An Enhanced Iterative Improvement Method for Evaluating the Maximum Number of Simultaneous Switching Gates for Combinational Circuits

Kai Zhang, Haruhiko Takase, Terumine Hayashi and Hidehiko Kita

Department of Electrical and Electronic Engineering
Faculty of Engineering, Mie University
Tel: +81-592-32-1211 ext.3901
e-mail: zhang@lotus.hayashi.elec.mie-u.ac.jp

Abstract— This paper presents an enhanced iterative improvement method with multiple pins (EIIMP) to evaluate the maximum number of simultaneous switching gates. Although the iterative improvement method is a simple algorithm, it is powerful to this purpose. Keeping this advantage, we enhance it by two points. The first one is to change values for multiple successive primary inputs at a time. The second one is to rearrange primary inputs on the basis of the closeness that represents the number of overlapping gates between fan-out regions. Our method is shown to be effective by experiments for ISCAS benchmark circuits.

I. Introduction

The concern of power dissipation and device reliability increases in proportion to the level of integration of LSI. The advent of VLSI has led to much recent work on the estimation of power dissipation and the enhancement of reliability during the design phase ([1]-[4]), so that designs can be modified before manufacturing.

In CMOS integrated circuits, both power consumption and long-term reliability are strongly related to the circuit switching activity. To guarantee the long-term reliability of VLSI chips, a worst-case reliability analysis is more desirable than the average-case [5]. Moreover, the maximum number of switching gates may also be used to evaluate the maximum power dissipation for enhancing the worst-case reliability of combinational circuits.

Many approaches have been proposed for evaluating the maximum number of switching gates, such as the approach based on max-satisfiability via disjoint cover enumeration [6], the partial exhaustive enumeration method [7], the branch-and-bound method [8], the method using genetic algorithm (GA) [9].

The approach based on max-satisfiability through disjoint cover enumeration [6] by Devadas et al. can make use of efficient disjoint cover enumeration and graph manipulation algorithms. However, it is only applied to CMOS circuits with below 1000 gates and 100 primary input pins.

Ueda and Kinoshita proposed three methods ([7]-[9]) of evaluating the maximum number of switching gates. The partial exhaustive method is powerful though the procedure is simple, but it tends to spend too much computational power on the local optimization. The branch-and-bound method needs much more CPU time than the other methods. The GA method may be effective if the appropriate parameters can be provided. However, it is difficult to decide the values of parameter properly.

First, we propose an iterative improvement method with multiple primary input pins (IIMP), hereafter, we call primary input pins "pins". In IIMP, an initial primary input vector pair is iteratively improved through changing values of orderly selected multiple pins so as to increase the number of switching gates in the circuit. To find a larger number of switching gates, the procedure is repeated with the different initial vector pairs generated randomly. Second, this method is enhanced by rearranging the order of pins according to the close relationship and it is referred to as **EIIMP**.

The rest of this paper is organized as follows. Section II indicates the origin of our research problem and explains the meaning of simultaneous switching gates. Description of **IIMP** is the subject of Section III. We show the algorithm **EIIMP** in Section IV, and Section V is devoted to discussions and presentation of experimental results. Finally, we conclude the paper with a summary and directions for future research.

II. Problem Formulation

Most of the power in CMOS circuits is dissipated by switching of output values of the gates in the circuits. The power dissipated by the circuit is given by:

$$P(V) = L * E^2 * \sum_i Ci * Ti(V) \qquad (1)$$

where V (V= (($u1,u2,\ldots,un$), ($v1,v2,\ldots,vn$)) $uj,vj \in \{0,1\}$) is a primary input vector pair (hereafter, we call it vector pair), which means that the values given to the pins are changed from $u1,u2,\ldots,un$ to $v1,v2,\ldots,vn$. $P(V)$ denotes the power dissipation, Ci is the load capacitance of gate i, E is the supply voltage and L is a constant. And $Ti(V)$ is 1 when the output of the gate i is changed by the vector pair V, otherwise, $Ti(V)$ is 0. Here, static currents are negligible compared to transient currents ([10], [11]). Although the load capacitance Ci varies with the number of fanouts and the other factors of each gate, it is assumed to be identical in order to make the problem simple in this paper. The program

