SIMMOS: A Multiple-Delay Switch-Level Simulator

Dan Adler

Motorola Semiconductor Israel (MSIL)
147 Bialik St., Ramat-Gan 61047, Israel

Abstract

SIMMOS is a multiple-delay logic simulator for MOS VLSI circuits based on the switch-level model. In addition to finding the ternary logic state at each node, SIMMOS estimates the time delay required for that state to become valid. The delay calculation method, based on the theory of RC trees, is introduced as a natural extension of the dominant-path algorithm used for node state evaluation.

Multi-level simulation in SIMMOS is achieved by using special models for gate-level primitives, and the ability to drive, and be driven by an RTL simulation environment.

For test-pattern grading, SIMMOS uses a probabilistic fault analysis algorithm, modified to operate on bidirectional as well as gate-level models.

Introduction

The design of VLSI chips, which may presently contain over 100,000 transistors, requires special simulation and checking tools to verify all aspects of the design process. If a single simulator is to carry the design through all its stages, it must be very flexible in its ability to model the circuit at different levels. Often, parts of a design may only have a behavioral model, while other parts may have gate-level models without transistor sizes and some parts may even be fully laid-out. The simulator must deal with the circuit as a whole, modeling different parts in RTL, gate-level or switch-level using a single test pattern.

Such a simulator was developed at MSIL, in an effort to provide the block-design engineers with a verification tool that can evolve with the design, and effectively model the circuit at each level. It consists of an RTL simulator and a gate-and-switch level simulator, which operate under a common shell and interface through a set of shared nodes. This paper describes the gate-and-switch level simulator called SIMMOS, which offers several improvements over the MOSSIM switch-level algorithm developed by Bryant [1].

SIMMOS extends Bryant's MOSSIM algorithm to deal with stages (nodes connected drain-to-source by conducting transistors) that are driven by gate-level primitives, and takes a more realistic approach to signal timing by computing RC delays and introducing a scheduling mechanism.

In MOSSIM, a ternary node state is found by tracing dominant charging and discharging paths to each node in a stage, and comparing their relative strengths. In terms of timing, MOSSIM utilizes an internal unit-step model. All state transitions within a stage occur instantaneously, creating internal events that cause other stages to be evaluated. The process continues until no more internal events are generated, and then the simulation clock is incremented.

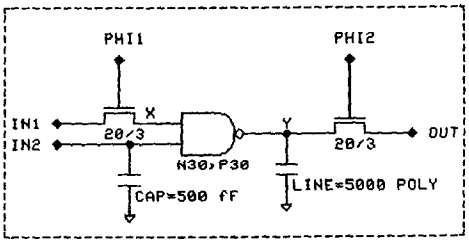
This highly idealized timing model has often been criticized as one of MOSSIM's weaker points, since logic verification with no timing information may fail to detect serious design errors. Problems such as signal delays, spikes, races, transient unknowns and synchronization in multi-clock chips cannot be modeled in MOSSIM, and are very difficult to diagnose even in a fabricated chip.

SIMMOS uses the fact that Bryant's dominant-path algorithm induces a tree structure over a stage to compute a dominant-path delay according to the TREE algorithm described in [4]. Several ad-hoc correction factors are used to deal with simple (one-transistor deep) parallel paths, slow-changing gate (rise-time ratio [5]), gate not fully turned on etc.

The Network Model

The basic network configuration and data structures of MOSSIM are retained, along with parts of the simulation algorithm. We assume the reader is familiar with [1], so we will not repeat the details here.

In SIMMOS, a transistor is described by its type, connections and W/L dimensions. The dimensions are mapped into two parameters: a resistance calibrated for timing, and a logic strength for contention. Similarly, each node is assigned a capacitance, which is used for timing calculation and is also mapped into a logic strength for charge sharing events.




```
SIMMOS NETWORK :

PHI1    .N      20/3      IN1   X
Y       .NAND   N30/P30   IN2   X
PHI2    .N      20/3      OUT   Y
.CAP    IN2     500
.LINE   Y       5000 POLY
```

Figure 1: A SIMMOS network example.

0738-100X/86/0000/0159$01.00 ©1986 IEEE

SIMMOS uses 32 different strengths, which are defined by the user in terms of W/L or node capacitance. Using a large number of strengths has several advantages: the mapping of actual parameters into strengths is realistic, large capacitive nodes can override weak transistors and transient X-states can be minimized (an important "escape" mechanism). Also, no ad-hoc models need be created to deal with exotic circuits, as is often the case in gate-level simulators.

