Bridging Fault Modeling and Testing of Integrated Circuits *

Kehuai Wu
s010596@student.dtu.dk

Xin Meng
s020983@student.dtu.dk

Abstract

This paper gives a general overview of the modeling and testing techniques of integrated circuit. First the models of bridging fault at circuit, gate and switch level are described. Then the bridging fault testing techniques are discussed, which involves both logic-based test and I_{DDQ} test. Finally the undetectable shorts and technique to avoid such faults are discussed.

1 Introduction

It is shown by inductive fault analysis that the most frequently occurred fault generated during chip fabrication is short between unexpected nodes. Shorts will in many cases cause chip's logical behavior differ from fault-free ones, sometimes even leads to oscillation at circuit's outputs. When two shorted nodes have different logical values, it will increase chip's power consumption and generate extra heat. Shorts between gate and source results in transistor stuck-open; those between drain and source are considered as transistor stuck-on; those between transistor input and power supply are modeled as stuck-at fault; those between transistor inputs are called bridging, while the shorts between gate and drain leads to more complicated circuit behavior, which is hard to model at abstract level.

Those shorts between different nodes makes the faulty chips behave in different manner. Some faults can be activated and propagated to PO as a logic error by a set of sequenced test vector. Some faults will only lead to delay fault which does not affect output's logic level. Some errors are even not possible to be detected by any logic-based test. The variety of circuit behaviors make the shorts difficult to detect with one test scheme.

This paper concentrates on the study of bridging fault, which is defined as the short between logic inputs. This subclass of short can be effectively detected by I_{DDQ} test. Such test could be carried out in two ways: on-chip testing and off-chip testing. Off-chip testing is less applicable due to the chip complexity and measurement accuracy conflict. The on-chip testing requires extra current sensor built on-chip, which results in some area trade-off, but is faster and more sensitive.

Unfortunately, the possible number of bridging fault in a circuit is of O(n^2), while n is the number of nodes in a circuit. Since the I_{DDQ} test takes very long time for each test pattern and the area overhead for test circuit on-chip become unacceptably large, the test strategy of bridging fault cannot be based on I_{DDQ} alone.

To make the bridging fault test possible, the logic test has to be used in part. How to model the bridging fault and how to effectively generate the test pattern for both logic-based bridging fault test and I_{DDQ} test has been the topics in this area. In this paper, we summarize the previous work and give an overview in this area

2 Bridging fault modeling

The unintentional shorts between connection lines are called bridging faults. With N gates in a circuit, there could exist O(N^2) possible bridging short even if we only consider short between two lines. In practice, it is not possible to consider all of them. Luckily, the occurrence of faults is strongly related to chip's layout geometry, thus a large fraction will scarcely occur. The bridging faults model discussed here only consider shorts between two gate inputs.

2.1 Circuit level fault model

Three draft requirements for a bridging to be logically observable are: a) in order to activate a bridging fault, the shorted nodes must be given different logic values; b) one of the shorted node's logic value must be corrupted enough c) the erroneous logic value must be propagated to the PO. The circuit model of a CMOS circuit is shown in figure 1.

*This paper is handed in as an assignment answer in lecture course *02208 Test of Integrated Circuits (2002)* at the Informatics and Mathematical Modelling department, Technical University of Denmark.

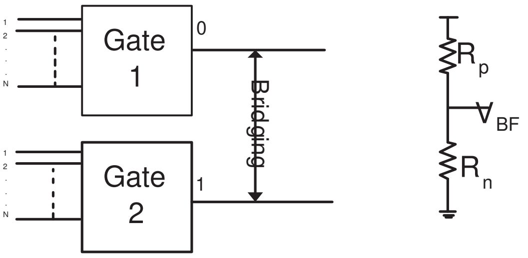

Figure 1. CMOS bridging fault and circuit model

Apparently, the logic level of the erroneous signal will be determined by the conductance of the pull up and pull down transistor network as discussed in [3]. Transistor dimension and process factor will also have deterministic effects. Figure 2 shows a circuit tested with 3 different patterns. The conductance of all transistors are assumed to be the same. Even though the logical level of each gate under all 3 test pattern are the same, the final erroneous output could be non-deterministic, 0 (wire AND) and 1 (wire OR) respectively, for certain transistor threshold voltage.

