On the development of a fast and accurate bridging fault simulator

Research Report
ISSN 0167-9708
Coden: TEUEDE

Eindhoven
University of Technology
Netherlands

Faculty of Electrical Engineering

On the Development of a Fast and Accurate Bridging Fault Simulator

by
Chennian Di
Jochen A.G. Jess

EUT Report 94-E-281
ISBN 90-6144-281-8
January 1994

CIP–DATA KONINKLIJK BIBLIOTHEEK, DEN HAAG

Di, Chennian

On the development of a fast and accurate bridging fault simulator / by Chennian Di, Jochen A.G. Jess. – Eindhoven : Eindhoven University of Technology, Faculty of Electrical Engineering. – Fig., tab. – (EUT report, ISSN 0167–9708 ; 94–E–281)
With ref.
ISBN 90–6144–281–8
NUGI 832
Subject headings: integrated circuits ; fault modeling / integrated circuits ; fault simulation.

On the Development of a Fast and Accurate Bridging Fault Simulator

Chennian Di and Jochen A. G. Jess

Abstract

This paper presents an alternative modeling and simulation method for CMOS bridging faults. The significance of the method is the introduction of a set of **unique–bridge table**s which characterize the bridged outputs for each bridge and a set of **unique–cell table**s which characterize how each cell interprets an analog input. These two sets of tables are derived dynamically for a specific design by using a SPICE circuit simulator first. Then they can later be used by any logic fault simulator to simulate bridging faults. In this way, the proposed method can perform very fast bridging fault simulation and yet with SPICE accuracy. The paper shows how these two sets of tables are derived and used in a parallel pattern fault simulator. Experimental results on ISCAS85 benchmarks are promising.

keywords: defect, bridging fault, fault modeling, testing, fault simulation

Addresss of the authors:
Section of Design of Electronic Systems(ES)
Faculty of Electrical Engineering
Eindhoven University of Technology
P.O. Box 513, 5600 MB, Eindhoven, The Netherlands
Tels. +31–(0)40–473373, +31–(0)40–473710
Fax +31–(0)40–464527
Email di@es.ele.tue.nl or jess@es.ele.tue.nl

Contents

1. Motivation

In the last decade, the gap between the so–called realistic faults caused by manufacturing defects and practically used Single Stuck–At (SSA) faults has been emphasized strongly [1,2,5,8,9,10] for CMOS Integrated Circuits (IC). It becomes evident that accurate modeling and efficient simulation of defect induced faults are essential for high quality testing of ICs. Particularly the bridging faults, one of the most frequently occurring faults, attract a lot of attention.

The complexity of modeling and simulating bridging faults has been very well studied. It is widely known that one of the difficulties of modeling a bridging fault is with the conducting circuitry created from power supply to ground since it changes a digital circuit into an analog one. To illustrate, Fig.1 shows a zero–ohm bridge between the outputs of a complex cell and a 2–in–NAND. Its bridged output obtained by SPICE is listed in Table 1. It is seen that the output may vary from 0.63V to 4.71V for different inputs. These values cannot easily be taken as logic "1" or "0" since it depends on how they drive the following cells. Fig.2 shows a situation in which the same bridged output value caused by the bridge in Fig.1 can drive one 2–in–NAND to 0.64V which can be interpreted as "0" and drives another structurally equivalent 2–in–NAND to 3.99V which can be interpreted as "1". Thus any simple model, such as wired–and or wired–or, is not sufficient here. It is obvious that the behavior of a bridge can only be accurately modeled if the following two issues can be solved efficiently:

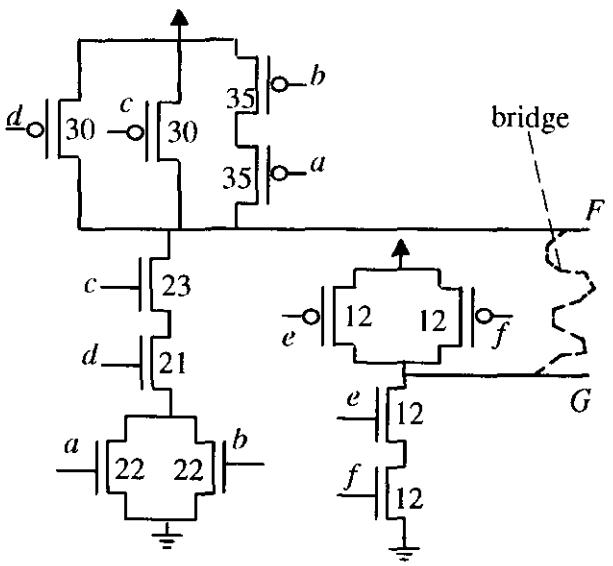

Figure.1 An example of a bridge.

1) the accurate evaluation of the bridged output voltage;
2) the accurate propagation or interpretation of an analog input caused by a bridge to decide if the bridge is detected.

Obviously a circuit simulator, such as SPICE, can accurately fulfil the tasks. However, for large circuits, this seems computationally intolerable .

Table.1 Bridged output obtained by SPICE

inputs		outputs	
a b c d	*e f*	*F G*	*bridged output*(V)
1 1 1 1	0 1	0 1	0.63
1 0 1 1	1 0	1 0	1.42
0 0 1 1	1 1	1 0	2.15
1 1 1 1	0 0	1 0	2.47
0 1 0 1	1 1	1 0	3.35
0 0 0 0	1 1	1 0	4.71

Many works have been done in order to solve this problem efficiently. Early solutions using a switch level model [3,4,15] have been widely recognized as inadequate for modeling and simulating CMOS bridging faults.

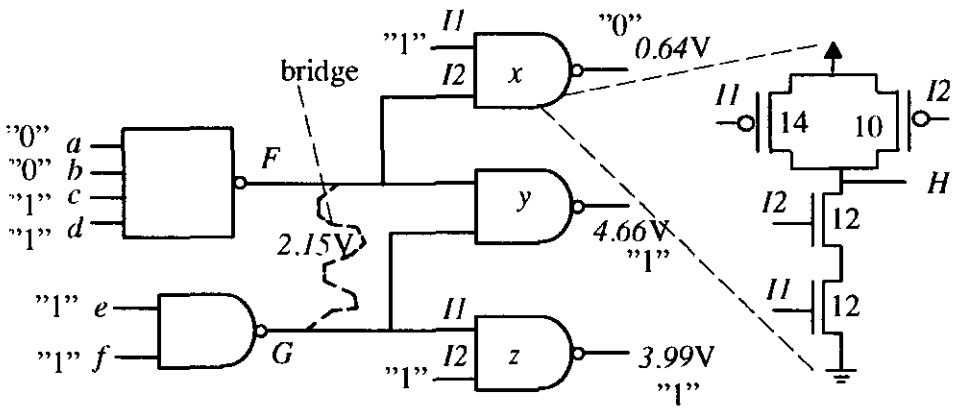

Figure.2 Impact of an analog input on fanout cells.

Many suggested methods attempt to improve the modeling accuracy by using an approximate model, such as the resistive network model [12] or voting model [6,13,14,20]. These methods allow very fast logic fault simulation but have a drawback that only the bridged outputs are analyzed without carefully considering how the fault propagates. An incorrect modeling of using such a method is illustrated by Fig.2. For the inputs in Fig.2 causing a conducting circuitry, the pull–down conductance is stronger than the pull–up conductance. The voting model would predict the bridged output as "0" but, in fact, it can be "1" or "0" depending on its fanout cells as shown in Fig.2.

