Testing Methods for Detecting Stuck-open Power Switches in Coarse-Grain MTCMOS Designs

Szu-Pang Mu⋆, Yi-Ming Wang⋆, Hao-Yu Yang⋆, Mango C.-T. Chao⋆

Shi-Hao Chen†, Chih-Mou Tseng†, Tsung-Ying Tsai†

⋆Dept. of Electronics Engineering, National Chiao-Tung University, Hsinchu, Taiwan

†Global Unichip Corp, Hsinchu, Taiwan

{genius548@gmail.com, ss91200@gmail.com, mango@faculty.nctu.edu.tw, hockchen@globalunichip.com}

Abstract

Coarse-grain multi-threshold CMOS (MTCMOS) is an effective power-gating technique to reduce IC's leakage power consumption by turning off idle devices with MTCMOS power switches. In this paper, we study the usage of coarse-grain MTCMOS power switches for both logic circuits and SRAMs, and then propose corresponding methods of testing stuck-open power switches for each of them. For logic circuits, a specialized ATPG framework is proposed to generate a longest possible robust test while creating as many effective transitions in the switch-centered region as possible. For SRAMs, a novel test algorithm is proposed to exercise the worst-case power consumption and performance when stuck-open power switches exist. The experimental results based on an industrial MTCMOS technology demonstrate the advantage of our proposed testing methods on detecting stuck-open power switches for both logic circuits and SRAMs, when compared to conventional testing methods.

I. Introduction

IC's leakage power consumption has greatly increased in the past due to the continual scaling of process technologies [1] [2]. One solution to reduce this leakage power is to use *power gating*, which will turn off the power of the devices with header or footer power switches when the circuit is idle. This power-gating technique is especially useful for those portable, event-driven applications, such as cell phones, GPS, or PDAs, whose certain functions (mostly multi-media functions) are in the idle state most of the time.

Being able to simultaneously reduce leakage power for the always-on circuits and maintain performance for the power-gated circuits, the *Multi-threshold CMOS* (MTCMOS) technology emerges to be an attractive process technology to realize the power-gating designs. For an MTCMOS design, the always-on power-gating circuits (including header/footer power switches, retention flip-flips, and always-on buffers) are implemented in high-V_t transistors, such that their leakage current can be lowered during the sleep mode. On the other hand, the power-gated circuits are implemented in low-V_t transistors, such that their performance can be increased during the active mode. By its granularity, the MTCMOS designs can be classified into (1) the *fine-grain* MTCMOS designs, in which a power switch is built into each cell, and (2) the *coarse-grain* MTCMOS designs, in which a relatively small number of power switches are used to supply the power mesh of all power-gated devices. In fact, fine-grain MTCMOS designs require high area overhead and may not be practical or cost-effective for large industrial designs. Therefore, this paper focuses on the discussion of coarse-grain MTCMOS technologies only.

During the past decade, a lot research effort has been put into the area of MTCMOS technologies. One group of research works focused on developing MTCMOS power-gating structures to lower the power consumption produced during the sleep-to-active mode transition [3] [4] or provide the flexibility for various demands of the power-performance trade-off [5]. Another group of research works attempt to optimize different design parameters under different constraints (such as circuit performance, mode-transition power consumption, power-supply noise, area overhead, and wake-up time) by using the sleep-transistor sizing [7] [8], circuit clustering [9], wake-up scheduling [10] [11], or simultaneous clustering and scheduling [12]. However, to the best of our knowledge, no previous work has discussed the testing-related issues for coarse-grain MTCMOS designs so far.

In current coarse-grain MTCMOS design flow, designers first need to determine the type and the number of (header) power switches used for connecting the true VDD with the power mesh of the virtual VDD, which can affect the IR drop between the true VDD and the virtual VDD as well as the affordable current supply to the power-gated circuits. A larger number of power switches in use can lead to a lower IR drop and a larger current supply, but at the same time requires a larger area overhead. Next, the given number of power switches will be evenly placed all over the chip except the hard macros, such as SRAMs or analog circuits, where no MTCMOS switches can be inserted. After the power-switch placement, the power switches are then be routed with a chain-style connection or a tree-style connection. A chain-style connection may result in a smaller sleep-to-active power consumption but a longer wake-up time while a tree-style connection does the other way. For most applications, chain-style connection is more preferred since adding several micro seconds to the wake-up time may not be felt by users when they wake up the system. But a large sleep-to-active power consumption may directly damage and fail the chip.

Once the power network is built, the rest design flow is the same as the conventional CMOS designs, including the scan-chain insertion and test generation. However, the conventional test generation only targets the modeled faults on the power-gated circuits, and the functionality of a power switch may not be guaranteed. Even though the wake-up and acknowledge signals can be provided from a chain-style switch connection, we can only know whether the wake-up signal is successfully sent to each switch. The actual on or off of a power switch remains unknown. If a power switch is stuck-on, the power consumption of the gated circuits may be significantly increased during its sleep mode. Such a stuck-on power switch can be easily defected by measuring the sleep-mode current. On the other hand, if a power switch is stuck-open, the current supply near the region of the faulty switch may become weaker. However,

978-1-4244-8194-1/10/$26.00 ©2010 IEEE

its neighbor power switches may help to supply the current to the virtual-VDD power mesh originally supplied by the faulty power switch. As a result, to detect such a stuck-open power switch may require specialized test patterns which can create a large number of 0-to-1 signal transitions within the region centered by the faulty power switch and then propagate a signal transition within the region through a critical long path.

