Charge Sharing Fault Detection for CMOS Domino Logic Circuits

C. H. Cheng, S. C. Chang, J. S. Wang*, and W. B. Jone**

Department of Computer Science and Information Engineering
*Department of Electrical Engineering
National Chung Cheng University
Chiayi, Taiwan, Republic of China

**Department of Computer Science
New Mexico Tech
Socorro, NM 87801
U. S. A.

{chc, scchang, jone@ cs.ccu.edu.tw, ieegsw@ccunix.ccu.edu.tw}

Abstract

Because domino logic design offers smaller area and higher speed than conventional CMOS design, it is very popular in the high performance processor design. However, domino logic suffers from several design problems and one of the most notable design problems is the charge sharing problem. In domino logic, there are two operations: the pre-charge phase and the evaluation phase. The charge sharing problem occurs when the charge which is stored at the output node in the pre-charge phase is shared among the junction capacitance of transistors in the evaluation phase. Charge sharing may degrade the output voltage level or even cause erroneous output value. In this paper, we describe a method to measure the sensitivity of the charge sharing problem for a domino gate. For each domino gate, we compute a value called CS-vulnerability which describes the degree of sensitivity for a domino gate to have the charge sharing problem. In addition, our algorithm also generates test vectors to activate the worst case of the charge sharing problem. We have performed experiments on a large set of MCNC benchmark circuits.

1. Introduction

Unlike a CMOS static logic gate where complementary functions are implemented in N and P transistors, an N-type domino gate requires only the N transistors to perform the pull-down function and uses a pre-charge clocked pMOS to perform the pull-up function. Due to the saving of P-transistors, domino logic offers significantly higher speed and smaller area than static CMOS circuits. As a result, domino logic is extensively used in today's high performance processors [1-3, 5, 9, 12]. A typical domino AND gate design is shown in Fig. 1.

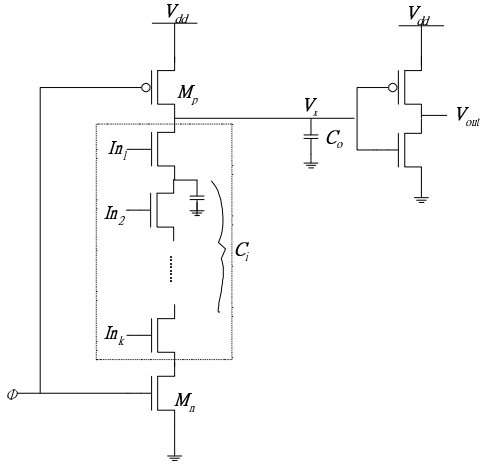

Figure 1. A CMOS domino AND gate.

Domino gates have two basic operating phases which are the pre-charge phase when the clock ϕ is low, and the evaluation phase when the clock ϕ is high. During the pre-charge phase, the capacitance C_o will be charged to high. During the evaluation phase, if all the inputs In_i to N-transistors are high, the voltage V_x is pulled to ground.

Despite several advantages, domino circuits require special care to avoid the charge-sharing problem which may result in an erroneous output value [11, 13]. The charge-sharing problem is termed as the ***CS*** problem in the following discussion. Consider the domino AND gate in Fig. 1. During the evaluation phase, suppose all inputs In_i are high except input In_k located next to the clocked transistor M_n is low. Since In_k is low, ideally, the value of V_x should remain high; however, the charge in C_o loaded in the pre-charge phase is shared or re-distributed to the (source-drain) junction capacitance of those turned-on transistors. Let the equivalent capacitance for the junction

capacitance be C_i as shown in Fig. 1. The output voltage V_x becomes V_{dd}* $C_o/(C_o+C_i)$ in the evaluation phase. If C_i is large enough, V_x may become too low to turn on the N-transistor of the subsequent inverter. As a result, under certain input conditions, charge re-distribution among junction capacitance at internal nodes can cause error or glitch at outputs [7, 10].

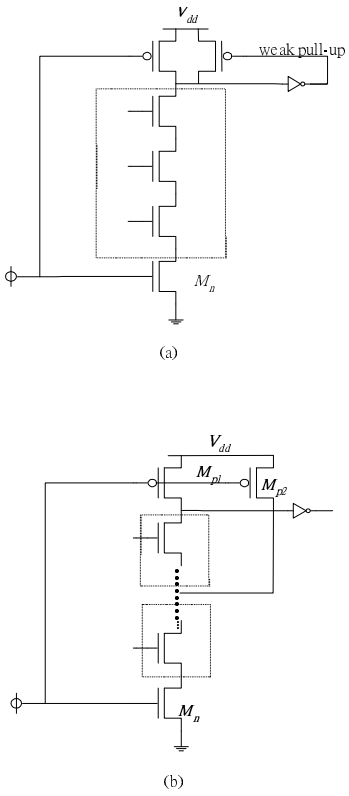

Figure 2. (a) Addition of a weak pMOS pull-up transistor.