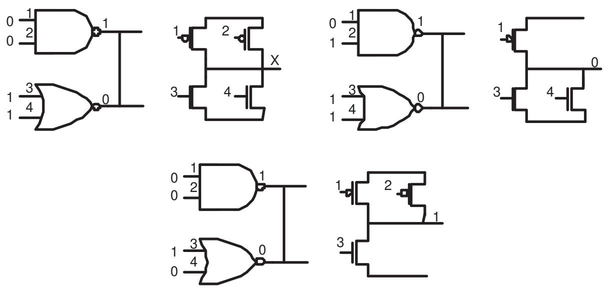

Figure 2. Same bridging fault and various outputs

Another factor which could have influence on fault behavior is the resistance of short. When the resistance of short is comparable with pull up/pull down network, the short will be even harder to detect. The only thing can be guaranteed is that a test which can detect a fault with short resistance of R could also test a short with resistance less than R.

The bridging fault is hard to be observed with logic-based test along. It could be seen that the test pattern generation has to consider more constraints compares to that of the stuck-at fault. The model at circuit level should be simulated with tools like SPICE, which needs more computation time compares to simple logic simulation. Conceptually, a realistic improvement could be to only simulate with SPICE the part of the circuit that pull up and pull down the faulty nodes and the path which is sensitized to propagate the fault-affected logic value till some point in the path the faulty voltage become logic 0 or 1. The rest of the circuit should still be simulated with simpler logic simulation.

2.2 Gate level fault model

The gate level bridging fault models the short as a simple wire AND or wire OR gate based on [7]. Such a model is used in bridging fault test simulation with stuck-at test sets. Although it is not the best way to present the circuit aspect of bridging short, it is very practical due to it's simplicity.

Any bridging fault falls into one of these two types: non-feedback bridging faults and feedback bridging faults. A bridging fault is called feedback bridging fault if one of the shorted node's logic value is affected by the other in the fault free circuit. Those two kinds of bridging fault are modeled in two different ways.

A non-feedback bridging fault could be modeled as in figure 3.

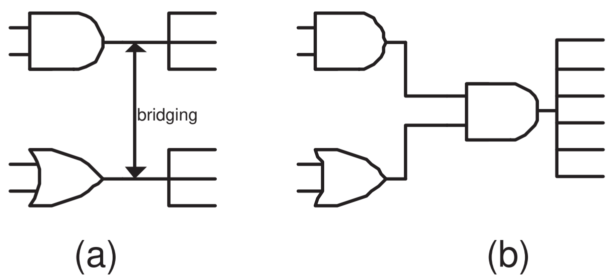

Figure 3. (a)Non-feedback faulty circuit (b) The Wire AND fault model

Feedback bridging fault's model is more complicated than that of a non-feedback one. The figure 4 shows how a wire AND logic fault can be modeled.

To simulate the faulty circuit, a test vector is applied on the circuit for 3 clock cycles. If the value of the flip-flop output changes at the second and third clock cycle, which corresponds to the oscillation in an actual circuit, the fault is detected by this pattern. In figure 4.b, the input set A=(a_1,...a_n) will be assigned constant values that sensitize z to x. Value of x must be a noncontrolling value to the wire logic, which is 1 in our case. The value z will only be guaranteed to oscillate during 2nd and 3rd clock cycle if looped circuit have a function

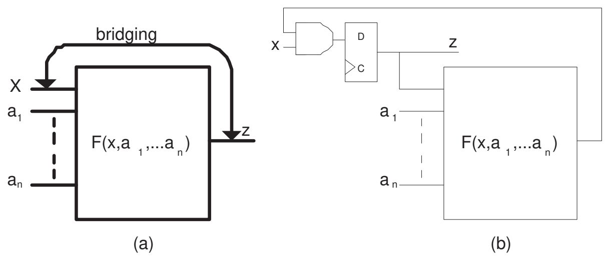

Figure 4. (a) Feedback faulty circuit (b) The Wire AND fault model

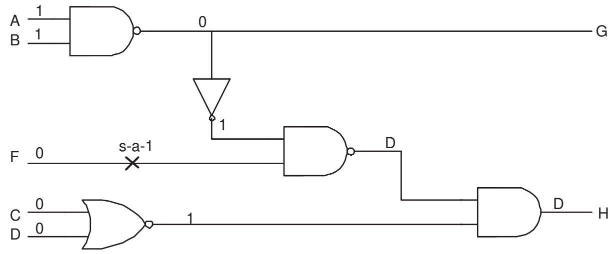

Figure 6. One of four possible fault manifestations as stuck-at fault in a modified circuit.

