A Hybrid-Systems View of Discrete Computation

Mark R. Greenstreet
Computer Science Department, University of British Columbia
Vancouver, British Columbia, Canada
mrg@cs.ubc.ca

I. Introduction

Discrete abstractions have allowed computer architects to design high performance machines that operate at ever increasing clock rates and with high degrees of parallelism. The discrete operations of these designs are implemented by circuits that are inherently analog. To obtain the highest possible performance, integrated circuit designers sometimes exploit analog properties of these circuits. Circuits with critical timing races, delayed clocks, multiple logic levels, etc., are in common use. Currently, such designs are verified by extensive simulation and the designer's intuition that "all possible behaviors" have been considered.

This paper presents a more systematic approach. CMOS digital design is introduced in section II. Section III presents continuous models for CMOS circuits. In section IV, abstraction mappings are presented that map continuous trajectories to sequences of discrete events. The key idea in this approach is that discrete behavior corresponds to topological features of invariant sets of the ODE model. Section V employs these ideas to examine the role of bifurcations in digital designs; synchronizers, sense amplifiers, and clock distribution are used as examples.

This paper emphasizes connections between the design and verification of VLSI circuits and similar problems that occur in control. Many control systems combine discrete and continuous models because a digital computer is used to control a plant with continuous processes. Digital VLSI design combines discrete and continuous models because the circuits that implement digital behavior are most accurately modeled using differential equations. In both cases, the mapping between continuous and discrete behaviors is critical. These mappings are the focus of this paper.

II. Digital CMOS Circuits

The operation of many digital circuits can be understood by considering a MOSFET to be a voltage controlled switch. In CMOS technologies, there are two types of transistors: N-channel and P-channel. As shown in figure 1, an N-channel transistor establishes a connection between its source (i.e. `s`) and drain (i.e. `d`) terminals when the voltage on the gate is high (relative to the minimum of the source and drain voltages. There is no connection when the gate is low. A P-channel device, on the other hand, establishes a connection when the gate is low and makes now connection when the gate is high. This "inverted" behavior is indicated by the "bubble" on the gate of the P-channel device.

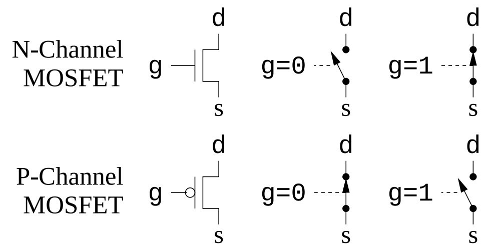

Fig. 1. N- and P-channel MOSFETs

Figure 2 shows implementations of an inverter and a NAND gate with switch model diagrams that indicate their operation. The symbol $\top$ represents the power supply which is always at the logical high level (i.e. 1). Likewise, the symbol ⏚ represents ground which is always at the logical low level (i.e. 0). When the input of the inverter is low, its P-channel transistor connects `y` to the power supply, and the N-channel transistor make no connection; thus, the output is high. Likewise, when the input of the inverter is high, the output is connected to ground and is therefore low. The operation of the NAND gate is similar. If either input is low, the corresponding P-channel transistor makes a connection from `y` to the power supply; the corresponding N-channel transistor breaks the connection from `y` to ground; and the output is driven high. If both inputs are high, then the output will low (i.e. connected to ground). Other logic functions can be implemented in a similar manner. For a general discussion of logic design using CMOS circuits, see [WE93], chapter 5.

To implement interesting computations, state

Currently on leave at the University of California, Berkeley. This work was supported in part by Canadian NSERC research grant OGP-0138501 a collaborative research grant from BC-ASI and SUN Microsystems, and United States Army Research Office grant ARO-MURI DAA404-96-1-0341

