TESTING TRI-STATE AND PASS TRANSISTOR CIRCUIT STRUCTURES

A Thesis

by

SHAISHAV PARIKH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2005

Major: Computer Engineering

TESTING TRI-STATE AND PASS TRANSISTOR CIRCUIT STRUCTURES

A Thesis

by

SHAISHAV PARIKH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Co-Chairs of Committee,	Duncan M. (Hank) Walker
	Jiang Hu
Committee Members,	Weiping Shi
	Jose Silva-Martinez
Head of Department,	Chanan Singh

August 2005

Major: Computer Engineering

ABSTRACT

Testing Tri-state and Pass Transistor

Circuit Structures. (August 2005)

Shaishav Parikh, B.E., L.D. College of Engineering

Co-Chairs of Advisory Committee: Dr. Duncan M. (Hank) Walker
Dr. Jiang Hu

Tri-state structures are used to implement multiplexers and buses because these structures are faster than AND/OR logic structures. But testing of tri-state structures has some issues associated with it. A stuck open control line of a tri-state gate will cause some lines in the circuit to float and take unknown values. A stuck-on control line can cause fighting when the two drivers connected to the same node drive different values. This thesis develops new gate level fault models and dynamic test patterns that take care of these problems. The models can be used with traditional stuck-at and transition fault automatic test pattern generation (ATPG) to ensure high fault coverage.

This research focuses on producing good test coverage with reduced effort for tri-state and pass transistor structures. We do circuit level modeling to help develop and validate gate level models, which can be used in production ATPG. We study the two primary effects of interest, capacitive coupling and leakage, and analyze the tri-state structures using these two effects. Coupling and leakage can cause a Z or X state to be seen as 0 or 1 in some cases. We develop parameterized models of behavior of common structures using these effects and some parameters such as number of fan-ins. We also develop gate level models of tri-state circuits that would replace the tri-state library cells

in the ATPG engine. This work develops a methodology to make tri-state and pass transistor circuit structures more usable in the industry.

DEDICATION

To my parents

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my graduate advisor, Dr. Hank Walker. I thank him for all his help, guidance and support throughout my research work at Texas A&M University. I would also like to thank my other Co-chair, Dr. Jiang Hu, and my committee members, Dr. Weiping Shi and Dr. Jose-Silva Martinez, for reviewing my research work. I would also like to thank the industrial liaison for this research, Grady Giles of AMD, in Austin, TX, who helped me throughout this work. Also thanks to Daniela Toneva and Joel Irby who work with Grady Giles for their input on this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Page

I. INTRODUCTION

Clock speed has always been the primary performance criteria in digital designs in the IC industry. Designs requiring high performance, such as microprocessors, continue to gain performance, using non-conventional circuits such as ratio, pre-charged or tri-state logic [1]. These circuits make conventional fault modeling and test generation techniques ineffective. So novel approaches to test such logic are needed to make the overall design reliable. Tri-state structures are mainly used in circuits to implement multiplexers and buses. Implementations using tri-state logic are much faster, smaller and lower power than those using AND/OR logic structures. Tri-state structures involve a third state of high impedance (Z state), along with the high and low states.

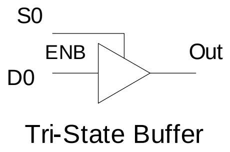


Figure 1. Tri-state buffer.

As shown in Figure 1, we can have a state of high impedance when the select line S0 is off, along with the high and low states under regular operation when the select line is on. In high impedance, the design acts as an open circuit, as if it has been disconnected from the rest of the much bigger implementation of which it might be a part.

The journal model is *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems.*

Pass transistor logic is similar to tri-state circuits in the sense that they also involve the third state of high impedance, along with the high and low states.

It is well known that tri-state logic designs have poor testability. Design for Test (DFT) techniques that have been developed for improving testability of tri-state designs have either suffered speed or hardware overheads [1, 2]. The two types of faults unique to with tri-state logic are floating and contention faults.

A. Floating fault

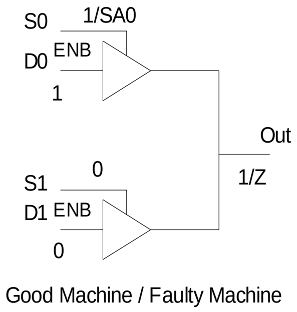


Figure 2. Floating fault.

In the case of the 2 × 1 multiplexer circuit shown in Figure 2, only one control line is on during functional operation, so that the input to that particular driver gets driven to the output node. In the case where the control line of the enabled driver gets stuck at zero (SA0), the multiplexer output will float (Z value) [1, 2, 3]. This floating line may store charge, which introduces a potential sequential mechanism that can invalidate test vectors that assume combinational behavior. In addition, floating values are easily influenced by noise and leakage, which can again invalidate the test setup. This situation is further complicated by the history mechanism of partially depleted silicon-on-insulator (SOI) technology, in which the transistor threshold depends on its prior activity [4, 5].

B. Contention fault

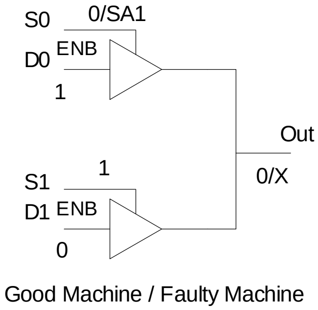


Figure 3. Contention fault.

As shown in Figure 3, a stuck-at 1 (SA1) fault on the control line of a tri-state gate can cause the output of the multiplexer to be at an intermediate value, due to two drivers, which are driving opposite values, both being ON at the same time. This is a well-known test generation problem, and the traditional solution is to add automatic test pattern generation (ATPG) constraints or hardware to ensure this clash does not occur in a fault-free circuit. This clash cannot be avoided in a defective circuit if a tri-state gate has a SA1 on its select line, producing an intermediate voltage (X value) on the output of the multiplexer or bus [1, 4, 5]. This intermediate voltage may be interpreted as a logic high or low value, depending on the logical threshold of the receivers and the design of the multiplexer. Existing ATPG tools simply ignore the floating case and treat the fighting case as a non-detect.

Furthermore, in order to implement the one-hot restriction (only one control line ON during functional operation) on the control lines of the multiplexer, sometimes designers implement a distributed decode function to enhance circuit speed [4]. The decode logic does not appear as a single, easily identifiable design block but is distributed over a larger portion of the design to share some of the common decode portions of the overall design. Due to this, a single SA0 or SA1 fault in the decode logic has been found equivalent to tens, or even hundreds of faults at the tri-state buffers.

C. Contributions of this research

Current ATPG tools do not handle X or Z states in tri-state and pass transistor circuits in a proper manner. This research will address the problem by first collecting data on circuit level models of these structures. This behavior will be used to generate pattern fault models encoding the necessary sensitization and propagation requirement to detect these faults. Since some ATPG tools do not support pattern faults, gate level fault models will be developed to ensure high fault coverage. The primary value of this work to the semiconductor industry is increased product quality, reduced engineering effort and increased product options. Existing ATPG tools often fail to produce good results on tri-state and pass transistor structures. This results in reduced test coverage or increased effort to improve coverage. So designers tend to avoid using these structures except in high performance applications. This research will develop a general methodology to reduce the effort of all companies and make such structures more usable. The remainder of this thesis is organized as follows. Section II describes all the previous work done on tri-state testing problems and solutions proposed to resolve to them. Section III describes the methodology used in this research to collect circuit level data on example tri-state and pass transistor circuits and the simulation results associated with it. Section IV introduces the idea used in this research to resolve the problem of testing tri-state designs. Section V describes the pattern faults and gate level fault models developed to improve test coverage of tri-state structures. Section VI concludes the thesis and introduces some future plans for this research.