ASP-DAC '97
0-89791-851-7$5.00

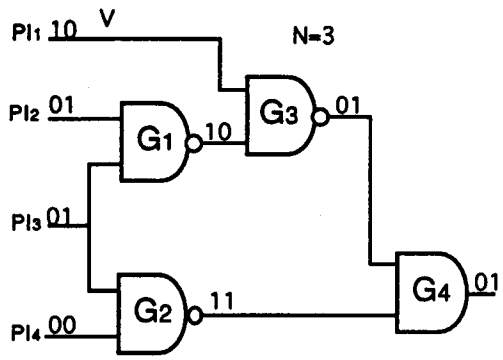

Fig. 1 Simultaneous Switching Gates

can be easily modified by giving the weight, corresponding to the load capacitance, to each gate. As E is a constant as well, the equation(1) can be changed to the following equation (2).

$$P(V) = K\sum_{i} Ti(V) \qquad (2)$$

Here, K is a constant and $\sum Ti\,(V)$ is the number of simultaneous switching gates. If we can find the maximum number of switching gates, we can evaluate the maximum power dissipation.

Next, we explain the meaning of simultaneous switching gates. A gate of which the output value changes with the input vector pair is called the switching gate. In Fig. 1, suppose that the vector pair $V = ((1, 0, 0, 0), (0, 1, 1, 0))$ is given for the primary inputs, *PI1*, *PI2*, *PI3*, *PI4*, then the output values of *G1*, *G3*, *G4*, are changed from 1, 0, 0 to 0, 1, 1, respectively. The output value of *G2* remains unchanged. The gates *G1*, *G3*, *G4*, are switching gates. There are three simultaneous switching gates for the vector pair *V*. Furthermore, the number of simultaneous switching gates is also changed, if the vector pair *V* is changed.

III. Iterative Improvement Method

A. Iterative improvement with a single pin

The iterative improvement method by a single pin is a method to evaluate the maximum number of simultaneous switching gates. It modifies the values of an initial vector pair pin by pin orderly toward the increase of the number of simultaneous switching gates. The procedure is roughed out as follows.

(Step1) Generate an initial vector pair randomly and compute the number of switching gates for it by logic simulation.

(Step2) Select a pin, modify the corresponding values of the selected pin in the vector pair and compute the number of switching gates for the new vector pairs. Additionally, keep the "current best vector pair" that brings the current largest number of switching gates obtained so far.

(Step3) *Step2* is repeated by carrying the "current best vector pair" to the next repetition. In the next repetition, another pin is selected to modify the "current best vector pair". The repetition continues until the current largest number of switching gates does not increase even though the values of every pin are modified.

(Step4) If the current largest number of switching gates exceeds the maximum switching gates *Nmax*, then update *Nmax* with the current largest number of switching gates. *Step1* to *Step3* are repeated by *Rmax* times. Here *Rmax* means the predetermined limit value.

(Step5) The procedure terminates and the maximum switching gates *Nmax* is evaluated.

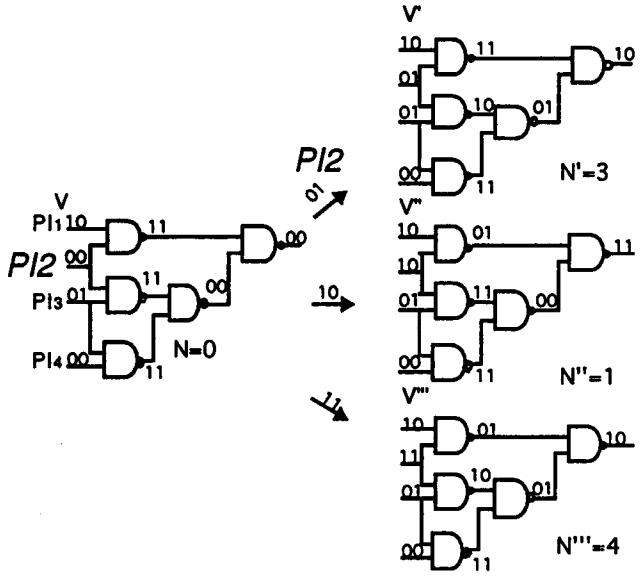

Fig. 2 Example for Pattern Generation

We illustrate *(Step2)* through the example in Fig. 2.

