NET-BASED MODELING AND VERIFICATION
OF ASYNCHRONOUS CIRCUITS

by

I. Reicher and M. Yoeli

Technical Report #463

August 1987
NET-BASED MODELING AND VERIFICATION OF ASYNCHRONOUS CIRCUITS

by

Iris Reicher* and Michael Yoeli

Computer Science Department
Technion - Israel Institute of Technology
Haifa 32000, Israel

ABSTRACT

This paper develops a novel approach to the behavioral description, specification and verification of digital circuits operating asynchronously. It combines the theory of trace structures developed by M. Rem and J.L.A. van de Snepscheut with a suitable extension of the theory of labeled Petri nets. The major advantage of the circuit model introduced is related to the derivation of the behavior of a composite circuit from the behavioral descriptions of its components. The approach developed in this paper is to serve as a framework for a method of designing useful asynchronous circuits.

* This research is part of the M.Sc. thesis of the first author
1. INTRODUCTION

This paper develops a novel approach to the behavioral description, specification and verification of digital circuits operating asynchronously.

The elegant theory of communicating processes due to C.A.R. Hoare [Ho85] has been suitably extended and applied to the study of asynchronous VLSI circuits by M. Rem [Re83] and J.L.A. van de Snepscheut [Sn85]. This paper combines the approach of [Re83] and [Sn85] with a suitable extension of the theory of labeled Petri nets and their subset languages [YE83], [RV83]; by doing so, it achieves a considerable simplification of the way, composite circuits are formally modeled and verified. This simplification is related to the fact that Petri nets have the capability of suitably capturing the concept of "true concurrency", whereas most algebraic theories of communicating processes [Ho85], [Sn85], [Mi80] replace true concurrency by "non-deterministic interleaving". The present paper extends the approach developed in [Yo86] where suitable classes of digital circuits were modeled by marked graphs.

Another novel feature of this paper is the way in which the concept 'implementation satisfies specification' is formalized (Section 5). This concept is applied in Section 6, where a number of verification examples are discussed.

We have omitted most proof details, which would require net-theoretical arguments beyond the scope of the present paper. The approach developed in this paper is to serve as a framework for a method of designing useful asynchronous circuits, particularly controllers of self-timed systems (cf. [Yo87], [BKR85]).

2. BEHAVIOR OF DIGITAL CIRCUITS

In this section we introduce the concept of (finite, dynamic) circuit behavior. Initially, we view a digital circuit as a process in the sense of [Ho85] or as a trace structure [Re83], [Sn85]. We then introduce our main concept of (finite, parallel (dynamic) behavior.

We consider a digital circuit $C$ with binary inputs and outputs. We use upper-case letters to denote the inputs and outputs of $C$, and denote by the corresponding lower-case letter, e.g. $x$,
any change of value of $X$. Thus $x$ denotes both the up-transition $X \uparrow$ from $X = 0$ to $X = 1$ as well as the down-transition $X \downarrow$ from $X = 1$ to $X = 0$. We refer to $x$ as an (input or output) event of $C$ and denote by $\text{in} C$ ($\text{out} C$) the set of input (output) events of $C$. We call $\alpha C = \text{in} C \cup \text{out} C$ the event alphabet of $C$.

It is noteworthy that our way of using lower-case letters to denote events is a notational convenience, and not necessarily a commitment to the so-called '2-cycle signalling scheme' [Se80].

Given a digital circuit $C$, we define its (finite) sequential (dynamic) behavior $sb(C, q)$ as the subset of $(\alpha C)^*$ consisting of all finite sequences of input and output events which may occur when $C$ is started in state $q$. Evidently, the pair

$$(\alpha C, sb(C, q))$$

may be viewed as a process [Ho85] or as a trace structure [Sn85].

We assume $sb(C, q)$ to be a regular, prefix-closed language. Thus there always exists a finite automaton $fa(C, q)$ such that

$$sb(C, q) = L(fa(C, q))$$

where $L(fa(C, q))$ is the regular language defined by $fa(C, q)$, with all its states viewed as accepting states.

The concept of (finite), sequential (dynamic) behavior introduced above is restricted by the assumption that two or more events (i.e. input or output changes) may not occur simultaneously. Under this restriction the case of two or more simultaneous events has to be modeled by the events in question appearing sequentially, in any possible order. Thus 'true concurrency' is simulated by non-deterministic interleaving. Indeed, this approach has been widely adopted [Mi80], [Ho85], [Re83], [Sn85]. On the other hand, if concurrent systems are modeled by means of Petri nets, true concurrency and non-deterministic interleaving are easily distinguished [Pe80], [CMPS83], [RV83], [YE83]. In the sequel we shall model digital circuits by means of labeled Petri nets (in the Appendix we summarize the net-theoretical concepts used in this paper). Applying this net-model, we introduce the concept of (finite,) parallel behavior, which takes into account the possibility of two or more events occurring simultaneously. The advantages of using
a net-based model, rather than e.g. trace structures [Sn85] or finite automata are the following.

