# A New Explanation of the Glitch Phenomenon [*]

James H. Anderson
Department of Computer Science
The University of Maryland at College Park
College Park, Maryland 20742

Mohamed G. Gouda
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188

August 1988
Revised July 1989

**Abstract**

We consider a discrete model for asynchronous circuits and show that, under very mild restrictions, this model excludes the existence of glitch-free arbiters. This result contradicts a long standing conjecture that the nonexistence of glitch-free arbiters is due to the continuous nature of such circuits.

**Keywords:** arbiter, asynchronous circuit, atomicity, glitch, history, interleaving semantics, metastability, nondeterminism, waiting

**CR Categories:** B.4.3, D.4.1, F.1.1

[*]Work supported in part by Office of Naval Research Contract N00014-89-J-1913.

## 1 Introduction

We consider a fundamental problem in the theory of asynchronous circuits, namely the construction of arbiter circuits; see, for example, Chaney and Molnar [4], Hurtado and Elliot [6], Marino [8], Lamport [7], Black [1], and Udding [13]. Of particular interest in the design of arbiter circuits is the so-called "glitch" phenomenon: An arbiter circuit may take an arbitrarily long time to adjust the values of its outputs in response to changes in the values of its inputs. Furthermore, the values of its outputs may oscillate before stabilizing. The glitch phenomenon usually manifests itself when the asynchronous inputs of an arbiter change value at approximately the same time, thereby forcing the arbiter to "choose" between two or more assignments to its outputs.

It has been shown by Vosbury [14] and by Palais and Lamport [11] that the glitch phenomenon is inherent in arbiter circuits. These proofs are based on a continuous model of asynchronous circuits. In fact, Palais and Lamport state that a proof of this result *must* be based on a continuous model. We show otherwise; that is, we give a proof of the unavoidability of the glitch phenomenon that is based on a discrete circuit model.

The rest of the paper is organized as follows. In the next section, we state the arbiter construction problem and define what it means for an arbiter to be "delay-bounded" and "stable." We illustrate these concepts by presenting in Section 3 an arbiter that is neither delay-bounded nor stable. For an arbiter circuit to be "glitch-free," it is usually taken to mean that it is both delay-bounded and stable. But, as shown in Sections 4 and 5, neither of these conditions can be met. In Section 6, we discuss several issues pertaining to our model of a circuit and our definition of the arbiter construction problem. Concluding remarks appear in Section 7.

## 2 The Arbiter Construction Problem

In this section, we define a general discrete model of a circuit, and state the arbiter construction problem in terms of that model.

**Definition:** A *gate* is a guarded command of the form

$$x \neq E \rightarrow x := E$$

where $x$ is a boolean variable, and $E$ is a boolean expression over a nonempty set of boolean variables. The variables appearing in $E$ are *inputs* of the gate, and variable $x$ is the *output* of the gate. $\Box$

**Definition:** A *wire* is a guarded command of the form

$$x \neq y \rightarrow x := y$$

where $x$ and $y$ are boolean variables. Variable $y$ is the *input* of the wire, and variable $x$ is the *output* of the wire. $\Box$

In the remainder of the paper, we use the term *action* to refer to gates and wires collectively. Next, we define a circuit to be a collection of "connected" actions; two actions are connected when the output of one is an input of the other, and each connection is between a gate and a wire. The definition takes into account the fact that a variable can either be used to connect a gate and a wire, or it can occur as an "external" input or output. We give a brief justification of this definition below.

**Definition:** A *circuit* is a finite collection of gates and wires whose variables are subject to the following restrictions.

- *Bipartition*: The variables can be partitioned into two classes. One class contains all gate inputs and wire outputs, and the other class contains all wire inputs and gate outputs.
- *Integrity*: Each variable is the output of at most one action.
- *Fanout*: Each wire output is the input of at most one gate. Each gate output, however, can be the input of several wires.
- *Feedback*: If the input of a wire is the output of some gate, then the output of the wire is not an input of that same gate.

A variable that is an output of no action is called an *external input*. Similarly, a variable that is an input of no action is called an *external output*. $\Box$

The bipartition restriction states that, with the exception of external inputs and outputs, a circuit is a directed bipartite graph where nodes are gates and wires and where edges are variables; each edge either leads from a gate to a wire or vice versa. The integrity restriction states that, with the exception of external inputs, each variable is the output of a unique action. According to the fanout restriction, the output of a wire is either an external output or an input of exactly one gate. The feedback restriction disallows "loops" that consist of only one gate and one wire. Together, the bipartition and feedback restrictions imply that every loop in a circuit consists of at least two gates and two wires. An example of a circuit is given pictorially in Figure 1.

The restrictions in our definition of a circuit reflect the physical limitations of "actual" circuits. The bipartition restriction reflects the fact that a signal experiences a delay as it propagates from one gate to the next. The integrity restriction rules out the absurd situation in which several gates or wires share a common output. The fanout restriction can be defended on the grounds that a physical wire can be used to deliver a signal to only one gate. The feedback restriction can also be justified based on the behavior of physical circuits; however, we defer the defense of its inclusion to Section 6.2.