Previous research works [13] [14] [15] [16] [17] [18] have suggested that testing delay faults needs to consider the effect of power supply noise in order to exercise the worst-case delay of a circuit and in turn guarantee the circuit's performance. ATPG tools were also developed to generate test pattern which can maximize the power supply noise while sensitizing the target delay fault. However, our test generation for MTCMOS power-switch faults is different from the previous power-noise test generation for delay faults due to the following reasons. First, our fault model is associated with power switches, while the target fault model for power-noise test is based on critical paths. It also means that the total number of faults considered in our test generation is equal to the number of power switches, while the total number of path delay faults can be huge. Second, our test generation attempts to maximize the signal transitions mainly within the region centered by a target power switch, while the power-noise test attempts to maximize the signal transitions for the whole chip or along the target path. Third, in our test generation, we only need to maximize the number of signal transitions with single direction, i.e., 0-to-1 transitions if header power switches are used. On the other hand, the signal transitions required in the power-noise test can be bidirectional.

In this paper, we first introduce the design flow of coarse-grain MTCMOS technologies and study the usage of MTCMOS power switches for logic circuits and SRAM macros. Next, we propose a SAT-based ATPG framework to generate the *hot-spot-attack delay-fault* (HSAD) tests, which can help to detect the stuck-open power switches used for logic circuits. Also, we propose a memory test algorithm with a specialized address traversing to detect stuck-open power switches used for SRAM macros. The experiments are conducted based on the MTCMOS technology provided by an IC foundry [19] and demonstrates that the proposed testing methods can effectively help to detect stuck-open power switches for both logic circuits and SRAM macros, when compared to the conventional testing methods.

II. Background

In this section, we will introduce the MTCMOS technology used in a major IC foundry [19] and its corresponding design flow. In the rest of this paper, we will mainly focus on the MTCMOS designs using header power switches. The proposed test generation can be easily applied to footer power switches in a similar manner.

A. Architecture of Header-Switch MTCMOS Designs

Figure 1 first illustrates the overview of an MTCMOS design using header switches. The power-gated low-V_t cells are connected to the virtual V_{DD}, whose power supply is controlled by the MTCMOS switches placed between the virtual V_{DD} and true V_{DD}. When the system turns on the wake-up-request signal, the header switches are turned on in order so that the virtual V_{DD} can obtain the power from the true V_{DD}. After the system receives the wake-up-acknowledge signal, the system starts to send jobs to the gated logics. When the system turns off the wake-up-request signal, the switches are turned off and the virtual V_{DD} cannot provide any power to the gated cells, meaning that no leakage current can be generated on the gated cells.

Figure 2 illustrates the true-V_{DD} mesh and the virtual-V_{DD} mesh used in general MTCMOS designs. Both true-V_{DD} mesh and virtual-V_{DD} mesh are formed by an outside power ring connected with horizontal stripes and vertical stripes inside the ring. But the virtual-V_{DD} mesh contains additional horizontal rails, where

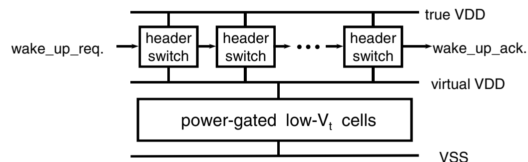


Fig. 1. **MTCMOS power-supply architecture using header switches**

standard cells can directly connect to. The connection between true-V_{DD} mesh and virtual-V_{DD} mesh is made by the MTCMOS power switches, which locate only on the intersection of a vertical true-V_{DD} stripe and a horizontal virtual-V_{DD} rail. Note that the V_{SS} mesh is omitted in Figure 2 for simplicity. The V_{SS} mesh is actually similar to the virtual-V_{DD} mesh, and the horizontal V_{SS} rails are placed alternately with the horizontal virtual-V_{DD} rails, such that a standard cell can directly connect to.

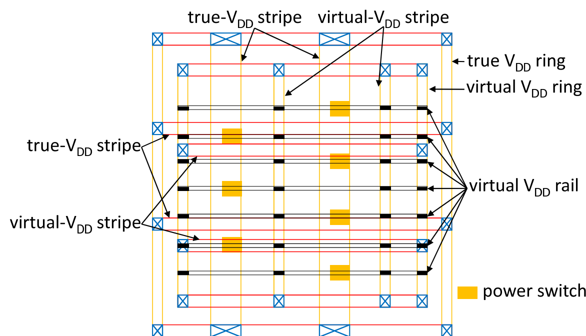


Fig. 2. **Physical layout of true-V_{DD} mesh and virtual-V_{DD} mesh.**

B. Switch Allocation

The number of switches in use is determined by the worst IR drop which can be tolerated between the true V_{DD} and virtual V_{DD}. Then, with a pre-defined placement pattern (mostly checkerboard), the switches are evenly placed over the IC except the hard macros, where no switch can be inserted. If the encountered hard macro is an SRAM core or any hard IP using the same power domain as the standard cells, extra switches are placed along the boundaries of the hard macro to strengthen its power supply. If the encountered hard macro is an analog IP, which has its own power domain, the switch-placement pattern around the boundaries remains the same. Figure 3 shows an exemplary switch allocation with hard macros.

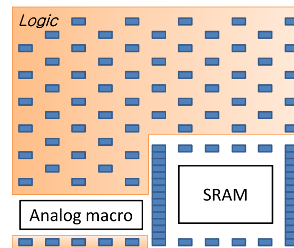


Fig. 3. **Switch allocation with hard macros.**

C. MTCMOS Header Switches and Switch Routing

The coarse-grain library [19] provides two types of header switches: the *single-input switches* and *double-input switches* as showed in Figure 4(a) and 4(b), respectively, where all the inverters in Figure 4 are directly supplied by true V_{DD}. If the single-input

switches are used, the switches are serially connected as Figure 5(a), where the $NSIn$ signal of the first routed switch is connected to system's wake-up-request signal and the $NSOut$ of the last routed switch is connected to the system's wake-up-acknowledge signal. Since the pins of wake-up-request and wake-up-acknowledge signals usually locate next to each other, the routed path of single-input switches looks like a Hamiltonian cycle.

