

# Partial Order Reduction for Detecting Safety and Timing Failures of Timed Circuits

Denduang Pradubsuwun<sup>1</sup>, Tomohiro Yoneda<sup>\*2</sup>, and Chris Myers<sup>\*\*3</sup>

<sup>1</sup> Tokyo Institute of Technology  
denduang@yt.cs.titech.ac.jp

<sup>2</sup> National Institute of Informatics  
yoneda@nii.ac.jp

<sup>3</sup> University of Utah  
myers@ece.utah.edu

**Abstract.** This paper proposes a partial order reduction algorithm for timed trace theoretic verification in order to detect both safety failures and timing failures of timed circuits efficiently. This algorithm is based on the framework of timed trace theoretic verification according to the original untimed trace theory. Consequently, its conformance checking supports hierarchical verification. Experimenting with the STARI circuits, the proposed approach shows its effectiveness.

## 1 Introduction

Nowadays, the role of timed circuits have rapidly arisen in integrated digital circuit design. Thus, the verification of such timed circuits is imperative. But, the cost of timing verification is quite high. Several approaches have been proposed in order to reduce the average complexity of verification, i.e. symbolic methods based on BDDs and partial order reduction. Symbolic methods are difficult to efficiently apply to timing verification, yet partial order reduction is one of the most promising solution to the state explosion problem, e.g. [1–4]. Hence, verification methods in which partial order reduction is well-suited are preferred.

One direction is a timing analysis algorithm to validate correctness in the level-ruled Petri net [5]. Another direction is the simple timed trace theory based on timed Petri net [6]. The partial order reduction is applied to both of them, but they are unable to hierarchically perform verification by using the conformation relation.

In addition, [6] has no ability to verify liveness property, i.e. only safety failures are detected. A framework of timed trace theoretic verification based on pseudo failure is proposed in [7], in which not only safety failures but also timing failures are examined. The timing failure is a restricted form of violation of liveness property. Practically, detecting timing failure is adequate to verify timed circuits. Moreover, even if this approach certainly supports hierarchical structure as the original trace theory [8], it still differs from the original one in many points. Eventually, the framework of timed

---

\* This research is supported by JSPS Joint Research Projects.

\*\* This research is supported by SRC contract 2002-TJ-1024, and NSF Japan Program award INT-0087281.

trace theoretic verification in accordance with the original one as well as the concept of semimodule and semimirror allowing conformance checking has been proposed in [9].

In this paper, we propose a partial order reduction algorithm for the verification method of the latest framework, and show its effectiveness through experimental results.

## 2 Timed Trace Theoretic Verification

Since our algorithm relies on the framework of timed trace theoretic verification proposed in [9], we briefly recall its idea in this section. The important notions are a module, a semimodule, and a timed trace structure.

A *module* is a tuple  $(I, O, N, \text{wire})$ , where  $I$  is a set of input wires,  $O$  is a set of output wires,  $N = (P, T, F, lb, ub, \mu^0)$  is a time Petri net, and  $\text{wire} : T \rightarrow I \cup O$  be a function from a set of transitions to a set of wires. We say that if  $\text{wire}(t) \in I$ ,  $t$  is an input transition, and otherwise,  $t$  is an output transition. In order to simplify the analysis algorithm, we assume that for each  $k \in O$ , there exists at most one output transition  $t$  such that  $\text{wire}(t) = k$  in a module.  $k(out)$  is used to represent such a transition  $t$  especially in the figures for simplicity. On the other hand, we allow multiple input transitions for an input wire  $k$ . Thus,  $k(in)_1, k(in)_2, \dots$  are used for such input transitions, while  $k(in)$  is used in cases that there exists only one corresponding input transition. Figure 1 shows examples of modules. A time Petri net is a Petri net except that each transition  $t$  of a time Petri net has two non-negative rationals, the earliest firing time  $lb(t)$  and the latest firing time  $ub(t)$ , and that each enabled transition must fire within this time bound. A *timed run*  $\rho = \sigma_0 \xrightarrow{t_1} \sigma_1 \xrightarrow{t_2} \sigma_2 \xrightarrow{t_3} \dots$  of  $N$  is a finite or infinite sequence of states and transitions such that  $\sigma_0$  is the initial state, and  $\sigma_{i+1}$  is obtained from  $\sigma_i$  by passing some time and then firing transition  $t_{i+1}$ . Its corresponding *timed trace* is  $e_1 e_2 e_3 \dots$ , where  $e_i = (\text{wire}(t_i), \tau_i)$  is an *event* and  $\tau_i$  denotes the time when the transition  $t_i$  fires.  $\text{trace}(N)$  be a set of all timed traces generated by  $N$ .

