MIT/LCS/TR-259

A SWITCH-LEVEL SIMULATION MODEL FOR INTEGRATED LOGIC CIRCUITS

Randal Everitt Bryant

A Switch-Level Simulation Model for Integrated Logic Circuits

by

Randal Everitt Bryant

March, 1981

This research was supported in part by the Department of Energy under contract number DE-AC02-79ER10473, and in part by the United States Air Force under contract number AFOSR F49620-80-C-0073.

Massachusetts Institute of Technology
Laboratory for Computer Science

Cambridge **Massachusetts 02139**

A Switch-Level Simulation Model for Integrated Logic Circuits

by

Randal Everitt Bryant

Submitted to the Department of Electrical Engineering and Computer Science on March 31, 1981 in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Abstract

Switch-level simulators model a metal oxide semiconductor (MOS) large scale integrated (LSI) circuit as a network of transistor "switches". They can simulate many aspects of MOS circuits which cannot be expressed in the Boolean logic gate model, such as bidirectional pass transistors, dynamic storage, and charge sharing. Furthermore, the logic network can be extracted directly from the mask specification by a relatively straightforward computer program. Unlike analog circuit simulators, however, the nodes are assigned discrete states 0, 1, and X (for unknown), and the transistors are assigned discrete states "open", "closed", and "unknown". As a consequence, switch-level simulators operate at speeds comparable to logic gate simulators.

In this thesis, a formal model of switch-level networks is developed. The networks in this model may contain transistors of different strengths and types, as well as nodes of different sizes and types, and hence the logical behavior of a wide variety of ratioed, complementary, and ratioless designs can be expressed. In keeping with the concept of a *logic* model, however, both the transistor strengths and the node sizes may take on only discrete values, and electrical behavior is modeled in a highly idealized way. The operation of a network is characterized by its *target state* function, which for a particular state of the network yields the logic states which the nodes would eventually reach if all transistors were held fixed in their initial states. This characterization abstracts away the rate at which nodes approach their target states and the voltages through which they pass but provides adequate detail for many simulation and analysis techniques. The target state function can be defined in terms of an abstraction called *logic signals*, where a logic signal gives a composite description of the network at some node much as a Thevenin equivalent network gives a composite description of a linear network at some port.

Logic signals can be formalized into a simple, discrete algebra with operations describing the effects of performing some elementary network transformations. A technique for finding the target state of an arbitrary switch-level network can be derived by utilizing concepts from abstract algebra and lattice theory. This technique leads to an algorithm for a switch-level simulator which improves on previous algorithms in its generality, speed, and simplicity. Furthermore, the mathematical formulation provides a means for proving useful properties about the simulation method and opens up further areas of application for the switch-level model.

Thesis Supervisor: **Jack B. Dennis**

Title: Professor of Electrical Engineering and Computer Science

Keywords: Switch-level simulation, logic simulation, computer-aided design, large scale integration.

Acknowledgments

I would like to thank Professor Jack B. Dennis for his continued intellectual and financial support during my graduate studies. He has taught me the value of defining a problem in terms of a set of simple, but often untraditional concepts upon which the solution can be based. Other present and past members of the Computation Structures Group have greatly contributed to this environment including Professor Arvind, Clement Leung, Dean Brock, Bill Ackerman, Andy Boughton, Narinder Singh, Sheldon Borkin, and Ken Weng.

My introduction to integrated circuit design was provided by Ms. Lynn Conway of Xerox PARC, and she has provided valuable encouragement of my research efforts. As a Teaching Assistant for Professor Jonathon Allen, I began serious work in this area, and his course provided a test facility for my simulator. The students in this course showed remarkable patience and enthusiasm. Both Professor Allen and Professor Paul Penfield have provided continued interest and encouragement in my work and served as readers for this thesis.

Much of the development of switch-level simulation has occurred through the efforts of Chris Terman and Clark Baker. My work has benefitted greatly from their ideas and experiences. The VLSI design communities at MIT and at other institutions have also provided a strong, motivating force for these developments.

The following people also assisted me by reading earlier drafts of this thesis: Dean Brock, Chris Terman, Clark Baker, and Wayne Gramlich.

This research was supported in part by the Department of Energy under contract number DE-AC02-79ER10473, and in part by United States Air Force Contract AFOSR F49620-80-C-0073.

