From Proceedings of 1996 Workshop on Designing Correct Circuits
eds. M. Sheeran & S. Singh, Springer-Verlag, September, 1996
See http://www.springer.co.uk/ewic/workshops/DCC96/

Proving Newtonian Arbiters Correct, almost surely

Ian Mitchell & Mark R. Greenstreet
{mitchell,mrg}@cs.ubc.ca
Department of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4
Canada

Abstract

This paper presents the verification of safety and liveness for two arbiters modeled with differential equations. The first is a "toy" arbiter, whose model was chosen for its tractability; the second is Seitz's nMOS arbiter, chosen as an example of a widely used and studied design. Because an arbiter cannot be guaranteed to respond correctly in any bounded amount of time, we verify liveness in an almost surely sense—with probability 1, the arbiter eventually grants some pending request. Our analysis is more general than previously published results in that we allow the inputs of the arbiter to vary while the arbiter is in the metastable region, and we give conditions that ensure liveness even in the presence of such variations.

1 Introduction

An arbiter is a circuit that grants its clients mutually exclusive access to a shared resource. Since Chaney and Molnar's original description of metastability more than two decades ago [CM73], there has been an interest in analyzing the behaviour of arbiter circuits. Many proofs have been published showing that no arbiter can be implemented that is both safe and that is guaranteed always to respond in a bounded amount of time [Hur75, Cha84, MS93]. These proofs use continuity arguments to show that inputs must exist that will result in an output of the arbiter being at a logically undefined value arbitrarily long after a pair of concurrent requests have been made—the arbiter has entered a metastable state.

Fewer positive results are available for arbiters. For fixed inputs, a simple linearization argument shows that the probability of remaining in the metastable region decreases exponentially with time [Sei79] for many practical designs. More recently, Kurshan and McMillan [KM91] verified an arbiter model using a numerical integrator and a model checker (see section 6). This technique allows for varying inputs, but their analysis avoided unbounded metastable behaviour by adding an axiom excluding it from their model.

In this paper, dynamical systems theory is used to verify safety and liveness for two arbiters modeled by systems of non-linear differential equations. We show that dynamical systems theory provides a geometric framework that can be used to map from continuous behaviours to discrete ones. In addition to providing a basis for our proofs, this framework allows us to describe meaningful classes of input and output signals—we do not require the inputs to take on fixed values. The well-known negative results regarding arbiters show that an arbiter cannot be live for all possible inputs. Instead we show that the liveness requirements are satisfied *almost surely:* given a smooth probability measure over the space of initial node voltages, the set of points in this space for which the arbiter does not eventually grant some pending request has probability zero.

Some of the techniques used in this paper are similar to those used by Henzinger and Ho [HH95] in their work on nonlinear hybrid systems. The proof of liveness outside the metastable region works on the same principles as rate translation—conservative bounding of the flow by piecewise-linear envelopes. But it is the metastability that makes arbiters interesting; in that neighbourhood a number of constructs and new variables were needed before a comparable linear bounding argument could be used to extract the desired result.

2 Specifications and Models

We consider the problem of showing that a circuit, as modeled by a system of non-linear differential equations, is a correct implementation of a temporal logic formula. This section presents the modeling framework used in this paper: non-linear differential equations to model the circuit and its environment, a temporal logic defined on probability spaces, and an abstraction mapping between the continuous and discrete domains.

2.1 Continuous Models

A continuous model is a tuple, (F, Ω, P) where F derivative function on some d dimensional space, and (Ω, P) is a probability space. We require that F is C^1, (i.e. $\partial F(x)/\partial x$) is continuous. Furthermore, we assume that Ω is the Borel Σ-algebra over U with $U \subseteq \mathbf{R}^d$, and $\dim(U) = d$.

We model circuits using autonomous non-linear, ordinary differential equations. A circuit with m inputs and n outputs is modeled by a function that gives the time derivative of each output as a function of the current values of the inputs and outputs. An environment for the circuit has n inputs corresponding to the outputs of the circuit, m outputs corresponding to the inputs of the circuit, and l internal variables. The behaviour of the environment is given by a function (mapping $\mathbf{R}^{l+m+n}$ to $\mathbf{R}^{l+m}$) that gives the time derivatives of each output and internal variable of the environment. Let $d = l + m + n$ denote the total dimension of the circuit with its environment. A state of this system corresponds to a point in $\mathbf{R}^d$. We write F to denote the time derivative function for the system state, $F : \mathbf{R}^d \mapsto \mathbf{R}^d$.

By modeling the circuit and environment together, we obtain a closed system. A trajectory, x, is a function from times to states such that $\forall \tau \in \mathbf{R}^+ .\, \dot{x}(\tau) = F(x(\tau))$. A trajectory is completely determined by its initial state. For $x_0 \in \mathbf{R}^d$, let $\varphi(x_0)$ denote the trajectory that satisfies $\varphi(x_0)(0) = x_0$. We write $\varphi^\tau(x_0)$ to denote $\varphi(x_0)(\tau)$. If x is a trajectory and $\rho \in \mathbf{R}^+$, we write *suffix*(x, ρ) to denote $\varphi(x(\rho))$. Because F is C^1 in R^d, $\varphi^\tau(x_0)$ is defined for all $x_0 \in \mathbf{R}^d$ and all $\tau \in \mathbf{R}$. Furthermore, $\varphi^\tau()$ is a one-to-one mapping from $\mathbf{R}^d$ to $\mathbf{R}^d$, and $\varphi^\tau(x_0)$ is continuous in τ and x_0 (see [HS74], chapter 8). Therefore, $\varphi^\tau()$ preserves the dimensionality of subsets of $\mathbf{R}^d$ (see [HW41], theorem VI 7).

As mentioned above, (Ω, P) is a probability space: an initial state x_0 corresponds to an event in Ω. We restrict our attention to systems where P is smooth in the sense that if $Q \in \Omega$ is a set with $\dim(Q) \leq d - 1$, then $P(Q)$ must be zero. This condition rules out overly determined initial states where, for example, the value of some variable is exactly specified. For $B \subset \mathbf{R}^d$ and $\tau \in \mathbf{R}^+$, $((\varphi^{-\tau}(B)) \cap U) \in \Omega$, denotes the set of initial points that can reach a point in B after exactly τ time units. By the observations in the previous paragraph,

$$P(\varphi^{-\tau}(B)) > 0 \quad \Rightarrow \quad \dim(B) > d - 1 \tag{1}$$

This observation is central to our proofs that properties hold "almost surely."

