NET-BASED MODELING OF COMMUNICATING PARALLEL PROCESSES WITH APPLICATIONS TO VLSI DESIGN

by

I. Reicher and M. Yoeli

Technical Report #532
December 1988
NET-BASED MODELING OF COMMUNICATING PARALLEL PROCESSES
WITH APPLICATIONS TO VLSI DESIGN

by

I. Reicher and M. Yoeli

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

ABSTRACT

In this paper we introduce a novel definition of the concept 'implementation satisfies specification' with regard to digital circuits.

We model composite circuits as communicating parallel processes, extending the well-known theory of communicating sequential processes due to C.A.R. Hoare.

We develop a practical approach, based on Petri nets, to the verification of asynchronous composite circuits.
1. INTRODUCTION

This paper applies a suitable extension of the theory of labeled Petri nets and their subset languages [YE83], [RV83] to the modeling of communicating parallel processes (Sections 3, 4). The concept of parallel process introduced (Section 2) is similar to that of process in [Ho85]; however true concurrency, i.e. the simultaneous occurrence of two or more events is admitted, rather than replacing it by non-deterministic interleaving.

The theory developed is particularly applicable to the behavioral description, specification and verification of asynchronous (VLSI) circuits. In [Yo87] the advantage of using marked graphs rather than trace structures [Rem85], [Sn85] to the modeling and verification of asynchronous circuits was demonstrated. As to the advantages of Petri nets for circuit specifications see also [Ch87]. The present paper extends the approach of [Yo87] from marked graphs to arbitrary Petri nets.

A novel feature of this paper is the way in which the concept implementation satisfies specification is formalized. Indeed, two alternative ways are proposed (Section 5) and then shown to be equivalent (Section 6). A number of verification examples are discussed in Section 7.

2. PARALLEL PROCESSES AND THEIR COMPOSITIONS

Given a finite alphabet \( \Sigma \), we define a subset language over \( \Sigma \) to be any language over the extended alphabet \( \tilde{\Sigma} = 2^\Sigma - \{\emptyset\} \), i.e. any subset of \( (\tilde{\Sigma})^* \) (cf. [RV83], [YE83]). 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 \( \alpha P \).

For the purpose of defining process compositions, we need an extension of the concept of restriction as 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 \( s \downarrow \Sigma \) is defined by induction. Let \( \Lambda \) denote
the empty subset sequence. Then
\[ \Lambda \setminus \Sigma = \Lambda. \]
For \( s = s_0 \cup s' \) we set
\[ s \setminus \Sigma = s_0 \setminus \Sigma \text{ if } s_0 \cap \Sigma = \emptyset \]
and
\[ s \setminus \Sigma = s_0 \cap \Sigma ; s' \setminus \Sigma \text{ otherwise.} \]

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:
\[ \alpha(P \parallel Q) = \alpha P \cup \alpha Q \]
\[ \beta(P \parallel Q) = \{ s \in \alpha^*(P \parallel Q) \mid (s \setminus \alpha P) \in \beta P \wedge (s \setminus \alpha Q) \in \beta Q \}. \]

This definition is an extension of the 'concurrency' operation defined in [Ho85], as well as the 'weave' operation in [Rem85], [Sn85].

Frequently it is convenient to assume that the interconnections between two (communicating) processes are hidden from an outside observer. This motivates the following definition (cf. [Rem85], [Sn85], [Yo87]).

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:
\[ \alpha(P \parallel Q) = \alpha P \Delta \alpha Q = (\alpha P \cup \alpha Q) \setminus (\alpha P \cap \alpha Q) \]
\[ \beta(P \parallel Q) = \beta(P \parallel Q) \setminus \alpha(P \parallel Q) \]
\[ = \{ s \setminus \alpha(P \parallel Q) \mid s \in \beta(P \parallel Q) \}. \]

### 3. LABELED NETS AND THEIR COMPOSITIONS

Henceforth we shall mainly be interested in parallel processes which may be represented by labeled Petri nets (in the Appendix we summarize the net-theoretical concepts used in this paper). We say that the parallel process \( P = (\alpha P, \beta P) \) is represented by the labeled net \( \Gamma \) with labeling alphabet \( \Sigma(\Gamma) \), iff \( \Sigma(\Gamma) = \alpha P \) and \( \pi(\Gamma) = \beta P \).