(1) The concurrency is modeled precisely and efficiently. This provides for the concise modeling of the (dynamic) behavior of complex circuits.

(2) Whenever true concurrency is replaced by non-deterministic interleaving, all events are assumed to be instantaneous or atomic. This assumption is not essential in our net-based model.

To illustrate this point, we consider the marked net \( S \) of Fig. 2.1(a) and the labeled net \( \Gamma \) of Fig. 2.1(b), obtained from \( S \) by replacing true concurrency by interleaving. Clearly, \( L(S) = L(\Gamma) \). If we now replace each transition labeled \( a \) in \( S \) and \( \Gamma \) as indicated in Fig. 2.1(c), we obtain nets \( S' \) and \( \Gamma' \), shown in Fig. 2.1(d) and (e), respectively. However, \( L(\Gamma') \neq L(S') \), but \( L(\Gamma') \subset L(S') \). In particular, \( \langle c, b, d \rangle \in L(S') \), but \( \langle c, b, d \rangle \notin L(\Gamma') \).

It follows that \( \Gamma' \) does not suitably represent the outcome of the replacement shown in Fig. 2.1(c), whereas \( S' \) does.

(3) Composite circuits are modeled easily and efficiently (see Section 3).

Consider again a digital circuit \( C \) with \( \alpha C = inC \cup outC \) as its event alphabet. We describe both the sequential and parallel behavior of \( C \), started in some initial state \( q \), by means of a labeled net (see Appendix) \( net(C,q) \) with \( \alpha C \) as its labeling alphabet. The marking of \( net(C,q) \) represents the initial state \( q \) of \( C \). The firing of a transition labeled by \( a \in inC \) or by \( z \in outC \) corresponds to the occurrence of the input change \( a \) or output change \( z \) in \( C \). Transitions labeled by \( \lambda \) represent internal changes of \( C \), concealed from an outside observer.

Formally, we define the (finite,) parallel (dynamic) behavior of \( (C,q) \) by

\[
pb(C,q) = \pi(net(C,q))
\]

where \( \pi(net(C,q)) \) represents the \( \pi \)-language [YE83] (or subset language [RV83]) of \( net(C,q) \) (see Appendix). Thus \( pb(C,q) \) is a prefix language over the alphabet \( \alpha C = 2^{\alpha C} - \{\lambda\} \).
As usual, we associate with any labeled net $\Gamma$ the ($P^\lambda$ - type [Pe81]) prefix language $L(\Gamma)$ over its labeling alphabet (see, Appendix). For a given circuit $C$ and initial state $q$ we assume that $L(\text{net}(C,q))$ is regular and that the sequential behavior of $(C,q)$ is given by

$$sb(C,q) = L(fa(C,q)) = L(\text{net}(C,q)).$$

We illustrate the above concepts by means of the well-known Muller $C$-element [Se80], [Yo86], which plays an important role in the design of self-timed systems. The logic symbol of this circuit, denoted $CE$, is shown in Fig. 2.2(a). Thus $\text{in}CE = \{a,b\}$ and $\text{out}CE = \{z\}$. A state of $CE$ is represented by the corresponding input-output state, i.e. the values of $A,B,Z$. Let $\text{net}CE$ be the labeled net shown in Fig. 2.2(b). Then

$$\text{net}(CE,000) = \text{net}(CE,111) = \text{net}CE$$

and consequently

$$\text{pb}(CE,000) = \text{pb}(CE,111) = \pi(\text{net}CE).$$

The following are examples of subset strings in $\pi(\text{net}CE)$ (to simplify our notation we replace e.g. $\{z\}$ by $z$):

$$\langle\langle a,b \rangle,z\rangle, \langle a,b,z\rangle, \langle\langle a,b \rangle,a\rangle, \langle a,\{a,b\},a,z\rangle.$$  

Usually a circuit is activated by its environment only under specific restrictions. In particular, we mention the following 'fundamental-mode' [Ko78] restriction $F$, which is adhered to in many applications of asynchronous circuits.

**Restriction F:** An input change may occur only if the circuit is stable.

Stated informally, a state $q$ of a circuit $C$ is stable, iff $C$ remains in state $q$ as long as no input change occurs. Formally, we say that $\text{net}(C,q)$ is stable, iff all the transitions enabled in $\text{net}(C,q)$ are input transitions, i.e. labeled by input events of $C$. Thus state $q$ of $C$ is stable iff $\text{net}(C,q)$ is stable.