2.2 Temporal formulas on probability spaces

Given the negative results for arbiters mentioned in section 1, our objective is to show that an arbiter satisfies its specification, including liveness, almost surely. The specification for a discrete model is naturally written as a temporal logic formula [Pnu81, RRH93], which motivates introducing a temporal logic defined relative to a probability space.

State predicate Q holds for trajectory x if Q is satisfied by state $x(0)$. Let G and H be temporal formulas. The formulas $\neg G$, $G \vee H$, $G \wedge H$, and $G \Rightarrow H$ have the obvious interpretations. The formula $\Box G$ (always G) holds for x if G holds for all suffixes of x; likewise, $\Diamond G$ (eventually G) holds if $\Box \neg G$ does not hold (recall that a trajectory is completely determined by its initial state). The formula $G \rightarrow H$ denotes a weak until:

$$\begin{array}{ll} \mathit{hold}(G \rightarrow H, x) & = \\ & \mathit{hold}(G, x) \Rightarrow (\exists \tau \in \mathbf{R}^+ . (\forall \rho \in [0, \tau).\, \mathit{hold}(G, \mathit{suffix}(x, \rho))) \wedge \mathit{hold}(H, \mathit{suffix}(x, \tau))) \\ \vee & \mathit{hold}(\Box G, x) \end{array}$$

The formula $G \leadsto H$ is an abbreviation for $G \Rightarrow \Diamond H$. For $\eta \in \mathbf{R}^+$, the formula $G \overset{\eta}{\mapsto\!\!\leadsto} H$ holds if H holds within η time units of G holding:

$$\mathit{hold}(G \overset{\eta}{\mapsto\!\!\leadsto} H, x) \quad = \quad \mathit{hold}(G) \Rightarrow \exists \tau \in [0, \eta).\, \mathit{hold}(H, \mathit{suffix}(x, \tau))$$

A formula is valid if it holds for all trajectories starting in Ω, and it is valid almost surely if it holds with probability 1:

$$\begin{aligned}
\mathbf{valid}(G) &= \forall \omega \in \Omega.\, \mathit{hold}(G, \omega) \\
\mathbf{as}(G) &= \exists A \subseteq \Omega.\, (\forall \omega \in A.\, \mathit{hold}(G, \omega)) \wedge (P(A) = 1)
\end{aligned}$$

2.3 Abstraction functions

This section has presented a framework for continuous models based on systems of non-linear differential equations. We are interested in showing that such models correctly implement discrete specifications. Such verification requires an abstraction function to map continuous trajectories to discrete sequences. When the discrete state space is finite, the abstraction function can be represented by an ensemble of state predicates. An abstraction function for an arbiter is given in section 4.3.

3 A Discrete Specification

For concreteness, we consider an arbiter with two clients. The interface between the arbiter and the first client consists of two boolean valued signals: a request, $\mathtt{r}_1$; and a grant, $\mathtt{g}_1$. Likewise, signals $\mathtt{r}_2$ and $\mathtt{g}_2$ are the interface between the arbiter and the second client. Recall that these signals are functions from the continuous state space to the booleans.

The allowed behaviour of the clients, C, is simple to specify. The communication between the clients and the arbiter follows a four-phase handshake: a client may only raise a request when its grant is false, and it may only lower a request when its grant is true. This is formalized by the temporal logic formula

$$C \quad \equiv \quad \bigwedge_{i=1}^{2} (\neg \mathtt{r}_i \rightarrow \neg \mathtt{g}_i) \wedge (\mathtt{r}_i \rightarrow \mathtt{g}_i) \tag{2}$$

Mendler and Stroup [MS93] have identified five attributes that capture the intuitive sense of an ideal arbiter:

1. **Closure**: No assumptions are made about the relative arrival times of the request signals. In particular, we are not allowed to specify that there is some minimum time separation between requests from different clients.
2. **Mutual Exclusion**: Both grants cannot be outstanding at the same time.
3. **Reset**: If a request has been withdrawn, the corresponding grant will be withdrawn after at most Γ time units. Furthermore, a grant cannot be given unless the corresponding request has occurred.
4. **Dominance**: If a grant has been given, it will not be withdrawn until the corresponding request has been withdrawn.
5. **Liveness**: If at least one request is outstanding, eventually one request will be granted.

As noted by Mendler and Stroup, no Newtonian arbiter can satisfy all five of these requirements. Therefore, we weaken the fifth requirement to:

5'. **Liveness (almost surely)**: If at least one request is outstanding, eventually one request will be granted almost surely.

Properties 2–5 are formalized in figure 1. The closure property is satisfied by *not* stating any assumptions about the relative arrival times of the request signals.

4 The Toy Arbiter

First, we consider a "toy" arbiter. This arbiter has a simple model that is amenable analysis, and many observations from the toy arbiter can be applied to the nMOS implementation considered in section 6.

$$\begin{array}{lcll}
G_2 & \equiv & \Box(\neg(\mathtt{g}_1 \wedge \mathtt{g}_2)) & \text{(* \textbf{Mutual Exclusion} *)} \\
G_3 & \equiv & (\bigwedge_{i=1}^{2} \Box(\neg \mathtt{r}_i \stackrel{\Gamma}{\leadsto} \neg \mathtt{g}_i)) \wedge (\bigwedge_{i=1}^{2} \Box(\neg \mathtt{g}_i \rightarrow \mathtt{r}_i)) & \text{(* \textbf{Reset} *)} \\
G_4 & \equiv & \bigwedge_{i=1}^{2} \Box(\mathtt{g}_i \rightarrow \neg \mathtt{r}_i) & \text{(* \textbf{Dominance} *)} \\
G_5 & \equiv & \bigwedge_{i=1}^{2} \Box((\mathtt{r}_1 \vee \mathtt{r}_2) \leadsto ((\mathtt{r}_1 \wedge \mathtt{g}_1) \vee (\mathtt{r}_2 \wedge \mathtt{g}_2))) & \text{(* \textbf{Liveness} *)}
\end{array}$$

We require: **valid**$(G_2 \wedge G_3 \wedge G_4) \wedge$ **as**(G_5)