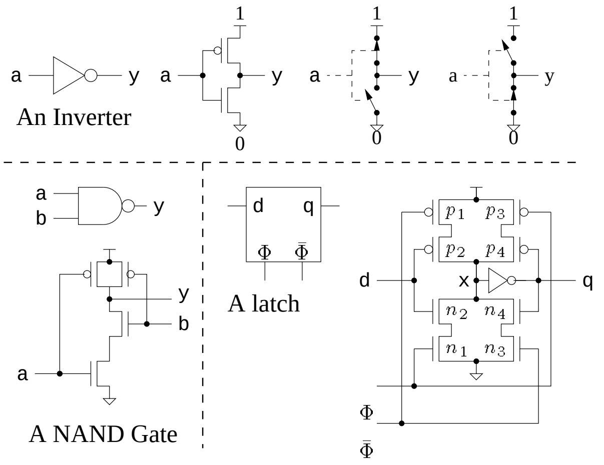

Fig. 2. Simple logic elements

holding devices are also required. The latch circuit shown in figure 2 implements a transparent latch. The data input is labeled `d`, and the data output is labeled `q`. The clock inputs are Φ and $\bar{\Phi}$ where $\bar{\Phi}$ is the logical negation of Φ. When Φ is high, transistors n_1 and p_1 make connections, whereas transistors n_3 and p_3 do not. In this case, `x` is driven to the logical negation of `d`, and therefore `q` is driven to the value of `d`. When Φ is low, transistors n_3 and p_3 make connections; `x` is driven to the logical negation of `q`; and `q` is driven to the logical negation of `x`. In this configuration, the values of `x` and `q` are stable. In other words, when Φ is high, the output `q` follows the value of the input `d`; when Φ is low, the output retains the value it had the last time Φ was high. This is the behavior of a transparent latch.

Arbitrary computations can be implemented using the combinational (i.e. not state holding) logic elements and transparent latches described above. Here, we describe the design of autonomous (no external inputs) finite state machines. Machines that interact with their environments are discussed in section V. An autonomous machine has some set of states, $\mathcal{S}$, and a state transition function: $F : \mathcal{S} \rightarrow \mathcal{S}$. Let F_1, F_2 be a decomposition of F; in other words, $F = F_2 \circ F_1$. Figure 3 shows a direct implementation of this autonomous machine. The clocks Φ_1 and Φ_2 are non-overlapping (i.e. Φ_1 and Φ_2 are never both high at the same time) with the same period. Signals `a` and `d` take values in $\mathcal{S}$, and `c` and `d` take values in the domain of F_2. Consider a state where Φ_1 and Φ_2 are both low and the value of `a` is a_0. When the combinational logic in module F_1 has settled to its equilibrium state, the value of `b` will be $F_1(a_0)$. When Φ_1 goes high, Φ_2 will still be low (because the two clocks are non-overlapping). This will lead to a state with `a` $= a_0$ (because `a` is unchanged), and `c` $=$ `b` $= F_1(a_0)$. Eventually, Φ_1 will return to low,

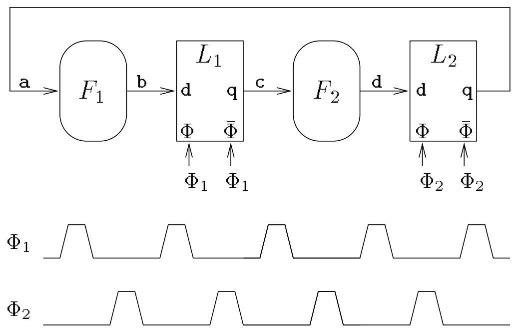

Fig. 3. A two-phase implementation of an autonomous, finite state machine

and the logic in module F_2 will settle to its equilibrium state with `d` $= F_2(c) = F(a_0)$. Then, after Φ_2 goes high, `a` will acquire the value $F(a_0)$, and the machine will have completed one step of execution.

This section has presented a simple introduction to digital, CMOS circuits. The circuits described here are quite conservative. No attempt has been made to be comprehensive in describing the wide variety of circuits in actual practice including precharged logic, dynamic storage, pass transistors, single-phase clocking, sense-amplifiers, boot-strap circuits, etc. Each of these introduce modeling, optimization, and verification problems that are promising areas for interactions between VLSI designers and control experts. Bakoglu [Bak90] provides a good survey of such design techniques.

