Proof

We define a relation $S$ between $(\text{env.SPnet} \parallel \text{IMPL.net})$ and $\text{SPEC.net}$ as follows.

1. $(M_0(\text{env.SPnet}), M_0(\text{IMPL.net})) S M_0(\text{SPEC.net})$

2. Let $w' \in L(\text{SPEC.net})$. Then $w' \in L(\text{env.SPnet})$.

By Condition (3), there exists $w'' \in L(\text{IMPL.net})$ such that $w'' = w'$.

Now let $M_0(\text{SPEC.net})[w' > M_1$, $M_0(\text{SPnet})[w' > M_2$, and $M_0(\text{IMPL.net})[w" > M_3$.

Then (and only then) $(M_2, M_3) S M_1$.

Assume now $M_2[u' > N_2$ and $M_1[u' > N_1$. Then $w';u' \in L(\text{SPEC.net})$ and $w';u' \in L(\text{env.SPnet})$. By Condition (5), there exists $u'$ such that $w'';u'' \in L(\text{IMPL.net})$, where $u'' = u'$.

Now, let $M_0(\text{IMPL.net})[w'';u'' > N_3$. Thus $M_3[u'" > N_3$. In view of the definition of $S$, we have $(N_2, N_3) S N_3$. It follows that $S$ is the required observation equivalence. Thus Theorem 6.2 has been proven.
Proof of (4)

Let \( w \in L(SPEC.net) \). It follows that \( w'' \in L(env.SPnet) \) with \( w'' \subseteq w \). This implies, in view of the Definition of \( env.SPnet \), that \( w''z \in L(env.SPnet) \).

In accordance with (4), we also have \( w';z \in L(IMPL.net) \).

Evidently, \( w';z = w''z = w'' \).

Now, let \( u=(u1,u2) \), where \( M0(env.SPnet)[w''u1 \text{ and } M0(IMPL.net)[w'u2} \).

Then \( u \) is \( S \)-related to \( u3 \), where \( M0(SPEC.net)[w''u3 \text{ with } w''' = w} \).

Now \( z \) is applicable to \( u \). It follows that \( \lambda';z;\lambda'' \) (where \( \lambda' \) and \( \lambda'' \) denote finite powers of \( \lambda \)) is applicable to \( u3 \).

Hence, \( w;z = w''';\lambda';z;\lambda'' \in L(SPEC.net) \), as required by (4).

Proof of (5)

\( w1;w2 \in L(SPEC.net) \Rightarrow w1 \in L(SPEC.net) \Rightarrow \)

\( w1'' \in L(SPEC.net) \) (where \( w1'' = w1) \Rightarrow w1'' \in L(env.SPnet) \).

This fact, together with \( w1' \in L(IMPL.net) \), implies that

\( u=(u1,u2) \), where \( M0(env.SPnet)[w1''u1 \text{ and } M0(IMPL.net)[w1'u2} \), is a reachable state of \( A \), \( S \)-related with \( u3 \), where \( M0(SPEC.net)[w1''u3} \).

Now, \( w2'' \) is applicable to \( u1 \), where \( w2'' = w2 \).

It follows, in view of the observation equivalence \( S \), that there exists \( w2''' \) with \( w2''' = w2'' = w2 \), which is applicable to \( u2 \).

Hence, \( w1';w2''' \in L(IMPL.net) \), as required by (5).

Thus, Theorem 6.1 is proven.

Theorem 6.2

Conditions (3) and (5) of Definition 2.1 imply

\( IMPL.net \text{ SAT } SPEC.net, \text{ i.e., } (env.SPnet \mid \mid IMPL.net) \text{ obs.-equiv. } SPEC.net \)
Proof of Theorems 6.1 and 6.2

Theorem 6.1

Assume IMPL.net SAT SPEC.net. Then Conditions (3), (4), and (5) of Definition 2.1 are satisfied.

Proof

By definition, IMPL.net SAT SPEC.net implies:

\( \text{env.SPnet} \parallel \text{IMPL.net} = A \) obs.equiv \( \text{SPEC.net} = B \)

It follows that there exists a relation \( S \) with domain(\( S \))=R(\( A \)), and range(\( S \))=R(\( B \)), where R(\( A \)) and R(\( B \)) denote the reachable markings of \( A \) and \( B \), respectively. Moreover, \( S \) satisfies the conditions (1)-(4) of Definition 5.1.