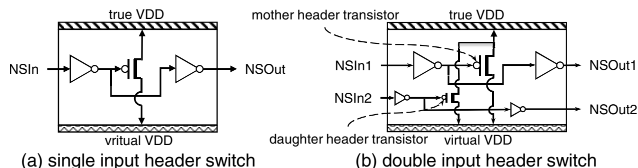


Fig. 4. **Types of MTCMOS header switches.**

If the double-input switches are used, the switches are connected as Figure 5(b), where the wake-up-request and wake-up-acknowledge signals are connected to the $NSIn2$ and $NSOut1$ signals of the first routed switch, respectively. Also, the $NSOut2$ signal of last routed switch is connected to the $NSIn1$ signal of itself. Therefore, the routed path of double-input switches looks like a Hamiltonian path. The routed path can end at any switch in the design. Note that, using such a connection between switches, the daughter pMOS transistor (the one with smaller driving capability) of a switch will be turned on first by $NSIn2$ during the sleep-to-active mode transition. After the daughter pMOS transistors of all switches are turned on, the mother pMOS transistors start to turn on from the last routed switch. As a result, the peak power consumption during the sleep-to-active mode transition can be reduced since charging virtual-V_{DD} mesh with daughter pMOS transistors is slower than mother pMOS transistors. After the virtual-V_{DD} is charged, the mother pMOS transistors will be turned on to strengthen the current supply during the active mode. Thus, using double-input switches is recommended by [19].

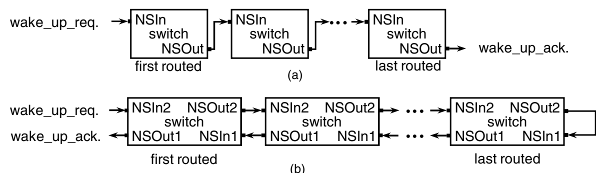


Fig. 5. **Switch routing for (a) single-input and (b) double-input switches.**

However, if some switches cannot be successfully included in the Hamiltonian path, those switches can only be connected by using a branch as shown in Figure 6 (same situation to single-input switches as well). For those switches on a branch, its $NSOut1$ cannot be sent back to the wake-up-acknowledge signal, meaning that the system cannot know whether the wake-up-request signal is successfully sent to those switches, which may create extra testing problem.

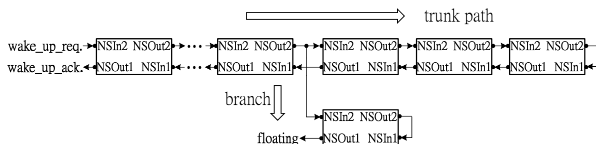


Fig. 6. **Trunk path and branch for switch routing.**

III. Testing Stuck-open Switch for Logic Circuit

In this section, we will introduce the proposed SAT-based ATPG framework for generating the *hot-spot-attack delay-fault* (HSAD) tests, whose objective is to generate a worst-case delay when a power switch is stuck-open. Thus, a HSAD test should propagate a signal transition inside the target region of the switch through the longest possible path and, at the same time, create as many 0-to-1 transitions within the target region as possible. In other words, the detection of a stuck-open switch is made through the detection of the worst-case delay when applying the HSAD tests at speed. In our ATPG framework, we assume that the double-capture scheme is used during the at-speed testing. In addition, this ATPG framework is applied after the physical layout of the target MTCMOS design can be obtained. In our MTCMOS design flow, we use SoC Encounter [20] to generate the physical design.

A. Overview of HSAD Test Generation

Figure 7 shows the overview of the proposed ATPG framework, which requires the following input files:

- **.v** file: the Verilog file of the target MTCMOS design. This file is an input of the timing analysis tool, PrimeTime [21].
- **.db** file: the database file describing the timing information. This file is an input of the timing analysis tool as well.
- **.bench** file: the netlist file of the target MTCMOS design. This netlist format is only used by our own ATPG and can be directly transferred from the **.v** file.
- **.def** file: the file describing the physical-design information of the MTCMOS design.

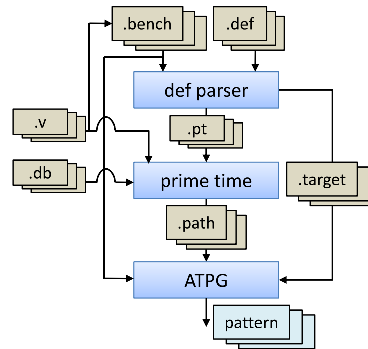


Fig. 7. **Overall flow of the proposed HSAD ATPG framework.**

At the beginning of this ATPG framework, we feed the **.bench** file and the **.def** file to a DEF parser. This parser will first output a **.target** file, which lists the cells (with the naming used in the **.bench** file) within the target region of each power switch. Also, this parser will output a **.pt** file, which is written in the PrimeTime script format and will ask PrimeTime to report the first n longest paths through each cell within each target region. In our experiment, we set n to 10. Next, based on the **.pt** file, PrimeTime will output the corresponding paths and its timing to the **.path** file. Last, our HSAD ATPG will generate the HSAD tests based on the netlist (**.bench** file), the cells in each target region (**.target** file), and the information of long paths (**.path** file).

B. Target Region of a Power Switch

Conceptually, the target region of a given switch is defined as the area of the virtual-V_{DD} mesh where most of its current supply is contributed by the given switch. In our framework, we determine whether a location on the virtual-V_{DD} mesh belongs to the target region of a switch by the physical mesh distance traversing from the location to the given switch. If its physical mesh distance to the given switch is not larger than the distance to any other switch, then the location belongs to the target region of the switch. Figure 8 illustrates how to identify the target region of a switch on an exemplary virtual-V_{DD} mesh. In Figure 8, the "borderlines" between two target regions are highlighted by a "×", where its mesh distance

to either switch is the same. As a result, if a switch is stuck-open, then the current supply of the cells within its target region may be affected more than that of the other cells.