III. ODE Models for CMOS Circuits

The previous section described CMOS circuits modeling transistors as switches: transistors were considered to be either "on" (i.e. making a connection) or "off" (i.e. not making a connection). This was sufficient to explain the operation of simple gates and latches. However, to understand trade-offs affecting the performance of these circuits, and to understand the operation of more subtle designs, a continuous model of circuit behavior is required. Here, we use models based on ordinary differential equations. These models have the advantage that they correspond to the best available physical models of the devices; they are the models in widespread use in circuit simulation programs such as SPICE [Nag75]; and they are amenable to analysis by methods from dynamical systems theory.

In a very simple model, a transistor has three terminals: the source, the drain, and the gate. The source and drain are symmetric, the use of two different names is primarily historical. The current flowing from the drain to the source is determined by the voltages on these three nodes. Thus, for each transistor, we have

$$i_{ds} = f(v_{gs}, v_{ds}) \quad (1)$$

where i_{ds} is the current flowing from the drain to the source, v_{gs} is the difference between the gate and source voltages, and v_{ds} is the difference between the drain and source voltages. A detailed discussion of transistor models is beyond the scope of this paper; the interested reader is referred to [GD85], chapter 2, for an introduction to this topic. For the sake of this introduction, we note that transistors are "enhancement mode" devices in typical CMOS processes. This means that there is a "threshold voltage" such that if v_{gs} is less than this threshold voltage, then i_{ds} is extremely small. For v_{gs} greater than the threshold, i_{ds} increases monotonically with v_{gs} and v_{ds}. These properties give rise to the switch-like properties described in section II.

Modeling a CMOS circuit as a collection of transistors and capacitors, let the vector i_T denote the total current flowing out of each node through transistors, and let i_C denote the total current flowing out through capacitors. By Kirchoff's current law, the sum of the currents flowing out of each node must be zero. Thus, $i_C + i_T = 0$. The current through a capacitor is equal to the capacitance of the capacitor times the derivate of the voltage across the capacitor. This capacitance can be represented by a matrix, C. Because the distribution of charges in a semiconductor depends strongly on the applied electric field, C is a function of node voltages. We write $i_C = C(v)\dot{v}$. Physical properties of the circuits ensure that $C(v)$ is real-symmetric and positive definite. Thus, $C(v)$ is invertible. This yields:

$$\dot{v} \quad = \quad -C^{-1}(v) i_T \tag{2}$$

This provides the non-linear, ODE model that captures the key aspects of circuit behavior. The more detailed models that are used in simulators such as SPICE are based on the same principles but provide much more detailed models of device behavior.

IV. Mappings from Continuous to Discrete Behaviors

In the previous two sections, both continuous and discrete models for CMOS circuits have been presented. Our goal is to use properties of the continuous dynamics of these circuits to verify that these designs function correctly and to improve their performance. To do this, we must understand the relationship between continuous and discrete behaviors. In other words, an abstraction mapping is needed.

A. Combinational logic

Combinational logic refers to circuits composed from simple gates where there are no cycles in the input-output relations of these gates. For example, the inverter and NAND gate shown in figure 2 is combinational, but the transparent latch is not. The inputs of such a network are those nodes that are not connected to the output of any gate in the network.

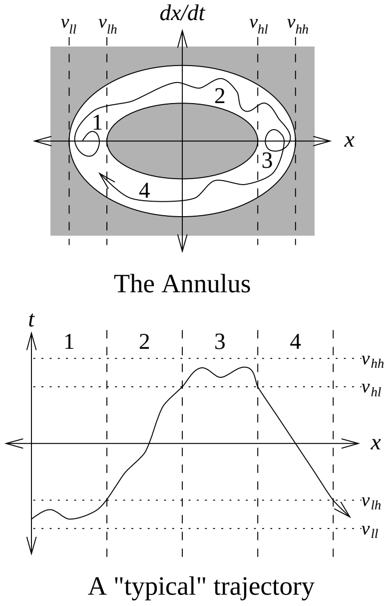

Fig. 4. Brockett's Annulus

If we assume an "inputs as parameters" perspective, then verifying the functionality and performance of the circuit is relatively straightforward. For each signal, we can define an interval in which the input is considered true and a separate interval in which the interval is considered false. These two intervals are separated by an interval in which the signal does not have a well-defined discrete interpretation. It is readily shown that a combinational logic circuit has a single fixed-point attractor for any fixed value of the inputs. Thus, verifying functionality amounts to showing that for any well-defined set of values on the inputs, the fixed-point attractor is at a location where all of the outputs have well-defined values that correspond to the desired logical function of the circuit.