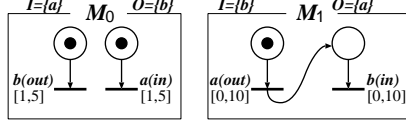
A *semimodule* is the same as a module, but the corresponding timed trace structure is distinctively defined, which is shown later.

A *timed trace structure* of a module  $M = (I, O, N, \text{wire})$ , denoted by  $\mathcal{T}(M)$ , is a tuple  $(I, O, S, F)$ , where  $S$  and  $F$  are sets of timed traces defined as follows.  $S$ , which is called a *success trace set*, is equal to  $\text{trace}(N)$ .  $F$ , which is called a *failure trace set*, contains a trace  $y(w, \tau) \notin S$ , iff either

- $y \in F$ , or
- $y \in S, \tau \leq \text{TL}(y, N), w \in I$ , or
- $y \in S, \tau > \text{TL}(y, N), \text{limit}(y, N) \subseteq I$ .

The first condition is easy, i.e. any extension of a failure trace is also a failure trace.  $\text{TL}(y, N)$  is the latest time until when the firings of all enabled transitions in  $N$  can be postponed after  $y$ . For example, in Figure 1,  $\text{TL}(\varepsilon, N_0)$  is 5 in  $M_0$ , and  $\text{TL}((a, 5), N_1)$  is 15 in  $M_1$ .

The second condition above states the case that the net can reach the time  $\tau$  with  $\tau \leq \text{TL}(y, N)$ , but cannot accept an input. It is considered as a failure. Note that if  $w$  is an output in this case, then  $y(w, \tau)$  is neither a success nor a failure, i.e. it is called



**Fig. 1.** Modules that cause a timing failure.

*not possible*. This difference comes from the fact that a net can control producing or not producing an output, but cannot control an input.

$\text{limit}(y, N)$  is the set of wires that correspond to the enabled transitions which determine  $\text{TL}(y, N)$ . In the example above,  $\text{limit}(\varepsilon, N_0) = \{a, b\}$ , because  $\text{TL}(\varepsilon, N_0) = 5$  and both transitions determine this time. In the third condition,  $\tau > \text{TL}(y, N)$  implies that some transition causes a time-out from  $\text{limit}(y, N) \subseteq I$ , because an input cannot be controlled by the net. Hence, this must be a failure. Again, if  $\text{limit}(y, N)$  contains an output wire, then  $y(w, \tau)$  is not possible, because an output is controlled by the net, and so, the net never reaches this time point. These considerations are summarized as follows, where  $P = S \cup F$  is a set of traces that are possible.

- C1:** For  $w \in I \cup O$ , and  $\tau \in \mathbf{R}^+$  (the set of non-negative real numbers),
1.  $S = \text{trace}(N)$ ,
  2. for  $y \in S$  and  $y(w, \tau) \notin S$ ,
    - (a) if  $\tau \leq \text{TL}(y, N)$ , then
      - i. if  $w \in I$ , then  $y(w, \tau) \in F$
      - ii. else  $y(w, \tau) \notin P$
    - (b) else if  $\text{limit}(y, N) \subseteq I$ , then  $y(w, \tau) \in F$
    - (c) else  $y(w, \tau) \notin P$
  3. for  $y \in F$ ,  $y(w, \tau) \in F$ ,
  4. for  $y \notin P$ ,  $y(w, \tau) \notin P$ .