Contents

1. Introduction

Recently, a new class of logic simulator has emerged specifically for simulating metal oxide semiconductor (MOS) large scale integrated (LSI) circuits. These *switch-level* simulators model an MOS design as a set of nodes connected by transistor "switches" with each node assuming a state 0, 1, or X (unknown) and each switch a state "open", "closed", or "unknown". Programs such as the author's MOSSIM [8, 9] and others [5] show remarkable accuracy and versatility in simulating such logic elements as logic gates, pass transistor logic, busses, and both static and dynamic memory. The accuracy results because the logic network closely matches the actual circuit, while the versatility results because transistors form a common denominator for all LSI design techniques. Furthermore these simulators operate at sufficient speeds to test entire LSI systems, because behavior is modeled at a logical rather than a detailed electrical level. Unlike previous attempts at developing MOS logic simulators by adding *ad hoc* extensions to gate-level simulators, switch-level simulators are based on a uniform and consistent model which provides a powerful level of abstraction for viewing MOS designs. In this thesis the concept of switch-level simulation is developed into a mathematical model of MOS logic networks from which simulation algorithms and other analytic tools can be derived.

The ability to implement digital logic has progressed greatly in the past decade with circuits of increasing size and complexity being fabricated at decreasing cost. Metal oxide semiconductor (MOS) technology has played a major role in this "integrated circuit revolution" due to its relative simplicity in both design and fabrication. In more recent years MOS design has become part of the university curriculum in both electrical engineering and computer science. By using simplified design rules and conservative clocking schemes, and by following systematic methodologies such as those presented by Mead and Conway [28], the basics of MOS design can be learned in one semester. As this training becomes widespread, we will see a new form of integrated circuit revolution in which nonspecialists design their own custom integrated circuits rather than relying on the limited variety of commercially available products.

With the increasing number of custom-designed integrated circuits, and with the growing size and complexity of commercial LSI products, the inadequacy of current LSI design techniques has become apparent. The semiconductor industry has traditionally relied on humans to design, lay out, and verify LSI systems. Typically many man hours are spent, and several prototype chip designs are fabricated in developing a single IC design. Industry analysts have extrapolated the current design techniques and estimated that a 100,000 device microprocessor (the expected state of the art in 1982) would take 60 man-years to lay out and another 60 to debug [41]. Rather than accepting such predictions as inevitable, a change in design techniques is called for.

Computerized tools have been applied to commercial LSI design, but most of these can be viewed as extensions of manual techniques (such as graphical layout systems), or as experts in a specialized domain (such as circuit simulators.) Both kinds require close cooperation with a human who understands the exact capabilities and limitations of the program. In addition, humans are required to bridge the gaps between programs with expertise in different domains. For example, before a design can be tested by a logic simulator, the actual design typically must be translated by hand into a description which can be understood by the simulator. This translation process wastes manpower in performing a rather tedious task and also decreases the level of confidence provided by the simulation.

Logic simulators form an important class of computerized tools for LSI design. Their utility has long been recognized for analyzing designs which by their size and complexity exceed the capability of humans to fully understand. The usefulness of a logic simulator depends greatly on the consistency and accuracy with which it can model the full range of design techniques available to the designer. Of course no logic simulator can model all designs with complete accuracy, because it does not simulate the detailed analog behavior. Nonetheless, it should provide as close a model as possible within a set of well-defined limitations. As a further requirement, a logic simulator for LSI must be efficient enough to simulate entire systems with reasonable speed. The size of single-chip, very large scale integrated (VLSI) systems will far exceed the small scale integrated (SSI) systems for which conventional logic simulators were designed.

A logic simulator has as its basis an abstract model of how digital systems function. This *logical model* describes both the structure and the behavior of a system in terms of a set of primitive elements, a set of interconnections, and a set of rules for operation. For a simulator to accurately and reliably simulate a system, the logical model must reflect its actual structure and operation.

Unfortunately, the development of logic simulators has not kept pace with LSI technology. This inadequacy stems in part from the lack of formal logic models for describing the behavior of MOS circuits.[1] Instead, systems are designed and simulated using an *ad hoc* combination of Boolean gate models, relay models, and electronic models. Such a representation may be appropriate for human designers, who can combine different modes of operation and resolve the conflicts between these models. Computers, however, lack the intuition required to simultaneously view a system at several different levels. Computerized tools must be based on models which can describe a large class of systems in a more uniform way.