We denote by $C/F$ the circuit $C$ with Restriction $F$ imposed on its environment. Also, we denote by $\text{net}(C,q)/F$ the labeled net obtained from $\text{net}(C,q)$ by considering all transitions labeled by output events or by $\lambda$ as having priority over all the input transitions. Thus an input transition in $\text{net}(C,q)/F$ may fire only if $\text{net}(C,q)$ is stable. The behavior of $F$-restricted circuits

Technion - Computer Science Department - Technical Report CS0463 - 1987
are defined by

\[ pb(C \mid F, q) = \pi(\text{net}(C, q) \mid F) \]
\[ sb(C \mid F, q) = L(\text{net}(C, q) \mid F). \]

Referring to the \( CE \)-circuit represented in Fig. 2.2(b), we have

\[ \langle (a, b), a \rangle \in pb(CE, 000) \]

but

\[ \langle (a, b), a \rangle \in pb(CE \mid F, 000) \]

since the marking reached in \( \text{net}CE = \text{net}(CE, 000) \) by firing \( \{a, b\} \) is not stable.

3. BEHAVIOR OF COMPOSITE CIRCUITS

Consider a number of digital circuits suitably interconnected to form a composite circuit. In this section we discuss the problem of deriving the behavior of the composite circuit from the behavioral descriptions of its components. We view each component as a process and view the composite circuit as a system of communicating processes. However, our approach differs from that of [Ho85], [Re83], [Sn85]. As discussed so far, we consider parallel behavior represented by nets, rather than sequential behavior represented by trace structures. We have already mentioned that our net-based approach has advantages over the trace-structure approach, when dealing with circuit compositions. Some of these advantages have already been demonstrated in [Yo86] for a special class of nets, namely marked graphs [CHEP71]. In this section and the following one we extend the approach developed in [Yo86] to nets in general.

We shall proceed as follows. First, we introduce the concept of 'parallel process'. This is an extension of the concept of deterministic process in [Ho85], taking into account the distinction between true concurrency and interleaving. Second, we define the (parallel) composition of parallel processes, extending (in a rather natural way) the concepts of 'concurrency' in [Ho85] and of 'weaving' in [Sn85]. Third, we indicate how the interconnections of digital circuits may be modeled by the compositions of parallel processes. Fourth, we define (in Section 4) the parallel composition of labeled nets and establish a theorem relating the composition of parallel processes to the parallel composition of labeled nets. This theorem is an extension of the corresponding
result derived in [Yd86].

Given a finite alphabet $\Sigma$, we define a subset language over $\Sigma$ to be any language over the extended alphabet $\Sigma = \Sigma \cup \{\emptyset\}$, i.e., any subset of $(\Sigma)^*$. A parallel process $P$ is defined as a pair $P = (\alpha P, \beta P)$ where $\alpha P$ is a finite alphabet (of events) and $\beta P$, the behavior of $P$, is a subset language over $\alpha P$. Thus, given a labeled net $\Gamma$, with labeling alphabet $\Sigma$, the pair $(\Sigma, \pi(\Gamma)) = P(\Gamma)$ is a parallel process. In particular, if $C$ is a digital circuit and $q$ its initial state, then $P(C,q) = P(\text{net}(C,q)) = (\alpha C, pb(C,q))$ is the parallel process associated with $(C,q)$. Similarly, we set $P(C,IF,q) = (\alpha C, pb(C,IF,q))$.

Our definition of process composition uses an extension of the concept of restriction defined in [Ho85].

Given a parallel process $P = (\alpha P, \beta P)$, let $\Sigma \subseteq \alpha P$ and $s \in \beta P$. If the sequence $s$ is nonempty, we denote (cf. [Ho85]) its first element by $s_0$ and the remaining sequence by $s'$. Thus

$s = s_0; s'$, where ';' denotes catenation. The restriction $s \uparrow \Sigma$ is defined by induction. Let $\Lambda$ denote the empty subset sequence. Then

$$\Lambda \uparrow \Sigma = \Lambda$$

For $s = s_0; s'$ we set

$$s \uparrow \Sigma = s_0 \uparrow \Sigma; s' \uparrow \Sigma$$

and

$$s \uparrow \Sigma = s_0 \cap \Sigma; s' \uparrow \Sigma$$

otherwise.

Given two parallel processes, $P = (\alpha P, \beta P)$ and $Q = (\alpha Q, \beta Q)$ we define their (parallel) composition $P \parallel Q$ as follows:

$$\alpha(P \parallel Q) = \alpha P \cup \alpha Q$$

$$\beta(P \parallel Q) = \{s \in \beta P \parallel Q | (s \uparrow \alpha P) \in \beta P \land (s \uparrow \alpha Q) \in \beta Q\}$$

This definition is an extension of the 'concurrency' operation defined in [Ho85], as well as the 'weave' operation in [Re83], [Sn85].