In order to define the correctness between modules, we use notions given in [8]. For  $\mathcal{T}_1 = (I_1, O_1, S_1, F_1)$  and  $\mathcal{T}_2 = (I_2, O_2, S_2, F_2)$  such that  $I_1 \cup O_1 = I_2 \cup O_2$ , the *intersection* of  $\mathcal{T}_1$  and  $\mathcal{T}_2$ , denoted by  $\mathcal{T}_1 \cap \mathcal{T}_2$ , is a timed trace structure  $(I_1 \cap I_2, O_1 \cup O_2, S_1 \cap S_2, (F_1 \cap F_2) \cup (F_1 \cap P_2))$ . Note that a trace is a failure of the intersection, only when it is a failure of one module as well as it is possible for the other. For considering the similar notion for modules with different wire sets, the following notions are needed. For a timed trace  $x = e_1 e_2 e_3 \dots$  and a set  $D$  of wires,

$$\text{del}(D, x) = \begin{cases} y & \text{if } e_1 = (w_1, \tau_1), w_1 \in D \\ e_1 y & \text{else,} \end{cases}$$

defines a timed trace obtained by projecting out the events of wires in  $D$ , where  $y = \text{del}(D, e_2 e_3 \dots)$ . If  $x = \epsilon$ , then  $\text{del}(D, x) = \{\epsilon\}$ . For a set  $X$  of timed traces,  $\text{del}^{-1}(D, X)$  is the set  $\{x \mid \text{del}(D, x) \in X\}$ . For  $X$  whose elements do not contain any wires in  $D$ ,  $\text{del}^{-1}(D, X)$  is the set of all timed traces that can be generated by inserting any event of  $D$  between any consecutive events in traces of  $X$ . This is extended for a timed trace structure  $\mathcal{T} = (I, O, S, F)$  such that  $\text{del}^{-1}(D, \mathcal{T}) = (I \cup D, O, \text{del}^{-1}(D, S), \text{del}^{-1}(D,$

$F$ ). Note that the inserted wires are always considered to be the inputs. The *composition* of  $\mathcal{T}_1$  and  $\mathcal{T}_2$  is defined as  $\mathcal{T}_1 || \mathcal{T}_2 = \text{del}^{-1}(D_2, \mathcal{T}_1) \cap \text{del}^{-1}(D_1, \mathcal{T}_2)$ , where  $D_1 = (I_1 \cup O_1) - (I_2 \cup O_2)$  and  $D_2 = (I_2 \cup O_2) - (I_1 \cup O_1)$ . From this,  $\mathcal{T}_1 || \mathcal{T}_2$  is failure-free, iff  $(\text{del}^{-1}(D_2, P_1) \cap \text{del}^{-1}(D_1, F_2)) \cup (\text{del}^{-1}(D_2, F_1) \cap \text{del}^{-1}(D_1, P_2)) = \emptyset$ .

The correctness between modules is defined as follows.  $M_1 = (I_1, O_1, N_1, \text{wire}_1)$  *conforms* to  $M_2 = (I_2, O_2, N_2, \text{wire}_2)$ , if  $I_1 = I_2$ ,  $O_1 = O_2$ , and it holds that for any timed trace structure  $E = (I_E, O_E, S_E, F_E)$  such that  $I_2 = O_E$  and  $O_2 = I_E$ , if  $E || \mathcal{T}(M_2)$  is failure-free, so is  $E || \mathcal{T}(M_1)$ <sup>4</sup>. This conformation relation implies that  $M_1$  behaves similarly to  $M_2$  with any environment with respect to failure-freeness. From this correctness definition, the following property is inherited.