By Definition 4.3 the set of reachable markings of \( \text{LPN1} \parallel \text{LPN2} \) can be represented by the set of all pairs \((M_1, M_2)\), where \( M_0(\text{LPN1})[a_1 > M_1, \text{ and } M_0(\text{LPN2})[a_2 > M_2, \) provided \( a_1 \neq a_2 \) (\( a \) is obtained from the word \( a \) by omitting all the instances of \( \lambda \) in \( a \)).

Now, A obs.equiv B implies

\( (x,y)Sz \Rightarrow (x',y')Sz', \) where \( x[a_1 > x', y[a_2 > y', z[a_3 > z', \text{ and } a_1 \neq a_2 \neq a_3. \)

Our final observation, before we turn to proving (3), (4), and (5), is the following:

\( L(\text{SPEC.net}) \subseteq L(\text{env.SPnet}) \).

Proof of (3)

Let \( w \in L(\text{SPEC.net}) \). It follows that \( w \) is applicable to \( M_0(B) \). Now

\( M_0(A) S M_0(B) \) implies the existence of \( w' \) and \( w'' \), such that \( w' \parallel w'' \parallel w \) and \( w' \) is applicable to \( M_0(\text{env.SPnet}) \) and \( w'' \) is applicable to \( M_0(\text{IMPL.net}) \).

Therefore, \( w'' \in L(\text{IMPL.net}) \); hence, \( w \parallel w'' \in L(\text{IMPL.net}) \). This proves (3).
of Speed-Independent Circuits,

in: [Asynch97], pp.254-265.


[SY95] A.Semenov and A.Yakovlev, Combining Partial Orders and Symbolic Traversal for Efficient Verification of Asynchronous Circuits,


[Yoe90] M.Yoeli, Net-Based Synthesis of Delay-Insensitive Circuits,


[YY96] T.Yoneda and T.Yoshikawa, Using Partial Orders for Trace-Theoretic Verification of Asynchronous Circuits,
in: [Asynch96], pp.152-163.


[Sem97] A.Semenov et al., Partial Order Based Approach to Synthesis
References

[Asynch96] Second International Symposium on
Advanced Research in Asynchronous Circuits and Systems,
March 1996, Aizu-Wakamatsu, Japan

[Asynch97] Third International Symposium on
Advanced Research in Asynchronous Circuits and Systems,
April 1997, Eindhoven, The Netherlands

[BE97] I.Benko and J.Ebergen, Unraveling the Petri-Net Weave,

[BS95] J.A.Brzozowski and C.-J. H.Seger, Asynchronous Circuits,
Springer Verlag, 1995.

[Dill88] Trace Theory for Automatic Hierarchical Verification

[HY92] I.Honma and T.Yoneda, Automatic Verification Method of
Asynchronous Circuits using Petri Nets,

[JL94] G. de Jong and B.Lin, A Communicating Petri Net Model for the

[LSV93] L.Lavagno and A.Sangiovanni-Vincentelli,
Algorithms for Synthesis and Testing of Asynchronous Circuits,

[McM95] K.L.McMillan, Trace-theoretic Verification of Asynchronous
Assume IMPL.net SAT SPEC.net.

Then Conditions (3), (4), and (5) of Definition 2.1 are satisfied.

**Theorem 6.2**

Conditions (3) and (5) of Definition 2.1 imply IMPL.net SAT SPEC.net.

The above two theorems are proven in Appendix A.

It follows that the verification process may essentially be based on a proof of the SAT relation and of Condition (6) of Definition 2.1.

In follow-up reports we illustrate the partial automation of this verification process, using Petrify (see Ref. [PETRIFY]).
5. Observation-Equivalent Nets

Let $LPN$ and $LPN'$ be two labelled Petri nets, having the same set of labels, namely $\Sigma \cup \{\lambda\}$. Here, $\lambda$ is a special symbol, not in $\Sigma$ (similar to $\tau$ in CCS or $i$ in LOTOS).

Let $a \in \Sigma$. We denote by ext$(a)$ the set $\lambda^* ; a ; \lambda^*$. Furthermore, we denote by $R(LP)$ and $R(LP')$ the sets of reachable markings of $LPN$ and $LPN'$, respectively.

**Definition 5.1**

A relation $S$ between $R(LP)$ and $R(LP')$ is an observational equivalence between $LPN$ and $LPN'$ iff the following conditions are satisfied.