The following example indicates how the concept of (parallel) composition may be applied to the behavioral description of composite circuits. Fig. 3.1 shows a circuit $C$, obtained by inter-
connecting components $C_1$ and $C_2$. We presently assume that the interconnections are observable, i.e., they also appear as outputs of $C$, as indicated in Fig. 3.1. Later on in this section we deal with the (preferable) assumption that interconnections represent internal events (i.e. circuit changes) concealed from the outside. Furthermore, we assume the interconnections to be delay-free. Otherwise explicit delay elements are to be incorporated in modeling composite circuits. Let $q_1$ and $q_2$ be compatible initial states of components $C_1$ and $C_2$, respectively, and let $q$ be the corresponding initial state of $C$. Under these assumptions we postulate the following to hold for the interconnection of arbitrary two circuits $C_1$ and $C_2$.

$$P(C, q) = P(C_1, q_1) \parallel P(C_2, q_2)$$

This postulate can be proved correct on the basis of a suitable structural model of asynchronous VLSI circuits e.g. at the gate level [BY79] or at the switch-level [YB85], [LN84].

We now turn to the assumption that interconnections between circuit components are hidden from an outside observer. This motivates the following definition.

Given two parallel processes, $P = (\alpha P, \beta P)$ and $Q = (\alpha Q, \beta Q)$ we define their **blend** or **merge** $P \parallel Q$ as follows:

$$\alpha(P \parallel Q) = \alpha P \Delta \alpha Q = (\alpha P \cup \alpha Q) - (\alpha P \cap \alpha Q)$$

$$\beta(P \parallel Q) = \beta(P \parallel Q) \uparrow \alpha(P \parallel Q)$$

$$= \{s \uparrow \alpha(P \parallel Q) \mid s \in \beta(P \parallel Q)\}$$

Let $C_1, C_2, C, q_1, q_2, q$ be as above. We denote by $\hat{C}$ the circuit $C$ with its interconnections between $C_1$ and $C_2$ concealed from the outside. We then postulate, similarly as before, that

$$P(\hat{C}, q) = P(C_1, q_1) \parallel P(C_2, q_2).$$

4. PARALLEL COMPOSITION OF LABELED NETS

In this section we define the parallel composition of labeled nets. Given two labeled nets $\Gamma_1, \Gamma_2$, their parallel composition $\Gamma_1 \parallel \Gamma_2$ is to satisfy the condition

$$P(\Gamma_1 \parallel \Gamma_2) = P(\Gamma_1) \parallel P(\Gamma_2).$$

We assume $\Gamma_1$ and $\Gamma_2$ to be node-disjoint, i.e.
where $P_i$ ($i=1,2$) is the set of places of $\Gamma_i$, and $T_i$ its set of transitions.

Let $\Sigma_i$ be the labeling alphabet of $\Gamma_i$ ($i=1,2$).

We shall assume throughout this paper that every symbol in $\Sigma_i$ indeed appears as a label of $\Gamma_i$.

We distinguish four cases.

Case 1: \[ \Sigma_1 \cap \Sigma_2 = \emptyset. \]

In this case $\Gamma_1 \parallel \Gamma_2$ coincides with their union, defined in the evident way.

Case 2: \[ \Sigma_1 \cap \Sigma_2 \neq \emptyset \]

and

\[ \forall \sigma \in \Sigma_1 \cap \Sigma_2. \quad \#(\sigma, \Gamma_1) = \#(\sigma, \Gamma_2) = 1, \]

where $\#(\sigma, \Gamma_i)$ denotes the number of transitions in $\Gamma_i$ labeled by $\sigma$.

This case is illustrated in Fig. 4.1 (cf. [Yo86]). The composition $\Gamma_1 \parallel \Gamma_2$ is derived from the union of $\Gamma_1$ and $\Gamma_2$ by merging (joining) into one transition, each pair of transitions having the same label $\sigma \neq \lambda$.

In order to discuss the remaining cases, we need the concept of 'transition splitting' (see Fig. 4.2). Let $t_0$ be a transition of the labeled net (see Appendix or [YE83]) $\Gamma = (P, T, V, \Sigma, \eta)$. We say that the labeled net $\Gamma' = (P, T', V', \Sigma, \eta')$ is obtained from $\Gamma$ by splitting $t_0$ (into $T_0$) iff the following holds.

\[ T' = (T - \{t_0\}) \cup T_0, \text{ where } T \cap T_0 = \emptyset \text{ and } |T_0| > 1. \]

\[ \forall p \in P. \quad \forall t \in (T - \{t_0\}), \quad V'(p, t) = V(p, t) \wedge V'(t, p) = V(t, p). \]

\[ \forall p \in P. \quad \forall t \in T_0, \quad V'(p, t) = V(p, t_0) \wedge V'(t, p) = V(t_0, p). \]

\[ \forall t \in T - \{t_0\}, \quad \eta'(t) = \eta(t). \]

\[ \forall t \in T_0. \quad \eta'(t) = \eta(t_0). \]

Case 3: \[ \exists \sigma \in \Sigma_1 \cap \Sigma_2. \quad \#(\sigma, \Gamma_1) > 1 \lor \#(\sigma, \Gamma_2) > 1 \]

and

\[ \forall \sigma \in \Sigma_1 \cap \Sigma_2. \quad \#(\sigma, \Gamma_1) = 1 \lor \#(\sigma, \Gamma_2) = 1. \]

Consider the case $\#(\sigma, \Gamma_1) = 1$ and $\#(\sigma, \Gamma_2) = n \geq 1$.

