Analysis of Switch-Level Faults by Symbolic Simulation

Lluís Ribas-Xirgo and Jordi Carrabina-Bordoll
Centre Nacional de Microelectrònica, CNM (CSIC)
Universitat Autònoma de Barcelona, UAB
Campus UAB, 08193 Bellaterra, Barcelona, Spain.

Abstract — This paper presents a symbolic method to detect short and open circuit faults in switch-level networks. Detection and fault sensitization vector determination are possible since the behavior of each node is described by a set of two functions: the on-set and the off-set functions. Their analyses provide designers with an efficient tool for circuit verification and test pattern generation.

I. Introduction

Design verification and test pattern generation must cover realistic faults originating in either mistakes made during schematic or layout design, or physical defects during IC fabrication.

Most faults consist of unwanted or missing connections between layers [1,2], creating false short or open circuits that are not always easily mapped onto stuck-at faults at gate-level [1].

On the contrary, these bridging and line open faults can be adequately represented at transistor level; consequently corresponding switch-level networks are considered for analysis and verification.

Several authors have considered switch-level circuits for verification and test pattern generation. Cerny and Gecsei [3], and Bryant [4,5,6] describe logic level circuit simulators that use symbolic MOS circuit analysis to obtain a set of Boolean expressions that simplify the simulation procedure and subsequent circuit analysis. In either case, symbolic information is maintained as ordered binary decision diagrams (OBDDs) [7], because of their properties in both function evaluation and comparison. Consequently, in the COSMOS symbolic simulator [6], a three-valued logic (0,1,X) can be combined with Boolean variables to produce results for both verification and test pattern generation purposes. Similar approaches are used in [8] and [9] for verification and test vector generation, respectively, considering only stuck-at faults.

Due to the difficulty involved in accurately representing CMOS failure modes for stuck-at-1/0 models, bridging and stuck-on/open faults should be considered instead. Especially, if bridging and transistor stuck-on faults may account for as much as 90% [1] of the realistic faults in present day digital CMOS processes.

Therefore, this paper aims to present a new simulation-based symbolic method to detect these switch-level faults and to provide a means of automatically generating test patterns that sensitize symbolically-detected faults.

Bridging and transistor stuck-on faults must be physically detected by IDDQ testing techniques [1,10,11], while sensitized by appropriate stimulus. As many of they result in nodes connected to both power voltage supply and ground terminals by low resistance paths, they are also referred to as short-circuit faults.

Open circuit faults can be detected by applying an initializing input vector prior to a sensitization vector, whose corresponding output is the opposite of the first vector. If there is no change in the output value after supplying the second vector, unwanted memorization occurs, revealing the presence of a fault.

After review of some definitions and notations used in this text, the theoretical basis of the method is explained. In the following sections, the symbolic simulation procedure and transistor models used to derive the Boolean behavior of circuits are described, and their analyses considered. Results and relevance of this work are discussed in the concluding section.

II. Definitions and Notations

This section is devoted to define some of the terms that will be used later on in the paper.

32nd ACM/IEEE Design Automation Conference ®

A literal is a Boolean variable, x_k, or its negation, x'_k. And is usually referred to in lower case letters.

A Boolean expression, F, is a combination of Boolean operators and literals.

A switching function is defined as an application of the form

$$f: B^n \rightarrow B = \{ 0, 1 \} \tag{1}$$

that has a non-unique representation as a Boolean expression. Note that switching functions can also be canonically represented by ordered binary decision diagrams (OBDDs).

The Boolean quotient of f with respect to a product term t is the function formed from f by imposing the constraint t=1. In particular, for a given variable x_i in $\underline{x}$, $f(\underline{x})/x_i$ satisfies

$$f(\underline{x})\Big/_{x_i} = f(\ldots x_{i-1}, 1, x_{i+1}, \ldots) \tag{2}$$

An activation term, $T_{i,j}$, is a Boolean expression that turns on an ideal switch whose source and drain nodes are i and j.

