True Concurrency in Models of Asynchronous Circuit Behaviour

by

Signe Johanna Silver

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 1999

I hereby declare that I am the sole author of this thesis.

I authorize the University of Waterloo to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the University of Waterloo to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other individuals for the purpose of scholarly research.

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below and give address and date.

Abstract

In the study of asynchronous designs most authors use the interleaving model of concurrency when describing the behaviour of a network, which is usually done for simplicity. The interleaving model assumes the behaviour of an asynchronous circuit can be adequately represented by allowing only one signal to change at a time. In contrast to this, true concurrency models allow an arbitrary number of simultaneous signal changes. It seems that little effort has been made to determine what effect the choice of model may have on the analysis of a network. In this thesis, we attempt to discover the circumstances under which the assumption of single signal changes can be made without affecting the results of circuit analysis. We prove, in a formal network model, that, in the context of delay-insensitivity and semi-modularity, the single change assumption is valid. We also prove that the same is true for a different definition of delay-insensitivity, restricted to deterministic behaviours. Consequently, in these cases, the more complicated true concurrency analysis is not required.

Acknowledgments

First of all, I would like to thank my supervisor, Prof. John Brzozowski, for all his help throughout my stay at Waterloo. He helped me determine a research topic and continued to lead me in the right direction through to the completion of my thesis. I appreciate all his useful feedback, meticulous proof reading and his continued encouragement. I would also like to thank my readers, Peter Buhr and Jo Ebergen, for taking the time to read my thesis carefully and provide me with encouraging and constructive comments. I would like to thank Richard Zhang for providing me with the background work on which my research was based and for taking the time to explain the finer points of his results when needed. I thank Radu Negelescu for examining my counter examples and for his continued concern and friendship. I would also like to thank other members of the Maveric research group: Robert Berks, Piotr Sidorowicz, and Igor Benko, for all their helpful suggestions and criticisms. I thank Michael Mercer for his never-ending patience and support and for making my stay in Waterloo an enjoyable one. Finally, I would like to acknowledge the financial support provided by grant No. OGP0000871 from the Natural Sciences and Engineering Research Council of Canada (NSERC).

Contents

List of Figures

Chapter 1

Introduction

Asynchronous circuits operate without the aid of a global clock. While this gives the potential for higher speed and lower power consumption, it also creates concerns similar to those in other concurrent systems, such as deadlock and safety violations. Much of the work on concurrency theory has been applied to asynchronous circuit theory, even though some assumptions made in concurrent models may not be valid in the case of asynchronous circuits. In particular, many models of concurrent behaviour assume that a system can be represented adequately by allowing only one action to occur at any given time. Most models of asynchronous circuit behaviour also use this assumption, although multiple signal changes sometimes produce different results than single signal changes. It seems that little effort has been made to determine what effect, if any, this assumption might have on the results of the analysis of a circuit. This thesis seeks to shed some light on this question. Our main results relate to delay-insensitivity and semi-modularity, but we also briefly mention such concerns as safety and speed-independence.

Semi-modularity and delay-insensitivity are frequently encountered concepts in asynchronous circuit theory. Semi-modularity was first introduced by Muller [18, 20]. Roughly speaking, it requires that no signal transition can be disabled until it is fired. Varshavsky et al. [30] proved that if a circuit is semi-modular then any state which is reachable from the initial state using multiple changes can also be reached through a sequence of single changes. It follows from their definition that, when checking for semi-modularity, it is sufficient to consider only single changes. Here we study the effects of simultaneous changes on the property of *quasi semi-modularity*, introduced in [5]. Quasi semi-modularity is a generalization of Muller's semi-modularity to nondeterministic components. The second property that we study is delay-insensitivity, which has recently received considerable attention. Roughly speaking, a circuit is delay-insensitive if it operates correctly regardless of the delays in the circuit components and wires. Some of the first work on circuit behaviours in the presence of arbitrary delays can also be traced back to Muller [18, 20]. He described the behaviour of a circuit by sets of allowed sequences that specify the order in which circuit elements can change. If all the allowed sequences for a given circuit with the same initial state end in the same state or a repeating set of states, then the circuit is said to be *speed-independent*. Later, Molnar et al. [19] introduced the "foam-rubber wrapper" postulate to describe delay-insensitive specifications of circuit components. In this point of view, each component is seen as being surrounded by a "foam-rubber wrapper". Signals travel through this wrapper, which assumes no specific form and may change shape over time. Thus, the time it takes for signals traveling to and from circuit components is unknown. Following this idea, J. T. Udding introduced the first formal definition of delay-insensitivity. This definition is based on trace theory and only considered single signal changes. We briefly discuss Udding's definition, which we call the *JTU rules*, in Section 6.1, but we concentrate on two more recent definitions of delay-insensitivity. The first is *strong delay-insensitivity*, introduced by Brzozowski and Zhang [5], which is based on the notion of observational equivalence or bisimulation. This definition uses a model which allows only single signal changes. We also consider a trace-based definition of delay-insensitivity, called *trace preservation*, which compares the traces of a circuit to those of the same circuit with arbitrary delays added.

In this thesis, we explore the effect of multiple signal changes on these definitions. We show that a circuit may be speed-independent under the single-change model but not under the multiple-change model. A circuit is said to be *safe* if no component is ever sent a signal it is not ready to receive. We show that a circuit may be safe in the single-change model but not in the multiple-change model. We also briefly mention possible problems associated with multiple changes and the JTU rules. On a more positive note, our main results are as follows:

- A circuit is quasi semi-modular in the single-change model if and only if it is quasi semi-modular in the multiple-change model.
- A circuit is strongly delay-insensitive in the single-change model if and only if it is strongly delay-insensitive in the multiple-change model.
- For deterministic behaviours, a circuit is trace preserving in the single-change model if and only if it is trace preserving in the multiple-change model. This result does not hold for nondeterministic behaviours.

The thesis is structured as follows. In the remainder of this chapter, we first briefly survey the main types of models of concurrency. Next, we give some examples where multiple signal changes can affect the results of the analysis of a circuit. In Chapter 2, we describe the circuit model and the behavioural model used for most of our results. Chapter 3 deals with semi-modularity. Chapters 4 and 5 deal with the two types of delay-insensitivity we consider. Chapter 6 concludes the thesis.

1.1 Models of Concurrency

The simplest, and probably most common, way of describing a concurrent process is with an *interleaving model*. In this type of model, concurrency is simulated by a nondeterministic choice of the ordering of actions. An interleaving is thus a total order of event occurrences. Each possible sequence of events is taken into account under the assumption that one (or more) of these interleavings represents reality. The main advantage of this approach is the simplicity of the underlying mathematics. Proponents of this approach argue that concurrency is not observable, and hence, it is not relevant in many circumstances [9]. Critics point out that the interleaving approach does not offer a faithful view of reality, since it does not take into account the possibility that actions may occur simultaneously. Nonetheless, this model is still probably the most popular model of concurrency in the study of asynchronous circuits, as well as in concurrency theory in general. Trace theory [7, 8, 28, 29, 31], often used to describe the behaviour of an asynchronous circuit, is a type of interleaving model.

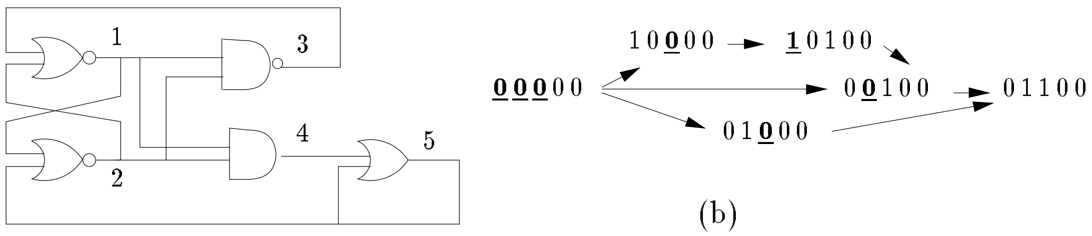
Non-interleaving approaches are usually referred to as *true concurrency* models because they handle concurrency explicitly. One true concurrency approach, which has been quite popular for general concurrency models, is the partial order approach [1, 6, 10, 21, 26]. This model has also been used in asynchronous circuit theory [17, 22, 23, 24]. In the case of partial order models, the behaviour of a concurrent system is represented by a set of possible actions and a partial order representing the necessary temporal precedences among these actions. The partial order is usually expressed by the causality relation "$\rightarrow$", which states that $a \rightarrow b$ if and only if action a must precede action b [21]. In this model, two actions are concurrent if they are independent, i.e., if there is no causal relation between them. There is no necessary relation between concurrency and simultaneity; if two events are independent then they may occur simultaneously or in either order. One of the main advantages of this approach over interleaving is that the difference between independence and nondeterministic interleaving, where two actions are not concurrent but may occur in either order, is easily expressed in partial order semantics. Also, since this model allows us to reason about sets of observations rather than considering every possible sequence of events, it helps to avoid the state explosion problem usually associated with modelling concurrency. If needed, one can easily retrieve the interleavings from a partial order model. Unfortunately, causal partial orders are not expressive enough to satisfactorily model the invariant properties of certain kinds of concurrent systems [15]. For example, a system that includes priority constraints on the ordering of events such as, "if event a has higher priority than event b, then, whenever it is possible to execute both events, b must not be executed." It is not always possible to describe the behaviour of such a system using only causality [2, 14]. In the case of asynchronous circuits, there is no way to represent one event disabling another. Consequently, this model does not address the possibility that there may be events, which may occur simultaneously, but cannot occur one after the other.

Some true concurrency models do express concurrency as simultaneity rather than as independence. One such model is *step sequences* or *step* *traces* [2, 13, 15, 25, 27, 32]. This model is similar to interleaving except that, instead of sequences of actions, it represents an execution as a sequence of sets of actions. Each of these sets is called a *step* and represents a concurrent firing of a set of transitions. In this model, simultaneous actions are easily represented, and there is no need to assume that, because two actions occur simultaneously, they must be able to occur in either order. Thus, this model gives a more realistic representation of the execution of an asynchronous circuit. Rozenberg and Verraedt [27] showed that using step sequences, it is possible to discover differences in behaviours of Petri nets, which cannot be observed using a string language approach. Reisig [25] gave similar examples. This model can also adequately represent systems with priorities, which cannot be represented with interleavings or partial orders [2, 14]. One of the major drawbacks of this approach is that it only makes the state explosion problem worse. In the case of asynchronous circuits, unless the circuit is totally sequential, there are many more step sequences to consider than interleaving sequences. Also, allowing sets of actions at each step can complicate the underlying mathematics.

The first model designed specifically for asynchronous circuits was introduced by Huffman [12] in 1954, and many more have been described since then. These can be divided into interleaving and true concurrency models, depending on whether or not they take simultaneous signal changes into account. We are using the GMW or *general multiple-winner* model to study true concurrency in the analysis of asynchronous circuits. In this model, any number of excited circuit components are allowed to change simultaneously. Such a model was first described by Muller [18, 20], but we use the notation of Brzozowski and Seger [4]. The GMW model is in contrast to the GSW or *general single-winner* model [4], which only allows one component to change at a time. Yakovlev et al. [33] also introduced a model that uses the general single-winner relation (called *single change model* by Yakovlev). As do step sequences, the GMW model provides a more faithful view of reality, but greatly increases the complexity of the analysis. If there are n possible next states to consider from a given state in the GSW model, then there are $2^n - 1$ possible next states to consider when multiple signal changes are taken into account. We use the GSW and GMW models to determine the effect of simultaneous signal changes for the results in Chapters 3–5.