(b) Internal node pre-charging.

Several researchers have attempted to prevent the CS problem in domino logic design. For example, a weak pMOS pull-up transistor can be added to the output of a domino gate shown in Fig. 2 (a) [4, 8]. The weak pull-up device in the feedback loop can be used to prevent the loss of output voltage level due to the CS problem. The weak pull-up logic, however, is sensitive to the pull-up and pull-down ratio of transistors and therefore is sensitive to process variation. Another way of preventing the CS problem is to pre-charge internal nodes as shown in Fig. 2(b) [4, 8]. Though the CS problem can be alleviated, both solutions can incur area overhead as well as circuit performance degradation. Further, faults occurring at the weak pull-up loop or the internal node precharing circuit may cause CS faults. Therefore, CS fault detection for domino circuits is necessary. In this paper, we model CS faults as stuck-at-1 faults and develop an efficient algorithm which finds the CS-vulnerability for each gate. We also generate the corresponding test vector of each gate for detecting CS faults.

Consider again the circuit in Fig. 1 which does not apply any charge sharing prevention technique. For the CS problem **not** to occur, the voltage $V_x = V_{dd}*C_o/(C_o+C_i)$ must be greater than the threshold voltage V_{thresh} of the inverter. Among the variables, the value C_o can be obtained from layout parameter extraction and V_{thresh} can be determined from the viewpoint of the reliable circuit design. After obtaining C_o and V_{thresh}, one can calculate the maximum value of C_i, called $C_{ithresh}$, so that the CS problem does not occur where $C_{ithresh} = (V_{dd}*C_o/V_{thresh}) - C_o$. On the other hand, the major contributor of C_i is the junction capacitance of consecutive turned-on transistors next to the output voltage V_x. Therefore, the value of C_i depends on the input conditions to turn on/off transistors. Also for a CS fault to be observable at outputs, some values must be set to sensitize propagating paths. Among all possible input patterns, suppose C_{imax} is the largest one of C_i for the fault to be observable. We define the *CS-vulnerability* of a domino gate to be $C_{imax}/C_{ithresh}$. This idea of CS-vulnerability is to describe the degree of sensitivity for a domino gate to have the CS problem. When the CS-vulnerability of a domino gate is larger than 1, that is $C_{imax} > C_{ithresh}$, the CS problem for the domino gate can occur for some input patterns. If the CS-vulnerability is smaller than 1 but close to 1, the CS problem is also likely to happen because process variation may change the values of C_i and C_o. On the other hand, when the CS-vulnerability is much smaller than 1, it is unlikely to have the CS problem for the domino gate. The advantages for finding the CS-vulnerability of each gate are two fold. First, in the designer side, the CS-vulnerability of a gate gives hints about whether the gate needs special care for preventing the CS problem. Secondly, for those gates with high CS-vulnerability, it is desirable for test engineers to find test vectors to detect possible CS faults. In this paper, we develop an efficient algorithm which finds the CS-vulnerability for each gate and also the corresponding test vector for detecting the CS fault.

The organization of this paper is as follows. In Section 2, we describe the CS fault model, then provide an algorithm to compute the CS-vulnerability and to generate the test patterns for CS faults. Experimental results obtained by simulating a set of MCNC benchmark circuits are presented in Section 3. Finally, conclusions are provided in Section 4.

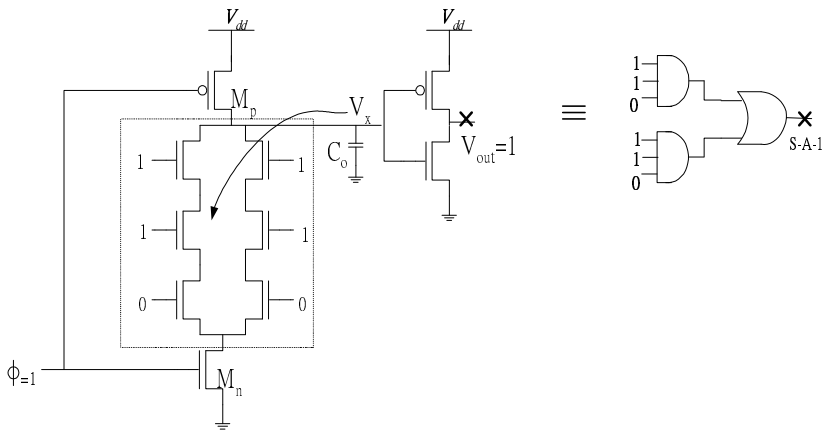

Figure 3. The gate level model for a CS fault.