**Theorem 1.**  $\{M_1, \dots, M_{k-1}, M_{k_1}, \dots, M_{k_m}, M_{k+1}, \dots, M_n\}$  *conforms* to  $M_S$ , if  $\{M_{k_1}, \dots, M_{k_m}\}$  *conforms* to  $M_k$ , and  $\{M_1, \dots, M_{k-1}, M_k, M_{k+1}, \dots, M_n\}$  *conforms* to  $M_S$ .

The proof is shown in [8]. This is what we call *hierarchical verification*.

Practically, however, considering all possible timed trace structures  $E$  is infeasible. Instead, the *mirror* of  $\mathcal{T}(M_2)$  can be used. Intuitively, it is a maximal  $E$  such that  $E || \mathcal{T}(M_2)$ , and formally, for a timed trace structure  $\mathcal{T} = (I, O, S, F)$ , its mirror, denoted by  $\mathcal{T}^m$ , is also a timed trace structure  $(I', O', S', F')$  satisfying  $I' = O$ ,  $O' = I$ ,  $S' = S$ , and  $F' = A^* - P$ , where  $A = I \cup O$  and  $A^*$  is any timed trace over  $A$ . Then, it can be shown that  $M_1$  conforms to  $M_2$ , iff  $\mathcal{T}(M_1) || \mathcal{T}^m(M_2)$  is failure-free [8], which we call *mirror property*. In the untimed trace theory, for a module  $M = (I, O, N, \text{wire})$ ,  $\mathcal{T}^m(M)$  is coincidentally equal to  $\mathcal{T}(M')$  such that  $M' = (O, I, N, \text{wire})$ . This can be explained intuitively as follows. Suppose that  $\mathcal{T}(M') = (O, I, S', F')$ . First,  $S' = S$ , because the net is the same. In untimed systems, every trace  $y(w, \tau)$  satisfies  $\tau \leq \text{TL}(y, N)$  because there is no upper bound. Thus, for  $y \in S'$ , if  $y(w, \tau) \notin S'$  is a failure, then  $w$  must be an input of  $M'$ , i.e.  $w \in O$ . This is an impossible trace for  $M$ , because  $w$  is an output for  $M$ . Hence,  $F' = A^* - P$  holds, and so  $\mathcal{T}(M') = (O, I, S, A^* - P)$ , which is equal to  $\mathcal{T}^m(M)$  by definition. From this fact, it is straightforward to implement conformance checking for untimed systems.

Unfortunately, for timed systems, this construction of mirror trace structures is not correct (See [9] for details). Note that such mirroring is only necessary for a module representing a specification. Thus, in order to obtain a timed conformance checking algorithm similar to the one of untimed case, we introduce a slightly different version of a module for the specification. Since this is no longer a module of which timed trace structure is defined by C1, we call it *semimodule*. A semimodule is also a tuple  $(I, O, N, \text{wire})$ , but its timed trace structure  $(I, O, S, F)$ , denoted by  $\mathcal{T}_s(M)$ , is defined as follows.

- C2:** For  $w \in I \cup O$ , and  $\tau \in \mathbf{R}^+$ ,
1.  $S = \text{trace}(N)$ ,
  2. for  $y \in S$  and  $y(w, \tau) \notin S$ ,

<sup>4</sup> For this definition, the composition ( $||$ ) can be replaced by the intersection ( $\cap$ ), because the signal sets are common.  $M_1$  is, however, given as a set of modules in many cases where  $\mathcal{T}(M_1)$  is defined by the composition of the timed trace structures of the elements in  $M_1$ . Thus, using composition in the conformance definition is useful.