1.2 Problems with Interleaving

In this section, we discuss some of the problems encountered if multiple changes are not taken into account. Problems arise when multiple changes permit the circuit to enter a state that could not have been entered if only single changes are considered. We first look at this problem in the context of *speed-independence*, first introduced by Muller [20]. Basically, a circuit is speed-independent if the outcome of each of its transitions does not depend on the speed of the individual circuit elements. Figure 1.1(a) shows a circuit C_1. Figure 1.1(b) shows a state diagram for this circuit using only single changes, with excited elements underlined. We can see that, starting from state 00000, at most one of the two NOR gates have output 1 at any given time. As a result, the AND gate, and hence also the OR gate, remain stable and the circuit always terminates in state 01100, independently of the delays in the individual circuit elements. However, when multiple changes are included, the two NOR gates may temporarily become high, changing the excitation of the AND gate, and causing the OR gate to stabilize in state 1 (see Figure 1.1(c)); this is impossible under a single-change model. Hence, a circuit may be speed-independent under an interleaving model, but not under a multiple-change model.



(a)

(b)

$$\underline{0}\,\underline{0}\,\underline{0}\,0\,0 \to \underline{1}\,\underline{1}\,0\,\underline{0}\,0 \to \underline{1}\,\underline{1}\,0\,1\,\underline{0} \to \underline{1}\,\underline{1}\,0\,1\,1 \to 1\,0\,\underline{0}\,\underline{1}\,1 \to \underline{1}\,0\,1\,\underline{1}\,1 \to \underline{1}\,0\,1\,0\,1 \to 0\,0\,1\,0\,1$$

(c)

Figure 1.1: (a) Circuit C_1; (b) state diagram with single changes; (c) an execution with multiple changes.

Another concern in the design of asynchronous circuits is *safety*. Roughly speaking, safety ensures that "something bad does not happen" [16]. This means that a circuit element never receives a signal for which it is not ready. Figure 1.2(a) shows a circuit C_2. The first three elements are $\widehat{C}$-elements defined by $C = \bar{a}\bar{b} + (\bar{a} + \bar{b})c$, where a and b are the inputs, c is the current output, and C is the current excitation. A MERGE element receives a signal from one of its input ports, produces an output, and then waits for another signal from an input port. It is not prepared to receive a second input before it has produced an output. This would be a safety violation. Figure 1.2(b) shows the state diagram of the circuit in Figure 1.2(a) under a single-change model. Only one of the AND gates may change, and hence the MERGE element receives only one input. Figure 1.2(c) shows an execution of the circuit taking into account simultaneous changes. In this case, both AND gates may change, and the MERGE element may receive a second input before it has time to process the first. This safety violation cannot be found using a single-change analysis.

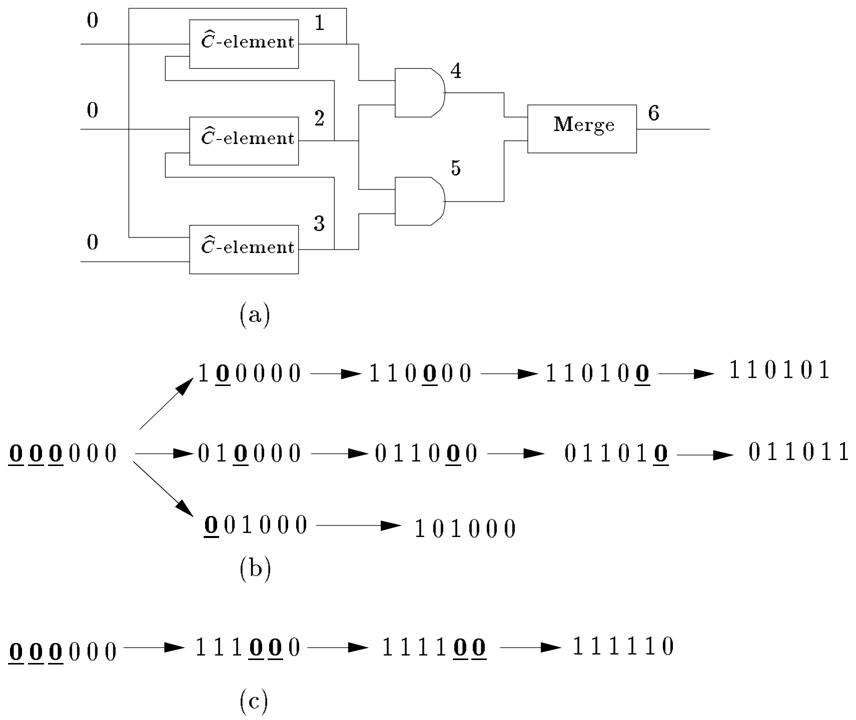


Figure 1.2: (a) Circuit C_2; (b) state diagram with single changes; (c) an execution using multiple changes.

Chapter 2

Network Model

We now define a formal model in which we prove our main results. For a complete description of the model see [5].

2.1 Modules

A network consists of components, called modules, which are nondeterministic sequential machines of the Moore type. To hide the details of the internal design and keep the model simple, we represent a module by an abstract internal state taking its values from a finite set; however, the inputs and outputs remain binary. Delays in the input and output wires of a module are not introduced, allowing us to model isochronic forks and similar components. A module is defined as follows.

Definition 2.1 A *module* M is a sequential machine $M = \langle \mathcal{S}, \mathcal{X}, \mathcal{Y}, \mathcal{Z}, \delta, \lambda \rangle$, where

- $\mathcal{S}$ is a finite set of *internal states* of M;

- $\mathcal{X} = \{x_1, \dots, x_m\}$ is the set of *binary input variables*;
- y is the *internal state variable*;
- $\mathcal{Z} = \{z_1, \dots, z_p\}$ is the set of *binary output variables*;
- δ is the *excitation function*, $\delta : \{0,1\}^m \times \mathcal{S} \to 2^{\mathcal{S}} \setminus \{\emptyset\}$, satisfying the restriction that, for any $a \in \{0,1\}^m$ and $b \in \mathcal{S}$, either $\delta(a,b) = \{b\}$ or $b \notin \delta(a,b)$;
- $\lambda = (\lambda_1, \dots, \lambda_p)$ is the *output function*, $\lambda : \mathcal{S} \to \{0,1\}^p$. □

The excitation function δ takes an input vector and the internal state and returns a set of possible next states. If $\delta(a,b) = \{b\}$, then the module is stable; otherwise the module is excited to change to a new state.

Table 2.1: Functions of a delay module.

y \ x	0	1	$\lambda(y)$
0	{0}	{1}	0
1	{0}	{1}	1

$\delta(x,y)$

Table 2.2: Functions of an arbiter.

$\mathcal{X} = \{x_1, x_2\}$

y	00	01	10	11	$\lambda(y)$
0	{0}	{2}	{1}	{1,2}	00
1	{0}	{0}	{1}	{1}	10
2	{0}	{2}	{0}	{2}	01

$\delta(x,y)$

Example 2.1 Tables 2.1 and 2.2 define the functions of a delay module and an arbiter respectively. A delay module has $\mathcal{S} = \{0,1\}$, $m = 1$, and $p = 1$. An arbiter has $\mathcal{S} = \{0,1,2\}$, $m = 2$, and $p = 2$.

2.2 Networks

We now define a network. This definition is restricted to closed, or autonomous, networks. An open network can be transformed into a closed network by modelling the environment as a module.

Definition 2.2 A *network* is a pair $N = \langle \mathcal{M}, \mathcal{K} \rangle$, where

- $\mathcal{M} = \{M^1, \ldots, M^n\}$, $n \geq 1$, is a set of modules;
- $\mathcal{K}$ is a set of *connections*, each of which is an ordered pair (z_i, x_j), where z_i is the output variable of some module, and x_j is the input variable of some module. Moreover, for each input variable x_j (respectively, for each output variable z_i), there is exactly one output variable z_i (respectively, one input variable x_j) such that $(z_i, x_j) \in \mathcal{K}$.

The state variable of module M^i is y^i and its state set is $\mathcal{S}^i$. The set of *state variables* of a network is $\mathcal{Y} = \{y^1, \ldots, y^n\}$. The set of *states* of a network is $\mathcal{S} = \mathcal{S}^1 \times \ldots \times \mathcal{S}^n$. For $y^i \in \mathcal{Y}$ and $s = (s_1, \ldots, s_n) \in \mathcal{S}$, the *excitation* of y^i in state s is denoted S_i. A state is said to be *stable* if $S_i = \{s_i\}$ for all i; otherwise s is unstable. The set of unstable state variables in state s is denoted by $\mathcal{U}(s) = \{y^i \in \mathcal{Y} \mid S_i \neq \{s_i\}\}$.

Example 2.2 Figure 2.1 shows an example of a network with state variables $\mathcal{Y} = \{y^1, y^2, y^3\}$. Let s be the current state of the network. For the OR gate module M^2, $S_2 = z^1 + z^3$ and $z^2_1 = z^2_2 = y^2$. For the two inverter modules M^1 and M^3, $S_1 = \overline{z^2_1}$, $z^1 = y^1$, $S_3 = \overline{z^2_2}$, and $z^3 = y^3$. After some substitutions we find $S_1 = \overline{y^2}$, $S_2 = y^1 + y^3$, $S_3 = \overline{y^2}$.

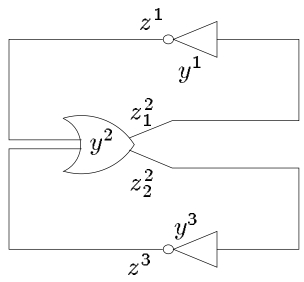


Figure 2.1: Example network N_1.

2.3 Network Behaviours

We now define the behaviour of a network. To allow for simultaneous signal changes, the GSW (general single-winner) relation used in the definition of a behaviour in [5] is replaced with the GMW (general multiple-winner) relation [4]. To avoid confusion, throughout this thesis, we use subscripts s and m to differentiate between GSW and GMW behaviours, respectively.

Definition 2.3 The *GSW relation* over the set of states of a network N is a binary relation $\mathbf{R}_s$, such that $(s,t) \in \mathbf{R}_s$ if and only if s differs from t in exactly one component, say i, $s_i \neq t_i$, $y^i \in \mathcal{U}(s)$, and $t_i \in \mathcal{S}_i$.

Definition 2.4 The *GMW relation* over the set of states of a network N is a binary relation $\mathbf{R}_m$, such that $(s,t) \in \mathbf{R}_m$ if and only if s differs from t in one or more components and for each such component i, we have $s_i \neq t_i$, $y^i \in \mathcal{U}(s)$, and $t_i \in \mathcal{S}_i$.