Assume $\eta_1(t_1) = \sigma$ and $\eta_2(t_2^i) = \sigma$ for $i=1, \ldots, n$. We treat this case as follows. We split transition
We then merge each of the $n$ transition pairs $(t^1_i, t^2_j)$ into a single transition $t^i$, labeled by $\sigma$.

The case $\#(\sigma, \Gamma_2) = 1$ and $\#(\sigma, \Gamma_1) > 1$ is handled similarly.

Case 3 will be illustrated in Section 6 (see Examples 6.2, 6.4); together with its application.

Case 4: \( \exists \sigma \in \Sigma_1 \cap \Sigma_2. \#(\sigma, \Gamma_1) > 1 \land \#(\sigma, \Gamma_2) > 1. \)

We consider the general case $\#(\sigma, \Gamma_1) = m \geq 1$ and $\#(\sigma, \Gamma_2) = n \geq 1$. Assume $\eta_1(t^i_i) = \sigma$ for $i=1, \ldots, m$ and $\eta_2(t^j_2) = \sigma$ for $j=1, \ldots, n$. We split each transition $t^i_i$ into $(t^1_i, \ldots, t^n_i)$ and each transition $t^j_2$ into $(t^1_2, \ldots, t^n_2)$. We then merge each of the $m \cdot n$ transition pairs $(t^i_i, t^j_2) \ (1 \leq i \leq m, 1 \leq j \leq n)$ into a new transition $t^i_j$, labeled $\sigma$. We proceed accordingly for every $\sigma \in \Sigma_1 \cap \Sigma_2$.

The following result generalizes Proposition 4(a) in [Yo86]. Its rather lengthy proof is beyond the scope of the present paper.

**Theorem 1:** \( P(\Gamma_1 \parallel \Gamma_2) = P(\Gamma_1) \parallel P(\Gamma_2). \)

Corresponding to the $\parallel$ -operation defined for parallel processes (Section 3), we define $\Gamma_1 \parallel \Gamma_2$ as the labeled net $\Gamma_1 \parallel \Gamma_2 \Lambda(\Sigma_1 \cap \Sigma_2)$, i.e. the labeled net $\Gamma_1 \parallel \Gamma_2$ with all labels in $\Sigma_1 \cap \Sigma_2$ replaced by $\Lambda$ (cf. [Yo86]).

One easily verifies the following

**Proposition 1:** \( P(\Gamma_1 \parallel \Gamma_2) = P(\Gamma_1) \parallel P(\Gamma_2). \)

5. SPECIFICATIONS OF DIGITAL CIRCUITS

A "specification" of a digital circuit is a description of the way the circuit is intended to behave [Ho85]. An implementation of a circuit "satisfies" a given specification, if it meets the behavioral requirements described by the specification. In this section we wish to formulate these concepts in a precise way. In order to properly motivate our final formulation, we first discuss
various examples.

In the sequel let $\Gamma_{\text{imp}}$ and $\Gamma_{\text{sp}}$ be labeled nets representing the implementation of a digital circuit and its specification, respectively. We wish to define the relation '$\Gamma_{\text{imp}} \text{ sat } \Gamma_{\text{sp}}$', i.e. $\Gamma_{\text{imp}}$ satisfies the specification $\Gamma_{\text{sp}}$.

The following two requirements are no doubt acceptable.

(1) $\Sigma(\Gamma_{\text{imp}}) = \Sigma(\Gamma_{\text{sp}})$

where $\Sigma(\Gamma)$ denotes the labeling alphabet of $\Gamma$.

(2) $\pi(\Gamma_{\text{sp}}) \subseteq \pi(\Gamma_{\text{imp}})$.

This 'inclusion' requirement assures that the implementation is capable of producing any finite sequence of multiple events specified by $\Gamma_{\text{sp}}$. However, the next two examples will point out that requirements (1) and (2) are not sufficient.

Example 5.1 Consider the labeled net $\Gamma_2$ of Fig. 5.1(c) to be an implementation of the specification net $\Gamma_1$ of Fig. 5.1(b). Evidently

$$\Sigma(\Gamma_1) = \Sigma(\Gamma_2) \text{ and } \pi(\Gamma_1) \subseteq \pi(\Gamma_2).$$

Thus requirements (1) and (2) are satisfied. Nevertheless we wish to argue that the implementation net $\Gamma_2$ does not satisfy the specification net $\Gamma_1$. Namely, $\Gamma_1$ contains the requirement that an input event $a$ is followed by the output events $z_1$, $z_2$ in the order specified, i.e. $z_1$ first, and then $z_2$. On the other hand, we have

$$\langle a, z_2, z_1 \rangle \in \pi(\Gamma_2)$$

as well as

$$\langle a, z_1, z_2 \rangle \in \pi(\Gamma_2).$$

Thus, a circuit represented by $\Gamma_2$ may produce undesirable event sequences.