II. PREVIOUS WORK

Digital designers have always tried to ignore tri-state structures wherever possible due to the testability issues associated with them. As such, the amount of prior research done in this field is limited. Today tri-state structures find extensive use only in high performance circuit designs and complex macros. Previous work attempted to improve the test coverage of designs using these structures has been either trying to study the detectability of the possible-detect faults (floating and contention faults) using a probabilistic model [4] or trying to improve testability by using non-conventional fault modeling and test generation techniques [1]. An approach based on a consistently dominant fault model has also been proposed in one of the earlier works [5]. The probabilistic model mentioned above is developed to analyze the SA1 possible-detect faults. The study assumes one-hot restriction on the control lines so that one and only one driver can be ON at a time in the multiplexer design. It tries to understand the impact of SA1 faults on controls lines when they are left unattended. In other words, it attempts to study the detectability of control line SA1 faults obtained from a set of patterns not specifically targeting these faults. The work concludes that the probability of a SA1 fault on control line being detected by a set of N patterns can be expressed as:

$$P_{det} = \{1 – (1 – P_{md} * P_c * P_p * P_e)^N \qquad (1)$$

where,

P_{det} : Probability that the possible-detect fault is completely detected.

P_{md} : Probability of a multi-drive. Probability of two drivers driving the output in case of the fault.

P_c : Probability of multi-drive resulting in contention. Probability of corresponding data lines being driven by opposite values.

P_p : Probability of contention detection at an observable point. Probability of contention propagating to an observable point, such as a primary output or a scan flip-flop.

P_e : Probability of contention resolving to an error value. Probability of X resolving to an error value on real silicon.

N : Total number of test patterns applied.

Using this equation it was shown that the probability of a control line SA1 fault being detected using a general ATPG test pattern set that does not target these faults is more than expected, for a reasonable pattern set size for the designs they tested. But these results also showed that the coverage was far below satisfactory test coverage standards. This suggests development of a model, which targets such faults directly and assures improvement of test coverage.

One of the earlier efforts on tri-state testing exploited circuit particularities of CMOS designs, used automatic learning of useful relations about nodes in the design, and innovative test vector generation [1] to improve testability. This research developed techniques for test generation, which exploits circuit design properties for resolving Z states to a binary value. It introduces the concept of pull-up devices, where by attaching a PFET on the output of a multiplexer would pull Z values to logic high. This idea is shown in Figure 4.

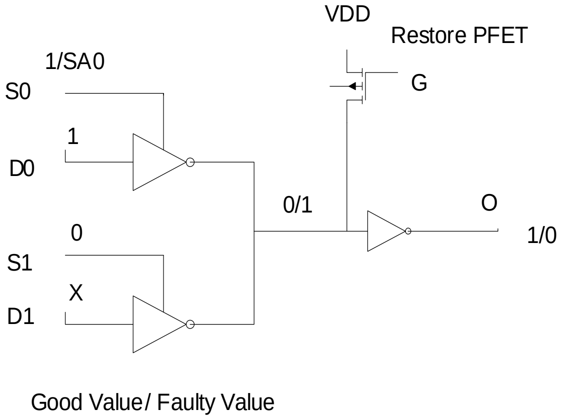


Figure 4. PFET attached at output of 2×1 multiplexer.

The gate G of the restore PFET is controlled such that a Z is resolved to a 1 on the tri-state output. This can then be exploited by ATPG to test SA0 faults on control lines which are otherwise un-testable. The efficient way to control G is to connect G to the output O of the inverter thereby creating a bus-keeper. This device can latch a 1 input but not a 0 input. The bus-keeper results in the output holding the previous cycle value in case of floating faults. Using this design, a sequence of patterns can be created where the restore PFET in ON and the subsequent test of the fault causes a change of state. This sequence of patterns can be applied as functional patterns or can be applied using scan chains. So this is similar to stuck-open or transition-fault testing, but used actually to test for stuck-at faults.

The consistently dominant fault (CDF) model was developed in one of the earlier research works [5]. This model assumes that when a floating or contention fault occurs, the gates whose inputs come from the tri-state output with the fault will interpret the value to be either a logical 1 or 0. The model assumed the output resolved to 1 or 0 in case of floating and contention faults separately, and generated vectors accordingly. The results of the research showed that the probability of fault detection in this way was high. So once a test pattern with output assumed to resolve to 1 (from Z or X) were developed and then a test pattern with output assumed to resolve to 0 were added. This whole set of test vectors guaranteed detection of faults whenever the output resolved to some logic value on real silicon. This CDF fault model is basically an extension of the stuck-at fault model and has good results for testing of tri-state circuits according to this work.

This idea of the output resolving to either 1 or 0 in case of floating or contention faults was adopted by Daniela Toneva, of the DFT group at Advanced Micro Devices, Incorporated, Austin, TX. She developed models using this idea as shown in Figure 5 [6].

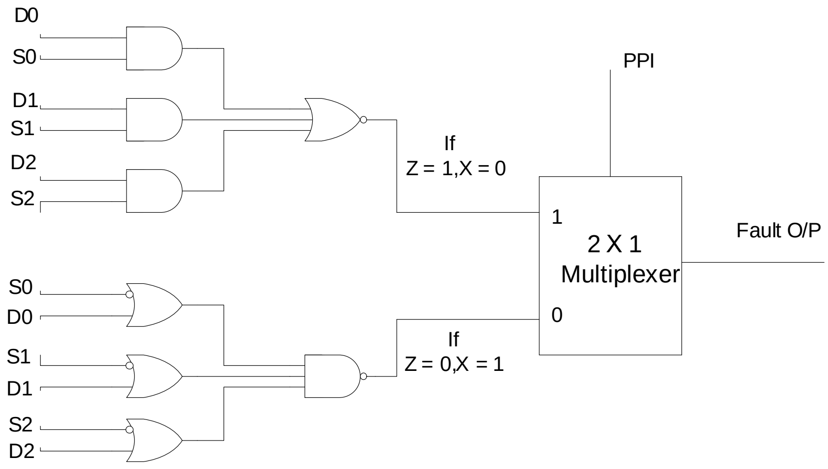


Figure 5. Toneva model.

This model in Figure 5 is implemented to represent a 3 × 1 inverting multiplexer design. The top input line to the 2 × 1 multiplexer represents logic for the assumption that Z at the output of the multiplexer, resolves to 1, and X at the output of the multiplexer resolving to 0 for a design. The bottom input line represents logic for the assumption of Z at the output resolving to 0, and X at the output resolving to 1 for a design. The control line of the 2 × 1 multiplexer in the above model is a pseudo primary input, so that it is directly controllable by the ATPG. This model is incorporated into the ATPG engine to produce the set of vectors for testing of stuck-at faults on the control lines. The ATPG tool tests for a stuck-at fault at the output of this 2 × 1 multiplexer in the model. The primary shortcoming of this model is that it does not force the ATPG to set any particular input to a value for testing of a specified fault, and so the ATPG might not produce the correct set of vectors when testing for the specified fault. This idea will be clear after we present our models developed in this research in the following sections.