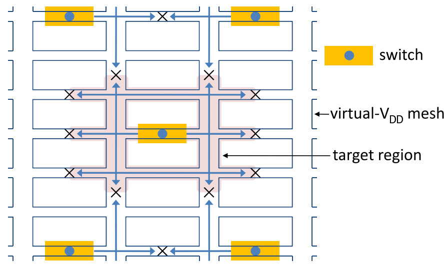


Fig. 8. **Finding the target region of a switch.**

C. Detailed Steps of HSAD ATPG

For the target region of a switch, the objective of a HSAD test is to create a maximal number of 0-to-1 transitions within the target region while generating a robust test on the longest possible path. Note that we use only the header power switches in this paper and hence only the 0-to-1 transitions are considered. If footer switches are used, then we need to maximize the number of 1-to-0 transitions. Figure 9 first shows the five steps (A to E) of generating a HSAD test for a target region.

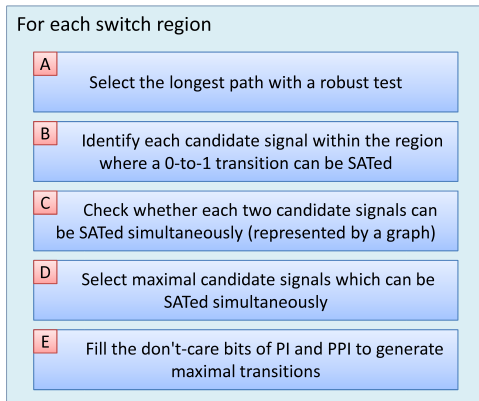
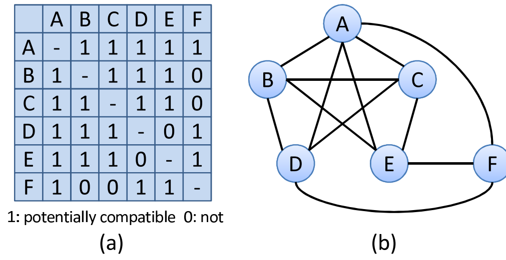


Fig. 9. **Steps of generating a HSAD test.**

1) Step-A: We first sort the paths passing through the target region (reported in **.path** file) by its delay. Then we start from the longest path and apply an efficient circuit-SAT solver [22] to determine whether a robust test can be generated for the longest path. If SATed, the path is selected as the target path and will be repeatedly used in later steps. If not SATed, we try the next longest path until we find one with a robust test. The same SAT solver will be applied later when we try to maximize the number of 1-to-0 transitions in the target region. The conditions for setting a robust test can be found in [23].

2) Step-B: For each cell within the target region, we will apply the SAT solver to check whether a 0-to-1 transition on the cell can be successfully generated along with a robust test of the selected path in Step-A. If SATed, the cell becomes a valid candidate cell for maximizing the 0-to-1 transitions within the target region.

3) Step-C: From all the candidate cells identified in Step-B, we pick each two of them and apply the SAT solver to check whether a 0-to-1 transition can be generated on both candidate cells simultaneously along with a robust test for the selected path. If SATed, these two cells are considered as *potentially compatible*. We record all the potentially compatible pairs by a two-dimensional array as shown in Figure 10(a). Figure 10(b) shows the corresponding *potentially-compatible graph* built according to the array, where the two cells connected by an edge are potentially compatible.



	A	B	C	D	E	F
A	-	1	1	1	1	1
B	1	-	1	1	1	0
C	1	1	-	1	1	0
D	1	1	1	-	0	1
E	1	1	1	0	-	1
F	1	0	0	1	1	-

Fig. 10. **(a) Array recoding the potentially-compatible information; (b) The corresponding potentially-compatible graph of (a).**

4) Step-D: In this step, we apply a greedy-based algorithm to iteratively adding a 0-to-1 transition into the target region. The SAT problem first starts with the conditions of generating a robust test for the selected long path. No candidate cell is selected initially. Then we iteratively choose the unvisited candidate cell with the largest degree on the potentially-compatible graph. If tie, we randomly pick one. Next, we add the 0-to-1 transition of the selected cell into the original conditions of the SAT problem, and apply the SAT solver to the SAT problem. If STAed, we select the candidate cell, keep all conditions in the SAT problem, and remove all candidate cells not directly connected to the selected cell from the graph (along with their edges). Those removed cells can never be SATed along with the selected cell since they are not potentially compatible. If not SATed, we exclude the cell, withdraw the new added condition from the SAT problem, and remove the cell from the graph.

Figure 11 shows an example of Step-D based on the potentially-compatible graph in Figure 10(b). In the first iteration, we choose cell A (with the largest degree, 5) and is SATed. Since cell A is connected to all other cells, no cell is removed. In the second iteration, several cells have the largest degree of 4 and we randomly pick cell B. Assuming cell B is SATed, then we need to remove cell F since it is not connected to cell B, which changes the degree of the cells connecting to cell F. In the third iteration, we choose cell C (with the largest degree 3). Assuming that cell C cannot be SATed, we need to remove cell C and again update the degree of other cells. By repeating the above process, we can obtain the final result as shown in Figure 11.