Figure 1: Arbiter specification

4.1 The continuous model

The toy arbiter has two inputs, a and b, and two corresponding outputs, x and y. As described below in section 4.3, signals in the interval $[-1, -0.9)$ are interpreted as false and signals in the interval $[0.9, 1)$ are interpreted as true. The model is

$$\begin{array}{rcl} \dot{x} & = & a - 1 - y - ay - x - x^3 \\ \dot{y} & = & b - 1 - x - bx - y - y^3 \end{array} \tag{3}$$

which was arrived at by algebraic simplification from $\dot{x} = (1 - y)(a - x) - (1 + y)(1 + x) + (x - x^3)$ and its dual for $\dot{y}$. The first term of this equation causes x to follow a when y is near -1. The second term forces x to -1 when y is near $+1$. The final term causes the system to favor values of x near $+1$ or -1.

hard to print MatLab stuff goes here	hard to print MatLab stuff goes here	hard to print MatLab stuff goes here
$a \in$ [-1, -0.9), $b \in$ [-1, -0.9)	$a \in$ [-1, -0.9), $b \in [0.9, 1)$	$a \in [0.9, 1)$, $b \in [0.9, 1)$

Figure 2: Phase space portraits for the toy arbiter

The behaviour of the toy arbiter is revealed by its phase space portraits as shown in figure 2. When a and b are both low, the system has a single attractor with x and y both low. This case corresponds to the discrete state where both requests and both grants are false. When a is low and b is high, the system has a single attractor with x low and y high. States near this attractor correspond to giving a grant to b. The case with a high and b low is symmetric. When a and b are both high, the system has two attractors—one with x high and y low and the other x low and y high. Respectively, these attractors correspond to giving the grant to a or b. There is also an unstable attractor near the origin. It is this saddle that gives rise to metastable behaviours. Because the values of a and b are not fixed, the exact location of this

$$
\begin{array}{rcl}
\textit{waiting}(\sigma,\tau) & = & \forall \rho \in [0,\tau]. \\
& & \qquad (a(\varphi^{\rho}(\sigma)) > 0.9) \wedge (b(\varphi^{\rho}(\sigma)) > 0.9) \\
& & \quad \wedge \quad (x(\varphi^{\rho}(\sigma)) < 0.8) \wedge (y(\varphi^{\rho}(\sigma)) < 0.8) \\
\textit{ClientsSame}(\sigma_1,\sigma_2) & = & (|\sigma_1 - \sigma_2| = |x(\sigma_1) - x(\sigma_2)| + |y(\sigma_1) - y(\sigma_2)|) \\
\textit{gap}_{ab}(\sigma_1,\sigma_2,\tau) & = & |a(\varphi^{\tau}(\sigma_1)) - a(\varphi^{\tau}(\sigma_2))| + |b(\varphi^{\tau}(\sigma_1)) - b(\varphi^{\tau}(\sigma_2))| \\
\textit{gap}_{xy}(\sigma_1,\sigma_2,\tau) & = & \int_0^{\tau} |x(\varphi^{\rho}(\sigma_1)) - x(\varphi^{\rho}(\sigma_2))| + |y(\varphi^{\rho}(\sigma_1)) - y(\varphi^{\rho}(\sigma_2))| \, d\sigma \\
\alpha\text{-insensitive}(\sigma) & = & \forall \sigma' \in \mathbf{R}^d. \\
& & \qquad \textit{ClientsSame}(\sigma,\sigma') \\
& & \quad \Rightarrow \quad \forall \tau \in \mathbf{R}^{+}. \\
& & \qquad\qquad (\textit{waiting}(\sigma,\tau) \wedge \textit{waiting}(\sigma',\tau)) \\
& & \qquad \Rightarrow \quad (\textit{gap}_{ab}(\sigma,\sigma',\tau) < \alpha \cdot \textit{gap}_{xy}(\sigma,\sigma',\tau))
\end{array}
$$

Where σ is a state, τ is a time, and $|\sigma_1 - \sigma_2|$ denotes the L^1 norm (sum of absolute values) of the component-wise difference between the two states.

Figure 3: α-insensitivity

saddle point can vary. In fact, it can be located anywhere in a region of positive measure (i.e. area > 0). Even with such variations, we can show that it is not possible to trap a set of trajectories corresponding to an initial state region of positive measure. This fact leads to the almost surely liveness of the arbiter.

4.2 The clients

In addition to the requirement stated in equation 2, we require that client transitions be monotonic and that the value of a or b is relatively insensitive to variations of x and y when a request has been made but is not yet granted. We state these two requirements here because neither applies to the discrete abstraction of section 3 but they are required to exclude pathological inputs.

The monotonicity requirements for a are:

$$
\begin{array}{rl}
& \Box(a \in [-1,1)) \\
\wedge & \Box(((a \in [-0.9,0.9)) \wedge (\dot{a} > 0)) \rightarrow (a \in [0.9,1))) \\
\wedge & \Box(((a \in [-0.9,0.9)) \wedge (\dot{a} < 0)) \rightarrow (a \in [-1,-0.9)))
\end{array}
\tag{4}
$$

The monotonicity requirements for b are equivalent.

The requirement that a and b be relatively insensitive to x and y while waiting for a grant is to exclude the case where the clients are carefully designed feedback controllers that force the arbiter to remain in the neighborhood of what would otherwise be the saddle point. In real circuits, this insensitivity is achieved by "metastability filters" that have very low gains when the arbiter is in its metastable region. We say that the clients are α-insensitive to the outputs of the arbiter if, given two trajectories that start from states that only differ in their x and y components, the difference of the a and b components of these trajectories is bounded by α times the integral of the differences of their x and y components. We use this integral as a measure of the history of the outputs observable by the client. A formal definition is given in figure 3. The predicate *waiting*(σ,τ) holds if both clients wait throughout the first τ time units of the trace that starts at state σ. The predicate *ClientsSame*(σ_1,σ_2) holds if the only components of states σ_1 and σ_2 that differ are x and y. The functions $\textit{gap}_{ab}(\sigma_1,\sigma_2,\tau)$ and $\textit{gap}_{xy}(\sigma_1,\sigma_2,\tau)$ are the differences in the a and b components at time τ and the integral of the differences in the x and y components for traces starting from states σ_1 and σ_2. We exclude clients that are feedback controllers with the following requirement:

$$
\Diamond(\neg((0.9 \leq a) \wedge (0.9 \leq b) \wedge (x < 0.8) \wedge (y < 0.8)) \vee \alpha\text{-insensitive})
\tag{5}
$$

If both clients are waiting for a grant, equation 5 requires that eventually a grant will be given or the clients will reach a state where they are α-insensitive to the outputs of the arbiter. It is not required that the clients settle to a specific value or even that the clients settle at all. While waiting, the a and b signals may wander in the $[0.9, 1)$ interval—only the *dependence* of this wandering on the observed history of x and y is restricted.