F(0,A)=1 and F(1,A)=0. If the circuit function differ from this pair, the fault might only be tested if there is an error value on node z, and that value can be propagated to PO.

Those gate level models require the two shorted nodes be set to two opposite logic. To generate test pattern, an ATPG which is different from stuck-at fault ATPG is required. Reference [2] have proposed a method that can represent a bridging fault as a stuck-at fault by some insignificant circuit modification. Figure 5 is a circuit with bridging fault present.

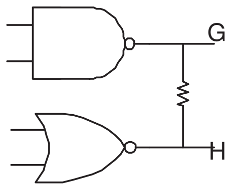

Figure 5. A faulty circuit.

There are 4 possible erroneous circuit behavior:

1. (G,H) justified to (0,1), but fault cause G=1;
2. (G,H) justified to (0,1), but fault cause H=0;
3. (G,H) justified to (1,0), but fault cause G=0;
4. (G,H) justified to (1,0), but fault cause H=1;

To generate test for the second case above, we can modify the circuit as shown in figure 6. The problem to generate a test for (G, H) = (0, D) will be transformed into the problem to generate a test for dummy input F stuck-at-1.

Figure 7 shows a modified circuit that transforms all 4 possible faulty behavior into 4 single stuck-at faults. The modified circuit can only assist on test pattern generation. The fault simulation is still carried out on the unmodified circuit. Since this model only consider logic function introduced by bridging fault to be ideal, it is possible that generated pattern will fail to make erroneous output become logic 1 or 0 in practice.

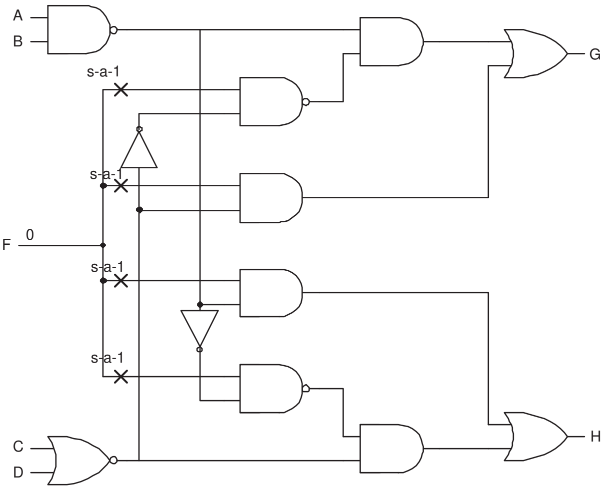

Figure 7. All four possible fault manifestations as stuck-at faults in a modified circuit.

2.3 Switch level fault model

In switch-level simulation, transistor conductance and node capacitance are modeled by discrete strengths. Transistors are assigned different strengths based on their conductance. The strength of a path of conducting transistor is decided by some transistors of dominant properties in the path. When several paths drive a node, the strongest path wins and drives the node to logic 0 or 1. If no strongest path is present, the node is set to logic X. Sometimes this model might be inaccurate, but the simulation computation of this model compares to SPICE model is still less.

Reference [2] defines strong and weak logic values. A "strong" logic value is a logic value produced by a gate whose inputs minimize the resistance between V_{DD} or V_{SS} and the output. A "weak" logic value is produces by gate inputs that maximize such resistance. Table 1 summarize the input combination required to produce strong and weak logic value on simple gates.

Gate	Strong 0	Weak 0	Strong 1	Weak 1
NAND	I	I	000...0	000...1
NOR	111...1	100...0	I	I
AND	I	I	I	I
OR	I	I	I	I
XOR	00	11	10	01
XNOR	10	01	11	00

Table 1. Input required to produce strong and weak logic values on simple gates

By using such a model, the test vector generated must meet the following three conditions which expand the earlier 3 requirements:

1. Two shorted nodes must be justified to opposite logic values, and they should be justified to opposite strengths if possible.
2. A logical error must be introduced on the node with weak logic value presented.
3. The error must be propagated to a PO.

By using such a model, the problem presented in figure 2 will be better considered. But it is still not possible to guarantee the weak logic value to be corrupted, since the transistor dimension is not addressed in this model.

