Event-driven incremental timing fault simulator

S.-J. Jou
S.-H. Chiou
Y.-S. Tao
W.-Z. Shen

Indexing terms: Fault simulation, MOS devices

Abstract: An efficient simulator of multiple sets of multiple faults, with electrical timing information for an MOS IC, is presented. The physical faults in a real circuit are modelled more realistically by the node-short, line-open and threshold voltage degradation faults at the transistor level. On using event-driven, selective trace and mixed incremental-in-space, signal and time simulation techniques, the simulation results show that it is superior to other approaches in speed, extra memory used, and precision. Moreover, this simulator is suitable for parallel simulation in a multiprocessor system.

1 Introduction

Higher clock rates and the increasing density of transistors on VLSI chips make it essential to test the dynamic behaviour. For this purpose, it would be inadequate to model a fault at a logic level like stuck-at fault [3]. This is because fault simulators based on gate level primitives can model only a limited class of faults that occur in MOS circuits. Moreover, some faults can often change even a simple MOS gate into a sequential circuit or change the logic function in such a way that it cannot be modelled as an input or output stuck-at fault [1–3].

Transistor level fault modelling gives a good approximation to faults in an active circuit. Therefore many fault simulators perform fault simulation at the transistor switch level, where a transistor stuck-open (on) fault can be modelled [4–6]. Also, some simulators include delay fault (transient, path and gate delay) simulation ability [7–11] to simulate the faults that not only cause logic state errors but also cause timing or transient errors. Hence the quality of the test vector is improved. All the above fault models and simulators have some distinct properties and applications, but all of them lack the voltage waveform and real timing information that are needed in high-performance or mixed analogue/digital circuits. Thus, it would be useful to have a fault simulator that could handle real electrical fault specifications and perform fault simulation on circuits which might be of mixed analogue and digital nature [12]. But it needs more computation and extra memory [13] to simulate the dynamic voltage waveform at the electrical level and, because the voltage waveform is simulated, traditional logic level simulation techniques such as parallel, concurrent, deductive and new differential [14] methods are no longer practical. Therefore, a new simulation technique is needed.

In this paper, FMOTA, an efficient simulator of MOS multiple sets of multiple faults with electrical timing information, is presented. The physical faults in the real circuit are modelled more realistically by the node-short, line-open and threshold voltage degradation faults at the transistor level. As for fault simulation, FMOTA copes with multiple faults (a set of faults) occurring in the circuit, and several fault sets can be inserted into the circuit. Instead of individually simulating the faulty circuit as many times as the number of the fault sets by using SPICE [15] or other circuit or timing simulators, FMOTA simulates the fault sets in one pass. This is possible in FMOTA because we isolate the part of the faulty circuit into a fault-source block and then a fault effect propagation is performed to outline the part of the circuit that is affected by the faulty signal. By using an event-driven, selective trace and mixed incremental-in-space, signal and time simulation techniques, we do all the simulation of the faulty circuits concurrently and therefore the simulation time and extra memory used are reduced. The simulation results show that it is superior to other approaches in speed, extra memory used and precision.

FMOTA is implemented in C under the UNIX operation system and adopts the same input format as SPICE, except that extra commands for fault specification are added. The simulation results given in this paper show the superiority of FMOTA over other approaches to fault simulation.

2 Fault models and basic simulation techniques

2.1 Fault model

The analysis of failure modes in VLSI circuits is a complex problem [16], but it is indicated that observed failure modes mainly consist of short and open circuits at the level of interconnection and devices [2, 17]. Also,

Paper 9054G (E10), first received 5th February and in revised form 22nd June 1992

S.-J. Jou is with the Department of Electrical Engineering, National Central University, ROC

S.-H. Chiou is with the HMC, Science based Industrial Park, ROC

Y.S. Tao and W.-Z. Shen are with the Institute of Electronics, National Chiao-Tung University, ROC

Some of the results of this work have been published in the proceedings of the International Symposium on VLSI Technology, Systems and Applications, Taipei, ROC, May 1991, pp. 424–427.

there are device failures caused by some physical or geometric errors that degrade the performance of the circuit. Thus the faults have been categorised by two types [18]: catastrophic and parametric. Catastrophic faults are random defects that cause hard failure of an active device or interconnection, and result in the failure of the whole chip. By parametric faults, is meant excessive statistical variations in process conditions which cause a soft failure. A soft failure is not sufficient to result in an inoperable IC, but is sufficient to cause performance (e.g. delay time or waveform distortion) to deviate outside some allowable limits.

In order to have a fault model which is close to the fault occurring in reality and which can easily model any type of physical fault [19], we use the fault models listed in Fig. 1. In this fault model, complete node-short and line-open faults can model the most frequently occurring failure modes in an IC that may result in catastrophic faults. In the incomplete node-short and line-open fault models, an *RC* network is introduced [2], and they have the ability to model the parametric faults. For an MOS device, the threshold voltage degradation fault model can cover device failures that cause either catastrophic or parametric faults. Finally, multiple faults are allowable [20] that enable the simulator to simulate multiple faults occurring in ICs.

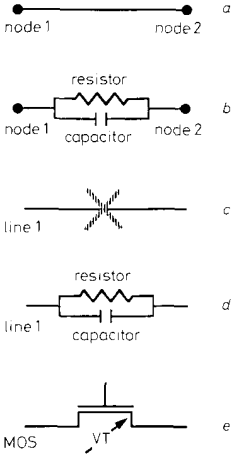

Fig. 1 *Five fault models used in FMOTA*
a Complete node-short
b Incomplete node-short
c Complete line-open
d Incomplete line-open
e Threshold voltage degradation

The circuit level models used in FMOTA have several advantages. First, they can cover all the fault models at the logic level, as shown in Fig. 2. For example, a stuck-at 1/0 fault can be modelled by a short fault, i.e. the node is shorted to V_{DD}/GND. A stuck-open/on fault can be modelled as threshold voltage degradation fault by setting the threshold voltage of the faulty device so that it is always off or on. Also, the gate–oxide shorts of an MOS device can be modelled by node-shorts among the gate, drain and source terminals. The bridge fault represents two shorted nodes, and is simply modelled by a short fault. The failure in an IC that causes a delay fault is more complicated. It may result from the abnormal incomplete line-open that cause resistance of some lines to increase, charging or discharging current variation due to change of threshold voltages, etc. These conditions are all well modelled by the fault models listed in Fig. 1. The second advantage is the representation of the real fault occurring in the circuit. Physical failures such as shorts and opens imply the consideration of the actual topology of the circuit. These lead to the rejection of the representation of the circuit by a logic diagram. Some connections of the real circuit are indeed not represented on the logic diagram and, inversely, some connections appearing on the logic diagram do not exist in the physical circuit [2]. The third advantage is that, because multiple faults are allowable, it can model the condition when a single type of failure causes multiple faults such as threshold voltage degradation of several MOS devices or multiple independent failures. The probability of these conditions is increased because of the reduction in the size of the IC features. In these cases, a single fault model may no longer be adequate. In addition, using the fault model at circuit level can model the faulty behaviour more accurately, and obtain the resultant waveform with high quality; i.e. the fault information is close to the physical nature of the circuit.