Most logic simulators are based on the Boolean gate model which adequately models systems built from SSI components but fails to support the wide variety of techniques available to the LSI designer, especially for MOS LSI. Many extensions of the Boolean gate model have been attempted with the usual result that only a slightly larger class of designs can be simulated and many sources of unpredictable or inaccurate behavior are introduced. This claim will be supported by briefly surveying the development of logic simulators with respect to their support for MOS LSI design.

1. Brzozowski and Yoeli [10] present a logic model in which "T-elements" provide a simplified model of field-effect transistors. This model, however, only expresses the operation of static logic gates. Furthermore, it has not received widespread attention.

1.1 The Boolean Gate Model

The Boolean logic gate model has formed the theoretical basis for logic design ever since the advent of electronic logic. In this model a system consists of a set of logic gates connected by unidirectional, memoryless wires. The logic gates compute Boolean functions of their input signals and transmit these values along the wires to the inputs of other gates. Each gate input has a unique signal source. Information is stored only in the feedback paths of sequential circuits. This model directly implements Boolean algebra [20] and hence has a well-defined specification which can guide the simulator implementation.

The Boolean gate model cannot describe many of the techniques available to the logic designer, especially the MOS LSI designer. MOS pass transistor networks can implement combinational logic in ways which more closely resemble relay contact networks than logic gate networks (see [28] or [16] for several examples.) Dynamic memory can store information without feedback paths by exploiting the capacitances of the wires and the gates of the transistors attached to them. A variety of bus structures can provide multidirectional, multipoint communication. A logic simulator which implements only the Boolean gate model provides limited support to the MOS LSI designer. Most existing logic simulators, however, extend the Boolean gate model in various ways. Hence, any evaluation of logic gate simulators must consider these extensions as well.

Many simulators extend the two-valued logic of Boolean algebra with a third value to represent an unknown or undefined logic level. This "X" level can indicate an uninitialized state variable, a signal held between the two logic thresholds, or a signal in transition between 0 and 1 or between 1 and 0. The X logic level can be handled algebraically by changing the two-valued Boolean algebra to a three-valued DeMorgan's algebra [3, 11, 23, 46].[1] Thus, even with this extension many of the desirable mathematical

1. A DeMorgan's Algebra satisfies all postulates of Boolean Algebra except for the Law of Excluded Middle ($A + \neg A = 1$).

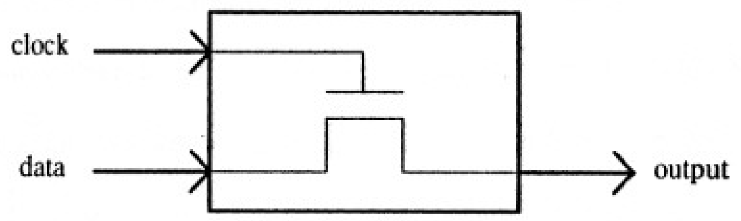
properties of the Boolean gate model are preserved. Alternatively, some simulators implement the X level by an enumeration technique in which the simulation is repeated with the nodes at the X level set to all possible combinations of 0's and 1's [6, 45]. Nodes which remain at a unique level for all combinations are set to this level, while all others are set X. Still other simulators [44] use *ad hoc* techniques to implement the X level often resulting in inconsistent or anomalous behavior. The X logic level is useful in simulating all forms of digital logic including MOS.

To model the behavior of bus structures, some logic simulators have a fourth or "high-impedance" logic level [12]. This H level corresponds to the third state of tri-state logic. To simulate a bus structure, the outputs of a number of gates are connected to a common node. Typically all but one output will be at the H level, and the level of that output will dominate. Unlike the X level which can be viewed as an extension of Boolean algebra, the H level violates a basic principle of the Boolean model, in that a logic gate input no longer has a unique signal source. Because the simulator is not based on a well-defined mathematical model, it becomes difficult to implement consistently and accurately. The H state may be adequate for simulating SSI designs in which only limited forms of tri-state busses can be implemented. The MOS LSI designer, on the other hand, can select from a wide variety of bus designs, such as pre-charge/discharge and multiple driver designs. The H state only partially captures the behavior of these bus structures. Nonetheless, the H logic level is seen in simulators for both MOS and other forms of logic.