Recently a more accurate method using a mixed or multi–level simulation techniques is proposed in [16,18]. This method switches from normal logic simulation to circuit–level simulation whenever a bridging fault is encountered. The bridge is simulated along its fanout cones until the analog signals can be interpreted as digital ones. Then the simulation is switched back. Such a method is very accurate. However it may not be very efficient to simulate a large circuit for lengthy test patterns. For instance, for a simulated bridge that is not yet detected, if two input patterns cause the

same conducting circuitry configuration for this bridge, e.g., in Fig.1 the inputs *abcdef* = 100111 and *abcdef*=110111 cause the same conducting circuitry, this method would invoke the expensive circuit simulator twice which is unnecessary. It is also not efficient to evaluate all the bridges that connect two cells having the same combination of cell types, such as a 2–in–NOR to a 3–in–NAND. Very recent improvements [21] use the pre–computed tables derived by a circuit simulator to avoid some unnecessary computations and use logic threshold to interpret an analog input. However, the pre–computed tables that are derived by enumerating all the possibilities from a given cell library may be both time and memory consuming and cannot be used efficiently for a specific design. Furthermore the multi–input threshold effects are not considered. For instance, assume the bridged output is 2.47V in Fig.2. Since two inputs of *y* are bridged together and have a threshold of 2.6V, the output of *y* is 4.99V. But if the single input thresholds, 1.89V and 2.20V, are used, then a "0" would be predicted which is not correct. Very recent improvement of the voting model [17, 22] is unfortunately still approximate in nature and does not avoid the same shortcoming of not considering multi–input thresholds.

Above the modeling problem of bridging faults and the shortcomings of other methods are reviewed. Following, based on our very recent development in [23], a more accurate modeling and yet very fast fault simulation alternative is proposed. Section 2 outlines the general strategy of the proposed method. Section 3 and 4 discuss some implementation issues. Experimental results are given in Section 5.

2. Fault simulation using unique–bridge table and unique–cell table

This paper concentrates on CMOS combinational circuits. A CMOS circuit can be viewed as an interconnection of CMOS cells. A CMOS **cell** has a network of serial–parallel PMOS transistors as pull–up and, its dual part, the pull–down part. A cell in a specific design is **unique** if both the Boolean function it realizes and its structure are unique. For example, though two inverters with different transistor sizes have the same function, each of them is considered unique. The bridging faults analyzed are non–feedback bridges between outputs of two cells. The resistance of the bridge is assumed to be negligible. Furthermore only static analysis is performed.

As it can be seen from the example, few centivolts difference of the input voltage can cause different outputs; thus any approximate method would easily lead to wrong decisions. To guarantee correct simulation, circuit–level accuracy must be obtained. In order to obtain both high accuracy and efficiency, let us review some design aspects. In modern CMOS designs, it is common practice that most designs are based on a given cell library. In a specific design, the number of cells is usually much larger than the size of the given cell library. It is very likely that several bridges connecting two cells have the same combination of cell types. These bridges can be represented by one bridge, called the **unique–bridge**. Formally a **unique–bridge** is a bridge that connects two cells having the unique–cell type combination. This observation can help to simplify the evaluation task since there is no need to evaluate all bridges but only the unique one. To keep the evaluated bridged

outputs for a bridge, a **unique–bridge table** is introduced for each unique–bridge. It consists of a set of pairs <b, d> in which b is the bridged output voltage value and d represents input conditions. The input conditions are represented symbolically in terms of the inputs of the unique–cell. If the input condition is satisfied, one cell will drive the bridged output to power supply and another cell will drive it to ground and the bridged output has value b. Fig.3 shows one unique–bridge table for the bridge shown in Fig.1. In fact, such a table can be viewed as a multiplex function

$$F_{bri} = b_1 \cdot d_1 + ... + b_n \cdot d_n$$

Each condition d_i is a Boolean term from the Boolean input space of the pair of two bridged cells. If d_i is satisfied, the F_{bri} takes a voltage value b_i. The unique–bridge table can be derived by a circuit simulator, in our case, SPICE. Thus the bridged output is characterized with the accuracy of SPICE. Usually the number of unique–bridges is far smaller than the number of possible bridges extracted from circuit layout. Therefore the computation task is relatively limited.

$$\begin{aligned}
&< \ \mathbf{0.00V}, \ (e \oplus f) \cdot a \cdot b \cdot c \cdot d \ > \\
&< \ 1.42V, \ (e \oplus f) \cdot (a \oplus b) \cdot c \cdot d \ > \\
&< \ 2.15V, \ e \cdot f \cdot \overline{a} \cdot \overline{b} \cdot c \cdot d \ > \\
&< \ 2.45V, \ \overline{e} \cdot \overline{f} \cdot a \cdot b \cdot c \cdot d \ > \\
&< \ 2.89V, \ \overline{e} \cdot \overline{f} \cdot (a \oplus b) \cdot c \cdot d \ > \\
&< \ 3.35V, \ e \cdot f \cdot (a + b) \cdot (c \oplus d) \ > \\
&< \ \mathbf{5.00V}, \ e \cdot f \cdot (\overline{a} \cdot \overline{b} \cdot \overline{c \cdot d} + \overline{c} \cdot \overline{d}) \ >
\end{aligned}$$

Figure.3 An example of a unique–bridge table.

Now let us review how a CMOS cell transfers an input voltage. It is known that CMOS cells have a very high gain around the logic threshold voltage of each input. A small variation at the input would yield a very big swing at the output. It is very likely that an input voltage lower than the input logic threshold would cause an output large enough to be a logic "1" and vice versa. This implies that most of the bridged analog voltages can be interpreted as logic levels just by propagating them one level up along their fanout cones. That is, without any computation, a bridged output can be predicted as a logic value by just comparing its voltage value with the logic threshold of the respective fanout cell input. Obviously there exist cases that a bridged output may have a value equal to (or very close to) the threshold voltage of an input it drives. In this situation, the output of the fanout cell may still be in the unknown domain and cannot be interpreted as a logic value at this stage. The bridged output should be propagated further before it can be interpreted as a logic value. However, more computations are needed. In our experiments on the ISCAS85 benchmarks, such situations hardly occur and count only up to about 0.04% of all interpretations during fault simulation. To balance more for efficiency, the principle of comparing the bridged output just with the thresholds of the immediate fanout cells (i.e., only one level up) is employed:

If the input voltage is higher than the threshold, a logic "0" would be interpreted at the output. Vice versa, if the input is lower than the threshold, a logic "1" would be interpreted. Otherwise the fault free value is assumed at the output to avoid the uncertainty.

Using the above principle, the advantage is obvious since only the threshold voltages of each unique–cell are needed and they can be pre–computed. In the course of the fault simulations, no more circuit level computations are needed. For the derivation of threshold voltages, as it is known that different input ports of each cell may have different thresholds, a **unique–cell table** is introduced to characterize the transfer characterization of each input of a unique–cell. For each input of a cell, an observable condition is defined. It is a Boolean expression in terms of other inputs of the cell. If the observable condition is satisfied, the input becomes observable at the output, that is, if the input changes, the output also changes. Similar to the unique–bridge table, for each input of a unique–cell, the unique–cell table consists of a set of pairs $<w, O>$ of which w is the logic threshold voltage value of that input and O denotes its observable condition. They are also represented symbolically in terms of the cell inputs. Fig.4 shows a unique–cell table for the 2–in–NAND in Fig.2. If $I1=1$, $I2$ is observable at the output, i.e., if $I2$ is "1", then H is "0" and vice versa. The last entry indicates that if both $I1$ and $I2$ are connected together, the threshold is 2.6V. Obviously the observable condition is always true in this case. The unique–cell table can be accurately derived by using a circuit simulator.