Finally, to ensure that the requirement of bounded time for reset is satisfied, we require:

$$\begin{array}{ll} & \Box(((a < 0.9) \wedge (\dot{a} < 0)) \overset{\Gamma}{\leadsto} (a < 0.89)) \\ \wedge & \Box(((b < 0.9) \wedge (\dot{b} < 0)) \overset{\Gamma}{\leadsto} (b < 0.89)) \end{array} \tag{6}$$

where Γ is the bound from figure 1.

4.3 The abstraction function

Our abstraction function is

$$\begin{array}{lcl} \mathtt{r}_1 & = & (a \geq 0.9) \vee ((a \geq -0.9) \wedge (\dot{a} > 0)) \\ \mathtt{r}_2 & = & (b \geq 0.9) \vee ((b \geq -0.9) \wedge (\dot{b} > 0)) \\ \mathtt{g}_1 & = & (x \geq 0.8) \wedge ((a \geq 0.9) \vee ((a \geq 0.89) \wedge (\dot{a} < 0))) \\ \mathtt{g}_2 & = & (y \geq 0.8) \wedge ((b \geq 0.9) \vee ((b \geq 0.89) \wedge (\dot{b} < 0))) \end{array} \tag{7}$$

The mapping for $\mathtt{r}_1$ is simple: $\mathtt{r}_1$ is true if a is high or a is at an intermediate value and rising. The mapping for $\mathtt{r}_2$ is equivalent. Although we require a and b to change monotonically, we do not give any lower bound on their slopes and it is therefore not possible to guarantee that x and y change monotonically as well. Strengthening the conditions for a and b would complicate the proof in section 5. Instead we condition the mappings for $\mathtt{g}_1$ and $\mathtt{g}_2$ on the values of a and b respectively. Intuitively, this proviso means that a client cannot interpret a high level of x or y as a grant unless it is currently making a request. The term $(0.89 \leq a) \wedge (\dot{a} < 0)$ ensures that some time elapses between the removal of a request and the dropping of the grant. It makes our mapping structurally stable (i.e. robust to small variations of the model and mapping parameters).

5 Verifying the Toy

In this section, we show that the toy arbiter system fulfills each of the five properties of an almost ideal arbiter; one subsection below is devoted to each property.

5.1 Closure

Closure is not a requirement for the arbiter, but a limitation on what restrictions can be applied to environment. The restrictions in section 4.2 do not constrain the relative arrival time of requests. Accordingly, this requirement is satisfied.

5.2 Mutual Exclusion

The requirement is $\Box\neg(\mathtt{g}_1 \wedge \mathtt{g}_2)$. By the definition of the abstraction function for the toy, this is equivalent to

$$\Box\neg((a > 0.89) \wedge (x \geq 0.8) \wedge (b > 0.89) \wedge (y \geq 0.8))$$

which is implied by the statement proven below: $\Box(x + y \leq 0)$.

We first show $\Box(-1 \leq x \leq +1)$ and $\Box(-1 \leq y \leq +1)$. The initial conditions ensure that the system starts in these ranges, and the conditions for the clients ensure that $\Box(-1 \leq a \leq +1)$ and $\Box(-1 \leq b \leq +1)$. From the definition of $\dot{x}$ (equation 3) we get

$$(x = -1) \wedge (-1 \leq y \leq +1) \wedge (-1 \leq a \leq +1) \quad \Rightarrow \quad \dot{x} = a - y - ay + 1 \geq 0$$

This proof shows that a trajectory with $x = -1$ will not cross into $x < -1$; consequently, by the continuity of trajectories, no trajectory can ever cross into $x < -1$. Similar arguments show that the system maintains $x \leq +1$ and $-1 \leq y \leq +1$ as well.

We now show that $\Box(x+y \leq 0)$. This requirement is satisfied by the initial conditions. The proof then demands that no trajectory can cross from $x+y \leq 0$ into $x+y > 0$. Continuity of trajectories again makes it sufficient to show $(x+y=0) \Rightarrow (\dot{x}+\dot{y} \leq 0)$. Recalling the definitions of $\dot{x}$ and $\dot{y}$ from equation 3, we get:

$$
\begin{array}{lll}
 & x+y & = 0 \\
\Rightarrow & \dot{x}+\dot{y} & = (2b-2)+(a-b)(x+1) \\
 & & = (2a-2)+(b-a)(y+1)
\end{array}
$$

For $a \leq b$, note that in the second line above $(2b-2)$ and $(a-b)$ are non-positive, and $\Box(-1 \leq x)$ ensures that $(x+1)$ is non-negative. Thus $\dot{x}+\dot{y} \leq 0$ as required. A symmetric argument using the third line applies for $b \leq a$.

5.3 Reset

We show $\Box(\neg\mathtt{r}_1 \stackrel{\Gamma}{\mapsto\!\!\leadsto} \neg\mathtt{g}_1)$ and $\Box(\neg\mathtt{g}_1 \rightarrow \mathtt{r}_1)$. The argument for $\mathtt{r}_2$ and $\mathtt{g}_2$ is equivalent. By the definition of the abstraction function,

$$\neg\mathtt{r}_1 \quad \Rightarrow \quad (a \leq -0.9) \vee ((a \leq 0.9) \wedge (\dot{a} < 0))$$

By the monotonicity requirement for the client

$$((a > -0.9) \wedge (a \leq 0.9) \wedge (\dot{a} < 0)) \rightarrow (a \leq -0.9)$$

Accordingly, $((a \leq 0.9) \wedge (\dot{a} < 0)) \leadsto (a \leq -0.9)$. By the definition of the abstraction function, $\mathtt{g}_1 \Rightarrow (a > 0.89)$. Therefore, $\Box\neg\mathtt{r}_1 \leadsto \neg\mathtt{g}_1$ as required. Likewise by monotonicity, $(a \leq -0.9) \rightarrow ((a > -0.9) \wedge (\dot{a} > 0)$. This equation ensures $\Box(\neg\mathtt{g}_1 \rightarrow \mathtt{r}_1)$.