Some simulators allow a special logic gate to represent the MOS pass transistor [44]. This logic gate models a field-effect transistor (FET) as a unidirectional device with two inputs and one output as shown in Figure 1.1. The simulator cannot help in cases in which the bidirectional property of the FET is important, such as in circuits where information may flow in either direction,[1] or where the design has a

1. In actual fact, the bidirectional property of the FET is rarely used intentionally. The author has seen only a few designs which have signals propagating in both directions through a pass transistor.

Fig. 1.1. The FET Logic Gate



clock	data	output
0	0	H
0	1	H
1	0	0
1	1	1

malfunction due to a sneak path. More recent gate model simulators have implemented bidirectional transistor models, but these transistors usually entail a much higher computational cost, and hence their use must be minimized. Furthermore, few of these simulators can simulate arbitrary combinational networks of pass transistors, because they cannot model the action of pullup resistors. Thus this extension does not fully capture the behavior of the MOSFET. Like the high-impedance logic level, it also lacks the algebraic properties of the Boolean gate model and hence forces an *ad hoc* implementation.

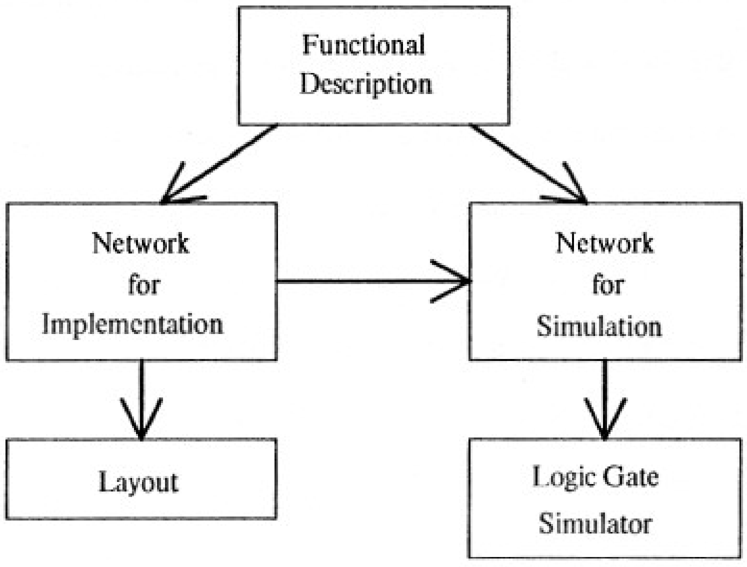
Finally, some simulators model dynamic memory in a limited fashion [44]. A node is allowed to remain at a previous logic level if the outputs of all logic gates connected to the node are at the H level. This extension is very limited in its generality and its accuracy.[1]

As new types of MOS logic circuits are developed, designers add more extensions of the Boolean gate model to their simulators. These extensions are doomed to failure, because they cannot correct the fundamental mismatch between the Boolean gate model and MOS logic circuits. MOS circuits consist of bidirectional switching elements connected by bidirectional wires with memory (considering the capacitive effects of the transistor gates as contributing to a wire's memory.) Instead, the simulators become increasingly cumbersome and unreliable, because they only partially capture the behavior of the logic technology.

1. SIMULOG introduces even greater inaccuracy by failing to differentiate between the undefined (X) level and the high-impedance (H) level.

Using a gate model simulator requires both intuition about how the design is supposed to function and detailed knowledge of the simulator implementation. The user must explicitly identify the logic gates, the signal directions through pass transistors, the locations of busses, the sites of dynamic memory, and sometimes even the feedback paths. Often the actual logic design must be transformed into one compatible with the simulator which may not display the exact same behavior. This transformation process not only decreases the level of confidence provided by the simulation, it virtually eliminates the possibility of automatically generating the simulation network from some specification of the actual design. Unless we restrict our attention to a limited class of designs, a very sophisticated program would be required to analyze the mask patterns for an MOS layout and convert this to a gate-level description, performing the necessary transformations to provide compatibility with the simulator. Without this capability, a logic simulator cannot be used to help verify the correctness of a layout. Gate-level logic simulators fit into the MOS LSI design process as shown in Figure 1.2. They provide mainly a verification of the high level functional description plus limited verification of the actual logic design.