**Definition:** A *state* of a circuit is an assignment of values to the variables of the circuit. One state of a circuit is designated as its *initial state*. □

**Definition:** An action is *enabled* in some state iff its guard is true in that state. An enabled action is *executed* by performing its assignment statement. □

**Definition:** Let $s$ be a state of a circuit and let $c$ be an action that is enabled in state $s$. If $s'$ is the result of executing action $c$ at state $s$, then we write $s \xrightarrow{c} s'$. We call $s'$ a *successor* of state $s$. □

**Definition:** A *history* of a circuit is either an infinite sequence $s_0 \xrightarrow{c_0} s_1 \xrightarrow{c_1} \cdots$ or a finite sequence $s_0 \xrightarrow{c_0} s_1 \xrightarrow{c_1} \cdots \xrightarrow{c_{k-1}} s_k$ with no enabled action in the final state $s_k$. A history whose first state is the initial state is called an *initialized history*. □

Our model of a circuit is discrete. Discrete models of asynchronous circuits date back to the work of Miller [10], and have been used more recently by Martin [9] and by Chandy and Misra [3]. A defense of such models is presented later in Section 6.3.

**Arbiter Construction Problem:** We are required to construct a circuit, called an *arbiter*, with two external inputs $p_{in}$ and $q_{in}$ and two external outputs $p_{out}$ and $q_{out}$ such that the following requirements are satisfied.

- In the initial state, $p_{in} = q_{in} = true$ and $p_{out} = q_{out} = false$.
- In the final state of every finite initialized history, $p_{out} \neq q_{out}$.
- There exists a finite initialized history with $p_{out} \wedge \neg q_{out}$ in its final state, and another with $\neg p_{out} \wedge q_{out}$ in its final state. $\square$

Although this definition captures the essence of arbitration, it does not deal with many issues that arise in the design of "practical" arbiters. For example, our definition does not specify how an arbiter should respond to the other three input combinations, i.e., $p_{in} = false$ and $q_{in} = false$, $p_{in} = true$ and $q_{in} = false$, and $p_{in} = false$ and $q_{in} = true$. It also does not specify the conditions under which the value of each input can change. The fact that our statement of the problem is rather weak — i.e., has few requirements — is no accident. By weakening the problem definition, the impossibility results given in Sections 4 and 5 are strengthened.

**Definition:** An arbiter is called *delay-bounded* iff each of its initialized histories is finite. An arbiter is called *stable* iff in each of its initialized histories the value of one external output remains unchanged. An arbiter is called *glitch-free* iff it is both delay-bounded and stable; otherwise it is called *glitch-prone*. $\square$

In the rest of the paper, we show that glitch-prone arbiters can be constructed (Section 3), but glitch-free arbiters cannot. In particular, we show that no arbiter is delay-bounded (Section 4) or stable (Section 5).

## 3 A Glitch-Prone Arbiter

The next theorem shows that a simple R-S flip-flop is a delay-unbounded, unstable arbiter. Such flip-flops are commonly used in the construction of more elaborate arbiter circuits; for example, see the circuits given in [1] and [8].

**Theorem 1:** There are glitch-prone arbiters.

**Proof:** We show that the R-S flip-flop shown in Figure 1 is a delay-unbounded, unstable arbiter. This circuit is defined by the following set of actions (the actions are labeled for convenience).

$$\begin{array}{lllll}
\text{NAND0}: & x \neq \neg(p_{in} \wedge a) & \rightarrow & x := \neg(p_{in} \wedge a) \\
\text{NAND1}: & y \neq \neg(q_{in} \wedge b) & \rightarrow & y := \neg(q_{in} \wedge b) \\
\text{WIRE0}: & p_{out} \neq x & \rightarrow & p_{out} := x \\
\text{WIRE1}: & q_{out} \neq y & \rightarrow & q_{out} := y \\
\text{WIRE2}: & a \neq y & \rightarrow & a := y \\
\text{WIRE3}: & b \neq x & \rightarrow & b := x
\end{array}$$

