PHYSICALLY REALIZABLE GATE MODELS

by

Paul R. Stephan and Robert K. Brayton

Memorandum No. UCB/ERL M93/33

12 May 1993

PHYSICALLY REALIZABLE GATE MODELS

by

Paul R. Stephan and Robert K. Brayton

Memorandum No. UCB/ERL M93/33

12 May 1993

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

PHYSICALLY REALIZABLE GATE MODELS

by

Paul R. Stephan and Robert K. Brayton

Memorandum No. UCB/ERL M93/33

12 May 1993

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Physically Realizable Gate Models

Paul R. Stephan Robert K. Brayton

Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
May 12, 1993

Abstract

We propose an objective criterion for determining if, given a specific circuit technology, a gate model is suitable for synthesis and verification. This is based on relating the analog circuit behavior to the digital model behavior using a formal definition of implementation. We show how the criterion is not satisfied for several gate models currently used for synthesizing asynchronous circuits, and illustrate the design errors which occur when these models are used. Finally we introduce a new gate model which is designed to satisfy the criterion.

1 Introduction

An important parameter in designing asynchronous logic circuits is the choice of a logic gate model. The model determines which synthesis algorithms may be used, as well as verification and testing strategies. Many basic gate models have been developed for asynchronous circuits, differing in the assumptions made and the amount of detail used. For example, speed-independent gate models assumes circuit delays can be lumped at gate outputs and such delays are unbounded, feedback delay models lump all delay into a minimal number of wires such that each cycle contains at least one delay, while inertial delay models vary how a gate responds to pulses of different widths[7].

With the variety of available gate models, it can be difficult to decide when each is appropriate. Currently gate models are compared subjectively as more or less accurate, realistic, conservative, etc. Given a specific circuit technology, there is no method for determining which models are suitable for synthesis and verification. In other words, how accurate is accurate enough?

We propose an objective criterion for gate models based on relating the digital gate behavior to the underlying analog circuit behavior. We start with a relational model of behavior which is general enough to apply to both analog and digital behavior. We develop a formal definition of "implementation" as a restricted type of homomorphism between two behaviors, called a *behavior epimorphism*. Applying this to the technology design step, a gate model is *physically realizable* with respect to a technology if there is a behavior epimorphism from the analog components to the corresponding digital gate model.

When a model satisfies the criterion, synthesis and verification results using the model are correct regardless of how detailed the model is. More detailed models may allow more efficient designs, but there is no tradeoff of correctness. When the criterion is not satisfied, the actual circuit may exhibit behavior which was not allowed by the digital specification, as illustrated by several examples.

Finally, we show how the criterion can be used to construct new gate models by analyzing the detailed analog behavior of a CMOS inverter, and deriving a suitable digital model. This model is used in a small example to contrast it with the physically unrealizable models. Although the criterion was originally developed to methodically evaluate gate models for asynchronous design, it applies as well to synchronous design.

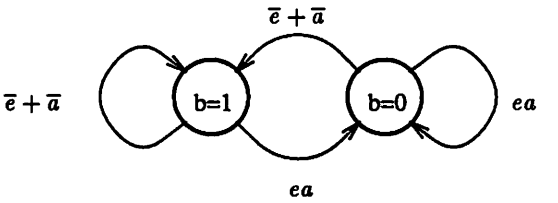

Figure 1: Example specification, with initial state $b = 1$.

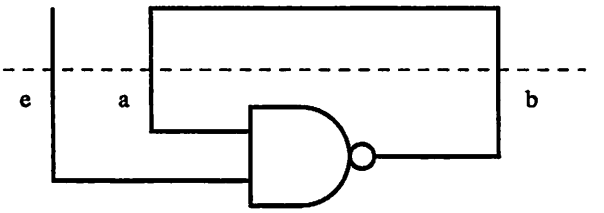

Figure 2: One-gate ring oscillator with enable.

1.1 An Example

The following small example is used throughout the paper to evaluate the different gate models. Consider the Moore machine specification in Fig. 1 with one output b and two inputs, e (enable) and a. The machine can be reset to its initial state by applying $e = 0$ for a sufficiently long time. When e is then changed to 1, the specification states that a is alternately 1 and 0, with output b inverting a. This specification can be translated to a flow table, event graph, signal transition graph, change diagram, etc. as necessary to apply the different synthesis algorithms found in the literature (see [9] for detailed examples).

With most asynchronous design methodologies, the gate netlist synthesized from this specification will be a single NAND gate, $b = \overline{ae}$. It is allowed to compose this with the function $a = b$, i.e. a wire. Thus an implementation of Fig. 1 with part of its environment is given by the gate netlist in Fig. 2. Because of the simple function of this system, most synthesis algorithms would determine that no delay constraints are needed on the NAND gate to insure correct digital operation. One expected digital behavior of this system is

$$\begin{aligned} e &= 011111111\ldots \\ b &= 110101010\ldots \end{aligned}$$

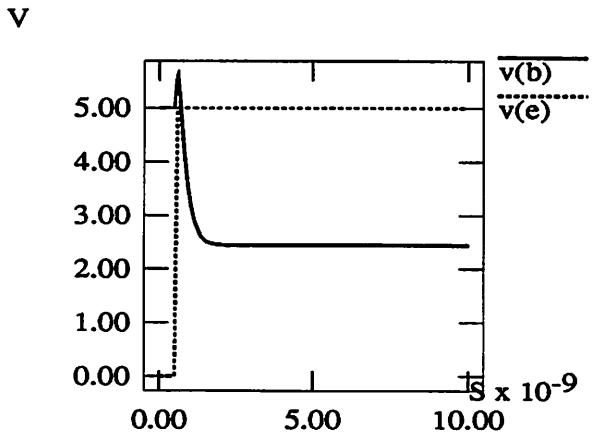

Figure 3: SPICE simulation of a CMOS one-gate ring oscillator.

Using SPICE[4] to simulate a CMOS implementation of this circuit results in the waveform shown in Fig. 3. After $v(e)$ is stepped to 5 volts, the output b shows a brief transient and then settles at 2.5 volts, which is not usually associated with an oscillating sequence of 1/0 values.

Intuitively, this discrepancy arises when the gate models are not accurate enough. For example, we know intuitively that by increasing the delay from b to a, the behavior should be as expected. But the details on how much delay as a function of the load are missing and is not part of current methodologies for formal synthesis and verification of asynchronous circuits. In fact, current theories of logic design lack objective criteria to determine if a gate model is accurate enough for synthesis and verification with respect to a particular circuit technology. This applies to synchronous designs as well.

The specification of Fig. 1 is a simple example of a nonconstant behavior which illustrates implementation and composition as used in technology mapping. Arbitration is a similar example where problems occur at the gate level. Given a logical specification of an arbiter, many asynchronous synthesis algorithms generate a gate-level implementation similar to Fig. 4. As with Fig. 2 there is a discrepancy between the analog behavior of the corresponding circuits and the original digital specification. The discrepancy can be explained by applying the criterion proposed in Section 2.