In this section we define the parallel composition of labeled nets. In the next section we prove that the parallel composition \( \Gamma_1 \parallel \Gamma_2 \) of the labeled nets \( \Gamma_1, \Gamma_2 \) satisfies the condition...
\[ P(\Gamma_1 \parallel \Gamma_2) = P(\Gamma_1) \parallel P(\Gamma_2), \]

where \( P(\Gamma) \) denotes the parallel process represented by \( \Gamma \), i.e. \( P(\Gamma) = (\Sigma(\Gamma), \pi(\Gamma)) \).

Let \( \Gamma_i = (P_i, T_i, V_i, M_i, \Sigma_i, \eta_i) \) \( i = 1, 2 \) be two labeled nets.

We assume \( \Gamma_1 \) and \( \Gamma_2 \) to be node-disjoint, i.e.

\[ (P_1 \cup T_1) \cap (P_2 \cup T_2) = \emptyset \]

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

We distinguish two 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.

In order to discuss the general case, we need the concept of 'transition splitting' (see Fig. 3.1). Let \( t_0 \) be a transition of the labeled net (see Appendix or [YE83]) \( \Gamma = (P, T, V, M, \Sigma, \eta) \). We say that the labeled net \( \Gamma' = (P', T', V', M, \Sigma, \eta') \) is obtained from \( \Gamma \) by splitting \( t_0 \) (into \( T_0 \)) iff the following holds:

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

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

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

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

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

**Case 2:** \( \Sigma_1 \cap \Sigma_2 \neq \emptyset \).

Let \( \sigma \) be in \( \Sigma_1 \cap \Sigma_2 \). We consider the general case \#(\sigma, \Gamma_i) = m \geq 1 \) and \#(\sigma, \Gamma_2) = n \geq 1, where \#(\sigma, \Gamma_i) denotes the number of transitions in \( \Gamma_i \) labeled by \( \sigma \). Assume \( \eta_1(t_{1i}) = \sigma \) for \( i = 1, \ldots, m \) and \( \eta_2(t_{2j}) = \sigma \) for \( j = 1, \ldots, n \). We split each transition \( t_{1i} \) into \( \{t_{1i}', \ldots, t_{1i}^\eta\} \) and each transition \( t_{2j} \) into \( \{t_{2j}', \ldots, t_{2j}^\eta\} \). We then merge each of the \( m \cdot n \) transition pairs \( \{t_{1i}', t_{2j}'\} \) (1 \( \leq i \leq m \), 1 \( \leq j \leq n \)) into a new transition \( t_i' \), labeled \( \sigma \). We proceed accordingly for every \( \sigma \in \Sigma_1 \cap \Sigma_2 \). This case is illustrated in Fig. 3.2.

The transition "merging" described in the above Case 2 is a well known concept of Petri net theory (see e.g. [Di82]). However, the following Theorem 1 is new.
Corresponding to the $\parallel$-operation defined for parallel processes in Section 2, we define $\Gamma_1 \parallel \Gamma_2$ as the labeled net $\Gamma_1 \parallel \Gamma_2$ with all labels in $\Sigma_1 \cap \Sigma_2$ replaced by $\lambda$ (cf. [Yo87]).

4. COMPOSITION THEOREM

In this section we provide the proof of the following result, which relates net compositions to the corresponding process compositions. This result generalizes related propositions stated in [Yo87].

Theorem 1: Let $\Gamma_1$, $\Gamma_2$ be labeled nets. Then

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

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

We prove this theorem by means of a sequence of rather simple lemmata.

The following lemma is an immediate consequence of the definition of transition splitting.

Lemma 1: Let $r=(P,T,V,M,I,\Sigma)$ be a labeled net and assume that $r'=(P',T',V',M',I',\Sigma')$ is obtained from $r$ by splitting one of its transitions. Let $s \in \Sigma^*$. Then

$M[s]M' \text{ in } r \iff M[s]M' \text{ in } r'$.

Thus $\pi(r') = \pi(r)$.