3 I_{DDQ} testing principle

Quiescent power supply current (I_{DDQ}) testing has been a very effective test in detecting bridging fault in CMOS circuit. But IC now-a-day is growing too large so that the I_{DDQ} test for one whole chip cannot guarantee a chip to be fault-free. Normally, a chip needs to be partitioned into sections and have many on-chip current sensor, e.g. BIC, doing measurement on chip. The measurement circuit area overhead need to be reasonably small. Also, I_{DDQ} testing takes very long time compare to logic test. The number of test pattern need to be the smaller the better in order to make I_{DDQ} test practical. Furthermore, a good test should detect the fault the sooner the better. These requirements demand the test generation of bridging fault well considered.

3.1 I_{DDQ} test approach

As stated earlier, a circuit which has N nodes will have O(N^2) possible faults between even only two nodes. It is not necessary the truth since most of those pairs will hardly be shorted due to their geometric distribution on chip. By using some fault extraction programs like FAULTAN the number of fault need to be tested can be reduced from O(N^2) to O(N). But, even though the number of fault has been significantly reduced by layout analysis, it is still not practical to implant so many current sensors on chip. Another good way to reduce the number of faults need to be tested by I_{DDQ} test is to carry out logic-based test on suspicious node pairs. If a bridging fault behaves like wire logic, it will normally be tested by single stuck-at test patterns.

Figure 8 describes the overall test strategy for I_{DDQ} test. It is important to note during the second phase where the logic based fault simulation take place, it is desired to assign the shorts high resistance. Since a high resistance fault is always harder to detect, a good simulation results coming from higher resistance implies a even better test result when fault resistance is not so high.

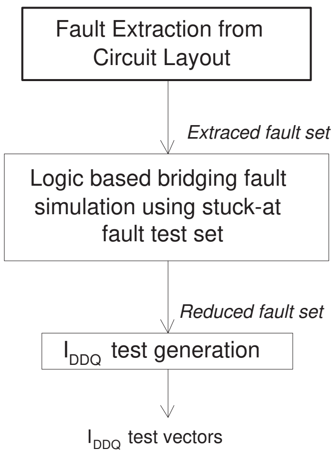

Figure 8. I_{DDQ} test strategy

3.2 I_{DDQ} test pattern generation

After the first two steps, there will be very few faults left that require I_{DDQ} test. But a highly efficient TPG scheme is still needed to minimize the test time. The I_{DDQ} have no observability requirements, but it requires a faults be found as soon as possible. A parallel test has to be the choice and good test should have maximum parallelism. Moreover, test sequence should be arranged in a way the earlier applied few vectors can activate more faults than later vectors.

Reference [1] and [5] have proposed a simulation-based genetic algorithm (GA) to achieve those demands. This GA has a population of strings (chromosomes), where each of these strings is a sequence of test vector. The ordering of vector in each string is implied by vector's neighboring relationship. The characters (gene) in each string will be evolved to fit a fitness function, where this fitness reflects the quality demand of test pattern. Each string's fitness is based on the circuit simulation result of this particular string.

Each string in the population is randomly initialized. Then each string's fitness will be evaluated. After that, the strings with best fitness will be selected. For each pair of selected strings, they will be crossed to create two offspring by randomly swapping their characters. Sometimes the offspring's gene will be mutated in order to diversify the gene pool. New individuals will form a new population with best individuals from earlier generation. After the population has been evolved for several generations, the best strings from all generations will be chosen as test set.

The fitness function is used to maximize fault coverage and minimize test set length.

$$fitness = \sum_{i=1}^{n}(n-i+1)F_i$$

where n is the length of the test vector sequence and F_i is the fraction of remaining fault first detected by the ith vector of the string. The GA will attempt to maximize the fault coverage of the first few vectors because of the way vectors are weighted. Figure 9 gives an overview of this GA.

```
specify sequence-length
specify maximum-test-set-length
test-set={}
test-set-length=0
fault-coverage=0
While test-set-length<maximum-test-set-length and
fault-coverage!=100%
        sequence=
                generate-test-sequence(sequence-length)
        fault-coverage=fault-simulate(sequence)
        test-set+=sequence
        test-set-length+=|sequence|
```

Figure 9. I_{DDQ} **test generation algorithm**

3.3 I_{DDQ} Test application

3.3.1 Improving test generation efficiency

Reference [4] have divided shorts into two types: internal shorts are shorts within a gate or short between gate input/output and somewhere inside a gate; external shorts are shorts between gate inputs, outputs and voltage sources.