1.2 Prior Work

Generally the relationship between analog circuit behavior and digital logic models has been expressed informally in terms of idealized characteristics. For example, in [3] digital logic circuits are analyzed in terms of several first order effects such as static noise margins, propagation delay, and rise and fall times. These parameters are estimated using simplified circuit models, and idealized waveforms such as ramp and step inputs. Propagation delay is defined with respect to the 50% points of each waveform, and rise/fall times are between 10% and 90% points, even though these are not the logic thresholds.

Two recent works have explored more precise relationships between analog and digital behavior. Brockett[2] synthesizes a system of differential equations which implements a finite automaton under a straightforward quantization, but does not discuss how these analog systems can be realized or verified physically. The differential equations are an idealization which can only be approximated by a real circuit. Kurshan et al.[5] relate a SPICE model to a digital gate model, but use idealizations both in relating the SPICE model to a circuit, and in relating the digital model to higher level logic models. Therefore their formal verification technique does not address any analog connection to physical devices. In comparison, we propose a single criterion to be satisfied between all levels of a design flow including the level from the manufactured circuit to the technology mapping netlist of gate models.

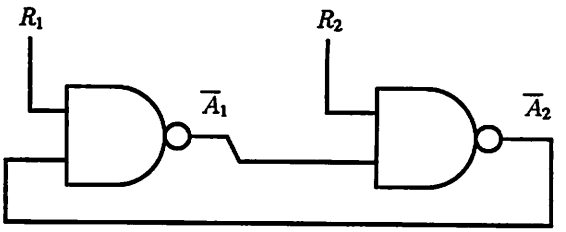

Figure 4: Naive implementation of an arbiter.

1.3 Notation

The formulation in the next section uses several general, binary relations which in practice are often functions. To simplify the notation for the general case, the following notation for functions is extended to apply to general relations. Let R be a binary relation on two sets $R \subseteq X \times Y$. An associated reference direction is defined from X to Y, and evaluation of R for some element $x \in X$ is defined as

$$R(x) = \{y : (x, y) \in R\}$$

This reduces to function evaluation when R is a function, but note that in general $R(x)$ may be empty for some x. The converse of R, denoted R^{-1}, is defined as

$$R^{-1} = \{(y, x) : (x, y) \in R\}$$

The evaluation of the relation for a set of elements $X' \subseteq X$ is defined in a straightforward manner, and is also denoted by $R(X')$ since the two uses can be distinguished from context. The domain of R is defined as $\mathrm{Dom}(R) = R^{-1}(Y)$; the range of R as $\mathrm{Ran}(R) = R(X)$. A relation R is *onto* if $\mathrm{Ran}(R) = Y$. The projection of R onto a subspace S is denoted by $\pi(R, S)$.

2 Deriving the Criterion

To define our notion of *physically realizable* for a gate model, we first define a mathematical formalization of the I/O behavior of a system, define implementation between two behaviors, and then apply this to the technology mapping design step.

2.1 Defining System

The characteristics of a system are derived from the relationship between a set set of observable quantities such as voltages, currents, and temperatures. This is formalized as a set of terminals, one for each observable quantity.

Definition 1 *A terminal is a triple $q = (n, \Sigma_q, d)$, where n is the terminal name, Σ_q is the range of the terminal values, and $d \in \{in, out, none\}$ is an assigned terminal direction.*

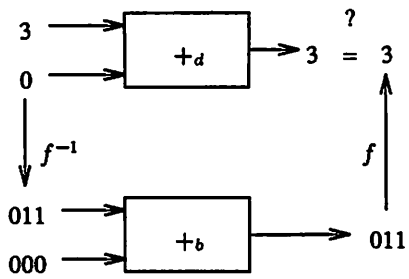

Figure 5: Intuition behind implementation.

A direction of *none* is allowed because the quantities of a physical system have no inherent direction. This is distinct from the concept of a bidirectional or tristate pin of a circuit. Let Q denote the set of terminals for a system. A behavior is defined by observing the values on the terminals Q over some interval of time. We arbitrarily consider semiclosed time intervals to make concatenation more convenient. Given a specific interval of interest, $T = [t_1, t_2)$, a *signal space* is the set of all possible observations for a set of terminals over interval T. Note that to describe analog behavior, Σ_q may be infinite.

Definition 2 *Given terminal $q_i = (n, \Sigma_{q_i}, d)$ and time interval T, a* signal *is a mapping $s : T \rightarrow \Sigma_{q_i}$. A* signal space *$\mathcal{S}_{q_i}$ is the set of all possible signals for q_i:*

$$\mathcal{S}_{q_i} = \{s : s \text{ is a signal over } \Sigma_{q_i}\}$$

A signal space over multiple terminals is the cross product of the individual signal spaces. In the sequel, we assume the observation interval T has been defined and will not refer to it explicitly.

Definition 3 *Given a set of terminals Q, a* behavior *is defined as*

$$B \subseteq \mathcal{S}_Q$$

This is sometimes called a relational model of behavior since (3) is equivalent to

$$B \subseteq \mathcal{S}_{q_1} \times \mathcal{S}_{q_2} \times \ldots \times \mathcal{S}_{q_n}$$

where the terminals may have different assigned directions. Similar formalizations of system behavior have been previously proposed [6]. Although a physical system has, by definition, a single behavior, a specification for such a system often consists of multiple behaviors. This is necessary to express general don't care conditions where several different output signals are possible for each input signal, and the different output signals are dependent. Thus the formalization of a system includes a set of allowed behaviors.

Definition 4 *A* system *is a pair $M = (Q, \mathcal{B})$, where $Q = \{q_1, q_2, \ldots, q_n)$ is a set of n terminals, and $\mathcal{B}$ is a set of possible behaviors. If every terminal of M has been assigned a direction of in or out, M is called a* directed *system.*

2.2 Defining Implementation

An element of the implements relation is defined by two systems, a specification and an implementation. Normally the implementation has more detail, and is closer in some sense to a physical realization of the original intended behavior.

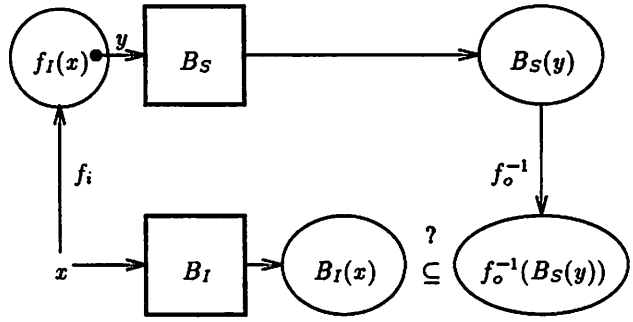

Figure 6: Relation f is a behavior homomorphism if this is satisfied for all x in DomB_I.