Fig. 1.2. Role of Logic Gate Simulators in LSI Design



1.2 Analog and Hybrid Simulators

LSI designers have recognized the limitations of conventional logic simulators for modeling MOS circuits and have at times resorted to analog or hybrid simulators. Analog simulators treat the entire design as network of analog circuit elements and try to model the detailed waveform at every node over time. Simulators such as SPICE [30] and even those which use faster (and more approximate) numerical techniques such as MOTIS [13] require very large amounts of computation. Some reports claim the amount of computation scales as the square of the network size [2]. Thus they are practical only for small designs or for small sections of larger designs. Analog simulators, however, are based on a uniform and general abstract model and hence have been well received because of their consistency and accuracy. Furthermore, computer programs exist for deriving the simulation network automatically from the layout descriptions [2].

The amount of computation can be reduced significantly by hybrid techniques such as in SPLICE [31] in which some sections of the design are simulated as logic gates and others are simulated as analog circuits. Hybrid simulation works well as long as only small, isolated sections of the design need be simulated as analog circuits. Unfortunately, a human must decide which portions of the network can be modeled as logic gates, and which portions require analog simulation. Furthermore, trying to combine analog and logic models in a single program requires rather unsatisfactory approximations at the interfaces. For example, if an output of a section modeled as logic gates is to be interfaced to an input of a section modeled as an analog circuit, the program must convert the logic signal into a voltage waveform. This, of course, cannot be done with any accuracy, because much of the necessary information is lacking. The resultant outputs of the analog section must then be viewed with skepticism. Similarly, if the logic simulator were extended to include the X state, it could not be interfaced to an analog simulator because this state does not represent a single voltage. Unless great care is exercised, a hybrid simulation could well provide the accuracy of a logic simulator at the speed of an analog simulator, rather than *vice-versa.*

For this reason, a human must monitor the simulation very carefully.

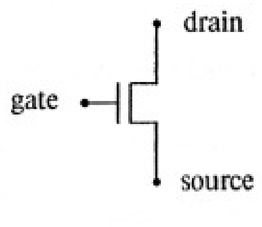
1.3 Switch-Level Simulators

As an alternative to conventional logic simulators, the author has developed the simulator MOSSIM [8, 9] specifically for the logical simulation of MOS LSI. With MOSSIM the Boolean gate concept is discarded altogether and replaced with a logical model which closely matches the structure and behavior of MOS circuits. A logic network consists of a set of nodes connected by a set of FET "switches". MOSSIM uses three logic levels: 0, 1, and X (undefined.) There are three types of nodes:

1. *Input* nodes provide a strong, externally generated signal (*e.g.* power lines, clock drivers, data inputs, etc.)
2. *Pullup* nodes are connected via a pullup resistor to a high voltage. They will generate a 1 signal unless grounded. The output of an nMOS logic gate is an example of a pullup node.
3. *Normal* nodes cannot generate a signal but can store a signal dynamically.

Only two types of network elements are allowed: p-type and n-type field-effect transistors. A transistor is a three node device which acts as a voltage-controlled switch with no assumed direction of signal flow as shown in Figure 1.3. No distinction is made between the labels "source" and "drain". When the gate node of a transistor is in the X state, the switch status is unknown: it may be open, closed, or somewhere between. The user interface of MOSSIM allows the user to describe the network in terms

Fig. 1.3. The MOSSIM Transistor Model



p-type

gate	effect
0	closed
1	open
X	unknown

n-type

gate	effect
0	open
1	closed
X	unknown

of transistors, logic gates, and user-defined macros, but these are all translated into a transistor-level representation for the simulation. C. Terman has developed a switch-level simulator patterned after MOSSIM [5] but differing in several respects, as is discussed at several points in this thesis. Researchers at Caltech [34] also developed a switch-level MOS logic simulator, but not to a degree of accuracy or generality required in a serious design tool. Researchers at other laboratories have developed their own switch-level simulators based on these earlier designs.

Switch-level simulators can simulate almost the full range of circuit designs available to the MOS designer without any special logic levels or poorly-defined logic elements. Both logic gate and pass transistor combinational logic are simulated without difficulty, as are both static and dynamic memory. A wide variety of bus structures can be simulated including tri-state busses, pre-charged busses, etc. Most significantly, the user need not tell the simulator what type of logic structure is intended, only the actual physical structure of the design.

