Verifying Safety Properties of Differential Equations

Mark R. Greenstreet, mrg@cs.ubc.ca
Department of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4
Canada

Abstract

This paper presents an approach to verifying that a circuit as described by a continuous, differential equation model is a correct implementation of a discrete specification. The abstraction from continuous trajectories to discrete traces is based on topological features of the continuous model rather than quantizing the continuous phase space. An practical verification method based on numerical integration is presented. The method is demonstrated by the verification that a toggle circuit satisfies a discrete specification.

1 Introduction

Most research in hardware verification has been based on discrete models for circuit behavior [Gup92]. In many high performance designs, discrete models are inadequate. Details of transition times, slew rates, capacitive coupling, etc. can be crucial for the correct operation of such designs. Accurate models for these phenomena are typically expressed as systems of non-linear differential equations. Thus, it becomes important to verify that a circuit modeled by non-linear differential equations is a valid implementation of a discrete specification.

This paper presents an approach to this problem based on methods from dynamical systems theory. Safety properties of the continuous model are verified by demonstrating the existence of suitable invariant manifolds in the continuous phase space. Interface specifications are expressed as constraints on the relationship between signals and their time derivatives, and continuous trajectories are mapped to discrete traces rather than attempting to discretize the continuous phase space to define discrete states. This topological approach to describing behavior is described in section 3.

Section 5 describes the verification method. The continuous system is verified by computing a conservative bound on the reachable region of the system throughout a continuous integration. The reachable region is represented by its projection onto planes defined by pairs of the continuous variables. This approach allows standard computational geometry algorithms to be used to maintain the data structure for the reachable region, and it avoids the exponential complexity of explicitly representing a high-dimensional object. A numerical integrator is used to determine the evolution of the reachable region. This approach is demonstrated by verifying that a toggle circuit implements its discrete specification. The circuit is described in section 4, and section 6 presents its verification.

Recently, there has been a large interest in the verification of continuous systems. Much of this is based on linear automata models [OSY94] which cannot be applied directly to the non-linear models of VLSI circuits. Henzinger and Ho [HH95] showed how these methods could be applied to non-linear systems by constructing asymptotically equivalent linear descriptions. The approach presented here differs from theirs in that the verification is performed directly on the system on non-linear differential equations. This facilitates using ideas from dynamical systems theory both for the specification and the verification of the design. Kurshan and McMillan [KM91] presented the verification of an arbiter circuit using a circuit model similar to the one used this paper. They partitioned the phase space into fixed boxes and computed a next-state relation between these boxes by integrating the non-linear circuit model for fixed time steps. This leads to an interaction of the time step size and the box size that is avoided by the methods presented here.

2 A Discrete Toggle

In this paper, discrete behaviors are described using finite state automata. A finite state automaton is described by a quadruple (I, O, Δ, Q_0), where I is a set of binary valued inputs and O is a set of of binary valued outputs. The state space, S, is $2^{I \cup O}$. For $s \in S$, $v(s)$ denotes the value of input or output v in state s. The set of initial states is given by Q_0, with $Q_0 \subseteq S$, and Δ is the state transition relation with $\Delta \subseteq S \times S$. A **trace** is a sequence of states, $s_0, s_1, \ldots$, such that $s_0 \in Q_0$ and $\forall i.\, (i \geq 0) \Rightarrow (s_i, s_{i+1}) \in \Delta$. The state transition relation is partitioned into circuit actions and environment actions. A circuit action only changes the values of outputs and an environment action only changes the values of inputs. More formally,

$$\forall (s_1, s_2) \in \Delta.\, (\forall v \in I.\, v(s_1) = v(s_2)) \vee (\forall v \in O.\, v(s_1) = v(s_2))$$