For a combinational circuit, it has been seen that external shorts can be easily tested by pseudo-random test pattern. In another words, test patterns aiming at internal short will be able to test large fraction of external short as by-product, since those patterns act as random test set for external shorts. A pattern generation strategy is presented as follows:

1. Test generation of the internal shorts.
2. Test generation of the external shorts.
    (a) Drop external short already tested from internal short test pattern.
    (b) Drop undetectable external shorts.
    (c) Generate test for the remaining external shorts.
3. Test quality evaluation.

3.3.2 Select most efficient patterns

Scan-based Sequential circuit has even higher demand for test efficiency. Besides the time it takes for I_{DDQ} testing, scanning a pattern into circuit make test even more expensive. What can be done in practice is to select the best vectors from a set of vectors that have higher fault coverage than required. Suppose a sequential circuit need to be tested and a fault coverage of C_{goal} must be achieved. By using some ATPGs a larger set of pattern S_N that has a coverage higher than C_{goal} can be generated. A greedy algorithm proposed in reference [4] can be used to pick up the shortest set of test $S_{N'}$ that have the coverage equal to C_{goal}. The greedy algorithm is shown in figure 10.

3.3.3 Test strategy

The test application should take coverage of different types of fault into account. Those faults that are most difficult to test should be detected since those patterns might have opportunity to detect an easier-to-detect fault. Reference [4] have ranked the test difficulty of internal and external shorts appear in two possible parts of the circuit as in table 2. Whenever a test on a chip carries out, it starts from the fault of rank 1 to that of rank 4.

Even though there are no observability requirements when doing I_{DDQ} test, it will be a good idea to scan out

```
CD := 0  K:=0  /*C D is the accumulative coverage*/
Select v from SN with higher CD
Update C and K
Load v into the scan path
While (CD <Cgoal )  do {
    Select v from  SN by using fault simulation, such that
    the L vectors produced by the shift of v into the scan
    path provide a higher increase in C D
    v-> SN'
    Remove v from SN
    Update CD and K
}
```

Figure 10. Greedy test selection algorithm

Rank	Fault type	Part of circuit	Description
1	Internal	Combinational	Harder to detect than external fault
2	External	Combinational	Also cover most of the external short in the scan path
3	External	Scan path	Also detect I_{DDQ} detectable internal short
4	Internal	Scan path	

Table 2. Fault ranking

the value in the memory element. The reason is that some shorts between flip-flops produce logic error in memory cell without affecting the level of I_{DDQ}.

4 Logic-based bridging fault test

By combining modeling technique discussed in figure 7 and strength model discussed in section 2.3, an efficient method of bridging fault testing could be found. Firstly, a circuit with bridging fault could be modified as shown in figure 7 with some more consideration of the input strength. Secondly a stuck-at fault TPG can be used to generate pattern which will try to maximize the strength difference of the bridged two nodes.

4.1 Test pattern generation of combinational circuit

Recall the faulty circuit presented in figure 5. By considering the strength model and the three conditions discussed in section 2.3, A complete test set for this bridging fault should try to set the logic values on node G and H as following, even if not all condition can be satisfied.

1. (G,H) justified to (weak 0, strong 1), but fault cause G=1;
2. (G,H) justified to (strong 0, weak 1), but fault cause H=0;
3. (G,H) justified to (weak 1, strong 0), but fault cause G=0;
4. (G,H) justified to (strong 1, weak 0), but fault cause H=1;

The corresponding stuck-at fault test circuit should be modified as shown in figure 11. Compare to the circuit in figure 7, more specific constraints on input combination are given here.