To analyse performance, we must consider the fact that outputs and internal nodes will change while the inputs are changing. Thus, the set of possible input waveforms must be considered. Such a characterization of trajectories is described below.

B. Clocked circuits

Rather than settling to a single state, clocked circuits transition through sequences of states. An abstraction mapping for these circuits must consider not only the levels but also the transitions of signals. We use an annulus proposed by Brockett [Bro89] to map continuous trajectories to discrete behaviors. When a variable is in region 1, its value is constrained but its derivative may be either positive or negative. When the variable leaves region 1, it must enter region 2. Because the derivative of the variable is positive in this region, it makes a monotonic transition leading to region 3. Regions 3 and 4 are analogous to regions 1 and 2 corresponding to logically high and

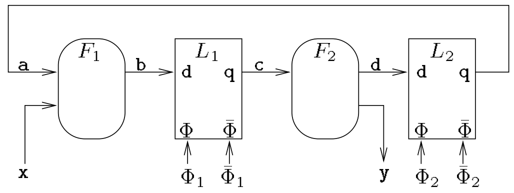

Fig. 5. A synchronous circuit with external inputs and outputs

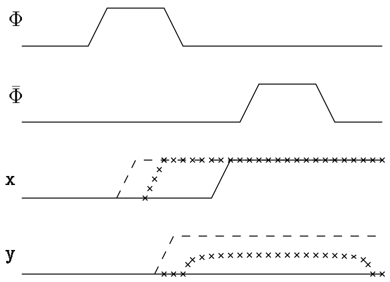

Fig. 6. Bifurcating trajectories

monotonically falling signals respectively. Because transitions through regions 2 and 4 are monotonic, traversals of these regions are distinct events. This provides a topological basis for discrete behaviors. Furthermore, the horizontal radii of the annulus define the maximum and minimum high and low levels of the signal (i.e. v_{ll}, v_{lh}, v_{hl}, and v_{hh} in figure 4). The maximum and minimum rise time for the signal correspond to trajectories along the upper-inner and upper-outer boundaries of the annulus respectively. Likewise, the lower-inner and lower-outer boundaries of the annulus specify the maximum and minimum fall times. This construction allows a large class of input signals to be described in a simple and natural manner.

To verify safety properties of a continuous model, we establish the existence of an invariant set in its phase space. We then show that all trajectories starting from points in the initial region are contained in this set, and that all trajectories in this set satisfy the specification (see [Bro89], [Gre96], [Gre97]). This technique is the continuous analog of using discrete invariants to verify properties of state transition systems [LS84].

V. Bifurcations

The behaviors arising from ODE models are continuous trajectories. Using Brockett annuli, these continuous trajectories can be interpreted as sequences of discrete states. In general, these trajectories are asymptotically stable – a slight perturbation of a trajectory at one clock cycle should have little effect on the continuous state many cycles later. This is what makes digital designs robust in the presence of noise, and is what allows computers to operate for literally trillions of clock cycles in a precise and repeatable manner.

These asymptotically stable trajectories are separated by unstable manifolds for trajectories that can only be explained by considering the underlying continuous model. This section considers these behaviors along with circuits that are designed either to avoid or exploit these unstable manifolds.

A. Synchronizers

In section II, autonomous systems were presented. Now, consider a system with external inputs as shown in figure 5: some inputs to combinational block F_1 come from the environment to communicate external events (e.g. keyboard events, sensor readings, data received from other computers, etc.); likewise, some outputs of F_2 are sent to the environment (e.g. to displays, actuators, storage devices, other computers, etc.). In many cases, it is impractical or impossible to control when external inputs arrive relative to the clock. This uncertainity corresponds to bifurcations of the trajectories of the continuous model according to the times at which input events occur. For some input events, the resulting trajectories do not have valid, discrete abstractions. These "metastability" events can cause failures of the design (see [CM73], [MS93]).