The following lemma follows easily from Lemma 1 and the definition of $\Gamma_1 \parallel \Gamma_2$.

Lemma 2: Let $\Gamma_i=(P_i,T_i,V_i,M_i,\Sigma_i,\eta_i)$, $i=1,2$, be two labeled nets and let $\Gamma = \Gamma_1 \parallel \Gamma_2 = (P,T,V,M,\Sigma,\eta)$. Then $P = P_1 \cup P_2$ and $M_i = M \upharpoonright P_i$, where the restriction $M \upharpoonright P_i$ is defined in the obvious way.

Furthermore, let $x$ be a nonempty subset of $\Sigma$, and assume that $M(x)M'$. Then

$M_i[x \cap \Sigma_i](M' \upharpoonright P_i)$ if $x \cap \Sigma_i \neq \emptyset$

and

$M' \upharpoonright P_i = M_i$ if $x \cap \Sigma_i = \emptyset$.

Henceforth, throughout this section, we use the notation of Lemma 2.

Lemma 3: $\beta P(\Gamma_1 \parallel \Gamma_2) \subseteq \beta(P(\Gamma_1) \parallel P(\Gamma_2))$. 


Proof: Let $s \in \pi(\Gamma)$. By induction on the length of $s$, it follows from Lemma 2 that

$$M(s)M' \Rightarrow M_i(s \uparrow \Sigma_i)(M'_i \uparrow P_i), \quad (i = 1, 2).$$

Hence

$$s \in \beta P(\Gamma_1 \parallel \Gamma_2) \Rightarrow s \in \beta (P(\Gamma_1) \parallel P(\Gamma_2)).$$

The following Lemma 4 is another easy consequence of Lemma 1 and the definition of $\Gamma_1 \parallel \Gamma_2$.

Lemma 4.

(a) Let $x \subseteq \Sigma_1, x \neq \emptyset, x \cap \Sigma_2 = \emptyset$, and $M_1[x] > M'_1$.

Then $M(x)M'$, where $M' = M'_1 \cup M_2$.

Clearly the case $x \subseteq \Sigma_2, x \neq \emptyset, x \cap \Sigma_1 = \emptyset$ and $M_2(x)M'_2$ is similar.

(b) Let $x \subseteq \Sigma, x \cap \Sigma_1 \neq \emptyset, x \cap \Sigma_2 \neq \emptyset$,

$M_1[x \cap \Sigma_1] M'_1, M_2[x \cap \Sigma_2] M'_2$.

Then $M(x)M'$, where $M' = M'_1 \cup M'_2$.

Lemma 5. $\beta (P(\Gamma_1) \parallel P(\Gamma_2)) \subseteq \beta P(\Gamma_1 \parallel \Gamma_2)$.

Proof: For $s \in \Sigma^*$ let

$$M_i(s \uparrow \Sigma_i)M'_i, \quad i = 1, 2.$$

Applying Lemma 4 and induction on the length of $s$, we obtain $M(s)M'$, where $M' = M'_1 \cup M'_2$.

Thus

$$s \in \beta (P(\Gamma_1) \parallel P(\Gamma_2)) \Rightarrow s \in \beta P(\Gamma_1 \parallel \Gamma_2).$$

Clearly,

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

Thus, in view of Lemmata 3 and 5, we have

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

This proves Theorem 1(a).

Part (b) of Theorem 1 follows easily from Theorem 1(a) and the definitions of the $\parallel$-compositions of nets and processes.
5. BEHAVIORAL DESCRIPTION AND SPECIFICATION OF ASYNCHRONOUS CIRCUITS

In this section we introduce the concepts of behavior and specification of digital circuits, viewed as parallel processes with distinct input and output events.

We consider a digital circuit $CCT$ with binary inputs and outputs. We use upper-case letters to denote the inputs and outputs of $CCT$, 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 $CCT$ and denote by $inCCT$ (out$CCT$) the set of input (output) events of $CCT$. We call $\alpha CCT = inCCT \cup outCCT$ the event alphabet of $CCT$.