This proof may appear odd at first glance, since reset is guaranteed by the operation of the client—the arbiter is not required to do anything. But it is the robustness of the arbiter that is shown: as soon as the client lowers its request, it may assume that the grant has been lowered and make another request. Even with the shortest of negative pulses on a request input, the arbiter will not malfunction.

5.4 Dominance

We show $\Box(\mathtt{g}_1 \rightarrow \neg\mathtt{r}_1)$. The argument for $\mathtt{r}_2$ and $\mathtt{g}_2$ is symmetric. By the definition of the abstraction function, the requirement is equivalent to

$$\Box((a > 0.89) \wedge (x \geq 0.8)) \quad \rightarrow \quad ((a \leq -0.9) \vee ((a \leq 0.9) \wedge (\dot{a} < 0)))$$

From the definition of $\dot{x}$:

$$(a > 0.89) \wedge (x = 0.8) \wedge (x+y < 0) \quad \Rightarrow \quad \dot{x} > 0.098$$

and so trajectories with $x = 0.8$ cannot enter $x < 0.8$ when $a > 0.89$, which establishes $\Box((a > 0.89) \wedge (x \geq 0.8)) \rightarrow (a \leq 0.89)$. By the monotonicity requirements on the client,

$$(a \leq 0.89) \Rightarrow ((a \leq -0.9) \vee ((a \leq 0.9) \wedge (\dot{a} < 0)))$$

which completes the proof.

5.5 Liveness, almost surely

We show $\Box(\mathtt{r}_1 \leadsto ((\mathtt{r}_1 \wedge \mathtt{g}_1) \vee (\mathtt{r}_2 \wedge \mathtt{g}_2)))$. The argument for a request at $\mathtt{r}_2$ is equivalent. Note that the conditions for the clients ensure $\Box(\mathtt{r}_1 \leadsto (a > 0.9))$; so it is sufficient to consider the case that $(a > 0.9)$. The proof proceeds by case analysis on b.

If $-0.9 < b \leq 0.9$ and $\dot{b} < 0$, then we say that b is falling. If $\mathtt{g}_1$ is satisfied sometime during b's fall, then the liveness requirement is satisfied. Otherwise, $\mathtt{r}_1$ still holds when $b < -0.9$—the situation covered in the next case.

If $b < -0.9$, then all trajectories head towards an attractor with $a > 0.9$ and $x > 0.8$. There is a non-empty neighborhood of this attractor in which $\mathtt{g}_1$ is satisfied. If $b < -0.9$ is maintained long enough, then the trajectory will enter this neighborhood and satisfy the liveness requirement. Otherwise, $\mathtt{r}_1$ still holds when $(b > -0.9 \wedge \dot{b} > 0)$, which leads to the next case.

If $-0.9 < b \leq 0.9$ and $\dot{b} > 0$, then we say that b is rising. If $\mathtt{g}_1$ is satisfied sometime during b's rise, then the liveness requirement is satisfied. Otherwise, $\mathtt{r}_1$ still holds when $b > 0.9$, which is the final case.

If $b > 0.9$, then both clients are requesting a grant. This is the scenario that can lead to metastable behaviours.

Liveness and metastability

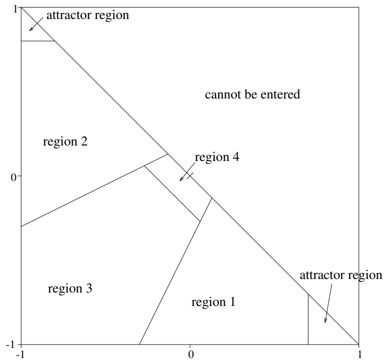

Figure 4: Regions for liveness proof

The remaining proof obligation is $\Box(((a > 0.9) \wedge (b > 0.9)) \leadsto ((\mathtt{r}_1 \wedge \mathtt{g}_1) \vee (\mathtt{r}_2 \wedge \mathtt{g}_2)))$. The proof involves dividing the state space into regions. As was shown in the proof for mutual exclusion, $\Box(x + y < 0)$; therefore, it is only necessary to analyze trajectories in the region below the $x + y = 0$ line. This region is subdivided into four parts, as shown in figure 4. In the proof below, we first show that any trajectory in region 1 will satisfy $\mathtt{r}_1 \wedge \mathtt{g}_1$ after a bounded delay. Likewise, trajectories in region 2 satisfy $\mathtt{r}_2 \wedge \mathtt{g}_2$ with the same delay bound. Next, we show that any trajectory can only remain in region 3 for a bounded amount of time. Region 4—the only remaining region—contains the metastable point. As a final step, we show that trajectories leave this region almost surely and enter region 1 or 2. In the remainder of this section, we will write A_1, A_2, A_3, and A_4 to denote regions 1 through 4 respectively.

First, any trajectory in A_1 satisfies $\mathtt{r}_1 \wedge \mathtt{g}_1$ after a bounded delay. Region 1 is defined by

$$(x + y \leq 0.0) \wedge (-0.4 \geq y - 2x) \wedge (y \geq -1)$$

It is straightforward to show that anywhere in this region $x \geq -0.3$ and $\dot{x} > 0.0176$ (the minimum for $\dot{x}$ occurs at $a = 0.9$, $x = 4/30$, $y = -4/30$). Such a positive $\dot{x}$ guarantees that $x > 0.8$ after at most 63 time units (a very loose upper bound). Therefore, $\mathtt{r}_1 \wedge \mathtt{g}_1$ is satisfied after a bounded delay, as required. A symmetric argument shows that any trajectory in A_2 satisfies $\mathtt{r}_2 \wedge \mathtt{g}_2$ after a bounded delay.

The third step is that any trajectory in A_3 leaves in bounded time. Region 3 is defined by

$$(x + y \leq -0.2) \wedge (-0.4 \leq y - 2x) \wedge (-0.4 \leq x - 2y) \wedge (x \geq -1) \wedge (y \geq -1)$$

It is easy to show that anywhere in this region $\dot{x} + \dot{y} > 0.37$ (all flow is up and to the right), which proves that all trajectories must eventually leave the region. Because $-0.2 - (x + y) \leq 1.8$, a trajectory must leave A_3 after at most 5 time units $(5 > 1.8/0.37)$.