Definition 2.5 A *GMW behaviour* of a network N is an initialized directed graph $B_m = \langle q_m, \mathcal{Q}_m, \mathcal{R}_m \rangle$, where

- $q_m \in \mathcal{S}$ is the initial state;

- $\mathcal{Q}_m$, specifying the vertices of B_m, is the set of states reachable via the relation $\mathbf{R}_m$, $\mathcal{Q}_m = \{s \in \mathcal{S} \mid (q_m, s) \in \mathbf{R}_m^*\}$;
- $\mathcal{R}_m$, specifying the edges of B_m, is the $\mathbf{R}_m$ relation restricted to $\mathcal{Q}_m$.

If $(s,t) \in \mathcal{R}_m$, we attach to edge (s,t) a *tag* $\tau(s,t) \in 2^{\mathcal{Y}}$, which denotes the set of state variables in which s and t differ. □

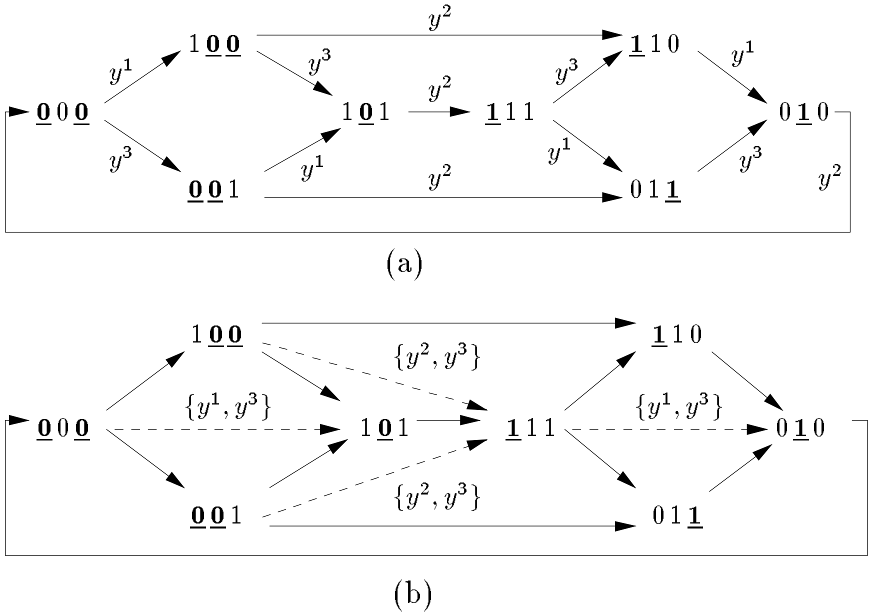


Figure 2.2: (a) A GSW behaviour of N_1; (b) a GMW behaviour of N_1.

We define the GSW behaviour, $B_s = \langle q_s, \mathcal{Q}_s, \mathcal{R}_s \rangle$, of a network N similarly, by replacing $\mathbf{R}_m$ with $\mathbf{R}_s$. Under the GSW model, $\tau(s,t) \in \mathcal{Y}$. Figure 2.2(a) shows the GSW behaviour of network N_1 of Figure 2.1 with initial state 000. Figure 2.2(b) shows the additional transitions possible under a GMW model. If state t is reachable from state s under the GSW model

($(s,t) \in \mathcal{R}_s^*$) through a sequence $w \in \mathcal{Y}^*$ of state changes, then we use the notation $t \in sw$; under the GMW model we write $t \in sW$ where $W \in (2^{\mathcal{Y}})^*$. If w consists of only one element y^i, then $t \in sy^i$; if W consists of only one set $\mathcal{T}$, then $t \in s\mathcal{T}$. It is easy to see that $\mathcal{R}_s$ is a subset of $\mathcal{R}_m$, and therefore $\mathcal{Q}_s$ is a subset of $\mathcal{Q}_m$.

It is shown in [5] that a behaviour $B = \langle q, \mathcal{Q}, \mathcal{R} \rangle$ of a network can also be represented as a nondeterministic finite automaton $\mathcal{B}$ where every state is an accepting state and the language $L(\mathcal{B})$ accepted by $\mathcal{B}$ is defined in the usual way.

Chapter 3

Quasi Semi-Modularity

Semi-modularity was first defined in [20] for deterministic behaviours, which include multiple signal changes. A generalization, called quasi semi-modularity, was introduced in [5] to allow nondeterminism, although the behaviours considered are restricted to those including only single signal changes. The name semi-modularity comes from the fact that a certain partially ordered set derived from the states of a semi-modular circuit forms a semi-modular lattice [20]. No such correspondence has been found for circuits exhibiting nondeterminism, hence the name quasi semi-modularity. When only deterministic circuits are considered, the two definitions are equivalent. Quasi semi-modularity ensures that, once a variable becomes excited to change to a certain value, that excitation cannot be destroyed until the variable changes. For example, the behaviour of the the circuit in Figure 1.1 is not semi-modular because, from the initial state, either of the NOR gates may be disabled without changing. We now modify the definition of quasi semi-modularity found in [5] to include simultaneous signal changes.

Definition 3.1 Let $B_m = \langle q_m, \mathcal{Q}_m, \mathcal{R}_m \rangle$ be a (GMW) behaviour of a network N. A state $s \in \mathcal{Q}_m$ is said to be *multiple-change quasi semi-modular* if, for all states $t \in \mathcal{Q}_m$ and $\mathcal{T} \in 2^{\mathcal{Y}}$, if $t \in s\mathcal{T}$, then the excitation of a module M^j in state s is a subset of its excitation in state t ($S_j \subseteq T_j$) for all j such that $y^j \notin \mathcal{T}$ and $y^j \in \mathcal{U}(s)$. If s is multiple-change quasi semi-modular for all $s \in \mathcal{Q}_m$, then B_m is *multiple-change quasi semi-modular.*

The type of semi-modularity introduced in [5], which we call *single-change quasi semi-modularity*, is a special case of this definition where only one element changes in going from state s to state t, i.e., $\mathcal{T}$ is a singleton set.

Let N be a network. Let $B_s = \langle q_s, \mathcal{Q}_s, \mathcal{R}_s \rangle$ be a GSW behaviour of N and let $B_m = \langle q_m, \mathcal{Q}_m, \mathcal{R}_m \rangle$ be the corresponding GMW behaviour of N such that the initial states are equal ($q_m = q_s = q$). We show that the single-winner behaviour B_s is single-change quasi semi-modular if and only if the multiple-winner behaviour B_m is multiple-change quasi semi-modular. It is proven in [30] that, if a circuit is semi-modular then the set of states reachable from the inital state is the same as the set of states which can be reached from the intial state by single changes. In order to prove the final theorem for this chapter, we first show that this result also holds for quasi semi-modular networks.

Proposition 3.1 *If the behaviour B_s is single-change quasi semi-modular, then $\mathcal{Q}_s = \mathcal{Q}_m$.*

Proof: Since we already know that $\mathcal{Q}_s \subseteq \mathcal{Q}_m$, it remains to be shown that if $s \in \mathcal{Q}_m$, then $s \in \mathcal{Q}_s$. The proof is by induction on the length l of the shortest path from q to s in B_m. If $s = q$, then the result holds trivially. Now suppose that $l = n$. There must be some state r in the path from q to s such that the length of the path from q to r is $n - 1$ and $s \in r\mathcal{T}$ for some $\mathcal{T} \in 2^{\mathcal{Y}}$. Assume that $r \in \mathcal{Q}_s$. Clearly, if $y^i \in \mathcal{T}$, then y^i is excited in r to change to its value in s ($s_i \in R_i$). Since B_s is single-change quasi semi-modular starting from state r, each $y^i \in \mathcal{T}$ can change to its value in s, one at a time, without destroying the excitations of the other variables of $\mathcal{T}$, which have not yet changed. Hence, $s \in rw$, where w consists of all the elements of $\mathcal{T}$ in any order, and $s \in \mathcal{Q}_s$. Therefore, every state reachable from q in B_m is also reachable from q in B_s, and hence $\mathcal{Q}_s = \mathcal{Q}_m$. □

Theorem 3.1 *A GSW behaviour $B_s = \langle q, \mathcal{Q}_s, \mathcal{R}_s \rangle$ of a network N is single-change quasi semi-modular if and only if the GMW behaviour $B_m = \langle q, \mathcal{Q}_m, \mathcal{R}_m \rangle$ is multiple-change quasi semi-modular.*

Proof: First we prove by contradiction that, if behaviour B_s is single-change quasi semi-modular, then B_m must be multiple-change quasi semi-modular. Assume that B_s is single-change quasi semi-modular. Suppose that B_m is not multiple-change quasi semi-modular. There must be some state $s \in \mathcal{Q}_m$ that is not multiple-change quasi semi-modular. Then there exists a state $t \in \mathcal{Q}_m$ such that $t \in s\mathcal{T}$ and there exists a j such that $y^j \notin \mathcal{T}$, $y^j \in \mathcal{U}(s)$, and $S_j \not\subseteq T_j$. By Proposition 3.1, $s \in \mathcal{Q}_s$. Since B_s is single-change quasi semi-modular, starting from s, the values of each $y^i \in \mathcal{T}$ can change, one at a time, without destroying the excitations of the other elements of $\mathcal{T}$ that have not yet changed. Therefore, $t \in sw$, where w consists of all the elements of $\mathcal{T}$ in any order. Since $y^j \notin \mathcal{T}$, y^j does not appear in w. Since B_s is single-change quasi semi-modular and $y^j \in \mathcal{U}(s)$, we know that $S_j \subseteq T_j$ which is a contradiction. Hence, B_m must be multiple-change quasi semi-modular.

To prove the second part of the theorem, we show that, if a state B_m is multiple-change quasi semi-modular, then B_s is single-change quasi semi-modular. Assume that B_m is multiple-change quasi semi-modular. Let s be any state in $\mathcal{Q}_s$. Since $\mathcal{Q}_s \subseteq \mathcal{Q}_m$, we know that $s \in \mathcal{Q}_m$. For all $t \in \mathcal{Q}_s$, if $t \in sy^i$, then $t \in s\mathcal{T}$ with $\mathcal{T} = \{y^i\}$. Since s is multiple-change quasi semi-modular with respect to B_m, $S_j \subseteq T_j$ for all $j \neq i$ such that $y^j \in \mathcal{U}(s)$; hence, s is also single-change quasi semi-modular. Since s could have been any state, B_s is single-change quasi semi-modular. $\square$

Chapter 4

Strong Delay-Insensitivity

Roughly speaking, a network is delay-insensitive if it continues to operate correctly even if the delays in its components and wires change arbitrarily [3]. Strong delay-insensitivity [5] compares the behaviour of a network to that of the same network with added delay elements. This new network is referred to as a *delay extension* of network N, denoted $\hat{N}$, and is obtained from N by inserting some number of delay modules in its connections. Figure 4.1 shows a possible delay extension of network N_1 from Figure 2.1, with a delay added between modules M^2 and M^1. A network is defined to be strongly delay-insensitive if the behaviour of the network $\hat{N}$, with added delays projected out, is observationally equivalent to the original network N. The set of all delay extensions of N is denoted $\mathcal{D}(N)$. Here we first define some of the terminology of [5], and then redefine strong delay-insensitivity to include behaviours with multiple signal changes. We then show that the class of strongly delay-insensitive networks does not change under this less restrictive definition of behaviours.

