges from the root to a node v, denoted as $ep(v)$; $op : V \rightarrow Z^+$ specifies the number of paths that have odd number of complement edges from the root to a node v, denoted as $op(v)$. Note that ep(ternimal-1) specifies the number of one-paths in an ROBDD, which is also equal to the number of disjoint sum-of-product terms.

Those two attributes mentioned above can be calculated in topological order starting from the root toward terminal-1. For ease of discussion, the set of all complement edges is denoted as E_C, and the set of all non-complement edges is denoted as E_N. Then, those two attributes can be calculated as:

$$ep(v) = \sum_{(u,v)\in E_N} ep(u) + \sum_{(u,v)\in E_C} op(u), \quad (1)$$

$$op(v) = \sum_{(u,v)\in E_N} op(u) + \sum_{(u,v)\in E_C} ep(u), \quad (2)$$

$$ep(root) = op(root) = 0. \quad (3)$$

With the help of above equations, the proposed algorithm can determine the number of product terms without exhaustively enumerating them all. The set of disjoint sum-of-product terms can be extracted after the ordering of variables is fixed. We present an approach that can find all of the product terms in a single traversal. The traversal is started from the root node, and a *path table* is allocated for each node v to record all paths from the root to v. In order to keep track the status of each path, we associate a Boolean variable *cp* to each path in the table. The variable *cp* is set as TRUE when the number of complement edges is even, otherwise the value is set as FALSE. All nodes are visited in topological order, the path table of each node is constructed by the union of all the path tables belong to its predecessor nodes. When an edge is a complement one, the path table is propagated from tail node to head node by flipping the value of *cp* of each path; otherwise, the path table is passed as it is. Fig. 1 illustrates an example of disjoint sum-of-product terms extraction. The set of disjoint sum-of-product terms of the target function f includes every path in the path table of terminal-1 node whose *cp* value is TRUE.

B. First Variable Exploration

The first variable that would be mapped to a reconfigure SET array is very important due to the structural nature and fabrication constraints of SET arrays. Based on the structure similarity of SET array and ROBDD, the first variable selection also has a great impact on the size of ROBDD in terms of the number of nodes and product terms. The key idea of this phase is to explore the search space of variable ordering by trying every variable as the first one. The search spaces that utilizing different first variables are guaranteed independent to each other. Therefore, this approach can help the next phase have a better chance to find a more globally optimized variable ordering at the cost of longer runtime.

C. Next Variable Determination

In the first phase, n independent solution spaces, with different first variable, of an n-variable ROBDD are identified. In this phase, our method continues to explore each space to find a better solution within the space. The proposed method sequentially determines the position of the next variable one by one based on the previously determined variables. Because the number of one-paths in an ROBDD is equal to the number of disjoint product terms, the key idea of this phase is to estimate the number of possible one-paths for each node and use the information to determine the optimized variable ordering. For ease of discussion, we define a *partial path* is a path that starts at the root and end at a non-terminal node in an ROBDD. Moreover, an *even-path* is a partial path that has even number of complement edges; an *odd-path* is a partial path that has odd number of complement edges. Since the number of one-paths is equal to the number of disjoint product terms, the number of partial paths terminated at an input variable x can also reflect the priority of x in the final ordering. However, the partial paths are propagated from the root to the terminal node due to the structural nature of ROBDD. The node located in deeper levels potentially has more partial paths. Therefore, to normalize the overall impact of partial paths for a node n, the *effective level* of n, $el(n)$, is defined as:

$$el(n) = \sum_{(p,n)\in E}[el(p) + 1], \quad (4)$$

$$el(root) = 0, \quad (5)$$

where E denotes the set of all edges in an ROBDD. The effective level reflects the combined effect of both the depth of a node as well as the size of its fanin cone. Therefore, the effective level of a node is proportional to the number of partial paths that end at this node. In order to evaluate the normalized impact of node n, the *score* of n is defined as:

$$score(n) = \frac{[\,ep(n) + op(n)\,]}{el(n)}, \quad (6)$$

where $ep(n)$ represents the number of even-paths ending at n, and $op(n)$ denotes the number of odd-paths that ending at n. After calculating scores for all nodes, the *weight* of variable v is further defined as:

$$weight(v) = \sum_{n\in node(v)} score(n), \quad (7)$$

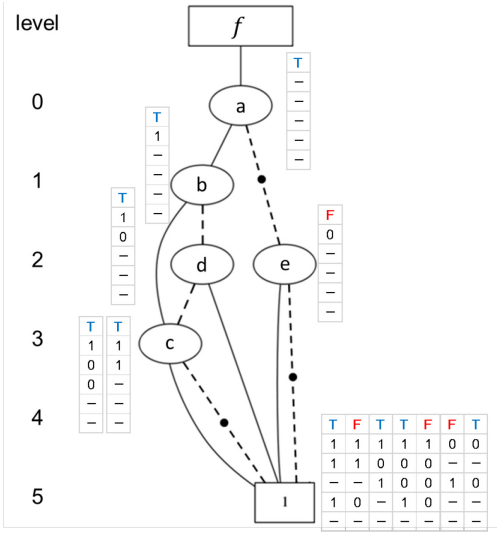

Figure 1. Identification of all disjoint product terms.