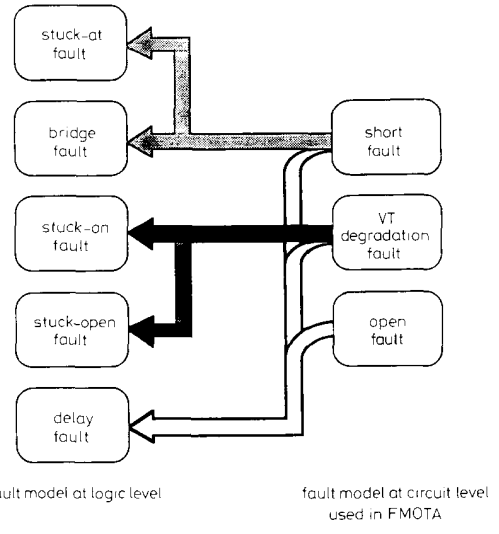

Fig. 2 *Comparisons of fault models at circuit level and logic level*

Although, the more these faults are related to the physical nature of the circuits, the higher the quality of the test, as a general consequence, the more laborious is the generation of the test sequence [2, 13]. Hence, we only simulate the fault list selected by user under the test sequence indicated by user. To reduce the simulation time and memory used, a new fault simulation technique will be presented in Section 3.

2.2 Basic simulation technique

To speed up the simulation, FMOTA based on EMOTA [21] which is two orders faster than SPICE2G.6, uses a unidirectional nonlinear Gauss–Seidel relaxation technique [22] to decouple the circuit into blocks of tightly coupled subcircuits. A partition subroutine is included to group the tightly coupled circuit into subcircuits, between which there are only loosely coupled relations. A gate evaluator and a block matrix solver are used to simulate

fast and accurately two different types of subcircuit, single logic gate (gate block) and tightly coupled circuit (block). An example is illustrated in Fig. 3 to show the partition scheme. More detailed descriptions of the simulation techniques can be found [21].

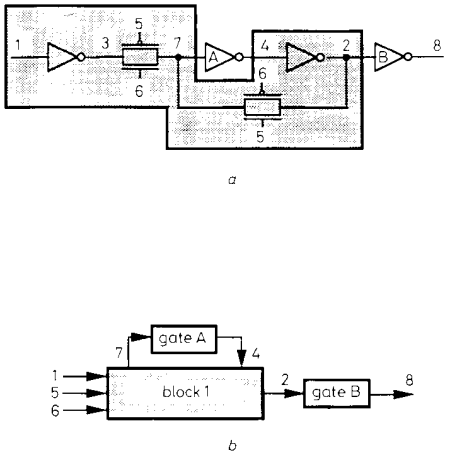

Fig. 3 *Example of partition algorithm*

a Circuit before partition
b Circuit after forming block 1 and single gate blocks A and B
Block internal node: 3; block output node: 2, 7; input source node: 1, 5, 6; single gate block output node: 4, 8

3 Fault simulation technique

As for fault simulation, FMOTA copes with multiple faults (a set of faults) occurring in the circuit, and several fault sets can be inserted into the circuit. Instead of individually simulating the faulty circuit as many times as the number of the fault sets by using circuit simulator like SPICE, FMOTA simulates the fault sets in one pass. To achieve this, a mixture of incremental-in-space, signal and time simulation techniques are used that we shall describe in the following Sections.

3.1 Fault-source block formation: incremental-in-space scheme

To achieve fault simulation, the multiple faults must first be injected into the circuits to form the fault circuit. The topology of the faulty circuit may be completely different from that of the original fault-free circuit. If the fault-free and faulty circuits are to be simulated concurrently and efficiently, then the common part between them must be carefully extracted and fully retained. Actually, the fault may only reconfigure part of the circuit. Thus, if we can extract the parts of the circuits that are reconfigured owing to the effect of the fault, there is no need to duplicate the whole circuit as the faulty circuit. This is like the incremental-in-space scheme [23]. The parts of the circuits that are different between a fault-free and a faulty circuit are called the fault-source block.

The formation of the fault-source block is not an easy task because the fault models used are at the circuit level and any two nodes are allowed to be shorted together. Rules are used in FMOTA to form the fault-source blocks. Before forming the fault-source block, the nodes of the circuit are categorised into four nonoverlapped types, namely block internal node, block output node, input source node, and single-gate block output node. One example is shown in Fig. 3*b*. Then FMOTA finds the associated objective blocks according to which types of faults and nodes are specified in the fault list. Since, multiple faults are allowed, the objective blocks may be fault-free blocks or fault-source blocks that have been formed previously. For the former case, the objective blocks are duplicated and reconfigured into fault-source blocks. For the latter case, we just reconfigure the objective blocks. There are five rules for short faults, three rules for open faults and one rule for threshold voltage degradation fault to form fault-source blocks. The details of the rules are listed below.

3.1.1 Rules for node-short faults

Suppose that nodes A and B are specified in the fault list to represent a short fault. For completely short faults, we just combine nodes A and B and their associated blocks to form the fault-source block. For incompletely short faults, we add an *RC* network between node A and node B. According to the node type of the fault location, five rules are described below:

Rule 1: short between two block internal nodes: As shown in Fig. 4*a*, there is a short between the internal nodes of B_0 and B_1, therefore B_0 and B_1 are the objective blocks and these two blocks are combined into a new fault-source block F_0.

Rule 2: short between two block output nodes: In the case of Fig. 4*b*, there is a short between node A and node B, and so B_0 and B_1 are the objective blocks. These two blocks are combined and modified, as in Rule 1, into fault-source block F_0. If A and B are completely shorted, B_3 or B_2 is also required to be duplicated and modified into another fault-source block. Here, B_3 is chosen to form a fault-source block F_3. This is because nodes A and B are the same in a faulty circuit but different in a fault-free circuit.