Let P = (n_0,d_0,n_1,d_1, ..., n_{k-1},d_{k-1},n_k) be a path in a network, where n_i are nodes and d_i are connecting devices. Then, the Boolean representation of the conducting link between the two terminal nodes in the path, n_0 and n_k, is the product of all the device activation terms,

$$P_{0 \rightarrow k} = \prod_{i=0+k-1} T_{i,i+1} \tag{3}$$

The on-set of a node is the set of paths driving it to a constant logic value one, that is, a node with a universal on-set,

$$\text{on-set}(\, n_j \,) : N_j^{on} = \left\{ P_{i \rightarrow j} \mid N_i^{on} = 1 \right\} \tag{4}$$

and the off-set of a node is the set of paths leading it to zero (a node containing an universal off-set),

$$\text{off-set}(\, n_j \,) : N_j^{off} = \left\{ P_{i \rightarrow j} \mid N_i^{off} = 1 \right\} \tag{5}$$

Both sets can be considered as a representation of a unique function, and thus denoted $f^1(\underline{x})$ and $f^0(\underline{x})$ as the on and off-set functions, respectively.

$$N_j^{on} \equiv f_j^1(x_0, x_1, \ldots, x_{m-1}) \tag{6}$$

$$N_j^{off} \equiv f_j^0(x_0, x_1, \ldots, x_{m-1}) \tag{7}$$

Therefore, the function implemented on one node is a set of two functions:

$$f_j(\underline{x}) = \left(f_j^1(\underline{x}),\ f_j^0(\underline{x}) \right) \tag{8}$$

A complementary logic function $f(\underline{x})$ will fulfill the following property,

$$f^1(\underline{x}) = \overline{f^0(\underline{x})} \tag{9}$$

There can be two erroneous cases for this type of functions, denoted as the short-circuit and the open-circuit (or memorization) states. Such conditions can be calculated as,

$$f^s(\underline{x}) = f^1(\underline{x}) \cdot f^0(\underline{x}) \tag{10}$$

$$f^z(\underline{x}) = \overline{f^1(\underline{x})} \cdot \overline{f^0(\underline{x})} \tag{11}$$

Equations (10) and (11) correspond to short-circuit and open-circuit function calculations. If either of these functions is not null, the originating function f is considered non-complementary and therefore, faulty.

III. Symbolic Fault Propagation

Switch level circuit faults can be classified into two main categories: open and short circuit faults. The latter have proven more likely to occur in a fabricated chip.

This section analyzes how these faults can be symbolically propagated through a network of switches and detected at the output. To this end, we shall first consider some properties of the functions of the form depicted in (8).

A correct, complementary logic circuit can be modeled as a complementary switching function, $g(\underline{x})$, such that

$$g^1(\underline{x}) \Leftrightarrow \overline{g^0(\underline{x})}, \tag{12}$$

and consequently,

$$g^1(\underline{x}) \cdot g^0(\underline{x}) = 0 \tag{13}$$

The former equation is still valid when a literal is withdrawn from both function representations. More formally, the following lemmata can be stated and proven.

Lemma 1. If $g(\underline{x})$ is the complementary function that characterizes a given switch network G, and x_i one of its input variables taken from $\underline{x}$, then the Boolean quotient of $g(\underline{x})$ with respect to x_i or x'_i is also a complementary function.

Proof. This lemma implies, as seen in the deduction of (13) from (12), that

$$g^1(\underline{x})_{/x_i} \cdot g^0(\underline{x})_{/x_i} = 0 \tag{14}$$

and

$$g^1(\underline{x})_{/\overline{x_i}} \cdot g^0(\underline{x})_{/\overline{x_i}} = 0 \tag{15}$$

By Boole's expansion theorem of both terms in (13), we obtain

$$x_i\left(g^1_{/x_i}\cdot g^0_{/x_i}\right)+\overline{x_i}\left(g^1_{/\overline{x_i}}\cdot g^0_{/\overline{x_i}}\right)=0 \tag{16}$$

which implies that both terms must be zero

$$x_i\left(g^1_{/x_i}\cdot g^0_{/x_i}\right)=0 \tag{17}$$

$$\overline{x_i}\left(g^1_{/\overline{x_i}}\cdot g^0_{/\overline{x_i}}\right)=0 \tag{18}$$

Given that x_i and x'_i are not present in the product of quotients in (17) and (18), respectively, they must be zero regardless of the multiplying literal. Therefore, (14) and (15) are true. ❑

Lemma 2. A variable, x_i, is redundant in a complementary function $f(\underline{x}) \mid x_i \in \underline{x}$, if and only if the Boolean derivative of $f(\underline{x})$ with respect to x_i is zero.