Iteration	A	B	C	D	E	F	
1	5	4	4	4	4	3	choose A, (SATed)
2	V	4	4	4	4	3	choose B, (SATed), remove F
3	V	V	4	3	3	X	choose C, (not SATed), remove C
4	V	V	X	2	2	X	choose D, (SATed), remove E
5	V	V	X	V	X	X	final result

V: selected X: excluded

Fig. 11. **Degree of each cell for an 5-iteration example of Step-D.**

5) Step-E: After we generate the robust test and maximize the 0-to-1 transitions within the target region, there may still exist some don't-care bits in the two-cycle test pattern. The objective of Step-E is to maximize the transitions occurring on PPIs and PIs between the two cycles by assigning the don't-care bits. With more transitions at PPIs and PIs, the number of 0-to-1 transitions in the whole chip can also be increased more likely, which may lead to a larger IR drop and a longer delay. In order to achieve this objective efficiently, an iterative approach is applied in Step-E. In each iteration, we take one of the following actions, which are listed in the order of their priority, if applicable.

- 1st priority: Assign an opposite value to a don't-care PPI or PI at the first cycle if its value at the second cycle is already determined.
- 2nd priority: Assign an opposite value to a don't-care PI at the second cycle if its value at the first cycle is already determined.
- 3rd priority: Try to SAT the opposite value to a don't-care PPI at the second cycle if its value at the first cycle is already determined.
- 4th priority: Randomly assign a value to a don't-care PPI at the first cycle or a don't-care PI at both cycle.

D. Experimental Results

In this subsection, we implement ISCAS benchmark circuits with a coarse-grain MTCMOS technology provided by an IC foundry [19], and then apply different test patterns to the circuits with or without a stuck-open defect at a power switch. For s1196, s5378, and s9234, their power consumption is low and only one power switch is enough to supply the current. However, in order to approximate the impact of injecting a stuck-open defect to a switch, we need multiple power switches in a design. Thus, we decided to over-design each of s1196, s5378, and s9234 by using 4 power switches. All power switches are double-input switches (as introduced in Section II-C). The physical layout of each MTCMOS design, including switch allocation and switch routing, is generated by using SoC Encounter [20].

Table I lists the experimental result of s9234. In this experiment, we first use our proposed HSAD ATPG to generate one HSAD test for each switch (denoted as SW1 to SW4). Then we extract the SPICE file from the layout of s9234 with StarRCXT [24] and measure the delay of different test patterns by using HSPICE. Row 2 of Table I reports the delay of each HSAD test based on the defect-free circuit. Row 3 reports the delay of each HSAD test based on the circuit with a stuck-open defect injected at the target switch of the HSAD test. Note that the defect is injected by adding a 10GΩ resistance between the source of the target pMOS transistor and the true VDD mesh. Row 4 lists the difference between Row 3 and Row 2. Next, for each power switch, we generate a robust test for the same long path sensitized in the corresponding HSAD test and randomly fill the don't-care bits (the way conventional path-delay ATPG does). Then we conduct the same experiment for these path-delay tests as for the HSAD tests. The corresponding results are reported in Row 5 to Row 7.

methods	circuits	pattern for				avg
		SW1	SW2	SW3	SW4	
our HSAD ATPG	without defect(a)	1.345ns	1.43ns	1.4ns	1.42ns	-
	with defect(b)	1.395ns	1.48ns	1.44ns	1.48ns	-
	b - a	0.05ns	0.05ns	0.04ns	0.06ns	0.05ns
conventional path-path ATAPG	without defect(a)	1.23ns	1.355ns	1.34ns	1.36ns	-
	with defect(b)	1.25ns	1.375ns	1.365ns	1.38ns	-
	b - a	0.02ns	0.02ns	0.025ns	0.02ns	0.021ns

TABLE I

Delay of test patterns based on s9234 with and without injecting the stuck-open defect to the corresponding power switch.

By comparing Row 3 with Row 6, we can find that a HSAD test can always generate a longer delay than the conventional path-delay test on the same defective circuit. Also, the average difference between its delays with and without the stuck-open switch is 0.05ns for a HSAD test while this difference is only 0.021ns for a conventional path-delay test. This result shows that a HSAD is indeed more sensitive to the existence of a stuck-open defect at the target switch and hence is more effective on detecting a stuck-open power switch.

In Table II and Table III, we conduct the similar experiments for s5378 and s1196. The same trend as in Table I can be observed in Table II and Table III as well. Note that using HSPICE to simulate a test pattern on the complete circuit of s9234 takes around 24 hours. To run similar SPICE simulation on a larger ISCAS circuit (such as s35932, s38417, or s38584) may take weeks or even months. Thus, we have already generated the HSAD tests for s35932, s38417, and s38584 but we do not have the SPICE result for these three circuits. Table IV reports more detailed information about the HSAD tests for 6 ISCAS benchmark circuits. Column 2 to Column 7 list the number of power switches, the average delay of the structural longest path passing through a target region, the average delay of the path sensitized by a HSAD test, the average number of cells in a region, the average number of 0-to-1 transitions within the target region generated by a HSAD test, the average number of total 0-to-1 transitions in the desing generated by a HSAD test, respectively.

methods	circuits	pattern for				avg
		SW1	SW2	SW3	SW4	
our HSAD ATPG	without defect(a)	1.16ns	1.16ns	1.165ns	1.09ns	-
	with defect(b)	1.2ns	1.19ns	1.195ns	1.12ns	-
	b - a	004ns	0.03ns	0.03ns	0.03ns	0.0325ns
conventional path-delay ATPG	without defect(a)	1.125ns	1.14ns	1.13ns	1.05ns	-
	with defect(b)	1.145ns	1.16ns	1.16ns	1.075ns	-
	b - a	0.02ns	0.02ns	0.03ns	0.025ns	0.02375ns

TABLE II

Delay of test patterns based on s5378 with and without injecting the stuck-open defect to the corresponding power switch.