Consider the intuition behind integer addition (over a limited domain) and a gate netlist which implements addition using three bit 1's-complement numbers. In addition to the two systems, a mapping f is given from implementation values to specification values, e.g. 000 $\mapsto$ 0, 011 $\mapsto$ 3, 111 $\mapsto$ 0, etc. Then, as illustrated in Fig. 5, for any two specification input values (decimal numbers), the result of simulating the gate netlist for the corresponding binary input values is the same (after converting to decimal) as the decimal addition of the two values.

The condition in Fig. 5 can be expressed in a form similar to the homomorphisms defined for other algebraic systems[9]. As a homomorphism, f is said to preserve addition, since the operators $+_d$ and $+_b$ have similar properties within their respective domains. To apply homomorphisms to system implementation, the equality test must be generalized to allow both the "operator" and the mapping f to be general relations. Here we simply present a generalized definition; the detailed arguments supporting this definition are given in [9].

For a system $M = (Q, \mathcal{B})$, let $i(M)$ and $o(M)$ denote the set of input and output terminals respectively for M. Since for a directed system, $i(M) \cup o(M) = Q$, a behavior $B \in \mathcal{B}$ can be considered a binary relation with associated reference direction $B \subseteq \mathcal{S}_i \times \mathcal{S}_o$. Referring to Fig. 6, we propose the following definition.

Definition 5 *Given $M_S = (Q_S, \mathcal{B}_S)$ and $M_I = (Q_I, \mathcal{B}_I)$ be directed systems with signal spaces $\mathcal{S}_{Q_S}$ and $\mathcal{S}_{Q_I}$ respectively. Consider two specific behaviors, $B_I \in \mathcal{B}_I$ and $B_S \in \mathcal{B}_S$. Let f be a signal space relation $f \subseteq \mathcal{S}_{Q_I} \times \mathcal{S}_{Q_S}$, and define the implied input relation*

$$f_i = \pi(f, \mathcal{S}_{i(M_I)} \times \mathcal{S}_{i(M_S)})$$

and the implied output relation

$$f_o = \pi(f, \mathcal{S}_{o(M_I)} \times \mathcal{S}_{o(M_S)})$$

Then relation f is a behavior homomorphism *if:*

$$\forall x \in \text{Dom}B_I \; \forall y \in f_i(x) \; B_I(x) \subseteq f_o^{-1}(B_S(y))$$

If this holds for all B_S, B_I, we say M_S is an abstraction *of M_I.*

When relations B_S, B_I and f are functions, this reduces to the more familiar condition of algebraic homomorphism

$$\forall x \in \text{Dom}B_I \;\; f_o(B_I(x)) = B_S(f_i(x))$$

Early work[6] focused especially on the subset of homomorphisms which are isomorphisms, i.e. f_i and f_o are 1-to-1 and

onto. This condition is too strong for implementation. On the other hand, a general homomorphism as defined above is too weak. Specifically, for implementation we require that f_i be onto, because for any value in the specification, there must be some way to apply it in the implementation. For example, in Fig. 5 if no binary value is mapped to 3, there would be no way to compute 3 + 0, which contradicts the intuitive meaning.

Definition 6 *Let f be a behavior homomorphism from B_I to B_S. If f_i is onto, f is called a behavior* epimorphism. *In this case we call M_S the* specification, *M_I an* implementation, *and we say M_I implements M_S.*

This can be considered a weak epimorphism since f_o need not be onto. The additional condition of requiring f_o to be onto is too strong for implementation, and does not seem to correspond to any standard concept in logic design. There is no clear precedent in abstract algebra since other algebraic epimorphisms are either defined for functions, or for which inputs and outputs are defined over a single set and the mappings f_i and f_o reduce to a single mapping. Thus we use *epimorphism* to denote the weak epimorphism in Definition 6. Note that f_o is still constrained to satisfy the condition for behavior homomorphism.

The intuition for the formal definition of implementation was derived by examining several cases of implementation, by generalizing the conditions as much as possible, and by checking the consistency of this definition with a large set of examples. Using a similar approach, we formally define the concepts of synthesis and verification. The detailed arguments supporting these definitions are found in [9].

Definition 7 *Given specification M_S, a* synthesis *algorithm either generates an implementation M_I and a behavior epimorphism f from M_I to M_S, or terminates in failure.*

The option of failure must be allowed in case the specification is so constrained that no implementation is possible or at least that the algorithm is unable to find one, in the case heuristics are used. For many design steps, e.g. logic minimization, the epimorphism is trivial and not given explicitly by the algorithm. In other cases, such as input/output encoding, the algorithm must indicate what the relation f is. Verification is closely related to synthesis, and can also be defined in terms of behavior homomorphism.

Definition 8 *Given systems M_S, M_I, and relation $f \subseteq S_{Q_I} \times S_{Q_S}$, a hardware* verification *algorithm is one which solves the decision problem "is f a behavior homomorphism from M_I to M_S?"*

While synthesis implies generating an *implementation*, verification is often used only to confirm an abstraction. Verification algorithms are categorized based on this distinction, e.g. implementation verification (confirming a behavior epimorphism) versus design or property verification (confirming a behavior homomorphism).

Note that the formal relationship between a specification and an implementation is a requirement of *design*, regardless of whether or not verification (or synthesis) is used. Verification is the process of *confirming* this relationship.

2.3 Technology Mapping

Using the preceding definitions for implementation and synthesis, we can now give a formal characterization of a technology mapping design step. Given some specification system M_1, a technology mapping design step generates a gate netlist M_2 which implements M_1. This means there is an associated mapping f_a which is a behavior epimorphism from M_2 to M_1, as illustrated in Fig. 7. Since f_a relates two digital spaces, it is usually implicit in the technology mapping algorithm.

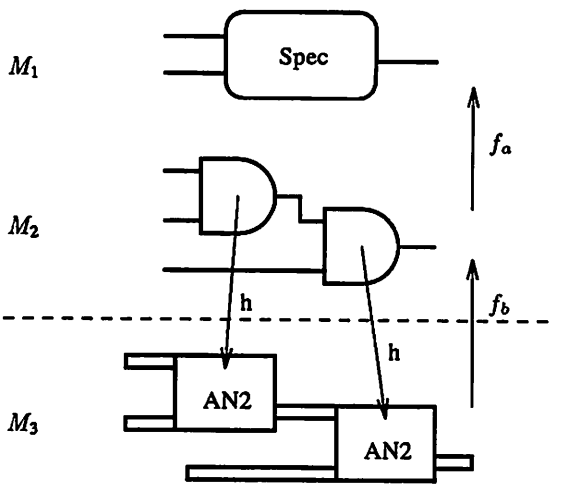

Figure 7: Technology mapping generates a netlist M_2 which implements M_1 *and* induces a physical implementation M_3.