Rule 3: short between source node and block output node: In Fig. 4*c*, the output of B_1 is shorted to the voltage source, thus block F_1 is formed to generate faulty signals to the following stages. Also, if it is a complete short, F_3 is needed to pass the voltage signals.

Rule 4: short between source node and block internal node: The internal node of B_1 is shorted to the voltage source, thus block F_1 is formed to generate faulty signals to the following stages, as illustrated in Fig. 4*d*.

Rule 5: short between block internal node and block output node: In Fig. 4*e*, the output node of B_0 and the internal node of B_1 is shorted. Blocks B_0 and B_1 are combined and modified to form the fault-source blocks F_0, and F_0 generates faulty signals to the following stages.

3.1.2 Rules for line-open faults

For the complete line-open fault, the node of the specified fault is split into two disconnected nodes. As for the incomplete line-open fault, a new node is added and a parallel *RC* network is inserted between the open node and the new node. According to the fault location and node type, three rules are shown in Fig. 5 and are described below:

Rule 6: open at the source node: In Fig. 5*a*, the objective block is B_1, thus B_1 is duplicated and modified to be the fault-source block F_1.

Rule 7: open at the output node: If the open occurs at the fanout stem, as illustrated in Fig. 5*b*, B_0 is duplicated and modified into the fault-source block F_1. If one of the fanout branches is incompletely open, as shown in Fig. 5*c*, B_0 is duplicated and modified by adding an *RC* network to form a fault-source block F_0. B_2 is also needed to form another fault-source block F_1, in order to pass the faulty signal. On the other hand, if it is a completely open fault at the branch, as shown in Fig. 5*d*, then only fault-source block F_1 is formed by the modified B_2.

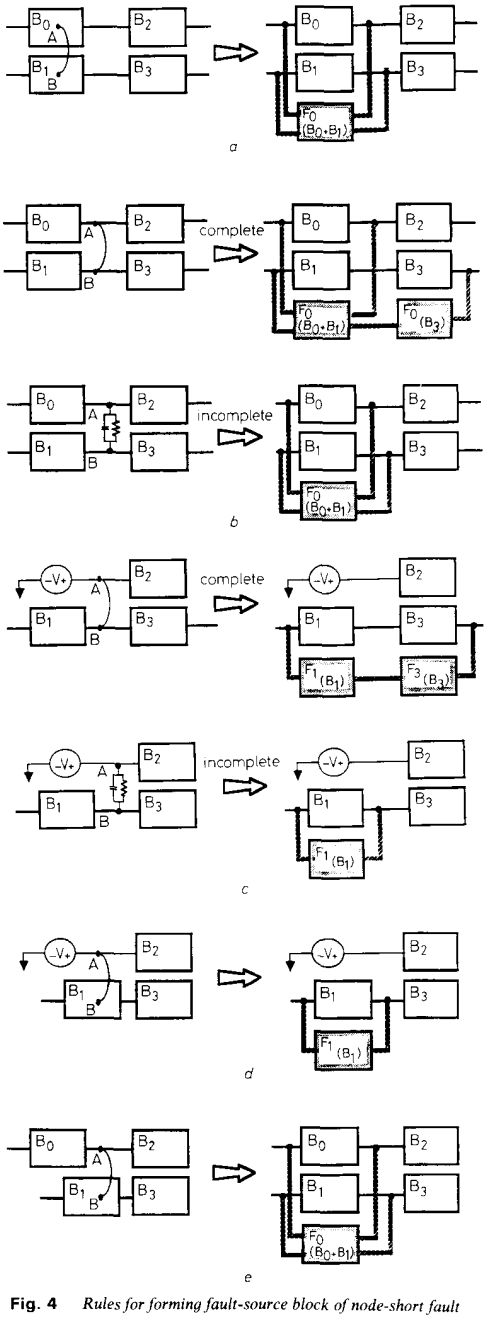

Fig. 4 *Rules for forming fault-source block of node-short fault*

and — original fault-free circuit

and ▬ fault-source block and connection

a Rule 1: short between block internal nodes
b Rule 2: short between block output nodes
c Rule 3: short between block output node and source node
d Rule 4: short between source node and block internal node
e Rule 5: short between block internal node and block output node

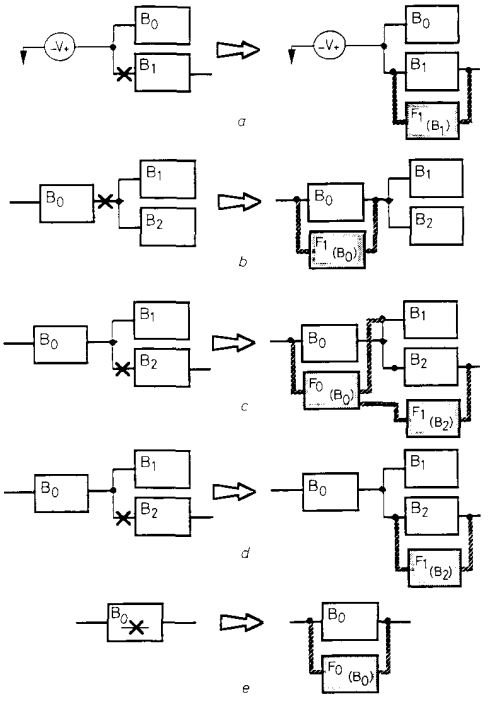

Fig. 5 *Rules for forming fault-source block of line-open fault*

a Rule 6: open at source node
b Rule 7: open at block output node (condition 1)
c Rule 7: open at block output node (condition 2)
d Rule 7: open at block output node (condition 3)
e Rule 8: open at block internal node

Rule 8: open at the block internal node: As shown in Fig. 5*e*, block B_0 is the objective block that contains the node. So B_0 is modified into the fault-source block F_0.

3.1.3 Threshold voltage degradation faults

The objective block is the block containing the faulty MOS component. So the objective block is copied and modified to form the fault-source block by changing the threshold voltage of the component to the specified value described in the fault list of the input file.