The region that remains to be considered is A_4. All flow on the boundary between regions 3 and 4 is into region 4, and no trajectory can cross into the region where $x + y > 0$. Therefore, if a trajectory leaves region 4, in must enter region 1 or 2. Region 4 contains the saddle point, and the location of this saddle depends on the values of a and b, so the saddle may move slightly even with $a, b \in (0.9, 1.0)$. By the requirements for the clients, either a grant is issued or the clients eventually become α-insensitive; it is sufficient to consider the latter case.

Consider two states, σ_1 and σ_2 in A_4 such that

$$\begin{array}{ll} & \mathit{waiting}(\sigma_1) \wedge \mathit{waiting}(\sigma_2) \\ \wedge & \mathit{ClientsSame}(\sigma_1, \sigma_2) \\ \wedge & (x(\sigma_1) + y(\sigma_1)) = (x(\sigma_2) + y(\sigma_2)) \\ \wedge & (x(\sigma_1) - y(\sigma_1)) \neq (x(\sigma_2) - y(\sigma_2)) \end{array}$$

We show that trajectories emanating from σ_1 and σ_2 must diverge; therefore, at most one of them remains in A_4 indefinitely. This proof demonstrates that the dimension of the set of states whose trajectories remain trapped in A_4 is at most $d-1$. As stated by equation 1, this set of trapped states corresponds to a set of initial states of dimension at most $d-1$, and this initial state set has probability zero by the smoothness requirement for P. The remainder of this section sketches the proof that the trajectories for σ_1 and σ_2 diverge.

For the rest of this analysis, it is convenient to perform a change of variables. Let

$$\begin{array}{rclcrcl} u & = & x - y & \quad & c & = & a + b \\ v & = & x + y & \quad & d & = & a - b \end{array}$$

Rewriting (3) yields the toy's dynamical system in the new coordinates:

$$\begin{array}{rcl} \dot{u} & = & d + uc/2 - vd/2 - u^3/4 - 3uv^2/4 \\ \dot{v} & = & c - 2 - v + ud/2 - vc/2 - v^3/4 - 3vu^2/4 \end{array} \tag{8}$$

Let $\tau \in \mathbf{R}^+$ such that

$$\forall \rho \in [0, \tau].\, \mathit{waiting}(\sigma_1, \tau) \wedge \mathit{waiting}(\sigma_2, \tau)$$

Let $u_1(\rho)$ be the u component of $\varphi^\rho(\sigma_1)$, u_2 be the u component of $\varphi^\rho(\sigma_2)$, and $\Delta u = u_2 - u_1$. Define v_1, v_2, Δv, c_1, c_2, Δc, d_1, d_2, and Δd in the same way.

The proof is based on a double cone in u-v space with the vertex at σ_1 and the axis lying parallel to $v = 0$. Note that the vertex will move as σ_1 moves. At $\rho = 0$, σ_2 will be inside this cone, since $v_2(0) - v_1(0) = \Delta v(0) = 0$. Mathematically, the cone is defined by (where k_v is some positive constant)

$$|\Delta v| \leq |\Delta u| / k_v \tag{9}$$

Showing that σ_1 and σ_2 diverge then requires demonstrating two properties:

- All trajectories in the cone diverge from the vertex in the u direction. In this case, $\Delta \dot{u} > k \Delta u$ for some fixed $k > 0$ in the cone.
- All trajectories that enter the cone stay in the cone. If trajectories are diverging in the u direction already, then $\Delta \dot{v} < -k \Delta v$ (for some other fixed $k > 0$) holding along the edges of the cone will ensure such trapping behaviour.

Without loss of generality, we assume $\Delta u > 0$. We consider the case where $\Delta v > 0$; the other case is equivalent. In order to establish these two properties stated above, we determine a loose lower bound on $\Delta \dot{u}$ in terms of Δu and a loose upper bound on $\Delta \dot{v}$ in terms of Δv. Finding these bounds requires finding a relationship between Δc, Δd and Δu (Δv is taken care of by the definition of the cone). Such a relationship can be established by placing an additional restriction on Δu

$$\Delta u(\tau) \geq k_u \int_0^\tau |\Delta u(\rho)| + |\Delta v(\rho)| \, d\rho \tag{10}$$

and then rewriting the earlier definition of α-insensitive in the new variables

$$|\Delta c(\tau)| + |\Delta d(\tau)| < 2\alpha \int_0^\tau |\Delta u(\rho)| + |\Delta v(\rho)| \, d\rho$$

or

$$\begin{array}{rcl} |\Delta c(\tau)| & < & 2\alpha \int_0^\tau |\Delta u(\rho)| + |\Delta v(\rho)|\, d\rho \\ |\Delta d(\tau)| & < & 2\alpha \int_0^\tau |\Delta u(\rho)| + |\Delta v(\rho)|\, d\rho \end{array} \tag{11}$$

Combining (10) and (11) gives the required relationship:

$$\begin{array}{rcl} |\Delta c| & \leq & 2\alpha|\Delta u|/k_u \\ |\Delta d| & \leq & 2\alpha|\Delta u|/k_u \end{array} \tag{12}$$

That (10) is merely a tightening of an already stated restriction can be seen by differentiating it and using the definition of the cone

$$\begin{array}{rcl} \Delta\dot{u}(\tau) & \geq & k_u(|\Delta u(\tau)| + |\Delta v(\tau)|) \\ & \geq & k_u(1 + 1/k_v)|\Delta u(\tau)| \end{array} \tag{13}$$

Which again requires $\Delta\dot{u}$ to be a positive function of Δu. Intuitively, this equation is providing a lower limit on the rate of exponential divergence of trajectories in the u direction.

With Δv, Δc, and Δd related to Δu, it is possible to find a lower bound in A_4 for $\Delta\dot{u}$ as a function of Δu, α, k_v, and k_u (starting from the rewritten dynamical system (8))

$$\Delta\dot{u} \geq (0.815625 - 2.45\alpha/k_u - 0.1175/k_v)\Delta u$$

For more details on obtaining this equation and those in the rest of the proof, see the appendix. Combining this equation with the requirement given by (13), an inequality on α, k_v, and k_u is found

$$0.815625 - 2.45\alpha/k_u - 0.1175/k_v - k_u - k_u/k_v \geq 0 \tag{14}$$

An assignment to α, k_v, and k_u which satisfies this inequality will guarantee that points within the cone will diverge from the vertex.