Input $I1$: < 1.89V, $I2$ >
Input $I2$: < 2.20V, $I1$ >
Input $I1\&I2$: < 2.60V, *TRUE* >

Figure.4 An example of a unique–cell table.

With the introduction of the two concepts, i.e. unique–bridge table and unique–cell table, the fault simulation works as follows after the fault free simulation:

1) For each bridge, find its respective unique–bridge table. Evaluate the input conditions according to the current input pattern to check which one is satisfied and obtain the voltage value. Then,
2) for each fanout cell from the bridged site, find its unique–cell table. Compare the bridged voltage value with the respective input threshold and interpret it as a logic value at the output of the fanout cell.
3) After all the fanout cells are processed and if there exist faulty values at the fanout cells, start the normal logic fault simulation until the bridge is detected.

Since these two sets of tables are derived in advance, during fault simulations, there is no expensive circuit simulation involved. Thus the fault simulations can be done solely at logic level with only some cost of the above interpretation procedure. Consequently both high accuracy and efficiency

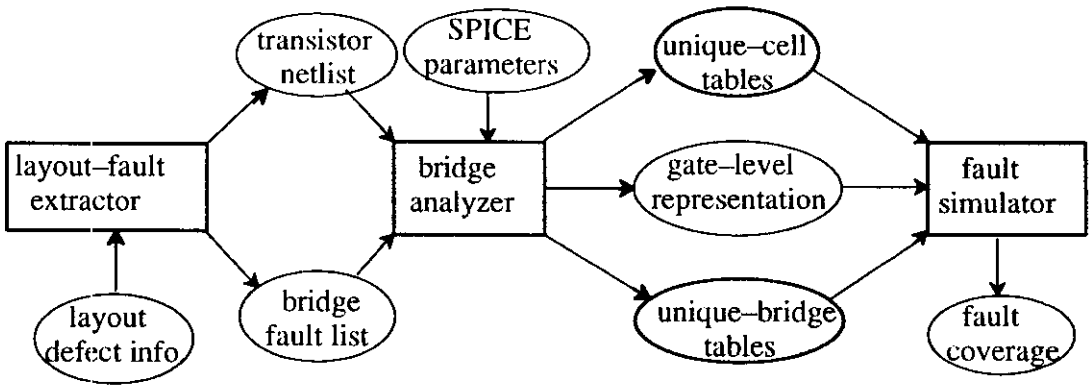

Figure.5 System overview.

are obtained. Fig.5 illustrates the overview of the whole system. Below some implementation details are discussed.

3. Dynamic derivation of unique–bridge table and unique–cell table

The derivation of the unique–bridge tables and unique–cell tables is performed by analyzing the extracted bridging faults for a specific design instead of enumerating the given cell library like other methods do. Thus the derivation is dynamic. This is because the following reasons:

1) Although the size of a given cell library can be large, the number of unique–cells in a specific design is relatively small. This indicates that the number of all possible combinations of two different cells for a design is small. Consequently the number of possible unique–bridges is small. Thus, the task of characterizing both tables for a design is easier.
2) The occurrence of bridging faults highly depends on the layout topology of a specific design. It is very likely that not every possible unique–bridge table derived by enumeration may actually occur. Such information can only be obtained by analyzing the extracted bridging faults for a specific design.
3) The number of logic thresholds of all possible *multi–inputs* (more than two inputs being connected together as illustrated in Fig.2) for a set of cells is usually very large. The actual number of multi–input situations depends on how many bridges actually connect more than two inputs of a cell and how a cell is used in a design. Again such information can only be obtained by analyzing the extracted bridging faults for a specific design.

Since both speed and memory are crucial for the simulation, the derivation is performed for each design by analyzing the extracted bridges. In this way, for each unique–bridge table, it is guaranteed that the respective bridge actually occurs in the design. Thus the amount of circuit simulations can be greatly reduced compared to purely enumerating the cell library. Such derived tables can also easily make use of some efficient techniques, such as parallel pattern simulation as will be shown.

The inputs of the *bridge analyzer* (Fig.5) are a plain representation of the transistor netlists and all possible bridging faults. Both are extracted from the layout of a design first. The SPICE simula-

tor is chosen for the derivation. Thus SPICE parameters for a specific process are also taken as an input.

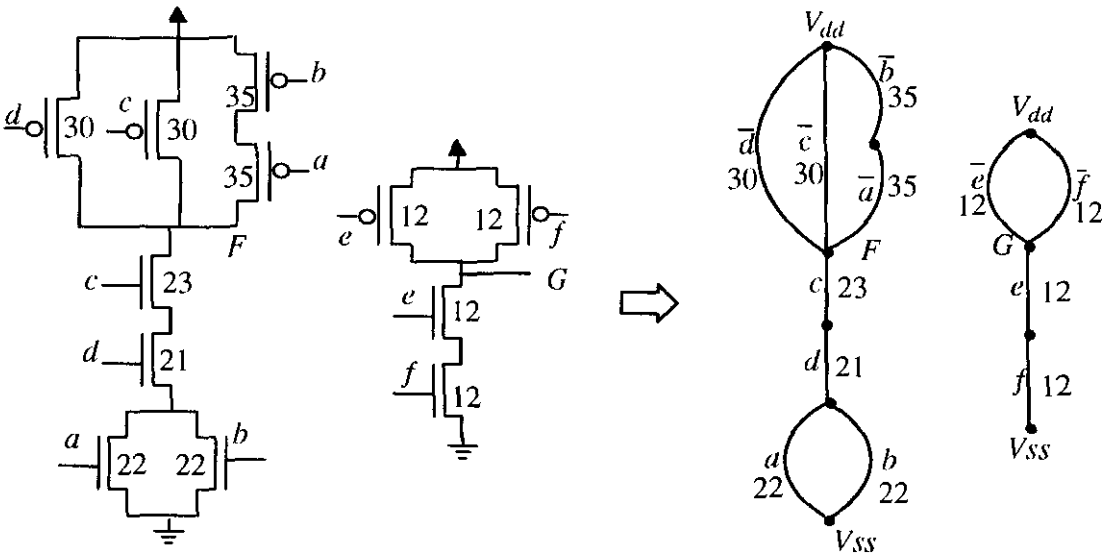

Figure.6 Illustration of a graph representation of a CMOS circuit.

The first step is the extraction of all the cells. The CMOS circuit can be represented by a connection graph $G(V, E)$. Each node $v \in V$ represents a network node which can be the drain, source or gate of a transistor. An undirected edge $e \in E$ represents a transistor and has an associated Boolean variable (defined by its gate input function) and a weight representing its transistor size. An example of this graph representation is shown in Fig.6. The Boolean function of a cell can be easily extracted by searching the pull–up and pull–down paths in the cell. Then the unique–cells are identified. Since each cell is represented by a graph, this step can be done by checking the isomorphism of two graphs. After all the cells are checked, all unique–cells in a design are obtained.

3.1. Unique–bridge table

Each bridge from the extracted fault list is first identified as "feedback" or "non–feedback". For each "non–feedback" bridge, its respective unique–bridge table is characterized if not yet derived. Then the two unique–cells involved by this bridge are identified. Afterwards for each input combination of these two cells that drives one cell "on" and another cell "off", its respective SPICE format description is generated in an input file. After all possible input combinations creating conducting circuitry from power supply to ground are enumerated, a SPICE call is invoked to compute the bridged output voltage. The results are collected to construct the table. The major cost of this procedure is the execution of SPICE. To speed up, the following techniques are used.