As a specific example, consider a circuit that counts the number of times the input is at a high level. In particular, assume that `a`, `b`, `c`, and `d` are each eight-bit values representing integers in the range $0 \ldots 255$. Let `x` be a single bit representing either 0 or 1. Let F_1 compute the sum of `a` and `x` (mod 256), and let F_2 be the identity function. The output, `y`, is the most significant bit of `d`.

Figure 6 depicts a low-to-high event on `x` when `a` has the value 01111111. If the transition of `x` occurs sufficiently after the falling edge of Φ_1 (solid lines for `x` and `y` in figure 6), then `x` is seen as low during the current clock cycle, the values of `b` and `c` and `d` acquire the same value as `a`, and `y` remains low. Conversely, if the transition of `x` occurs sufficiently before the falling edge of Φ_1 (dashed lines in figure 6), then `y` is seen as high during the current clock cycle, `b`, `c`, and `d` acquire the value of 10000000, and `y` transitions to high. We can think of `x` as a family of signals with a parameter μ such that with input $\mathtt{x}(\mu_{\text{early}})$, `y` is high when Φ_2 makes its high-to-low transition, and with input $\mathtt{x}(\mu_{\text{late}})$, `y` is low when Φ_2 falls. Our circuit models are smooth; therefore, the value of `y` when Φ_2 falls is a continuous function of μ. This means there is a μ_{mid} with $\mu_{\text{early}} < \mu_{\text{mid}} < \mu_{\text{late}}$ such that with input $\mathtt{x}(\mu_{\text{mid}})$, `y` is half-way between the high an low values when Φ_2 falls. Such a trajectory is depicted by the lines of $\times$'s in figure 6. This trajectory has no valid, discrete abstraction, and indicates a failure mode of the system. This failure is called a

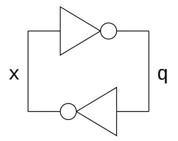

Fig. 7. A pair of inverters

"synchronization error."

At a lower level, consider the operation of the latch shown in figure 2. As described in section II, when Φ is low and $\bar{\phi}$ is high, the circuit behaves like the pair of inverters shown in figure 7. This circuit has three equilibrium points: `x` low and `q` high, `x` high and `q` low, and a point where both `x` and `q` have intermediate values. The first two equilibria are stable, and the last one is unstable. If the `d` input of the latch is changing when Φ falls, the values of `d` and `q` may be close to the stable manifold of the unstable equilibrium. In this situation, an arbitrarily long time may elapse after the fall of Φ before `q` reaches a well-defined discrete value.

Synchronization errors are unavoidable if a clocked, digital system must respond to inputs whose event timings are not determined by the clock. This observation runs counter to the intuition of many digital designers, but is apparent when viewed from a phase-space perspective: there must be a stable manifold separating the basins of attraction for the two discrete states corresponding to the "event occurred" and "event did not occur" scenarios. In the simplest case (see [Hur75], [MS92], [MG96]) this manifold has a single attractor, and this attractor corresponds to a saddle-point (see [HS74]) in the full state space. In the neighborhood of the saddle-point, the Jacobian of the derivative function has one positive eigenvalue. Thus, for trajectories chosen with starting point distributed uniformly in some neighborhood of the saddle-point, the probability of not reaching a well-defined, discrete state within some time τ, decreases exponentially with τ.

To build a reliable system, the probability of failure must be minimized. This can be done by increasing the speed of the circuits used (i.e. increasing the value of the positive eigenvalue of the Jacobian) and/or by increasing the time allowed between sampling an external signal and using it as a digital value. This time may be critical to system performance which motivates designing faster synchronizers as well as more accurately modeling existing designs. An example of the former is the "pipeline synchronizer" presented by Seizovic [Sei94] where the latency must be large enough to ensure reliable synchronization, but many synchronizations can be in progress at the same time allowing the throughput to be greater than the inverse of the latency. As examples of the latter, many analyses of synchronizer failure probabilities have been published including [Vee80], [HLS84], [HEC89]. In spite of this, much folklore persists in the design of synchronizer circuits; developing more systematic approaches to design appears to be an area where control experts and circuit designers could collaborate productively.