- (a) if  $\tau \leq \text{TL}(y, N)$ , then
  - i. if  $w \in I$ , then  $y(w, \tau) \in F$
  - ii. else  $y(w, \tau) \notin P$
- (b) else if  $\text{limit}(y, N) \subseteq O$ , then  $y(w, \tau) \notin P$
- (c) else  $y(w, \tau) \in F$
- 3. for  $y \in F$ ,  $y(w, \tau) \in F$ ,
- 4. for  $y \notin P$ ,  $y(w, \tau) \notin P$ .

The idea is to modify the case for  $\tau > \text{TL}(y, N)$  such that  $F' = A^* - P$  holds for  $P$  of a module  $(I, O, N, \text{wire})$  and  $F'$  of a semimodule  $(O, I, N, \text{wire})$ .

For a module  $M = (I, O, N, \text{wire})$ , we say that a semimodule  $(O, I, N, \text{wire})$  is the *semimirror* of  $M$ , denoted by  $M^{sm}$ . Consequently for timed systems,  $M_1$  conforms to  $M_2$  iff  $\mathcal{T}(M_1) \parallel \mathcal{T}_s(M_2^{sm})$  is failure-free [9].

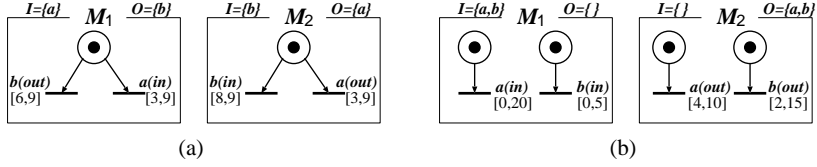
### 3 Verification Algorithm

In order to describe the algorithm, we first make several definitions as follows.

The state space of time Petri nets is infinite, because each clock function takes real values. Thus, a set of inequalities is used to represent a number of different clock functions in order to obtain the finite representations of state space. That is, a state of a time Petri net  $N_i$  is  $(\mu_i, K_i)$ , where  $\mu_i$  is a marking of  $N_i$ , and  $K_i$  is a set of inequalities over both past and future variables of transitions. For a transition  $t$ , its past variable, denoted also by  $t$ , represents its most recent firing time, while its future variable, denoted by  $\dot{t}$ , represents its next firing time. The variables representing older firings are projected out from  $K_i$ . In the initial state, a virtual past variable  $v$  is used. It can be considered that  $v$  represents the time when the net is initialized, and that all of the initially enabled transitions become enabled at that time.

For  $\mathcal{M} = \{M_0, M_1, \dots, M_{n-1}\}$  with  $M_i = (I_i, O_i, N_i, \text{wire}_i)$  such that  $M_0$  is a semimodule,  $\{M_1, M_2, \dots, M_{n-1}\}$  is a set of modules, and  $N_i = (P_i, T_i, F_i, lb_i, ub_i, \mu_i^0)$ . Its state is  $S = (\sigma_0, \sigma_1, \dots, \sigma_{n-1})$ , where  $\sigma_i = (\mu_i, K_i)$  is a state of  $N_i$ . For a transition  $t \in T_i$ ,  $\bullet t$  denotes the set of source places of  $t$ , i.e.,  $\{p \mid (p, t) \in F_i\}$ . Similarly, the set of destination places of  $t$  is denoted by  $t\bullet$ .  $\text{enabled}(\sigma_i) = \{t \mid t \in T_i, \bullet t \subseteq \mu_i\}$  and  $\text{enabled}(S) = \bigcup_{i=0}^{n-1} \text{enabled}(\sigma_i)$  denote a set of enabled transitions in  $N_i$  at  $\sigma_i$ , and in  $\mathcal{M}$  at  $S$ , respectively. A fireable transition  $t$  in  $\mathcal{M}$  is an enabled transition that can fire earlier than any other enabled transitions in  $\mathcal{M}$ , i.e.  $\text{Solution}(\text{first}(S, t) \cup \bigcup_{i=0}^{n-1} K_i) \neq \emptyset$  holds, where  $\text{Solution}(K)$  is a set of feasible vectors of  $K$  and  $\text{first}(S, t) = \{\dot{t} \leq \dot{t}' \mid t' \in \text{enabled}(S)\}$ .  $\text{firable}(S)$  denotes the set of fireable transitions in  $\mathcal{M}$  at  $S$ . For an enabled transition  $t$ ,  $\text{Eft}(t)$  and  $\text{Lft}(t)$  are  $t_p + lb(t)$  and  $t_p + ub(t)$ , respectively, where  $t_p$  is the past variable for a true parent of  $t$ , which is an output transition whose firing time made  $t$  enabled most recently. These expressions represent the lower bound and the upper bound of the next firing time of  $t$ .