Definition 4.1 Let $\hat{N} \in \mathcal{D}(N)$. The *projection of a state* $\hat{s}$ of $\hat{N}$ with respect to N is the $|\mathcal{Y}|$-tuple $s = \hat{s} \downarrow \mathcal{Y}$ obtained by removing all components

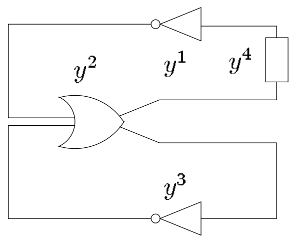


Figure 4.1: Network $\hat{N}_1$, a delay extension of N_1.

corresponding to variables not in $\mathcal{Y}$. The *projection of a word* $\hat{W} \in (2^{\mathcal{Y}})^*$ with respect to N is the word $W = \hat{W} \downarrow \mathcal{Y}$ obtained from $\hat{W}$ by erasing all the symbols not in $\mathcal{Y}$. The projection of a word $w \in \mathcal{Y}^*$ is defined similarly.

A state $\hat{s}$ is said to be an *extension* of s if s is a projection of $\hat{s}$. If $\hat{s}$ is the extension of s where all the inserted delays are stable, then $\hat{s}$ is the *stable-delay extension* of s. We say that a behaviour $\hat{B} = \langle \hat{q}, \hat{\mathcal{Q}}, \hat{\mathcal{R}} \rangle$ of $\hat{N}$ is *initial state compatible* with a behaviour $B = \langle q, \mathcal{Q}, \mathcal{R} \rangle$ of N if $\hat{q}$ is the stable-delay extension of q.

The definition of strong delay-insensitivity depends on the notion of safety, which guarantees that, for any transition in a delay extension of a network, there is an equivalent transition in the original network. Here we define safety for GMW behaviours. The definition for GSW behaviours can be obtained by substituting $\hat{w} \in \mathcal{Y}^*$ in place of $\hat{W}$, and y^i, for some $y^i \in \mathcal{Y}$, in place of $\mathcal{T}$.

Let $\hat{N} \in \mathcal{D}(N)$. Let $B_m = \langle q_m, \mathcal{Q}_m, \mathcal{R}_m \rangle$ be a behaviour of N, and let $\hat{B}_m = \langle \hat{q}_m, \hat{\mathcal{Q}}_m, \hat{\mathcal{R}}_m \rangle$ be the behaviour of $\hat{N}$, which is initial-state compatible with B_m.

Definition 4.2 The behaviour $\hat{B}_m$ is said to be *safe with respect to* B_m if whenever $\hat{s} \in \hat{\mathcal{Q}}_m$ is an extension of $s \in \mathcal{Q}_m$, then for all $\hat{t} \in \hat{\mathcal{Q}}_m$ and

$\hat{W} \in (2^{\hat{\mathcal{Y}}})^*$, if $\hat{t} \in \hat{s}\hat{W}$ and $\hat{W} \downarrow \mathcal{Y} = \mathcal{T}$ for some $\mathcal{T} \in 2^{\mathcal{Y}}$, there exists $t \in \mathcal{Q}_m$ such that $t \in s\mathcal{T}$ and t is the projection of $\hat{t}$ with respect to N.

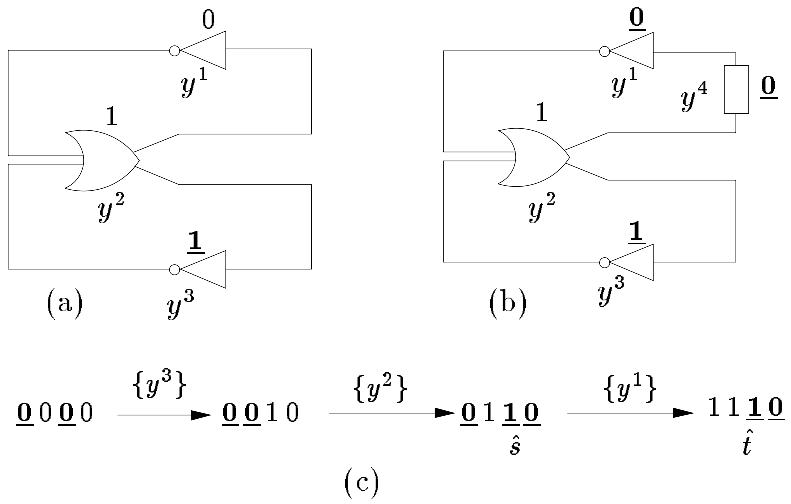


Figure 4.2: (a) Network N_1 in state s; (b) network $\hat{N}_1$ in state $\hat{s}$; (c) an execution of $\hat{N}_1$.

Example 4.1 Consider networks N_1 and $\hat{N}_1$ of Figure 4.2. Let the states of the modules be as shown, with unstable states underlined. Let $s = 011$ be the current state of N_1 and let $\hat{s} = 0110$ be the current state of $\hat{N}_1$. Obviously, $\hat{s}$ is an extension of s. From Figure 2.2, we can see that s is reachable from initial state 000. We can see from the execution in Figure 4.2(c) that $\hat{s}$ is reachable from the initial state 0000, the stable-delay extension of 000. We can also see that $\hat{t} \in \hat{s}\hat{W}$, where $\hat{t} = 1110$ and $\hat{W} = \{y^1\}$, so that $\hat{W} \downarrow \mathcal{Y} = \{y^1\}$. Since y^1 is not excited in state s, there is a violation of safety.

Definition 4.3 A network N is *strongly delay-insensitive with respect to q* if, for any delay extension $\hat{N}$ of N, $\hat{B}$ is safe with respect to B.

In order to prove the final theorem, we need the following auxiliary results.

In [4] it was shown that, if a sufficient number of delays are included in the network, then the GSW model can simulate the GMW model. We expand upon this idea by introducing the notion of a delay-completion which is used to prove the main theorem for this chapter. A *delay-completion* of a network N, denoted $\tilde{N}$, is a delay extension obtained by inserting one delay module in each connection.

Let N be a network and let $\tilde{N}$ be the delay-completion of N. Let $B_m = \langle q_m, \mathcal{Q}_m, \mathcal{R}_m \rangle$ be the GMW behaviour of N and let $\tilde{B}_s = \langle \tilde{q}_s, \tilde{\mathcal{Q}}_s, \tilde{\mathcal{R}}_s \rangle$ be the GSW behaviour of $\tilde{N}$ that is initial-state compatible with B_m. To prove the main result of this chapter, we first show that, for any transition in B_m, there is a series of transitions in $\tilde{B}_s$ that produce the same result.

Proposition 4.1 *If $t \in s\mathcal{T}$ and $\tilde{s} \in \tilde{\mathcal{Q}}_s$, where $\tilde{s}$ is an extension of s, then there exists an extension $\tilde{t}$ of t such that $\tilde{t} \in \tilde{s}\tilde{w}$ for some $\tilde{w}$ consisting of all the elements of $\mathcal{T}$ and possibly some elements of $(\tilde{\mathcal{Y}} - \mathcal{Y})$.*

Proof: By allowing the unstable inserted delays to changes one at a time, starting with those closest to an original module, as suggested in Proposition 4.2 of [5], we get $\tilde{w}_1 \in (\tilde{\mathcal{Y}} - \mathcal{Y})^*$ such that $\tilde{s}\tilde{w}_1 = \tilde{s}'$ where $\tilde{s}'$ is the stable-delay extension of s. If $\tilde{s}'$ is the stable-delay extension of s, then the excitations of all the original modules of N are the same in s and $\tilde{s}'$. This result is proven in Proposition 4.1 in [5]. Hence, $S_i = \tilde{S}'_i$ for all i such that $y^i \in \mathcal{T}$. Since the inputs of every M^i in $\tilde{N}$ such that $y^i \in \mathcal{T}$ are inserted delays, starting from $\tilde{s}'$, the elements of $\mathcal{T}$ can change to their values in t one at a time without affecting the excitation of the other elements of $\mathcal{T}$ that have not yet changed. Thus, $\tilde{t} \in \tilde{s}\tilde{w}$ with $\tilde{w} = \tilde{w}_1\tilde{w}_2$ where $\tilde{w}_2$ consists of all the elements of $\mathcal{T}$. Since $\tilde{t}$ is an extension of t, our claim holds. □

It is easy to see that the result of this proposition can easily be extended to the case where $t \in sW$ for some $W \in (2^{\mathcal{Y}})^*$.

Proposition 4.2 *For every $s \in \mathcal{Q}_m$ there exists at least one $\tilde{s} \in \tilde{\mathcal{Q}}_s$ such that $\tilde{s}$ is an extension of s.*

Proof: The proof is by induction on the length l of the shortest path from q_m to s in B_m. If $q_m = s$, then the result is trivial since $\tilde{q}_s$ is an extension of q_m with respect to N. Now suppose $l = n$. There must be some state r in the path from q_m to s where the length of the path from q_m to r is $n-1$ and $s \in r\mathcal{T}$ for some $\mathcal{T} \in 2^{\mathcal{Y}}$. Assume that there is an $\tilde{r} \in \tilde{\mathcal{Q}}_s$, which is an extension of r with respect to N. By Proposition 4.1, there exists an $\tilde{s} \in \tilde{\mathcal{Q}}_s$ such that $\tilde{s}$ is an extension of s with respect to N. □

We now let $\tilde{\hat{N}}$ be the delay-completion of some $\hat{N} \in \mathcal{D}(N)$ and let $\tilde{\hat{B}}_s = \langle \tilde{\hat{q}}_s, \tilde{\hat{\mathcal{Q}}}_s, \tilde{\hat{\mathcal{R}}}_s \rangle$ be the GSW behaviour of $\tilde{\hat{N}}$. Since $\tilde{\hat{N}}$ is a delay extension of N, if N is strongly delay-insensitive with respect to q under the GSW model, then $\tilde{\hat{B}}_s$ is safe with respect to B_s. Suppose $\hat{t} \in \hat{s}\hat{W}$ in $\hat{B}_m$ and that the projection of $\hat{W}$ is $\mathcal{T}$. Then we know from our propositions that $\tilde{\hat{t}} \in \tilde{\hat{s}}\tilde{\hat{w}}$ in $\tilde{\hat{B}}_s$, where $\tilde{\hat{w}}$ consists of all the elements of $\hat{W}$ and some inserted delays. Since $\tilde{\hat{B}}_s$ is safe with respect to B_s we know that $t \in sw$ in B_s where w consists of the elements of $\mathcal{T}$. Since the first element in w can be any element of $\mathcal{T}$, all the elements of $\mathcal{T}$ must be excited in state s. Thus, $t \in s\mathcal{T}$ in B_m and $\hat{B}_m$ is safe with respect to B_m. This result is the basis for our proof.

Theorem 4.1 *A network is strongly delay-insensitive with respect to a state under the GSW model if and only if it is strongly delay-insensitive with respect to that state under the GMW model.*

Proof: Let N be a network and let $B_s = \langle q_s, \mathcal{Q}_s, \mathcal{R}_s \rangle$ be a GSW behaviour of N. Let $B_m = \langle q_m, \mathcal{Q}_m, \mathcal{R}_m \rangle$ be the GMW behaviour of N with $q_m = q_s$. Let $\hat{N} \in \mathcal{D}(N)$ and let $\hat{B}_m = \langle \hat{q}_m, \hat{\mathcal{Q}}_m, \hat{\mathcal{R}}_m \rangle$ be the GMW behaviour of $\hat{N}$ which is initial state compatible with B_m. First, suppose that N is strongly delay-insensitive with respect to q_s under the GSW model. We shall prove that N is also strongly delay-insensitive, with respect to the same state, under the GMW model by showing that $\hat{B}_m$ is safe with respect to B_m.