Example 5.2 Referring to Fig. 5.2, we again have $\Sigma(\Gamma_3) = \Sigma(\Gamma_4)$ as well as $\pi(\Gamma_3) \subseteq \pi(\Gamma_4)$. However, we would reject $\Gamma_4$ as an implementation of $\Gamma_3$ since $\langle b, z \rangle \in \pi(\Gamma_4)$, whereas $\langle b, z \rangle \notin \pi(\Gamma_3)$.

In view of the above two examples, we formulate the following "exclusion" requirement.

(3) Let 'out' be the set of output events of both $\Gamma_{\text{sp}}$ and $\Gamma_{\text{imp}}$. Then, for every $s \in \pi(\Gamma_{\text{sp}})$ and
The following example illustrates yet another difficulty. Let $\Gamma_5$ and $\Gamma_6$ be the labeled nets shown in Fig. 5.3. We set $\Gamma_{sp} = \Gamma_5$ and $\Gamma_{imp} = \Gamma_6$. The pair $\Gamma_{sp} = \Gamma_5$ and $\Gamma_{imp} = \Gamma_6$ satisfy all the three preceding requirements (1), (2), (3). However, the non-deterministic behavior of the circuit represented by $\Gamma_6$ is no doubt undesirable. We overcome this difficulty by postulating that both $\Gamma_{sp}$ and $\Gamma_{imp}$ have the "SD-property" (SD = Structurally Deterministic, cf. [YE83]) defined as follows:

A labeled net has the SD-property iff no two transitions having the same label are enabled under the initial marking or any other reachable marking.

The specification and implementation of most useful circuits will no doubt meet this postulate.

We are thus led to the following definition. Let $\Gamma_{sp}$ and $\Gamma_{imp}$ be labeled nets which have the SD-property. Then $\Gamma_{imp} sat \Gamma_{sp}$ holds iff the above requirements (1), (2) and (3) are met.

6. VERIFICATION OF ASYNCHRONOUS CIRCUITS

Given a digital circuit $C$, its initial state $q$, and a specification net $\Gamma_{sp}$, we verify $(C,q)$ with respect to $\Gamma_{sp}$, by proving that the relation $\colrule (C,q) sat \Gamma_{sp}$, holds. In this section we illustrate the verification of various digital circuits. In particular, we shall be concerned with the verification of asynchronous circuits which contain CE-components. In most of the examples we omit the details of the proof, since they require net-theoretical arguments beyond the scope of the present paper. Such arguments will be incorporated in a forthcoming paper dealing with the synthesis of a variety of asynchronous circuits.

Following [Yo86], we denote the net shown in Fig. 6.1(a) by $*[a,z]$. Thus the net of Fig. 6.1(b) becomes $*[a,z] \parallel *[b,z]$.

Example 6.1 The net shown in Fig. 6.1(b) is obtained from netCE of Fig. 2.2(b) by omitting
some of the input transitions. Consequently, one easily verifies the following.

\[
\text{net}(CE,000) = \text{net}(CE,111) = \text{net}CE \mathbin{\mathbin{\mathcal{S}} \mathbin{\mathcal{A}} \mathbin{\mathcal{T}} [a,z] \mathbin{\mathcal{I}} *[b,z] .
\]

Example 6.2 Consider the composite circuit \(CCT1\) shown in Fig. 6.2, with the initial state \(A=B=C=Z=0\). We wish to show that

\[
\text{net}(CCT,0000) \mathbin{\mathcal{S}} \mathbin{\mathcal{A}} \mathbin{\mathcal{T}} [a,z] \mathbin{\mathcal{I}} *[b,z] \mathbin{\mathcal{I}} *[c,z].
\]

We denote by \(CE[IN1,IN2;OUT]\), the \(CE\)-circuit with its terminals relabeled accordingly. Relabeling of a net is indicated similarly. Thus

\[
\text{net}(CCT,0000) = \text{net}(CE[A,B;Y],000) \mathbin{\mathcal{I}} \text{net}(CE[Y,C;Z],000) = \text{net}CE[a,b;y] \mathbin{\mathcal{I}} \text{net}CE[y,c;z].
\]

The parallel composition of \(\text{net}CE[a,b;y]\) and \(\text{net}CE[y,c;z]\) is shown in Fig. 6.3. Note that the label \(y\) appears once in the first net and twice in the second net. Thus Case 3 (see Section 4) applies. By replacing the \(y\)-labels in Fig. 6.3 by \(\lambda\) we obtain the \(\mathbin{\mathcal{I}}\) -composition (merge, blend) of the two nets, which coincides with \(\text{net}(CCT,0000)\). The above statement (1) is now easily proven.