2. Fault Model, CS-vulnerability and Test Pattern Generation

The operation of domino CMOS logic circuits is based on first precharging the output node capacitance, and subsequently evaluating the output level according to the applied inputs. When the clock signal is low (precharge phase), the pMOS precharge transistor M_p is conducting, at the same time the complementary nMOS transistor M_n is off as shown in Fig. 1. The output parasitic capacitance of the circuit is charged to logic 1 through the conducting M_p ($V_x = V_{dd}$). Note that the inputs are also applied in this phase, but they have no influence upon the output V_{out} as the M_n is off. When the clock signal is high (evaluation phase), the output capacitance will be discharged or remain at the high state depending on the input values.

Consider the CMOS domino logic shown in Fig. 1 where the intermediate node capacitance C_i is comparable in size to the output node capacitance C_o. We further assume that all inputs are low initially, and that the intermediate node voltage of C_i equals *0 V*. In the precharge phase, the output node capacitance is charged to V_{dd}. If In_1 is high and In_2 is low, the stored charge in C_o will be shared by C_i and the output voltage becomes $V_x = V_{dd} * C_o/(C_o+C_i)$ during the evaluation phase. When the uppermost k-1 inputs are all high, the phenomenon of *charge sharing* is especially significant. Several methods have been proposed to prevent erroneous output levels due to charge sharing in domino CMOS gates. By adding a weak pMOS pull-up transistor to the dynamic gate output as shown in Fig. 2(a), we can have the gate output remain at logic 1 unless all pull-down transistors are on. It is also possible to add several weak pull-up transistors to internal nodes among pull-down transistors as shown in Fig. 2(b).

The idea of using weak pull-up transistors to deal with the charge sharing problem appears good; however, the addition of weak pull-up transistors also turns the domino design to a circuit which is sensitive to the pull-up and pull-down ratio. Thus, the circuits are sensitive to fabrication process variation. Even worse, faults occurred at the charging node M_p or on the charging path (e.g., the conducting line width is too small) will destroy the original design. These abnormal cases might drop V_x below the threshold voltage of the following inverter which will erroneously switch to high. Thus, charge-sharing faults must be tested.

2.1 Single CS Fault

This subsection concentrates on the behavior for a single CS fault. Thus unless otherwise stated, by CS (stuck-at) fault, we mean single CS (stuck-at) fault. A CS fault behaves as if a stuck-at-1 fault appears on the output stage during the evaluation phase, but the fault may not occur constantly. As shown in Fig. 3, if only the lowest inputs on both pull-down paths are low and other inputs are high, then the probability of a CS fault occurrence is high. However, the fault might not be activated by other inputs if the charge sharing effect is too senseless to be observed. Thus, the CS fault may or may not occur depending on the input patterns applied. Unless one of the most sensitive test patterns is used, detection of the CS fault cannot be guaranteed. Consequently, the detection of

CS faults is different from that of stuck-at faults because test pattern generation for the former case requires to activate (turn on) several inputs at the same time.

2.2 The CS-vulnerability of a Domino Gate

Let V_x be the input voltage of a domino's inverter. When V_x is evaluated to high in the evaluation phase, because of the CS problem, voltage V_x will not be equal to V_{dd}. Instead, V_x is equal to $V_{dd}*C_o/(C_o+C_i)$ where C_o is the effective capacitance load at the input of the inverter and C_i is the effective junction capacitance. When C_i is large enough, the value of V_x can be smaller than the logic threshold voltage V_{thresh} which is the minimum voltage to turn on the N-transistor of the subsequent inverter. Therefore, a CS problem can cause an erroneous output value.

When the value $V_{dd}* C_o/(C_o+C_i)$ is smaller than V_{thresh}, a CS fault can arise. Among these variables, the values of C_o and V_{thresh} can be extracted from a domino gate and are independent of the structure of a circuit to which a domino gate belongs. Suppose C_o and V_{thresh} are obtained in advance. We define $C_{ithresh}$ to be $(V_{dd}*C_o/V_{thresh}) - C_o$ which describes the maximum value of C_i to avoid a CS problem. On the other hand, the value of C_i is largely contributed by the junction capacitance of those consecutive transistors that are turned on and next to the output V_x. So, the value of C_i depends on the input patterns. In addition, for a CS fault to be observable at outputs, some values must be set to sensitize the fault. For an observable CS fault, among all possible input patterns, let C_{imax} be the largest C_i. We define the CS-vulnerability of a domino gate to be $C_{imax}/C_{ithresh}$.