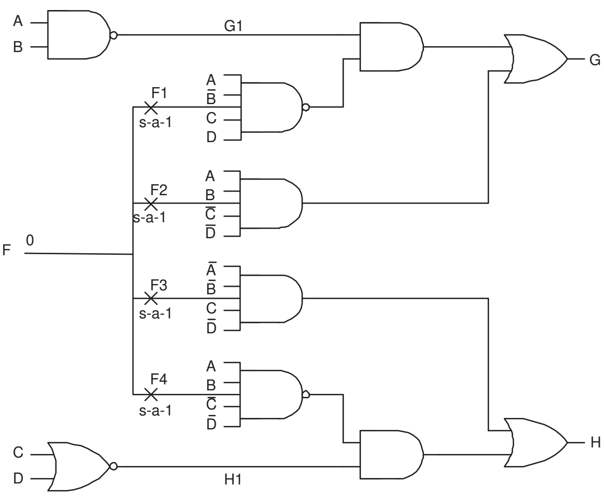

Figure 11. Circuitry for strength model

By using any stuck-at fault TPG to generate test for F1, F2, F3 and F4 stuck-at-one, a complete set of test for the bridging fault can be found. Table 3 summarizes the resulting test.

Fault	A	B	C	D	G1	H1	G	H
F1 s-a-1	1	0	1	1	weak 1	strong 0	1/0	0
F2 s-a-1	1	1	0	0	0	1	0/1	1
F3 s-a-1	0	0	1	0	strong 1	weak 0	1	0/1
F4 s-a-1	1	1	0	0	0	1	0	1/0

Table 3. Test generation result for strength model

4.2 Generalized model for sequential circuit

Due to the complicity of the sequencing of test input, the circuit modification as shown in figure 11 is not efficient. A more neat way to model the sequential circuit is presented in figure 12.

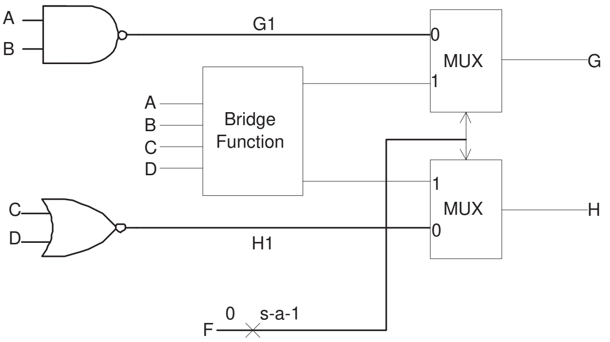

Figure 12. Generalized bridging fault model

During the test generation, F remains at 0 as before and the targeted stuck-at fault is still the F stuck-at-one. The group of bridging functions produces faulty outputs under various input vectors and sequences. It can be seen that the number of bridging function is very large if all possible inputs are considered. Generally it is impossible and unnecessary to find the complete test set for all bridging function, thus some technique to reduce the test set is needed, e.g. finding and eliminating equivalent inputs.

5 Undetectable bridging faults

5.1 Impact on TPG

A bridging fault is said undetectable if this fault cannot make any value on any PO differ from fault-free circuits as defined in [6] . In another word, it does not change circuit's logic function. Even though such fault has no influence on circuit logic output, it cannot be ignored. The reason is such fault will cause reliability problem by increasing chip's power consumption, introducing extra heat, noise and creating circuit delay.

Undetectable bridging fault can be divided into two classes: nonstimulatable and nonpropagatable. A bridging fault is nonstimulatable if two shorted nodes always have the same logic value. A bridging fault is nonpropagatable if the corrupted fault cannot be propagated to PO. As test pattern generator sometimes uses heuristic to generate test patterns, the undetectable faults are likely going to make trouble. If a heuristic program tries to generate a test for nonstimulatable fault, it will waste computation time trying to justify all the paths from PI to fault site in the fault site's downcone. A similar problem will occur if TPG tries to propagate a nonpropagatable fault to PO. It will be an improvement to identify the undetectable fault and drop them out of the fault list before generating test.

5.2 Avoid undetectable fault

In order to avoid the undetectable bridging, the origin of these faults must be understood. After that, strategy to avoid undetectable fault and identify them before test generation will be possible.

5.2.1 Nonstimulatable fault

Nonstimulatable fault could only come from redundancy. However, redundancy cannot be eliminated from layout easily. Consider hierarchical design as shown in figure 13 and redundancy in figure 14. Those duplication or redundancies are very hard to avoid, some time they are even wanted. Another two kinds of wanted redundancy are introduced to drive several part of circuit in different speed, and redundancy used to reduce wire delay when chip feature size is so small that wire delay become dominant.