methods	circuits	pattern for				avg
		SW1	SW2	SW3	SW4	
our HSAD ATPG	without defect(a)	0.96ns	0.94ns	0.85ns	0.875ns	-
	with defect(b)	0.975ns	0.98ns	0.89ns	0.91ns	-
	b - a	0.015ns	0.04ns	0.04ns	0.035ns	0.0325ns
conventional path-delay ATPG	without defect(a)	0.96ns	0.94ns	0.82ns	0.86ns	-
	with defect(b)	0.97ns	0.96ns	0.845ns	0.88ns	-
	b - a	0.01ns	0.02ns	0.025ns	0.02ns	0.01875ns

TABLE III

Delay of test patterns based on s1196 with and without injecting the stuck-open defect to the corresponding power switch.

circuits	# of switch	average result for each target region				avg # of total 0-to-1 trans.
		structural max path	selected path	# of cells	# of 0-to-1 trans.	
s38417	12	1.69ns	1.57ns	606.7	164.1	2154.7
s35932	20	1.35ns	1.35ns	294.6	102	3964.4
s38584	12	2.01ns	1.83ns	650.4	132.1	1849.2
s9234	4	1.5ns	1.44ns	317.3	85	434.7
s5378	4	1.1ns	1.07ns	283	94	376.3
s1196	4	1.05ns	0.98ns	66.8	16.8	90.8

TABLE IV

Statistics of HSAD tests for different benchmark circuits.

Table V reports the runtime of each step of the HSAD ATPG for the 6 ISCAS circuits. In Table V, s38417 requires the longest runtime (around 3.3 hours) and 96.5% of its runtime is spent on Step-C, which needs to build the potentially-compatible graph and requires a $O(n^2)$ time complexity (n is the number of candidate cells in the target region). Once the potentially-compatible graph is constructed, then maximizing the number of 0-to-1 transitions in the target region (Step-D) can be done efficiently with our greedy-based iterative algorithm as shown in Section III-C.4. Table VI compares the number of 0-to-1 transitions within the target region generated by our HSAD ATPG with that generated by an optimum solution, which enumerates all the possible combinations of candidate cells for the SAT problem by a branch-and-bound search. As the result shows, this number of 0-to-1 transitions generated by our HSAD ATPG is very close to the optimum solution for most target regions, while our runtime can be more than several-hundred times faster for s9234 and s5378.

circuits	PrimeTime + DEF parser	Step-A	Step-B	Step-C	Step-D	Step-E	total runtime
s38584	0.4	4.4	27.1	6714.5	24.7	301.4	7072.5
s38417	0.3	3.6	39.2	11489	31.5	339.4	11903
s35932	0.5	4.1	7.4	341.3	5.1	437.2	795.6
s9234	0.2	0.4	1.4	133.5	1.1	2.9	139.5
s5378	0.2	0.1	0.6	78.3	1	2.1	82.3
s1196	0.2	0.1	0.1	1	0.1	0.1	1.6

TABLE V

Runtime (in seconds) for each step of the proposed HSAD ATPG.

circuits	methods	pattern for				avg	runtime(s)
		SW1	SW2	SW3	SW4		
s1196	HSAD test(a)	11	18	16	22	16.75	1.4
	optimal(b)	20	20	16	23	19.75	61.4
	a / b	55.0%	90.0%	100.0%	95.7%	84.8%	-
s5378	HSAD test(a)	74	96	104	102	94	82
	optimal(b)	75	99	106	108	97	20289.3
	a / b	98.7%	97.0%	98.1%	94.4%	96.9%	-
s9234	HSAD test(a)	69	65	75	86	73.75	138
	optimal(b)	71	74	79	88	78	46772.4
	a / b	97.2%	87.8%	94.9%	97.7%	94.6%	-

TABLE VI

Comparison between the numbers of 0-to-1 transitions generated by our HSAD tests and an optimal solution.

IV. Testing MTCMOS Switches for SRAMs

Testing stuck-open MTCMOS power switches for SRAMs is different from that for logic circuits. For most SRAM macros provided by current IC foundries, MTCMOS power switches are usually not embedded inside the SRAM macros. When integrating such a SRAM macro to a MTCMOS design, the MTCMOS power switches can only be placed outside the SRAM macro, not like the logic circuits, on which the MTCMOS power switches are evenly distributed within the logic cells. Thus, in order to guarantee enough power supply of an SRAM macro, designers usually place as many power switches as possible along the boundaries of the SRAM macro, which may provide much more power than the SRAM macro really requires. As a result, one stuck-open switch may not affect the functionality of the SRAM macro. Only multiple stuck-open switches at the same time may fail the SRAM macro. Multiple stuck-open switches may occur when the switch routing network (as shown in Figure 5) fails to send the wake-up-request or wake-up-acknowledge signal to a series of connected switches, such as an open defect on the wire of a switch-network's branch as shown in Figure 6.

In this section, we use an in-house 256Kb SRAM design as an example to discuss the impact of multiple stuck-open switches and the corresponding test algorithms. Figure 12 illustrates the overall architecture of the SRAM design, which is implemented in an UMC process technology, occupies 283x575μm^2 chip area, and operates at 300MHz. This SRAM array contains 512 word-lines, each word-line contains 16 words, and each word contains 32 bits. Note that the layout topology of this SRAM array utilizes the distributed folding scheme, where the ith bit of the jth word is adjacent to the ith bit of the (j+1)th word, not the (i+1)th bit of the original jth word. Thus, when performing a read or write operation to a word, the operating cells, bit-line pairs, and sense amplifiers are actually distributed over the whole design, not like logics circuits, where we can create an intensive signal transitions with in a target region.