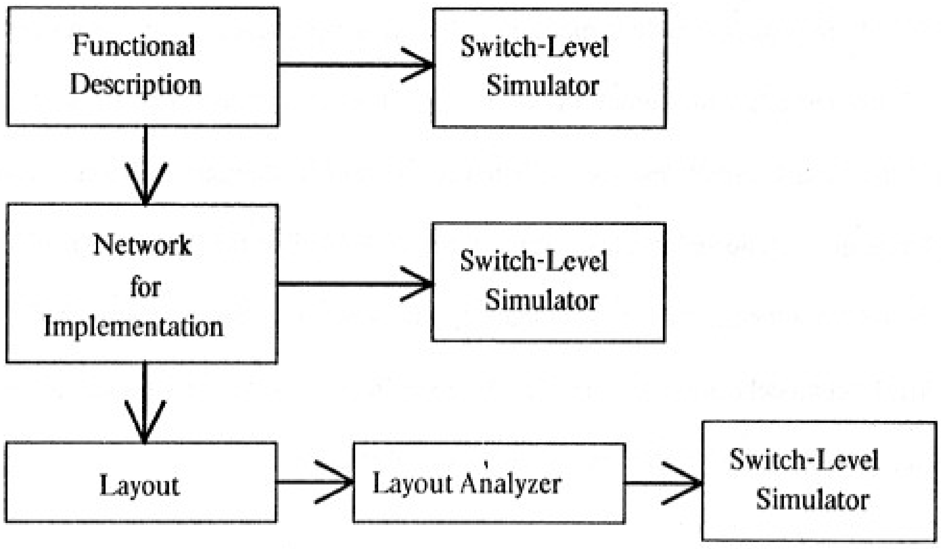
Switch-level simulators have been tested on a wide variety of MOS designs ranging from student homework problems to a LISP microprocessor chip containing over 10,000 transistors [21]. They have proved remarkably general and accurate, correctly simulating logic design techniques which were not even anticipated in the simulator design. The confidence in the simulation results is greatly enhanced by the fact that the user can see an exact correspondence between the actual design and the simulation network. Moreover, the simulation is fast enough that entire designs can be simulated. For the LISP microprocessor chip, MOSSIM requires between 5 and 12 seconds of CPU time on a DEC20/60 to simulate each clock cycle. The designers were able to fully test the system by simulating 700 clock cycles. Experience has shown that simulation inevitably uncovers fatal errors in the design.

C. Baker [4] has written a program which can take layouts specified in the Caltech Intermediate Form [28] and generate the equivalent transistor-level network. Unlike a program which generates a logic gate description, this program needs no special intuition about logic design. It need only look for electrical connectivity and transistors in the mask patterns. The simulation network for the LISP microprocessor chip can be generated with 30 minutes of CPU time on a PDP 11/70. This technique has proved extremely valuable, uncovering errors both in the layout and the logic design. Furthermore, it saves the duplicative effort of entering the design by hand for the two different representations. Switch-level simulators can fit into the MOS LSI design cycle at several levels, being applied independently to the high level description, the actual design, and the layout, as shown in Figure 1.4.

1.4 An Abstract Model for MOS LSI

The generality and accuracy of switch-level simulators suggest that a formal model of MOS based on the switch-level concept can be developed. The uniformity and consistency of the switch-level approach are precisely those properties which make a concept amenable to a mathematical treatment. This model would serve not only as a basis for verifying the correctness of a logic simulator, but also as the foundation for new computer tools for MOS design. One need only look at the many advances made possible by Shannon's development of logic models based on Boolean algebra [38, 39] to understand the value of abstract logic models. While the expected benefits of a MOS logic model are much more

Fig. 1.4. Role of Switch-Level Simulation in LSI Design



modest, its utility could be significant. Unlike traditional switching theory which was developed to help humans analyze and synthesize networks containing small numbers of elements, we now want models which can form the basis of computer programs to be applied to networks containing thousands of elements. This places more emphasis on the generality and uniformity of the model and the algorithmic complexity with which it can be implemented.