A network file can be generated automatically either from a layout extraction program [8] or from work-station schematics. In the former case, all relevant information is supplied in the network file: transistor dimensions and connectivity and parasitic node resistance and capacitance. When a network is created from schematics, several preprocessing steps take place. First, the hierarchy is removed and the design "flattened". The network in this case consists of both transistors and logic-gates which are defined by their type, W/L dimensions and connectivity. Next, all active-device loads are estimated by the program, based on a parameter file and the connectivity, and added to the interconnect loads specified by the user.

Internally, the network is represented in three main linked-lists: a node list, a logic-gate list and a transistor list. The connectivity lists for each node are similar to MOSSIM, except that gate-fanout and gate-fanin lists are also retained. Node states are 0, 1 or X, with some additional symbols used to mark nodes that are changing, nodes that suffer a threshold drop, and nodes that are decaying (tri-state).

Switch-Level Delays

Switch-level simulation requires that groups of nodes connected drain-to-source by conducting transistors be evaluated collectively, through a relaxation process. This slows down the simulation relative to gate-level programs and complicates delay modeling.

Some switch-level timing simulators, like RSIM [2] and VTIsim [3], use a Thevenin-equivalent model to represent a stage, relative to each node, as a linear resistor and a voltage source. The Thevenin resistor is then multiplied by the node capacitance, yielding an RC delay for the node. The RSIM algorithm is more accurate than MOSSIM in solving competing drivers, but it poses two major problems. One is that solving a general resistive network with respect to each node of the stage is a heavy computational task, and the other is that the delay estimate does not take into account how capacitances are distibuted in the stage.

Bryant's MOSSIM algorithm is computationally simple, linear and tree-based. It requires three relaxation steps per stage: one to determine the blocking strength at each node, and two more to find the strongest charging path and the strongest discharging path. An important observation is that, during the last two steps, we always traverse the transistors in the stage from the driving end (strongest node) of the stage towards the loading end, thus tracing through the actual charging/discharging path to each node. We can then compute local RC delays and accumulate them over the path, resulting in a calculation method similar to the TREE algorithm described in [4].

An inherent limitation of the TREE algorithm is that it cannot be applied to a non-tree network because the driving and loading networks of a node in such a network are not explicit. This can cause serious timing errors in the evaluation of stages containing parallel paths. A general solution for RC networks has been developed by Lin and Mead [4], using a technique called load redistribution, which requires that a stage be decomposed into a *number* of tree networks. In SIMMOS, we have adopted a less general solution, which is limited to dealing with parallel paths that are one transistor deep. This covers most practical cases such as CMOS transfer gates, multi-input logic gates and PLAs.

The method used to deal with simple parallel connections in SIMMOS is by not cosidering such paths blocked, even though they are disregarded by Bryant's algorithm. When SIMMOS detects simple parallel connections, it alters the blocking strength of the node farther down the path enabling it to overpower an identical single transistor. Also, the resistance used for delay calculation is recomputed to reflect the parallel connection.

Following a stage evaluation, 'projected' values for all node states are found, and the nodes are scheduled according to their delay. An important question is what value should the node assume during this transition time. An X-state would be very pessimistic, since an unknown value has a tendency to spread around the circuit with harmful results. A more conservative solution is to leave the node in its old state until the new value is scheduled. This solution is the simplest to implement, but it requires that the delay be defined as the time between when the input becomes valid until the output *begins* to change, which is a somewhat vague definition as it does not apply equally to all configurations.

Another related problem is event descheduling. Often, while a node is changing towards one value, a later event may cause the stage to be reevaluated. The node may then have to be rescheduled with a new projected value. If the more recent projected value is different than the older one, a spike or race are reported and the old projected value is discarded. The case of a new projected value that is equal to the old one is more difficult to deal with. When the second event occurs, the scheduled node is already in transition, so the delay must reflect the reduced voltage swing.