Given a digital circuit $CCT$, and its initial state $q$, we associate with $(CCT,q)$ a parallel process $P(CCT,q) = (\alpha P, \beta P)$, where $\alpha P = \alpha CCT$. We assume that $P(CCT,q)$ may be represented by a labeled net, denoted $net(CCT,q)$. Thus $P(net(CCT,q)) = P(CCT,q)$. The behavior $\beta P$ of $P(CCT,q)$ corresponds to the set of all finite sequences of multiple (input and output) events which may occur when the circuit $CCT$ is started in state $q$.

Assume now that the digital circuit $CCT$ is obtained by suitably interconnecting two components $CCT_1$ and $CCT_2$. Furthermore, let $q_1$ and $q_2$ be compatible initial states of $CCT_1$ and $CCT_2$, respectively, and let $q$ be the corresponding initial state of $CCT$. Under these assumptions we postulate the following to hold, provided the interconnections are observable, i.e. they also appear as outputs of $CCT$.

$$P(CCT,q) = P(CCT_1,q_1) \parallel P(CCT_2,q_2).$$

Let $CCT'$ be the circuit $CCT$ with its interconnections between $CCT_1$ and $CCT_2$ concealed from the outside. We then postulate that

$$P(CCT',q) = P(CCT_1,q_1) \perp P(CCT_2,q_2).$$

These postulates are based on the assumption that the output of one circuit is connected to the input of the other circuit by interconnections which are delay free. The postulates can be proven formally by extending the "GMW" (General Multiple Winner) model of [BY 79].
In the sequel we wish to discuss circuit specifications. It is noteworthy that our concept of circuit specification also incorporates restrictions imposed on the environment. In particular, we wish to formulate precisely the relation between circuit specification and circuit implementation. We define a circuit net to be a labeled net $\Gamma$, whose labeling alphabet $\Sigma(\Gamma)$ is partitioned into an input alphabet $in\Gamma$ and an output alphabet $out\Gamma$. In particular, if $\Gamma = net(CCT, q)$, we assume that $in\Gamma = inCCT$ and $out\Gamma = outCCT$.

Furthermore, we assume circuit nets to have the "SD-property" (SD = Structurally Deterministic, cf. [YE83]) defined as follows:

A labeled net has the SD-property iff no two transitions $t_1, t_2$ having the same label in $\Sigma$ (i.e., $\eta(t_1) = \eta(t_2) \neq \lambda$) are 'weakly enabled' under the initial marking or any other reachable marking. A transition $t$ is weakly enabled under the marking $M$ iff $\exists w \in T^* : \eta(w) = \lambda \wedge M[w] > 0$.

Nets representing useful circuits will no doubt have the SD-property.

Given a circuit net $\Gamma_{imp}$ representing a circuit implementation and another circuit net $\Gamma_{sp}$ representing a circuit specification we say that $\Gamma_{imp}$ satisfies $\Gamma_{sp}$ (notation: $\Gamma_{imp} sat \Gamma_{sp}$) iff the following requirements are met:

1. $in\Gamma_{sp} = in\Gamma_{imp}$
2. $out\Gamma_{sp} = out\Gamma_{imp}$
3. $\pi(\Gamma_{sp}) \subseteq \pi(\Gamma_{imp})$
4. For every $s \in \pi(\Gamma_{sp})$ and every $x \subseteq out\Gamma_{sp}$, $s; x \in \pi(\Gamma_{imp})$ implies $s; x \in \pi(\Gamma_{sp})$.

Requirement (1) is obvious. Requirement (2) postulates that the implementation "can do" everything it is specified to do. However, the implementation should not produce "unwanted" outputs. This motivates Requirement (3). The next two examples will explain requirement (3).

Example 5.1: Consider the circuit net $\Gamma_2$ of Fig. 5.1(b) to be an implementation of the specification net $\Gamma_1$ of Fig. 5.1(a). Evidently
Thus requirements (1) and (2) are satisfied.

Nevertheless, 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

$$<a, z_1, z_2> \in \pi(\Gamma_2)$$

as well as

$$<a, (z_1, z_2) > \in \pi(\Gamma_2)$$

and

$$<a, z_2, z_1 > \in \pi(\Gamma_2).$$

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 $<b, z > \in \pi(\Gamma_4)$, whereas $<b, z > \notin \pi(\Gamma_3)$.