1. Every marking in $R(LP)$ is $S$-related to at least one marking in $R(LP')$, i.e., $\text{dom} S = R(LP)$.
2. Every marking in $R(LP')$ is $S$-related to at least one marking in $R(LP)$, i.e., $\text{ran} S = R(LP')$.
3. Assume the marking $M_1$ in $R(LP)$ is $S$-related to the marking $M_1'$ in $R(LP')$.
   a) Furthermore, assume $M_1 e(a) > M_2$ in $LPN$, where $e(a) \in \text{ext}(a)$.
      Then there exists $e'(a) \in \text{ext}(a)$ and $M_2' \in R(LP')$, such that $M_1' e'(a) > M_2'$, and $M_2$ and $M_2'$ are $S$-related.
   b) Similarly, assume $M_1' e'(a) > M_2'$ in $LPN'$, where $e'(a) \in \text{ext}(a)$. Then there exists $e(a) \in \text{ext}(a)$ and $M_2 \in R(LP)$, such that $M_1 e(a) > M_2$, and $M_2$ and $M_2'$ are $S$-related.

6. Verification Approach

The above SAT relation is applicable, together with suitable Petri-net based tools, to the efficient, formal verification of modular, asynchronous circuits. This approach is based on the following theorems.

**Theorem 6.1**
merging into a single transition each pair of transitions of the two nets having the same label.

### 4.2 Multiply Labelled Nets

To handle the general case, we first need the concept of "transition splitting".

**Definition 4.2**

Let $t$ be a transition of a labelled net. By *splitting transition $t$ into $k$ parts* we refer to the replacement of $t$ by $k$ new transitions. The input and output places of the new transitions are the same as those of the replaced transition $t$.

**Definition 4.3**

Let $LPN_1$ and $LPN_2$ be labelled nets with $m \geq 1$ and $n \geq 1$ appearances of the label 'a', respectively. The following procedure describes how such a situation is handled, when generating the parallel composition $LPN_1 | | LPN_2$. To simplify our notation, let $m=2$ and $n=3$.

The required steps are as follows.

1. relabel the 2 a-labels of net $LPN_1$ as a/1 and a/2.
2. relabel the 3 a-labels of net $LPN_2$ as a/1, a/2, and a/3.
3. split each transition labelled a/i (i=1,2) of $LPN_1$ into 3 new transitions, labelled a/i/1, a/i/2, and a/i/3.
4. split each transition labelled a/j (j=1,2,3) of $LPN_2$ into 2 new transitions, labelled a/1/j and a/2/j.
5. merge each pair of equally labelled transitions of $LPN_1$ and $LPN_2$, obtained by (3) and (4).
6. relabel each merged pair obtained by (5) into 'a'.
Definition 3.1

Let SPEC.net be as described above. Its *Environment-Restrictions Net* env.SPnet is obtained from SPEC.net by omitting all the edges incoming to output transitions, i.e., transitions labelled by output labels.

Definition 3.2

Let IMPL.net, SPEC.net, and env.SPnet be as indicated above. Then

IMPL.net SAT SPEC.net

iff

(env.SPnet || IMPL.net) obs.equiv SPEC.net

Here, '||' denotes the parallel composition operator of nets, as discussed in the sequel, 'obs.equiv' refers to observational equivalence. The usual definition of observational equivalence is easily applied to Petri nets (see below).

4. Parallel Composition of Nets

4.1 Uniquely Labelled Nets

In this section we deal with the parallel composition of "uniquely labelled" nets. Such a net is a labelled net, where each label appears only once. Our approach is based on [RY88], where the general case of labelled nets is treated extensively (see also [JL94] and [BE97]).

Definition 4.1

Let LPN1 and LPN2 be ordinary uniquely labelled Petri nets with labelling alphabets \( \Sigma_1 \) and \( \Sigma_2 \), respectively (generally, \( \Sigma_1 \) and \( \Sigma_2 \) are not disjoint).

The *parallel composition* \( \text{LPN1} \parallel \text{LPN2} \) is the net obtained from LPN1 and LPN2 by...
**Definition 2.1**

We say that IMPN is a *realization* of SPN (Notation: IMPN|\(\models\)SPN ), iff the following conditions are satisfied:

Note: we use ‘;’ to denote concatenation.