In fact, the most power consuming area of the SRAM macro is the pre-charge circuit, which locates on the lower region of the SRAM macro as shown in Figure 12. As a result, the impact of stuck-open switches on the bottom of the SRAM macro is stronger than that on the top of the SRAM macro. Figure 13 shows the SPICE-simulation result (at TT corner, 25°C) of a bit-line pair when all the power switches on top of the macro are stuck-open and we repeatedly write opposite data background to the same word

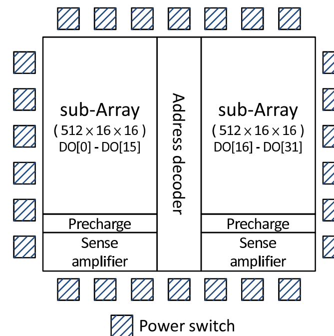


Fig. 12. **Overview of the 256Kb SRAM macro used in our experiment.**

for four consecutive cycles. On the other hand, Figure 14 shows the corresponding simulation result when all the power switches on bottom of the macro are stuck-open.

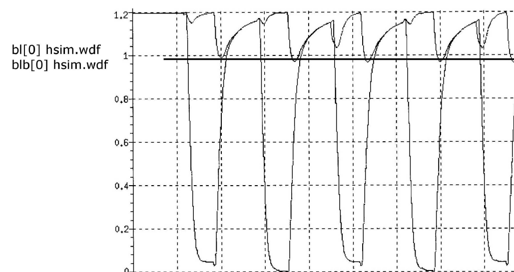


Fig. 13. **Voltage of a bit-line pair for 4 consecutive write operations when all switches on the top of the macro are stuck-open.**

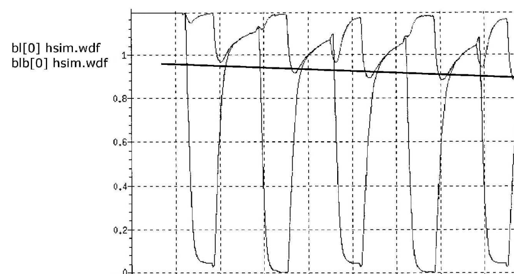


Fig. 14. **Voltage of a bit-line pair for 4 consecutive write operations when all switches on the bottom of the macro are stuck-open.**

As the result shows, after the four consecutive write operations, the bit-line pair in Figure 13 can still be pre-charged to almost the same voltage level, while the pre-charged voltage at the bit-line pair in Figure 14 is continually decreased. If we keep performing write operations on the same bit-line pair in 14, a write fail may occur due to its low pre-charged voltage. This result first demonstrates that the lower region of the SRAM macro indeed requires more current supply. Also it shows that, when designing the test algorithm for detecting the stuck-open faults, we need to repeatedly perform a write operation at the same bit-line pair to capture the stuck-open defects at power switches. Otherwise the pre-charge circuit for the bit-line pair will have enough time to recover the voltage.

In this paper, we suggest the following test algorithm to create the maximal power consumption and detect the stuck-open switches. The proposed test algorithm includes the following two march elements:

- 1st element: Write each word with alternating opposite data background and follows a *bit-line-fixed address-complement traversing*.
- 2nd element: Read each word in the reverse address traversing as in the 1st element.

Figure 15 illustrates the bit-line-fixed address-complement traversing for our 512x16x32b SRAM. The address of this SRAM macro contains 13 bits. The first 9 bits select the 512 word-lines and the last 4 bits select the 16 words at a word-line. The bit-line-fixed address-complement traversing will first change the first 9 bits while fixing the last four bits, such that the same bit-line pairs are consecutively operated. Also, the traversing order of the first 9 bits is an general increasing order of the 9-bit address alternating with a complement 9-bit address (as shown in Figure 15).

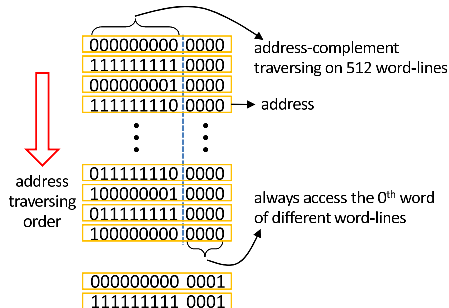


Fig. 15. **The bit-line-fixed address-complement traversing.**

In the following experiment, we will compare the proposed test algorithm with two other algorithms with one write element and one read element (denoted as *Algor2* and *Algor3*). Algor2 uses the alternating opposite data background for write operations (same as our algorithm) but follows the general incremental address traversing. Algor3 uses the identical data background for write operations and the general incremental address traversing. Table VII shows the average and peak power by each algorithm for read operations and write operations, respectively, assuming that all the power switches at the top are stuck-open. As the result shows, the proposed algorithm can always generate a higher power than Algor2 and hence is more likely to create the worst-case IR drop condition for detecting the stuck-open switches. This result also demonstrate the effectiveness of applying the bit-line-fixed address-complement traversing. In addition, the power consumption by Algor2 is larger than the Algor3 as well, showing that the alternating opposite data background can effectively help to create a worst-case IR drop.

	ours	Algor2	Algor3
avg write power	12.33mW	12.22mW	12.18mW
avg read power	15.50mW	15.31mW	15.23mW
peak write power	22.70mW	22.40mW	22.20mW
peak read power	20.80mW	20.60mW	20.50mW

TABLE VII

Power consumption resulting from different memory test algorithms.

Table VIII further shows the average word-line turn-on delay resulting from each algorithm when all top switches are stuck-open. As the result shows, our proposed algorithm can generate a slower average word-line turn-on delay than the other two algorithms. This slow word-line turn-on delay mainly results from its address-complement traversing, which can create maximal transitions on the address decoders. This result again demonstrates the advantage of using the bit-line-fixed address-complement traversing.

	ours	Algor2	Algor3
WL turn-on time	0.2679ns	0.2673ns	0.2671ns

TABLE VIII

Average word-line turn-on delay of different memory test algorithms.