However, unlike most other design steps, technology mapping implies another implementation based on the cell library. For each "gate" in the netlist, there is an associated cell in the library. Denote this mapping by h. Then the interconnection of gates in M_2 defines an interconnection M_3 of cells from the library via h, and M_3 must also implement M_1. Thus by definition there is a second relation f_b such that $f_a \circ f_b$ is a behavior epimorphism from M_3 to M_1.

The dotted line represents the boundary between the mathematical models used for digital design, and the analog physical devices manufactured from these models. We assume that the conceptual epimorphism f_b must be as rigorous as that for any other design step. This leads to the following criterion for the gate models used in a technology mapping step.

Definition 9 *Let M_2 be a digital gate netlist, and M_3 a corresponding manufactured circuit, with relation $f_b \subseteq S_{M_3} \times S_{M_2}$. If f_b is a behavior epimorphism, then M_2 is* physically realizable.

In other words, a physically realizable netlist of gate models is actually implemented by the corresponding library cells. This condition is more difficult to satisfy than it may seem because these mappings are from analog to digital behavior, and must hold under all worst case operating conditions of the physical circuit. Because the ultimate aim of synthesis and verification is to build and verify that it works in the real world physical environment, it is not sufficient to map only a set of idealized voltage waveforms with instantaneous transitions or smooth, monotonic characteristics. The epimorphism must also hold under temperature variations, crosstalk, electromagnetic interference, and all the other nonideal conditions that real circuits must tolerate.

In practice, most technology mapping steps are based on a set of gate models which both implement some logic computation, and in turn are implemented by one cell from a technology library. Complicated functions are synthesized using composition, where the composition of the library cells implements the corresponding composition of logic functions (this is not required in general).

In this case, it is meaningful to consider whether the individual gate models are physically realizable. Since this special case is so common, we informally refer to the criterion as "physically realizable gate models" with the understanding that the general case is given by Definition 9. The composition of two behaviors is defined as follows.

Definition 10 *Given two systems $M_a = (Q_a, \mathcal{B}_a)$, $M_b = (Q_b, \mathcal{B}_b)$, and a terminal q such that $q \in Q_a$ and $q \in Q_b$, the composition of two behaviors $B_a \in \mathcal{B}_a$ and $B_b \in \mathcal{B}_b$ with respect to q is the largest subset $B_c \subseteq \mathcal{S}_{Q_a \cup Q_b}$ which satisfies*

$$\begin{aligned} \pi(B_c, \mathcal{S}_{Q_a}) &\subseteq B_a \\ \pi(B_c, \mathcal{S}_{Q_b}) &\subseteq B_b \end{aligned}$$

In other words, the composition consists of all the signals where the terminal q has the same values in both B_a and B_b. It is straightforward to generalize this definition to allow composition of two terminals with different names or ranges. Note that two systems do not have to be directed in order to compose them, and that there is no restriction on the direction of q. Practical composition operators may add such restrictions in order to define *useful* compositions.

The criterion of physical realizability (PR) defines a new approach to the gate models and composition operators used in technology mapping. Currently models are motivated from the point of view of analysis and simulation and are compared subjectively as more or less detailed, with no good answer to "how much detail is enough?" The new approach formalizes model requirements for synthesis and verification. Given a specific circuit technology, any PR model is good enough for this; different PR models offer tradeoffs in tractability versus flexibility.

2.4 Asynchronous Logic

The terms "synchronous" and "asynchronous" have several different definitions. To define these terms in the context of logic circuit design, consider synchronous logic circuits and the behavior epimorphisms where the clock signal(s) is first introduced. Typically this is in the last stage of logic design: the technology mapping step. At the analog level each signal alternates between intervals where it is logically valid, e.g. is within the thresholds for a logic 0 or 1, and intervals where it is unstable or transitioning. In a synchronous circuit, these intervals are synchronized to transitions of the clock. The clock signal is primarily an analog signal with timing constraints which support the epimorphism from analog behavior to logical behavior. An epimorphism from analog to digital behavior which does not synchronize intervals with a global reference is then called "asynchronous".

Thus for hardware design, the term "asynchronous" categorizes the epimorphism used for technology mapping, and the criterion of physical realizability can be used to evaluate the gate models used for synthesizing asynchronous logic circuits. Most existing asynchronous design methodologies are not based on fully defined technology mapping steps. Timing constraints and dynamic behavior are considered only at the digital level, for example in terms of a state graph. This leads to the type of problem illustrated in Section 1.1.

3 Binary Gate Models

Many gate models are defined in terms of two Boolean values, logic 0 and 1 (see [7] for several examples). This includes binary models where time is explicit, such as in [1], and models which define a third value $X = \{0, 1\}$ with corresponding extensions to the logical operators (sometimes called ternary models), such as in [8]. In this section, we consider several possible mappings from the analog behavior of a CMOS inverter to a corresponding binary digital model, and show why they are physically unrealizable.

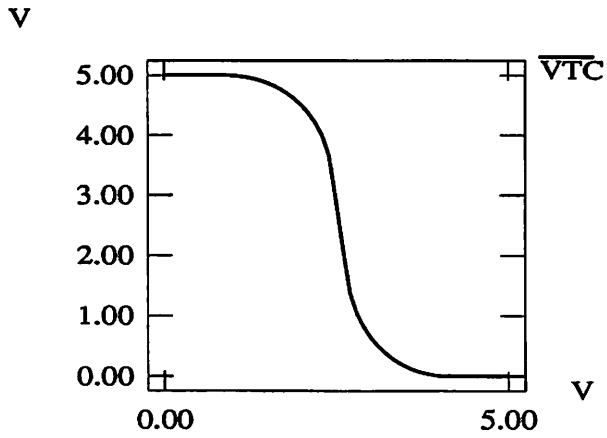

Figure 8: Typical voltage transfer characteristic (VTC) for a static CMOS inverter.

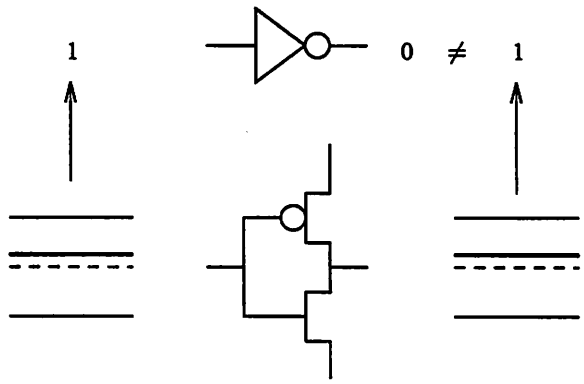

Figure 9: One-threshold quantization is not an epimorphism when the VTC shifts to the right.