One of the earlier works describes an algorithmic test pattern generation method named ZALG* to test circuits involving tri-state logic [7]. ZALG* takes bus clash (contention fault) and memory retention (floating fault) into consideration and uses a multiple path sensitization method to generate the test pattern generation algorithm.

In one of the earlier works, a built-in-self-test (BIST) circuitry to test for SoC designs involving tri-state buffers and buses [8] is developed. This BIST block configuration is not specified by any SoC structure, so it is suitable for a general/reusable testable IP.

III. METHODOLOGY AND CIRCUIT SIMULATION RESULTS

Circuit level modeling of the microprocessor circuits provided to us by AMD was carried out using AMD SOI CMOS proprietary models and UC Berkeley BSIM4 bulk CMOS device models [9] incorporated into the Cadence Spectre circuit simulator. The two microprocessor circuits provided by AMD are discussed in the following sections.

A. Tri-state driver design

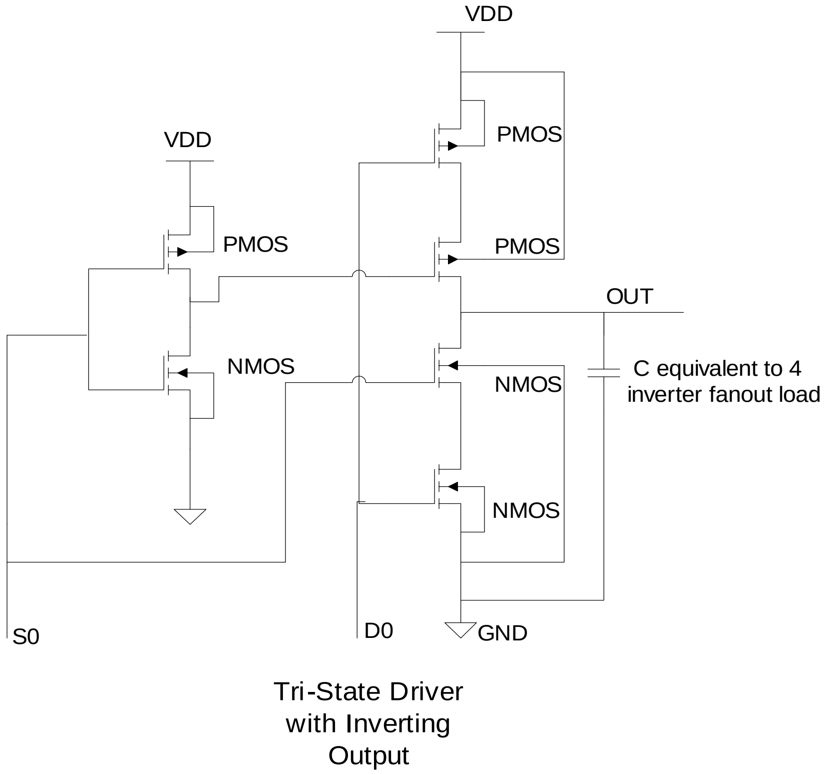


Figure 6. Tri-state driver design.

The circuit shown in Figure 6 is a single tri-state driver with an inverting output. We can have several such drivers connected in parallel to create the multiplexer design. In such a design one can enforce the one-hot restriction on the control lines by providing decode logic to ensure one and only one control line is ON at a time during functional operation. We injected SA0 and SA1 faults on the control lines (S0 in Figure 6) and observed the behavior of the output in terms of voltage levels and whether the output state (Z or X) resolved to a Boolean logic value in some reasonable amount of time. We used a 4 inverter fan-out load to represent the actual downstream logic which this multiplexer design might be a part of, when used in real microprocessor circuitry. The results obtained for this multiplexer design are as follows.

1. 5 × 1 multiplexer design results for SOI model

A 5 × 1 multiplexer was analyzed using the AMD SOI models for 65nm technology.

- Output was set high through good driver in one clock cycle and then set to floating in the next clock cycle due to a single SA0 fault on the chosen select line. Output simulated for a time period of 0-1 µs. The results are shown in Table I.

TABLE I. SIMULATION RESULTS FOR FIRST DESIGN WHEN PREVIOUS CYCLE OUTPUT IS 1

No. of data inputs high	Output value for 0-1 μs time period
0	High
1	High
2	High
3	High up to 20 ns and intermediate after that
4	High up to 13 ns and intermediate after that
5	High up to 8 ns and becomes low after 86 ns

The left hand column gives the number of data inputs that are high in a given vector. So the first entry (0) means that a given vector makes all the data inputs low while the output is floating (all control lines off). Since this is an inverting multiplexer, a zero on a multiplexer input will increase the leakage paths to Vdd, while a one on an input will increase the leakage paths to ground. We can see from the output results that the output remains high regardless of the time period when no more than two data inputs are high. For cases of three and four data inputs high the output remains high for a few nanoseconds (ns) and then assumes an unknown value. For the case of all five data inputs high, the output remains high for even less time and does assume a low value after a while. These results indicate that the floating faults are strongly input data dependent and also previous cycle dependent. Note that the output was set high in the previous cycle in this case. Also note that the output value is sampled less than 1 ns after the test is applied.

- Output was set low through good driver in one clock cycle and then set to floating in the next clock cycle due to a single SA0 fault on the chosen select line. Output simulated for a time period of 0-1 μs.

TABLE II. SIMULATION RESULTS FOR FIRST DESIGN WHEN PREVIOUS CYCLE OUTPUT IS 0

No. of data inputs high	Output value for 0-1 μs time period
0	Low up to 4 ns and becomes high after 34 ns
1	Low up to 6 ns and intermediate after that
2	Low up to 8 ns and intermediate after that
3	Low up to 12.5 ns and intermediate after that
4	Low up to 35 ns and intermediate after that
5	Low

The results shown in Table II again justify the observation that the output values are strongly input data and previous cycle dependent in the floating fault case. We have set the previous cycle output low, which is why the output remains low for a certain time period, depending on the input data for most of the cases. It only stays low for the case where we have all the data inputs driving the output low maximizing the leakage to ground.

- Output obtained for the contention fault case when we inject a SA1 fault on one of the tri-state drivers was an intermediate value showing a little bit of bias towards the logic low, irrespective of the previous cycle output value. The output was input data independent (on the deactivated lines), which was expected because the ON currents of the activated lines are much higher than the gate and sub-threshold leakages of the deactivated lines.

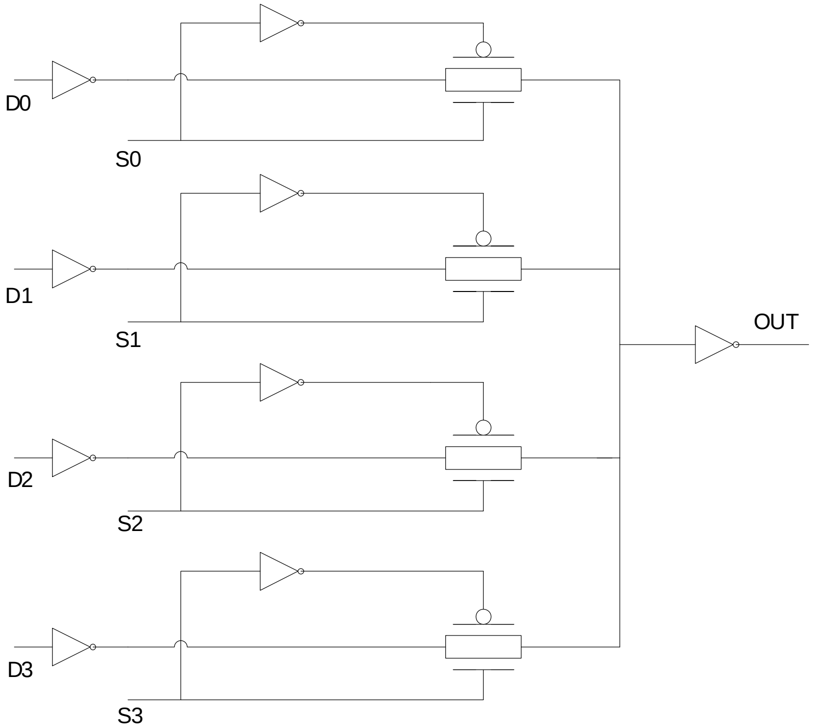
2. 5 × 1 multiplexer design results for bulk CMOS model

A 5 × 1 multiplexer was analyzed using the bulk CMOS models for 90nm technology.

