A Survey and Comparison of Digital Logic Simulators*

M. Gunes
Department of Computer Science
University of Texas at Dallas
Dallas, Texas U.S.A.

M. A. Thornton, F. Kocan, S. A. Szygenda
Department of Computer Science and Engineering
Southern Methodist University
Dallas, Texas U.S.A.

Abstract— **In this paper, we present a comprehensive survey of digital logic simulators from the past twenty years to present. First, we summarize the digital logic simulation characteristics. Next, we analyze a comprehensive set of simulators for their timing model, simulation mode, logic values, and fault simulation capabilities.**

I. Logic Simulation Characteristics

Several fundamental issues describe the functionality and flexibility of a digital logic circuit simulator. Primary among these considerations is the implementation of the basic simulation mechanism. There are two basic methods for digital logic simulation, compiled and event-driven simulations [23].

A. *Compiled simulation*

A compiled simulator converts the circuit netlist into a sequence of machine language instructions that reflect the functions and interconnections of every element in the circuit. A circuit value table contains an entry that stores the current value of each logic element or net. The instructions obtain input values from this table and store results back into the table. Output values are computed using machine language instructions that reflect the function of the element. Compiled simulation is mainly oriented toward functional verification particularly since it is not concerned with the timing of the circuit. Since compiled simulators do not provide the accurate timing estimates, we will focus on event-driven simulators in the remainder of this paper.

B. *Event-driven simulation*

An event-driven simulator can be implemented to operate on a set of tables that model the circuit. For each circuit node (net), the table has an entry of the logic state and the strength of the source or sources driving the node. An event is said to have occurred when a signal changes state on an input or the output of a circuit element, and is stored in an event queue to be scheduled. The simulator keeps track of the current time and the event queue that holds future events. An evaluation list keeps track of circuit elements whose inputs have changed. The evaluation list is processed after the event queue is processed for that time point. This chain of event-queue processing followed by the evaluation-list processing is a simulation cycle.

One efficient method to increase simulator performance is that of *selective trace* which greatly reduces the number of events to be triggered. It is based on the fact that some of the events may not alter the output of an element, so fan-out of this element is unaltered by the excitation that caused the evaluation. Thus, when an element's output does not change after computing with updated inputs, following its fan-out is unnecessary.

C. *Timing model*

The circuits being modeled rely both on correct logical operation of the components in the circuit and on correct relative timing of signals passing through the circuit. In an event-driven simulation, wires are usually assumed to be without any delay and logic elements are typically assumed to have lumped delay. The timing model of a simulator can vary from zero delay to min/max delay. A *zero-delay* simulator simply calculates the logic function performed by a logic element ignoring delay values within the element. With *unit-delay* simulation each logic element is assigned a fixed delay. A *nominal-delay* simulator assigns distinct delay values to logic elements based on the manufacturer specifications of each element. Finally, the *min/max-delay* model defines the range where the transition may occur by accurate characterization of the bounds on gate and wire delays. Computation of the exact timing with a min/max-delay model is, however, computationally intractable.

The identification of timing characteristics such as races, hazards, and spikes is valuable as these are problematic conditions. A *race* is a situation in which two or more signals are changing simultaneously in a circuit. A *critical race* occurs when an incorrect state is assumed to be stable due to the race. A *hazard* is the possible occurrence of a momentary value opposite to what is expected, and may cause unanticipated events in circuits. A *spike* is produced when a module's inputs change earlier than the propagation delay of the module.

Races and hazards can present problems in zero-delay simulators since they do not indicate when the event occurs. The nominal-delay simulator gives more precise timing results and may catch spikes, but with computational

*This work was supported by the Texas Advanced Technology Program (ATP) grant 003613-0029-2003.

overhead. The unit-delay simulator requires more computations than a zero-delay simulator, and its event scheduling mechanism is simpler than the nominal-delay simulator. The min/max-delay simulator is the most precise, but also the most computationally intense.