A simple toggle element has a clock input Φ and two outputs $\mathtt{a}$ and $\mathtt{b}$. the singleton initial state set $\{(F, F, F)\}$, where F denotes the logical value false and state triples are written $(\Phi, \mathtt{a}, \mathtt{b})$. The state transition relation is

$$\begin{array}{l} \{\ ((F,F,F),\ (T,F,F)),\ ((T,F,F),\ (T,T,F)),\ ((T,T,F),\ (F,T,F)), \\ \quad ((F,T,F),\ (F,T,T)),\ ((F,T,T),\ (T,T,T)),\ ((T,T,T),\ (T,F,T)), \\ \quad ((T,F,T),\ (F,F,T)),\ ((F,F,T),\ (F,F,F))\ \} \end{array}$$

Figure 1 depicts the state space and state transition relation of this toggle embedded in a binary 3-cube. The salient features of this circuit description include:

Environment assumption: The clock input Φ only changes in states where no circuit actions are enabled.

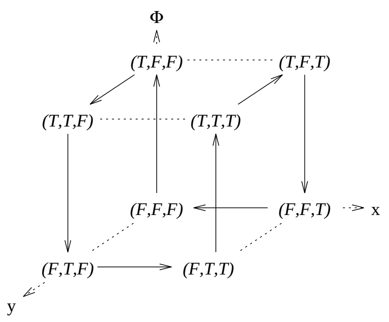


Fig. 1. A Discrete Toggle Element

Toggling: The period of the cycle of states for the toggle is twice the period of the clock input.

Compositional elements: The `a` output makes exactly one low-to-high and one high-to-low transition during each cycle of states. Likewise for the `b` output. This allows, for example, a counter to be constructed by connecting the output of one toggle element to the input of another as long as the environment assumption can be shown to be satisfied.

Continuous behaviors can be described using ordinary differential equations (ODEs). By analogy with a finite state automaton, we describe ODEs with a tuple, $(\mathcal{I}, \mathcal{P}_{\mathcal{I}}, \mathcal{O}, \delta, \mathcal{Q}_0)$. $\mathcal{I}$ is a set of real-valued inputs, and $\mathcal{P}_{\mathcal{I}}$ is a condition that must be satisfied by the inputs. For example, we assume that inputs are a continuous, bounded functions of time; additional conditions for inputs are described in section 3. $\mathcal{O}$ is a set or real-valued outputs. If the model has d_i inputs and d_o outputs, then the state space is R^d where $d = d_i + d_o$. The derivative function, $\delta : R^d \rightarrow R^{d_o}$, gives the time derivative of each output as a function of the current state. The initial point is any point in $\mathcal{Q}_0$ where $\mathcal{Q}_0 \subseteq R^d$. We assume that δ is Lipschitz, and that the inputs are continuous functions of time. These conditions guarantee that the outputs are uniquely defined for any inputs and initial state. We call a tuple, $(\mathcal{I}, \mathcal{P}_{\mathcal{I}}, \mathcal{O}, \delta, \mathcal{Q}_0)$ a **continuous model**.

A continuous model defines a set of trajectories. A trajectory, η is a differentiable function from time to R^d where

$$\begin{array}{l} \quad \eta(0) \in \mathcal{Q}_0 \\ \wedge \, \mathcal{P}_{\mathcal{I}}(\eta) \\ \wedge \, \dfrac{d}{dt}\mathcal{O}(\eta) = \delta(\mathcal{I}(\eta), \mathcal{O}(\eta)) \end{array}$$

Given a continuous model Ω and a finite state automaton F, an abstraction function maps trajectories of Ω to traces of F. We say that Ω is an implementation of F with abstraction function A iff for every trajectory η of Ω, $A(\eta)$ is a valid trace of F. Note that abstractions map continuous trajectories to discrete traces rather than states to states. In this approach, discrete behaviors can be understood as topological properties of families of trajectories which allows concepts from dynamical systems theory to be applied to the problem of verifying that a continuous model of a circuit satisfies a discrete specification.