One way of dealing with this problem is to consider initial-charge as an additional parameter in node delay calculation [4]. Another is to schedule transitions as voltage ramps [3], so that when such a case occurs, the time-constant is multiplied by a speed-up factor which takes into account how long the node had already been in transition when it was rescheduled. We are currently investigating both approaches in SIMMOS, since we have found this to be the most serious cause of overestimation of path delays. The most common path of this kind is a CMOS transfer-gate operating in synchronous mode driving a large capacitive load. If one of the gates turns on after the other, the equivalent driving strength increases *after* the drain node has already started its transition, so the speed-up is very significant.

The rise/fall delay for a transistor output node is defined as the time until the next stage starts to change. This is different for source-drain connections (transistor stages) than for gate-triggered transitions. In the latter case, the delay time is selected as the time for the second transistor's output to reach a certain voltage. This voltage is taken to be the minimum voltage the user might consider a glitch. The calibration factors are extracted from circuit simulation for stages with indentical and balanced (rise and fall) W/L values.

Simulation primitives

Gate-level modeling is an important feature of SIMMOS. It saves a great deal of CPU time in comparison to full switch-level simulation, and makes the simulation data-base more conformant with the schematics, therefore easier to debug.

When a node is only driven by unidirectional logic-gates, SIMMOS uses a table lookup method to evaluate the output state and a gate-delay procedure to compute the delay. Gate delays are a function of the W/L dimensions, the gate-type and the projected output state. Gates which have serial/parallel connections of transistors are scaled accordingly, so that the RC constant reflects the equivalent resistance as seen from the gate output node.

Logic gates that are part of a transistor stage are treated differrently, as part of the stage-evaluation process. When a perturbed node has bidirectional transistor connections, a stage is created based only on the transistor connectivity lists. The first step in Bryant's relaxation algorithm is to find nodes connected to signal sources by scanning the input-connectivity lists. At this point each node's gate-fanin list is also scanned and treated as an extension of the input-connectvity list. A special lookup table similar to the gate-output evaluation table is used to find the gate path-blocking strength, its up-driving and down-driving strengths. A gate-delay evaluation is computed only if the gate-output node's current state is different than the gate's driving state.

An important requirement is that a stage driven by a logic-gate, and a similar stage comprised only of switch-level transistors produce identical results, both in terms of logic states and delays. To achieve this, SIMMOS tries to distinguish between stages that behave like distributed RC trees, and those that behave more like a Thevenin source driving a capacitance through an equivalent resistor.

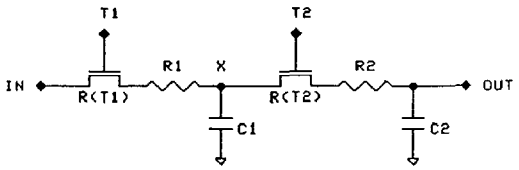


Figure 2: Pass-transistor chain model.

When C1<<C2 and R1 is much smaller than T1's on-resistance, we consider the two transistors as one, with an on-resistance equal to R(T1)+R(T2) driving a capacitance C1+C2. This corresponds to the case where T1 and T2 are very close physically, e.g. when they are part of a logic-gate described at full switch-level. The second case is when the two transistors are separated by a large capacitance C1, such that the delay [R(T1)+R1] x C1 is not negligible. In this case the delay at OUT is estimated using the RC-TREE approach.

SIMMOS makes the distinction between these cases at pre-processing, by marking nodes as "intermediate nodes". These are nodes with a small capacitance, which are connected to only one drain and one source of two different transistors. At run-time, their on-resistances are accumulated on the driving path, until a non-intermediate node is reached - and only then is the time constant computed. We have found that this method produces almost identical results for multi-level and pure switch-level runs, and models pass-transistor chains more accurately.

Pass transistor chains also affect the logic strength of signals. When such a chain is encountered, besides finding the combined on-resistance of the whole path, SIMMOS also recomputes the logic strength of non-intermediate nodes, thereby causing the "strength" of the signal at such nodes to decrease. This option can also be over-ridden by the user for pure logic simulations.

Special cases

Several ad-hoc mechanisms exist for treating special cases. One such case is the scheduling of an X-state. When contention occurs at some node, an unknown value is scheduled. Care must be taken not to let transient X-states ruin a simulation, since they are always treated too pessimistically by ternary simulation algorithms. Instead of scheduling an X-state immediately, SIMMOS schedules it as if it were a full swing state transition, preserving the 'old' state. If the X-state is descheduled before its delay is over, the new event is scheduled as if the node were a partially charged node as explained earlier.