D. Simulation modes

Simulation modes vary from abstract high-level to precise low-level, as the simulation gradually becomes more accurate but also complex and time consuming [29].

A *behavioral simulation* models large pieces of a system as black boxes with inputs and outputs. The purpose of a behavioral simulator is to give the designer a general overview of the design to experiment with high-level alternatives.

A *functional simulation* ignores timing and includes unit-delay simulation. A functional simulator processes the input just like the corresponding hardware component, but more flexibility is permitted with respect to how the input is presented to the unit and how it is processed to produce output.

A *gate-level simulation* treats each logic element as a black box modeled by a function where input signals to the element are variables. The function may model the delay through the element as well. Setting all the delays to unit value would result in a functional simulation.

A *switch-level simulation* models transistors as switches being on or off. In switch-level simulators, very little attention is given to the details of other transistor attributes. A switch-level simulator keeps track of voltage levels as well as logic levels. It can provide more precise timing than gate-level simulation, but lacks the ability to use logic element delays as a parameter.

Transistor-level or *circuit-level simulation* requires models of transistors that describe their nonlinear voltage and current characteristics. It is the most accurate, but also the most complex and time-consuming simulation.

Mixed-mode simulation uses different simulation modes for different parts of a circuit to provide more accuracy at critical parts of a circuit and efficiency at the rest. Table I shows some primitives at each abstraction level, and some simulators that work in the specified abstraction level.

E. Simulation Values

Binary simulation uses logic values 0 and 1, and they perform pure binary simulation. *Ternary simulation* adds a third value X to represent unknown or un-initialized.

TABLE I. ABSTRACTION LEVELS

Abstraction	**Primitives**	**Simulator**
Behavioral	Register, Counter, MUX	HDL
Functional	Functional Units, ALU, CPU	GRAFCET
Gate	Gate, Flip-Flop, Memory Cell	MTV, Halotis
Switch	Switch	COSMOS, RSIM
Transistor	Transistor, Resistor, Capacitor	SirSim, STEED
Circuit	Resistor, Capacitor, Current, Voltage	SPICE

The meaning of the fourth value added depends on the simulator. Abstraction Levels For example, Z is often used as a fourth value to represent high impedance, whereas T is used to indicate an over-constrained quantity in Voss and its successors. Rising and falling transitions are taken into consideration in five-valued simulation {0, 1, X, R, F} where R is upward transition and F is downward transition. Common simulation values have increased up to 13 values. MVL9, which uses values {0, 1, U, X, Z, W, L, H, '-'}, is notable as it has been standardized by IEEE (IEEE Standard 1164) [14] and is supported by most commercial HDL simulators.

F. Symbolic simulation

Symbolic simulation employs a built-in model of hardware behavior and a simulation engine that evaluates the behavior of a design for some given input. A symbolic simulator can simulate the response to a set of values with a single simulation run using symbols instead of actual values. The symbolic values encode a range of circuit operating conditions and thus yield more coverage per simulation. For example, in Symbolic Simulation, using *a* and *b* variables along with binary values at inputs of a gate-level model, the simulator finds $a \bullet b'$ as an output result.

Symbolic simulators can also utilize accurate electrical and timing models to compute circuit behavior as logic simulators. These models may include a detailed switch-level model containing charge sharing and subtle strengths phenomena and timing models containing bounded delay assumptions.

G. Symbolic trajectory evaluation

Symbolic trajectory evaluation is different from conventional symbolic simulation as it verifies the validity of formulas expressed in a limited, but accurately defined temporal logic. Using temporal logic, properties of a circuit are expressed over trajectories that are bounded-length sequences of circuit states [12]. Symbolic trajectory evaluation uses partially ordered four-valued logic.

H. Fault Simulation

Fault simulation analyzes the behavior of a circuit when faults exist in the circuit. Fault simulation is used to create fault dictionaries and in evaluating proposed test sets [23].