The first technique makes use of the fact that if the bridged output voltage is very close to the potential of power supply or ground, it can be certainly interpreted as a logic value. For example, for a typical 5V CMOS technology, an input above 4V, which is the lowest "hard" logic "1" value V^1_{hard}, or below 1V, which is the highest "hard" logic "0" V^0_{hard}, can definitely be interpreted as

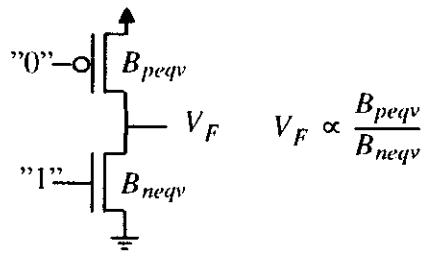

Figure.7 Equivalent conducting circuitry

"1" or "0" respectively. If there is a method that without running SPICE simulation the voltage can be predicted with sufficient accuracy in this range, then the expensive circuit simulations can be avoided. Here we use an estimation method developed in [18] to predict the voltage range. This method uses a simplified transistor model to estimate the voltage. Using this model, an NMOS transistor either works in a linear region as in eq.(1) or is off.

$$I_{ds} = K\ B\ (V_{gs} - V_t - \frac{1}{2}V_{ds})V_{ds} \tag{1}$$

K is process dependent. V_t is the zero–bias transistor threshold. V labels the voltage and I the current. The subscripts g, d, s indicate the gate, drain and source of a transistor. B is the transistor width–to–length ratio. Using this model, any conducting circuitry can be simplified to the one shown in Fig.7 with B_{neqv} and B_{peqv} as equivalent transistor sizes of pull–down and pull–up parts respectively. The output V_F can be derived by solving the following equation,

$$(\beta - 1)V_F^2 + 2(V_{dd} - V_{tn} - \beta V_{tp})V_F - \beta(V_{dd} - 2V_{tp})V_{dd} = 0$$

where $\beta = B_{peqv}/B_{neqv}$. Obviously V_F is an increasing function of β. Two ratios β^1_{hard} and β^0_{hard} corresponding to V^1_{hard} and V^0_{hard} can be found such that the following relations hold:

$$\beta > \beta^1_{hard} \Rightarrow V_F > V^1_{hard} \text{ and } \beta < \beta^0_{hard} \Rightarrow V_F < V^0_{hard}$$

This means that, for a specific technology, it is not even necessary to solve all equations. The estimation is simply a comparison of the equivalent ratio with β^1_{hard} and β^0_{hard}. The computation of the equivalent ratio is linear to the number of transistors in a cell and very fast. This estimation method appears to be accurate enough. More details can be found in [18].

The second reduction technique is based on structure equivalence. For a bridge, many conducting circuitries activated by different combination of input excitations, in fact, have the same structure configuration. Consequently the bridged output for these different input excitations is the same. These conducting circuitries are said to be structurally equivalent. For instance, with the bridge in Fig.1, the four different input combinations shown in Fig.8(a) cause the same conducting circuitry as shown in Fig.8(b) with the bridged output being 1.42V. For those structurally equivalent conducting circuitries, there is no need to repeat the same SPICE simulation. In the course of the analysis, all the processed conducting circuitries, each of which is in fact a graph, are kept in a list.

	a b c d e f	bridged output
input1	0 1 1 1 0 1	1.42V
input2	0 1 1 1 1 0	1.42V
input3	1 0 1 1 0 1	1.42V
input4	1 0 1 1 1 0	1.42V

(a)

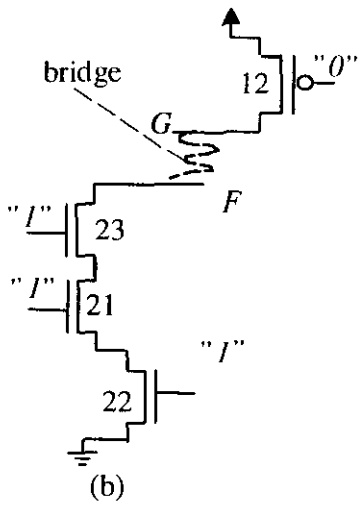

Figure.8 Illustration of structural equivalence.

If a new conducting circuitry is found equivalent to one already in the processed list (their graph representations are isomorphic), its SPICE format description is not generated. Only its input condition is merged with the one in the processed list as shown in Fig.3 for the example shown in Fig.8.

It will be shown by experimental results that above two techniques are very effective.

3.2. Unique–cell table

After all unique–cells are identified, the logic thresholds of each single–input of a unique–cell can be derived. For an inverter, the logic threshold voltage is the input voltage value that makes the output have the same voltage value. It is well–known that for other types of cells, functionally equivalent inputs may have different thresholds as shown in Fig.4. However it seems that not enough emphasis has been given to another fact. That is, for more complex cells, even one single input may have several threshold voltages depending on how the cell is driven. Fig.9(a) shows an example in which the input *a* has three different thresholds. Their values are listed in Table.2. The impact of *single input multi–threshold* effects is significant and can be exemplified if the input *a* is assumed to be 2.15V. The same input in Fig.9(a) can be interpreted as unknown (in case the threshold is 2.15V), "1" (in case the threshold is 2.17V) and "0" (in case the threshold is 2.08V) at the output "*F*" depending on how the cell is driven. It is observed in experiments that for some complex cells, up to 7 different thresholds for a single input can be derived. Thus the multi–threshold effects cannot be ignored. In this paper, all possible configurations in a cell which may result in a different threshold voltage for a specific input are enumerated. Similarly for each possible configuration, the respective SPICE format description is generated in an input file. Then SPICE is invoked to determine the threshold voltage and the results are collected to construct the unique–cell table. This procedure is repeated for each input of every unique–cell.

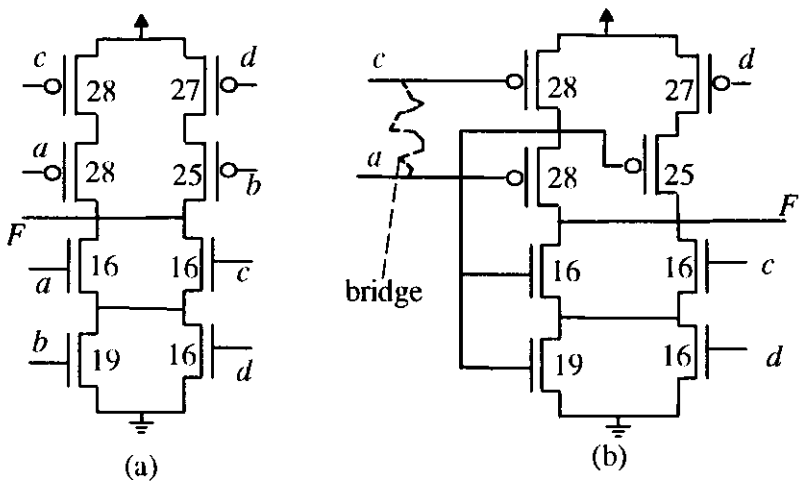

Figure.9 (a) A complex cell. (b) Illustration of multi–input thresholds.