In the search for an upper bound on $\Delta\dot{v}$, it is possible to replace the inequality defining the cone (9) with $|\Delta v| = |\Delta u|/k_v$ since we are only interested in the value of $\Delta\dot{v}$ along the edges of the cone. This replacement enables Δc and Δd to be directly related to Δv, which in turn allows an upper bound to be placed on $\Delta\dot{v}$

$$\Delta\dot{v} \leq (-1.9 + 0.1175k_v + 2.45\alpha k_v/k_u)\Delta v$$

Since a trapping cone requires $\Delta\dot{v}$ to be a negative function of Δv, this upper bound becomes the inequality

$$-1.9 + 0.1175k_v + 2.45\alpha k_v/k_u \leq 0 \tag{15}$$

Which, for feasible α, k_v, and k_u, guarantees that trajectories that enter the cone cannot leave.

Any assignment of α, k_u, and k_v which satisfies (14) and (15) will guarantee that σ_2 will remain within the cone and that it will diverge from the vertex (and thus σ_1). Because A_4 has bounded extent, at least one of the diverging σ_1 and σ_2 must escape A_4—completing the proof. An example of a feasible assignment is

$$\alpha = 0.05 \qquad k_u = 0.35 \qquad k_v = 4.06$$

6 Seitz's Arbiter

The techniques used to verify the toy arbiter can be employed to verify a real arbiter circuit. In particular, we consider Seitz's interlock circuit [Sei79]. As mentioned in the introduction, Kurshan and McMillan [KM91] have published a verification of this arbiter. Their verification was based on model checking using a coarsely quantized representation of the phase space and a next state relation obtained by numerical integration. Their analysis could not, however, provide any guarantees that the arbiter would eventually leave the metastable region. Instead, they avoided the issue by adding an assertion that the arbiter escaped the metastability. Since this assumption does not hold for any real arbiter, we find its inclusion in their model unsatisfying.

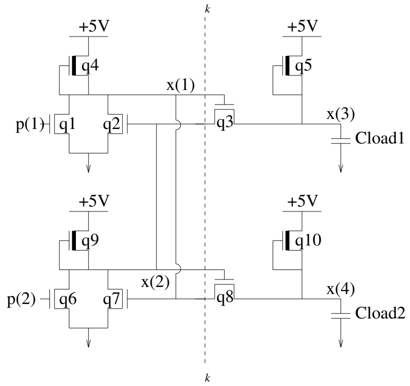

Figure 5: Seitz Interlock Circuit

Figure 5 shows Seitz's arbiter. The six transistors to the left of line kk form a cross-coupled pair of NOR gates. These transistors are the heart of the interlock, and like all arbiters they are subject to metastable states. The remaining four transistors to the right of kk are a "metastability filter." In a circuit model that neglects subthreshold conduction, the outputs of the arbiter, $x(3)$ and $x(4)$, are independent of the internal nodes $x(1)$ and $x(2)$ when the arbiter is near the metastable point. This filter guarantees the insensitivity of the clients that we required for the toy.

We use the same first-order transistor model as Kurshan and McMillan. This model ignores non-linear capacitances, subthreshold conduction and short channel effects such as channel-length modulation and drain-induced threshold lowering. The interlock circuit as a whole uses active low inputs and outputs. Our analysis focuses on the cross-coupled NOR gates which have active low inputs and active high outputs.

hard to print MatLab stuff goes here	hard to print MatLab stuff goes here	hard to print MatLab stuff goes here
$p(1) \in [4,5), p(2) \in [4,5)$	$p(1) \in [4,5), p(2) \in [0,0.6)$	$p(1) \in [0,0.6), p(2) \in [0,0.6)$

Figure 6: Phase space portraits for Seitz's arbiter

Choosing process parameters appropriate to a 2 μm nMOS process [GD85, p. 453] produces the phase portraits shown in figure 6. Like the toy arbiter, there is only one attractor in the system unless both inputs are asserted. In this final case there are two attractors and a single, central saddle point.

The proofs for closure, mutual exclusion, reset, and dominance are equivalent to those presented in section 5. Because the model is more complex, we used numerical optimization algorithms when verifying bounds.

The proof of liveness was actually simpler for the Seitz arbiter than for the toy. This is due to the first order transistor model that ignores subthreshold conduction. When an input is asserted, it is below the enhancement device threshold voltage and the currents into nodes $x(1)$ and $x(2)$ are independent of variations in the inputs. The location of the saddle-point is therefore fixed, and its stable manifold is of dimension $d-1$. Increasing the sophistication of the circuit model to include subthreshold conduction would result in movement of the saddle point with varying inputs. However, this movement would be much smaller in relative terms than for the toy. Therefore, an argument similar to the one in section 5.5 should apply.

7 Conclusions

This paper has presented an approach to verifying arbiters based on dynamical systems theory. Two arbiters were examined: a "toy" arbiter that is amenable to analytical techniques, and Seitz's nMOS design for which numerical methods were employed in the verification. In both cases, safety properties were verified showing that the arbiter follows a four-phase handshake protocol and guarantees mutual exclusion. Liveness was shown in an almost surely sense—the arbiter eventually gives a grant with probability 1, even when the inputs to the arbiter are allowed to vary.

We believe that the dynamical systems approach taken in this paper has several strengths. It provides a common framework for modeling the arbiter and its environment while allowing the details of the environment to be left unspecified. Furthermore, abstracting discrete behaviours from phase space portraits of the arbiters led to simple abstraction mappings, and provided intuition for the proofs.

We see many avenues for further investigation. First, we would like to derive a model for an environment that satisfies the requirements stated in section 4.2. We expect that such a model could lead to a clearer statement of the closure requirement. Second, we would like to investigate the effect of noise on the arbiter. The arguments about diverging traces in section 5.5 suggest an exponential "stretching" process that is relatively insensitive to small variations in the signals. This insensitivity suggests that noise may be inconsequential from a probabilistic perspective. Finally, we would like to apply these techniques to other circuits where metastability is a consideration.

Acknowledgments

We would like to express our appreciation to Michael Mendler, Charles Molnar, and Nick Pippenger for helpful discussions about arbiters and dynamical systems. This paper has also benefited from suggestions made by the referees. This work was supported in part by NSERC Research Grant 138501, a fellowship from the B.C. Advanced Systems Institute, a UBC Graduate Fellowship, and equipment grants from Digital Equipment Corporation and Sun Microsystems.