I. Fault models and fault insertion

The system may simulate behavior of faulty logic design by injecting faults into logic circuit and running logic simulation to determine the faulty behavior. A fault simulation system becomes more powerful as it provides more fault models to be analyzed. The system should be capable of inserting various faults such as the stuck-at fault into the model. One such technique is to insert a stuck-at one or zero on the output signal from a logical element.

II. Logic Simulators

In this section we survey various timing simulators that are used for design verification. We outline their simulation level, logic values, delay mode and fault simulation capabilities. The list is not a complete set of logic simulators, but is meant to be a representative set.

A. *TEGAS2*

TEGAS2 is a multi-modal, five-valued simulator [13,29]. It performs gate-level and functional-level simulation. Nominal and critical timing (min/max) delays are used in simulation. Simulation values are {0, 1, U, D, E}.

TEGAS2 has a fault simulation ability by fault generation and injection into the simulated circuit. Provided fault models are stuck-at, shorts, transient fault models, and multiple faults. Performance is improved by parallel simulation of faults where a specified number of faults are simulated in one pass. The number of faults per simulation is determined from indistinguishable fault classes, fault blocking characteristics, and the desired diagnostic resolution.

B. *MOSSIM*

MOSSIM is a unit-delay, ternary, switch-level simulator [5]. In addition to ternary node values, each transistor has three states, open, closed and unknown. In each pass, MOSSIM performs repeated iterations of signal flow model to simulate the behavior of the system with applied inputs. During an iteration, transistors are marked as closed, open or unknown based on the logic levels of their gate nodes. Next, signal flow between interconnected nodes is computed to form new logic levels, and the affected nodes are updated. These iterations are repeated until all nodes have a consistent steady state.

C. *FMOSSIM*

FMOSSIM is the extension of MOSSIM with concurrent fault simulation capability [7]. Similarly, FMOSSIM is a unit delay, ternary, switch-level simulator. FMOSSIM can inject node stuck-at and transistor stuck-open or stuck-closed faults into the model for fault simulation. The simulator also utilizes concurrent fault simulation using isolated lists for each node. Separate lists are used to keep track of the state for every faulty circuit.

D. *Abramovici et al.*

Abromovici et al. proposed a logic simulation machine utilizing distributed and parallel processing [1]. The proposed architecture can perform simulation at both the functional and gate level. The system is re-configurable and expandable to handle different logic values, delay models, and types of timing analysis. It utilizes a distributed processing architecture operating on a pipeline type data flow. Each task of the system is assigned to a dedicated processing unit. The processors are micro-programmable so that all the tasks are implemented in micro-code. Functional evaluation at a functional-level is realized by using parallel processors called functional evaluators.

E. *RSIM*

RSIM is a ternary switch-level simulator where transistors are modeled as switches with series resistors [17,18,30]. Using the model, a network of transistors and nodes is simulated as a network of resistors and capacitors. The network is a collection of small stages whose operation can be predicted.

RSIM uses a linear model as an approximation, and considers the effects of the input waveform timings on gate propagation delay. It combines logic-level simulation with automatic estimation of transition times from electrical properties of the circuit components. Node voltages and transition times are determined using three resistances that are determined by experiments. Node values are determined by the voltages calculated.

F. *IRSIM*

IRSIM is an extension of RSIM with an incremental simulation capability [25]. IRSIM performs ternary, switch-level simulation. Incremental simulation is achieved by maintaining a history of circuit activity during simulation and selectively simulating the portions of the circuit that deviated from their history. IRSIM uses a restructured evaluation routine to improve timing and charge-sharing.

G. *Mokkarala et al.*

In [24], Mokkarala et al. analyzed an integrated tool for timing verification and logic simulation. The timing verifier uses nominal delay values for input ports, output ports, and paths for each block. The logic values are classified as {0, 1, X, Z, R, S, C} for verification and {0, 1, X, U} for simulation.

The system works in a mixed-mode of functional, gate and switch levels. Functional verification utilizes functional description of the component blocks of the design.