Since the differences of the topology between the fault-free and faulty circuits are usually small, by using the incremental-in-space rules to form the fault-source block, the extra memory used due to fault injection is kept small, and the faults are isolated inside fault-source blocks. To simulate the fault-free and faulty circuits concurrently in one pass, the fault-free and faulty signals that are generated by fault-free and fault-source blocks must be simulated and propagated in a proper way. In the following Section, the fault-effect propagation and an incremental-in-signal scheme will be described that will further reduce the memory used, and at the same time speed up the simulation.

3.2 Fault-free propagation: incremental-in-signal scheme

To simulate the faulty circuit, starting from the fault-source blocks, all the blocks that may be affected by the fault are duplicated to form the faulty blocks. The procedure is illustrated by the example shown in Figs. 6*a* and *b* and is called fault effect propagation in FAUST [13]. Assume there are two single faults in Fig. 6*a* that we wish to simulate; one is the block S_2 and the other is in block S_3. The circuit size after injection these two faults is about 2.25 times larger than that of the original fault-free circuit. On carefully examining Fig. 6*b*, we see that, after the fault-source blocks have been simulated, the faulty signals just pass those duplicated blocks whose configurations are the same as in the fault-free circuit. Hence, in FMOTA, only the signal lines of the blocks which are affected by the fault-source blocks are doubled to carry the faulty signal; i.e. a node is not just carrying the fault-free signal but is also carrying the faulty signals if the faulty signals will affect this node. The new fault effect propagation method is shown in Fig. 6*c*, where only block $S_2(1)$ is formed and nodes E, F and G are duplicated for fault 1, and block $S_3(2)$ is formed and nodes F and G are duplicated for fault 2. This scheme is called incremental-in-signal. Using this method, only two extra fault-source blocks are added into the original fault-free circuit. Compared with the circuit in Fig. 6*b*, the extra memory used in FMOTA is only 40% of that used in FAUST. A more general case is shown in Fig. 7, where the circuit is partitioned into seven blocks and three fault sets are injected into the circuit. Moreover, if the output signal of the fault-source and fault-free block are the same, only one signal is evaluated. This procedure is determined by the event-driven control scheme, and the other signal only duplicates the results. So the whole fault simulation technique is a mix of incremental-in-space, signal and time; i.e. *duplicate the fault-source blocks only, increase the faulty signal lines, and dynamically simulate the block and signal by the event-driven technique.* This scheme is the key point of FMOTA to speed up the fault simulation and reduce the extra memory used. In the following Section, the algorithm of the faulty event-driven technique will be briefly described.

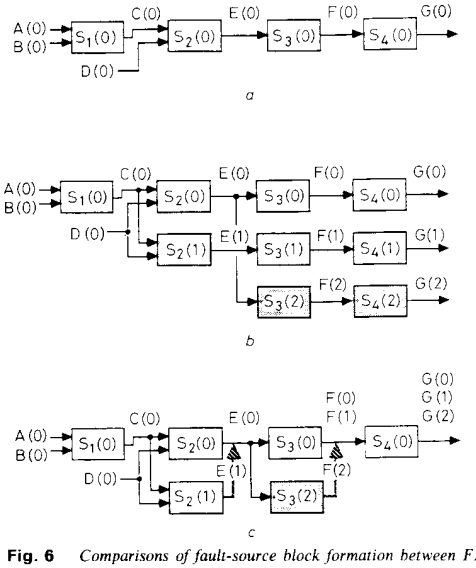

Fig. 6 *Comparisons of fault-source block formation between FAUST and FMOTA*

a Fault-free circuit after partition
b Faulty and fault-free circuit formed by FAUST
c Faulty and fault-free circuit formed by FMOTA

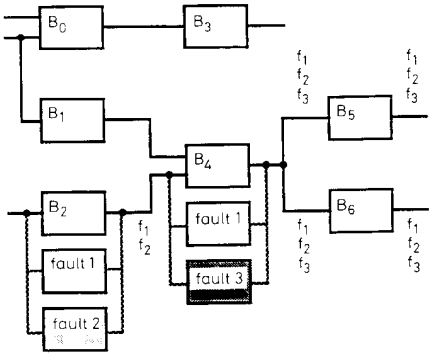

Fig. 7 *Example of fault effect propagation*

3.3 Algorithm of faulty event-driven technique

The algorithm of the event-driven technique is similar to that of EMOTA. There are just two minor differences for fault simulation. One is that a fault number is added into the event description to identify whether the event is generated by fault-free or faulty signals, and the other is in the control of the event as shown by an example in Fig. 8. In Fig. 8*a*, suppose that an event activated by B_1 is a fault-free event, then, among the fanout blocks of B_1, B_2 is certainly triggered to create another fault-free event. The fault-source block of fault 2 is also triggered to create a new faulty event, but the fault-source block of fault 1 is not triggered because it can be triggered only by events with a fault number of 1 that are triggered by previous fault-source blocks of fault 1. If an event is a faulty event, only the fanout blocks with the same fault number are triggered. If no such blocks exist, then the fault-free blocks are triggered to pass the faulty event. This case is shown in Fig. 8*b*.

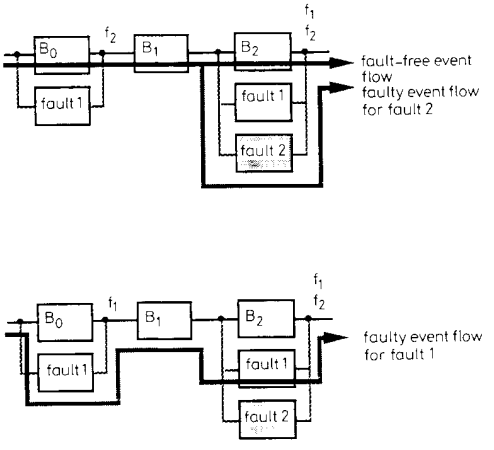

Fig. 8 *Example of faulty and fault-free event-driven control schemes*

The event is initially activated by
a a fault-free block
b a fault-source block

The performance of the above mixed incremental-in-space, signal and time fault simulation technique will be shown in Section 5.