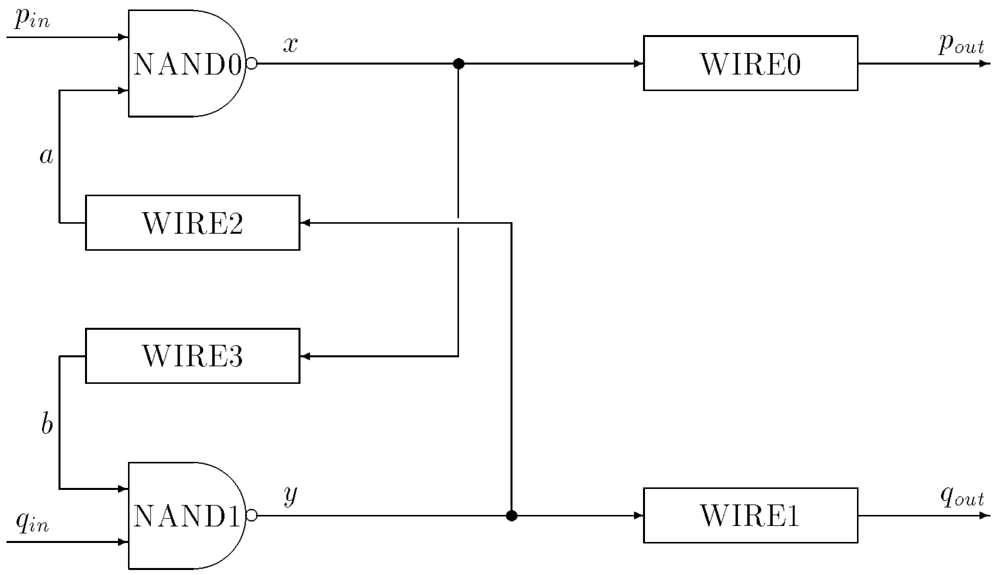

Figure 1: R-S flip-flop.

Initially, let $p_{in} = q_{in} = true$ and $a = b = p_{out} = q_{out} = x = y = false$. We first show that this circuit satisfies the three conditions of the arbiter construction problem, and then show that it is delay-unbounded and unstable. Clearly, this circuit satisfies the first requirement given in the problem statement.

To verify the second requirement, observe that in each final state the guard of each action is false. Therefore, by taking the conjunction of the negation of each guard (and substituting $p_{in} = true$ and $q_{in} = true$) we get the following predicate, which holds in each final state.

$$(x = \neg a) \wedge (y = \neg b) \wedge (p_{out} = x) \wedge (q_{out} = y) \wedge (a = y) \wedge (b = x)$$

This predicate implies that $p_{out} \neq q_{out}$.

To check the third requirement, note that a final state in which $p_{out} = true$ and $q_{out} = false$ is reached by executing the actions in the order NAND0, WIRE3, WIRE0. Similarly, a final state in which $p_{out} = false$ and $q_{out} = true$ is reached by executing the actions in the order NAND1, WIRE2, WIRE1.

An infinite initialized history is obtained by repeatedly executing the following sequence of actions: NAND0, NAND1, WIRE0, WIRE1, WIRE2, WIRE3. Therefore, this arbiter is delay-unbounded. Moreover, in this history the value of each external output changes more than once (in fact an infinite number of times) indicating that the arbiter is unstable. $\Box$

## 4 Nonexistence of Delay-Bounded Arbiters

In this section, we prove that delay-bounded arbiters do not exist. The following definitions are used in the proof. (The first definition is adapted from Fischer, Lynch, and Patterson [5].)

**Definition:** A state of an arbiter is called *p-valent* (or *q-valent*) iff each history starting from that state is finite and ends with a final state in which $p_{out} \wedge \neg q_{out}$ (or $\neg p_{out} \wedge q_{out}$). A state is called *bi-valent* iff it is neither p-valent nor q-valent. $\Box$

Note that each successor of a p-valent (q-valent) state is p-valent (q-valent). Also, observe that if all histories starting from a bi-valent state are finite, then both outcomes $p_{out} \wedge \neg q_{out}$ and $\neg p_{out} \wedge q_{out}$ are reachable from that state; hence the name *bi*-valent.

**Definition:** Two states of an arbiter are *compatible* iff it is not the case that one is p-valent and the other is q-valent. □

Observe that, from the problem statement, the initial state of every arbiter is bi-valent. Therefore, to prove that no delay-bounded arbiter exists, it suffices to show that each bi-valent state of an arbiter has a bi-valent successor. The result is given in Theorem 2 below. The following two lemmas are used in the proof of the theorem.

**Lemma 1:** Let $s \xrightarrow{c} t$ and $s \xrightarrow{c'} t'$, where $c \neq c'$. Then, $c'$ is enabled in state $t$ or $c$ is enabled in state $t'$ (or both).

**Proof:** We prove that if $c'$ is not enabled in state $t$, then $c$ is enabled in state $t'$. By symmetry, this proves the lemma.