3 Continuous Realizations of Discrete Behaviors

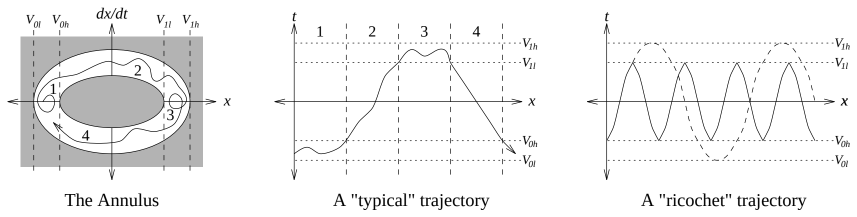


Fig. 2. Brockett's Annulus

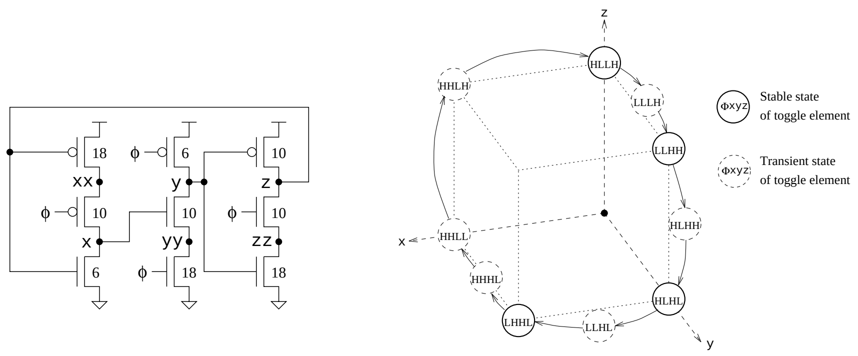
Figure 2 depicts an annulus proposed by Brockett [Bro89] that provides a topological basis for mapping continuous trajectories to discrete behaviors. When a variable is in region 1, its value is constrained but its derivative may be either positive or negative. When the variable leaves region 1, it must 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 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_{0l}, V_{0h}, V_{1l}, and V_{1h} in figure 2). 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.

If the annulus is given by two ellipses, then the trajectories corresponding to the inner and outer boundaries of the annulus are sine waves, and it is tempting to think of these as giving upper and lower bounds for the period of the signal. This is not the case. First, note that a signal may remain in regions 1 or 3 arbitrarily long. This is essential when verifying the toggle where we must show that the output satisfies the constraints assumed of the input, even though the period of the output is twice that of the input. Furthermore, the signal is not required to spend any time in regions 1 and 3. The minimum period signal corresponds to a "ricochet" trajectory as depicted by the solid curve in the right most plot of figure 2. The period of this signal can be much less than that of the sine wave corresponding to the outer boundary of the annulus (the dashed curve). It is desirable to independently specify constraints on signal levels, transition times, and period. We achieve this by imposing minimum times that a signal must remain in regions 1 and 3. 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 manifold in R^d. We then show that all trajectories starting from points in the initial region are contained in this manifold, and that all trajectories in this manifold satisfy the specification. This technique is the continuous analog of using discrete invariants to verify properties of state transition systems [LS84].

For the toggle element, all trajectories in the invariant manifold should have a period twice that of the clock signal. This notion can be formalized using a Poincaré section [PC89]. Let ϕ be the continuous signal corresponding to Φ, and let c be some constant with $V_{0h} < c < V_{1l}$. Consider the intersection of the manifold with the $\phi = c$ hyperplane. These intersections form a Poincaré map. This intersection must consist of four disjoint regions (two for rising ϕ crossing c, and two falling crossings) and all trajectories must visit these four regions in the same order. Continuous trajectories can be mapped to discrete sequences of the discrete toggle described in section 2 by mapping regions of the manifold to states of the discrete toggle. The toggle element is compositional if it there is an output variable such that for all trajectories in the manifold, the value and derivative of this variable satisfy the constraints of the input ring. It must also be shown that this output satisfies the minimum high and low time constraints.