Conceptually, when the CS-vulnerability of a domino gate is greater than 1 (that is $C_{imax} > C_{ithresh}$), the CS problem will occur for some input patterns. If the CS-vulnerability of a domino gate is less than 1 but close to 1, it is still possible to have the CS problem because process variation may change the capacitance values of C_o and C_i. On the other hand, if the CS-vulnerability of a domino gate is much less than 1, the CS fault is unlikely to occur in the domino gate. Therefore, this CS-vulnerability describes the sensitivity of a domino gate which may have the CS problem. Finding the CS-vulnerability is very valuable. First, in the designer side, a designer can use the information of CS-vulnerability to decide which gates require special attention for the CS problem. One can use, for example, the pre-charge internal logic or the weak pull-up logic to alleviate the CS problem for a node with a high CS-vulnerability value. Secondly, in the test engineer side, test patterns must be derived to test those domino gates with high CS-vulnerability values. We now describe how C_{imax} and CS-vulnerability can be found. Without lose of generality, in the following, we assume that each N-transistor has the same size. A general case can be easily extended.

If each N-transistor has the same size, then the junction capacitance is the same and each of the capacitances of C_{imax} *and* $C_{ithresh}$ can be both represented by a certain number of transistors. A transistor which is turned on and can share the pre-stored charge is called a CS-*transistor*. For example in Fig. 4, to propagate the CS fault to primary outputs, there exists a pattern to activate $\{a=1, b=0, d=1, e=0\}$. In this case, transistors $\{Ta, Td\}$ are turned on and can share the pre-stored charge so the number of CS-transistors is two for this test pattern. In fact, among all possible input conditions, one can find that the maximum number of CS-transistors is also two in this example. Our objective is to find a test pattern which can result in the maximum number of CS-transistors for each domino gate.

For a domino gate, one can quickly find that the largest possible number of CS-transistors is equal to the total number of transistors minus the "width" of the domino gate. The width of a domino gate is the number of parallel gates. Such a situation is called the *structural worst* case. For example, in Fig. 4, in order for a CS problem to occur, one transistor in $\{Ta, Tb, Tc\}$ and one in $\{Td, Te, Tf\}$ must be turned off so that the pre-stored charge will not be conducted to ground. The structural worst case for the CS problem is that both the transistors *Tc* and *Tf* are turned off and the charge is shared among the junction capacitances of transistors $\{Ta, Tb, Td, Te\}$ which are turned on. In this case, the maximum possible CS-transistors is 4. This case is referred to as the structural worst case for the domino gate.

In our experience with practical circuits, the structural worst case may not happen for some gates. For example, consider Fig. 4. To simplify the discussion, the domino logic circuit is expressed using the gate level expression and only the domino gate in consideration is expressed in the transistor form shown in Fig. 4(b). The structural worst case is that transistors $\{Ta, Tb, Td, Te\}$ are turned on and transistors $\{Tc, Tf\}$ are turned off. To do so, we must assign $\{a=1, b=1, d=1, e=1, c=0, f=0\}$, and the assignment of which can be shown to be impossible. Therefore, assuming the structural worst case for this domino gate is too pessimistic.