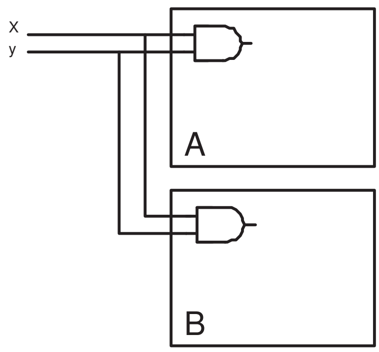

Figure 13. Redundancy from hierarchical design

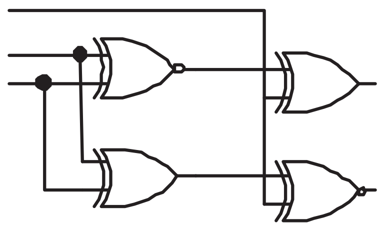

Figure 14. Redundancy from circuit design

Those faults would not introduce any logical error, nor extra quiescent current. The only way that can probably detect them is by delay fault test. Those potential faults can be predicted by checking each node's logic function. Those faults connecting nodes which have equivalent function should be dropped from I_{DDQ} test and logic-based test target list. It should be also possible to reduce the probability of their occurrence by separating them on the chip layout.

5.2.2 Nonpropagatable fault

Nonpropagatable faults occur in two cases. The first case is called function masking. On figure 15 there is a fault that might be either a wire AND or a wire OR, depending on the transistor strength. If n-channel MOS is stronger than p-channel transistor, the short will act as a wire AND. In this case, the output of gate F will be 1, which is the same as the fault-free circuit. In general, a fault masking will occur when the short perform the same function as part of its propagation path. In our case, short acts as a wire AND and both shorted nodes are connected to an NAND gate.

The second case is called stimulation mask. In figure 15, assume the short still act as wire AND like before. Another gate G is added to propagate the corrupted logic 1 from $\bar{x}$. In order to corrupt the logic 1 on $\bar{x}$, y must be set to 1. This 1 happens to be the controlling value of gate G. Still, the bridging fault cannot be tested.

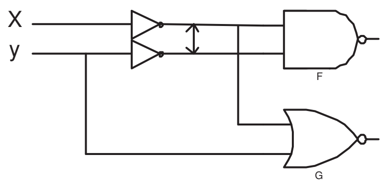

Figure 15. Nonpropagatable bridging fault

Prediction of nonpropagatable fault requires more techniques, since the behavior of short need to be known forehand. By analyzing fault behavior and its propagation path, function masked faults could be dropped from fault list. But those faults should be easily tested with I_{DDQ} test.

6 Conclusion

In this paper, we have given an overview about bridging fault testing. We have discussed several ways of modeling the faults and have summarized the strategy to organize and generate the test generation scheme based on those models. Some method of test generation about both logic-based and I_{DDQ} based test are discussed in detail. Finally, undetectable faults are considered.

Bridging fault is a very expensive test to carry out in terms of test time and hardware, especially for large chips and sequential circuit. Topics like modeling faults with maximum analogue aspect but minimum computation to evaluate short effect, detecting most of bridging fault with logic-based test scheme and organizing the test to explore fault earlier has led to serious consideration. It has been shown as the feature size decreases, the fault testing with high coverage becomes more difficult.

References

[1] T. Chen, I. N. Hajj, E. M. Rudnick, and J. H. Patel. An efficient i_{DDQ} test generation scheme for bridging faults in cmos digital circuits. *International Workshop on IDDQ Testing*, pages 74–78, 1996.

[2] J. P. Cusey and J. H. Patel. Bart: A bridging fault test generator for sequential circuits. *International Test Conference*, pages 838–847, 1997.

[3] M. Favalli, P. Olivo, M. Damiani, and B. Ricco. Fault simulation of unconventional faults in cmos circuits. *IEEE TCAD*, 10(5):677–682, May 1991.

[4] E. Isern and J. Figueras. Test of bridging faults in scan-based sequential circuits. *European Design and Test Conference*, pages 366–370, 1994.

[5] T. Lee, I. N. Hajj, E. M. Rudnick, and J. H. Patel. Genetic-algorithm-based test generation for current testing of bridging faults in cmos vlsi circuits. *Proceedings of the VLSI Test Symposium*, pages 456–462, 1996.

[6] R. McGowen and F. J. Ferguson. A study of undetectable non-feedback shorts for the purpose of physical dft. *European Design and Test Conference*, pages 371–375, 1994.

[7] S. D. Millman and E. J. McCluskey. Detecting bridging faults with stuck-at test sets. *International Test Conference*, pages 773–783, 1988.