4 A toggle circuit



Schematic State Transition Diagram

Fig. 3. Yuan and Svensson's Toggle

Figure 3 shows a toggle circuit that was originally published by Yuan and

Svensson [YS89]. The operation of this circuit can be understood by using a simple switch model starting from a state where the ϕ input is low. In this case, $\mathbf{y}$ will eventually become high, $\mathbf{z}$ is floating, and $\mathbf{x}$ is the logical negation of $\mathbf{z}$. If we assume that the value stored on node $\mathbf{z}$ is a well-defined logical value, then the circuit has two possible states when ϕ is low: $(\mathbf{x}, \mathbf{y}, \mathbf{z}) = (L, H, H)$, and $(\mathbf{x}, \mathbf{y}, \mathbf{z}) = (H, H, L)$. Starting from these two states, we can derive the corresponding stable successor states for when ϕ is high. If the circuit is allowed to reach a stable state before each transition of the ϕ input, then it implements a toggle as illustrated by the state transition diagram shown on the right half of figure 3.

The analysis presented in this paper is based on a simple circuit model using a standard, first-order transistor model [GD85] with three regions of operation: cut-off, saturation, and "linear." Capacitors are of fixed value, and all capacitances are to ground. Using basic circuit analysis techniques, we obtain a system of non-linear differential equations that is our continuous model for the circuit. A more detailed description of this model is given in [GC94].

5 Verification

Let Ω be a continuous model. Properties of Ω can be verified by finding a manifold that contains all trajectories of Ω. This can be done by starting with the initial region of the model and integrating the system of differential equations to compute a bounding region at each step. In the present work, variations in the input signal and initial state are considered, but the model parameters are fixed. Because the non-linear equations that arise from circuit models cannot be integrated analytically, this integration is performed numerically. Thus, this verification requires an assumption of the validity of the numerical integrator. The verification described in this paper uses a fourth-order Runge-Kutta integrator adapted from [PF$^+$88].

The Brockett annulus provides a convenient way to perform this integration. When the input signal ϕ is in the first or third region of figure 2, either the N-channel or the P-channel transistors controlled by ϕ are in cut-off. For typical CMOS circuits including the toggle this ensures that each node is either floating in which case the time derivative of its voltage is zero, or that it there is a conducting path to either Vdd or ground, but not both. In this case the voltage of the node asymptotically approaches the corresponding power supply value. Given a bounding region for trajectories upon entry to the first or third region of the annulus, we integrate for the minimum low or high time respectively and then determine the bounding box for the reachable region. For nodes that are asymptotically approaching a power supply value, the box is expanded to include that value. The expanded box is used as the starting region for the next phase of integration. When ϕ is in the second or fourth region, its value is changing monotonically. This allows the integration to be performed with respect to ϕ which reduces the dimension of the phase space by one and reflects the natural dependence of the circuit on its input.

At each integration step, the reachable region is implicitly represented by a set of two types of constraints: simple and polygonal. Simple constraints give upper and lower bounds for the value of a single variable. Polygonal constraints give bounds on the values of pairs of variables: the polygon is a projection of the reachable region onto the plane corresponding to the two variables. In the current implementation, polygonal constraints are represented by simple, rectilinear polygons without convexity restrictions. Each polygon corresponds to a prism in the complete phase space, and the reachable region is represented by the intersection of these prisms clipped by the simple constraints. This approach avoids the exponential growth in complexity that would occur if the reachable region were explicitly constructed and it allows efficient algorithms from two-dimensional computational geometry to be utilized. The representation is conservative; accordingly, our verification method is sound but not complete.

At each integration step, a conservative estimate of the bounding region is computed. For each face of the reachable region, a maximum outward translation is determined. This translation is an upper bound on the outward normal component of any trajectory starting from some point on the face. Since the entire face is translated outward by this amount, this bounds all trajectories starting from that face. By performing this computation for each face, a conservative estimate is obtained for the bounding region at the end of the integration step.