Assume that $c'$ is not enabled in state $t$, and let $x$ and $x'$ be the outputs of actions $c$ and $c'$, respectively. By the integrity restriction, $x$ and $x'$ are distinct, i.e., $x'$ is not the output of $c$ and $x$ is not the output of $c'$. Note that states $s$ and $t$ differ only in the value of variable $x$. Thus, because $c'$ is enabled in $s$ but not in $t$, variable $x$ is an input of $c'$ (as established above, $x$ is not the output of $c'$). Because the output of $c$ is an input of $c'$, the bipartition restriction implies that one of the actions is a gate and the other is a wire. Thus, by the feedback restriction, $x'$ is not an input of action $c$. Note that states $s$ and $t'$ differ only in the value of variable $x'$. Therefore, because $x'$ is neither an input nor the output of $c$, the fact that $c$ is enabled in state $s$ implies that it is also enabled in state $t'$. □

**Lemma 2:** All successors of each state are compatible.

**Proof:** The lemma is trivially true for a state with zero or one successor. So, let $s$ be a state with at least two successors. It suffices to prove that if $s \xrightarrow{c} t$ and $s \xrightarrow{c'} t'$, where $c \neq c'$, then $t$ and $t'$ are compatible. By Lemma 1, $c'$ is enabled in $t$ or $c$ is enabled in $t'$. Without loss of generality, assume the former. Then, there exists a state $u$ such that $t \xrightarrow{c'} u$ as depicted in Figure 2(a).

Let action $c$ be of the form $x \neq E \rightarrow x := E$. Note that states $u$ and $t'$ differ only in the value of variable $x$. By the bipartition restriction, $x$ does

$$\begin{tikzcd}
 & s \arrow[dl, "c"'] \arrow[dr, "c'"] & \\
t \arrow[dr, "c'"'] & & t' \\
 & u &
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
 & s \arrow[dl, "c"'] \arrow[dr, "c'"] & \\
t \arrow[dr, "c'"'] & & t' \\
 & u \arrow[ur, "c"'] &
\end{tikzcd}$$

(b)

$$\begin{tikzcd}
 & s \arrow[dl, "c"'] \arrow[dr, "c'"] & \\
t \arrow[dr, "c'"'] & & t' \arrow[dl, "c"] \\
 & u &
\end{tikzcd}$$

(c)

Figure 2: Proof of Lemma 2.

not appear in $E$. Therefore, expression $E$ has the same value in both states $u$ and $t'$. Thus, since all variables are boolean, action $c$ is enabled in exactly one of $u$ and $t'$ and either $u \xrightarrow{c} t'$ or $t' \xrightarrow{c} u$. These two cases are illustrated in Figures 2(b) and 2(c), respectively. In the first case, because $t'$ is reachable from $t$, $t$ and $t'$ are compatible. In the second case, if $u$ is bi-valent, then $t$ and $t'$ are both bi-valent, and hence are compatible. If $u$ is p-valent (q-valent), then neither $t$ nor $t'$ is q-valent (p-valent), and thus $t$ and $t'$ are compatible. $\Box$

**Theorem 2:** No arbiter circuit is delay-bounded.

**Proof:** Recall that the initial state of every arbiter is bi-valent. Therefore, we can prove that every arbiter has an infinite initialized history by proving that every bi-valent state has a bi-valent successor. The fact that a bi-valent state has a successor follows directly from the definition of "bi-valent." By Lemma 2, all successors of a bi-valent state are compatible. Therefore, either all successors of a bi-valent state are p- or bi-valent, or all are q- or bi-valent. Thus, because all successors of a bi-valent state cannot be p-valent (or q-valent), a bi-valent state has at least one successor that is bi-valent. $\Box$

## 5 Nonexistence of Stable Arbiters

We show in this section that it is impossible to construct a stable arbiter, even if delay-unbounded behavior is allowed. We base the proof on the notion of circuit executions, defined next.

**Definition:** A sequence of actions $c_0 \cdot c_1 \cdot \ \cdots \ \cdot c_k$ is an *execution* iff there exist states $s_0, s_1, \ldots, s_{k+1}$ such that $s_0 \xrightarrow{c_0} s_1 \xrightarrow{c_1} \cdots \xrightarrow{c_k} s_{k+1}$ is a prefix of some initialized history. $\Box$

**Notation:** We use $x$, $y$, and $z$ to denote sequences of actions; $c$ and $d$ to denote single actions; $g$ to denote gates; and $w$ to denote wires. Also, we use subscripts and superscripts as necessary. $\Box$

The next two definitions are used in the proof of Theorem 3. The first defines a relation $[c]$ on executions; this definition is adapted from Chandy and Misra [2].

**Definition:** Let $x$ and $y$ be executions and $c$ an action. Then, $x[c]y$ iff execution $x$ yields the same values for the inputs and output of action $c$ as does execution $y$. $\Box$

**Definition:** Action $c$ *reads from* action $d$ iff $c$ has an input that is the output of $d$. $\Box$

The following properties of circuit executions can be derived from the above definitions.

**Property 1:** If $x \cdot c \cdot y$ is an execution, where $y$ does not include $c$, and no action in $y$ reads from $c$, then $x \cdot y$ is also an execution.

**Property 2:** If $x \cdot y$ and $x \cdot c$ are executions, where $y$ does not include $c$, and no action in $y$ reads from $c$, then $x \cdot c \cdot y$ is also an execution.