The vector pair V is ((1, 0, 0, 0), (0, 0, 1, 0)) in the left part of Fig. 2 and the number N of switching gates for V equals to 0. Suppose that we select the pin *PI2* at first. Compute the numbers N', N'', N''' of switching gates for V', V'', V''' modified from V. Although V, V', V'', V''' are different from each other, they have the same values except the selected *PI2*. In this case, the value on *PI2* for V is 00, then the values on *PI2* for V', V'', V''' are 01, 10, 11, respectively, and the computed values of N', N'', N''' are 3, 1, 4, respectively. The maximum number of switching gates (N''') and its vector pair (V''') is reserved. As the maximum number of switching gates (N''') in this step has become larger than the number of switching gates (N) for the vector pair (V), we call such a step an improved process and the new vector pair (V''' in this step) is carried to the next step. If every value of N', N'' and N''' does not exceed N, we call it an unimproved process. In this way, the "current best vector pair" V''' that brings the current largest number of switching gates obtained so far is carried to the next step.

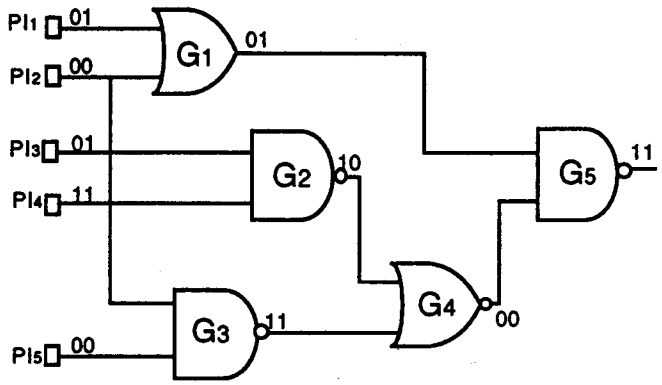

Fig. 3 Example for the Order of Primary Input Pins

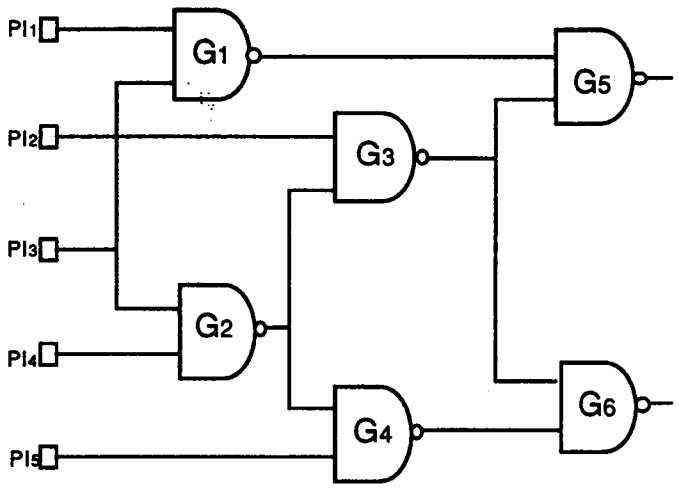

Fig. 4 Example of Calculation for Close Relationship

B. Iterative improvement with multiple pins (IIMP)

In the above iterative improvement procedure, only a single pin is selected for improvement each time. We can select multiple pins instead of a single pin. The number of switching gates is computed for every combination of values on the selected pins. We call the method of iterative improvement with multiple pins **IIMP(*K*)**, Here, *K* is the number of the selected pins. *K* is equal to 1 at the iterative improvement method of Section III-A. The larger the number of selected pins is, the greater is the possibility of finding the optimal solution. When the number of selected pins is equivalent to the number of primary input pins, it results in exhaustive enumeration and the optimal solution can be found. As the exhaustive enumeration needs enormous computational time, it is impracticable. The more the number of selected pins is, the more computational time is needed.

The procedure for **IIMP(*2*)** is given as follows and it can be extended to **IIMP(*K*) (*K*≥3)**.

(Step1) An initial vector pair *V* is generated randomly, and the number *N* of switching gates for *V* is computed.

(Step2) Select two successive pins *(K=2)* for improvement at random. For example, if the pin *PIj* and *PIj+1* are the selected pins, the number of combinational vector pairs is 16 ($=2^{2K}$) and the vector pairs are *V*, *V1*,....,*V15*. Compute the numbers *N1*, *N2*,...,*N15* of switching gates for *V1*, *V2*,...,*V15* modified exhaustively from *V*. Although *V*, *V1*,....,*V15* are different from each other, they have the same values except the values on the pins *PIj* and *PIj+1*. The maximum number among *N*, *N1*,..., *N15* is reserved and its vector pair is carried to the next step.