4 Overall description of FMOTA

4.1 Input format of FMOTA

The well known SPICE format is adopted as the input format of FMOTA, but extra commands for fault specification are added for the injection of the multiple sets of multiple faults. In the input format of SPICE, only the nodes of the circuits can be assigned by users, but in FMOTA, the user must specify the lines for the line-open faults and the MOS devices for threshold voltage degradation faults. Here we use node and MOS devices to indicate the line location and, owing to the subcircuit expansion, the name of an MOS device is memorised as a hierarchical name list. For example, if an MOS device is named as SUBCKT1/SUBCKT2/COM_NAME, then it means the component named COM_NAME is within the SUBCKT2 subcircuit and the SUBCKT2 subcircuit is within the SUBCKT1 subcircuit. The fault injection command is listed in Fig. 9.

```
*.FAULT fault-name [[fault type] [fault location] [fault attributes]] . .

   where fault type = "S" for short fault,
                    = "O" for open fault,
                    = "T" for threshold voltage degradation fault.

      fault location = "node1 node2" for short fault,
                     = "node comp_1 comp_2" for open fault,
                     = "MOS_name" for threshold voltage
                     = degradation fault.

      fault attributes = "R = value1 C = value2" for incomplete fault,
                       = "value" for threshold voltage degradation fault.
```

Fig. 9 *Fault injection command of FMOTA*

4.2 Program structure of FMOTA

The program structure of FMOTA is shown in Fig. 10. It consists of several stages, and among them FAULT PREPROCESS, FAULT BLOCK FORMATION, and FAULT SIGNAL PROPAGATION are the main stages for fault process. In the FAULT PREPROCESS stage, the fault information of the injected fault sets, including the fault types, fault locations and fault attributes are recorded and sorted. After the PARTITION stage, all the circuits are decoupled into loosely coupled blocks. The block oriented circuit structure is not only of great service for the event-driven technique, but also the fault effects can be limited into some blocks of the circuit so that there is no need to treat the faulty circuit as a wholly new circuit, as we have described in Section 3. Then the stage of FAULT BLOCK FORMATION sets up the fault-source blocks (increment-in-space) according to the injection rules. These fault-source blocks are marked with different numbers in order to distinguish them between different fault sets. The general procedure in the following FAULT SIGNAL PROPAGATION stage is to search and to duplicate the nodes that are affected by the faults so that the faulty signals can flow in a correct manner (incremental-in-signal). Finally, in the TRANSIENT analysis stage, the events of fault-free and fault-source circuits that are marked with different numbers are evaluated according to the control of time-wheel event-driven scheme (increment-in-time).

The fault simulation algorithm used in FMOTA makes it very suitable for parallel simulation in a multiprocessor system. This is because the whole circuit is partitioned into blocks, and between blocks only unidirectional signal flow will occur. Thus we can assign blocks into different processor elements (PE) and schedule them carefully so that blocks can simulate in parallel if possible [24]. One drawback in the parallel simulation of an event-driven simulation scheme is that, although we can balance statically the working load among PEs, the dynamic working load of each PE may be different because the events occurring in each block may be different. In fault simulation, because we use mixed incremental-in-space, signal and time simulation scheme, when we simulate multiple sets of faults, the circuit size is not increased too much, but the events in each block may increase. This is because there are fault-free and faulty events in the circuit. More events occurring in circuit may well balance the dynamic working load because more events are expected to occur during each time step. The parallel version of FMOTA is underdeveloped on a hypercube multiprocessor system.

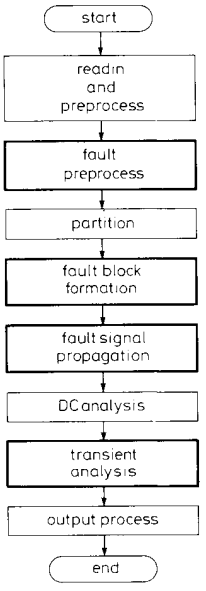

Fig. 10 *Program flow chart of FMOTA*

5 Simulation results and discussion

In this Section, we first give some examples to show the simulation results and to demonstrate the ability of FMOTA to simulate complicated faults. All the simulations were performed on a SUN SPARCstation 2. Each example is designed to illustrate some special fault effects. In these demonstrations, we can see that some fault effects are unpredictable, because they make the circuit behaviour change from combinational circuit to sequential circuit, or make the logic state become intermediate, i.e. between '0' and '1'. So we need to do the simulation to get the reliable waveform. Finally, the memory consumption and speed are compared. The memory consumption estimated from the algorithms of FAUST and FMOTA

are compared with the actual expense of memory run by FMOTA. The speed improvement on using FMOTA for multiple faults of multiple set will also be shown.

The first circuit shown in Fig. 11 is designed to show the validity and precision of FMOTA. The W/L ratio of MOS is marked in the Figure. Then, from the simulation results shown in Fig. 11*b* for simulating f_1, it is clear that the simulated waveforms of FMOTA and SPICE2G.6

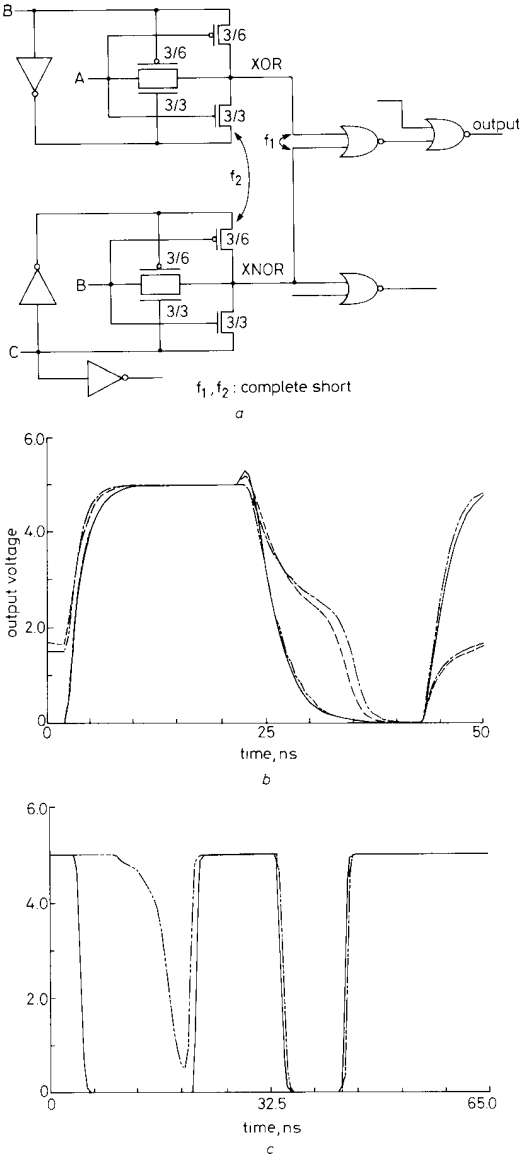

Fig. 11 *Circuit diagram and simulation results of example 1*

a Circuit with two single faults
f_1, f_2 = complete short
b Simulated output waveforms for fault f_1
——— fault-free circuit by SPICE
—·—·— fault-free circuit by FMOTA
– – – faulty circuit by SPICE
—··— faulty circuit by FMOTA
c Simulated output waveforms for fault f_2
——··— fault-free circuit by FMOTA
—·—·— faulty circuit by FMOTA