Our goal is to decide whether a set  $\{M_1, M_2, \dots, M_{n-1}\}$  of modules conforms to a module  $M_0'$  or not, which is achieved by checking whether  $\mathcal{T}(\mathcal{M}) = \mathcal{T}_s(M_0) \parallel \mathcal{T}(M_1) \parallel \mathcal{T}(M_2) \parallel \dots \parallel \mathcal{T}(M_{n-1})$  is failure-free or not, where  $M_0$  is the semimirror of  $M_0'$ . For this purpose, the state space of  $\mathcal{M}$  is traversed, and if every reachable state



**Fig. 2.** Modules that cause safety and timing failures.

satisfies the following three conditions, then  $\mathcal{T}_s(M_0) \parallel \mathcal{T}(M_1) \parallel \mathcal{T}(M_2) \parallel \dots \parallel \mathcal{T}(M_{n-1})$  is concluded to be failure-free.

**Condition 1**

For any output transition  $t \in \text{firable}(S)$  of a (semi)module  $M_i$  with  $0 \leq i \leq n-1$ , and for any (semi)module  $M_j$  with  $0 \leq j \leq n-1$  such that  $\text{wire}_i(t) \in O_j$ , there exists an enabled input transition  $u$  in  $M_j$  with  $\text{wire}_j(u) = \text{wire}_i(t)$  such that  $\text{Solution}(\{\text{Eft}(u) > \dot{t}\} \cup \bigcup_{l=0}^{n-1} K_l) = \emptyset$  holds.

**Condition 2**

For any input transition  $t \in \text{firable}(S)$  of a module  $M_i$  with  $1 \leq i \leq n-1$ , there exists an output transition  $u \in \text{firable}(S)$  in some (semi)module such that  $\text{Solution}(\{\ddot{u} > \text{Lft}(t)\} \cup \bigcup_{l=0}^{n-1} K_l) = \emptyset$  holds.

**Condition 3**

For any input transition  $t \in \text{firable}(S)$  of the semimodule  $M_0$ , either

1. there exists an output transition  $u \in \text{firable}(S)$  in a module  $M_i$  with  $1 \leq i \leq n-1$  such that  $\text{Solution}(\{\ddot{u} > \text{Lft}(t)\} \cup \bigcup_{l=0}^{n-1} K_l) = \emptyset$  holds, or
2. there exists an output transition  $u \in \text{firable}(S)$  in the semimodule  $M_0$  such that  $\text{Solution}(\{\ddot{u} \geq \text{Lft}(t)\} \cup \bigcup_{l=0}^{n-1} K_l) = \emptyset$  holds.