H. *Lsim*

Lsim is a mixed mode simulator that combines functional, gate, switch and circuit level simulators [9]. Simulation is handled using three delay modes; zero, unit, nominal and min/max. It uses seven logic states that are divided into the two categories of "stable" using {High, Low, z, x} and "transient" using {r, f, t} logic values.

Stable states are used in all timing modes whereas transient states are used in variable (min/max) delay mode to represent intermediate states during a transition. For better modeling, *strength* is associated with each signal besides its logical state. There are two strength levels, strong and weak, corresponding to a high and low current drive capability.

After its commercial development by Mentor Graphics, Lsim went through some changes. The signals are represented by three components: logic state, signal strength and weight. Logic states are reduced to low, high and unknown. Signal strength uses five different values to represent the mechanism used to drive the signal line.

Signal weight represents how strongly the signal is driven using a value in the 00-31 range. For charged strength, the weight represents charge storing capacity, whereas it represents conductance for driven strength. Valid combinations of signal state, strength, and weight are given in Table II.

TABLE II. SIGNAL STRENGTH AND STATES IN LSIM

Graphics	**State**		
Strength	Low (L)	High (H)	Unknown (U)
Initial (I)	IL00	IH00	IX00
Charged (C)	CL00-CL31	CH00-CH31	CX00-CX31
Driven (D)	DL00-DL31	DH00-DH31	DX00-DX31
Supply (S)	SL31	SH31	SX31

I. *SIMMOS*

SIMMOS, an extension of MOSSIM, is a mixed mode simulator that operates at behavioral, gate and switch levels [2]. It utilizes a nominal delay mode for timing analysis. It performs ternary logic simulation with 32 strength levels.

SIMMOS performs statistical fault analysis for fault verification and test grading to detect single stuck-at faults based on STAFAN [15]. The method uses probability theory to construct equations for observability, controllability and input sensitization probabilities. These equations are used to produce probability of each signal being stuck-at-1 and stuck-at-0.

J. *COSMOS*

COSMOS is a unit-delay, ternary, switch-level simulator [6]. COSMOS consists of three modules, namely ANAMOS, LGCC, and a C compiler. ANAMOS partitions switch-level circuits into channel-connected sub-networks, and derives Boolean models for each sub-network's behavior. LGCC translates this representation into the C language. Finally, the C compiler simulates the code produced by LGCC with a simulation kernel.

The ANAMOS symbolic analyzer [3] captures bi-directional transistor, stored charge, different signal strengths, and indeterminate logic features of the switch-level network. Transistor strength varies from 3 to 6 for typical circuits. A system of Boolean equations is used to analyze the network using Gaussian elimination.

K. *TRANALYZE*

TRANALYZE transforms transistor-level networks to the gate-level which gives a combination of switch-level generality with gate-level compatibility and performance [4]. It generates the gate model using methods derived from switch-level simulation. Similar to ANAMOS, TRANALYZE performs a symbolic, switch-level analysis to extract a logic representation of the circuit behavior. Instead of binary encoding of signals, TRANALYZE uses a four-valued algebra for the generated gate model using values {0, 1, X, Z}.

In terms of timing support, TRANALYZE has two modes of operation, zero delay with Boolean optimization and unit delay with conservative X modeling.

L. *Voss*

Voss is a formal verification system that supports symbolic trajectory evaluation (STE) with built in OBDD support [26]. Voss operates at switch, gate and behavioral levels. The switch-level model is the same as COSMOS. The system handles various timing models, i.e. unit, nominal, min/max and bounded delay. Four logic values are used in the system {0, 1, X, $\top$} are encoded using a dual rail (H, L).

M. *FORTE*

FORTE is Intel's FORmal Tools Environment and is built on the Voss system that uses BDDs with symbolic variables at circuit inputs [10]. In addition to STE for simple logic, it supports bSTE for Boolean expression graphs and iSTE for scalar simulation.