V. Conclusion

This paper presented an SAT-based ATPG framework to generate HSAD tests, whose propagation delay is sensitive to the functionality of the power switches and hence can help to detect the stuck-open power switches for logic circuits. Next, we proposed a memory test algorithm with a specialized address-traversing order, which can effectively exercise a worst-case performance and help to detect the multiple stuck-on power switches for SRAM macros. The experimental results based on a current MTCMOS technology demonstrated the advantage of the proposed ATPG framework and memory test algorithm over the conventional testing methods.

References

[1] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits," *Proceeding of the IEEE*, vol. 91, No. 2, Feb. 2003, pp. 305-327.

[2] J. Kao, S. Narendra, and A. Chandrakasan, Subthreshold Leakage Modeling and Reduction Techniques, *ACM/IEEE International Conference on Computer Aided Design*, pp.141-148, Nov., 2002.

[3] Z. Liu and V. Kursun, Charge Recycling Between Virtual Power and Ground Lines for Low Energy MTCMOS, *IEEE/ACM International Symposium on Quality Electronic Design*, pp.239-244, March 2007.

[4] E. Pakbaznia, F. Fallah, and M. Pedram, Charge Recycling in MTCMOS Circuits: Concept and Analysis, *ACM/IEEE Design Automation Conference*, pp.97-102, July 2006.

[5] S. Kim, S. V. Kosonocky, D. R. Knebel, and K. Stawiasz, Experimental Measurement of a Novel Power Gating Structure with Intermediate Power Saving Mode, *International Symposium on Low Power Electronics and Design*, pp.20-25, Aug. 2004.

[6] C. Long and L. He, Distributed Sleep Transistors Network for Power Reduction, *ACM/IEEE Design Automation Conference*, pp.181-186, July 2003.

[7] J. Kao, S. Narendra, and A. Chandrakasan, MTCMOS Hierarchical Sizing based on Mutual Exclusive Discharge Patterns, *ACM/IEEE Design Automation Conference*, pp.495-500, June 1997.

[8] C. Hwang, C. Kang, and M. Pedram, Gate Sizing and Replication to Minimize the Effects of Virtual Ground Parasitic Resistances in MTCMOS Designs, *IEEE/ACM International Symposium on Quality Electronic Design*, March 2006.

[9] M. Anis, S. Areibi, and M Elmasry, Design and Optimization of Multithreshold CMOS (MTCMOS) Circuits, *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pp.1324-1342, Volume 22, Issue 10, Oct. 2003.

[10] A. Ramalingam, A. Devgan, and D. Z. Pan, Wakeup Scheduling in MTCMOS Circuits Using Successive Relaxation to Minimize Ground Bounce, *JOURNAL OF LOW POWER ELECTRONICS*, pp.1-8, Vol.3, No.1, 2007.

[11] H. Jiang and M. Marek-Sadowska, Power Gating Scheduling for Power/Ground Noise Reduction, *ACM/IEEE Design Automation Conference*, pp.980-985, June 2008.

[12] A. Abdollahi, F. Fallah, and M. Pedram, A Robust Power Gating Structure and Power Mode Transition Strategy for MTCMOS Design, *IEEE Trans. on Very Large Scale Integration Systems*, pp.80-89, Volume 15, Issue 1, Jan. 2007.

[13] J.-J. Liou, A. Krstic, Y.-M. Jiang, and K.-T. Cheng, Path Selection and Pattern Generation for Dynamic Timing Analysis Considering Power Supply Noise Effects, *ACM/IEEE International Conference on Computer Aided Design*, pp. 493-496, 2000.

[14] A. Krstic, Y.-M. Jiang, and K. T. Cheng, Pattern Generation for Delay Testing and Dynamic Timing Analysis Considering Power-Supply Noise Effects, *IEEE Transactions on Computer-Aided Design*, Volumne 20, Issue 3, March, 2001.

[15] C. Tirumurti, S. Kundu, S. Sur-Kolay, and Y.-S. Chang, A Modeling Approach for Addressing Power Supply Switching Noise Related Failures of Integrated Circuits, *IEEE/ACM Design, Automation, and Test in Europe*, pp. 1078-1083, 2004.

[16] M. K. Butler and O. N. Mukherjee, Power-Aware DFT-Do We Really Need it?, *IEEE International Test Conference*, Panel, 2007.

[17] J. Ma, J. Lee, and M. Tehranipoor, Layout-Aware Pattern Generation for Maximizing Supply Noise Effects on Critical Paths, *IEEE VLSI Test Symposium*, pp. 221-226, 2009.

[18] X. Liu, Y. Zhang, F. Yuan, and Q. Xu, Layout-Aware Pseudo-Functional Testing for Critical Paths Considering Power Supply Noise Effects, *IEEE/ACM Design, Automation, and Test in Europe*, 2010.

[19] Taiwan Semiconductor Manufacturing Company, Ltd., TSMC Reference Flow 7.0, 2007.

[20] Cadence, "Encounder®User Guide," version 8.1.2, 2009.

[21] Synopsys, PrimeTime, version B-2008.12-SP3-2, 2009.

[22] Feng Lu, L.-C. Wang, K.-T. Cheng, J. Moondanos, and Z. Hanna, Power Gating Scheduling for Power/Ground Noise Reduction, *ACM/IEEE Design Automation Conference*, pp. 436-441, 2003.

[23] A. Krstic and K.-T. Cheng, Delay Fault Testing for VLSI Circuits, *Kluwer Academic Publishers*, Boston, MA, 1998.

[24] Synopsys, StarRCXT, version 2007.06.SP1-5.

[25] Synopsys, HSPICE, version C-2009.03-SP1.