where *node*(*v*) specifies the set of nodes that controlled by variable *v*. With the help of *weight*(*v*), the proposed method can gradually complete the variable ordering by selecting the most important variable for variable swapping one at a time. A high value of *weight*(*v*) suggests variable *v* is very important for reducing the number of product terms. The reason for summing up scores for all nodes controlled by variable *v* is that whether a partial path is eventually a one-path or not is not conclusive yet. In each iteration, the variable *v* that has highest score among all unlocked variables is selected. The proposed approach then swaps the variable *v* while keeping the order of other variables unaltered. The number of product terms is then calculated for each possible position of *v*, and *v* is eventually locked in the position that has the minimum number of product terms. The relative positions of locked variables are preserved for the rest of iterations. This process is not ended until a complete variable ordering is finalized (i.e., all variables locked).

The above process is further demonstrated in Fig. 2. The first phase of the proposed method divides the entire solution space of *n*-variable ROBDD into *n* subspaces, each is starting with a distinct variable. In Fig. 2, the varible ordering starting with variable 2 is used to demonstrate the process of the second phase of the proposed method. The number beside each ordering specifies the number of product terms associated with the ordering. Finally, the variable ordering that has the smallest number of product terms among *n* subspaces is identified as the final outcome.

III. Experimental Results

The proposed algorithm has been implemented in C++/Linux environment and evaluated on a workstation with an Intel Xeon 2.4GHz CPU and 192GB RAM. To evaluate the proposed method we compare it against two prior synthesis techniques [17][20]. A set of experiments have been conducted for comparisons over a set of 21 test cases, which are selected from the MCNC benchmark suite [21] and IWLS benchmark suite [14].

The experimental results are presented in Table 1. The first column lists the names of test cases used for evaluation; the second and the third give the numbers of primary inputs and primary outputs, respectively. The original method only uses the heuristic function *CUDD_REORDER_SYMM_SIFT*, which tries to find a variable ordering that can minimize the node size. We have also re-implemented the Modified Rudell's sifting algorithm [20] as a trivial variable swapping method. Then, we combine the first phase of the proposed method with the Modified Rudell's sifting algorithm to demonstrate the effectiveness of the first phase. Furthermore, the results of the complete proposed flow (first phase + second phase) are shown in the last part in Table 1. The synthesis technique for reconfigurable SET array proposed in [17] is used to generate the final mapping results.

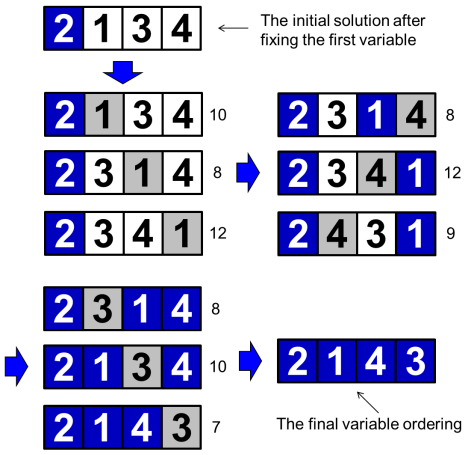

Figure 2. Determination of variable ordering.

The overall results show that our method (*Proposed Flow*) can achieve a reduction of the number of disjoint product terms by 55%, 44% and 13% on average as compared to the original CUDD method, the Modified Rudell's sifting algorithm, and the method combines our first phase and the Modified Rudell's sifting algorithm (*Enhanced Modified Rudell's*), respectively. After performing the actual SET mapping [17], the results demonstrate that our approach can reduce 49% and 38% of the SET array width as compared to [17] and [20], respectively. The results also show that the use of the number of one-paths as a metric is very effective for product term minimization in a reconfigurable SET synthesis flow. Regarding the runtime efficiency, our method takes hours to finish the largest test case. However, our method can also achieve 62% reduction in SET array width as compared with the original method for the same case. Therefore, it is obvious that the additional runtime is worthwhile and acceptable. Consequently, it is conclusive that the proposed algorithm can find a reasonably good variable ordering for the given ROBDD such that the number of disjoint product terms can be significantly reduced, which leads to a better area-minimized SET synthesis outcome than the prior arts.