Let $\tilde{\hat{N}}$ be the delay-completion of $\hat{N}$. Since $\hat{N}$ is a delay extension of N and $\tilde{\hat{N}}$ is a delay extension of $\hat{N}$, $\tilde{\hat{N}} \in \mathcal{D}(N)$. Let $\tilde{\hat{B}}_s = \langle \tilde{\hat{q}}_s, \tilde{\hat{\mathcal{Q}}}_s, \tilde{\hat{\mathcal{R}}}_s \rangle$ be the GSW behaviour of $\tilde{\hat{N}}$ which is initial state compatible with $\hat{B}_m$. $\tilde{\hat{B}}_s$ must also be initial state compatible with B_s, since $\hat{B}_m$ is initial state compatible with B_m, which has the same initial state as B_s.

Let $\hat{s} \in \hat{\mathcal{Q}}_m$ be an extension of $s \in \mathcal{Q}_m$. Let $\hat{t} \in \hat{\mathcal{Q}}_m$ and $\hat{W} \in (2^{\hat{\mathcal{Y}}})^*$ such that $\hat{t} \in \hat{s}\hat{W}$ and $\hat{W} \downarrow \mathcal{Y} = \mathcal{T}$ for some $\mathcal{T} \in 2^{\mathcal{Y}}$. Let y^i be an arbitrary element of $\mathcal{T}$. To prove that $\hat{B}_m$ is safe with respect to B_m, it is sufficient to show that $\hat{t}_i \in S_i$, i.e., that module M^i is excited in state s to change to its value in state $\hat{t}_i$. Let $\hat{W} = U_1 \hat{\mathcal{T}} U_3$ where $U_1, U_3 \in (2^{\hat{\mathcal{Y}}-\mathcal{Y}})^*$ and $\hat{\mathcal{T}} \downarrow \mathcal{Y} = \mathcal{T}$ and let $\hat{W}' = U_1\{y^i\}$. There must be some state $\hat{r} \in \mathcal{Q}_m$ such that $\hat{r} \in \hat{s}\hat{W}'$ and, since y^i does not appear in U_3, $\hat{r}_i = \hat{t}_i$. Note that y^i is the only original module of N included in $\hat{W}'$.

By Proposition 4.2 we know that there exists an $\tilde{\hat{s}} \in \tilde{\hat{\mathcal{Q}}}_s$, where $\tilde{\hat{s}}$ is an extension of $\hat{s}$ with respect to $\hat{N}$. Hence, by Proposition 4.1, there exists an $\tilde{\hat{r}} \in \tilde{\hat{\mathcal{Q}}}_s$ such that $\tilde{\hat{r}} \in \tilde{\hat{s}}\tilde{\hat{w}}$, where $\hat{r}$ is the projection of $\tilde{\hat{r}}$ with respect to $\hat{N}$ and $\tilde{\hat{w}}$ consists of all the elements of $\hat{W}'$, including y^i, and possibly some elements of $(\tilde{\hat{\mathcal{Y}}} - \hat{\mathcal{Y}})$.

Since N is strongly delay-insensitive under the GSW model and $\tilde{\hat{N}} \in \mathcal{D}(N)$, $\tilde{\hat{B}}_s$ is safe with respect to B_s. Hence, there must exist an $r \in \mathcal{Q}_s$ such that $r \in sw$, where r is the projection of $\tilde{\hat{r}}$ with respect to N and $w = \tilde{\hat{w}} \downarrow \mathcal{Y}$. Since the only element of $\tilde{\hat{w}}$ that corresponds to an original module of N is y^i, $w = y^i$. Hence, $y^i \in \mathcal{U}(s)$ and $r_i \in S_i$. Since $\hat{r}_i = \hat{t}_i$ and r is a projection of $\hat{r}$, $r_i = \hat{r}_i = \hat{t}_i$ and hence, $\hat{t}_i \in S_i$. Since y^i could have been any element of $\mathcal{T}$, $\hat{t}_j \in S_j$ for all j such that $y^j \in \mathcal{T}$. If t is the projection of $\hat{t}$ with respect to N, then $t \in s\mathcal{T}$. Therefore, $\hat{B}_m$ is safe with respect to B_m.

To prove the second half of the theorem, we show that, if $\hat{B}_m$ is safe with respect to B_m, then $\hat{B}_s$ is safe with respect to B_s. Let $s \in \mathcal{Q}_s$ be a projection of $\hat{s} \in \mathcal{Q}_s$. Since $\mathcal{Q}_s \subseteq \mathcal{Q}_m$ and $\hat{\mathcal{Q}}_s \subseteq \hat{\mathcal{Q}}_m$, $s \in \mathcal{Q}_m$ and $\hat{s} \in \hat{\mathcal{Q}}_m$. It is not hard to see that for any $\hat{t} \in \hat{\mathcal{Q}}_s$ and $\hat{w} \in \hat{\mathcal{Y}}^*$ such that $\hat{t} \in \hat{s}\hat{w}$ and $\hat{w} \downarrow \mathcal{Y} = y^i$, $\hat{t} \in \hat{s}\hat{W}$ where $\hat{W}$ consists of all the elements of $\hat{w}$ in singleton sets. Hence, $\hat{W} \downarrow \mathcal{Y} = \{y^i\}$. Since $\hat{B}_m$ is safe with respect to B_m, $t \in s\{y^i\}$, where t is the projection of $\hat{t}$ with respect to N. Hence, y^i is excited in state s and $t \in sy^i$. Therefore, $\hat{B}_s$ is safe with respect to B_s. $\square$

Chapter 5

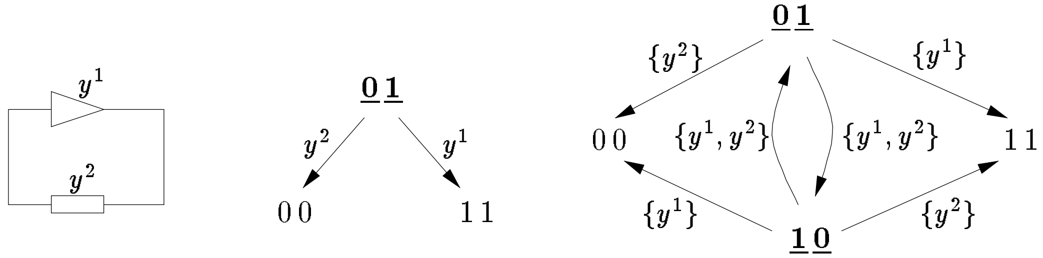
Trace Preservation

Trace theory [7, 8, 28, 29, 31] is a commonly used formalism for asynchronous circuit specification. In terms of trace theory, there are two ways of defining delay-insensitivity. The first, introduced by J.T. Udding [28], is a set of rules which each module in the circuit must obey. The JTU rules insure the absence of ordering between certain signals since there is an unknown delay between the times a signal is sent and received. This definition is discussed in Section 6.1. Another way of checking delay-insensitivity is by adding delays to connections and comparing the behaviour to that of the network without these delays. We refer to this type of delay-insensitivity as trace preservation. If a network is strongly delay-insensitive, then it must be trace preserving, but the opposite is not always true.

A *trace structure* is a pair $T = \langle \mathbf{a}T, \mathbf{t}T \rangle$, where $\mathbf{a}T$ is the *alphabet* of T and $\mathbf{t}T \subseteq (\mathbf{a}T)^*$ is the *trace set* of T. In our application, with the GSW model, the alphabet of a trace structure is the set $\mathcal{Y}$ of internal state variables of a network. In the GMW behaviour of a network, multiple state changes are allowed, and so, our traces consist of sequences of sets of symbols. Hence, $\mathbf{a}T$ is the set of all non-empty subsets of $\mathcal{Y}$. This model is similar to step sequences [2, 13, 15, 27, 32]. Note that we do not assume that, if two events occur simultaneously, then they must also occur in sequence. Recall that $L(\mathcal{B})$ is the language defined by the finite automaton derived from a behaviour B.

Definition 5.1 Let $B_s = \langle q_s, \mathcal{Q}_s, \mathcal{R}_s \rangle$ be a GSW behaviour of network N. The *trace structure induced by* B_s is a pair $T_s = \langle \mathbf{a}T_s, \mathbf{t}T_s \rangle$, where $\mathbf{a}T_s = \mathcal{Y}$ and $\mathbf{t}T_s = L(\mathcal{B}_s)$.

Definition 5.2 Let $B_m = \langle q_m, \mathcal{Q}_m, \mathcal{R}_m \rangle$ be a GMW behaviour of network N. The *trace structure induced by* B_m is a pair $T_m = \langle \mathbf{a}T_m, \mathbf{t}T_m \rangle$, where $\mathbf{a}T_m = 2^{\mathcal{Y}} \setminus \{\emptyset\}$ and $\mathbf{t}T_m = L(\mathcal{B}_m)$.



(a) (b) (c)

Figure 5.1: (a) Network N_3; (b) GSW behaviour of N_3 with initial state 01; (c) GMW behaviour of N_3 with initial state 01.

Example 5.1 The trace set of the GSW behaviour of Figure 5.1(b) is $\{y^1, y^2\}$, while that of the GMW behaviour of Figure 5.1(c) is

$$\{\{y^1\}, \{y^2\}, \{y^1, y^2\}, \{y^1, y^2\}\{y^1\}, \ldots\}.$$

We denote traces of a GMW behaviour by uppercase letters U, V, W, and those of a GSW behaviour, by lowercase letters u, v, w. We say that state s is *reached by* trace w (W) if, starting from the initial state, the network could be in state s after trace w (W) has occurred.

Let N be a network and let $\hat{N}$ be a delay extension of N. Let B_s be the GSW behaviour of N with initial state q. Let $\hat{B}_s$ be the GSW behaviour of $\hat{N}$ which is initial state compatible with B_s. Let T_s and $\hat{T}_s$ be the trace structures of B_s and $\hat{B}_s$, respectively.

Definition 5.3 A network N is said to be *single-change trace preserving* with respect to state q if, whenever $\hat{w} \in \mathbf{t}\hat{T}_s$ there exists a $w \in \mathbf{t}T_s$ such that $\hat{w} \downarrow \mathbf{a}T = w$.

We define a multiple-change trace preserving network similarly.

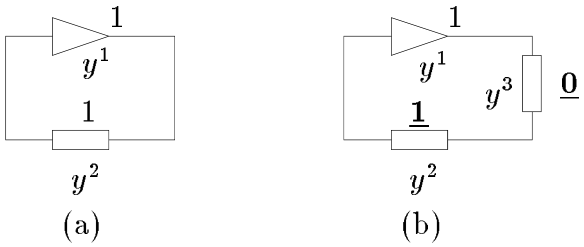


Figure 5.2: (a) Network N_3 in state s; (b) network $\hat{N}_3$, a delay extension of N_3, in state $\hat{s}$.