References

[Cha84] Daniel M. Chapiro. *Globally-Asynchronous, Locally-Synchronous Systems*. PhD thesis, Department of Computer Science, Stanford University, October 1984. Tech. Report STAN-CS-84-1026.

[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.

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

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

[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.

[HW41] Witold Hurewicz and Henry Wallman. *Dimension Theory*. Princeton University Press, 1941.

[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.

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

[Pnu81] Amir Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981.

[RRH93] Anders P. Ravn, H. Rischel, and K.M. Hansen. Specifying and verifying requirements of real-time systems. *IEEE Transactions on Software Engineering*, 19(1):41–55, 1993.

[Sei79] Charles L. Seitz. System timing. In *Introduction to VLSI Systems* (Carver Mead and Lynn Conway), chapter 7, pages 218–262. Addison Wesley, 1979.

A Details of Divergence Proof

In addition to the variables defined earlier, define $u_{avg} = (u_1 + u_2)/2$ and v_{avg}, c_{avg}, d_{avg} similarly. With these definitions, it is possible to write the derivative of the divergence of the two trajectories (starting from (8)):

$$\begin{aligned}
\Delta\dot{u} &= \Delta d + 1/2(c_{avg}\Delta u + u_{avg}\Delta c) - 1/2(d_{avg}\Delta v + v_{avg}\Delta d) \\
&\quad -1/4\Delta u(u_2^2 + u_2u_1 + u_1^2) - 3/4(1/2)(\Delta u(v_2^2 + v_1^2) + 4u_{avg}v_{avg}\Delta v) \\
\Delta\dot{v} &= \Delta c - \Delta v + 1/2(d_{avg}\Delta u + u_{avg}\Delta d) - 1/2(c_{avg}\Delta v + v_{avg}\Delta c) \\
&\quad -1/4\Delta v(v_2^2 + v_2v_1 + v_1^2) - 3/4(1/2)(\Delta v(u_2^2 + u_1^2) + 4u_{avg}v_{avg}\Delta u)
\end{aligned}$$

Since the states under investigation are in A_4 and both are *waiting*, (loose) bounds can be placed on all basic variables involved

$$\begin{array}{rcccl}
-0.3 & \le & u & \le & 0.3 \\
-0.15 & \le & v & \le & 0 \\
1.8 & \le & c & \le & 2.0 \\
-0.1 & \le & d & \le & 0.1
\end{array}$$

The definition of the cone gave a relation (9) between Δv and Δu

$$|\Delta v| \le |\Delta u|/k_v$$

while (12) related Δc and Δd to Δu.

$$\begin{aligned}
|\Delta c| &\le 2\alpha|\Delta u|/k_u \\
|\Delta d| &\le 2\alpha|\Delta u|/k_u
\end{aligned}$$

A minimum for $\Delta\dot{u}$ in terms of Δu can now be found. Note that most terms in this expansion can be chosen positive or negative because of the absolute value signs in the bounds above. In these cases, the sign is chosen so that the term as a whole has the worst (most negative) value. The only terms which cannot change sign are c (always positive), v (always negative), and squares. The latter two values have zero in their range, so that if the term cannot be made negative, the term is zeroed out of the equation.

The (very loose) bound calculated in this way is:

$$\begin{aligned}
\Delta\dot{u} &\ge \Delta d_{min} + 1/2(c_{min}\Delta u + u_{min}\Delta c_{max}) - 1/2(d_{max}\Delta v_{max} + v_{min}\Delta d_{min}) \\
&\quad -1/4\Delta u(3u_{max}^2) - 3/4(\Delta u v_{min}^2 + 2u_{max}v_{min}\Delta v_{min}) \\
&\ge -2\alpha\Delta u/k_u + 1/2((1.8)\Delta u + (-0.3)2\alpha\Delta u/k_u) \\
&\quad -1/2((0.1)\Delta u/k_v + (-0.15)(-2\alpha\Delta u/k_u)) \\
&\quad -3/4\Delta u(0.3)^2 - 3/4(\Delta u(-0.15)^2 + 2(0.3)(-0.15)(-\Delta u/k_v)) \\
&\ge (0.815625 - 2.45\alpha/k_u - 0.1175/k_v)\Delta u
\end{aligned}$$

The differentiated requirement (13) that needed to be satisfied by $\Delta\dot{u}$ is

$$\begin{aligned}
\Delta\dot{u} &\ge k_u(|\Delta u| + |\Delta v|) \\
&\ge k_u(1 + 1/k_v)\Delta u
\end{aligned}$$

Combining these two inequalities yields the final restriction (14) placed on α, k_u, and k_v by $\Delta\dot{u}$

$$\begin{array}{rcl}
0.815625 - 2.45\alpha/k_u - 0.1175/k_v & \ge & k_u(1 + 1/k_v) \\
0.815625 - 2.45\alpha/k_u - 0.1175/k_v - k_u - k_u/k_v & \ge & 0
\end{array}$$

Replacing the definition of the cone by $|\Delta v| = |\Delta u|/k_v$, a loose upper bound on $\Delta \dot{v}$ can be found through similar substitutions. For upper bound calculation, terms are chosen most positive (or zeroed if no positive value is possible)

$$
\begin{array}{rcl}
\Delta \dot{v} & \leq & \Delta c_{max} - \Delta v + 1/2(d_{max}\Delta u_{max} + u_{max}\Delta d_{max}) - 1/2(c_{min}\Delta v + v_{min}\Delta c_{max}) \\
& & -1/4\Delta v(3v_{min}^2) - 3/4(\Delta v u_{min}^2 + 2u_{max}v_{min}\Delta v) \\
& \leq & +2\alpha\Delta v/k_u - \Delta v + 1/2((0.1)k_v\Delta v + (0.3)2\alpha k_v\Delta v/k_u) \\
& & -1/2((1.8)\Delta v + (-0.15)(2\alpha k_v\Delta u/k_u)) \\
& & -0 - 3/4(0 + 2(0.3)(-0.15)(k_v\Delta v)) \\
& \leq & (-1.9 + 0.1175k_v + 2.45\alpha k_v/k_u)\Delta v
\end{array}
$$

Since $\Delta \dot{v}$ need only be a negative function of Δv (we didn't need any more restrictions), the last line is the required inequality (15)

$$-1.9 + 0.1175k_v + 2.45\alpha k_v/k_u \leq 0$$