are closely matched. In this circuit, if both logic states of the output nodes are the same, then the fault effect is masked, but a waveform distortion is observed. However, if the logic states are opposite, the voltage level of the output faulty node is determined by the strength of the signal level, which is determined by the aspect ratio of the MOS device. The circuit in Fig. 11*a* is used again to show the effect of node-short fault which only affects the fall time of the output waveform. Here, the fault injected is labelled f_2 in Fig. 11*a*. The exhaustive test is performed, and the sequence of the applied input patterns is as follows:

A	0	1	1	0	0	1	1	0
B	1	1	0	0	0	0	1	1
C	0	0	0	0	1	1	1	1

The simulated output waveforms are shown in Fig. 11*c*. It is evident that the shape of the faulty output waveform is similar to that of the fault-free one, except that, during the transition from (0, 1, 0) to (1, 1, 0), a remarkable slow-to-fall effect appears at the output. For the rest of the input patterns, the injected fault acts as a redundant fault. In other words, at steady state, the faulty circuit behaves as a fault-free circuit. This is because, at the output of the XOR, in addition to the discharging path of the fault-free circuit, the fault produces one extra charging path. When the logic levels of the two paths contradict, the final logic level will depend on the strength of the two paths, and the strength is a function of the dimensions of the MOS transistors. The time interval of transient analysis performed by FMOTA is 65 ns, and it only needs 3.63 CPU seconds to get the results.

The second example is a CMOS gate as shown in Fig. 12*a*. This example is given by FAUST [13] and is

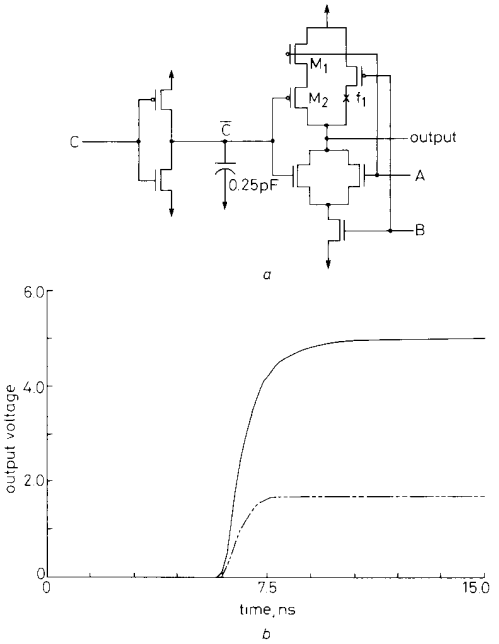

Fig. 12 *Circuit diagram and simulation results of example 2*

a Circuit with one open fault
b Output waveforms of fault-free and faulty circuit
——— fault-free circuit by FMOTA
– – faulty circuit by FMOTA

designed to show the effect due to time delay. The fault injected is a complete line-open fault which breaks the charging path from V_{DD} to output via a PMOS transistor. Assume that the input pattern (A, B, C) is changing from (1, 0, 1) to (0, 0, 0). If time delay is not considered, the first input pattern will ensure that the output node is isolated from V_{DD} and GND. When the input pattern changes, the output node will still be in the same condition owing to the fault. But if the gate delay is considered, it takes time for the transition of input C to reach the $\bar{C}$ node. In this case, a charging path appears. The output node therefore has the opportunity to be charged through M_2 and M_1 before $\bar{C}$ turns off M_2. Therefore the logic level of the output node depends on the charging time, in other words, it depends on the gate delay of the inverter. In this example, we add an extra capacitance of 0.25 pF to enlarge the gate delay, and the logic level of the output node is raised to about 1.5 V as shown in Fig. 12*b*, which may be treated as a logic '1' or '0' state depending on the logic threshold voltage of the next logic gate. If a fault simulator at logic level without timing information is used, this fault may be treated as a stuck-open fault, and the output node is stuck-at 0. As for the timing delay, it becomes an undetected fault.

To show the advantage of using mixed incremental-in-space, signal and time scheme, an 8-bit counter and a 74181 4-bit ALU are used as examples to estimate the performance of FMOTA. The counter circuit consists of eight T-type flip–flops which are serially cascaded. The counter circuit is a sequential circuit and is highly serial; there is virtually only one path from input to output. The total number of MOS devices is 368, and to demonstrate clearly the performance of FMOTA, eight fault sets are injected into the circuit which are the complete shorts between the outputs Q and $\bar{Q}$ of each T-type flip–flop, respectively. The time interval of transient analysis performed by FMOTA is 200 ns (10 clock cycles), and the performance analysis is shown in Table 1. The number of MOS devices in the fault-source blocks corresponding to each injected fault is listed in the second row. The third row is the ratio of the second row to the total number of MOS devices used in counter. The number of MOS devices that will be affected by the faulty signal is listed in the fourth row and the fifth row represents the ratio of the fourth row to the total number of MOS devices. The sixth row lists the memory requirement of simulating counter after the injection of 'fault_no' sets of faults. For instance, at the fifth column where 'fault_no' is 3, the memory requirement is for fault-free and faulty circuits 1, 2, and 3. The seventh row is the ratio of the sixth row to the memory of the fault-free circuit. The execution time of FMOTA to simulate the counter after the injection of the 'fault_no' sets of faults is listed in the eighth row. The ninth row lists the corresponding execution time for simulating the counter for each fault individually. The tenth row is the accumulation of the ninth row. To simulate eight sets of faults, FMOTA needs only 104.4 s, whereas individual simulation needs 166.5 s. Thus there is a 1.60 speed up in simulation on using the mixed fault simulation technique. This is reasonable because, for individual simulation, the circuit size needed to be simulated is almost nine times that of the original fault-free circuit, whereas, in the mixed fault simulation technique, the circuit size is only 5.5 times (the sum of the fifth row). The memory used to simulate eight sets of faults is only 1.507 times that needed for the original fault-free circuit.