(Step3) The multiple pins for improvement are moved to *PIj+1* and *PIj+2* and the above procedure is repeated. The iterative improvement continues until the times of successive unimproved processes has reached the number of primary input pins.

(Step4) *(Step1)* to *(Step3)* is repeated with the different initial vector pair generated randomly by *Rmax* times.

(Step5) The procedure terminates and the maximum switching gates *Nmax* is evaluated.

Our method is simple and effective, since it uses the different initial vector pair generated randomly to repeat Rmax times. For the same purpose, we may also make use of the simulated annealing method and we will try this method in our future work.

Although **IIMP** is effective, there still exists a problem that has to be enhanced. Fig. 3 is used to indicate this problem, where *PI1*, *PI2*, *PI3*, *PI4* and *PI5* express the primary input pins in order of the circuit information. Suppose that the input vector pair is ((0, 0, 0, 1, 0), (1, 0, 1, 1, 0)). Switching gates are *G1* and *G2* and the number of switching gates is unable to be improved anymore by **IIMP(*2*)**. If we move the pin *PI5* next to the pin *PI2*, it is possible to change the output value of *G3*, *G4* and *G5* and increase the number of switching gates. Therefore, we try to enhance **IIMP** algorithm in Section IV.

IV. An Enhancement of IIMP Algorithm

In section III-B, we select successive multiple pins in order of the circuit information for improvement. In this section, we take into consideration the relationship of pins and rearrange the pins so that the neighbor pins have the close relationship, then execute **IIMP** just like Section III-B. Here the close relationship is defined as the number of gates in the overlap fan-out regions of pins. The procedure is referred to as **EIIMP** and we can rearrange the pins with short CPU time.

Now, we describe the procedure **EIIMP** as follows.

(Step1) Compute the fan-out region of every pins *PI1*, *PI2*,..., *PIn*, where *n* is the number of pins. Given a set *I*, where *I* is the set of pins. *X* denotes an ordered set of the close multiple pins separated from *I* and *Y* is *X* 's complementary set, that is, $X \cup Y = I$. Here, let *X* be empty.

(Step2) The pin *PI1* is defined as the first element of *X*. *PI1* is called the newest element of the ordered set *X*..

Let $X \Leftarrow X \cup \{PI_1\}$ and $Y \Leftarrow I - \{PI_1\}$.

(Step3) We select the next element of *X*

Calculate the overlap fan-out region between the newest element in the ordered set *X* and every pin in the set *Y*. The pin *PIi* is defined as the second element of *X*, here *PIi* has the largest closeness among the pins of *Y*. Move the pin *PIi*

Overlap (gates)	PI_1	PI_2	PI_3	PI_4	PI_5
PI_1		1	2	1	0
PI_2	1		3	3	1
PI_3	2	3		5	2
PI_4	1	3	5		2
PI_5	0	1	2	2	

Fig. 5 Computation for the Closeness corresponding to Fig. 4

TABLE I
ISCAS'85 BENCHMARK CIRCUITS

Circuit	#Inputs	#Outputs	#Gates
c432	36	7	160
c499	41	32	202
c880	60	26	383
c1355	41	32	546
c1908	33	25	880
c2670	233	140	1193
c3540	50	22	1669
c5315	178	123	2307
c6288	32	32	2416
c7552	207	108	3512

from Y to X; update $X \Leftarrow \{PI_1\} \cup \{PI_i\}$, and $Y \Leftarrow Y - \{PI_i\}$, and the newest element of the ordered set X is PI_i.

(Step4) Repeat *Step3* until $|Y| = 0$, here, $|Y|$ denotes the number of element in set Y.

(Step5) Execute the procedure **IIMP** with ordered set X.

The purpose of this procedure is to make the neighbor pins have the close relationship, so the selection of the first element of X is not important. In *Step2*, we simply select PI_1.