- Output was set high through good driver in one clock cycle and then set to floating in the next clock cycle due to a single SA0 fault injected on the select line of the activated driver. Simulated for a time period of 0-1 μs. Output remains high for a few tens of ns and goes to 0 after 116 ns. This result was independent of the input data values. This is very different from the SOI behavior, where the output was strongly affected by the data inputs.
- Output was set low through good driver in one clock cycle and then set to floating in the next clock cycle due to a single SA0 fault injected on the select line of the activated driver. Simulated for a time period of 0-1 μs. Output remains low all the way up to 1 μs. Again this result was independent of the input data values.
- Output obtained for the contention fault case when we inject a SA1 fault on one of the tri-state drivers was the X at the output which resolved to 0 fast enough to be

detected (less than 70 picoseconds). This is really positive from the ATPG perspective as no extra logic is needed to resolve the output to a Boolean value. This is in contrast to SOI, where the output is an intermediate voltage without the addition of any special test logic.

B. Pass transistor design



4 × 1 Pass Transistor Multiplexer

Figure 7. Pass transistor design.

The design in Figure 7 is a 4 × 1 multiplexer with pass transistors making up transmission gates which are used as individual drivers for the multiplexer. The inverters at the inputs and output of the multiplexer represent the upstream and downstream logic, which this circuit might be a part of, when used in real microprocessor circuitry. The one-hot restriction is ensured on this design through decode logic. We injected SA0 and SA1 faults on the control lines (S0-S3) of this design and observed the behavior of the output in terms of voltage levels and whether the output state (Z or X) resolved to a

Boolean logic value in a reasonable amount of time. The results obtained on this design are as follows.

1. 4×1 multiplexer design results for SOI model

A 4 × 1 multiplexer was analyzed using the AMD SOI models for 65nm technology.

- Output was set high through good driver in one clock cycle and then set to floating in the next clock cycle due to a single SA0 fault injected on the select line of the chosen driver. Output simulated for a time period of 0-1us. The results are as shown in Table III.

TABLE III. SIMULATION RESULTS FOR 2nd DESIGN WHEN PREVIOUS CYCLE OUTPUT IS 1

No. of data inputs high	Output value for 0-1 µs time period
0	High up to 12 ns and becomes low after 20 ns
1	High up to 20 ns and becomes low after 44 ns
2	High up to 78 ns and intermediate after that
3	High
4	High

In this design the output is strongly input data and previous cycle dependent in the floating case similar to the previous design. But as we can see from the results the output dependence on data input values is much more in this design than the previous one. The output also fluctuates a lot more over time for different data inputs in this design.

- Output was set low through a good driver in one clock cycle and then set to floating in the next clock cycle due to a single SA0 fault injected on the select of the chosen driver. Output simulated for a time period of 0-1 μs.

TABLE IV. SIMULATION RESULTS FOR 2nd DESIGN WHEN PREVIOUS CYCLE OUTPUT IS 0

No. of data inputs high	Output value for 0-1 μs time period
0	Low
1	Low
2	Low
3	Low up to 23 ns and becomes high after 89 ns
4	Low up to 14 ns and becomes high after 25 ns

Again we see in Table IV that the results fluctuate for different input data values. We also see that there is a small amount of bias towards the output going low, if we consider both the cases where the previous cycle output was set high as well as low.

- The output obtained for the contention fault case when we inject a SA1 fault on one of the tri-state drivers was logic state high, irrespective of previous cycle output values. The output was independent of data inputs on deselected lines, which was expected. This result is really positive from the ATPG perspective, as in the following section.

2. 4×1 multiplexer design results for bulk CMOS model

- The output was set high through good driver in one clock cycle and then set to floating in the next clock cycle due to a single SA0 fault injected on the select line of the chosen driver. Simulated for a time period of 0-1 μs. The output remains 1 all the way up to 1 μs. The output is input data independent similar to the previous design for the Bulk CMOS model.
- The output was set low through a good driver in one clock cycle and then set to floating in the next clock cycle due to a single SA0 fault injected on the select line of the chosen driver. Simulated for a time period of 0-1 μs. Output remains low for a few ns and then goes to 1 after 33 ns. Again the output dependence on the input data values is negligible.
- The output obtained for the contention fault case, when we inject a SA1 fault on the select line of the chosen tri-state driver, was the X at output, which resolved to 0 fast enough to be detected (less than 70 picoseconds).

IV. ATPG APPROACH

The circuit level simulation results obtained for both the tri-state driver and pass transistor multiplexer circuits for both the SOI and bulk CMOS processes, suggested taking a novel approach to maximize the testability of these two circuit structures. The results for the floating fault case for both designs and processes suggested a transition-like fault model to improve the testability of the circuit. The simulation results showed that the output state depended on its previous cycle state and that it retained that state for some time, independent of the data input values. The data input values affected the time and value to which the output floated, but overall the output always started from the value of the previous cycle. This fact can be exploited and one can set the output to a logic value in one cycle and then drive it to the opposite value in the next cycle. Then if we test the output fast enough, the output will hold the previous cycle value in the case of a floating fault on its control line, which will result in detection of the fault. This idea will be clearer in the following sections.

A. ATPG approach for tri-state driver design

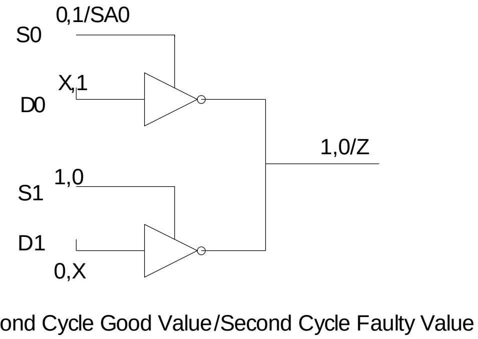

S0
0,1/SA0
D0
X,1
1,0/Z
S1
1,0
D1
0,X
First Cycle, Second Cycle Good Value/Second Cycle Faulty Value

Figure 8. ATPG approach for tri-state driver design.