Proof. Here, we recall the definition of the Boolean derivative of a Boolean function f from [12], defined as,

$$\frac{\partial f(\underline{x})}{\partial x_i}=f(\underline{x})_{/x_i}\oplus f(\underline{x})_{/\overline{x_i}} \tag{19}$$

As complementary functions are represented by a set of two functions that follow (9), the former equation can be extended to,

$$\frac{\partial f(\underline{x})}{\partial x_i}=f^1_{/x_i}\cdot f^0_{/\overline{x_i}}+f^0_{/x_i}\cdot f^1_{/\overline{x_i}} \tag{20}$$

which corresponds to the normal expansion of the exclusive-or operation with function negations substituted by off-set functions.

On the other hand, if x_i is a redundant variable in the expression of f, then

$$f(\underline{x})\Big|_{x_i\to 1}=f(\underline{x})\Big|_{x_i\to 0} \tag{21}$$

which can also be expressed as

$$f(\underline{x})_{/x_i}=f(\underline{x})_{/\overline{x_i}} \tag{22}$$

The former identity implies that

$$f(\underline{x})_{/x_i}\oplus f(\underline{x})_{/\overline{x_i}}=0 \tag{23}$$

which is the same as stating that the Boolean derivative of f with respect to x_i is zero. ❑

Corollary 1. The influence of a Boolean variable x_i on a node n_p is determined by the Boolean derivative of the node related function $f_p(\underline{x})$ with respect to x_i. In particular, if x_i is redundant, its influence is null.

Corollary 2. The Boolean derivative of $f_p(\underline{x})$ with respect to x_i corresponds to the set of all possible sensitization paths from node n_i to node n_p.

In this case, node n_i is represented by the following set of functions,

$$f_i(\underline{x})=\left(f_i^1=x_i,\ f_i^0=\overline{x_i}\right) \tag{24}$$

The former corollaries imply that the relation between two nodes is given by the Boolean derivative of the ending node function with respect to the starting one.

Short and open circuit faults are detected by inspecting the expressions resulting from (10) and (11). This examination requires calculating f^s and f^z for each node in a network. A closer look at the problem shows that a simple inspection of the output nodes is sufficient in most circuit realizations.

Theorem 1. Let H be a network of switches with a set of input nodes N= $(n_0,n_1,...,n_{m-1})$ and an output node n_z. And let h be the corresponding function of n_z, described as a vector of variables $\underline{x}=(x_0,x_1,...,x_{m-1})$ corresponding to N, whose related functions are in the form of (13). If one of these functions, f_i, does not have complementary behavior, then h will also be non-complementary.

Proof. Assume that $h(\underline{x})$ has complementary behavior when all input functions f_i are of the form shown in (13). With no loss of generality, we will select f_0 as one of the functions with non-complementary behavior. Then $h(\underline{x})$ can be written in the following form, having replaced variable x_0 with a non-complementary (or faulty) function f_0.

$$h(\underline{x})=\begin{cases}h^0=f_0^0\cdot h^0_{/\overline{x_0}}+f_0^1\cdot h^0_{/x_0}\\ h^1=f_0^0\cdot h^1_{/\overline{x_0}}+f_0^1\cdot h^1_{/x_0}\end{cases} \tag{25}$$

To find the non-complementarities of h, we calculate the short and open circuit functions, according to (10) and (11). For this proof, only short-circuit functions are considered and results will be shown as extensible to open-circuit functions.

$$\begin{aligned}h^s=h^0\cdot h^1=f_0^1\cdot f_0^0\cdot\left(h^0_{/\overline{x_0}}\cdot h^1_{/x_0}+h^0_{/x_0}\cdot h^1_{/\overline{x_0}}\right)+\\ +f_0^0\cdot\left[h^0_{/\overline{x_0}}\cdot h^1_{/\overline{x_0}}\right]+f_0^1\cdot\left[h^0_{/x_0}\cdot h^1_{/x_0}\right]\end{aligned} \tag{26}$$

The last terms of the above equation are null according to lemma 1, and the first term can be expressed through (20) as

$$h^s=f_0^s\cdot\frac{\partial h}{\partial x_0} \tag{27}$$

A similar procedure to calculate h^z will obtain the following formula

$$h^z = f_0^z \cdot \frac{\partial h}{\partial x_0} \tag{28}$$

Examination of the former equations may infer that function h will show a non-complementary behavior if x_0 is not redundant (lemma 2). ❑

Corollary 3. A fault (open or short circuit on a node) is reflected at the output if the node where it is located is not redundant with respect to the complementary function that takes it as an input.