B. Sense Amplifiers

Memory components are ubiquitous in computer systems. In a DRAM, bits are stored on capacitors, and the value of a stored bit is read by connecting this capacitor to a wire called the "bit line" and sensing the change of voltage on this wire due to the presence or absence of charge on the capacitor. The connection is made using a transistor whose source forms the capacitor and whose drain is connected to the bit line. In a typical DRAM, around a thousand transistors are connected to the same bit line; thus, the ratio of the storage capacitance to the bit-line capacitance is quite small. Reading the value stored in a DRAM requires sensing a voltage change on the order of a millivolt. This is done using a sense-amplifier circuit which exploits the same bifurcation phenomenon described above.

A simple sense amplifier consists of two inverters connected as shown in figure 7 (see [Bak90], fig.16). To read a value from a DRAM, both the `d` and `x` nodes are forced to the same voltage, typically the midpoint between power and ground. The inverters are designed to be closely matched, so this is near the stable manifold for the circuit's saddle point. Then, `d` is connected to the bit-line that is to be read, and `x` is connected to a "dummy" line that reads a value of 1/2. The difference between the two bit-lines drives the trajectory away from the stable manifold to the attractor with `d` high if the value stored in the capacitor was a 1 and to the attractor with `d` low if a 0 had been stored. Reading is "destructive;" after the value has been sensed, the original charge must be restored on the storage capacitor.

Sense amplifiers are becoming increasingly common in digital design with applications including low-voltage swing I/O pads (to reduce power consumption), applications in cache memory designs (to increase speed), etc. Bakoglu [Bak90] provides many examples. The design of a fast sense amplifier depends on the dynamics of the amplifier circuit as well as the characteristics of the signals being sensed. It seems likely that the characterizations provided by Brockett annuli and similar techniques may lead to better amplifier designs. This is another promising area for interaction between the control community and VLSI designers.

C. Clock Distribution Networks

A limitation of the performance of synchronous designs it the uncertainty in the delays in the interconnect between the clock generator and the circuits that it controls. There are theoretical results showing that passive clock distribution networks must introduce skew that grows with the size of the network (see [FK85], [DS91]); in practice, this is reflected by designs with large clock drivers, extensive clock distribution networks, and high power dissipation. An alternative is to use an active network composed of coupled phase-locked loops. Nowatzyk [Now89] and Pratt and Nguyen [PN95] have shown that such coupled networks can overcome many of the limitations of passive schemes. There designs require arguments about global stability of their networks, and this appears to be yet another area where ideas from the control community can contribute to VLSI design.

VI. Conclusions

This paper has presented an overview of systems with ODE models can be shown to implement discrete behaviors. This problem is important in the design, verification, and optimization of VLSI designs. Rather than using a mixed (i.e. hybrid) model, the approach described here uses two separate models: one purely continuous, and the other purely discrete. From this perspective, the distinction between continuous and discrete is one of an abstraction mapping. Presumably, hybrid systems could be understood as ones where some of their behavior is interpreted in the continuous domain and some in a discrete domain. Exploring applications to hybrid control problems is a problem for future work.

The methods that are used here to analyze VLSI circuits are similar to those that are used in control theory to verify and optimize controllers. By describing the VLSI design process and illustrating verification and optimization problems that occur in these designs, it is the author's hope that this paper will encourage researchers in both disciplines to better understand both fields and thereby contribute to research in both.

Acknowledgements

The work describe here has benefited from interaction with excellent colleagues from a variety of disciplines. In particular, I am indebted to Roger Brockett for several enlightening discussions of discrete computation and control theory. I also wish to thank the Avalanche design group at the university of Utah, Jo Ebergen, Ian Mitchell, and Peggy Pang for helpful discussions and comments.

References

[Bak90] H.B. Bakoglu. *Circuits, Interconnections, and Packaging for VLSI.* Addison-Wesley, 1990.

[Bro89] R. W. Brockett. Smooth dynamical systems which realize arithmetical and logical operations. In Hendrik Nijmeijer and Johannes M. Schumacher, editors, *Three Decades of Mathematical Systems Theory: A Collection of Surveys at the Occasion of the 50th Birthday of J. C. Willems*, volume 135 of *Lecture Notes in Control and Information Sciences*, pages 19–30. Springer, 1989.