In this thesis, a formal switch-level model of MOS logic networks is developed. The network model closely resembles the network model of MOSSIM but generalizes it in several respects. As with MOSSIM, a logic network consists of a set of nodes interconnected by a set of transistors. Unlike MOSSIM, however, only two types of nodes: input and normal are allowed. To model ratioed circuits, transistors may have different strengths, with a stronger transistor (such as an inverter pulldown) being able to override a weaker one (such as a pullup load transistor). A third type of transistor, d-type (for "depletion") is introduced which is closed regardless of the gate signal. This new model more closely matches the actual structure of MOS networks, because in MOS networks the relative sizes of transistors determine the logical behavior. The pullup node used in MOSSIM is a rather *ad hoc* way of representing this. The new model can also describe a wider variety of networks, including circuits which rely on multiple levels of ratioing.

In MOSSIM, each normal node is modeled as having a capacitance of unknown value which can store a signal dynamically but cannot drive this signal onto another node in a different state. Unfortunately this model cannot describe the behavior of many bus designs in which a relatively high capacitance bus node is connected to a lower capacitance node (such as the storage node of a 3-transistor dynamic RAM cell) resulting in both nodes obtaining the same logic state as was originally on the bus. Our new switch-level model can model this effect by assigning each normal node a *size*, where the signal on a larger node will predominate when connected to a smaller node.

Our abstract model describes both the time and electrical behavior of a network in a highly idealized way. The time behavior is described by the *target state* function giving the logic states which the normal nodes would reach for a particular set of input node, transistor, and initial normal node states. For designs containing no critical races, the logical behavior can be modeled by repeated application of the target state function. To model the electrical behavior, the target state is defined in terms of the set of steady state voltages in an "order of magnitude" electrical network. This class of networks models the conducting transistors by linear resistors, where the conductances of the resistors for different strength transistors differ by orders of magnitude. As a consequence, any path to an input node containing only transistors with strength greater than or equal to some value is modeled as overriding any path containing a transistor with strength less than this value. Similarly, the normal nodes are modeled by capacitors where the capacitances of the capacitors for different size nodes differ by orders of magnitude. As a result, the target states formed on a set of nodes through charge sharing depends only on the state(s) of the largest node(s) in the set. Furthermore, no attempt is made to accurately compute the node voltages. Instead, they are classified into the three logic states 0, 1, and X. This model provides a simplified view of ratioed circuits and charge sharing which adequately describes the logical behavior of most MOS circuits.

Although the target state is defined in terms of an electrical model, we will find that the target state of an arbitrary switch-level network can be computed without evaluating any electrical networks. Instead, by introducing an abstraction called *logic signals*, an iterative method for computing the target state can be developed which uses only operations in a simple, discrete algebra. A logic signal provides a composite description of a switch-level network at some node for a particular set of node and transistor states, much as a Thevenin network [15] provides a composite description of a linear network at some port for a particular set of network parameters. However, whereas finding the Thevenin equivalent generally requires solving a set of simultaneous linear equations, finding the logic signal requires much less effort. A simple set of rules describes the logic signals created by the input and normal nodes in their initial states and the effects on a node of other nodes connected through conducting transistors. With these rules we can develop an equation expressing a set of constraints which must be satisfied by the signal for each normal node in terms of the initial node signal and the signals for input nodes and for other normal nodes connected through conducting transistors. It can then be shown that the minimum solution of this set of equations equals the set of logic signals describing the network at each node, and the set of steady logic states can easily be derived from this solution. Logic signals can be formalized into simple, discrete algebra with a domain corresponding to the signal values and operations describing the effects of the rules for logic signals. This algebra allows us to apply elementary concepts from abstract algebra and lattice theory to develop techniques for computing the target state. These techniques can be further developed into efficient simulation algorithms.

1.5 Relation to Relay Networks

The switch-level MOS model can be viewed as an extension of Shannon's relay network model [38, 39]. The algebra of signal strengths introduced in Chapter 5 bears many similarities to Boolean algebra. As Shannon observed, a relay can be viewed as a switch with conductance 1 when closed and 0 when open.[1] The rules for connecting relays in series and in parallel and the methods of analyzing relay networks are special cases of those for transistor networks.