Table.2 Multi–thresholds

input a		input a & b		input a & b & c	
threshold(V)	b c d	threshold(V)	c d	threshold(V)	d
2.08	1 0 1	1.89	1 0	1.58	1
2.15	1 0 0	2.11	0 1	2.44	0
2.17	1 1 0	2.62	0 0		

There are several situations making it necessary to derive multi–input thresholds. Fig.9(b) shows a possible use of the cell in Fig.9(a) in an actual design. It can be seen that even a single input may drive two inputs (a and b in the original cell) of a cell at the same time. Assume that a bridge between a and c in Fig.8(b) occurs, then three inputs (a, b and c in the original cell) are connected together. Table.2 also lists the threshold values for these two situations. It can be seen that the same input can result in completely different outputs. Their impacts on logic modeling are also obvious. Thus it is necessary to know the multi–input thresholds in order to interpret the input correctly. As discussed that it is not efficient to enumerate all possibilities for a cell, instead, they are derived by analyzing the actual design and the extracted bridges. For each bridge, every fanout cell from the bridged site is checked. If more than two inputs in one cell are bridged or one input drives more than two inputs and their thresholds are not derived yet, then all possible configurations that may result in different threshold voltages for these inputs are enumerated and their thresholds are characterized in a similar way as described above. It should be noted that for multi–input situations, there may also be multi–thresholds as it can be seen in Table 2. They are all derived by our method. This procedure is repeated for every bridge. Eventually all multi–input thresholds which are really necessary to simulate bridges for an actual design are obtained in the tables.

The unique–cell table derived in this paper is complete since not only the multi–threshold voltages for single input but also the multi–input multi–threshold voltages are derived. Furthermore the number of multi–input situations remains minimal due to our dynamic analysis feature.

3.3. Boolean function representations

During the analysis and the derivation of these two sets of tables, the Boolean function of each unique–cell and each table entry involves Boolean manipulations and the results need to be stored for the simulations. This does not seems an important issue. However if this is not properly handled, it may cost a lot of memory. Here the Reduced Ordered Binary Decision Diagram (ROBDD) [11] data structures are used. Due to the *strong canonical* representation feature of ROBDD, the amount of memory required to store both unique–bridge tables and unique–cell tables are small. This can be explained by Fig.10. Assume a bridge between unique–cells *c1* and *c2*. To construct its unique–bridge table, it is necessary to construct each pull–up Boolean term of *c1* and each pull–down Boolean term of *c2* and vice versa. Now assume another unique bridge between *c1* and *c3*. Since all the pull–up and pull–down terms of *c1* are already constructed, there is no need to construct them again during the construction of the unique–bridge table for the bridge between *c1* and *c3*. In this way, each unique cell is only processed once. Thus the eventual space is small.

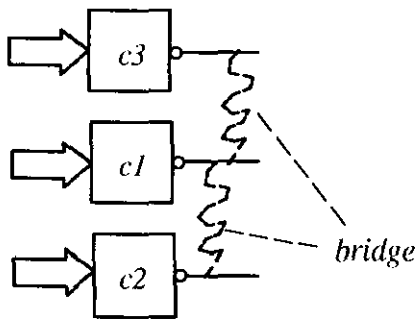

Figure.10 Illustration of small memory usage.

4. Fault simulation

The two sets of tables derived above are now applied for fast fault simulations. The fault simulation works essentially like any other logic fault simulator. Thus any efficient technique can be applied. To show the advantage of using the two sets of tables, the well–known parallel pattern and single fault propagation (PPSFP) [7] technique is used. In the following, we derive the evaluation and interpretation procedure symbolically in order to show the parallel techniques.

Before the details are explained, a corollary which will be used for the interpretation of a bridge is presented. Assume a cell realizing a Boolean function F is one of the fanout cells from the bridged site as shown in Fig.11(a). The output of *cell3* is now a function of both A and B. Following the modeling strategy developed in Section 2, the new input space ($|A + B|$) as shown in Fig.11(b) can be partitioned into a part Y of which the inputs would not activate the bridging fault (no conducting circuitry) and another part X in which the inputs activate the bridge (a conducting circuitry exists from V_{dd} to V_{ss}). The second part X can be further partitioned as one part X^1 in which the inputs would cause a **faulty "on"** behavior, that is, the fault free output is "0" but faulty one is

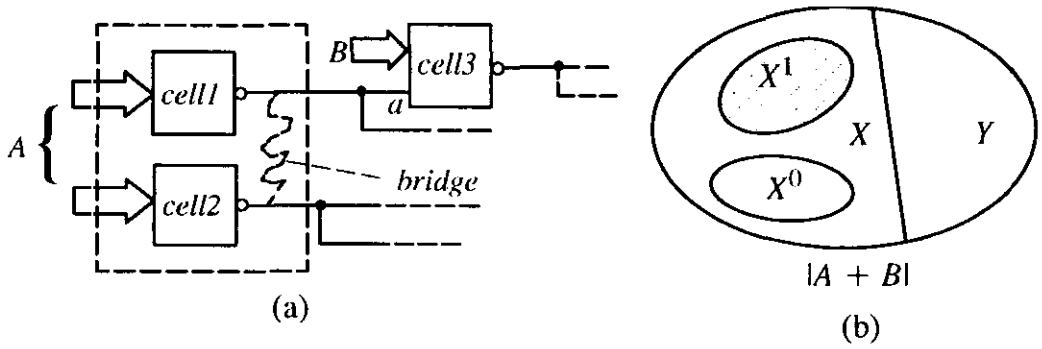

Figure.11 Illustration of a bridge interpretation procedure.

"1" at the output of *cell3*, a part X^0 in which the inputs cause a **faulty "off"** behavior, that is, the fault free output is "1" but faulty one is "0" and the rest part of X in which no faulty value is induced. Then the following corollary holds.

Corollary: Assume a cell takes a bridged output voltage as an input and its fault free function is F. Let f^1 and f^0 be all the faulty "on" and all the faulty "off" Boolean expressions of the cell respectively, that is, if any input satisfies f^1, a faulty "on" behavior is induced at output and vice versa if f^0 is satisfied, a faulty "off" behavior is induced at output, then the Boolean function of the cell in case of the bridge is specified as:

$$F_f = F \cdot \overline{f^0} + f^1 \tag{2}$$

Proof: The example in Fig.11 is used for illustration. According to the definitions, we know $f^1 \Rightarrow F_f$ and $f^0 \Rightarrow \overline{F_f}$ are true. Regarding to the input space in Fig.11(b), f^1 is a function mapping all inputs in X^1 and f^0 is a function mapping all inputs in X^0. Obviously $f^1 \cdot f^0 = 0$ is true since no input can cause a faulty "on" and "off" at the same time. The output behavior of other inputs in $X - X^1 - X^0$ and Y is characterized by F. Therefore the faulty behavior of the cell can be specified as $F_f = F \cdot \overline{f^1} \cdot \overline{f^0} + f^1$. Thus eq.(2) holds.