**Property 3:** If $x[c]y$ then $x \cdot c$ is an execution iff $y \cdot c$ is.

Before proving Theorem 3, we first prove two lemmas. The following definitions are used in the proofs.

**Definition:** A circuit is called *well-formed* iff it has at least one gate and for each gate there exists an execution that includes that gate. □

**Definition:** A gate $g$ of a circuit is called *subordinate* iff for each other gate $g'$ there exists an execution $x \cdot g'$ where $x$ is an execution that does not include $g$. □

**Lemma 3:** Every well-formed circuit has a subordinate gate.

**Proof:** Consider an arbitrary well-formed circuit. Define the relation $\prec$ over the gates of the circuit as follows: for gates $g$ and $g'$, $g \prec g'$ iff in every execution that includes $g'$, $g$ appears before the first appearance of $g'$. It is straightforward to show that $\prec$ is an irreflexive partial order. Because $\prec$ is an irreflexive partial order, and because the set of gates is finite, there exists a gate $g$ such that for each other gate $g'$, it is not the case that $g \prec g'$. In other words, there exists a gate $g$ such that for each other gate $g'$ there exists an execution $x \cdot g'$ where $x$ does not include $g$ — i.e., $g$ is a subordinate gate. □

**Lemma 4:** Every well-formed circuit has an execution in which each gate appears at least once.

**Proof:** Consider an arbitrary well-formed circuit that has $N$ gates. We prove, by induction on $N$, that the circuit has an execution in which each gate appears at least once.

*Base Case*: ($N = 1$) Since the circuit is well-formed, it has an execution in which its only gate appears.

*Induction Step*: ($N > 1$) By Lemma 3, the circuit has a subordinate gate; call it $g$. Let $W$ denote the set of wires that $g$ reads from, and let $C$ denote the circuit obtained from the original circuit by removing gate $g$, along with each wire in $W$. This partitioning is depicted in Figure 3. (External inputs and outputs are not shown in this figure — in particular, note that $g$ may have an external input or output and that a wire in $W$ may have an external input.)

We now show that $C$ is well-formed, i.e., for each gate $g'$ in $C$ there

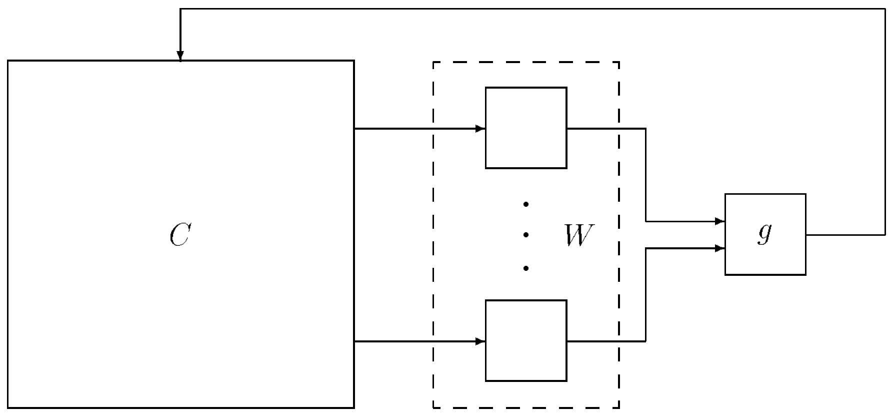

Figure 3: Proof of Lemma 4.

exists an execution of $C$ that includes $g'$. Because $g$ is subordinate, there exists an execution $x{\cdot}g'$ of the original circuit where $x$ does not include $g$. Execution $x$ may contain wires from $W$; therefore, without loss of generality, let $x = x_0{\cdot}w_0{\cdot}x_1{\cdot}w_1{\cdot}\ \cdots\ {\cdot}w_{k-1}{\cdot}x_k$ where each $w_i$ is a wire in $W$ and no $x_j$ contains a wire in $W$. (If $x$ contains no wires from $W$, then $k = 0$.) Due to the bipartition and fanout restrictions, no action in $C$ reads from one of the wires in $W$. Therefore, by repeatedly applying Property 1, first with $w_{k-1}$, then with $w_{k-2}$, etc., we see that $x_0{\cdot}x_1{\cdot}\ \cdots\ {\cdot}x_k{\cdot}g'$ is an execution of the original circuit. Since this execution only contains actions in $C$, it is also an execution of circuit $C$. This establishes that $C$ is well-formed.

Because $C$ is well-formed, by the induction hypothesis, it has an execution in which each of its gates appears at least once. Call this execution $y$. Clearly, $y$ is also an execution of the original circuit. Because the original circuit is well-formed, it has an execution $z{\cdot}g$ where $g$ does not appear in $z$. We will use $y$ and $z$ to construct an execution of the original circuit in which each gate, including $g$, appears at least once. This is accomplished by interleaving each wire from $W$ that appears an odd number of times in $z$ with the actions in $y$.