Figure 8 is the gate level model for the floating case for tri-state driver design multiplexer. As shown in the figure, we can set the output to a logic value (1 in this case) through the good tri-state driver and then try and set the output to the opposite logic value (0 in the above case) through the tri-state driver whose control line is under test for a SA0. If the control line is SA0, then the good value for the output will be different from the faulty value, if we test the output quickly enough. Here as shown, the good value is 0 for the second cycle, but the SA0 on the control line of the first tri-state driver makes the output floating. This floating state will hold the value from the previous clock cycle (1) for some time and so we can detect the fault. As shown from the previous simulation results (Table 2) the shortest amount of time for which the output held its previous cycle value was 4 ns for the SOI process, for the case where we had set the output to 0 in the previous clock cycle and all the inputs were driving the output to 1 in the next clock cycle. So if we can test the output within 4 ns, the transition-like testing model suggested above would work. Note that the model used above is different from the regular transition fault model in the sense that we set the output to a logic value through a good driver in the previous cycle and then test through the faulty driver rather than using the same driver as in the transition fault model. The criterion of testing within 4 ns is not very stringent if we consider the testing methods used in industry today. The most common mode of testing used today is scan-based testing. The launch-on-capture mode of scan-based testing works at close to mission mode speed. The average clock speed of microprocessors today is around 3 GHz, which makes the launch-on-capture mode test around 300 ps. This is well within the 4 ns constraint.

The results obtained for the contention case for the tri-state driver circuit for the SOI process are more challenging, because the output does not resolve to any logic value within the simulated amount of time. The output did show some bias towards the low logic state but not significant enough to draw any conclusions. But this small amount of bias can be exploited to pull the output to a low logic state with some design modifications. We can use a pull-down transistor attached to the output of the multiplexer to pull the X state to an actual 0 as shown in Figure 9.

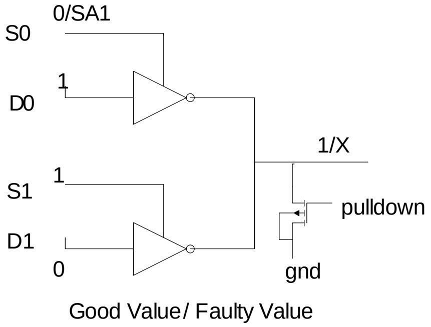
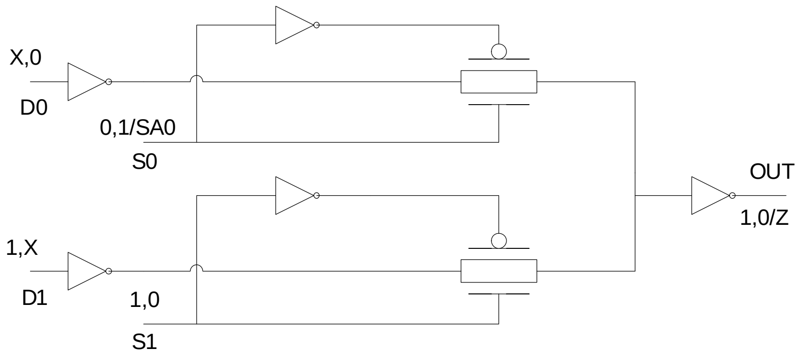


Figure 9. NFET pull-down attached at output.

As shown in the figure, we can attach a weak NFET transistor to the output of the multiplexer [1]. This transistor can be turned ON only during test mode to weakly drive a 0 on the output. The weak 0 can replace the X state on the output and is weak enough to be overdriven by a 1 value on the output line. The pull-down structure is typically much smaller (5-8 times) than the rest of the circuitry and so is almost always absorbed within the overall area of the design and adds negligible delay. It does require a global test signal, which is turned ON only during the test mode of operation, which is the only overhead that one incurs in terms of routing costs. Variations of the above scheme like pull-ups, pull-ups/pull-downs pairs and bus-keepers are some others ideas which can be employed depending on the circuit properties [4]. As for the Bulk CMOS process, no extra logic is needed in the design as the output resolves to 0 by itself.

B. ATPG approach for pass transistor design



First Cycle, Second Cycle Good Value/Second Cycle Faulty Value

Figure 10. ATPG approach for pass transistor design.

The approach for the floating fault case for the pass transistor multiplexer design is the same as for the tri-state driver design. As shown in Figure 10, we can set the output to a logic state through the good transmission gate in one clock cycle and then try and set it to the opposite value in the next clock cycle through the driver under test. If we have a SA0 fault on the driver control line, we will get a Z state on the output, which will hold the previous cycle value for some time. So again if we test the output fast enough we would detect the fault. As shown from the previous simulation results (Table 3) the shortest amount of time for which the output held the previous cycle value is 12 ns for the SOI process, for the case when the previous cycle output value was 1 and all the data inputs drive the output to 0 in the next clock cycle. This timing constraint is even less stringent than the previous design and can be easily achieved.

The results obtained for the contention case for this design were very positive because the output during contention resolved to logic state high in less than 100 ps for the SOI process. This result can be used to test the design for SA1 faults on the control lines by assuming the contention state to be equivalent to high logic state. This idea can be explained by the use of an example shown in Figure 11.

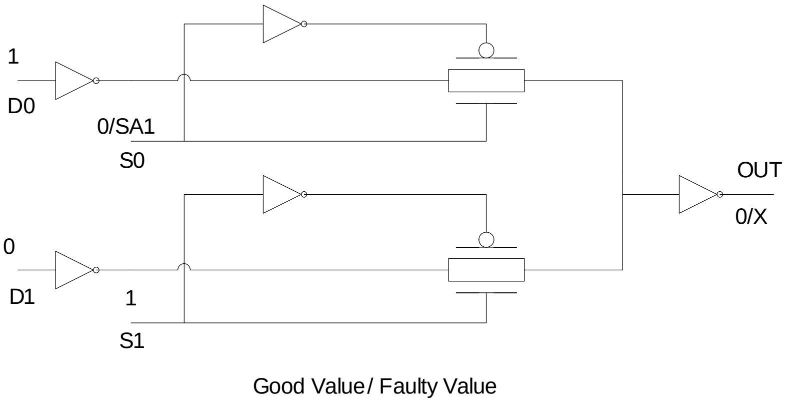


Figure 11. Input vector to test for contention in SOI.

As shown in the figure, the input vector applied on the 2 × 1 multiplexer data inputs is 10 whereas the control inputs are 01. In case of a SA1 fault on the control line of the first driver, a contention fault will occur, causing the output to assume an intermediate state X in place of the good value 0. But as seen from the simulation results, this X state resolves quickly to a high logic state. So if we test the output in this case, we will see a 1 on the output and the contention fault would be detected. In the Bulk CMOS process, the output resolves to 0 in a reasonable amount of time. So we can treat the X at the output of this design to be equivalent to a 0.

V. FAULT MODELS AND PATTERN FAULTS

This section elaborates on the ATPG models for the floating and contention cases in the multiplexer circuits. The floating case is tested by setting the output to a logic value, through a fault free tri-state driver (or pass transistor) in one clock cycle and then driving it to the opposite value in the next cycle through the driver under test. The output retains the previous cycle value when the driver under test has a SA0 on its control line, so the fault would be detected. This section describes the ATPG fault model required to ensure all the restrictions needed on the inputs and control lines to implement this test sequence.