N. *SirSim*

SirSim [21] is a transistor-level symbolic timing simulator based on the delay calculation procedures in IRSIM. Symbolic timing simulation is an extension to symbolic simulation that was developed for functional verification. Symbolic simulation performs input pattern independent simulation using Boolean variables instead of constant 0 or 1. The simulator handles data-dependent delay in addition to zero, unit and nominal delay models. Delay values are computed using MTBDDs to represent data-dependent delay values, and masks indicate the logic conditions under which events happen.

O. *STEED*

STEED [22] is a transistor-level symbolic timing simulator extending SirSim. Mainly, STEED solves the event multiplication problem of SirSim. In the data-dependent delay mode of SirSim, the effect of a single event multiplies at each level because new node values are scheduled for each potential delay. STEED uses node value and delay-calculation engines from SirSim with an event management methodology based on event clusters. An event cluster is defined as a set of events on a single node with additional parameters

P. *Krishnaswamy et al.*

Krishnaswamy et al. presented a ternary, switch-level simulator with fault simulation capabilities [19]. The simulator extracts a circuit model similar to the COSMOS system.

Q. *HALOTIS*

HALOTIS [31] is a gate-level logic timing simulator with results comparable to electrical simulators using Inertial and Degradation Delay Model (IDDM). IDDM combines the degradation effect of glitches with inertial effect handling [16]. In order to improve accuracy, transition from 0-to-1 or 1-to-0 is approximated by a linear curve determined by the rise or fall time and transition start time. Transitions may trigger an event at the gate's threshold voltage, which is a predefined value associated to the gate input. Gate outputs may be stable at logic 1 or logic 0, or at a transition from 0-

to1 or 1-to-0. Thus, four simulation values {0, 1, R, F} are used.

R. Silos

Silos is a logic simulator operating at behavioral, gate and switch levels [27]. Silos supports the IEEE standard delay format in addition to zero, unit and nominal delays. Silos is four valued {0, 1, X, Z} simulation, but also defines signal strength. Signal strengths are Supply, Driving, Resistive, High-Z, and additional three internal states Uncertain, Decaying and Spike. Strength-Value pairs are shown in Table III.

TABLE III. SIGNAL STRENGTHS IN SILOS

Strength	Low	Unknown	High	High Voltage
Supply	0	X	1	1
Driven	0	X	1	1
Resistive	0	X	1	1
High-Z	Z	Z	Z	Z
Uncertain	X	X	X	X
Decaying	D	D	D	D

S. ModelSim

ModelSim is a functional verification and debugging system that simulates designs at the gate and RTL levels [23]. The simulator uses MVL9, IEEE Standard 1164.

III. COMPARISON OF SIMULATORS

Table IV outlines analyzed simulators. For each simulator, simulation level, simulation values and timing modes are indicated. If fault simulation is available, utilized fault models are indicated. Finally, any other significant feature of the simulator is pointed out as a comment.

IV. CURRENT NEEDS IN SIMULATION

Many of the analyzed simulators perform static timing analysis, but lack critical timing simulation. A logic simulator that takes the dynamic effects with SPICE-level accuracy would be beneficial in analyzing high-density circuits.

Some of the simulators are capable of fault simulation; the faults modeled are primitive ones. Efficient fault simulators capable of handling other faults such as I_{DDQ}, path-delay, or crosstalk fault models are needed as well.

More timing accuracy requires more computation during simulation. Thus, simulators should benefit from concurrent and distributed simulation to meet the computational needs of accurate simulators. Abromovici et al. mentions distributed and parallel processing, but they propose a machine particularly designed for logic simulation. However, a system that uses any grid of concurrent computers is desirable.

[1] M. Abramovici and Y. H. Levendel and P. R. Menon. A logic simulation machine. In *Proceedings of the 19th conference on Design automation*, 1982, pages 65-73.

[2] D. Adler. SIMMOS: a multiple-delay switch-level simulator. In Proceedings of the 23rd ACM / IEEE conference on Design automation, 1986, pp.159-163.