In the next sections, the application of this study is described in detail.

IV. Symbolic Simulation

Circuit analysis is possible when all nodal functions are available. These functions are obtained by applying a special type of symbolic, event-driven [13] simulation to the circuit.

A scheme follows of the simulation procedure for a switch network G(N, D), where N is the set of nodes $n_{i=0\div k-1}$ in the network and D is the set of devices $d_{j=0\div m-1}$ whose model is described by a set of functions Φ_j that depend on the set F_j of its input node related functions f_i.

```
// Simulation pseudo-code.
For each initializing device dj {
        Apply device function Φj (Fj)
        Enqueue modified nodes fi | fi ∈ Fi
}
While ( QUEUE not empty ) {
        Dequeue node ni
        For all devices dj | fi ∈ Fi {
                Apply device function Φj (Fj)
                Enqueue modified nodes fi | fi ∈ Fi
        }
}
```

Device functions Φ must be correctly established to ensure that no destabilizing conditions leading to instability are present in the device model.

Only independent voltage sources (to set primary input nodal functions to variables) and transistors (to process symbolic information) are considered in switch-level circuits.

An independent voltage source device can supply a constant binary value (0 or 1) or a variable. In the last case, its model is described by the following equation

$$v_i(\underline{x}) = \begin{cases} v_i^0(\underline{x}) = \overline{x_i} \\ v_i^1(\underline{x}) = x_i \end{cases} \tag{29}$$

where $v_i(\underline{x})$ is the terminal node n_i of the device.

Transistors are considered as bi-directional devices that modify both drain and source nodal functions. The equations for a P-type transistor drain are given in the following formulae.

$$f_d(\underline{x}) = \begin{cases} f_d^0(\underline{x}) = f_d^0(\underline{x}) + f_g^0(\underline{x}) \cdot f_s^0(\underline{x}) \\ f_d^1(\underline{x}) = f_d^1(\underline{x}) + f_g^0(\underline{x}) \cdot f_s^1(\underline{x}) \end{cases} \tag{30}$$

where f_d, f_g, and f_s are the nodal functions of the transistor drain, gate and source, respectively.

The complete set of equations for a PMOS switch can also be expressed as follows.

$$\begin{aligned} f_d(\underline{x}) &= f_d(\underline{x}) + f_g^0(\underline{x}) \cdot f_s(\underline{x}) \\ f_s(\underline{x}) &= f_s(\underline{x}) + f_g^0(\underline{x}) \cdot f_d(\underline{x}) \\ f_g(\underline{x}) &= f_g(\underline{x}) \end{aligned} \tag{31}$$

Note that the product term is a scalar multiplication of a function $f^0(\underline{x})$ by a vector of functions $f(\underline{x})$.

A similar system would describe the behavior of a NMOS switch, provided that the off-set function of the gate is substituted in (31) by the on-set function.

The simulation algorithm described in this section stops when no new events occur (device functions do not modify nodal functions). That is, when all nodes have reached a stable, final set of functions that reflects their logical behavior. Consequently, its analysis and subsequent detection of errors is possible. In addition, input configurations causing circuit malfunction or fault propagation to the output can be obtained by simple Boolean operations.

V. Circuit Verification

Using the symbolic simulation procedure described previously enables obtaining the Boolean behavior of each node as a pair of functions (on-set and off-set). The results can serve for both formal verification and test pattern generation. This section covers their analysis and some considerations when the working circuit is not implementing complementary logic.

If a given circuit is known to have complementary behavior, all output nodal functions must be complementary. Therefore, any single short or open circuit fault injected in a node will cause at least one of these output to have the same faulty behavior, as stated in Theorem 1. Any cover of the faulty function will be a sensitization vector for that fault.

If an injected fault is not reflected at the output, the originating node is redundant and therefore, can be eliminated.

For circuit design verification purposes, any non-complementary output node will reveal a short/open circuit fault in the circuit caused by a bridge or a break in the layout.