The transition-like fault model introduced above sets the output to a known logic value through the fault-free tri-state driver. This driver can be chosen at random by the ATPG fault engine. So for the design in Figure 6, if we test for SA0 on S0, we can set the output in the first cycle through any of the other four drivers. We assume that the decode logic for the control lines ensures that one and only one of the control lines is set high at a time. So when the ATPG model enforces a control line to go high, it is assumed that the decode logic takes care of the fact that the other control lines should go low. The ATPG model suggested here is a slow-to-fall (STF) transition-like fault model. So when the model enforces a particular control line to go high on the first cycle, it also has to enforce the corresponding data input to go low for the tri-state design (Figure 6) and go high for the pass transistor design (Figure 7) to make the output go high in the first cycle. Then it enforces the output to go low in the second cycle through the tri-state driver under test. For that it ensures that the particular control line is set high and the data input is set high for the tri-state design and set low for the pass transistor design to drive the output low in the second clock cycle. So this model will detect the SA0 fault on that control line whenever the output does not fall (go low).

The idea described above can be easily specified using dynamic pattern faults [10]. However few ATPG tools support dynamic pattern faults. An exception is Cadence Encounter Test. Mentor Graphics Fast Scan and Synopsys Tetramax do not support static or dynamic pattern faults. For such tools we propose to develop a gate level fault model to be incorporated into the ATPG engine as described in the next subsection.

The test can be described using 2 dynamic patterns whose Boolean logic description is given below. We develop the dynamic pattern implementation logic similar to STF transition faults. The patterns shown are for testing of a SA0 fault on control line S0. We assume the tri-state drivers implement a 5×1 inverting multiplexer design.

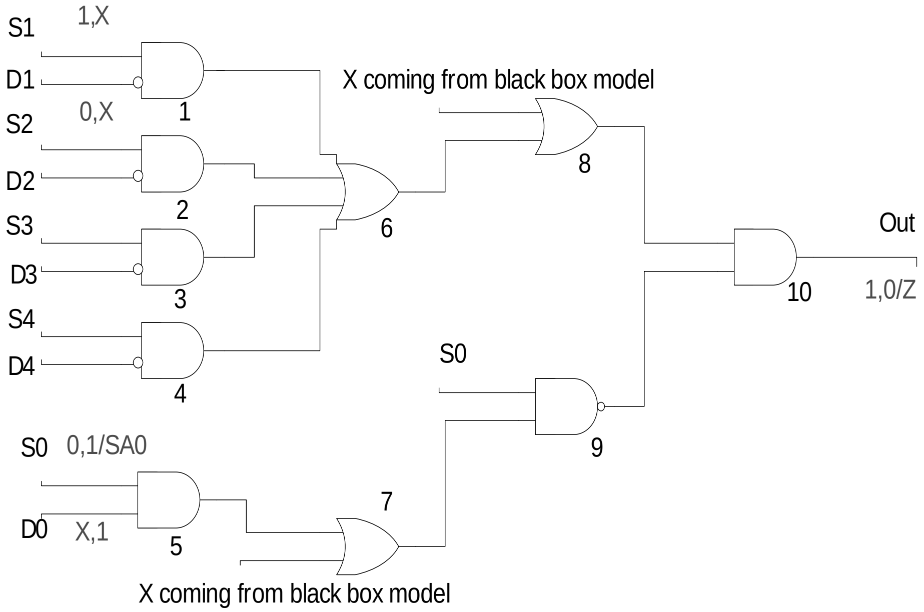
A. Dynamic patterns

- First Pattern: (S0′) & ((S1 & D1′) | (S2 & D2′) | (S3 & D3′) | (S4 & D4′))
- Second Pattern: (S0 & D0)

The first pattern sets S0 low and sets one of the remaining select lines S1-S4 high, and the corresponding data line low. This sets the output high for first clock cycle. The second pattern sets S0 and D0 high, attempting to set the output low. If S0 is SA0, a STF fault will occur at the output. Similar patterns can be developed for the other select lines and the pass transistor multiplexer.

B. ATPG model for S0, SA0

The model shown in Figure 12 is a slow-to-fall transition-like fault model for testing of control line S0 SA0.



First Cycle, Second Cycle Good Value/Second Cycle Faulty Value

STF Transition-like ATPG fault model for S0,SA0 for Inverting Multiplexer

Figure 12. ATPG model for S0, SA0.

The model, when incorporated into the ATPG engine will ensure that the correct set of input patterns are generated by ATPG for testing of SA0 on S0. The model assumes that the tri-state drivers implement an inverting multiplexer where the output is the inverse of the data input driving it. The fault to test for in the ATPG tool is a STF transition fault on the output of the model (Out). The requirements for the logic in terms of enforcing values on the inputs of the model above would be to make S0 go low while any one of the other tri-state drivers drive the output high in first clock cycle. The logic should also force S0 and D0 high in second clock cycle to try and drive the output low in the second cycle. The model assumes the decode logic for one-hot restriction on the control lines is available. Development of this one-hot logic restriction is trivial in any case. So when we drive the control line S0 high, we assume the decode logic resolves the issue of driving the other control lines low.

To ensure the requirement of driving the output high in first clock cycle, the inputs to the AND (10) gate at the output in the model should both be high. To make the bottom input high, one of the inputs to the NAND (9) gate must be low. So either S0 must be low or the other input to the NAND (9) gate must be low. As seen from the figure, the X input at the OR (7) gate blocks a 0 at its output, so driving the other input of the NAND (9) gate to 0 is not possible. This enforces the ATPG to make S0 go low in first cycle. This X input coming into the OR (7) gate can easily be modeled by assuming a black box model feeding the input cone of that line. Now to make the other input of the AND (10) gate at the output high, we need to drive the other input of the OR (8) gate feeding it high. This requires any one of the 4 AND (1,2,3,4) gates at the inputs to be high. So one of the tri-state drivers associated with S1-S4 must be driving the output high to ensure the AND (10) gate is high. This restriction enforced by the logic ensures the requirement of our model for the first cycle.

The requirement of the model for second cycle is S0 and D0 should be high while trying to drive the output to 0. So to try to make the output go low in the second cycle, one of the inputs to the AND (10) gate must be 0. The top input cannot be 0 because that is blocked by the X feeding the OR (8) gate at the top input. So this forces the bottom input to go low. Now to ensure a 0 at the output of the NAND (9) gate, both the inputs must be 1. So the top input of the OR (7) gate feeding it, is forced to be high because of the other input being an X. This forces both the inputs S0 and D0 of the AND (5) gate feeding it to be high. So the requirement of the model for the second cycle also gets satisfied. So this model ensures that the proper pair of input vectors gets generated to test for the SA0 fault on S0.

C. *ATPG model for 5 × 1 multiplexer (floating case)*

Figure 13 is the ATPG fault model for testing of SA0 fault on all of the 5 control lines S0-S4, of the multiplexer.