[3] R.E. Bryant. Boolean analysis of MOS circuits. In IEEE Transactions on Computer Aided Design of Integrated Circuits, February 1987, pp. 634-649.

[4] R.E. Bryant. Extraction of Gate Level Models from Transistor Circuits by Four Valued Symbolic Analysis. In *International Conference on Computer Aided Design*, November 1991, pp. 350-353.

[5] Randal E. Bryant. MOSSIM: A switch-level simulator for MOS LSI. In *Proceedings of the 18th conference on Design automation*, 1981, pp. 786-790.

[6] R. E. Bryant, D. Beatty, K. Brace, K. Cho, and T. Sheffler. COSMOS: a compiled simulator for MOS circuits. In *Proceedings of the 24th ACM/IEEE conference on Design automation*, 1987, pp. 9-16.

[7] R.E. Bryant and M.D. Schuster. Performance evaluation of FMOSSIM, a concurrent switch-level fault simulator. In *Proceedings of the 22nd ACM/IEEE conference on Design automation*, 1985, pp. 715-719.

[8] S. Chakraborty and D.L. Dill. More accurate polynomial-time min-max timing simulation. In *Proceedings of 3rd International Symposium on Advanced Research in Asynchronous Circuits and Systems*, April 1997, pp.112-123.

[9] R.D. Chamberlain and M.A. Franklin. Collecting fata about logic simulation. In *IEEE Transactions on Computer-Aided Design,* July 1986, pp. 405-411.

[10] Forte/FL User Guide, Version 1.0. By *Technical Publications and Training, Intel Corporation*, Jan 2003.

[11] S. Hazelhurst and C-J. Seger. A Simple Theorem Prover Based on Symbolic Trajectory Evaluation and BDDs. In *IEEE Transactions on Computer-Aided Design*, April 1995, pp. 413-422.

[12] S. Hazelhurst, C-J. Seger. Symbolic Trajectory Evaluation. In *Formal Hardware Verification - Methods and Systems in Comparison*, 1997, pp. 3-78.

[13] C. W. Hemming and S. A. Szygenda. Modular requirements for digital logic simulation at a predefined functional level. In *Proceedings of the ACM annual conference*, 1972, pp. 380-389.

[14] IEEE Std 1164-1993, IEEE Standard Multivalue Logic System for VHDL Model Interoperability (Std_logic_l 164). *IEEE Computer Society*. 1993

[15] S.K. Jain and V.D. Agrawal. STAFAN: An alternative to fault simulation. In *Proceedings of the 21st conference on Design automation*, 1984 pp. 18-23.

[16] J. Juan-Chico, P. Ruiz-de-Clavijo, M.J. Bellido, A.J. Acosta, M. Valencia. Inertial and degradation delay model for CMOS logic gates. In *Proceedings of IEEE International Symposium on Circuits and Systems*. May 2000, pp. 459-462.

[17] R. Kao and M. Horowitz. Piecewise linear models for Rsim. In *Proceedings of the 1993 IEEE/ACM international conference on Computer-aided design*, 1993 pp. 753-758.

[18] R. Kao and M. Horowitz. Timing analysis for piecewise linear Rsim. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1994, pp. 1498-1512.

[19] V. Krishnaswamy and J. Casas and T. Tetzlaff. A switch level fault simulation environment. In *Proceedings of the 37th conference on Design automation,* 2000, pp. 780-785.

[20] C.B. McDonald and R.E. Bryant. Symbolic functional and timing verification of transistor-level circuits. In *Proceedings of the 1999 IEEE/ACM international conference on Computer-aided design*, 1999, pp. 526-530.

[21] C. B. McDonald and R.E. Bryant. Symbolic timing simulation using cluster scheduling. In *Proceedings of the 37th conference on Design automation*, 2000, pp. 254-259.

[22] A. Miczo. Digital Logic Testing and Simulation, Second Edition. By *John Wiley & Sons, Inc.* New Jersey. 2003.