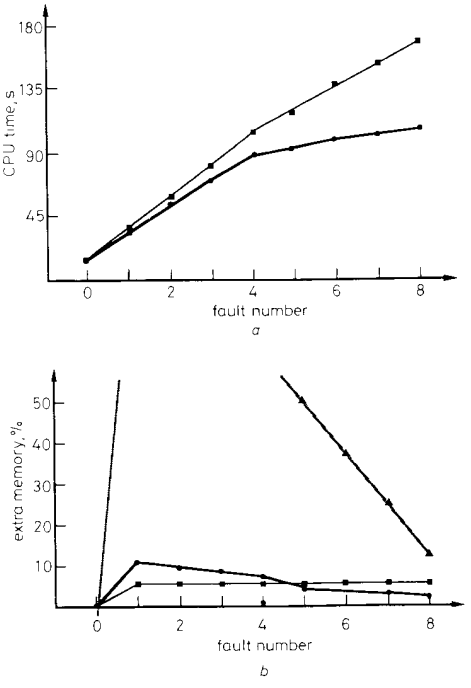

Fig. 13 *Comparison of speed and memory for stimulating 8-bit counter*
a Comparison of accumulated execution time
total time to individually simulate each faulty circuit
run time by FMOTA
b Comparison of memory used for each fault
Faust (estimate)
FMOTA (estimate)
FMOTA (actual)

The advantages of FMOTA can be clearly seen by the curves in Fig. 13. In Fig. 13*a*, the comparison of CPU time used in the simulation is shown. The upper line is the accumulated execution time of individual faulty circuit simulation. It increases almost linearly when the

Table 1: Performance analysis of simulation of 8-bit counter

Fault number	0	1	2	3	4	5	6	7	8
Faulty gate number	0	4	4	4	4	4	4	4	4
Faulty gate ratio	0.000	0.056	0.056	0.056	0.056	0.056	0.056	0.056	0.056
Fanout faulty gate number	0	72	63	54	45	36	27	18	9
Fanout faulty gate ratio	0.000	1.000	0.875	0.750	0.625	0.500	0.375	0.250	0.125
Memory, K	1168	1296	1408	1504	1584	1648	1696	1728	1760
Memory ratio	1.000	1.109	1.205	1.288	1.356	1.411	1.452	1.479	1.507
Accumulated execution time, s	14.7	33.5	53.0	70.2	86.3	93.0	98.5	102.8	104.4
Execution time (faulty circuit)	14.7	20.2	23.4	22.5	22.5	16.3	16.4	15.9	14.7
Accumulated time (above)	14.7	34.9	58.3	80.8	103.2	119.5	135.9	151.8	166.5

Total number of MOS devices: 368
Total time of transient analysis: 200 ns (10 clock cycles)

Table 2: Performance analysis of simulation of 4-bit 74181 ALU

Fault number	0	1	2	3	4	5	6	7	8
Faulty MOS number	0	24	82	16	10	10	34	92	12
Faulty MOS ratio	0.000	0.087	0.297	0.058	0.036	0.036	0.123	0.333	0.043
Fanout faulty MOS number	0	44	126	26	20	10	44	138	138
Fanout faulty MOS ratio	0.000	0.159	0.0456	0.094	0.072	0.036	0.159	0.500	0.500
Memory, K	1552	1552	1560	1560	1560	1560	1568	1584	1600
Memory ratio	1.000	1.005	1.005	1.005	1.005	1.005	1.010	1.021	1.031
Accumulated execution time, s	24.7	28.2	40.1	47.0	48.3	55.2	60.2	65.3	66.2
Execution time (faulty circuit)	24.7	24.8	21.3	23.1	24.9	24.0	24.8	23.3	23.3
Accumulated time (above)	24.7	49.4	70.8	93.8	118.8	142.8	167.6	190.9	214.2

Total number of MOS devices: 276
Total time of transient analysis: 200 ns (7 patterns)

number of the fault sets increases. The lower line is the execution time of FMOTA. The trend of ascendence depends on the locations of the fault injection; i.e. if the locations of the faults are near the primary output, then only a small number of blocks are affected by the faults. On the contrary, if the common part of the fault-free and faulty circuits increases, then the execution time will be reduced. In Fig. 13*a*, for the first few faults, the two curves are closed. This is because each fault affects nearly all the circuit. In Fig. 13*b*, the comparison of memory requirement is plotted, the upper two lines are the estimations of the percentage of the ratio of memory requirement for each faulty circuit and the original fault-free circuit used in FAUST and in FMOTA. The lowest line shows the actual memory used when executing FMOTA.

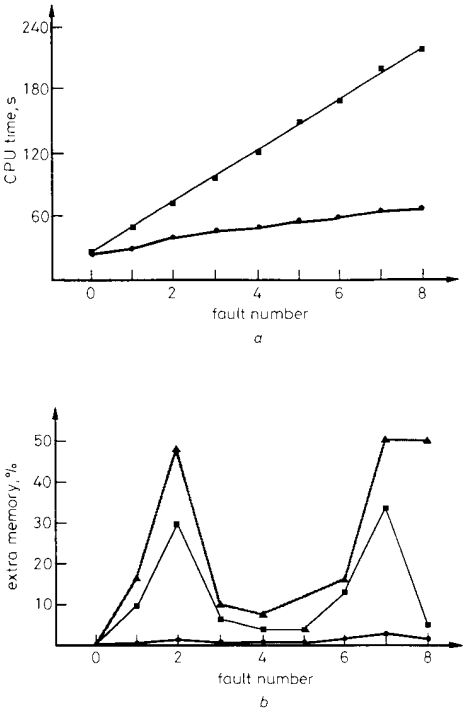

Fig. 14 *Comparison of speed and memory for simulating 74181*
a Comparison of accumulated execution time
total time to individually simulate each faulty circuit
run time by FMOTA
b Comparison of memory used for each fault
Faust (estimate)
FMOTA (estimate)
FMOTA (actual)

It shows that the extra memory used in FMOTA by each faulty circuit is small compared with that of a fault-free circuit, whereas that for FAUST is large. As the fault number is less than five, the actual memory requirement of FMOTA is larger than the estimated memory requirement of FMOTA. This is because the memory of the duplicated nodes which are affected by the fault sets are not included in the estimation of the memory requirement.