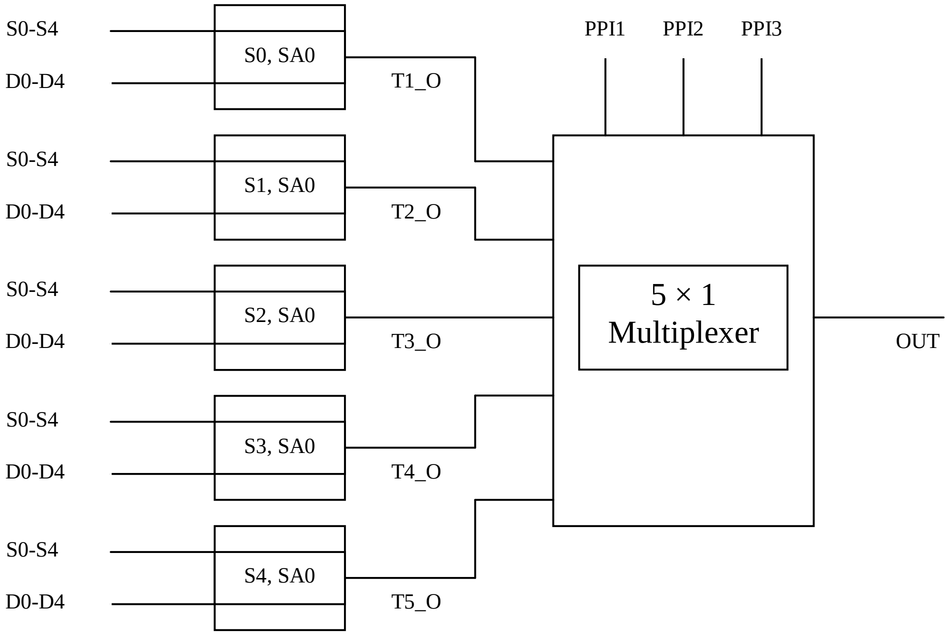


Figure 13. ATPG model for floating case for 5 × 1 inverting multiplexer.

This model will ensure that the ATPG will generate the correct set of input vectors for testing of all 5 SA0 faults on the control lines. The block S0, SA0 represents the entire logic shown in Figure 12 for testing of S0 SA0. Similarly the other four blocks shown represent logic for testing of the individual control lines SA0. This logic is similar to Figure 12 with the individual inputs to that tri-state driver swapped with the input lines S0 and D0 in Figure 12. So the model for S1, SA0 would be similar to Figure 12 with the input lines S1, D1 swapped with S0, D0 everywhere in the model. These 5 blocks form the inputs of a 5 × 1 multiplexer in this model, whose control lines are pseudo primary inputs so that they are directly controllable by the ATPG engine. So by testing for STF transition faults on the inputs to this multiplexer T1_O - T5_O, the ATPG would be able to generate the correct set of patterns to detect the SA0 faults on the individual control lines of the tri-state drivers.

As for the contention case, we propose a stuck-at fault model to be incorporated into the ATPG tool for testing of SA1 faults on the control lines of the tri-state drivers. For the tri-state design for the SOI process, the output resolved to a low logic state in the contention case using a transistor pulldown as shown in Figure 9, whereas the output resolved to a 0 by itself for the Bulk CMOS process. So an input vector, which drove the output to 1 through the good driver and to 0 through the driver under test, will detect the SA1 on the control line of this driver for both the processes. For the pass transistor design, the output resolved to a high logic state in the contention case for the SOI process, whereas it resolved to a 0 for the Bulk CMOS process. So an input vector, which drove the output to 0 through the good driver and 1 through the driver under test, will detect the SA0 on the control line of this driver for the SOI process technology. For the Bulk CMOS process, the input vector requirements will be the same as for the first design. This idea is further explained along with the developed models in the next section.

D. ATPG model for S0, SA1 when X resolves to 1

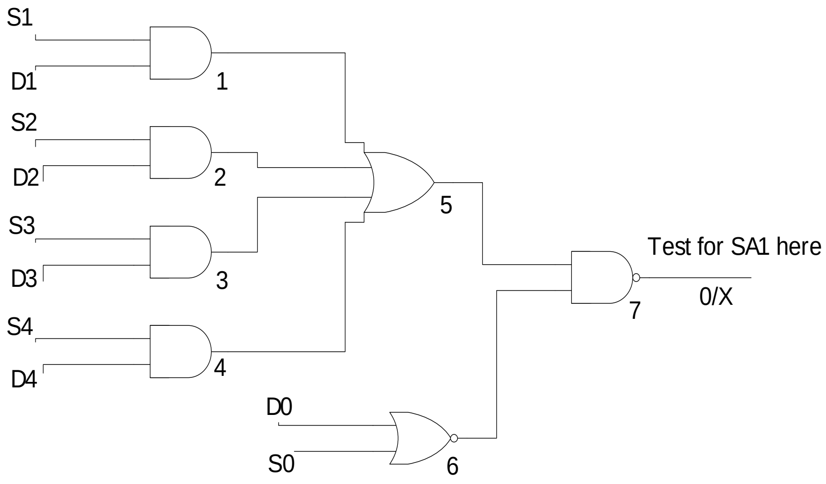
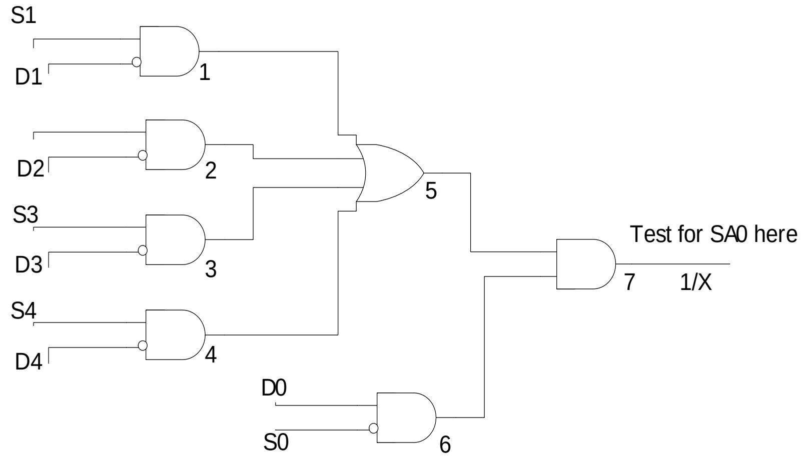


Figure 14. ATPG model for S0, SA1 when X resolves to 1.

The model shown in Figure 14 is for testing S0 SA1. This model, when incorporated into the ATPG engine, will ensure that the correct pattern will be generated to test for S0 SA1. It is for designs where X resolves to a 1 at the output. That is true for the tri-state design in the SOI process. So trying to drive the output to 0 when testing for contention would be the right approach for designs like these. In other words, we need to test for a SA1 fault on the output of this model. The model assumes that the tri-state drivers implement an inverting multiplexer where the output is the inverse of the data input driving it. So the requirements in terms of values on the input lines of this model would be S0 and D0 should both be 0 while either one of the remaining tri-state drivers should be driving the output to 0 (select line is 1, data input is 1). Now to satisfy this requirement of trying to drive the output of this model to 0, we need both the inputs of the NAND (7) gate to be 1. This forces both S0 and D0 to be 0, through NOR (6) gate. For the top input of the NAND (7) gate to be 1, we need either one of the control line and data input combination at the inputs of the 4 AND (1,2,3,4) gates to be high. So this logic enforces the requirements of the model. So by using this logic in place of the multiplexer model, we will ensure, that the correct test vectors would be generated.

E. ATPG model for S0, SA1 when X resolves to 0



SA0 ATPG fault model for S0 SA1.
If X resolves to a 0.

Figure 15. ATPG model for S0, SA1 when X resolves to 0.