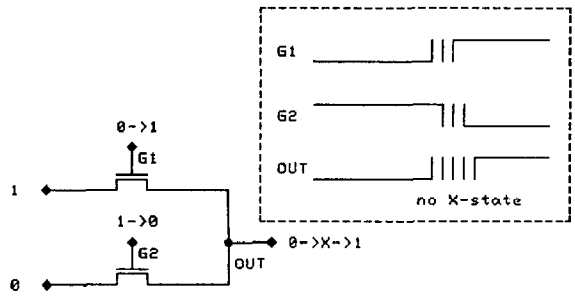


Figure 3: X-state scheduling example.

Scheduling transitions from an X-state to a valid state are also problematic. Taking the maximum of the low-to-high delay and the high-to-low delay produces over-pessimistic timing in many situations. For example, modeling a chain of inverters whose input becomes valid, using only low-to-high delays (or vice-versa) will produce a delay that is grossly exaggerated. The least of all evils seems to use minimum delays in such cases, or to perform an initialization routine to flag nodes whose states must be opposite.

Other special cases treated by SIMMOS are threshold-voltage drops suffered by a node driven through a pass transistor. If such a node drives a transistor gate, it multiplies its on-resistance by some predetermined slow-down factor. Slow rising/falling gate states are treated using the concept of rise-time ratio [5]. This applies only when a gate transition causes the stage to be evaluated, and only to that transistor. Finally, clocked bootstrap-drivers are recognized and do not produce a threshold drop.

One important case that SIMMOS cannot deal with is the precise timing of ratioed transistors driving a stage from multiple sources. This is very common in NMOS designs, and therefore SIMMOS is only used in timing simulation mode on CMOS designs where such situations are rare and usually insignificant.

Event scheduling

The event-scheduling mechanism used in SIMMOS is very similar to the one used in most gate-level logic simulators. A time-queue is maintained with a basic resolution of one nanosecond. Nodes are placed in a time

slot according to their delay and, if they are not descheduled in the meanwhile, they assume their projected value when that time slot becomes the current-time. Special care must be taken to avoid reevaluating a stage whose nodes are placed on the time-queue. To this end, a boolean flag is kept for each node, marking whether the stage it belongs to needs to be reevaluated or not. This flag is set to one if an event occurs that has some effect on the node or the states of its linked transistors. If the flag is zero, only the transistor fanout list and logic-gate fanout list need to be perturbed.

Faster Simulation

Switch-level simulation is inherently slower than gate-level, and the multiple-delay mode slows it down even more. Retaining logic gates as functional elements whenever they occur saves a great deal of CPU time in event processing and stage manipulation, but for large designs even this is not enough.

A speed-up method used in SIMMOS to reduce large redundant stages is parallel-connection extraction. It applies to PLAs and large NOR gates where many transistors are connected in parallel, usually as pull-downs. At preprocessing, such transistors are identified and merged into one large transistor whose gate state is an OR function of the other gate states. The result is identical in terms of the logic state, and yields a constant worst-case delay. This can be done regardless of whether the network originates from layout or from schematics, since it is based solely on the connectivity. In well structured VLSI blocks, the number of simulated transistors can thus be greatly reduced with a similar effect on runtime.

Fault analysis

Fault coverage is becoming an increasingly important factor in the design-verification phase, as well as for test-pattern grading. Switch-level fault simulation [7] techniques have been developed, but their practicality in a VLSI design environment is questionable. Another promising developement in the field of fault modeling is the concept of statistical fault analysis [9]. This method, originally developed for gate-level designs, yields results comparable to exhaustive stuck-at fault simulation, with very little increase in run-time over a single "good-machine" simulation.

The algorithm uses probability theory to construct equations for the propagation of observabilities through the network. These, together with controllabilities and gate-input sensitization probabilities which are computed during the course of each regular simulation session, are manipulated to yield a probability for each signal to be stuck-high detectable and stuck-low detectable, for the given set of test vectors.

A similar technique has been implemented in SIMMOS using a simplified model to compute observability propagation through bidirectional transistors. The simplification is in assuming that although the transistors are treated as bidirectional during the simulation, they are in fact unidirectional in the sense that they always conduct in the same direction. This is an acceptable premise for most designs, and the results are not greatly affected if a small number of transistors are truly bidirectional. Two additional parameters are required per transistor, a 1-pass probability and a 0-pass probability. The 1-pass probability is a measure of the probability of the transistor being on and passing a high value, and the 0-pass probability is, likewise, the probability of an on-transistor passing a low. These parameters are updated during the regular simulation session, and are used to compute the observabilities later on.