Fig. 4 is an example to explain the above procedure (*Step1* to *Step4*). Here, PI_1, PI_2, PI_3, PI_4 and PI_5 express the primary input pins in order of the circuit information. Fig. 5 represents the computation of closeness corresponding to Fig. 4. First, PI_1 is defined as the first pin of X. Let $X = \{PI_1\}$ and $Y = \{PI_2, PI_3, PI_4, PI_5\}$. Second, because the closeness between PI_1 and PI_3 is largest, so PI_3 is selected the second pin of X. Let $X = \{PI_1, PI_3\}$ and $Y = \{PI_2, PI_4, PI_5\}$. Then, move PI_4 from Y to X as the closeness between PI_3 and PI_4 is largest. Let $X = \{PI_1, PI_3, PI_4\}$ and $Y = \{PI_2, PI_5\}$. Similarly, shift PI_2 from Y to X. Finally, PI_5 is put into X and the procedure concludes. Consequently, the order of close pins (PI_1, PI_3, PI_4, PI_2, PI_5) is gotten and the pins in new order have the closer relationship than those in original order.

V. Experimental Results and Discussion

A. Experiment condition

We set R_{max} =150 and make the experiments for K=2, 3 in view of the limits permitted by time. Our method has been implemented on an IBM compatible personal computer (Pentium-120 MHz). We used the ISCAS'85 benchmark circuits[12] for our experiments. The statistics are summarized in Table I. The numbers of inputs, outputs and gates in each circuit are given.

B. Results

It will be useful to keep these points in mind as we draw conclusions from a single run of **IIMP** and **EIIMP**. The experimental results depend on the generated random number, since the methods use random initial input vector pairs. Table II and Table III denote the minimum and average statistics obtained from experiments with ten random seeds. We present the minimum numbers in Table II as it can represent the worst case. For reference, we show the average numbers in Table III. We don't present the maximum numbers, because they are under the influence of lucky random seeds.

The results in Table II and Table III show the minimum and average comparisons among four kinds of our methods respectively, i.e., **IIMP**(*2*), **EIIMP**(*2*), **IIMP**(*3*) and **EIIMP**(*3*). In the case of our method **EIIMP**(*2*), we have obtained the larger minimum and average results for c2670, c5315 and c7552 than those by **IIMP**(*2*). The method **EIIMP**(*3*) has not only enhanced the minimum results obtained by the **IIMP**(*3*) for most of circuits except for c880 and c6288, but also found the largest average results for c1355, c1908, c2670, c5315 and c7552 among all methods.

C. Discussion

We conclude the following from the above experimental results.

- The more the number of multiple selected pins is, the better is the number of switching gates.
- **EIIMP** can obtain better results for most of circuits than those by **IIMP** and the CPU time by **EIIMP** is approximately equal to that by **IIMP** (Table III). Therefore, the validity of iterative improvement by close multiple selected pins **EIIMP** is indicated.

TABLE II
MINIMUM EXPERIMENTAL RESULTS

Methods / Circuit	IIMP(2)	EIIMP(2)	IIMP(3)	EIIMP(3)
c432	144	145	144	144
c499	116	116	117	117
c880	318	318	319	317
c1355	290	292	293	296
c1908	597	597	599	599
c2670	795	802	799	801
c3540	919	919	920	920
c5315	1466	1474	1470	1477
c6288	1564	1564	1564	1560
c7552	2144	2158	2156	2171

TABLE III
AVERAGE EXPERIMENTAL RESULTS

Methods / Circuit	IIMP(2)		EIIMP(2)		IIMP(3)		EIIMP(3)	
	Nmax	CPUt	Nmax	CPUt	Nmax	CPUt	Nmax	CPUt
c432	145	36	145	40	145	132	145	150
c499	118	30	117	30	118	118	118	117
c880	319	176	319	178	319	686	318	680
c1355	293	143	293	150	298	585	299	605
c1908	600	211	599	218	601	827	601	816
c2670	804	2120	806	2131	805	8395	809	8397
c3540	922	674	921	667	922	2607	921	2525
c5315	1473	3581	1478	3703	1478	13852	1484	14395
c6288	1567	728	1564	741	1566	2647	1564	2649
c7552	2159	6477	2164	6690	2165	25531	2178	25352

Nmax: the maximum number of switching gates by IIMP or EIIMP
CPUt : CPU time
machine: IBM compatible personal computer (Pentium-120 MHz)
(125,000dhrystone)