The following example illustrates the importance of the SD-property in connection of sat.

Example 5.3: 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 three preceding requirements of sat. However, the non-deterministic behavior of the circuit represented by $\Gamma_6$ is no doubt undesirable as an implementation of $\Gamma_5$, and indeed $\Gamma_6$ does not have the SD-property.

We shall now discuss another method of relating a circuit implementation to its specification. A circuit specification also imposes some restrictions on the behavior of the environment. We wish to express the following idea: if the environment restrictions contained in the specification are imposed on its implementation, then the outcome should be equivalent to the specification.

To formalize this idea, we introduce the following definitions.

Let $\Gamma$ be a circuit net. With $\Gamma$ we associate its environment restrictions net $\Gamma_{env}$ obtained from $\Gamma$ by omitting all the edges incoming to output transitions, i.e. transitions labeled by
symbols from $\text{out} \Gamma$. Thus, $e \Gamma$ contains only the incoming edges to input transitions which restrict the operation of the input transitions, i.e., the restrictions imposed by the environment on the inputs:

Let $\Gamma_{sp}$ and $\Gamma_{imp}$ be circuit nets. $\Gamma_{imp}$ is said to cover $\Gamma_{sp}$ (notation: $\Gamma_{imp} \text{ cov } \Gamma_{sp}$) iff the following conditions are met:

\begin{align*}
(C1) \quad & \text{in} \Gamma_{imp} = \text{in} \Gamma_{sp} \\
& \text{out} \Gamma_{imp} = \text{out} \Gamma_{sp}
\end{align*}

\begin{align*}
(C2) \quad & \pi(\Gamma_{imp} \parallel e \Gamma_{sp}) = \pi(\Gamma_{sp}).
\end{align*}

In the next section we compare the binary relations sat and cov.

6. AN EQUIVALENCE THEOREM

In this section we shall state and prove that the binary relations sat and cov are "essentially" the same. To make this statement precise, we need the following definition.

A circuit net $\Gamma$ is said to be well-formed iff the following condition is always met:

Let $a \subseteq \text{in} \Gamma$, $z \subseteq \text{out} \Gamma$, and $s \in \pi(\Gamma)$. Assume that

$s; a; z \in \pi(\Gamma)$

and

$s; z; a \in \pi(\Gamma)$

then

$s; a \cup z \in \pi(\Gamma)$.

One realizes that any actual digital circuit will satisfy this condition, since under the above assumptions the environment cannot prevent the concurrent occurrence of the multiple input event $\bar{a}$ and the multiple output event $z$. Hence we may restrict our considerations to specification nets which are well-formed. We now have
Theorem 2. Let $\Gamma_{sp}$ and $\Gamma_{imp}$ be circuit nets and assume $\Gamma_{sp}$ to be well-formed. Then

$$\Gamma_{imp} \, sat \, \Gamma_{sp} \iff \Gamma_{imp} \, cov \, \Gamma_{sp}.$$ 

Proof:

(A) In Part (A) we shall prove

$$\Gamma_{imp} \, cov \, \Gamma_{sp} \Rightarrow \Gamma_{imp} \, sat \, \Gamma_{sp}.$$ 

Thus, we assume

$[1] \quad \Gamma_{imp} \, cov \, \Gamma_{sp}$

Requirement (1) (see Definition of sat) is clearly satisfied, since it coincides with (C1). Now

$[1] \Rightarrow \pi(\Gamma_{imp} \parallel e \Gamma_{sp}) = \pi(\Gamma_{sp}) \quad \text{[by (C2)]}$

$[2] \Rightarrow \beta(P(\Gamma_{imp} \parallel P(e \Gamma_{sp})) = \pi(\Gamma_{sp}) \quad \text{[by Theorem 1(a)]}$

$[3] \Rightarrow \pi(\Gamma_{sp}) \subseteq \pi(\Gamma_{imp}) \quad \text{[since } \Sigma(\Gamma_{imp}) = \Sigma(\Gamma_{sp}) \text{]}$.

Thus Requirement (2) is met.