The model shown in Figure 15 is for testing S0 SA1, but for designs where the X at the output resolves to a 0. This is true for the tri-state design for both processes, as well as for the pass transistor design for the Bulk CMOS process. So trying to drive the output to 1 when testing for contention would be the right approach for designs like these. In other words, we need to test for SA0 fault on the output of this model. . The model assumes that the tri-state drivers implement an inverting multiplexer where the output is the inverse of the data input driving it. So the requirements in terms of values on the input lines of this model would be to make S0 low and D0 high, while either one of the remaining tri-state drivers try to drive the output high (select line is 1, data input is 0). Now to satisfy the requirements of the output being high, we need both the inputs of the AND (7) gate at output to be high. So this forces the ATPG to make S0 low and D0 high automatically through the AND (6) gate they feed to. To make the other input of the AND (7) gate high, we need to make one of the AND (1,2,3,4) gates feeding the OR (5) gate high. So this satisfies the requirements of making either one of the remaining tri-state drivers drive the output high. So for instance by making S0 high and D0 low, the ATPG would be able to satisfy the requirement of making the top input of the AND (7) gate at the output of the model high. So by feeding the requirement of testing for SA0 at output of this model into the ATPG tool we would be able to detect SA1 fault on S0.

F. ATPG model for 5 × 1 multiplexer (contention case)

Figure 16 is the ATPG fault model for testing of SA1 fault on all of the 5 control lines S0-S4, of the multiplexer.

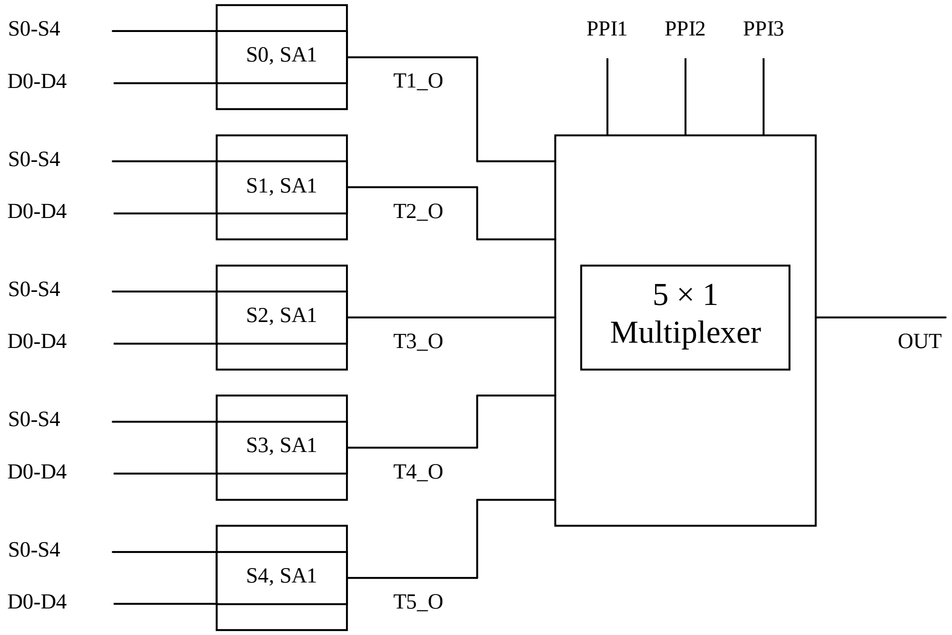


Figure 16. ATPG model for contention case for 5 × 1 inverting multiplexer.

This model, when incorporated into the ATPG engine, will ensure that the correct set of vectors are generated to test for SA1 faults on all the control lines of the tri-state drivers. The block S0, SA1 represents the entire logic shown in Figure 14 or Figure 15, depending on the multiplexer design being tested for (whether X resolves to 0 or 1) for testing of S0 SA1. Similarly the other four blocks shown represent logic for testing of the individual control lines SA1. This logic is similar to Figure 14 or Figure 15 with the individual inputs to that tri-state driver swapped with the input lines S0 and D0 in the respective figure. So the model for S1, SA1 would be similar to the Figure 14 or Figure 15 with the input lines S1, D1 swapped with S0, D0 everywhere in the model. These 5 blocks form the inputs of a 5 × 1 multiplexer, whose control lines are pseudo primary inputs so that they are directly controllable by the ATPG fault engine. So testing for SA1 or SA0 fault, depending on the previous model adopted, on the inputs to these multiplexer T1_O – T5_O, would detect the SA1 faults on the individual control lines of the tri-state drivers.

VI. CONCLUSIONS AND FUTURE WORK

This research focuses on improving the testability of digital circuits that contain tri-state structures. The ideas presented in this thesis are design specific and technology dependent, but can easily be applied to more general circuit structures. Circuit simulation was used to analyze the detectability of commonly used tri-state structures in high performance digital circuits. Using these simulation results, pattern faults and gate level fault models were developed which can be inserted into ATPG to improve test coverage of tri-state and pass transistor structures.

As part of the future work, the gate level fault models and the pattern faults developed in this research will be tested at Advanced Micro Devices, Inc. to observe their impact on test coverage, test pattern count and ATPG time.

We would like to consider the SOI history effect [4, 5] into our analysis, whereby we will account for the voltage threshold variability of transistors over multiple cycles in the case of partially-depleted SOI process designs. The simulations will be run over multiple cycles and the output will be observed for different input data combinations over those cycles in both functional and test modes. A key challenge in this work is ensuring the history effect in test mode that is similar to test mode.

This work, including the future plans, will enable widespread use of tri-state structures leading to much better performance digital designs.

REFERENCES

[1] P. Wohl, J. Waicukauski, and M. Graf, “Testing untestable faults in three-state circuits”, in *Proc. VLSI Test Symposium*, Apr.-May 1996, pp. 324-331.

[2] R. Raina, C. Njinda, and R. Molyneaux, “How seriously do you take possible-detect faults”, in *Proc. Int. Test Conf*., Nov. 1997, pp. 819-828.

[3] T.J. Powell, “Consistently dominant fault model for tri-state buffer nets”, in *Proc. VLSI Test Symposium*, Apr.-May 1996, pp. 400-404.

[4] G. G. Shahidi, “SOI technology for the Ghz era”, *IBM Journal of Research and Development*, vol. 46, no. 2/3, pp. 121, 2002.

[5] F. Assaderaghi, G. G. Shahidi, M. Hargrove, K. Hathorn, H. Hovel, S. Kulkarni, W. Rausch, D. Sadana, D. Schepis, R. Schulz, D. Yee, J. Sun, R. Dennard, and B. Davari, “History dependence of non-fully depleted (NFD) digital SOI circuits”, in *Proc. Symposium on VLSI Technology*, Jun. 1996, pp. 122-123.

[6] D. Toneva and G. Giles, Private communication, Advanced Micro Devices, Austin, TX, Nov. 2004.

[7] N. Itazaki and K. Kinoshita, “Test pattern generation for circuits with tri-state modules by Z-algorithm”, IEEE *Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 12, pp. 1327-1334, Dec. 1989.

[8] T. Kishi, M. Ohta, T. Taniguchi, and H. Kadota., “A new inter-core built-in-self-test for tri-state buffers in the system-on-a-chip”, in *Proc. Asian Test Symposium*, Nov. 2001, pp. 462.

[9] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit design", in *Proc. Custom Integrated Circuits Conf.*, May 2000, pp. 201-204. Available: http://www-device.eecs.berkeley.edu/~ptm

[10] R.D. Blanton and J.P. Hayes, “The input pattern fault model and its application”, in *Proc. European Design and Test Conference*, Mar. 1997, pp. 628.

VITA

Shaishav Parikh was born on October 14, 1980 in Ahmedabad, India. He received his B.E. degree in instrumentation and control engineering in 2002 from L.D. College of Engineering, Ahmedabad, India. His research interests are in the field of testing and design for testing for digital circuits. His permanent mailing address is: 201, Sathsangath Apartments, Satellite, Ahmedabad, India – 380015.