Sequential circuits are not complementary in the form described by (9). Usually, in CMOS logic circuits, nodal functions with memorization capabilities are of the following form

$$f = \left(f^1 = g \cdot h\,,\; f^0 = \bar{g} \cdot h\right) \qquad f^z = \bar{h} \tag{32}$$

In such a case, verification should be extended to compare the calculated memorization function f^z with the desired one. This function contains clock signals. Special care should be taken when simulating such circuits because a clock-controlled switch may annihilate the propagation of product terms containing the negation of the clock. This is prevented by using different variables to identify different clock phases.

Precharged logic stages have their Boolean functions in only one of the two sets (off or on-set) of their output nodes, while the other only contains the propagation controlling terms. That is, output nodal functions, assuming precharge to one, are

$$f = \left(f^1 = h\,,\; f^0 = \bar{h} \cdot \bar{g}\right) \qquad f^z = \bar{h} \cdot g \tag{33}$$

where h is the precharging condition, which usually is opposed to that of the evaluation, h'. For weak, constant loads $\left(f^1 = 1, f^0 = \bar{g}\right)$ the short-circuit condition contains the negation of the implemented function, g. However, this last solution will lead to permanent current consumption which should be avoided in circuit designs.

Symbolic propagation of functions across precharged logic stages is an issue for consideration. The most effective design techniques for dynamic (precharged) logic circuits alternate P-type with N-type evaluation blocks. That is, functions in a path appear alternatively in f^1 and in f^0, being the controlling terms for N-type and P-type evaluation blocks, respectively.

If that were not the case, circuit functionality cannot be adequately obtained, unless predominant values are indicated at stage output nodes. For instance, assuming zero as the dominant value, the output of a precharged stage may be rewritten as

$$f = \begin{cases} f^1 = h \xrightarrow{\text{weak}} h \cdot \overline{\bar{l} \cdot \bar{g}} = h \cdot \bar{l} + h \cdot g \\ f^0 = l \cdot \bar{g} \end{cases} \tag{34}$$

Note that the variables in the precharge term h are usually opposed to those in the evaluation term l. Therefore, they must be identified differently to avoid mutual exclusion. This operation lets the function term g also be present in f^1 and thus, symbolically propagable to stages in which evaluation blocks are controlled by on-sets.

VI. Results and Conclusions

An event-driven symbolic simulator (*SymSim*) has been implemented to obtain the nodal set of functions of a given circuit, described in SPICE [14] format. Device models are implemented through a series of Boolean operations performed by the BDD routines of the SIS package [15,16].

CPU time and memory requirements depend on both the number of nodes and on the number and order of the Boolean variables in the BDD structures. Therefore, the analysis of large circuit will require embedding variable ordering and partitioning strategies [17] into the analyzer.

Table I gives an idea of the computational effort (in terms of CPU processing time) required to verify correct, small/medium size circuits. However, CPU times may vary when faults are introduced in a circuit. These variations strongly depend on the type of the injected faults.

TABLE I
SIMULATION TIMES IN A SUN SPARC 10. SDFFRP IS A SEQUENTIAL CIRCUIT. CXXXX ARE TRANSISTOR-LEVEL DESCRIPTIONS OF THE CORRESPONDING ISCAS'85 BENCHMARKS.

Circuit	#Vars	Nodes	Transistors	Time(s)
ADD2	5	27	40	<1
MULT2	4	29	45	<1
SDFFRP	7	36	54	<1
c17	5	19	24	<1
c432	36	396	753	54
c499	41	613	1349	308
c1355	41	1165	2308	434
c1908	33	1758	3446	509

Table II contains the results of the c432 circuit analysis for some single fault injections. The last column indicates the type of error caused. The meaning of each error is explained in the following paragraphs.

TABLE II
SIMULATION TIMES OF ISCAS'85 C432 CIRCUIT, WITH SINGLE FAULT INJECTION.

Fault	Time(s)	Error
None	54	None
191gat stuck at 1	47	*Cmp*
203gat stuck at 0	20	*Cmp*
309gat short-circuited to Power	28	$I\uparrow$
203gat and 239gat (feedback) bridged	77	$I\uparrow$
344gat and 376gat bridged	63	$I\uparrow$
nMOS stuck-on in 332gat (xor)	72	$I\uparrow$
pMOS stuck-on in 353gat (nand)	96	$I\uparrow$
340gat (nand) pMOS open	60	*Q*
Floating pMOS in 118gat	64	*Q*
158gat and 159gat swapped	80	*Cmp*

"$I\uparrow$" indicates that there is at least one input configuration that provokes an increase of the current consumption due to the sensitization of a conditional short-circuit in a certain node.