Considering Requirement (3), we assume (further to [1])

$[4] \quad s; w \in \pi(\Gamma_{imp})$

where $[5] \quad s \in \pi(\Gamma_{sp})$

and $[6] \quad w \in out \, \Gamma_{sp}$.

$[5] \Rightarrow [7] \quad s \in \pi(e \Gamma_{sp}) \quad \text{[since } \pi(e \Gamma) \subseteq \pi(e \Gamma)]$

In view of [6], [7] and the definition of $e \Gamma$ we have

$[8] \quad s; w \in \pi(e \Gamma_{sp});$

Clearly, $\Sigma(e \Gamma_{sp}) = \Sigma(\Gamma_{sp}) = \Sigma(\Gamma_{imp})$. Thus, by [2],[4], and [8] we get

$[9] \quad s; w \in \pi(\Gamma_{sp}).$

It follows that Requirement (3) is also satisfied. This completes the proof of Part (A).

(B) We now turn to proving

$$\Gamma_{imp} \, sat \, \Gamma_{sp} \Rightarrow \Gamma_{imp} \, cov \, \Gamma_{sp},$$

where $\Gamma_{sp}$ is assumed to be well-formed. Thus, we assume

$[1] \quad \Gamma_{imp} \, sat \, \Gamma_{sp}$

Condition (C1) is evidently met. We now proceed to prove Condition (C2).
Furthermore, \( \pi(\Gamma_{sp}) \subseteq \pi(\Gamma_{imp}) \) [generally \( \pi(\Gamma) \subseteq \pi(e\Gamma) \)].

Thus, \( \pi(\Gamma_{sp}) \subseteq \pi(\Gamma_{imp}) \cap \pi(e\Gamma_{sp}) \).

Now, \( \pi(\Gamma_{imp} \parallel e\Gamma_{sp}) = \pi(\Gamma_{imp}) \cap \pi(e\Gamma_{sp}) \),

in view of Theorem 1(a), since \( \Sigma(\Gamma_{imp}) = \Sigma(\Gamma_{sp}) = \Sigma(e\Gamma_{sp}) \). It follows that

Now, \( \pi(\Gamma_{imp} \parallel e\Gamma_{sp}) = \pi(\Gamma_{imp}) \cap \pi(e\Gamma_{sp}) \).

in view of Theorem 1(a), since \( \Sigma(\Gamma_{imp}) = \Sigma(\Gamma_{sp}) = \Sigma(e\Gamma_{sp}) \). It follows that

Next, we assume that

\[ w \in \pi(\Gamma_{imp}) \]

and

\[ w \in \pi(e\Gamma_{sp}). \]

We shall use induction on the length of \( w \) to show that

\[ w \in \pi(\Gamma_{sp}). \]

The case \( \text{length}(w) = 0 \) is trivial.

Let now \( w = s; x \), where \( \text{length}(s) = n \geq 0 \) and \( x \subseteq \Sigma(\Gamma_{sp}) \).

By the induction hypothesis, \( s \in \pi(\Gamma_{sp}) \). We distinguish three cases.

Case 1: \( x \subseteq \text{out}\Gamma_{sp} \).

In this case, we have

\[ s; x \in \pi(\Gamma_{imp}) \] (by [4])

\[ s \in \pi(\Gamma_{sp}) \]

\[ x \subseteq \text{out}\Gamma_{sp}; \]

Thus, Requirement (3) applies. Hence

\[ s; x \in \pi(\Gamma_{sp}). \]

Case 2: \( x \subseteq \text{in}\Gamma_{sp} \).

As stated above, we have \( s \in \pi(\Gamma_{sp}) \). Let the net obtained by applying \( s \) to \( \Gamma_{sp} \) be \( \Gamma'_{sp} \). In view of [5], \( s \) is also applicable to \( e\Gamma_{sp} \). Evidently, the net reached by applying \( s \) to \( e\Gamma_{sp} \) is \( e\Gamma'_{sp} \).

By [5], \( x \) is applicable to \( e\Gamma'_{sp} \). In view of the definition of \( e\Gamma \), input transitions enabled in \( e\Gamma \),
are also enabled in $\Gamma$. Thus $x$ is also applicable to $\Gamma_{sp}$. It follows that

$$s; x \in \pi(\Gamma_{sp})$$

also holds in Case 2.