IV. Conclusion

In this paper, we address the issue of area minimization synthesis for reconfigurable SET arrays. We first point out that the minimization of the number of disjoint product terms can generally reduce the mapped area of SET array. We then propose a dynamic variable shifting algorithm for product term minimization for an ROBDD. The proposed algorithm consists of two phases, the first variable exploration for larger solution space as well as the next variable determination that can find a fairly good variable ordering within each solution space. The experimental results demonstrate that the proposed method achieves 55% and 44% reduction of the number of product terms as compared to [17] and [20], respectively. Besides, our approach can also reduce the mapped width of SET array by 49% and 38% as compared to [17] and [20], respectively. Consequently, it is convincing that the proposed approach should be integrated into the reconfigurable SET array synthesis flow for better area minimization.

References

[1] International Technology Roadmap for Semiconductors, Semiconductor Industry Association, 2006.

[2] H. W. Ch. Postma, T. Teepen, Z. Yao, M. Grifoni and C. Dekker, "Carbon nanotube single-electron transistors at room temperature," *Science*, vol. 293, pp. 76–79, Jul. 2001.

[3] Y.-T. Tan, T. Kamiya, Z. A. K. Durrani, and H. Ahmed, "Room temperature nanocrystalline silicon single-electron transistors," *Journal of Applied Physics*, vol. 94, pp. 633–637, Jul. 2003.

[4] L. Zhuang, L. Guo, and S.-Y. Chou, "Silicon single-electron quantum-dot transistor switch operating at room temperature," *Applied Physics Letters*, vol. 72, pp. 1205–1207, Mar. 1998.

[5] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Computers*, vol. 35, pp. 677–691, Aug. 1986.

[6] N. Asahi, M. Akazawam, and Y. Amemiya, "Single-electron logic device based on the binary decision diagram," *IEEE Trans. Electron Devices*, vol. 44, pp. 1109–1116, Jul. 1997.

[7] H. Hasegawa and S. Kasai, "Hexagonal binary decision diagram quantum logic circuits using Schottky in-plane and wrap gate control of GaAs and InGaAs nanowires," *Physica E: Low-dimensional Systems and Nanostructures*, vol. 11, pp. 149–154, Oct. 2001.

[8] S. Kasai, M. Yomoto, and H. Hasegawa, "Fabrication of GaAs-based integrated 2-bit half and full adders by novel hexagonal BDD quantum circuit approach," *in Proc. International Symposium on Semiconductor Device Research*, 2001, pp. 622–625.

[9] S. Kasai and H. Hasegawa, "A single electron binary-decision-diagram quantum logic circuit based on Schottky wrap gate control of a GaAs nanowire hexagon," *Electron Device Letters*, vol. 23, pp. 446–448, Aug. 2002.

Table 1. Experimental Results and Comparisons.