1. inSPN = inIMPN = in
2. outSPN = outIMPN = out
3. \(L(SP N) \subseteq L(IMPN)\)
4. assume \(w \in L(SP N)\), \(z \in L(SP N)\), \(w' \in L(IMPN)\), and \(w'=w\), where \(w'\) is the restriction of \(w'\) to \(in \cup out\).

Then \(w; z \in L(SP N)\).
5. assume \(w_1; w_2 \in L(SP N)\), and \(w_1' \models w_1\), for some \(w_1' \in L(IMPN)\).

Then there exists \(w_2'\) such that \(w_1'; w_2' \in L(IMPN)\), and \(w_2' \models w_2\).
6. let \(w \in L(SP N)\), \(w' \in L(IMPN)\) and \(w' \models w\).

Then there exists a positive integer \(k\) such that for any word \(w'' \in (intIMPN)^*\), \(w' ; w'' \in L(IMPN)\) implies \(\text{length}(w'') < k\).

For an informal motivation of the above requirements, see [YG98].

3. The SAT Relation

We now introduce an alternative relation between SPEC.net and IMPL.net, namely the “SAT Relation”. We then discuss the applicability of the SAT relation to the net-based verification of asynchronous circuits.

We denote the part of SPEC.net representing environment restrictions by the net env.SPnet. A method of deriving env.SPnet from SPEC.net is described next (cf. [RY88]).
Ref.[PETRIFY]). Thus our approach facilitates the partially automatic verification of a
given modular, asynchronous circuit.

1.3 The Basic Problem

We assume that we are given a specification SPEC.net of an asynchronous circuit in
the form of a labelled Petri net. The labelling alphabet of SPEC.net is partitioned into
three parts: input labels, output labels, and internal (dummy) labels.

Similarly, the behaviour of a relevant implementation is represented by a labelled Petri
net IMPL.net, with its labelling alphabet also partitioned into the above three parts.

All the labels involved represent level transitions. In our approach the same label
represents both an up-transition (i.e., a change from logic-level 0 to 1) at the
corresponding port, as well as a down- transition at this port.

As usual, we assume that SPEC.net specifies the required behaviour of the circuit, as
well as the restrictions imposed on the circuit environment. Such restrictions are essen-
tial to ensure that the implementation indeed performs properly, as required.

2. The Concept of Realization

Let SPEC.net and IMPL.net be as above. The following definition is intended to for-
malize the intuitive concept "IMPL.net is a realization of SPEC.net" (cf. [YG98]).

Let SPN be a SPEC.net, and IMPN an IMPL.net, as described above. We denote by
inSPN, outSPN, and intSPN the sets of input, output, and internal labels of SPN,
respectively. The corresponding notation is also applied to IMPN.

Both nets, SPN and IMPN are assumed to be deterministic. We denote by L(SPN) the
language of SPN (every reachable marking is viewed as an accepting state), and by
L(SPN)\ the language L(SPN), restricted to inSPN\cup outSPN. L(IMPN) and L(IMPN)\are defined similarly.
1. Introduction

In [YG98] we discussed a LOTOS-based approach to the formal verification of asynchronous circuits. In this publication we develop an alternative approach to the above problem, based on labelled Petri nets [Mur89].

1.1 Related Work

Presently there exists an extensive literature, dealing with the application of labelled Petri nets, and related concepts, to the synthesis and verification of asynchronous circuits. A selection of such papers is listed in [Ref]. We assume familiarity with the advantages of using Petri nets and similar concepts for the purpose of formal verification of asynchronous circuits.

1.2 Outline of Our Work

The research reported in this paper has a number of outstanding features not available in recent publications (see [Ref]). The major points of interest are the following.

(1) We provide a mathematically precise framework, within which the relationship between specification and realization of asynchronous circuits is precisely formulated. We assume the realization to be composed of basic components (modules), which are in general more powerful than simple gates (cf. [Sut89]).

(2) We assume that the realization may be more "powerful" than the specification (cf. [Dill88]).

(3) We assume that the realization may behave non-deterministically. Hence we are concerned with comparing circuits and specifications with respect to observational equivalence, rather than trace equivalence (the latter is the widely accepted viewpoint of most publications! cf. [BE97]).

(4) We relate our verification approach to available net tools, particularly Petrify (see...
Abstract:

The paper introduces a precise formulation of the relationship between specification and realization of asynchronous circuits.

It demonstrates the suitability of Petri nets to the verification of modular, asynchronous circuits.