[CM73] T.J. Chaney and C.E. Molnar. Anomalous behavior of synchronizer and arbiter circuits. *IEEE Transactions on Computers*, C-22(4):421–422, April 1973.

[DS91] Marios D. Dikaiakos and Kenneth Steiglitz. Comparison of tree and straight-line clocking for long systolic arrays. *Journal of VLSI Signal Processing*, 3(4), 1991.

[FK85] Allan L. Fisher and H.T. Kung. Synchronizing large VLSI processor arrays. *IEEE Transactions on Computers*, C-34(8):734–740, August 1985.

[GD85] Lance A. Glasser and Daniel W. Dobberpuhl. *The Design and Analysis of VLSI Circuits.* Addison-Wesley, 1985.

[Gre96] Mark R. Greenstreet. Verifying safety properties of differential equations. In *Proceedings of the 1996 Conference on Computer Aided Verification*, pages 277–287, New Brunswick, NJ, July 1996.

[Gre97] Mark R. Greenstreet. A smooth dynamical system that counts in binary. In *Proceedings of the 1997 International Symposium on Circuits and Systems*, volume II, pages 977–980, Hong Kong, June 1997. IEEE.

[HEC89] Jens U. Horstmann, Hans W. Eichel, and Robert L. Coats. Metatstability behavior of cmos asic flip-flops in theory and test. *IEEE Journal of Solid-State Circuits*, 24(1):146–157, February 1989.

[HLS84] Jakob H. Hohl, Wendell R. Larsen, and Larry C. Schooley. Prediction of error probabilities for integrated digital synchronizers. *IEEE Journal of Solid-State Circuits*, SC-19(2):236–244, April 1984.

[HS74] Morris W. Hirsch and Stephen Smale. *Differential Equations, Dynamical Systems, and Linear Algebra.* Academic Press, San Diego, CA, 1974.

[Hur75] Marco Hurtado. *Structure and Performance of Asymptotically Bistable Dynamical Systems.* PhD thesis, Sever Institute, Washington University, Saint Louis, MO, 1975.

[LS84] Leslie Lamport and Fred B. Schneider. The 'Hoare logic' of CSP, and all that. *ACM Transactions on Programming Languages*, 6(2), April 1984.

[MG96] Ian Mitchell and Mark Greenstreet. Proving newtonian arbiters correct, almost surely. In *Proceedings of the Third Workshop on Designing Correct Circuits*, Båstad, Sweden, September 1996.

[MS92] Michael Mendler and Terry Stroup. Newtonian arbiters cannot be proven correct. In *Proceedings of the 1992 Workshop on Designing Correct Circuits*, January 1992.

[MS93] Michael Mendler and Terry Stroup. Newtonian arbiters cannot be proven correct. *Formal Methods in System Design*, 3(3):233–257, December 1993.

[Nag75] L.W. Nagel. SPICE2: a computer program to simulate semiconductor circuits. Technical Report ERL-M520, Electronics Research Laboratory, University of California, Berkeley, CA, May 1975.

[Now89] Andreas Nowatzyk. *A Communication Architecture for Multiprocessor Networks.* PhD thesis, School of Computer Science, Carnegie Mellon University, December 1989. Technical report CMU-CS-89-181.

[PN95] Gill A. Pratt and John Nguyen. Distributed synchronous clocking. *IEEE Transactions on Parallel and Distributed Systems*, 6(3):314–328, March 1995.

[Sei94] Jakov N. Seizovic. Pipeline synchronization. In *Proceedings of the 1994 International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 87–96. IEEE Computer Society Press, 1994.

[Vee80] Harry J.M. Veendrick. The behavior of flip-flops used as synchronizers and prediction of their failure rate. *IEEE Journal of Solid State Circuits*, SC-15(2):169–176, April 1980.

[WE93] Neil H.E. Weste and Kamran Eshragian. *Principles of CMOS VLSI Design; A Systems Perspective.* Addison-Wesley, second edition, 1993.