A ratioed, static CMOS inverter has nearly ideal characteristics for a logic gate[3]. A typical static voltage transfer characteristic (VTC) is shown in Fig. 8. The behavior of the corresponding binary digital gate model is defined over $\{0, 1\}^2 \times [t_1, t_2)$ for some arbitrary observation interval $[t_1, t_2)$. The value of an output waveform is the complement of the input waveform, possibly with some time shifting. The details may vary somewhat (e.g. uniform versus nonuniform time shifts) without affecting the following analysis since we only consider static values or single, isolated transitions.

3.1 One-Threshold Quantization

A straightforward mapping uses a single threshold at $V_{th} = 2.5$ to map $v > V_{th}$ to logic 1 and $v \leq V_{th}$ to logic 0. Assume that because of process variations, the VTC does not pass exactly through the point $(2.5, 2.5)$, but instead passes to the right of it. As illustrated in Fig. 9, if an input voltage slightly above V_{th}

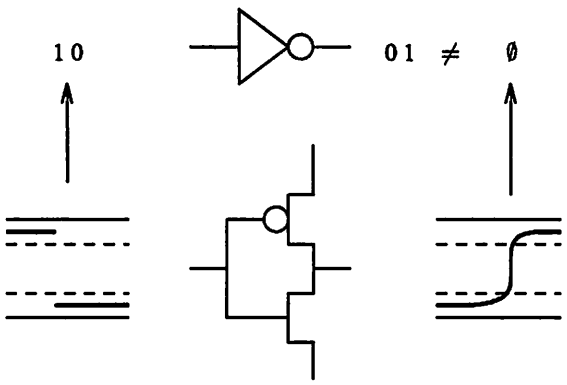

Figure 10: Two-threshold quantization with instantaneous jumps.

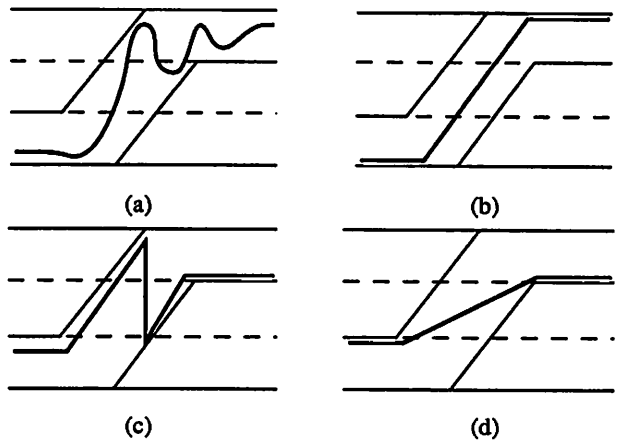

Figure 11: Four analog waveforms which fit the same digital parameters.

is applied to the inverter *circuit*, the output will also be slightly above V_{th}. But then both waveforms map to logic value 1, which does not satisfy the condition for behavior epimorphism.

To accommodate process variations, assume the switching threshold V_{th} is defined independently for each inverter circuit. Thus the previous example would have a higher input threshold, say $V_1 = 2.6$, while some other inverter might have an input threshold of $V_2 = 2.4$. Individually, each mapping is an epimorphism. However, it is no longer practical to have a fanout greater than one since in general the different fanouts will have different input thresholds, and no binary quantization will define an epimorphism for the composition. Another argument against this mapping is a variation of the first, considering the dynamic changes in switching thresholds caused by temperature variations.

3.2 Two-Threshold Quantization

Instead of a single threshold, we can consider mappings based on two thresholds, V_L and V_H, for each signal. Since the digital model has only the two values, it is necessary to construct a mapping which somehow "hides" the intermediate voltages between V_L and V_H.

First consider simply restricting the analog waveforms to only those which instantaneously jump across the gap, as illustrated in Fig. 10. This mapping fails because even with a discontinuous input, the analog output of the inverter circuit will not in general be similarly discontinuous. Since this mapping is not defined for such an output, the epimorphism condition is not satisfied.

An alternative mapping might use some sort of average. Assume the analog waveform crosses V_H at a time t_1, and later crosses V_L at time t_2. We could define a mapping such that from $[t_1, (t_1 + t_2)/2)$ is mapped to logic 1, and from $((t_1 + t_2)/2, t_2)$ is logic 0. Such mappings introduce other problems. For example, the mapping must also be defined in the case that the waveform crosses V_H into the undefined region and then back across V_H without crossing V_L. Also, under such a mapping, the delay of a gate can become negative which is often not allowed by the composition operators used in technology mapping.

3.3 Other Variations

Many other mappings can be devised for a binary gate model, including models with forms of inertial delay. Those we have examined are all physically unrealizable[9] with respect to standard technologies such as CMOS, and we conjecture that this is true for any binary valued gate model since they seem incapable of abstracting the dynamic behavior of logic circuits. This shortcoming leads to the problem in Section 1.1.

The intuition that these gate models are insufficiently realistic is expressed objectively by showing that they are physically unrealizable. Note that it is important to consider the mappings for compositions of gates as well as for the individual models, since most technology mapping steps are based on composition.

4 A New Gate Model

The criterion would be of dubious value if all gate models proved to be physically unrealizable. In this section, we use the criterion to develop a gate model which is physically realizable. The gate model can be composed, but we have not yet developed a general composition operator to form the basis of a synthesis algorithm. Following Definition 6, we first define a signal mapping, then derive the parameters of a model using this mapping to satisfy the epimorphism condition. For simplicity, we only consider the details for an inverter.

4.1 Signal Mapping

The mapping from analog waveforms to digital values is illustrated in Fig. 12. Two thresholds are defined for each signal, e.g. V_{IL} and V_{IH} for an input signal. The transition region for a signal is defined in terms of a parallelogram which is defined by two parameters. Parameter U represents the width, or uncertainty, of the parallelogram, and T represents the transition time as defined in Fig. 12. An analog signal making a low to high transition can be described by the sequence of digital values `L U T H`. Thus the digital model uses a four-valued algebra.

4.2 Behavior

The behavior of the digital model is defined with signals over $\Sigma_S = \{L, I, U, T\}$. The input/output relationship of an inverter can be described using an additional parameter P, the propagation delay, bounded by limits P_{min} and P_{max}. Since the behavior of the inverter depends strongly on the load L it is driving, the inverter gate model has eleven parameters: V_{IL}, V_{IH}, V_{OL}, V_{OH}, U_I, T_I, U_O, T_O, L, P_{min} and P_{max}. The digital behavior must be defined in terms of these parameters to accommodate worst case variations in temperature, supply voltage, noise, etc.