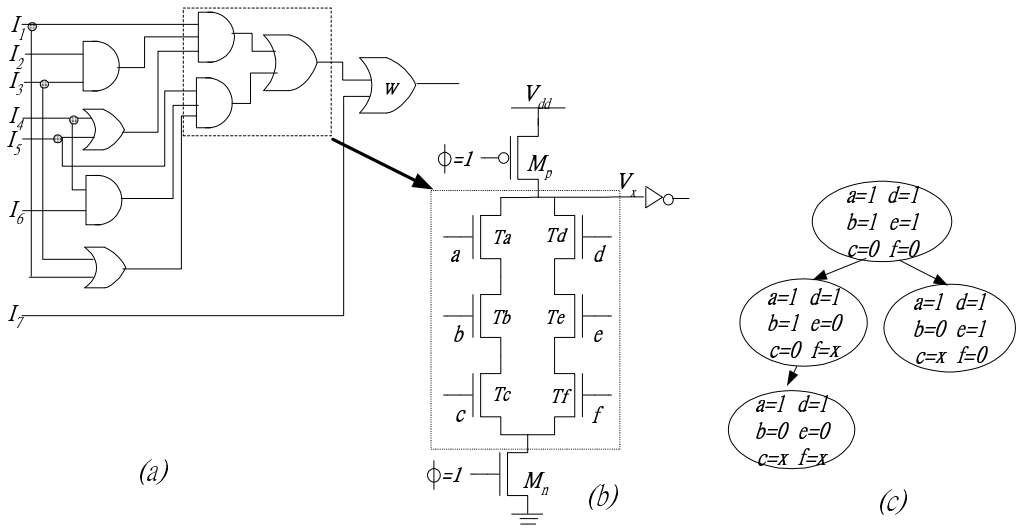

Figure 4. An ATPG example of the domino circuit

2.3 Derivation of Maximum of CS-transistors

Again, our objective is to find the largest number of CS-transistors for a domino gate. To do so, we adopt an exhaustive search method. This exhaustive method attempts to find a test vector which can turn on the largest number of CS-transistors. It starts from the structural worst case of the domino gate. If the fault cannot be activated and propagated, it then selects another fault which has one less CS-transistor than the structural worst case. The process continues until we find a fault that has a test vector. The reason of using the exhaustive method is that domino circuits are usually small and can afford to be searched in an exhaustive manner. Secondly, our algorithm stops once a test vector which results in the largest possible CS-transistors is found, if our goal is to determine C_{imax} only. In other words, our algorithm does not have to "really" go through all possible cases even though the algorithm may, in the worst case, be exhaustive.

We now use an example to illustrate our exhaustive algorithm. Consider again the example in Fig. 4 whose searching tree is shown in Fig. 4(c). Our exhaustive search starts from the structural worst case by assigning {*a=1, b=1, c=0, d=1, e=1, f=0*} for which, there does not exist a test vector. Since there are four CS-transistors in the structure worst case, we continue to check whether it is possible to have three CS-transistors. There are two different cases shown in the second level of the tree in Fig. 4(c). We first check whether there exists a vector for the assignment {*a=1, b=1, c=0, d=1, e=0*} and then the assignment {*a=1, b=0, d=1, e=1, f=0*}. Again both assignments are not possible. Then, we try to find a test vector for two CS-transistors. Since there indeed exists a test vector for the assignment {*a=1, d=1, b=0, e=0*}, our algorithm stops and we claim that the largest number of CS-transistors is 2.

2.4 Pseudo Gates and ATPG

For finding the largest number of CS-transistors and the corresponding test pattern for a CS fault, we need to find a test vector for the fault. One way is to modify a test generation algorithm but this may require intensive programming work. In our algorithm, we apply a pseudo gate approach which changes the circuit structure so that the traditional ATPG for single stuck-at faults can be utilized for each CS fault. The pseudo gate approach is described as follows. For example, in Fig. 5, suppose we want to find a test vector for the structural worst case. That is to turn on transistors {*Ta, Tb, Td, Te*} and turn off transistors {*Tc, Tf*}. We can modify the circuit as in Fig. 5(c) and use the traditional single stuck-at fault model to obtain a test vector for the CS fault. In Fig. 5(c), for wire $x \rightarrow w$ stuck-at-1 fault, node *x* must have the assignment 0 which can result in two assignments {*y=0, z=0*} in both of its inputs. Because of {*y=0, z=0*}, the assignments of {*a=1, b=1, d=1, e=1, c=0, f=0*} must be satisfied. The conditions to satisfy these assignments are the same as the assignments to turn on transistors {*Ta, Tb, Td, Te*} and to turn off transistors {*Tc, Tf*} for the CS fault.

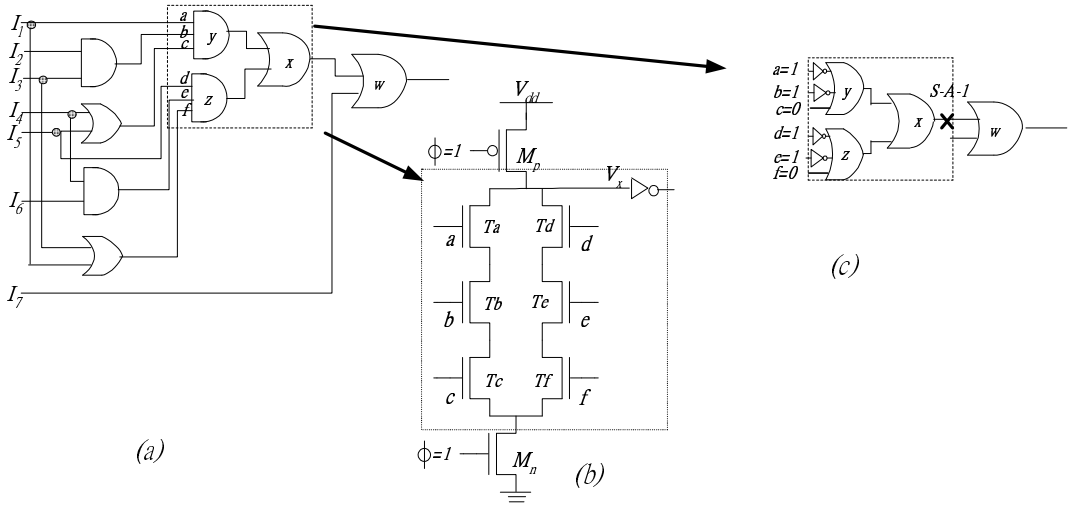

Figure 5. The example of generating a pseudo gate.