This formula will be used to evaluate and interpret the bridges in the course of the simulation. First of all, the fault free simulation is executed for a given pattern or a set of patterns in a bit vector manner. Then for each bridge, the respective unique–bridge table and the input order matched to the bridged local cells are found. Afterwards, the interpretation of the bridged value is conducted for each of the fanout cells. For illustration, the same example in Fig.11 is used. Assume that it has a set of thresholds $T_{cell}(a) = \{(w_1, O_1), (w_2, O_2), \ldots, (w_n, O_n)\}$. The respective control conditions are expressed in local variables in the design. The unique–bridge table of the bridge is in a set T_{bri} and is arranged in two parts. The first part contains entries for which the output of *cell1* is "0" and the output of *cell2* is "1" if their input conditions are satisfied and the second part contains the entries with the opposite behavior. For any entry in the first part of the table, it is only necessary to check if the bridged voltage is higher than the threshold voltage of input a. This is because for the

inverted cells, if the observable condition is satisfied, any input lower than the threshold would be interpreted as a logic "1" at the output of *cell3*. It is known that the fault free value of *cell1* would be "0" resulting in "1" at the output of *cell3*. There is no faulty value caused at *cell3*. Let $T^1_{bri}(w)$ contain all the ones for which their bridged output is higher than a value w. Then $T^1_{bri}(w) = \{(b,d) \in T_{bri} | \ b > w\}$ where w can be a value from $T_{cell}(a)$. The respective input condition of $T^1_{bri}(w)$ can be expressed as,

$$C^1(w) = \sum_{(b,d) \in T^1_{bri}(w)} d \tag{3}$$

Eq.(3) means that if $C^1(w)$ is satisfied, then the bridged output would be higher than a voltage value w. Following the similar reasoning, for any entry in the second part of the table, it is only necessary to check if the bridged voltage is lower than the threshold voltage of input a. Then after the checking, all entries of which their bridged output is lower than a value w is obtained in $T^0_{bri}(w)$. Obviously $T^0_{bri}(w) = \{(b,d) \in T_{bri} | \ b < w\}$ and its respective input condition is

$$C^0(w) = \sum_{(b,d) \in T^0_{bri}(w)} d \tag{4}$$

If $C^0(w)$ is satisfied, the bridged output voltage would be lower than w. If eq.(3) and eq.(4) are evaluated for all thresholds, then the following formulas are obtained:

$$f^1 = \sum_{(w,O) \in T_{cell}(a)} C^0(w) \cdot O \tag{5}$$

$$f^0 = \sum_{(w,O) \in T_{cell}(a)} C^1(w) \cdot O \tag{6}$$

Eq.(5) and eq.(6) mean that if any input satisfies eq.(5), then the output of *cell3* would have a faulty value "0" and vice versa any input satisfying eq.(6) introduces a faulty "1" at the output of *cell3*. Therefore, according to eq.(2), the faulty behavior of *cell3* is characterized as

$$F_f = F \cdot \overline{f^0} + f^1 \tag{7}$$

Eq.(7) can be evaluated according to the current input patterns. If the value of F_f is indeed different from F, the output node of *cell3* is inserted in an event list. After all the fanout cells are processed, the logic fault simulation can be started from the fanout cells.

For the case of more than two inputs bridged together, the interpretation procedure is very similar. It is not difficult to observe that each formula can be evaluated in a bit vector manner. Thus the whole procedure can be done for parallel patterns.

5. Experimental results

The entire system is implemented in C on a HP–9000/755 workstation. For experiments, the ISCAS85 benchmark circuits are used. They are implemented in a standard cell design approach

for a 2μ CMOS technology at MCNC (Microelectronics Center of North Carolina). The cell library consists of both simple (such as NAND and NOR) and complex (such as AOI and OAI) cells. The bridging faults are extracted from the circuit layout and supplied by Jee [14]. For the SPICE simulator, the level 3 MOS model is used for the analysis.

Table.3 Some circuit statistics

circuit	c432	c499	c880	c1355	c1908	c2670	c3540	c5315	c6288	c7552
#input	36	41	60	41	33	233	50	178	32	207
#output	7	32	26	32	25	140	22	123	32	108
#transistors	728	1396	1164	1768	2058	2974	4122	6734	8464	8854
#total cells	152	284	236	366	411	604	791	1288	1848	1795
#bridge	1546	2747	3227	4356	4669	13589	16316	40143	21475	53439
#non-feedback	894	2145	2890	3130	3601	12604	13631	39739	13014	49640

Table.4 Results of bridge analysis

circuit	c432	c499	c880	c1355	c1908	c2670	c3540	c5315	c6288	c7552
#UBT	57	28	130	43	92	212	262	312	24	268
#UCT	18	9	20	10	20	31	29	36	7	31
size of tables(Kb)	27.5	8.9	68.6	18.1	47.3	127.1	129.4	250.5	7.5	163.2
CPU time(sec)	11.9	4.0	44.0	9.9	28.9	99.0	89.9	194.0	7.6	147.7

(UBT: Unique-Bridge Table; UCT: Unique-Cell Table)

Table.3 shows some circuit statistics and in Table.4 the analysis results for the bridging faults are summarized. In general, the number of unique-cells (same as the number of unique-cell tables) in each circuit is far less than the actual cell library. The c6288 of 1848 cells has only 7 unique-cells. The actual number of unique-bridge tables derived from the extracted bridging faults is also far less than the number of extracted bridges and even less than the number of combinations of the unique-cells for each design. For instance, the c7552 has 49640 bridges but only 268 unique-bridge tables are derived. The combinations of two cells of 31 unique-cells in c7552 are already 465 ! Table.5 shows the effectiveness of using the techniques described in Section 4. The reduction technique in Section 4.1 reduce about 40% to 80% of the total SPICE simulation tasks. On average 65% SPICE computations are saved. The dynamic derivation of multi-input thresholds also reduces on average about 65% of the SPICE simulations compared to enumerating the unique-cells for each design. Consequently the unique-bridge tables and unique-cell tables are derived very fast. The times listed in Table.4 are the actual CPU times in seconds. The dynamic derivation for a specific circuit instead of enumerating the whole cell library not only results in very fast execution time but also uses a small amount of memory. In Table.4 the total size of both tables are listed. Only up to 250 kilo-bytes are required for the largest circuit. It is clear that both the CPU time and the memory use depend on the number of unique-bridge tables and the unique-cells as shown in Fig.11.

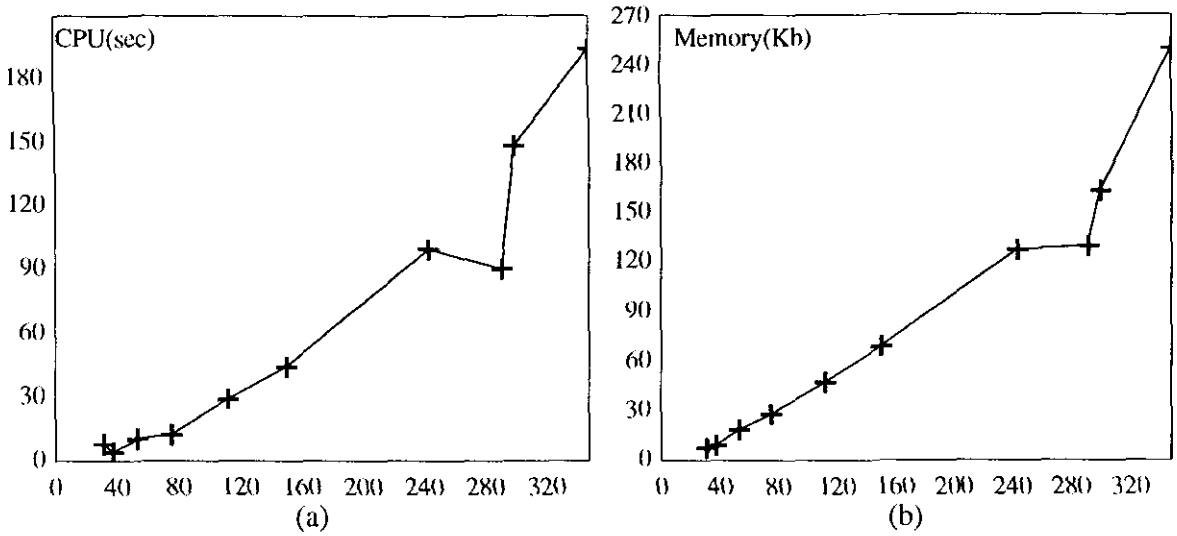

Figure.12 (a) CPU time vs. (#UBT+#UCT). (b) Memory vs. (#UBT + #UCT).