MOS networks, however, have several characteristics which are not found in relay networks. First, relay networks are used as a *current-driven* logic, in which the logic state of a node is determined by the connection between the node and the current source. Thus a simple characterization of the connection to the signal source determines the state of a node. MOS networks, in contrast, are used as a *voltage-driven* logic, where the state of a node is determined by both its connection to the supply voltage and its connection to ground. One must characterize both the states of the signal sources and the connections to them to determine the state of a node. Furthermore, in a voltage-driven logic erroneous behavior may

1. Shannon actually described the state of a relay by its *hindrance*, the complement of its conductance.

result due to a short circuit between signal sources, and hence we also require the state X for logical completeness. Second, relay networks do not allow ratioing in which one closed switch can override another. Thus, a Boolean characterization of a conductance path suffices. Third, relay networks can only store information in feedback paths. No modeling of dynamic memory or charge sharing is needed. Finally, most theoretical work on relay networks was conducted before the widespread availability of digital computers. Thus, most techniques were developed to aid the hand design of small circuits. The standards by which ideas are measured change greatly when they are to be incorporated into computerized tools to aid the design of very large systems.

1.6 Outline of Thesis

In the next chapter, the details of how the switch-level model describes the structure and operation of MOS networks is presented. By modeling the network structure at a transistor level and by allowing transistors of different strengths and nodes of different sizes, this model covers a large variety of MOS design techniques in a way which closely matches the actual circuit designs. The switch-level model can be viewed as either a simplification of analog network models or an extension of relay network models. The time behavior of a network is described by the target state function, giving the logic states toward which the nodes are driven or charged given the current node and transistor states. The value of this function is defined in terms of the steady state voltages in a linear electrical network that models the transistor network. The logical behavior of many MOS networks is described by repeated applications of their target state functions. In computing the target state, the transistors are modeled with time-invariant elements, thereby simplifying the analysis considerably.

In Chapters 3 and 4 the electrical circuit-oriented view of the target state function provided by the definition given in Chapter 2 is transformed into a more abstract and logical view. It is shown that the target state can be defined in terms of the *steady states* of a set of *logical conductance* networks, where a logical conductance network represents a switch-level network in which each transistor is either nonconducting or fully conducting. The concept of logic signals is then developed to express the behavior of logical conductance networks. With the logic signal abstraction we can derive an equation which gives the steady state of a logical conductance network, and consequently the target state of a switch-level network, which does not require evaluating any electrical networks.

In Chapter 5, an algebra of logic signals is developed with operations describing the effects of a set of network transformations. This algebra allows us to apply elementary concepts from abstract algebra and lattice theory to the study of switch-level networks.

In Chapter 6 the mathematical formalism presented in Chapter 5 is used to derive a technique for computing the target state of a switch-level network. This development utilizes only the logic signal abstraction as expressed by the algebra of Chapter 5 and two equations which are derived in Chapters 3 and 4 from the analysis of the electrical model. Although we could arrive at the desired results more directly by utilizing additional properties of the electrical model, this approach demonstrates the power of our abstract approach.

In Chapter 7, the abstract solution technique of Chapter 6 is developed into an efficient algorithm for a switch-level simulator. By exploiting the sparseness of the network, the simulator requires at most linear time to simulate one clock cycle for almost all networks. This algorithm improves on previous switch-level simulation algorithms in several respects. Some performance data for MOSSIM is presented to demonstrate the performance characteristics of switch-level simulation and how it compares to logic gate simulation.

In Chapter 8, the simplified timing model of MOSSIM is investigated more closely to see for what classes of systems it is valid. Possible methods of implementing logic simulators with other timing models are presented. In addition, a *ternary* simulation algorithm is developed which uses the X state to detect potential races in MOS networks. This algorithm is a straightforward extension of Brzozowski and Yoeli's algorithm for logic gate networks [11]. Ternary simulation requires a much more accurate and efficient implementation of the X state than is required for functional simulation, because the X state will become the most prevalent state in the network. The algorithm presented in Chapter 7 provides this accuracy and efficiency.

Finally, in Chapter 9 some ideas for further improvements of logic simulators and for future applications of the switch-level model are described.

1.7 Notation

In the remainder of this presentation, the following notational conventions are observed. Scalar values are denoted with lower case letters (e.g. a, b); vectors with boldface, lower case letters (e.g. **a**, **b**); and matrices with boldface, upper case letters (e.g. **A**, **B**). Mathematical domains, i.e. sets of values with particular mathematical properties are denoted with script, upper case letters (e.g. $\mathcal{A}$, $\mathfrak{B}$), while ordinary sets are written with italic, upper case letters (e.g. *A*, *B*). The extension of a domain $\mathfrak{D}$ to vectors of size n is denoted $\mathfrak{D}^{n}$, and the extension to matrices with n rows and m columns is denoted $\mathfrak{D}^{n \times m}$.