2.5 Fault Simulation and Test Length Reduction

We now discuss how the test vector length can be reduced using fault simulation. For each domino gate, one can find a test vector to activate the largest number of CS-transistors. On the other hand, there may be many of such test vectors for a domino gate. It is possible that, to activate the largest number of CS-transistors, several domino gates can share the same test vector. To reduce the number of test vectors, our algorithm makes use of a fault simulator. The algorithm first processes each domino gate in sequence. For a domino gate, if a test vector v has been found, we then perform fault simulation for test vector v. Suppose n_i CS transistors of another domino gate d_i can be activated by test vector v. The number n_i and the test vector v are both stored. Later on, when processing d_i, suppose the maximum number of CS-transistors activated is the same as n_i and the test generation algorithm returns another test vector v_i. Since either one of test vectors v and v_i can activate n_i CS-transistors, rather than using vector v_i, we can reuse vector v. In this way, we can save one test vector.

3. Experimental Results

We have implemented the algorithm in Fig. 6 and tested a set of MCNC benchmark circuits based on a 0.6-μm CMOS process. In our experiment, all N-transistors are assumed to have the same size. For real circuits, each domino gate should have its own C_o and $C_{ithresh}$ ($=V_{dd}*C_o/V_{thresh} - C_o$) values which can be obtained by layout extraction. However, to simplify the experiment, we have made two assumptions. First, the inverter in a domino gate is assumed to have the same size, so we assign the same capacitance C_o for all domino gates. Second, each CS-transistor contributes the same junction capacitance $C_{i\text{-}tr}$ to C_{imax}. That is $C_{imax}= N* C_{i\text{-}tr}$ where N is the number of CS-transistors. In this experiment, we take a five-input domino AND gate as an example to estimate C_o and $C_{i\text{-}tr}$. After the design and layout processes, we have extracted C_o=42.397pf and $C_{i\text{-}tr}$ =6.67pf. By assuming V_{thresh}=1/2 V_{dd}, we can have $C_{ithresh}$= 42.397pf, C_o=42.397pf, and C_{imax}= (the number of CS-transistors)*6.67pf. Based on this data, Table 1 shows our results. The domino circuits shown in Table 1 are obtained from [6]. Column one of the table gives the name of each circuit, column two shows the total number of gates for each original random circuit, and column three shows the number of domino gates for each circuit. Note that without the fault simulation process, the number of test vectors is equal to the number of domino gates. Column four gives the total number of transistors in the structural worst case, and column five gives the total number of CS-transistors. Column six provides the CS-transistors turn-on ratio which is obtained by dividing the value of column five by that of column four. Column seven (eight) presents the lowest (highest) CS-vulnerability value for each circuit. Column nine shows the ratio of vulnerable domino gates which are the gates with CS-vulnerability values larger than one. Finally, column ten gives the number of required test patterns with fault simulation executed.

For example, in circuit ***dalu*** of Table 1, the total number of structurally worse-case transistors is 1373, and the total number of CS-transistors is 1290. Thus, the CS-transistors turn-on ratio is 0.94. The largest CS-vulnerability is 1.259, while the lowest CS-vulnerability is only 0.157. Further, out of 636 gates, there are only one gate whose CS-vulnerability is larger than 1. So, the ratio of vulnerable gates is 0.002. Finally, after test vector reduction, this circuit requires only 183 (instead of 636) vectors to test all CS faults.

From Table 1, by observing the average CS-vulnerabilities, we have found that the CS-transistors turn-on ratios are generally quite large in most circuits. Meanwhile, the CS-vulnerabilities of each circuit vary between the extreme values dramatically. The important implications of such results for circuit designers are two fold. First, the CS problem of domino circuits is very significant (because of the large CS-transistors turn-on ratio), and must be well solved. Second, the large variation of CS-vulnerability values in each circuit indicates that, instead of adding weak pull-up transistors to every domino gate, circuit designers should only deal with the gates with higher CS-vulnerability values. We have also found that fault simulation is very efficient in reducing the number of test patterns. We emphasize that there is no vulnerable gate in some circuits; however, weak pull-up transistors might still need to be added to certain gates with high CS-vulnerability values. For example, circuit ***majority*** does not have any gate with CS-vulnerability greater than one as shown in Table 1. But, one gate has the value equal to 0.944 which is very critical. Thus, it will be safer to add weak pull-up transistors for this circuit. On the contrary, it is not necessary to add weak pull-up circuits for circuit ***parity*** because of the low CS-vulnerability values (0.315). But the CS faults of ***parity*** must be tested since fabrication error might prevent the pull-up transistor from charging the gate output adequately.