TABLE IV
EXPERIMENTAL RESULTS BY THE PREVIOUS METHODS

Methods / Circuit	Partial Exhaustive Enumeration [7]				Method Using Genetic Algorithm [9]				Branch-and-Bound Method [8]	
	RND2(N=6)*3		Back Operation*3		M1*4		M2*4			
	Nmax	CPUt*1	Nmax	CPUt*1	Nmax	CPUt*2	Nmax	CPUt*2	Nmax	CPUt*2
c432	---	---	---	---	---	---	---	---	---	---
c499	---	---	---	---	---	---	---	---	---	---
c880	300	180	315	91	318	468	315	373	313	2521
c1355	296	269	290	378	288	203	296	309	305	3337
c1908	591	241	592	395	588	438	587	307	590	3676
c2670	758	826	776	623	791	2439	755	1368	806	6024
c3540	915	454	904	726	919	833	901	495	869	8381
c5315	1429	1406	1412	1511	1402	4528	1449	3005	1434	36000
c6288	1556	1484	1449	823	1538	1226	1539	1100	1516	6511
c7552	2094	2974	2125	2114	2100	6434	2099	4462	2133	10878

*1: cpu time(in sec.), on Fujitsu S-4/LC
*2: cpu time(in sec.), on Sun SS/Classic
*3: the difference for initial pattern
*4: the difference for probability of mutation

Finally, Table IV presents the results obtained using the partial exhaustive enumeration method[7], branch-and-bound method [8] and method using genetic algorithm[9]. **EIIMP** can get better average results for most of circuits in almost the same time as those by the previous methods.

VI. Conclusions

We present an iterative improvement method with the multiple pins **IIMP** and its enhanced method **EIIMP**. **EIIMP** rearranges the order of primary input pins into the close order, then the close multiple pins are selected for improvement just like **IIMP**. For each circuit of ISCAS'85, we compared the number of the switching gates obtained by **EIIMP** to those obtained by **IIMP**. These results show the effectiveness of our method **EIIMP**. Especially, our methods is effective for the large-scale circuits. The more the number of multiple selected pins is, the better is the number of switching gates. Therefore, we will strive for further improvement of our algorithm in search of better results, that is, the improved algorithm will be carried out in a practical time with the more multiple selected pins.

REFERENCES

[1] A. Ghosh, S. Devadas, K. Keutzer and J. White, "Estimation of Average Switching Activity in Combinational and Sequential Circuits," *Proc. 29th Design Automation Conference*, pp.253-259, 1992.

[2] A. P. Chandrakansan, S. Sheng and R. W. Brodersen, "Low-Power CMOS Digital Design," *IEEE Journal of Solid-State Circuits*, vol.27, no.4, pp.473-483, April 1992.

[3] S. Chowdhury and J. S. Barkathullah, "Estimation of Maximum Currents in MOS IC Logic Circuits," *IEEE Trans. on Computer-Aided Design, vol.9 , no.6, pp.642-654, June 1990*

[4] S. Iman and M. Pedram, "Multi-Level Network Optimization for Low Power," *Proc. ICCAD-94*, pp372-377, 1994.

[5] C. C. Teng, A. M. Hill and S. M. Kang, "Eatimation of Maximum Transition Counts at Internal nodes in CMOS VLSI Circuits," *Proc. ICCAD-95*, pp366-370, 1995.

[6] S. Devadas, K. Keutzer, J. White, "Estimation of Power Dissipation in CMOS Combinational Circuits Using Boolean Function Manipulation," *IEEE Trans: on Computer-Aided Design, vol.11, no.83, pp.373-383, March1992*

[7] H. Ueda and K. Kinoshita, "Evaluation of the Maximum Number of Switching Gates for CMOS Circuits," *Technical Report of IEICE, Japan*, vol.93, no.303, pp.49-56, 1993.

[8] H. Ueda and K. Kinoshita, "Evaluation of the Maximum Number of Switching Gates for CMOS Circuits," *IEICE Trans. Inf.&Syst.*, vol.J78-D-I, no.3, pp.367-375, Mar. 1995.

[9] H. Ueda, A. Kiyama and K. Kinoshita, "Evaluation of the Maximum Number of Switching Gates for CMOS Logic Circuits Using Genetic Algorithm," *In the 33rd FTC Meeting, Japan*, 1995.

[10] P. Nigh and W. Maly, "Test Generation for Current Testing," *IEEE Design and Test*, vol.7, no.2, pp.26-38, Feb. 1990.

[11] C. H. Chen and J. A. Abrahan, "High Quality Tests for Switch-Level Circuits Using Current and Logic Test Generation Algorithms," *Proc. Int. Test Conf.*, pp.615-622, Oct. 1991.

[12] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translation in Fortran," *ISCAS'85 : Special Session on ATPG and Fault Simulation*, 1985.