**Case 3:** $x \cap in_{sp} \neq \emptyset$

$$x \cap out_{sp} \neq \emptyset$$

By [4] and [5] we have

$$[7] \quad s; x \in \pi(\Gamma_{imp}) \cap \pi(\Gamma_{sp})$$

Let $xin = x \cap in_{sp}$ and $xout = x \cap out_{sp}$.

Evidently, by [7],

$$s; xin; xout \in \pi(\Gamma_{imp}) \cap \pi(\Gamma_{sp})$$

and

$$s; xout; xin \in \pi(\Gamma_{imp}) \cap \pi(\Gamma_{sp})$$

In view of Cases 1 and 2,

$$s; xin; xout \in \pi(\Gamma_{sp})$$

and

$$s; xout; xin \in \pi(\Gamma_{sp})$$

Since $\Gamma_{sp}$ is well-formed,

$$s; x \in \pi(\Gamma_{sp})$$

Thus, [6] is proven for all 3 cases.

Hence, in view of [2], [4], [5], and [6], we have

$$[8] \quad \pi(\Gamma_{imp} \parallel e_{\Gamma_{sp}}) \subseteq \pi(\Gamma_{sp})$$

and by [3] and [8],

$$\pi(\Gamma_{imp} \parallel e_{\Gamma_{sp}}) = \pi(\Gamma_{sp})$$

Thus Condition (C2) holds.

This completes the proof of Part (B), and of Theorem 2.
7. VERIFICATION EXAMPLES

In this section $\Gamma_{imp}$ will denote a circuit net which represents the behavior of a digital circuit $CCT$ with $q$ as its initial state, i.e. $\Gamma_{imp} = \text{net}(CCT, q)$. $\Gamma_{sp}$ will be a well-formed circuit net representing a circuit specification for $(CCT, q)$.

We verify $\Gamma_{imp}$ with respect to $\Gamma_{sp}$, by proving that one of the following relations holds:

$\Gamma_{imp} \text{ sat } \Gamma_{sp}$ or $\Gamma_{imp} \text{ cov } \Gamma_{sp}$.

We shall illustrate that the formal concepts sat and cov suit our intuitive understanding. These examples complement the examples which have been presented in Section 5.

In the following examples we shall represent any environment restriction net $e$ in a reduced form obtained by omitting redundant places (i.e. places without outgoing edges).

Example 7.1: Let $\Gamma_{sp}$ and $\Gamma_{imp}$ be the circuit nets shown in Fig. 7.1(a) and 7.1(b). Intuitively we don't accept $\Gamma_{imp}$ as an implementation of $\Gamma_{sp}$, because $\Gamma_{imp}$ contains a nondeterministic input-output behavior, which contradicts the specification. If we use the 'sat' relation we can see that requirements (1) and (2) hold, whereas requirement (3) does not hold. Indeed

$$ az \in \pi(\Gamma_{imp}) $$

$$ a \in \pi(\Gamma_{sp}) $$

$$ z \in \text{out} \Gamma_{sp} $$

but

$$ az \notin \pi(\Gamma_{sp}). $$

Alternatively, if we use the 'cov' relation we can easily see that requirement (C1) holds, whereas (C2) does not hold. ($e \Gamma_{sp}$ is shown in Fig. 7.1(c).) Indeed, $\pi(e \Gamma_{sp} \parallel \Gamma_{imp}) \neq \pi(\Gamma_{sp})$ since

$$ az \in \pi(e \Gamma_{sp} \parallel \Gamma_{imp}) $$

whereas

$$ az \notin \pi(\Gamma_{sp}). $$

Example 7.2: Given two specification nets $\Gamma_{sp}$, and $\Gamma_{sp'}$ and an implementation net $\Gamma_{imp}$, in Fig. 7.2. We wish to show that $\Gamma_{imp}$ satisfies $\Gamma_{sp}$, but does not satisfy $\Gamma_{sp'}$. This example points out that if the order in which the input signals appear is important then this fact must be expressed explicitly in the specification. A correct implementation ensures that for every signal
sequence that is contained in the specification, the required result will be achieved; otherwise -
the behavior of the implementation is not guaranteed. In this example $\Gamma_{sp_2}$ specifies that sequence
$<b,a>$ does not produce any output, whereas $\Gamma_{sp_1}$ does not require anything about the sequence
$<b,a>$.