Focusing on the non-degenerate cases[1], each face of the reachable region corresponds to a bound of a simple constraint or an edge from a polygonal constraint. If the constraint corresponds to a polygon edge, it gives an exact value for one variable and a bounding interval for the other. If the constraint is an upper or lower bound of a simple constraint, it gives a value for that variable. Given these explicit constraints, bounds on other variables can be derived from the polygonal constraints. In this way, we compute bounding intervals for each variable for each face. From this, maximum and minimum values are computed for the outward component of the derivative vector for points on the face. For models arising from CMOS circuit models, this requires calculating upper and lower bounds for the drain current of each transistor, and the monotonicity of the transistor model simplifies this calculation. Because a fourth-order integration algorithm is used, four of these derivative calculations are performed at each step, and an error estimate is calculated to adjust the step size.

There are several details that must be considered. First, as the integration is performed, some polygon edges will grow. To avoid excessively conservative estimates of the reachable region, edges are split into smaller edges when they exceed a pre-specified length. Conversely, when adjacent edges become sufficiently short, they are conservatively merged into a single edge for efficiency. If two polygonal constraints involve the same variable, then they each have edges corresponding to the maximum and minimum values of this variable. When one of these extremal edges is split, then it may be possible to compute a tighter bound for its outward derivative than for the corresponding edge of the other

[1] Zero-length polygon edges and coincident constraints can occur as a consequence of constraint splitting described shortly.

polygon. In this case, the unsplit edge may acquire an infeasible value at the end of an integration step. When this occurs, the algorithm solves the system of constrains and moves the infeasible edge to its maximally outward feasible position.

In addition to the change of variables to integrate with respect to ϕ, it is sometimes convenient to perform additional changes of the variable of integration. Once it is shown that some variable, u, changes monotonically with respect to ϕ, then u can be used as the variable of integration. This can provide tighter estimates of the reachable region, but it requires a relation to bound ϕ given u. This relation can be represented by another polygon, and the polygon manipulation routines for the integrator can be used to perform this change of variables as well.

6 Verifying the Toggle Circuit

The toggle element of section 4 can be verified by chosing an initial region and integrating that region through two periods of the clock input as described in section 5. An invariant manifold is identified by showing that the reachable region at the end of this integration is contained in the initial region. By computing the intersection of this manifold with the $\phi = 2.5$volts hyperplane, it is shown that the manifold has a period that is twice that of ϕ as required. We use z as the output of the toggle, and by computing bounds on z and dz/dt at each integration step, we show that z satisfies the same ring constraints as the input. Details of this process are described in the remainder of this section.

The specification for the ϕ input is an annulus whose inner-boundary corresponds to a 100 MHz., 4.5 volt peak-to-peak sine wave centered at 2.5 volts. The outer boundary corresponds to a 700 MHz. 5.5 volt peak-to-peak sine wave also centered at 2.5 volts. The large difference between these frequencies demonstrates the robustness of the toggle to variations in the input signal. The minimum high and low times for ϕ are each 1 nano-second. This yields a minimum period of ϕ of ~ 2.87 nano-seconds which corresponds to a maximum frequency of 348 MHz.

The circuit model is simplified by assuming that the capacitances at nodes xx, yy, and zz are negligible, and a four-terminal device model is used for pairs of transistors of the same type in series. A 160 femtofarad load is added to the z node to simulate the effect of driving the ϕ input of another toggle element. "Typical" values for the MOSIS 2μ n-well CMOS process were used for the analysis. All transistors have a 2μ gate length and shape factors are shown in figure 3. Diffusion and gate capacitances are included in the model; for simplicity, interconnect capacitance is ignored.

The initial region is given by the constraints: $\phi = 0.25$; $-0.1 \leq x \leq 0.1$; $4.9 \leq y \leq 5.1$; and $4.8 < z < 5.1$. The integration starts with ϕ entering the second region of Brockett's annulus. The integration is performed in four phases: (1) ϕ rising and high, (2) ϕ falling and low, (3) ϕ rising and high, (4) ϕ falling and low. Each phase is started with a new bounding box, and at the end of each phase, we verify that the reachable region is contained in the initial bounding box

Fig. 4. Reachable region for z and dz/dt

for the next phase. This allows the four phases to be verified separately. After integration for two periods of ϕ, the reachable region satisfies the constraints: $\phi = 0.25$; $-1.57 * 10^{-13} \leq x \leq 1.61 * 10^{-13}$; $4.99 \leq y \leq 5.00$; and $4.83 < z < 5.05$. This demonstrates the existence of an invariant manifold as required.