[23] ModelSim SE, PSL Assertions Guide. Version 5. 8 Beta. By *Model Technology*, August 2003.

[24] V.R. Mokkarala, A. Fan, and R. Apte. A unified approach to simulation and timing verification at the functional level. In *Proceedings of the 22nd ACM/IEEE conference on Design automation*, 1985, pp. 757-761.

[25] A. Salz and M. Horowitz. IRSIM: an incremental MOS switch-level simulator. In *Proceedings of the 26th ACM/IEEE conference on Design automation*, 1989, pp. 173-178.

[26] C. Seger, VOSS - A Formal Hardware Verification System User's Guide. *Technical Report 93-45, Department of Computer Science, University of British Columbia,* 1993.

[27] Silos User's Manual Version 2002.1. By *Simucad Inc*., 2002.

[28] M.J.S. Smith. Application-specific integrated circuits. *Addison-Wesley Longman Publishing Co., Inc.* Boston. 1997.

[29] S. A. Szygenda. TEGAS2—Anatomy of a General Purpose Test Generation and Simulation system for Digital Logic. In *Proceedings of the 9th workshop on Design automation*, 1972, pp. 116-127.

[30] C.J. Terman. Simulation tools for digital LSI design. *PhD Thesis, Massachusetts Institute of Technology*, 1983.

[31] R. Vazquez, J. Juan-Chico, M. Bellido, A. Acosta, and M. Valencia. HALOTIS: High Accuracy LOgic TIming Simulator with inertial and degradation delay model. In *Proceedings of the conference on Design, automation and test in Europe*, 2001, pp. 467-471.

TABLE IV. SUMMARY OF DIGITAL LOGIC SIMULATORS

NAME	**Year**	**Simulation Level**	**Simulation Values**	**Timing Modes**	**Fault Model**	**Comments**
TEGAS2	1972	Functional, Gate	0,1,U,D,E	Unit, Nominal, Min/Max	Stuck-at, Shorts, Transient faults	
MOSSIM	1981	Switch	0,1,X	Unit		
FMOSSIM	1985	Switch	0,1,X	Unit	Stuck-at, Stuck-open, Stuck-closed	
Abramovici et.al.	1982	Functional, Gate	Any	Any		Dedicated logic simulation machine
Mokkarala et.al.	1985	Functional, Gate, Switch	0,1,X, {U} | {Z,R,F,S,T}	Nominal		Integrates Simulator and Verifier
RSIM	1983	Switch	0,1,X	Unit		
IRSIM	1989	Switch	0,1,X			
LSim	1986	Functional, Gate, Switch, Circuit	0,1,X,Z,R,F,T	Zero, Unit, Nominal, Min/Max		Signal strength level
SIMMOS	1986	Behavioral, Gate, Switch	0,1,X	Nominal	Stuck-at	
COSMOS	1987	Switch	0,1,X	Unit		Signal strength level
TRANALYZE	1991	Switch	0,1,X,Z	Zero, Unit		
Voss	1993	Behavioral, Gate, Switch	0,1,X,T	Unit, Nominal, Min/Max, Bounded		
FORTE		Behavioral, Gate, Switch	0,1,X,T	Unit, Nominal, Min/Max, Bounded		Symbolic variables
SirSim	1999	Transistor	0,1	Zero, Unit, Nominal, Data-dependent		Symbolic variables
STEED	2000	Transistor	0,1	Zero, Unit, Nominal, Data-dependent		Symbolic variables
Krishnaswamy et.al.	2000	Switch	0,1,X	Zero, Unit	Stuck-at, Bridging, Transition	
HALOTIS	2001	Gate	0,1,R,F	IDDM		Inertial and degradation delay model
Silos		Behavioral, Gate, Switch	0,1,X,Z	Zero, Unit, Nominal		Seven signal strengths
ModelSim		RTL, Gate	0,1,X,Z,U,W,L,H,'-'			