Example 5.2 Consider networks N_3 and $\hat{N}_3$ of Figure 5.2. Starting from state 01, network N_3 reaches the current state after trace y^1. Starting from state 010, the stable-delay extension of 01, network $\hat{N}_3$ also reaches its current state after trace y^1. We see that y^1y^2 is also a trace of this GSW behaviour of $\hat{N}_3$. This trace is not in the GSW behaviour of N_3; hence N_3 with initial state 01 is not single-change trace preserving.

5.1 Deterministic Behaviours

In the remainder of this chapter, we consider only delay-dense networks, i.e., networks in which in every pair of connected modules at least one is a delay.

First,we restrict our discussion to deterministic behaviours. In this case, the finite automaton $\mathcal{B}$ of a behaviour B is a deterministic finite automaton and the state reached by a trace is unique.

Let N be a delay-dense network whose behaviour is deterministic, and let $\hat{N}$ be a delay extension of N. Let B_s and B_m be the GSW and GMW behaviours, respectively, of N with initial state q. Let $\hat{B}_s$ and $\hat{B}_m$ be GSW and GMW behaviours of $\hat{N}$ which are initial state compatible with B_s and B_m. Let T_s, T_m, $\hat{T}_s$, and $\hat{T}_m$ be the trace structures of B_s, B_m, $\hat{B}_s$, and $\hat{B}_m$ respectively.

Proposition 5.1 *Let $\hat{w} \in \mathbf{t}\hat{T}_s$ and let $\hat{s}$ be the state reached by $\hat{w}$. Let $s \in \mathcal{Q}_s$ be the projection of $\hat{s}$. Then there exists $u \in (\mathbf{a}\hat{T}_s - \mathbf{a}T_s)^*$ such that $\hat{w}u \in \mathbf{t}\hat{T}_s$ and the state reached by $\hat{w}u$ is a stable-delay extension of s.*

Proof: Create u by stabilizing the inserted delays, which are unstable in state $\hat{s}$, one at a time, starting with those delays closest to the original modules, as suggested in Proposition 4.2 of [5]. $\square$

The next proposition is similar to Proposition 4.1. Let $\tilde{N}$ be the delay-completion of N. Let $\tilde{B}_s$ be the GSW behaviour of $\tilde{N}$, which is initial state compatible with B_m, and let $\tilde{T}_s$ be the trace structure of $\tilde{B}_s$.

Let $\mathcal{T} \in 2^{\mathcal{Y}}$ and let $\Pi(\mathcal{T})$ be any string in $\mathcal{T}^*$ in which each element of $\mathcal{T}$ appears exactly once. We call $\Pi(\mathcal{T})$ a *permutation string* of $\mathcal{T}$.

Proposition 5.2 *Let $W \in \mathbf{t}T_m$ where $W = \mathcal{T}_1 \ldots \mathcal{T}_n$. There is a trace $\tilde{w} \in \mathbf{t}\tilde{T}_s$ such that $\tilde{w} = d_1\Pi(\mathcal{T}_1)\ldots d_n\Pi(\mathcal{T}_n)$, where $d_i \in (\tilde{\mathcal{Y}} - \mathcal{Y})^*$. Furthermore, the state $\tilde{s}$ reached by $\tilde{w}$ is an extension of the state s reached by W.*

Proof: We will prove this by induction on the length of the trace W. If $W = \epsilon$, then obviously $\tilde{w} = \epsilon \in \mathbf{t}\tilde{T}_s$ is equal to W and the states reached by W and $\tilde{w}$ are the initial states of the respective networks. Now suppose that $W = U\mathcal{T}_l$, where $U = \mathcal{T}_1 \ldots \mathcal{T}_{l-1} \in \mathbf{t}T_m$ and $\mathcal{T}_l \in \mathbf{a}T_m$. Let r be the state reached by U and s be the state reached by $U\mathcal{T}_l$. Suppose there exists a trace $\tilde{u} \in \mathbf{t}\tilde{T}_s$ such that $\tilde{u} = d_1\Pi(\mathcal{T}_1)\ldots d_{l-1}\Pi(\mathcal{T}_{l-1})$ and state $\tilde{r}$ reached by $\tilde{u}$ is an extension of r. From Proposition 5.1, we know that there exists a $\tilde{v} \in (\mathbf{a}\tilde{T}_s - \mathbf{a}T_s)^*$ such that the state $\tilde{r}'$ reached by $\tilde{u}\tilde{v}$ is a stable-delay extension of r. Let $d_l = \tilde{v}$. The excitation of each module in N, and therefore each module whose internal state variable appears in $\mathcal{T}$, is the same in states $\tilde{r}'$ and r. This result was proven in Proposition 4.1 of [5]. Since the inputs of every module M^i in $\tilde{N}$ such that $y^i \in \mathcal{T}$ are attached to inserted delays, from $\tilde{r}'$, the module variables of $\mathcal{T}$ can change to the same value as in state s, one at a time, in any order, without affecting the excitation of the other elements of $\mathcal{T}$ which have not yet changed. Thus, $\tilde{w} = \tilde{u}d_l\Pi(\mathcal{T}) = d_1\Pi(\mathcal{T}_1)\ldots d_{l-1}\Pi(\mathcal{T}_{l-1})d_l\Pi(\mathcal{T}) \in \mathbf{t}\tilde{T}_s$ for any permutation string $\Pi(\mathcal{T}_l)$ of $\mathcal{T}_l$. Furthermore, since $\tilde{r}$ is an extension of r and each element of $\mathcal{T}_l$ changed to its value in s, state $\tilde{s}$, reached after $\tilde{w}$, is an extension of s.
$\square$

In the following proposition we show that, if we have a number of singleton sets which may appear together in any order in a GMW trace, then including all these elements in one set, i.e., allowing them to change simultaneously, has no effect on the rest of that trace. Let $\Sigma = \{\{y^i\} | \ y^i \in \mathcal{Y}\}$ be the subset of $\mathbf{a}T_m$ consisting of only singleton sets. Let Γ be any subset of Σ, and let V be any string in Γ^* in which each element of Γ appears exactly once. We call V a *permutation string* of Γ. For example, if $\mathcal{Y} = \{y^1, \ldots, y^5\}$ and $\Gamma = \{\{y^1\}, \{y^3\}, \{y^4\}\}$, then $V = \{y^3\}\{y^4\}\{y^1\}$ is a permutation string of Γ.

Proposition 5.3 *If there exist $U_1, U_2 \in (\mathbf{a}T_m)^*$ such that $U_1 V U_2 \in \mathbf{t}T_m$ for each permutation string V of Γ, then $U_1 \mathcal{T} U_2 \in \mathbf{t}T_m$ where $\mathcal{T} = \bigcup_{\{y^i\} \in \Gamma} \{y^i\}$.*

Proof: Let s be the state reached by U_1. Since V can be any permutation string of Γ, all the elements of Γ must be excited in s and hence $U_1 \mathcal{T} \in \mathbf{t}T_m$. To show that $U_1 \mathcal{T} U_2 \in \mathbf{t}T_m$, it is sufficient to show that the state reached by $U_1 \mathcal{T}$ must be the same as the state reached by $U_1 V$ for some permutation string V. Let V be one of the permutation strings in which all the non-delay elements come before any delay elements. Let r be the state reached by UV. Since the network is delay-dense, the input to each non-delay element is a delay, and hence, the excitation of each of these elements must stay the same as in s until it changes. The same is true of all the delay elements in Γ since they are all excited in s, and so, if the excitation changed before the element changed, it would be disabled and V would not be possible from state s. Thus, $r_i = S_i$ for each $\{y^i\} \in \Gamma$ and $r_j = s_j$ for each $\{y^j\} \notin \Gamma$. Obviously, the same is true for the state reached by $U_1 \mathcal{T}$, and hence, $U_1 \mathcal{T} U_2 \in \mathbf{t}T_m$. $\Box$

Theorem 5.1 *A deterministic delay-dense network N is single-change trace preserving with respect to state q if and only if it is multiple-change trace preserving with respect to q.*

Proof: Let $\tilde{\hat{N}}$ be the delay-completion of $\hat{N}$. Let $\tilde{\hat{B}}_s$ be the GSW behaviour of $\tilde{\hat{N}}$ which is initial state compatible with B_m (and hence with $\hat{B}_s$ as well). Let $\tilde{\hat{T}}_s$ be the trace structure of $\tilde{\hat{B}}_s$.

First, suppose that N is single-change trace preserving with respect to state q. To prove that N is multiple-change trace preserving, it must be shown that for any trace $\hat{W} \in \mathbf{t}\hat{T}_m$ there is a trace $W \in \mathbf{t}T_m$, where $\hat{W} \downarrow \mathbf{a}T_m = W$. Let $\hat{W} = \hat{\mathcal{T}}_1 \ldots \hat{\mathcal{T}}_n$ and let $\hat{\mathcal{T}}_i \downarrow \mathcal{Y} = \mathcal{T}_i$ for each i. Then, $W = \mathcal{T}_1 \ldots \mathcal{T}_n$. From Proposition 5.2, we know that for any trace $\hat{W} \in \mathbf{t}\hat{T}_m$ there is a trace $\tilde{\hat{w}} \in \mathbf{t}\tilde{\hat{T}}_s$, where $\tilde{\hat{w}} = d_1\Pi(\hat{\mathcal{T}}_1) \ldots d_n\Pi(\hat{\mathcal{T}}_n)$ and $\Pi(\hat{\mathcal{T}}_i)$ is any permutation string of $\hat{\mathcal{T}}_i$. Since N is single-change trace preserving with respect to state q and $\tilde{\hat{B}}_s$ is initial state compatible with B_s, there must exist a trace $w \in \mathbf{t}T_s$ such that $\tilde{\hat{w}} \downarrow \mathbf{a}T_s = w$. Hence, $w = \Pi(\mathcal{T}_1) \ldots \Pi(\mathcal{T}_n)$. Since each $\Pi(\hat{\mathcal{T}}_i)$ could be any permutation string of $\hat{\mathcal{T}}_i$, each $\Pi(\mathcal{T}_i)$ could be any permutation string of $\mathcal{T}_i$. It is easy to see that for every trace in $w \in \mathbf{t}T_s$ there is a trace $W' \in \mathbf{t}T_m$ with all the elements of w in singleton sets and in the same order as they appear in w. Thus, $W' = \Phi_1 \ldots \Phi_n$, where Φ_i is a string of singleton sets consisting of all the elements of $\Pi(\mathcal{T}_i)$. Since $\Pi(\mathcal{T}_i)$ could be any permutation string of $\mathcal{T}_i$, the singleton sets in Φ_i could appear in any order. It remains to be shown that the elements of each $\mathcal{T}_i$ may occur simultaneously in B_m so that $W \in \mathbf{t}T_m$. Let $U_1 = \epsilon$, $\Gamma = \{\{y^i\} | y^i \in \Pi(\mathcal{T}_1)\}$, and $U_3 = \Phi_2 \ldots \Phi_n$. Each permutation string of Γ is equal to some ordering of Φ_1, and since $\Phi_1 \ldots \Phi_n \in \mathbf{t}T_m$ for any ordering of Φ_1, $U_1 V U_3 \in \mathbf{t}T_m$ for any permutation string V of Γ. Since $\mathcal{T}_1 = \bigcup_{\{y^i\} \in \Gamma} \{y^i\}$, from Proposition 5.3, $U_1 \mathcal{T}_1 U_3 = \mathcal{T}_1 \Phi_2 \ldots \Phi_n \in \mathbf{t}T_m$. We can repeat this procedure by substituting $\mathcal{T}_1$ for Φ_1 in W' and letting $U_1 = \mathcal{T}_1$, $\Gamma = \{\{y^i\} | y^i \in \Pi(\mathcal{T}_2)\}$, and $U_3 = \Phi_3 \ldots \Phi_n$. This procedure can be repeated until $W' = \mathcal{T}_1 \ldots \mathcal{T}_n = W$. Thus, $W \in \mathbf{t}T_m$ such that $\hat{W} \downarrow \mathbf{a}T_m = W$ and N is multiple-change trace equivalent with respect to q.