To simplify this example we fix the load to a single inverter of the

same type (allowing the model to be applied to Fig. 1), but similar results can be derived for other loads. Fixing the thresholds to reasonable values based on Fig. 8, leaves six parameters. The condition in Definition 6 is applied by assigning values to T_I and U_I to define a digital input signal, and performing the following computation.

- Define a fixed T_O with respect to these parameters as the slope of the output from simulating a ramp input with rise time T_I.
- For each analog waveform w which fits these parameters, simulate w (with SPICE) for worst case variations in supply voltage, etc. and parameterize the output waveforms by the smallest $U_{O,w}$ with respect to the fixed T_O.
- Define P_w with respect to the input and output parallelograms as illustrated in Fig. 12.
- Let U_O be the maximum of all the $U_{0,w}$, P_{min} the minimum of all the P_w, and P_{max} the maximum of all the P_w.

By repeating this for different T_I and U_I, the parameters of the digital model are computed relative to the circuit technology being simulated. Note that the objective of this procedure is not to develop an epimorphism from the SPICE models to the digital gate model, but rather to use a suitable set of SPICE models to help define an epimorphism from a circuit technology to the gate model.

As the different possibilities cannot be exhaustively simulated, it is necessary to use engineering judgement to select a practical set of input waveforms and parameter variations (combined with suitable margins of error) to define the boundary conditions for the model. For input waveforms, instead of arbitrary analog signals such as Fig. 11(a), seven piecewise linear waveforms representing different extremes were simulated, including the three in Fig. 11(b)-(d).

The results of this experiment are plotted in four graphs to show the interaction of the six parameters. One of the most important relationships is the dependence of output uncertainty U_O on input uncertainty U_I. Since asynchronous circuits generally have feedback, if the uncertainty always increases, the gate model will be of limited value.

In Fig. 13, U_O is plotted versus U_I and T_I. The vertical segment in each curve occurs when U_I becomes large enough to allow significant glitches in the input, as shown in Fig. 14. The first transition is not wide enough to force the output across the threshold, while the second one is. For $T_I = 0$, the output uncertainty is always larger than the input, since for small values of U_I, the rectangle is not a good fit for the nonlinear output transition, and for larger U_I, glitches become a problem. As the transition time T_I is increased, the parallelograms fit the response waveforms better, and for $U_I < 0.7$ns the output uncertainty is significantly decreased. For comparison, note that on this graph most asynchronous gate models would be a dot at the origin.

In Fig. 15, the output transition time is plotted relative to the input transition. Recall that to simplify the model, we fixed T_O with respect to U_I. For relatively steep input waveforms, $T_I < 0.47$ns, the output parameter T_O actually increases. Note that on this graph also most asynchronous gate models would be a dot at the origin.

Finally, Fig. 16 and Fig. 17 show how P_{min} and P_{max} respectively depend on U_I and T_I.

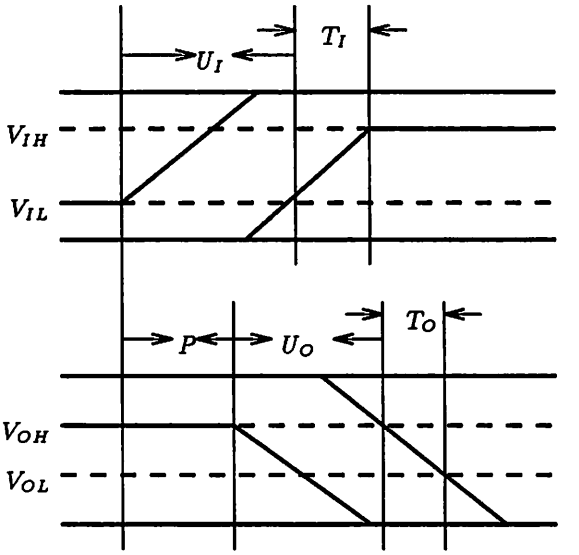

Figure 12: Mapping from analog waveforms to digital signals for an input and output signal.

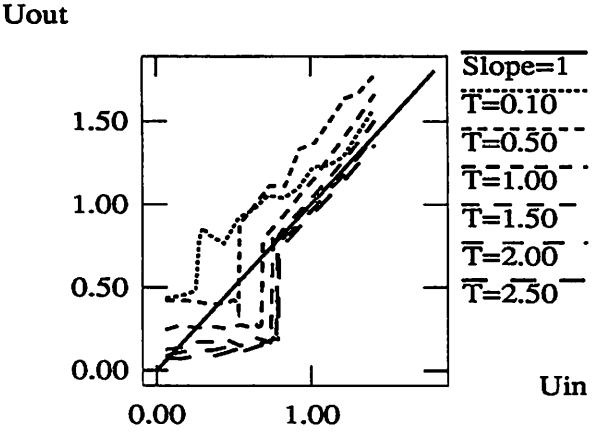

Figure 13: Plot of U_O versus U_I and T_I (all values in ns).

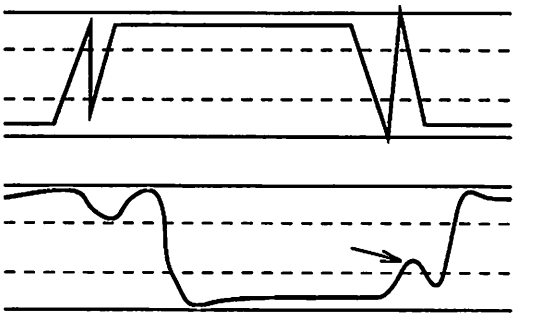

Figure 14: The discontinuity in U_O occurs when U_I is large enough for the indicated glitch to cross the threshold.

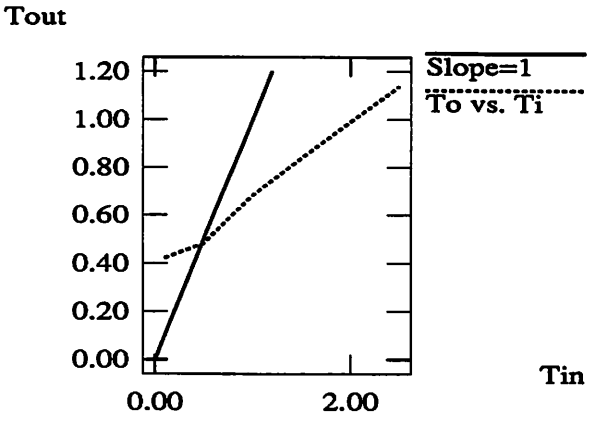

Figure 15: Plot of T_O versus T_I.