Detailed algorithms cannot be included here, but they follow closely the computation method described in [9]. The fact that full scale fault coverage can be achieved so quickly, without need to simulate the "bad" machines, compensates for a few inaccuracies in the detectability of certain nodes. The overhead required while running the "good machine" is quite small, consisting mainly of updating a few counters per event.

The run-time command FSIM triggers the evaluation of observabilities and consequently, the fault analysis. It can be repeated at different timepoints, to see how the coverage is proceeding and whether more tests are required.

Performance

SIMMOS has been used mainly as a block-verification tool in the design of the MC68605 X.25 Controller and the MC68824 Token-Bus Controller. It is currently being used in the design of several other VLSI chips at MSIL. It has uncovered many design errors that would normally have been discovered only at the silicon-testing phase such as complex timing problems, signals driven "in the wrong direction", logic errors caused by bad sizing of transistors, race conditions etc. Timing estimates in SIMMOS are generaly within 20% of (worst case) circuit simulation results, with ad-hoc solutions for problematic configurations like pass-transistor chains, bootstrap drivers etc. producing a consistent level of accuracy.

Since SIMMOS treats certain configurations (like PLAs, decoders, logic-gates etc.) differently; the number of events-per-second is strongly dependent on the type of circuit being simulated, and therefore not a good indicator. We have, however, found that SIMMOS runs only slightly slower than our version of MOSSIM II (which may not be an optimal implementation), and about 2-3 times slower than a commercial gate-level software simulator for large designs.

Conclusion

A multiple-delay switch level simulator has been described that uses a mixture of gate-level and switch-level concepts in a single environment.

Used in conjunction with RTL simulation, SIMMOS provides a means of simulating the design at the lowest level available at each stage of the design. The common simulation mode allows any part of a single design to be described in behavioral, gate or switch levels.

Process-dependence, as well as temperature, voltage level, short channel effects etc. are taken into account in the form of approximate constant factors used to compute transistor on-resistance and node capacitance and resistance to produce worst-case timing.

A probabilistic fault analysis module is used for fast and accurate fault verification and test grading by finding all single-node-stuck-at faults.

Recently, a timing-verification program similar to TV [6] and CRYSTAL [5] has been developed at MSIL based on SIMMOS. The delay-evaluation method, where the charging path and discharging path delays are computed separately, and the built-in scheduling mechanism, have made it possible to develop such a tool in a relatively short time.

Acknowledgement

The multi-level environment that merges SIMMOS with MSIL-RTL was developed by Ronen Keinan and the author. The SIMMOS timing-verifier was written by Carina Ben-Zvi.

Ilia Greenblat defined an intermediate language, MSIL-DBB, which interfaces between SIMMOS to either work-station schematics or MSIL's layout extractor [8].

The author wishes to thank Israel Kashat, Arie Brish and Yehuda Shvager for their advice and support, and all MSIL chip-designers for their important feedback and suggestions.

References

[1] R. Bryant, "A Switch-Level Model and Simulator for MOS Digital Systems". IEEE Transactions on Computers, Feb. 1984.

[2] C. Terman, "RSIM - A Logic-level Timing Simulator".
IEEE International Conference on Computer Design, 1983.

[3] T. Schaefer, "A Transistor-Level Logic-with-Timing Simulator for MOS Circuits".
22nd Design Automation Conference, 1985.

[4] T. Lin and C. Mead, "Signal Delay in General RC Networks".
IEEE Transactions on CAD, Oct. 1984.

[5] J. Ousterhout, "Switch-Level Delay Models for Digital MOS VLSI".
21st Design Automation Conference, 1984.

[6] N. Jouppi, "Timing Analysis for NMOS VLSI".
20th Design Automation Conference, 1983.

[7] R. Bryant and M. Schuster, "Performance Evaluation of FMOSSIM, a Concurrent Switch-Level Fault Simulator".
22th Design Automation Conference, 1985.

[8] A. Brish, C. Ben-Zvi and O. Pardo, "RC - Parasitic Load Extractor From Layout".
To be published.

[9] S. Jain and V. Agrawal, "STAFAN: An Alternative To Fault Simulation".
21th Design Automation Conference, 1984.