The \(CE\)-circuit is frequently used with the following 'A-Restriction' ("Acknowledge"-Restriction) imposed on its inputs (cf. [Se80]):

An input may change only if it has the same value as the output.

Whenever relevant we use the notation \(CCT/A\) and \(\Gamma/A\) to denote circuits and nets with the A-restriction imposed on them. Thus

\[
\text{net}(CE/A,000) = \text{net}(CE/A,111) = \text{net}CE/A = *[a,z] \mathbin{\mathcal{I}} *[b,z].
\]

We may consider \(*[a,z] \mathbin{\mathcal{I}} *[b,z] \mathbin{\mathcal{I}} *[c,z]\) the specification of a 3-input \(CE\)-circuit with the A-restriction imposed on its environment. Thus, Example 6.2 may be viewed as a precise formulation of the "associativity" of the \(CE\)-circuit mentioned in [Se80].

Example 6.3 The labeled net \(\Gamma_7\) of Fig. 6.4 may be viewed as a specification of a 3-input \(CE\)-circuit, without the A-restriction. We wish to show that the statement

\[
\text{net}(CCT,0000) \mathbin{\mathcal{S}} \Gamma_7
\]

does not hold. Indeed (see Fig. 6.3)
\[(a,b),y,b,c,z) \in \pi(\text{netCE}[a,b;y] \parallel \text{netCE}[y,c;z])\

hence

\[(a,b),y,b,c,z) \in \pi(\text{net}(\text{CCT1,0000}))\

but

\[(a,b),y,b,c,z) \in \pi(\Gamma_7)\

whereas

\[(a,b),y,b,c,z) \in \pi(\Gamma_7).\

Consequently, the above statement (2) is not valid.

Stated informally, the circuit \(\text{CCT1}\) (Fig. 6.2) with the initial state \(A=B=C=Z=0\) does not satisfy the requirements of the specification represented by net \(\Gamma_7\) of Fig. 6.4. According to this specification the sequence

\[\langle A \uparrow, B \uparrow, B \downarrow, C \uparrow\rangle\]

should not produce the output change \(Z\uparrow\), since the 'request' \(B \uparrow\) has been 'canceled' by \(B \downarrow\). However, in the circuit \(\text{CCT1}\) with the given initial state, the above sequence will produce the output change \(Z\uparrow\).

Example 6.4  Fig. 6.5(a) shows the logic symbol of an inverter (INV) and Fig. 6.5(b) its net representation \(\text{netINV}\). The following clearly holds:

\[\text{net}(\text{INV},01) = \text{net}(\text{INV},10) = \text{netINV}\]

\[\text{net}(\text{INV/F},01) = \text{net}(\text{INV/F},10) = \text{netINV/F} = *[a,z].\]

We now consider the composite circuit \(\text{CCT2}\) of Fig. 6.6 and show that

\[\text{net}(\text{CCT2,100}) \text{ sat } \text{netCE}.\]  \(3\)

It follows that \(\text{CCT2}\) behaves similarly to a SR-flipflop with \(A\) as its reset and \(B\) as its set input. Its interesting feature (in which it differs from the usual SR-flipflop) is that both \(A=B=0\) and \(A=B=1\) are input combinations which remember the previous state and output value.

Let

\[\Gamma_1 = \text{net}(\text{INV}[A;Y],10) = \text{netINV}[a;y]\]

and

\[\Gamma_7 = \text{net}(\text{INV}[A;Y],10) = \text{netINV}[a;y]\]
\[ \Gamma_2 = \text{net}(CE[Y, B; Z], 000) = \text{netCE}[y, b; z]. \]

Thus

\[ \text{net}(CCT2, 100) = \Gamma_1 \parallel \Gamma_2. \]

\( \Gamma_1 \parallel \Gamma_2 \) is shown in Fig. 6.7. This example again illustrates Case 3 of the relevant definition (see Section 4). \( \Gamma_1 \parallel \Gamma_2 \) is obtained from \( \Gamma_1 \parallel \Gamma_2 \) by replacing both \( y \)-labels by \( \lambda \). Statement (3) follows easily.
REFERENCES

LNCS = Lecture Notes in Computer Science, Springer Verlag.


[Pe80] Petri, C.A., "Concurrency", in [Br80].


APPENDIX - Nets and Net Languages

In this Appendix we summarize the concepts concerning (Petri) nets and net languages which are used in this paper. We assume some familiarity with the basic concepts of Petri nets and their graphical representation (cf. [Ag79], [Br80], [Pe81], [Re85]). The following summary is mainly based on [YE83], [RV83].

A (Petri) net consists of
(1) a finite set \( P \) (or \( S \)) of places;
(2) a finite set \( T \) of transitions;
(3) a finite set \( F \) of arcs, representing the flow relation, where \( F \subseteq (P \times T) \cup (T \times P) \).