Let $W'$ denote the set of wires in $W$ that appear an odd number of times in $z$. To see that each wire in $W'$ can be interleaved with the actions of $y$, suppose that we have interleaved some, but not all, of the wires in $W'$ with the actions of $y$ to obtain an execution $y'$. Let $w$ denote a wire in $W'$ that does not appear in $y'$. We show that $w$ can then be interleaved with the actions of $y'$, yielding a new execution $y''$.

If the input of $w$ is external, then because the values of the external inputs do not change, $y'' = y' \cdot w$ is an execution. If, on the other hand, $w$ reads from some gate in $C$, then because this gate appears at least once in $y'$, there exists $y_0$ and $y_1$, where $y' = y_0 \cdot y_1$, such that $y_0 \cdot w$ is an execution. By the bipartition and fanout restrictions, no action in $y_1$ reads from a wire in $W$. Thus, by Property 2, $y'' = y_0 \cdot w \cdot y_1$ is an execution.

If we apply the above "insertion" procedure once for each wire in $W'$, then we obtain an execution $y^*$ where $y^*[g]z$. Now, because $y^*$ contains each action of $y$, it contains each gate in $C$ at least once. Moreover, because $z \cdot g$ is an execution of the original circuit, by Property 3, $y^* \cdot g$ is also an execution of the original circuit. In this execution, each gate appears at least once. □

**Theorem 3:** No arbiter circuit is stable.

**Proof:** Given some arbiter $A$, we first show how to construct another arbiter $A'$ that is stable iff $A$ is. Then, we show that $A'$ cannot be stable.

The arbiter $A'$ is constructed from $A$ in two steps. First, each gate that can never execute (i.e., is included in no execution) is removed from $A$. Second, if $A$ has an external output $y$ that is the output of a wire, then we introduce a gate $x \neq y \;\; \rightarrow \;\; x := y$, where $x$ is not a variable of $A$, and initially $x = y$; thus, $x$ is an external output of $A'$. Such a gate is introduced for each output of $A$ that is the output of a wire (we, of course, require each new external output introduced to be distinct). It is easy to see that $A$ is stable iff $A'$ is.

By construction, arbiter $A'$ is well-formed and each of its external outputs is the output of a gate. Because $A'$ is well-formed, then by Lemma 4, it has an execution in which each gate appears at least once. Therefore, because each of its external outputs is the output of a gate, it has an initialized history in which the value of each external output changes at least once. Hence, it is not stable. □

## 6 Discussion

The generality of a proof concerning a physical phenomenon is dependent on how accurately the corresponding model approximates reality. In this section, we attest to the generality of our results by considering several issues pertaining to our circuit model. In Section 6.1, we consider how signal propagation is modeled by our definition of a circuit. Then, in Section 6.2, we explain the motivation behind the feedback restriction by considering the propagation of a signal along a loop in a circuit. Finally, in Section 6.3, we consider some of the merits and shortcomings of discrete circuit models.

### 6.1 Signal Propagation

The four restrictions (bipartition, integrity, fanout, and feedback) given in our definition of a circuit can be viewed as constraints on the rate of signal propagation. According to the bipartition restriction, if two gates are connected in sequence via a wire, then it takes at least two atomic steps for a signal to propagate from the input of the first gate to the input of the second gate. By the bipartition and integrity restrictions, in order to "merge" the output signals of two gates, it is necessary to connect each gate output to a separate wire, and then connect the output of each wire to a common gate. Thus, it takes three atomic steps for a signal to propagate from the input of one of the two gates to the output of the common gate. The bipartition and fanout restrictions together prohibit a signal from being simultaneously delivered to several gates in one atomic step. The bipartition and feedback restrictions together imply that it takes at least four atomic steps for a signal to traverse a loop in a circuit.

### 6.2 The Feedback Restriction

The first three restrictions in our definition of a circuit reflect the physical limitations of actual circuits. The feedback restriction, however, may seem harder to accept. This restriction has been introduced in order to limit the propagation of a signal along a loop in a circuit. As stated above, the feedback restriction guarantees that each loop consists of at least four atomic steps. Thus, the "delay" experienced by a signal as it travels from an input to the output of a gate (one atomic step) is less than the "delay" experienced by the signal as it traverses the feedback path from the gate's output to its input (at least three atomic steps).

This assumption is not without precedent. In Vosbury's proof of the glitch phenomenon (which is based upon a continuous model of a circuit) each gate is assumed to be "ideal" — i.e., there is no lag time between a change in the value of an input of a gate and the corresponding change in the value of the gate's output. Furthermore, Vosbury assumes that "each closed loop contains at least one non-zero delay" [14]. Therefore, in Vosbury's model, the delay experienced by a signal as it travels from an input to the output of a gate (zero) is less than the delay experienced by the signal as it traverses the feedback path (nonzero). Thus, Vosbury's assumptions achieve the same effect as our feedback restriction.