Now we shall prove that $\Gamma_{imp} \text{ cov } \Gamma_{sp_1}$.

$\Sigma(\Gamma_{sp_2}) = \Sigma(\Gamma_{imp})$ holds (requirement (C1)). Next, we compose $e \Gamma_{sp_1}$ (shown in Fig. 7.3(a)) with
$\Gamma_{imp}$ (shown in Fig. 7.2(c), to obtain $\Gamma_{comp}$ (shown in Fig. 7.3(b)). After omitting redundant
edges from $\Gamma_{comp}$ (cf. [Yo 87]) we get $\Gamma_{sp_2}$. It follows that requirement (C2) is also satisfied. This
completes the proof of the above statement.

Now we shall show that the statement $\Gamma_{imp} \text{ sat } \Gamma_{sp_2}$ does not hold.

Requirements (1) and (2) (see definition of sat) are clearly satisfied, but the following contradicts
requirement (3).

$\text{baz } \in \pi(\Gamma_{imp})$
$\text{ba } \in \pi(\Gamma_{sp_2})$
$z \in \text{out } \Gamma_{sp_2}$

whereas
$\text{baz } \notin \pi(\Gamma_{sp_2})$.

8. CONCLUSIONS

We have introduced a novel definition of the concept 'implementation satisfies
specification' with regard to digital circuits, and have developed a practical approach to circuit
verification. Note that any net $\Gamma_{imp}$ representing an actual digital circuit is bounded [Pe81], and
therefore $\pi(\Gamma_{imp})$ is a regular language over $\mathcal{L}(\Gamma_{imp})$. Similarly $\pi(\Gamma_{sp})$ is also regular. Hence the
statement $\Gamma_{imp} \text{ sat } \Gamma_{sp}$ can always be verified using the theory of regular languages. On the
other hand, $\Gamma_{imp} \text{ cov } \Gamma_{sp}$ can be verified using net reduction methods, cf [Yo87] and [Be86]. In
actual applications one of the two methods will usually be preferable over the other.
We have also represented a method for deriving the behavior of a composite circuit from the behaviors of its components. This method facilitates the construction of the implementation net for a modular circuit composed of known basic components.

The research reported in this paper has been expanded towards a synthesis methodology for delay-insensitive modular circuits, [Re88]. [Re88] also contains additional results which facilitate circuit verification.
REFERENCES

LNCS = Lecture Notes in Computer Science, Springer Verlag.


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. [Pe81],[Rei85]). The following summary is mainly based on [YE 83],[RV 83].

A (Petri) net consists of

1. a finite set $P$ 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. [YE 83],[RV 83]). 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 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 [YE 83] 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 \( \Sigma = 2^\Sigma - \{ \emptyset \} \).

For any nonempty subset \( U \) of \( T \), let
\[
\eta(U) = \begin{cases} 
\{ \eta(t) \mid t \in U \} = \{ \lambda \} & \text{then } \Lambda \\
\{ \eta(t) \mid t \in U \} \cap \Sigma & \text{else}
\end{cases}
\]

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. 3.1 - (a) labeled net $\Gamma$, (b) labeled net $\Gamma'$ obtained from $\Gamma$ by transition splitting

Fig. 3.2 - (a) $\Gamma_1$, (b) $\Gamma_2$, (c) $\Gamma_1 \parallel \Gamma_2$
Fig. 5.1 - (a) labeled net $\Gamma_1$, (b) 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. 7.1 - (a) $\Gamma_{sp}$, (b) $\Gamma_{imp}$, (c) $e\Gamma_{sp}$
Fig. 7.2 - (a) $\Gamma_{sp_1}$, (b) $\Gamma_{sp_2}$, (c) $\Gamma_{imp}$

Fig. 7.3 - (a) $e \Gamma_{sp_1}$, (b) $\Gamma_{comp}$