Benchmark			Original Method in CUDD			Modified Rudell's [20]			Enhanced Modified Rudell's			Proposed Flow		
Name	PI	PO	PT	Width	Runtime (s)	PT	Width	Runtime (s)	PT	Width	Runtime (s)	PT	Width	Runtime (s)
c17	5	2	8	28	0.00	8	27	0.02	7	24	0.05	7	25	0.07
cu	14	11	23	97	0.02	22	96	0.25	22	96	1.85	22	95	4.58
cmb	16	4	26	51	0.02	26	51	0.27	26	51	0.54	26	51	1.71
Cm163a	16	5	31	111	0.03	27	100	0.31	27	100	2.39	27	100	4.20
i1	25	16	38	104	0.06	38	99	0.74	38	104	1.61	30	95	22.69
x2	10	7	40	130	0.02	36	130	0.12	29	101	0.70	28	98	1.88
cm162a	14	5	41	143	0.04	37	131	0.27	37	131	1.36	37	131	2.65
pcle	19	9	45	112	0.04	45	129	0.56	45	112	1.29	45	112	5.64
cm138a	6	8	48	134	0.01	48	138	0.04	48	134	0.07	48	134	0.34
unreg	36	16	48	195	0.11	48	195	2.21	48	195	4.55	48	195	41.58
pm1	16	13	49	131	0.04	48	137	0.32	40	112	3.72	37	102	7.76
cc	21	20	53	179	0.05	53	200	0.62	53	179	1.31	53	179	14.78
pcler8	27	17	67	220	0.13	67	220	1.67	61	202	4.95	61	198	36.39
cht	47	36	81	344	0.27	81	360	4.46	81	344	11.58	81	344	182.72
c8	28	18	85	260	0.16	85	259	1.83	79	254	5.71	79	254	28.47
cm85a	11	3	98	239	0.29	90	288	0.38	98	239	0.56	98	239	0.88
sct	19	15	153	464	0.39	107	362	0.73	104	355	3.88	103	354	23.65
lal	26	19	171	454	0.66	169	438	2.04	171	467	10.10	168	433	26.92
count	35	16	200	406	0.77	184	394	4.67	184	394	50.16	184	394	88.19
alu2	10	6	320	971	19.73	289	914	12.72	246	799	8.24	217	710	7.27
b9	41	21	376	1200	9.49	338	1054	11.31	211	651	179.86	177	566	506.15
usb_phy	112	116	438	1545	11.65	380	1435	90.74	348	1301	1494.43	335	1265	34062.40
example 2	85	66	447	1233	9.92	415	1133	57.99	377	1033	670.40	367	979	11205.47
apex7	49	37	739	2688	317.23	739	2710	407.15	470	1804	310.14	497	1967	1079.07
stepperm otordrive	28	29	766	1847	6.61	747	1810	11.18	665	1587	64.06	661	1539	138.50
x3	135	99	1051	3185	534.20	1035	3248	1088.75	796	2860	1364.73	726	2519	26530.31
sasc	132	129	2362	7615	351.68	1512	5134	397.24	883	3492	16175.59	798	2903	184085.26
i2c	145	140	2308	5989	14735.46	2286	5539	1067.15	1044	3612	20538.76	964	3228	144819.67
alu4	14	8	3580	9155	68513.92	1930	5833	6984.01	1313	4540	2982.96	1,078	4029	1358.87
x1	51	35	3588	10994	29960.41	3026	8951	16410.09	1403	4740	1574.57	802	2508	2746.87
Total	**-**	**-**	**17280**	**50224**	**114473.4**	**13916**	**41515**	**26559.8**	**8954**	**30013**	**45470.1**	**7804**	**25746**	**407034.9**
Norm. original	**-**	**-**	**1.00**	**1.00**	**1.00**	**0.81**	**0.83**	**0.23**	**0.52**	**0.60**	**0.40**	**0.45**	**0.51**	**3.56**
Norm. [20]	**-**	**-**	**1.24**	**1.21**	**4.31**	**1.00**	**1.00**	**1.00**	**0.64**	**0.72**	**1.10**	**0.56**	**0.62**	**15.3**

[10] S. Eachempati, V. Saripalli, N. Vijaykrishnan, and S. Datta, "Reconfigurable Bdd-based Quantum Circuits," *in Proc. International Symposium on Nanoscale Architectures*, 2008, pp. 61–67.

[11] Y.-C. Chen, S. Eachempati, C.-Y. Wang, and S. Datta, "Automated mapping for reconfigurable single-electron transistor arrays," *in Proc. Design Automation Conference*, 2011, pp. 878–883.

[12] Y.-C. Chen, S. Eachempati, C.-Y. Wang, S. Datta, Y. Xie, and N. Vijaykrishnan, "A synthesis algorithm for reconfigurable single-electron transistor arrays," *ACM Journal on Emerging Technologies in Computing System*, vol. 9, no. 1, article 5, Feb. 2013

[13] C.-E. Chiang, L.-F. Tang, C.-Y. Wang, C.-Y. Chen, S. Datta, and N. Vijaykrishnan, "On reconfigurable single-electron transistor arrays synthesis using reordering techniques," *in Proc. Design, Automation and Test in Europe*, 2013, pp. 147–152.

[14] http://iwls.org/iwls2005/benchmarks.html

[15] F. Somenzi, CUDD: CU decision diagram package - release 2.4.2, 2009. http://vlsi.colorado.edu/~fabio/CUDD/

[16] C.-W. Liu, C.-E. Chiang, C.-Y. Huang, C.-Y. Wang, Y.-C. Chen, S. Datta and V. Narayanan, "Width minimization in the Single-Electron Transistor array synthesis," *in Proc. Design, Automation and Test in Europe*, 2014, Article No. 122.

[17] Y.-H. Chen, J.-Y. Chen, and J.-D. Huang, "Area minimization synthesis for reconfigurable single-electron transistor arrays with fabrication constraints," *in Proc. Design, Automation and Test in Europe*, 2014, Article No. 123.

[18] G. Fey And R. Drechsler, "Minimizing the number of paths in BDDs: Theory and algorithm," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 25,, pp. 4–11, Jan. 2006.

[19] K. Brace, R. Rudell, and R. Bryant, "Efficient implementation of a BDD package," *in Proc. Design Automation Conference*, 1990, pp. 40–45.

[20] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," *in Proc. International Conference on Computer-Aided Design*, 1993, pp. 42–47.

[21] S. Yang, "Logic Synthesis and Optimization Benchmarks, Version 3.0," *Tech. Report*, *Microelectronics Center of North Carolina*, 1991.