Table.5 Reduction of SPICE simulations

circuit		c432	c499	c880	c1355	c1908	c2670	c3540	c5315	c6288	c7552
#spice computations	actual	547	148	1919	412	1182	4010	3789	7545	175	5838
	total	1837	774	5116	1469	3836	10485	10853	20799	280	15715
	reduce%	70%	81%	63%	71%	69%	62%	65%	64%	38%	63%
#multi–input thresholds	actual	29	18	67	27	46	76	103	130	4	152
	enumerate	119	34	155	58	170	269	243	430	12	360
	reduce%	76%	63%	57%	54%	73%	72%	58%	70%	50%	58%

The fault simulation results are shown in Table.6. The bridges are simulated for the pattern sets that are derived at the gate level representations for SSA faults. The fault simulation is performed in one run for both stuck–at and bridging faults. The fault coverage is classified as *SSA coverage* and *bridge coverage*. The fault coverage results are not really surprising. The execution time, however, is very fast. The last row (*errors %*) indicates the possible false interpretation percentages during the whole fault simulation. This is, the percentage of the situations that the analog input is the same as or very close to (with $\pm$0.02V difference) the threshold of fanout cells. It is not a substantial problem for this set of benchmarks.

Table.6 Results of PPSFP simulation for stuck–at test pattern set

circuit	c432	c499	c880	c1355	c1908	c2670	c3540	c5315	c6288	c7552
#pattern	75	71	95	101	147	160	242	211	44	318
SSA%	99.7	100	100	100	100	98.5	98.2	100	99.9	99.6
bridge%	83.4	82.2	87.9	78.7	71.1	84.1	47.5	78.2	52.7	67.1
CPU time(sec)	0.76	0.86	1.96	1.62	3.32	7.75	25.8	30.2	10.1	70.1
errors %	0.02%	0.0%	0.01%	0.1%	0.01%	0.02%	0.01%	0.04%	0.2%	0.03%

6. Concluding remarks

It is hard to make a comparison with other methods since most of those methods do not include the time and memory use of the preprocessing. Further, how the bridging faults are selected, the test pattern sets, design approach and process parameters (SPICE parameters) can make a lot of difference as well. Nevertheless, intuitively this method is much more accurate than other approximation methods. Compared to methods of using pre–computed tables by enumerating cell library, our method is also more accurate since the multi–threshold and multi–input threshold effects are considered. The introduction of the unique–bridge table and the unique–cell table greatly facilitate the use of efficient simulation technique as shown by the parallel pattern simulation that principally makes our fault simulation fast. Furthermore the dynamic derivation by analyzing the extracted bridges for a specific design makes the analysis very fast and requires much less memory than others. The technique of reducing unnecessary SPICE simulations proves to be effective. In addition, this method does not require cell library information and the inputs are plain representations of extracted transistors. The unique–cells are derived for a specific design. Thus it can be easily used for any other design style. Lastly, the same idea can be used for bridging faults involving internal nodes of cells. We believe that it is a good alternative for simulating bridging faults.

This method, however, does have several limitations. The assumption of the inputs being the full voltage value of "1" and "0" during the derivation makes it inadequate to handle feedbacks since in case of the feedbacks, some of the inputs may remain at an intermediate value. Further if the situations that a bridged voltage is the same as (or very close to) the threshold of fanout cells happen very frequently, our method may induce many errors and the results are not reliable although our experiments show that it is very unlikely.

7. References

[1].Malaiya Y.K. and S.Y.H. Su, "A New Fault Model and Testing Technique for CMOS Devices". In: *Proc. of International Test Conference, Philiadelphia, USA, September, 1982.* P. 25–34.

[2].Acken J. M., "Testing for Bridging Faults (shorts) in CMOS Circuits". In: *Proc. 20th Design Automatic Conference, Miami Beach, FL, , June 27–29, 1983.* P.717–718.

[3].Bryant R. E. and M.D. Schuster. "Fault Simulation of MOS Circuits". *VLSI Design*, Vol.4 (1983), No. 6, p. 24–30.

[4].Saab D. and I. Hajj, "Parallel and Concurrent Fault Simulation of MOS Circuits". In: *Proc. Int. Conf. Computer Design, Cambridge, Mass., USA, October, 1984.* P.752–756.

[5].Banerjee P. and J. A. Abraham, "Characterization and Testing of Physical failures in MOS Logic Circuits". *IEEE Design & Test*, August 1984, p. 76–86.

[6].Acken J. M., "Driving Accurate Fault Models". *Computer System Lab. Standford Univ.*, USA. Report, Computer System Lab. nr. CSL–TR–88–365, October 1985.

[7].Waicukauski J.A. and E.B. Eichelberger, D.O. Forlenza, E. Lindbloom, Th. McCarthy, "Fault Simulation for Structured VLSI". *VLSI Systems Design*, Dec. 1985, p. 20–32.

[8].Maly W., "Realistic Fault Modeling for VLSI Testing". In: *Proc. 24th Design Automatic Conference, Miami, USA, June, 1987.* P. 173–180.

[9].Ferguson F. J. and J. P. Shen, "A CMOS Fault Extractor for Inductive Fault Analysis". *IEEE Trans. Computer–Aided Design*, vol. CAD–7(1988), no. 11, p. 1181–1194.

[10].Soden J.M. and C.F. Hawkins, "Electrical properties and detection methods for CMOS IC defects". In: *Proc. European Test Conf., Paris, France, April, 1989.* P. 159–167.

[11].Bracs K.S. and R.L. Rudell, R.E. Bryant, "Efficient Implementation of a BDD Package". In: *Proc. 27th ACM/IEEE Design Automation Conf., Orlando, USA, June, 1990.* P. 40–45.

[12].Storey T.M. and W. Maly, "CMOS Bridging Fault Detection". In: *Proc. Int. Test Conf., Washington, USA, September, 1990.* P. 842–851.

[13].Millman S. D. and J. P. Garvey, "An Accurate Bridging Fault Test Pattern Generation". In: *Proc. Int. Test Conf., Nashville, USA, September, 1991.* P. 411–418.

[14].Ferguson F. J. and T. Larrabee, "Test Pattern Generation for Realistic Bridge Faults in CMOS ICs". In: *Proc. Int. Test Conf., Nashville, USA, September, 1991.* P. 492–499.

[15].Vandris E. and G. Sobelman, "Algorithms for fast and memory efficient switch–Level fault simulation". In: *Proc. 28th ACM/IEEE Design Automation Conf., San Francisco, USA, 1991.* P.138–143.

[16].Greenstein G. S. and J. H. Patel, "E–PROOFS: A CMOS Bridging Fault Simulator". In: *Proc. Int. Conf. on Computer–Aided Design, Santa Clara, USA, November, 1992.* P. 268–272.

[17].Acken J. M. and S.T. Millman, "Fault Model Evolution for Diagnosis: Accuracy vs Precision". In: *Proc. of Custom Integrated Circuits Conf., Boston, USA, 1992.* P.13.4.1–13.4.4.

[18]. Di C. and J. Jess, "On CMOS Bridge Fault Modeling and Test Pattern Evaluation". In: *Proc. 11th IEEE VLSI Test Symp., Altantic City, USA, April, 1993.* P.116–119.

[19]. Chuang W. and I. N. Hajj, "Fast Mixed–mode Simulation for Accurate MOS Bridging Fault Detection". In: *Proc. International Conf. on Circuits and Systems, Chicago, USA, 1993.* P.1503–1507.