4. Conclusion

The CS problem is a notable issue in domino logic design. Although there are methods to alleviate the problem, often, they introduce area and delay overhead. In this paper, we have introduced the concept of CS-vulnerability which can be used to evaluate the sensitivity of the CS problem for a domino gate. A designer can make use of this information to take special cares of a domino gate with high probability to have the CS problem. In addition, we have also proposed an algorithm to compact the number of test vectors needed to test the worst-case situation of the CS problem.

Acknowledgments

We thank M. R. Prassed, and Professor R. K. Brayton in U. C. Berkeley for providing domino benchmark circuits, and G. Z. Wu, H. Y. Lee in Chung-Cheng University for their help on the simulation program and layout capacitance extracted.

References:

[1] R. D. Adams, E. S. Cooley, P. R. Hansen, "Quad DCVS Dynamic Logic Fault Modeling and Testing," *Proc. IEEE International Test Conference*, pp. 14.1, 1998.

[2] K. Bernstein, J. Ellis-Monaghan, and E. Nowak, " High-Speed Design Styles Leverage IBM Technology Prowess," *IBM Micro News*, Vol. 4, Number 3, 1998.

[3] A. Dharchoudhury, D. Blaauw, J. Norton, S. Pullela, J. Dunning, "Transistor-level Sizing and Timing Verification of Domino Circuits in the PowerPC™ Microprocessor," *Proc. IEEE International Conference on Computer Design*, pp. 143-148, 1997.

[4] S. M. Kang, Y. Leblebici, *CMOS Digital Integrated Circuits: Analysis and Design*, McGraw-Hill Book Co., 1996.

[5] R. Puri, A. Bjorksten, T. E. Rosser, "Logic Optimization by Output Phase Assignment in Dynamic Logic Synthesis," *Proc. IEEE International Conference on Computer-Aidded Design*, pp. 2-8, 1996.

[6] M. R. Prasad, D. Kirkpatrick, R. K. Brayton, A. L. Sangiovanni-Vincentelli, "Domino Logic Synthesis and Technology Mapping," *Proc. IEEE/ACM International Workshop on Logic Synthesis*, 1997.

[7] J. A. Pretorius, A. S. Shubat, C. A. Salama, "Charge Redistribution and Noise Margins in Domino CMOS logic," *IEEE Transactions on Circuits and Systems*, Vol. CAS-33, pp. 786-793, Aug. 1986.

[8] N. H. E. Weste, K. Eshraghian, *Principles of CMOS VLSI Design-A Systems Perspective*, 2nd Edition, Addison-Wesley Pub. Co., 1992.

[9] Z. Wang, G. A. Jullien, W. C. Miller, J. Wang, S. S. Bizzan, "Fast Adder Using Enhanced Multiple-Output Domino Logic," *IEEE Transaction on Solid-State Circuits*, Vol. 32, pp. 206-214, Feb. 1997.

[10] Srivastava, P. Pua, A. Weich, Larry, "Issues in the Design of Domino Logic Circuits," *Proc. IEEE Great Lakes Symposium on VLSI*, pp. 109-113, 1998.

[11] P. E. Gronowski, W. J. Bowhill, D. R. Donchin, R. P. Blake-Campos, D. A. Cralson, E. R. Equi, B. J. Loughlin, S. Mehta, R. O. Muller, A. Olesin, D. J. W. Noorlag, R. P. Preston, "A 433-MHz 64-b Quad-Issue RISC Microprocessor," *IEEE Transcation on Solid-State Circuits*, Vol. 31, pp. 1687-1695, Nov. 1996.

[12] P. E. Gronowski, W. J. Bowhill, R. P. Preston, M. K Gowan and R. L. Allmon, "High-Performance Microprocessor Design," *IEEE Transication on Solid-State Circuits*, Vol. 33, pp. 676-686, May 1998.

[13] K. J. Lee, M. A. Breuer, "On the Charging Sharing Problem in CMOS Stuck-Open Fault Testing", *Proc. International Test Conference*, pp. 417-426, 1990.