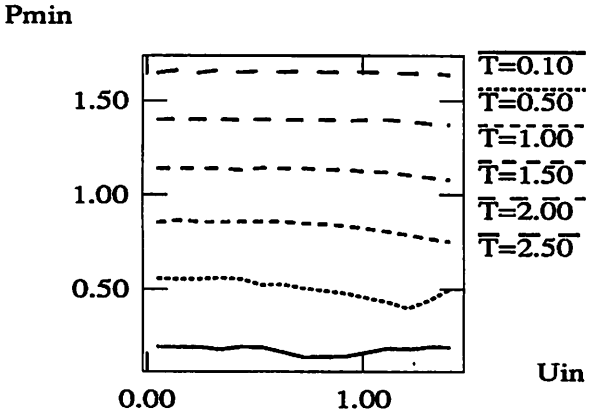

Figure 16: Plot of $P_{\min}$ versus U_I and T_I.

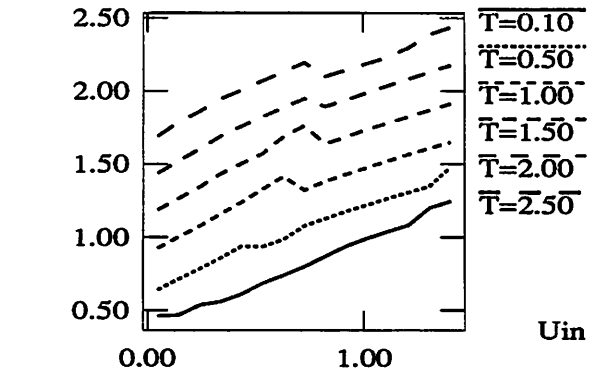

Figure 17: Plot of $P_{\max}$ versus U_I and T_I.

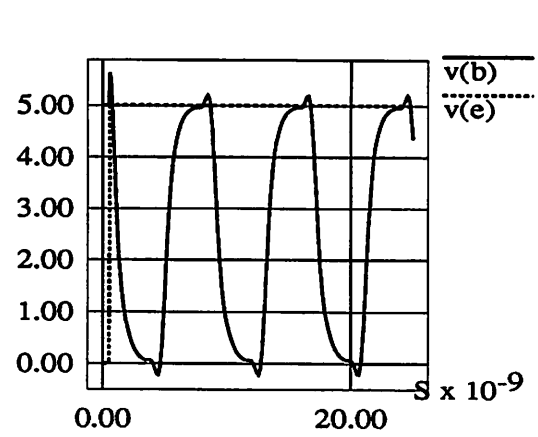

Figure 18: SPICE simulation of a CMOS seven-gate ring oscillator.

4.3 Using the Model

For a tractable gate model, it is necessary to abstract the data in Figs. 13-17 as much as possible while remaining consistent with Definition 6. For example, in Fig. 13, it is possible to increase the value of U_O without violating Definition 6. In terms of Fig. 6, this corresponds to increasing $B_S(y)$, which is allowed because we are defining the gate model, and which preserves the subset containment since the superset is being increased.

Thus we could choose to represent part of the curve for $T_I = 1.0$ and $U_I < 0.6$ by the segment $U_O = 0.30$, and similarly for the other curves. Likewise, in Fig. 16 the curve for P_{min} at $T_I = 0.10$ could be modified to be $P_{min} = 0.18$, and the model would still be physically realizable.

To define the digital model, several additional simplifying assumptions are made. First, let $P_{min} = 0.18$ independent of T_I and U_I. Next simplify the relation of T_O versus T_I to

$$T_O = 0.38 + 0.3T_I$$

Finally, simplify the parameterization of U_O by two linear segments as illustrated for $T_I = 0.5$ in Fig. 20 (the other curve is used later). Note that each point on this curve is no less than the corresponding point in Fig. 13. Using these simplifications, $T_O = T_I$ at the value 0.54, so if the environment has the same characteristics as this inverter, this will be the limiting value of the digital slope parameter.

To apply this model to the example in Section 1.1 requires a model for a NAND gate. The model for a two-input NAND gate has ten parameters instead of six, which is awkward to present here, so we will illustrate the basic results using the inverter model. Assume the enable input is raised sharply, i.e. by a transition with $T \approx 0$ and $U \approx 0$. When the enable input is raised, the circuit behaves essentially like an inverter with input connected to output, with the input having made a sharp transition to logic 1.

Because of this transition the inverter is unstable and the output begins a transition to 0. After a propagation delay of at least 0.18ns, the output will change from H to U. From Fig. 13, the duration of the U will be at most 0.45ns, and then the output should transition to T. However this assumes the input is being

held stable at L, which is not true if the environment, i.e. the feedback wire, has short propagation delay. Thus this model cannot predict the repeating sequence of L and H values (with intervening Us and Ts) necessary to satisfy the specification.

To satisfy the assumption of a stable input during the output transition, the propagation delay of the environment must exceed the transition time of the inverter. Evaluating the response of the inverter to a step transition as on the enable input, and assuming the environment has the same dynamic characteristics, the behavior will reach a fixed point at $T_O \approx 0.54$ns, and $U_O \approx 0.50$ns, for an overall transition time of 1.04ns. If the environment is a chain of inverters, at least six (the least even number greater than 1.04/0.18) are needed to guarantee the necessary propagation delay.

Thus this model predicts that for a ring oscillator of seven inverters, the cumulative propagation delay is sufficient to guarantee that the signal voltages will exceed the thresholds for digital oscillation. As illustrated in Fig. 18, simulation of one possible analog behavior of this circuit corresponds with the digital specification. As long as the physical devices can be guaranteed to satisfy the gate model parameters, we can formally guarantee (as a mathematical proof) that the physical circuit must have the digital behavior specified by Fig. 1.

A more detailed model (particularly for P_{min}) could perhaps be used to synthesize a five inverter ring, but any circuits which can be synthesized using the simpler model are correct. On the other hand, any suspicion that the gate model is incorrect can be confirmed objectively by demonstrating an analog input and output waveform which do not satisfy Definition 6.

4.4 A Simplified Model

A simpler gate model is generated by fixing T_I and T_O to zero, making all the transition regions into rectangles. The signal mapping from analog to digital is illustrated in Fig. 19. Essentially the digital behavior has been modified so that the uncertainty U is the sum of the previous U and T parameters. Although the model is still physically realizable, the output uncertainty is always larger than the input uncertainty, as shown by the curve "T = any" in Fig. 20. In an asynchronous circuit, this uncertainty will propagate until the entire state of the circuit is unpredictable.

However, for a synchronous circuit, the clock signal breaks this feedback of increasing U. By making the clock period long enough, the U of the transitions on the latch outputs are related to the uncertainty of the clock signal, not to the U of the latch inputs. Since the clock signal uncertainty does not degrade, this stabilizes the behavior of the synchronous circuit, and a synchronous implementation of Fig. 1 can be derived even with the simplified gate model[9].