In most of the phases, only a single variable has any large change in its value, and it is sufficient to approximate the reachable region by a bounding box. However, in the phase where ϕ and z make low-to-high transitions, x and y also make high-to-low transitions (see figure 3). The correct operation of the toggle requires that y complete its transition before x goes too far low. In this phase, coupling of each pair of variables with polygonal constraints was required along with an additional change of variable of integration from ϕ to y.

At each step of the integration, bounds on z and dz/dt are computed. These are shown in figure 4 with the annulus used to specify the input. It can be seen that z satisfies the specification for an input to the toggle. Furthermore, the integration shows that the minimum low-time for z is at least 2.74 nanoseconds and the minimum high time is at least 2.16 nanoseconds. Thus, z satisfies the requirements for an input signal to the toggle.

The current implementation of the verification algorithm is only for proof of concept and no effort has been for optimization. The run-time is dominated by the time for integration when z makes its low-to-high transition. When regions are estimated using polygons with a 0.25 volt nominal edge length, this step takes about forty minutes on a 50 MHz SPARC 10. When the nominal edge length is increased to 0.5 volts, the verification can be performed in just over five minutes. Further work will be required to optimize the implementation and characterize its performance on a larger set of examples.

7 Conclusions

This paper has presented a method for verifying that a circuit modeled by a system of non-linear differential equations satisfies a discrete specification. The approach is based on topological properties of the continuous model. Verification of the continuous model is performed by numerical integration to determine a manifold containing all feasible trajectories. Properties of the trajectories can be derived from the manifold by using methods from dynamical systems theory such as Poincaré sections.

The method has been applied to a toggle element. It was shown that the toggle operates correctly for a large class of input signals, and that its output satisfies the constraints required of its input. This means that these toggle elements can be connected in a chain to form a verified ripple counter. Although the toggle is a relatively simple circuit, its complexity is comparable to that of many cells in a typical standard-cell library. A potential application of these methods would be to verify that such a library has been properly designed and characterized.

Acknowledgements

I would like to express my appreciation to Peter Cahoon, Nick Pippenger, and Jim Varah for helpful suggestions throughout this work. Further thanks to the anonymous referees whose comments have contributed to the clarity of this paper.

References

[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-Verlag, 1989.

[GC94] Mark R. Greenstreet and Peter Cahoon. How fast will the flip flop? In *Proceedings of the 1994 International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 77–86, Salt Lake City, November 1994. IEEE Computer Society Press.

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

[Gup92] Aarti Gupta. Formal hardware verification methods: A survey. *Formal Methods in System Design*, 1(2/3):151–258, October 1992.

[HH95] Thomas A. Henzinger and Pei-Hsin Ho. Algorithmic analysis of nonlinear hybrid systems. In *Proceedings of CAV 95*, pages 225–238, 1995.

[KM91] R.P. Kurshan and K.L. McMillan. Analysis of digital circuits through symbolic reduction. *IEEE Transactions on Computer-Aided Design*, 10(11):1356–1371, November 1991.

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

[OSY94] A. Olivero, J. Sifakis, and S. Yovine. Using abstractions for the verification of linear hybrid systems. In David L. Dill, editor, *Proceedings of 1994 Workshop on Computer Aided Verification*, volume 818 of *Lecture Notes in Computer Science*, pages 81–94. Springer-Verlag, June 1994.

[PC89] Thomas S. Parker and Leon O. Chua. *Prectical Numerical Algorithms for Chaotic Systems*. Springer-Verlag, New York, 1989.

[PF$^+$88] William H. Press, Brian P. Flannery, et al. *Numerical Recipes in C: The art of numerical computing*. Cambridge University Press, 1988.

[YS89] Jiren Yuan and Christer Svensson. High-speed CMOS circuit technique. *IEEE Journal of Solid-State Circuits*, 24(1):62–70, February 1989.