One may argue that the feedback restriction assigns more delay to a feedback path (three atomic steps) than necessary. After all, if all that is needed is to make the delay of each feedback path larger than that of a single gate, then a delay of two atomic steps for each feedback path will suffice. In fact, this argument is correct. Notice that the feedback restriction is used only in the proof of Lemma 1, and this proof only depends on the fact that each loop consists of at least three actions. Therefore, the feedback restriction can indeed be relaxed to require that each loop has at least three actions. Nonetheless, the bipartition restriction implies that each loop consists of an even number of actions. Thus, we still end up with the magic number of at least four atomic actions per loop.

The feedback restriction is actually not needed to prove the nonexistence of stable arbiters — although this restriction is depicted in Figure 3, it is not necessary for the proof of Theorem 3. The proof that delay-bounded arbiters do not exist is another matter. In fact, as we now show, if the feedback restriction is removed from our definition of a circuit, then a delay-bounded arbiter exists.

Consider the arbiter circuit given in Figure 4. This circuit satisfies the bipartition, integrity, and fanout restrictions, but violates the feedback restriction since the input (output) of WIRE0 is an output (input) of AND. Initially, let $p_{in} = q_{in} = x = z = true$ and $p_{out} = q_{out} = y = false$. We first show that this circuit satisfies the three conditions of the arbiter construction problem, and then show that it is delay-bounded. The first requirement is trivial.

To verify the second requirement, note that the following predicate is true

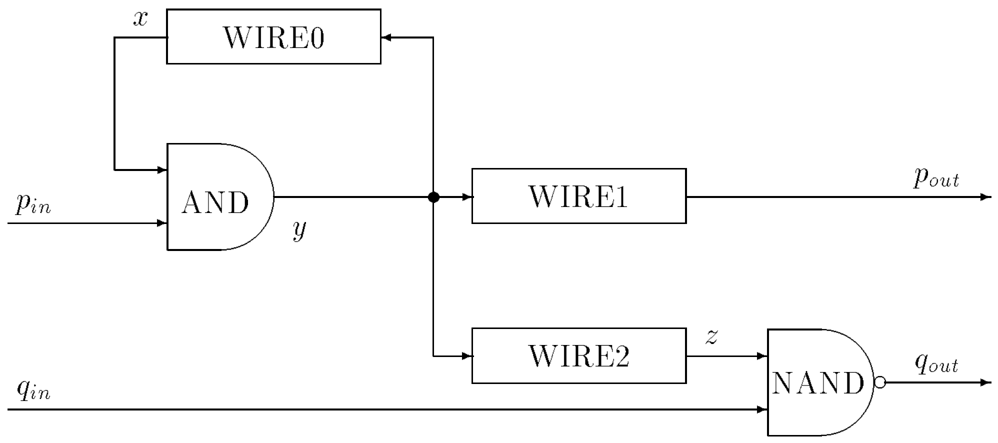

Figure 4: An arbiter that violates the feedback restriction.

in each final state.

$$(x = y) \wedge (z = y) \wedge (p_{out} = y) \wedge (q_{out} = \neg z)$$

This predicate implies that $p_{out} \neq q_{out}$.

To verify the third requirement, observe that a final state in which $p_{out} = \mathit{true}$ and $q_{out} = \mathit{false}$ is reached by executing the actions in the order AND, WIRE1. A final state in which $p_{out} = \mathit{false}$ and $q_{out} = \mathit{true}$ is reached by executing the actions in the order WIRE0, WIRE2, NAND.

To see that this circuit is delay-bounded, note that in every history one of WIRE0 and AND is executed exactly once, and the other is not executed at all. As WIRE0 and AND constitute the only loop of the circuit, this implies that all histories are finite.

## 6.3 Discrete Versus Continuous

Many hardware designers dismiss discrete circuit models because they are inadequate for studying certain aspects of circuit design, e.g. heat dissipation, circuit layout, etc. Nonetheless, the discrete model should not be completely ignored. When dealing with circuits at such an abstract level, powerful verification techniques can be employed. The desired behavior of a circuit can be specified by using a formal notation such as temporal logic. The design of a given circuit can then be formally checked by proving that the required specification is satisfied. Such an approach has been used by Martin, for example, to construct verifiably correct VLSI chips [9].

Some of the perceived inadequacies of the discrete model can be dismissed. The glitch phenomenon is a case in point. While it was previously believed that the unavoidability of the glitch phenomenon is a result of the continuous nature of circuits, we have shown that the discrete model is adequate in this respect, provided the atomicity of a circuit is carefully defined.

## 7 Concluding Remarks

Palais and Lamport state that a proof of the existence of the glitch phenomenon "must be based upon some continuity assumption" concerning asynchronous circuits [11]. We have shown that such a proof can, in fact, be based upon a discrete model of a circuit, provided very mild restrictions are placed on the atomic actions of a circuit.