[20]. Chess B. and T. Larrabee, "Bridge Fault Simulation Strategies for CMOS Integrated Circuits". In: *Proc. 30nd ACM/IEEE Design Automation Conf., Dallas, USA, June, 1993.* P.1503–1507.

[21]. Rearick J. and J. H. Patel, "Fast and Accurate CMOS Bridging Fault Simulation". In: *Proc. of International Test Conference, Baltimore, USA, September, 1993.* P.54–62.

[22].Maxwell P.C. and R. Aitken, "Biased Voting: a Method for Simulating CMOS Bridging Faults in the presence of Variable Gate Logic Thresholds". In: *Proc. of International Test Conference, Baltimore, USA, September, 1993.* P. 63–72.

Eindhoven University of Technology Research Reports ISSN 0167-9708
Coden: TEUEDE

Faculty of Electrical Engineering

(256) Bäckx, A.C.P.M and A.A.H. Damen
IDENTIFICATION FOR THE CONTROL OF MIMO INDUSTRIAL PROCESSES.
EUT Report 91-E-256. 1991 ISBN 90-6144-256-7

(257) Maagt, P.J.I. de and H.G. ter Morsche, J L.M. van den Broek
A SPATIAL RECONSTRUCTION TECHNIQUE APPLICABLE TO MICROWAVE RADIOMETRY.
EUT Report 92-E-257. 1992. ISBN 90-6144-257-5

(258) Vleeshouwers, J.M.
DERIVATION OF A MODEL OF THE EXCITER OF A BRUSHLESS SYNCHRONOUS MACHINE.
EUT Report 92-E-258 1992. ISBN 90-6144-258-3

(259) Orlov, V.B.
DEFECT MOTION AS THE ORIGIN OF THE 1/F CONDUCTANCE NOISE IN SOLIDS.
EUT Report 92-E-259. 1992. ISBN 90-6144-259-1

(260) Rooijackers, J.E.
ALGORITHMS FOR SPEECH CODING SYSTEMS BASED ON LINEAR PREDICTION.
EUT Report 92-E-260. 1992. ISBN 90-6144-260-5

(261) Boom, T.J.J. van den and A.A.H. Damen, Martin Klompstra
IDENTIFICATION FOR ROBUST CONTROL USING AN H-infinity NORM.
EUT Report 92-E-261. 1992. ISBN 90-6144-261-3

(262) Groten, M and W. van Etten
LASER LINEWIDTH MEASUREMENT IN THE PRESENCE OF RIN AND USING THE RECIRCULATING SELF HETERODYNE METHOD.
EUT Report 92-E-262. 1992. ISBN 90-6144-262-1

(263) Smolders, A.B.
RIGOROUS ANALYSIS OF THICK MICROSTRIP ANTENNAS AND WIRE ANTENNAS EMBEDDED IN A SUBSTRATE.
EUT Report 92-E-263. 1992. ISBN 90-6144-263-X

(264) Freriks, L.W. and P.J.M. Cluitmans, M.J. van Gils
THE ADAPTIVE RESONANCE THEORY NETWORK: (Clustering-) behaviour in relation with brainstem auditory evoked potential patterns.
EUT Report 92-E-264. 1992. ISBN 90-6144-264-8

(265) Wellen, J.S. and F. Karouta, M.F.C Schemmann, E. Smalbrugge, L.M.F. Kaufmann
MANUFACTURING AND CHARACTERIZATION OF GAAS/ALGAAS MULTIPLE QUANTUMWELL RIDGE WAVEGUIDE LASERS.
EUT Report 92-E-265. 1992. ISBN 90-6144-265-6

(266) Cluitmans, L.J.M.
USING GENETIC ALGORITHMS FOR SCHEDULING DATA FLOW GRAPHS.
EUT Report 92-E-266. 1992. ISBN 90-6144-266-4

(267) Józwiak, L. and A.P.H. van Dijk
A METHOD FOR GENERAL SIMULTANEOUS FULL DECOMPOSITION OF SEQUENTIAL MACHINES: Algorithms and implementation.
EUT Report 92-E-267. 1992. ISBN 90-6144-267-2

(268) Boom, H. van den and W. van Etten, W.H.C. de Krom, P. van Bennekom, P. Huijskens, L. Niessen, F de Leijer
AN OPTICAL ASK AND FSK PHASE DIVERSITY TRANSMISSION SYSTEM.
EUT Report 92-E-268. 1992. ISBN 90-6144-268-0

Eindhoven University of Technology Research Reports ISSN 0167-9708
Coden: TEUEDE

Faculty of Electrical Engineering

(269) Putten, P.H.A. van der
MULTIDISCIPLINAIR SPECIFICEREN EN ONTWERPEN VAN MICROELEKTRONICA IN PRODUKTEN (in Dutch).
EUT Report 93-E-269. 1993. ISBN 90-6144-269-9

(270) Bloks, R.H.J.
PROGRIL: A language for the definition of protocol grammars.
EUT Report 93-E-270. 1993. ISBN 90-6144-270-2

(271) Bloks, R.H.J.
CODE GENERATION FOR THE ATTRIBUTE EVALUATOR OF THE PROTOCOL ENGINE GRAMMAR PROCESSOR UNIT.
EUT Report 93-E-271. 1993. ISBN 90-6144-271-0

(272) Yan, Keping and E.M. van Veldhuizen
FLUE GAS CLEANING BY PULSE CORONA STREAMER
EUT Report 93-E-272. 1993. ISBN 90-6144-272-9

(273) Smolders, A.B
FINITE STACKED MICROSTRIP ARRAYS WITH THICK SUBSTRATES.
EUT Report 93-E-273. 1993. ISBN 90-6144-273-7

(274) Bollen, M.H.J. and M.A. van Houten
ON INSULAR POWER SYSTEMS: Drawing up an inventory of phenomena and research possibilities.
EUT Report 93-E-274 1993. ISBN 90-6144-274-5

(275) Deursen, A.P.J. van
ELECTROMAGNETIC COMPATIBILITY: Part 5, installation and mitigation guidelines, section 3, cabling and wiring.
EUT Report 93-E-275. 1993. ISBN 90-6144-275-3

(276) Bollen, M.H.J.
LITERATURE SEARCH FOR RELIABILITY DATA OF COMPONENTS IN ELECTRIC DISTRIBUTION NETWORKS.
EUT Report 93-E-276. 1993. ISBN 90-6144-276-1

(277) Weiland, Siep
A BEHAVIORAL APPROACH TO BALANCED REPRESENTATIONS OF DYNAMICAL SYSTEMS.
EUT Report 93-E-277. 1993. ISBN 90-6144-277-X

(278) Gorshkov, Yu.A. and V.I. Vladimirov
LINE REVERSAL GAS FLOW TEMPERATURE MEASUREMENTS: Evaluations of the optical arrangements for the instrument.
EUT Report 93-E-278. 1993. ISBN 90-6144-278-8

(279) Creyghton, Y.L.M. and W.R. Rutgers, E.M. van Veldhuizen
IN-SITU INVESTIGATION OF PULSED CORONA DISCHARGE.
EUT Report 93-E-279. 1993. ISBN 90-6144-279-6

(280) Li, H.Q. and R.P.P. Smeets
GAP-LENGTH DEPENDENT PHENOMENA OF HIGH-FREQUENCY VACUUM ARCS.
EUT Report 93-E-280. 1993. ISBN 90-6144-280-X

(281) Di, Chennian and Jochen A.G. Jess
ON THE DEVELOPMENT OF A FAST AND ACCURATE BRIDGING FAULT SIMULATOR.
EUT Report 94-E-281 1994. ISBN 90-6144-281-8