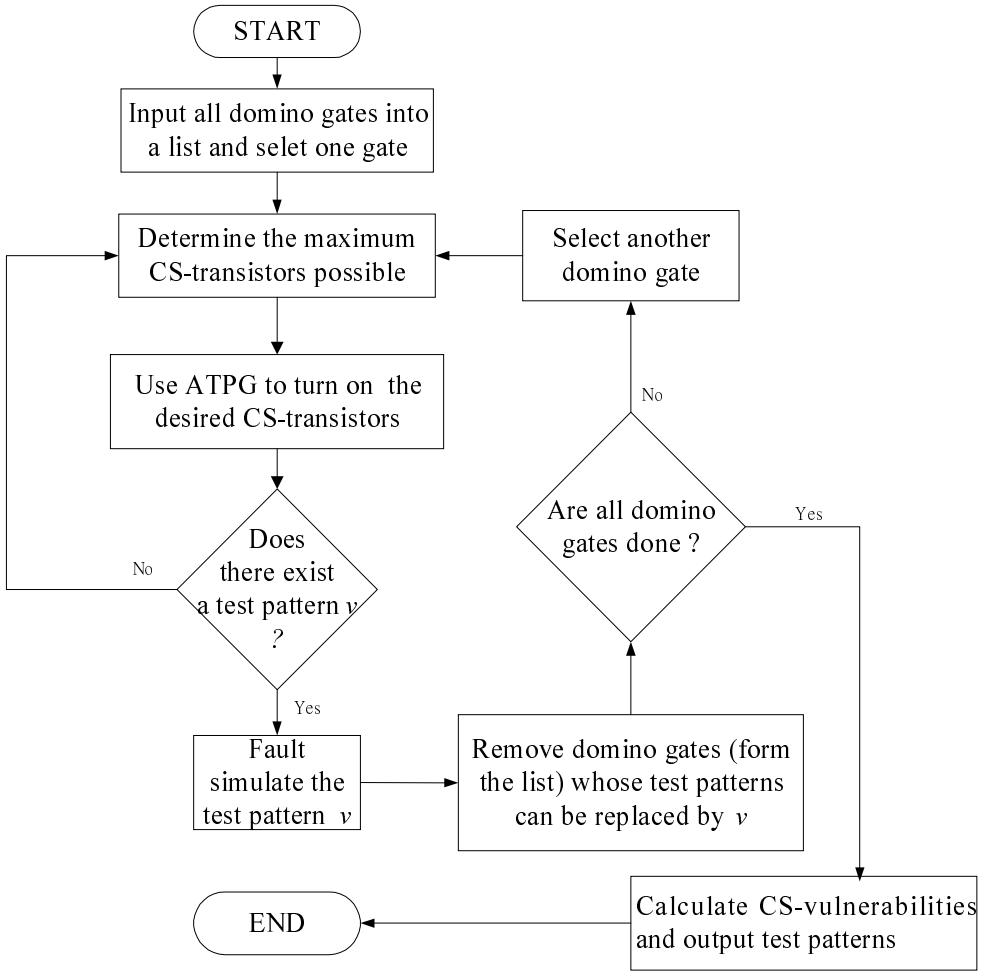

Figure 6. The flowchart of ATPG with fault simulation considered.

Table 1. The simulation results of several benchmark circuits

Benchmark Circuits	Total Gate Count	Domino gate Count	The total # of transistors for structural worst case	The total # of transistors which can be activated.	The CS-transistors turn-on ratio	The lowest CS-vulnerability	The highest CS-vulnerability	The ratio of vulner. gates	# of test vectors after fault simulation
C7552	2832	939	2206	1748	0.792	0	7.080	0.011	Not available
C5315	1860	528	1938	1816	0.937	0	2.832	0.047	Not available
pair	1821	449	1045	962	0.921	0.157	1.573	0.007	Not available
Too_large	167	20	65	61	0.938	0.157	1.416	0.15	Not available
i8	1008	148	1966	1677	0.853	0.315	3.776	0.905	148
dalu	2787	636	1373	1290	0.94	0.157	1.259	0.002	183
frg2	1394	230	1136	1066	0.938	0.157	2.36	0.191	144
C499	884	186	400	396	0.99	0.315	1.573	0.011	74
mux	31	5	51	36	0.706	0.944	1.888	0.2	5
C1908	1570	247	557	546	0.98	0.157	4.877	0.028	137
C1355	900	338	600	596	0.993	0.157	1.573	0.006	228
parity	87	29	58	58	1	0.315	0.315	0	14
majority	5	1	8	6	0.75	0.944	0.944	0	1
count	126	32	64	64	1	0.315	0.315	0	26
cordic	100	20	57	56	0.982	0.157	1.259	0.05	14
cm85a	40	12	16	16	1	0.157	0.315	0	9
cm150a	78	15	30	30	1	0.315	0.315	0	8
c8	97	24	12	12	1	0.315	0.315	0	12
b9	142	33	60	51	0.85	0.157	0.787	0	23
C17	6	2	4	3	0.75	0.157	0.315	0	2
i6	344	67	557	489	0.878	0.787	1.146	0.582	67
C880	525	140	255	254	0.996	0.157	1.101	0.014	77
C432	575	81	478	478	1	0.157	3.776	0.605	30