Now suppose that T_m is trace preserving. Let $\hat{w} \in \mathbf{t}\hat{T}_s$ be a trace. Let $\hat{W} \in \mathbf{t}\hat{T}_m$ consist of all the elements of $\hat{w}$ in singleton sets. Since T_m is trace preserving, there is a trace $W \in \mathbf{t}T_m$ such that $\hat{W} \downarrow \mathbf{a}T = W$. Trace W must also consist of singleton sets, and so we also have a trace $w \in \mathbf{t}T_s$ such that $\hat{w} \downarrow \mathbf{a}T = w$. Therefore, N is also single-change trace preserving with respect to q. $\square$

5.2 Nondeterministic Behaviours

If we allow nondeterminism in the behaviours we are considering, then the state reached by a particular trace may not be unique, and hence, the results of the previous section do not hold. A delay-dense network whose behaviour is nondeterministic may be single-change trace preserving but not multiple-change trace preserving. This situation can happen when there is a trace leading to two different states from which two or more modules can change in any order but not simultaneously. It is possible that, in a delay extension of the network, an extension of that trace may lead to a state in which those modules may change simultaneously. Such a network would not be multiple-change trace preserving but may still be single-change trace preserving. The following example illustrates how this can occur.

Figure 5.3 shows a delay-dense network N_4 with a module M^1 and three delay modules. The functions of M^1 are shown in Table 5.1. Figure 5.4 shows the single-change behaviour of N_4 with initial state 0000. This behaviour is nondeterministic since, from state 0100, module M^1 is excited to change to both states 1 and 2. Notice that after trace y^2y^1 the network may either be in state 1100 or state 2100. The single-change behaviour of N_4 can be described as follows. From the initial state, either y^1 or y^2 may change. If

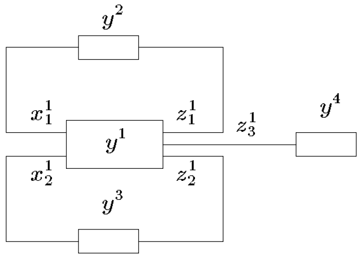


Figure 5.3: Network N_4.

y^1 changes first then each of y^1, y^2, y^3 change once in any order with y^4 changing any time after y^1 changes for the second time. If y^2 changes first, then y^1 changes followed by y^1 and y^3 in any order, with y^4 changing any time after y^1 changes for the second time.

Table 5.1: Functions of module M^1.

	x				
y	00	10	01	11	$\lambda(y)$
0	{3}	{1,2}	{0}	{0}	100
1	{0}	{4}	{0}	{0}	100
2	{0}	{2}	{0}	{4}	110
3	{4}	{4}	{4}	{4}	110
4	{4}	{4}	{4}	{4}	111
	$\delta(x, y)$				

Now suppose there are zero or more delays inserted in each connection of N_4. We denote the delays inserted in the connection from module M^i to module M^j as D_{ij}. The single-change behaviour of the original modules of the network, in any such delay extension, is as follows. Since the inital state of the network is a stable-delay extension of 0000, only y^1 and y^2 are excited in the initial state. If y^1 changes first, then each of y^1, y^2, and the

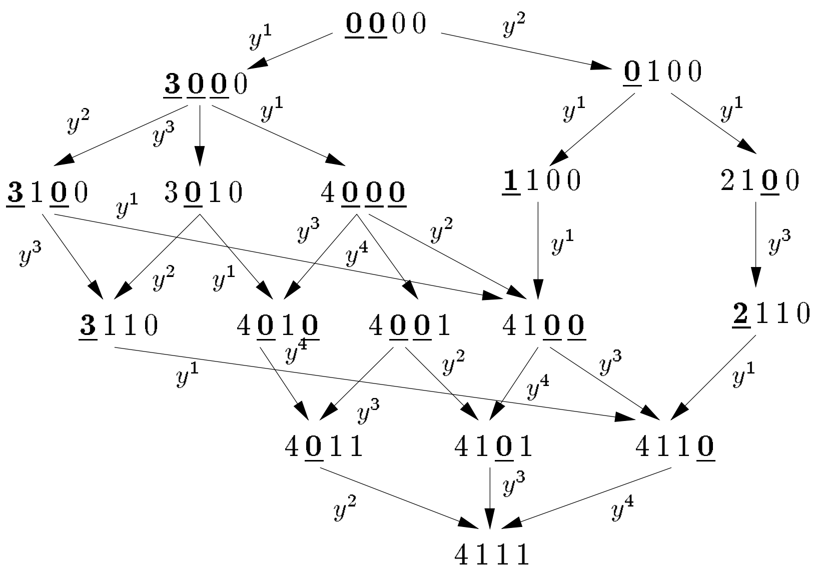
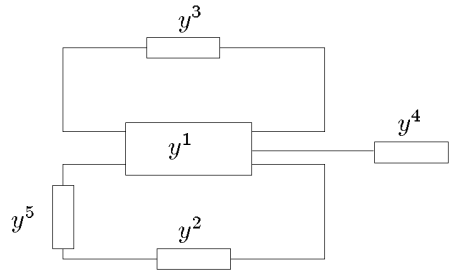


Figure 5.4: State diagram with single changes.

first delay in D_{13} are excited. Once all the delays in D_{13} have changed, y^3 becomes excited. No combination of inserted delays can cause any of these three modules to be disabled or become excited more than once, after the initial change in y^1. Hence, y^1, y^2, and y^3 each change once in any order. Once y^1 has changed for the second time, all the delays in D_{14} may change, causing y^4 to become excited. The delays in D_{14} may change any time after y^1 has changed, and hence, with respect to the original modules, y^4 may change any time after y^1 has changed for the second time.

If y^2 changes first, then y^1 changes, followed by y^1 and y^3 in any order. Again, y^3 does not change until after all the delays in D_{13} have stabilized, which may happen any time after y^1 has changed for the first time. Once, y^1 has changed for the second time, the first delay in D_{14} becomes excited.



(a)

$$\underline{0}\,\underline{0}\,0\,0\,0 \xrightarrow{\{y^2\}} \underline{0}\,1\,0\,\underline{0}\,0 \xrightarrow{\{y^1\}} \underline{\mathbf{3}}\,1\,\underline{0}\,\underline{0}\,0 \xrightarrow{\{y^1, y^3\}} 4\,1\,1\,\underline{0}\,0$$

(b)

Figure 5.5: (a) Network $\hat{N}_4$, a delay extension of N_4; (b) an execution of $\hat{N}_4$ using multiple changes.

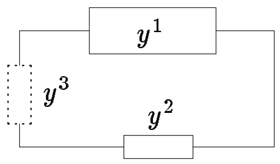
After all the delays in D_{14} have stabilized, y^4 becomes excited. Hence, with respect to the original modules of the network, y^4 may change any time after y^1 has changed for the second time. All of these possibilities can also occur in the behaviour of the original network. Hence, N_4 is single-change trace preserving.

Figure 5.5(a) shows a delay extension, $\hat{N}_4$, of N_4. Figure 5.5(b) shows a possible execution of $\hat{N}_4$, using multiple changes, giving trace $\{y^2\}\{y^1\}\{y^1, y^3\}$. This trace is not possible in the multiple-change behaviour of N_4, since after trace $\{y^2\}\{y^1\}$, the network is either in state 1100 or state 2100 and only one of y^1 or y^3 is unstable. Hence, N_4 is not multiple-change trace preserving with respect to state 0000. Consequently, if nondeterminism is allowed in the behaviour of a network, single-change and multiple-change analysis do not always give the same results in terms of trace preservation.

5.3 Trace Preservation vs. Strong Delay-Insensitivity

Trace preservation is a weaker condition than strong delay-insensitivity . If a network is strongly delay-insensitive with respect to a state, then that network is also trace preserving with respect to that state. If a network is strongly delay-insensitive, then, for any series of state transitions in a delay extension of that network, there is a series of state transitions in the original network, where each state transition y^i in the original network is the projection of the corresponding series of state transitions $\hat{w}$ in the delay extension. Since a trace represents a series of state transitions, for every trace of a delay extension of the network, there must be a trace of the original network which is a projection of it. For delay-dense networks, the same holds true for quasi semi-modularity because a delay-dense network whose behaviour is quasi semi-modular is also strongly delay-insensitive [5].

Conversely, a network may be trace preserving but not strongly delay-insensitive or quasi semi-modular. For example, consider network N_5 shown in Figure 5.6(a) consisting of a module M^1 (the functions of which are shown in Figure 5.6(b)) and a delay. This network is single-change trace preserving with respect to state 00 because, in the single-change behaviour, no matter where delays are added, both modules y^1 and y^2 change state once in either order as in the behaviour of the original network shown in Figure 5.6(c). Since the behaviour is deterministic, N_5 must also be multiple-change trace preserving with respect to state 00. The network is not strongly delay-insensitive with respect to state 00 because, if a delay y^3 is placed between y^2 and y^1, then it is possible to get from an extension of state 01 to an extension of state 11 as shown in Figure 5.6(d). This transition is impossible in the single-change or multiple-change behaviour of the original network. This network is also not quasi semi-modular with respect to state 00 because the excitation of y^1 in state 00 is 1 but after the state of y^2 changes to 1, the excitation of y^1 becomes 2. Thus, for delay-dense networks with deterministic behaviours, trace preservation is the weakest condition studied here for which the single-change assumption is valid. If a network is not delay-dense or its behaviour exhibits nondeterminism, then the behaviour of a network must be quasi semi-modular in order to ensure that single-change analysis is adequate.

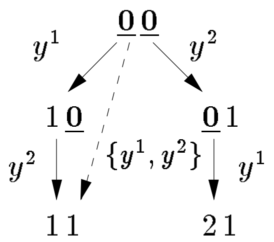


(a)

	x		
y^1	0	1	$\lambda(y)$
0	$\{1\}$	$\{2\}$	1
1	$\{1\}$	$\{1\}$	1
2	$\{2\}$	$\{2\}$	1

$\delta(x, y)$

(b)



(c)

$$\underline{0}\,\underline{0}\,0 \xrightarrow{y^2} \underline{0}\,1\,\underline{0} \xrightarrow{y^1} 1\,1\,\underline{0} \xrightarrow{y^3} 1\,1\,1$$

(d)

Figure 5.6: (a) Network N_5; (b) functions of M^1; (c) single-change behaviour of N_5; (d) an execution of a delay extension of N_5.

Chapter 6

Concluding Remarks

We have shown that the multiple-change model produces the same results as the single-change model for three properties: quasi semi-modularity, strong delay-insensitivity, and, for deterministic behaviours, trace preservation.