A synchronous implementation is shown in Fig. 21. The synthesis algorithm still needs to insure that the minimum delay from a clock edge along the path $(2,4,1,2)$ exceeds the delay along path $(2,4,5,4)$, i.e. the data must be latched faster than the feedback path. This will require increasing the propagation delay through gate 1, similar to the asynchronous example.

4.5 Verification Models

Gate models which are physically realizable appear to require more detail than most of the gate models currently used for asynchronous circuits. Since more detailed models generally require more complicated synthesis algorithms, it is worth considering whether physically unrealizable models can be used to synthesize a candidate implementation, and then use formal verification to check the result.

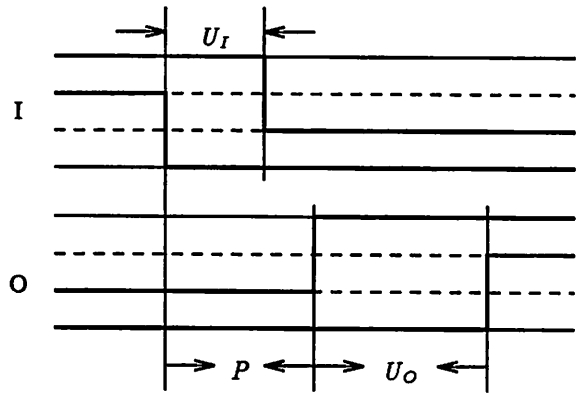

Figure 19: Simplified model with $T_I = T_O = 0$.

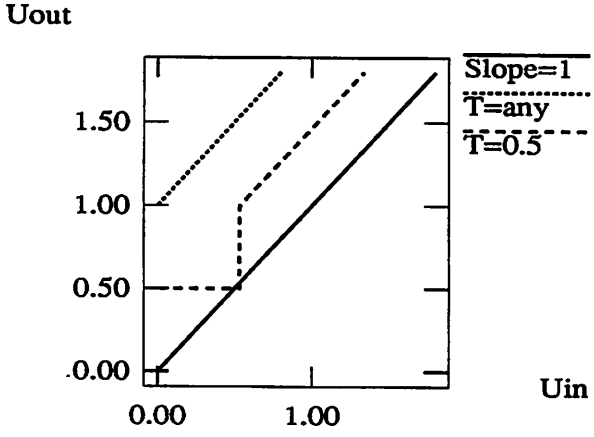

Figure 20: Plot of U_O versus U_I for a simplified model.

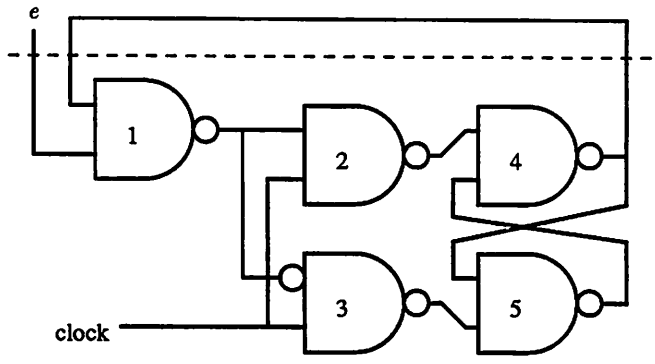

Figure 21: Level-sensitive synchronous implementation of Fig. 1.

Let M_S be a specification, M_U be a physically unrealizable gate netlist synthesized from M_S, and M_C be a corresponding circuit. Because M_U is not physically realizable, M_C is not guaranteed to implement M_S; consequently this is what must be verified. Since M_C is a physical circuit, it cannot be verified directly. Instead, another model of the circuit, M_R, must be used, and in order for the verification result to apply to M_C, M_R must be physically realizable.

Thus formal verification may allow simpler models to be used for synthesis in a trial-and-error approach, but cannot avoid the need for physically realizable models.

5 Conclusions

We have proposed an objective criterion for determining if a gate model can be used for synthesis and verification of *circuits*, i.e. physical, manufactured components. The criterion is derived using a formal definition of implementation as a *behavior epimorphism*. By generalizing this to apply to both analog and digital behavior, it can be applied to the technology mapping design step. A gate netlist is *physically realizable* if the corresponding physical circuits implement the netlist.

As a technology mapping step, a synthesis algorithm for asynchronous logic must define a gate model, composition operators, an analog-to-digital signal mapping f, and the technologies for which f is an epimorphism. When f is not, the circuit behavior does not satisfy the specification under all operating conditions. Experienced engineers can manually correct these problems, but for synthesis and verification CAD tools it is crucial that the models be physically realizable.

The criterion also can be used to develop new digital gate models for a technology, as illustrated for a CMOS inverter. This is important both for modeling new technologies such as optical logic devices, and for determining the model parameters of standard logic families as feature sizes decrease.

In continuing research, we are working to define a suitable composition operator for the new gate model and a parameterization of two-input NAND gates to form the basis of a synthesis algorithm for asynchronous circuits. We are also developing a new semantics for logic design models in terms of the relational model of behavior.

Acknowledgements

This work was supported by the Semiconductor Research Corporation under contract 93-DC-008.

References

[1] L. M. Augustin. An algebra of waveforms. In L. J. M. Claesen, editor, *Formal VLSI Specification and Synthesis: VLSI Design Methods-I*, pages 309–320. North Holland, NY, 1990.

[2] R. W. Brockett. Smooth dynamical systems which realize arithmetical and logical operations. In H. Nijmeijer and J. M. Schumacher, editors, *Three Decades of Mathematical System Theory: A Collection of Surveys at the Occasion of the 50th Birthday of Jan C. Willems*, volume 135 of *Lecture Notes in Control and Information Sciences*, pages 19–30. Springer-Verlag, Berlin, 1989.

[3] D. A. Hodges and H. G. Jackson. *Analysis and Design of Digital Integrated Circuits*. McGraw Hill Book Company, 1988. Second Edition.

[4] B. Johnson, T. Quarles, A. R. Newton, D. O. Pederson, and A. Sangiovanni-Vincentelli. SPICE3 version 3F user's guide. Technical report, U.C. Berkeley, Oct. 1992.

[5] R. P. Kurshan and K. L. McMillan. Analysis of digital circuits through symbolic reduction. *IEEE Trans. Computer-Aided Design*, 10(11):1356–1371, November 1990.

[6] M. D. Mesarović, editor. *Views on General Systems Theory*. Wiley & Sons, NY, 1964.

[7] R. E. Miller. *Sequential Circuits and Machines*, volume 2 of *Switching Theory*. Wiley, 1965.

[8] D. E. Muller. Treatment of transition signals in electronic switching circuits by algebraic methods. *IRE Transactions on Electronic Computers*, EC-8(3):401, September 1959.

[9] P. R. Stephan. *Logical Mapping of Analog Waveforms in Digital Sequential Circuits*. PhD thesis, U.C. Berkeley, 1993. (in preparation).