"*Q*" indicates that a memorization state has been propagated to at least one of the output and its sensitization path is given by (11).

On the contrary, if every output node shows complementary logic behavior, verification by Boolean comparison is necessary and indicated by "*Cmp*". The corresponding test vector (which sensitizes the path from faulty node to output) is the Boolean difference between the resulting and the desired expressions. This is the case of most stuck-at faults.

In conclusion, the method presented enables detecting and locating faulty (non-complementary) nodes in a digital CMOS circuit. This is possible by operating the function representing the set of paths that puts a node through a logic one with that which represents the set of paths that drives the same node to a logic zero.

What is more, the sensitization functions thus calculated permit direct determination of the corresponding test vectors.

In addition to this, it can also be used to verify a number of circuits, including sequential ones, if clock phases are conveniently assigned to different Boolean variables. This limitation is overcome by considering temporal logic or timing in the simulation procedure of *SymSim* [18,19].

The implemented program can be accelerated by using gate-level models and hierarchical simulation. However, these options will lead to some loss of detail in the resulting analysis.

REFERENCES

[1] M. Saraiva et al., "Physical DFT for high coverage of realistic faults", Proc. of the ITC, pp.642-651, 1992.

[2] Y.K. Malaiya, and R. Rajsuman (editors), "Bridging faults and IDDQ testing", IEEE CS Press Technology Series, 1992.

[3] E. Cerny, and J. Gecsei, "Simulation of MOS circuits by decision diagrams", IEEE Trans. on CAD, Vol. 4, No. 4, October, 1985.

[4] R.E. Bryant, "Algorithmic Aspects of Symbolic Switch Network Analysis", IEEE Trans. on CAD, Vol. 6, No.4, July, 1987.

[5] R.E. Bryant, "Boolean analysis of MOS circuits", IEEE Trans. on CAD, Vol. CAD-6, No.4, July, 1987.

[6] R.E. Bryant, D. Beatty, K. Brace, K. Cho, and T. Sheffler, "COSMOS: A compiled simulator for MOS circuits", 24th ACM/IEEE DAC, pp.9-16,1987

[7] R.E. Bryant, "Graph based algorithms for Boolean function manipulation", IEEE Trans. on Computers, August, 1986.

[8] S. Malik, A.R. Wang, and R.K.Brayton, A. Sangiovanni-Vincentelli, "Logic verification using binary decision diagrams in a logic synthesis environment", Proc. of the ICCAD'88, pp.6-9, 1988.

[9] K. Cho, and R.E. Bryant, "Test pattern generation for sequential MOS circuits by symbolic fault simulation", 26th. ACM/IEEE DAC, pp. 418-423, 1989.

[10] P. Nigh, and W. Maly, "Test generation for current testing", IEEE Design and Test of Computers, pp. 26-28, Feb. 1990.

[11] C. Ferrer et al., "An approach to the development of an IDDQ testable cell library", IEEE Int'l Workshop on Defect and Fault Tolerance in VLSI Systems, October, 1994.

[12] F.M. Brown, "Boolean reasoning. The logic of Boolean equations", Kluwer Academic Press, 1990.

[13] P. Russell Lamb, "Object-oriented techniques for mixed-mode circuit simulation", Series in Microelectronics Vol. 11, Hartung-Gorre, 1991.

[14] Cohen E., Vladimirescu A., and Pederson D.O., "User's guide for SPICE", Univ. of California, March 1979.

[15] R. Brayton et al., "MIS: Multiple-level interactive logic optimization system", IEEE Trans. on CAD, vol. CAD-6, no.6, pp. 1062-1081, Nov. 1987.

[16] K.S. Brace, R.L. Rudell, and R.E. Bryant, "Efficient implementation of a BDD package", 27th. ACM/IEEE DAC, pp.40-45, 1990.

[17] U.Hübner, and H.T. Vierhaus, "Efficient partitioning and analysis of digital CMOS Circuits", Proc. of the ICCAD'92, 1992.

[18] J.R. Burch, E.M. Clarke, and K.L. McMillan, "Sequential circuit verification using symbolic model checking", 27th. ACM/IEEE DAC, 1990.

[19] Y. Matsunaga, M. Fujita, and H. Tanaka, "Symbolic verification of CMOS synchronous circuits using characteristic functions", Proc. of the CICC, 1991.