As a consequence of our theorems, we have shown that the results of [5] proved in a single-change model also hold in the multiple-change model. In particular, the following more general versions of theorems from [5] hold:

Theorem (6.1 of [5]) *If a network is strongly delay-insensitive with respect to a state under the GMW model, then its GMW behaviour originating from that state is multiple-change quasi semi-modular.*

Theorem (6.2 of [5]) *If a GMW behaviour of a delay-dense network is multiple-change quasi semi-modular, then the network is strongly delay-insensitive under the GMW model with respect to the initial state of that behaviour.*

To conclude this thesis we discuss the effects multiple changes may have on the JTU rules and then list some other open problems.

6.1 The JTU Rules

The first formal definition of delay-insensitivity was made by J.T. Udding [28]. This definition consists of a number of rules, later called the JTU rules, which each module must comply with in order to be delay-insensitive. As with trace preservation, this definition is based on trace theory but, instead of analyzing the trace structure of the entire network, each module has its own trace structure. A trace structure T, in this case, is a triple $T = \langle \mathbf{i}T, \mathbf{o}T, \mathbf{t}T \rangle$, where $\mathbf{i}T$ is called the *input alphabet*, $\mathbf{o}T$ is the *output alphabet*, and $\mathbf{t}T$ is the trace set. The alphabet of T is $\mathbf{a}T = \mathbf{i}T \cup \mathbf{o}T$. The input and output alphabets correspond to the input and output terminals of a module. The trace set represents all possible sequences of communication between a module and its environment.

Udding suggested that a trace structure is delay-insensitive if there is no *computation interference* or *transmission interference*. Computation interference occurs when a signal is sent to a module when the module is not ready to receive it. Transmission interference is an occurrence of two consecutive signals on the same wire. Udding defines a trace structure $T = \langle \mathbf{i}T, \mathbf{o}T, \mathbf{t}T \rangle$ to be delay-insensitive if it obeys the four rules stated below. Two signals a and b are said to be of the same type if they are both inputs or both outputs; otherwise they are of different types. The rules are as follows:

1. There should never be two consecutive signals along the same wire. Formally, for trace u and symbol $a \in \mathbf{a}T$, $uaa \notin \mathbf{t}T$.

2. There should not be any ordering among input signals of a component and the same holds for output signals. Formally, for traces u and v and for symbols a and b of the same type, $uabv \in \mathbf{t}T$ if and only if $ubav \in \mathbf{t}T$.

3. If a and b are signals of different types, and both are enabled after some trace u, then the occurrence of one signal should not disable the other signal. Formally, for trace u and symbol a and b of different types, if $ua \in \mathbf{t}T$ and $ub \in \mathbf{t}T$, then $uab \in \mathbf{t}T$.

4. Suppose signals a and b are of different types, and c is of the same type as a. If a and b can occur in either order at some point, and if c can occur when the order is ab, then it can also occur when the order is ba. Formally, for traces u and v, and for symbols a, b, and c with b a different type than a and c, if $uabvc \in \mathbf{t}T$ then $ubavc \in \mathbf{t}T$.

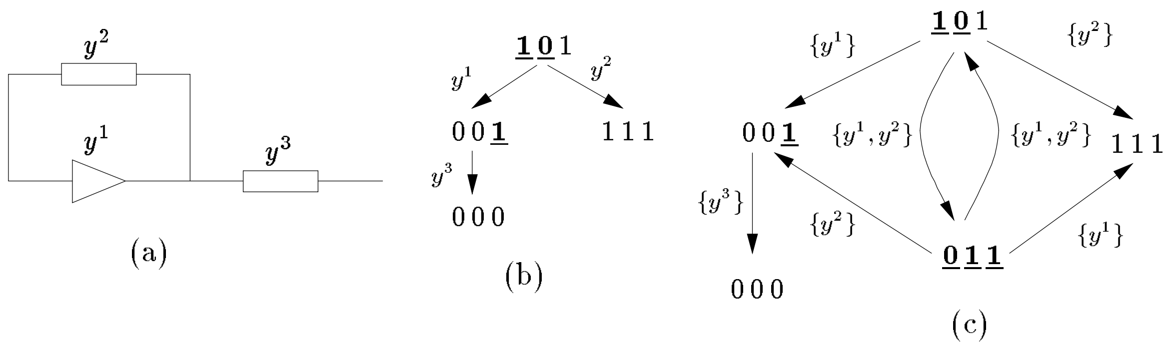


Figure 6.1: (a) Network N_6; (b) single-change behaviour of N_6; (c) partial multiple-change behaviour of N_6.

Udding showed that, if a components specification satisfies these rules, then the component is free of computational and transmission interference. If a trace set is prefix-closed and satisfies these four rules then it is called delay-insensitive. It is unclear however, whether or not a network which satisfies these rules under a single-change model is necessarily free of computational and transmission interference in a multiple-change model. For example, consider the network shown in Figure 6.1(a). From the single-change behaviour shown in Figure 6.1(b) it is easy to see that there is no transmission interference since each module fires only once. On the other hand, in the multiple-change behaviour, y^1 and y^2 can change together any number of times, and y^1 can send multiple signals towards y^3 without any response from y^3. This situation results in transmission interference.

The effect of multiple signal changes on the remaining JTU rules remains an open problem.

6.2 Other Open Problems

1. Since delays can be arbitrary, it seems logical that simultaneous changes should not play a significant role if a network is to be considered delay-insensitive. However, an open question remains what constitutes a "delay-insensitive property"? With trace preservation, we seem to be on the boundary, since allowing multiple changes does not have an effect on the analysis of a deterministic trace preserving network, but when nondeterminism is allowed, multiple-changes become a concern.

2. Is there a general property that would ensure multiple-change analysis does not differ from single-change analysis?

Bibliography

[1] E. Best and R. Devillers. "Interleaving and Partial Orders in Concurrency: A Formal Comparison". In M. Wirsing, editor, *Formal Description of Programming Concepts - III*, pages 299–322. North-Holland, 1987.

[2] E. Best and M. Koutny. "Petri Net Semantics of Priority Systems ". *Theoretical Computer Science*, 96:175–215, 1992.

[3] J. A. Brzozowski. "Delay-Insensitivity and Ternary Simulation". *Theoretical Computer Science*. to appear.

[4] J. A. Brzozowski and C-J. Seger. *Asynchronous Circuits.* Springer-Verlag, New York, NY, 1995.

[5] J. A. Brzozowski and H. Zhang. "Delay-Insensitivity and Semi-Modularity". *Formal Methods in System Design.* to appear. Also Research Report CS-97-11, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, March 1997.

[6] P. Degano, R. De Nicola, and U. Montanari. "A Partial Ordering Semantics for CCS". *Theoretical Computer Science*, 75:223–262, 1990.

[7] D. L. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits.* The MIT Press, Cambridge, MA, 1989.

[8] J. C. Ebergen. *Translating Programs into Delay-Insensitive Circuits.* PhD thesis, Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands, October 1997. Also, CWI Trace 56, Center for Mathematics and Computer Science, Amsterdam, The Netherlands, 1989.

[9] G. L. Ferrari and U. Montanari. "The Observation Algebra of Spatial Pomsets". In *CONCUR '91: 2nd International Conference on Concurrency Theory*, pages 188–202, 1991.

[10] V. Gupta, R. Jagadeensan, and V. Saraswat. "Truly Concurrent Constraint Programming". In *Proceedings of the 7th International Conference on Concurrency Theory*, Pisa, Italy, August 26-29 1996.

[11] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.

[12] D.A. Huffman. "The Synthesis of Sequential Switching Circutis". *IRE Transactions on Electronic Computers*, 257:161–190 and 275–303, 1954.

[13] R. Janicki. "A Formal Semantics of Concurrent Systems with a Priority Relation". *Acta Informatica*, 24(1):33–55, 1987.

[14] R. Janicki and M. Koutny. "Invariant Semantics of Nets with Inhibitor Arcs". In *CONCUR '91: 2nd International Conference on Concurrency Theory*, pages 317–331, 1991.

[15] R. Janicki and M. Koutny. "Invariants and Paradigms of Concurrency Theory". In *Proceedings of Parallel Architectures and Languages Europe*, pages 59–74, 1991.

[16] L. Lamport and N. Lynch. Distributed Computing: Models and Methods. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, Volume B, pages 1159–1196. the MIT Press - Elsevier, 1990.

[17] A. J. Martin. "The Limitations to Delay-Insensitivity in Asynchronous Circuits". In *Advanced Research in VLSI*, pages 263–278, 1990.

[18] R. E. Miller. *Switching Theory, Volume II: Sequential Circuits and Machines.* John Wiley & Sons, 1965.

[19] C. E. Molnar, T. P. Fang, and F. U. Rosenberger. "Synthesis of Delay-Insensitive Modules". In H. Fuchs, editor, *Proceedings of the 1985 Chapel Hill Conference on VLSI*, pages 67–86, Rockville, Maryland, 1985. Computer Science press.

[20] D. E. Muller and W. S. Bartky. "A Theory of Asynchronous Circuits". In *Proceedings of an International Symposium on Switching Theory*, pages 204–243, April 1957.

[21] V. R. Pratt. "Modeling Concurrency with Partial Orders". *Int. J. of Parallel Programming*, 15(1):33–71, 1986.

[22] D. K. Probst and H. F. Li. "Modeling Reactive Hardware Processes Using Partial Orders". *Semantics for Concurrency*, pages 324–343, 1990.

[23] D. K. Probst and H. F. Li. "Using Partial-Order Semantics to Avoid the State Explosion Problem in Asynchronous Systems". *Computer-Aided Verification*, pages 146–155, 1990.

[24] D. K. Probst and H. F. Li. "Partial-Order Model Checking: A Guide for the Perplexed". *Computer-Aided Verification*, pages 322–331, 1991.

[25] W. Reisig. "On Semantics of Petri Nets". *Formal Models in Programming*, pages 347–372, 1985.

[26] W. Reisig. "Temporal Logic and Causality in Concurrent Systems". In *Concurrency 88*, pages 121–139, 1988.

[27] G. Rozenberg and R. Verraedt. "Subset Languages of Petri Nets Part I: The Relationship to String Languages and Normal Forms". *Theoretical Computer Science*, 26:301–326, 1983.

[28] J.T. Udding. *Classification and Composition of Delay-Insensitive Circuits*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 1984.

[29] J. L. A. van de Snepscheut. *Trace Theory and VLSI Design*. Springer-Verlag, New York, NY, 1985.

[30] V. Varshavsky, M. Kishinevsky, V. Marakhovsky, L. Rosenblum, A. Taubin, et al. *Self-timed Control of Concurrent Processes*. Kluwer Academic Publishers, 1990.

[31] T. Verhoeff. *A Theory of Delay-Insensitive Systems*. PhD thesis, Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands, May 1994.

[32] Vogler. "A Generalization of Traces". *Theoretical Informatics and Applications*, 25(2), 1991.

[33] A. Yakovlev, L. Lavagno, and A. Sangiovanni-Vincentelli. “A Unified Signal Transition Graph Model for Asynchronous Control Circuit Synthesis”. *Formal Methods in System Design*, 9:139–188, 1996.