Unlike the counter, the ALU is mostly a parallel circuit. The gates of the 74181 are implemented by static CMOS logic, and the total number of MOS transistors is 276. There are eight randomly generated sets of multiple faults that are simulated by FMOTA. Table 2 and Fig. 14 also show the performance of FMOTA in simulating a 74181 circuit. Because the signal flow of the ALU is more parallel than that in the counter, a fault in the ALU circuit will, in general, affect fewer blocks than in the counter circuit. In the ALU case, to simulate eight sets of faults, FMOTA needs only 66.2 s, whereas individual simulation takes 214.2 s. So there is a 3.24 speed up in simulation time, and only 3.1% more memory is used to simulate eight sets of faults by using the mixed incremental-in-space, signal and time fault simulation technique.

6 Conclusions

FMOTA, an efficient simulator of multiple sets of multiple faults, with electrical timing information for an MOS IC, has been presented. The fault models used are node-short, line-open and threshold voltage degradation faults at the transistor level. This fault model can cover all the faults models at logic or switch level and models the physical faults precisely as we have shown in the simulation results. Moreover, multiple faults are allowed. By using event-driven, selective trace and mixed incremental-in-space, signal and time fault simulation techniques, the multiple sets of multiple faults can be simulated concurrently. The simulation results show that the simulation speed of FMOTA is 1.6 or 3.2 times faster than for individual simulation when simulating eight faults in an 8-bit counter or a 4-bit 74181 ALU, and at the same time only 51%, or 3% extra, memory is used. FMOTA is superior to other approaches in speed, extra memory used and precision. Moreover, because the fault simulation algorithm used, FMOTA is suitable for parallel simulation in a multiprocessor system.

7 References

1 BEH, C.C., *et al.*: 'Do stuck fault models reflect manufacturing defects?'. Proceedings of IEEE Test Conference, 1982, pp. 35–42
2 GALIAY, J., CROUZET, Y., and VERGNIAULT, M.: 'Physical versus logical fault models MOS LSI circuits: impact on their testability', *IEEE Trans. Comput.*, 1980, **C-29**, (6), pp. 527–531

3 ABRAHAM, J.A., and SHIN, H.-C.: 'Testing of MOS VLSI circuits'. Proceedings of IEEE International Symposium on Circuits and systems, 1985, pp. 1297–1300

4 SHUSTER, M.D., and BRYANT, R.E.: 'Performance evaluation of FMOSSIM, a concurrent switch-level fault simulator'. Proceedings of 22nd Design Automation Conference, 1985, pp. 715–719

5 ELZIQ, Y.M.: 'Automatic test generation for stuck-open faults in CMOS VLSI'. Proceedings of 18th Design Automation Conference, 1981, pp. 347–354

6 REDDY, M., REDDY, M.K., and AGRAWAL, V.: 'Robust tests for stuck-open faults in CMOS combinational logic'. Proceedings of 14th International Symposium on Fault-tolerant computing, 1984, pp. 20–22

7 LEVENDEL, Y., and MENON, P.R.: 'Transition fault in combinational circuits: input transition test generation and fault simulation'. Proceedings of 16th International Symposium on Fault-tolerant computing, 1986, pp. 278–283

8 SCHULZ, M.H., and BRGLEZ, F.: 'Accelerated transition fault simulation'. Proceedings of 24th Design Automation Conference, 1987, pp. 237–243

9 WAICUKAUSKI, J.A., LINDBLOOM, E., ROSEN, B.K., and IYENGAR, V.S.: 'Transition fault simulation', *IEEE Design and Test*, April 1987, pp. 32–38

10 LESSER, J.D., and SHEDLETSKY, J.J.: 'An experimental delay test generator for logic LSI', *IEEE Trans. Comput.*, 1980, **C-29,** (3), pp. 235–248

11 LO, C., NHAM, H.N., and BOSE, A.K.: 'Algorithms for an advanced fault simulation system in MOTIS', *IEEE Trans.*, 1987, **CAD-6,** (3), pp. 232–240

12 YUN CHENG, JU, FRED L., YANG, and SALEH, RESVE A.: 'Mixed-mode incremental simulation and concurrent fault simulation'. IEEE ICCAD, 1990, pp. 158–161

13 SHIH, H., RAHMEH, J.T., and ABRAHAM, J.A.: 'Faust: an MOS fault simulator with timing information', *IEEE Trans.*, 1986, **CAD-5,** (10), pp. 557–563

14 CHENG, W.T., and YU, M.-L.: 'Differential fault simulation — a fast method using minimal memory'. Proceedings of 26th Design Automation Conference, 1989, pp. 424–428

15 NAGEL, W.: 'SPICE2, a computer program to simulate semiconductor circuits'. University of California, Berkeley, Memorandum ERLM520, May 1975

16 FANTINI, F., and MORANDI, C.: 'Failure modes and mechanisms for VLSI ICS: a review', *IEE Proc. G*, 1985, **132,** (3), pp. 74–81

17 MANGIR, T.E.: 'Source of failures and yield improvement for VLSI and restructurable interconnects for RVLSI and WSI. Part 1 — sources of failures and yield improvement for VLSI', *Proc. IEEE.*, 1984, (6), pp. 690–708

18 STROJWAS, A.J., and DIRECTOR, S.W.: 'An efficient algorithm for parametric fault simulation of monolithic ICs', *IEEE Trans.*, 1991, **10,** (8), pp. 1049–1058

19 MALAIYA, Y.K., and SU, S.: 'A new fault model and testing technique for CMOS devices'. Proceedings of IEEE Test Conference, 1982, pp. 25–34

20 HUGHES, J.: 'Multiple fault detection using single fault test sets', *IEEE Trans.*, 1988, **CAD-7,** (1), pp. 100–108

21 SHEN, W.Z., JOU, S.J., and TAO, Y.S.: 'EMOTA: an event-driven MOS timing simulator for VLSI circuits', *IEE Proc. G*, 1990, **137,** (4), pp. 279–291

22 NEWTON, A.R., and SANGIOVANNI-VINCENTELLI, A.L.: 'Relaxation-based electrical simulation', *IEEE Trans.*, 1983, **ED-30,** (9), pp. 1184–1207

23 HWANG, Y., BLANK, T., and CHOI, K.: 'Fast functional simulation: an incremental approach', *IEEE Trans.*, 1988, **CAD-7,** (7), pp. 765–774

24 JOU, S.J., JEN, CHEIN-WEI, and SHEN, WEN-ZEN: 'Parallelism of circuit simulation on array processor'. IEEE International Symposium on Circuits and systems, Singapore, 1991, pp. 2725–2728