Our results point out the importance of atomicity when defining a discrete model of a circuit. We have carefully defined the atomicity of a circuit so as to facilitate the proof of the two results that we were after, namely the existence of glitch-prone arbiter circuits and the nonexistence of glitch-free ones. The simplicity of our proofs leads to new insights into the nature of arbiter circuits. In particular, if the atomicity restrictions disallow "tight" loops of one or two actions, then it is impossible for an arbiter to always make a nondeterministic choice in a finite number of steps. Moreover, if each wire output is the input of at most one gate, then oscillating behavior cannot be prevented. Proofs that are based on the more acceptable continuous models seem to obscure these important points.

According to Seitz [12], it is necessary in the design of self-timed circuits to place restrictions on *when* the value of the input of an action can change; such restrictions are called *domain* constraints. For example, a reasonable constraint on the input of a wire is that its value should remain unchanged until the values of the input and output of the wire are equal. We have not attempted to devise a set of domain constraints that an arbiter circuit should satisfy. It would be interesting to see if various constraints render the arbiter construction problem unsolvable using our model of a circuit.

We end the paper with a comment on terminology. After reading an earlier version of this paper, Chuck Seitz informed us that the term "glitch" has many meanings to hardware designers, and suggested that we use the term "indeterminate delay" instead. Although his point is well taken, we decided to keep our original terminology since, by then, the paper had already been widely distributed and had become known by that name. We hope that our terminology has not been confusing.

**Acknowledgements:** We would like to thank Edsger Dijkstra and the members of the Austin Tuesday Afternoon Club (Ken Calvert, Allen Emerson, David Goldschlag, Ted Herman, Charanjit Jutla, Edgar Knapp, Chris Lengauer, Jay Misra, J. R. Rao, Lou Rosier, and Lincoln Wallen) for their comments on this paper. We would also like to thank Nancy Lynch and Alan Fekete for prompting us to include the material in Sections 5 and 6.2. Comments from K. Mani Chandy, Leslie Lamport, Chuck Seitz, Tom Verhoeff, and the members of the Eindhoven VLSI Club led to the inclusion of Section 6.3; we are grateful for their helpful remarks. We are also thankful to Jim Burns for suggesting several improvements in the presentation, and for bringing reference [7] to our attention, and to Nick Pippenger for bringing references [11] and [14] to our attention. Finally, we would like to thank the two anonymous referees for their detailed and insightful comments, and also Anish Arora, Bryan Bayerdorffer, Geoffrey Brown, and Mark Staskauskas for their comments on an earlier draft of the paper.

## References

[1] D. Black, "On the existence of delay-insensitive fair arbiters: trace theory and its limitations," *Distributed Computing*, Vol. 1, No. 4, 1986, pp. 205-225.

[2] K. Chandy and J. Misra, "How processes learn," *Distributed Computing*, Vol. 1, No. 1, 1986, pp. 40-52.

[3] K. Chandy and J. Misra, *Parallel Program Design: A Foundation*, Addison Wesley, 1988, pp. 89-94.

[4] T. Chaney and C. Molnar, "Anomalous behavior of synchronizer and arbiter circuits," *IEEE Transactions on Computers*, Vol. C-22, no. 4, April 1973, pp. 421-422.

[5] M. Fischer, N. Lynch, and M. Patterson, "Impossibility of distributed consensus with one faulty process," *Journal of the ACM*, Vol. 32, No. 2, April 1985, pp. 374-382.

[6] M. Hurtado and D. Elliot, "Ambiguous behavior of logic bistable systems," *Proceedings of the 13th Annual Allerton Conference on Circuit and Systems Theory*, October 1975, pp. 605-611.

[7] L. Lamport, "Buridan's principle," SRI Technical Report, October 1984.

[8] L. Marino, "General theory of metastable operation," *IEEE Transactions on Computers*, Vol. C-30, No. 2, Feb. 1981, pp. 107-115.

[9] A. Martin, "Compiling communicating processes into delay-insensitive VLSI circuits," *Distributed Computing*, Vol. 1, No. 3, 1986, pp. 226-234.

[10] R. Miller, "Speed independent switching circuit theory," in *Switching Theory. Sequential Circuits and Machines*, Vol. 2, Chapter 10. Wiley, New York, 1965.

[11] R. Palais and L. Lamport, "On the glitch phenomenon," Technical Report CA-7611-0811, Massachusetts Computer Associates, Wakefield, Massachusetts, November 1976.

[12] C. Seitz, "System timing," in *Introduction to VLSI Systems*, by C. Mead and L. Conway, Chapter 7. Addison-Wesley, 1980.

[13] J. Udding, "A formal model for defining and classifying delay-insensitive circuits and systems," *Distributed Computing*, Vol. 1, No. 4, 1986, pp. 197-204.

[14] M. Vosbury, *Hazards in Asynchronous Sequential Circuits due to Unrestricted Input Changes*, Ph.D. dissertation, Rensselaer Polytechnic Institute, Troy, New York, December 1973.