We assume \( P \cap T = \emptyset \) and \( P \cup T \neq \emptyset \).

With a given net we associate a valuation (or weight) function \( V : (P \times T) \cup (T \times P) \rightarrow \{0,1\} \), where
\[
V(x,y) = 1 \text{ if } (x,y) \in F, \text{ else } V(x,y) = 0.
\]

We assume the reader to be familiar with the net concepts of marking, firing, and firing sequence. Let \( N \) be a net and \( M \) its initial marking. We associate with the marked net \( S = (N,M) \) a prefix-closed language \( L(S) \) over the alphabet \( T \), consisting of all firing sequences of \( S \), including the empty sequence \( \lambda \).

In our paper the multiple (or concurrent) firing of transitions plays an important role (cf. [YE83], [RV83]). Let \( U \) be a non-empty set of transitions of the marked net \( S \). \( U \) is concurrently firable in \( S = (N,M) \) iff
\[
\forall p \in P. \ M(p) \geq \sum_{t \in U} V(p,t)
\]
In Fig. A.1, \( \{a,b\} \) is concurrently firable in the marked nets \( S_1, S_2, \) and \( S_4 \), but not in \( S_3 \) and \( S_5 \).

If \( U \) is concurrently firable in \( S = (N,M) \), the firing of \( U \) yields a new marking \( M' \), which can also be reached by firing all the transitions of \( U \) sequentially (one after the other) in any order. One easily verifies that the marking \( M' \) reached does not depend on the order in which the transitions of \( U \) are fired. Multiple-firing sequences are defined in the obvious way, similarly to the usual (single-) firing sequences. Following [YE83] we denote by \( \pi(S) \) the set of all finite
multiple-firing sequences of $S$. Thus $\pi(S)$ is a prefix-closed (subset) language over the alphabet $2^T - \{\emptyset\}$.

A labeled net $\Gamma$ consists of a marked net $S$, a finite labeling alphabet $\Sigma$ and a labeling function $\eta: T \to \Sigma \cup \{\lambda\}$, where $\lambda$ denotes the empty sequence in $\Sigma^*$.

The language $L(\Gamma)$ over the alphabet $\Sigma$ is given by

$$L(\Gamma) = \{\eta(w) \mid w \in L(S)\}$$

where $\eta(w)$ is defined in the evident way.

We now wish to define the (subset) language $\pi(\Gamma)$ over the alphabet $\Sigma = 2^T - \{\emptyset\}$.

For any nonempty subset $U$ of $T$, let

$$\eta(U) = \begin{cases} \lambda & \text{if } \{\eta(t) \mid t \in U\} = \{\lambda\} \\ \emptyset & \text{else} \end{cases}$$

Here $\Lambda = \{\lambda\}$ denotes the empty sequence in $\Sigma^*$. $\pi(\Gamma)$ is now given by

$$\pi(\Gamma) = \{\eta(W) \mid W \in \pi(S)\}.$$
Fig. 2.1 - (a) Marked net $S$
(b) Labeled net $\Gamma$ ($L(\Gamma) = L(S)$)
(d) Marked net $S'$ obtained by applying (c) to $S$.
(e) Labeled net $\Gamma'$ obtained by applying (c) to $\Gamma$ ($L(\Gamma') \neq L(S')$).
Fig. 2.2 - (a) Logic symbol for the CE-circuit
(b) netCE
Fig. 3.1 - Example of composite circuit C.
Fig. 4.1 - Illustrating the parallel composition $\Gamma_1 \parallel \Gamma_2$ (Case 2)

(a) $\Gamma_1$  (b) $\Gamma_2$  (c) $\Gamma_1 \parallel \Gamma_2$
Fig. 4.2 - (a) labeled net $\Gamma$
(b) labeled net $\Gamma'$ obtained from $\Gamma$ by transition splitting

Fig. 5.1 (a) Abbreviated notation for nets
(b) Labeled net $\Gamma_1$
(c) Labeled net $\Gamma_2$
Fig. 5.2 - (a) Labeled net $\Gamma_3$
(b) Labeled net $\Gamma_4$

Fig. 5.3 - (a) Labeled net $\Gamma_5$
(b) Labeled net $\Gamma_6$
Fig. 6.1 - (a) net *\[a,z\]
(b) net *\[a,z\] \parallel *\[b,z\]

Fig. 6.2 - Composite circuit CCT1
Fig. 6.3 - The parallel composition of netCE[a,b;y] and netCE[y,c;z]

Fig. 6.4 - Labeled net $\Gamma_7$
Fig. 6.5 - Inverter (INV):
(a) Symbol
(b) netINV

Fig. 6.6 - Composite circuit CCT2
Fig. 6.7 - netINV[α;γ] || netCE[γ,β;ζ]

Fig. A.1 - {α,β} is concurrently firable in the marked nets S₁, S₂, and S₄, but not in nets S₃ and S₅.