Intuitively, an output produced by a (semi)module must be accepted by some other (semi)module, otherwise a failure exists. The condition 1 checks this kind of failures, which we call *safety failures*. A state that causes a safety failure is called the *safety failure state*. Consider the example shown in Figure 2(a), where  $t$  is  $b(\text{out})$  of  $M_1$  and  $u$  is  $b(\text{in})$  of  $M_2$ . Suppose that both  $M_1$  and  $M_2$  are modules. For timed trace  $y$  where a transition  $t$  is enabled in its last state, let  $\text{ep}(t, y)$  denote the earliest firingtime point of  $t$  in  $y$ , i.e., the value of  $\text{Eft}(t)$  obtained by assigning  $t_p$  to its actual firingtime in  $y$ .  $\text{lp}(t, y)$  is defined similarly. If  $\text{ep}(t, y) < \text{ep}(u, y)$  holds, then  $y(b, \tau) \in P_i$  and  $y(b, \tau) \in F_j$  for  $\text{ep}(t, y) \leq \tau < \text{ep}(u, y)$ , where  $b = \text{wire}_i(t) = \text{wire}_j(u)$ . Hence,  $y(b, \tau)$  is a failure of  $\mathcal{T}(M_i) \parallel \mathcal{T}(M_j)$ . In this example,  $\text{ep}(b(\text{out}), \varepsilon) = 6$ ,  $\text{ep}(b(\text{in}), \varepsilon) = 8$ , and so, for example,  $(b, 7) \in P_1$  and  $(b, 7) \in F_2$ . Thus,  $(b, 7)$  is a failure in  $\mathcal{T}(M_1) \parallel \mathcal{T}(M_2)$ . In this case, such modules have a feasible solution in  $\{\text{Eft}(u) > \dot{t}\} \cup \bigcup_{l=0}^{n-1} K_l$ , and so, condition 1 does not hold.

On the other hand, an input expected by a (semi)module must be given in time by some other (semi)module, otherwise a failure occurs. The condition 2 and condition 3 check this kind of failures, which we call *timing failures*. A state that causes a timing

failure is called the *timing failure state*. Consider an example shown in Figure 2(b), where  $t$  is  $b(in)$  of  $M_1$  and  $u$  is  $a(out)$  of  $M_2$ . Suppose that both  $M_1$  and  $M_2$  are modules, and so, this is handled by the condition 2. For timed trace  $y$ , suppose that  $\text{lp}(t, y) < \text{lp}(u, y)$  holds and  $\text{lp}(u, y)$  is the smallest latest firing time point among the enabled output transitions. Then,  $y(b, \tau) \in F_i$  and  $y(b, \tau) \in P_j$  for  $\text{lp}(t, y) < \tau \leq \text{lp}(u, y)$ , where  $b = \text{wire}_i(t)$ . The former holds because  $\text{limit}(y, N_i) \subseteq I$  holds from the assumption and C1. Hence,  $y(b, \tau)$  is a failure of  $\mathcal{T}(M_i) \parallel \mathcal{T}(M_j)$ . In this example,  $\text{lp}(b(in), \varepsilon) = 5$  and  $\text{lp}(a(out), \varepsilon) = 10$ , and so, for  $\tau = 9$ ,  $(b, 9) \in F_1$  and  $(b, 9) \in P_2$ . Thus,  $(b, 9)$  is a failure in  $\mathcal{T}(M_1) \parallel \mathcal{T}(M_2)$ . In this case, such modules have a feasible solution in  $\{\ddot{u} > \text{Lft}(t)\} \cup \bigcup_{i=0}^{n-1} K_i$ , and so, condition 2 does not hold.

The condition 3.1 is for the case where  $t$  is an input transition of a semimodule  $M_0$  and  $u$  is an output transition of some other module such that  $\text{lp}(u, y)$  is the smallest latest firing time point among the enabled output transitions. This case can be handled in the same way as the condition 2.

The condition 3.2 is for the case where output transitions with the smallest latest firing time point exist only in  $M_0$ . In this case, we further need to consider a special case where such an output transition  $u$  has the same latest firing time point as  $t$ , i.e., for timed trace  $y$ ,  $\text{lp}(u, y) = \text{lp}(t, y)$  holds. Consider the example shown in Figure 1 again where  $t$  is  $a(in)$  of  $M_0$  and  $u$  is  $b(out)$  of  $M_0$ , but suppose  $M_0$  is a semimodule this time. Since  $M_0$  is a semimodule and  $\text{limit}(y, N_0)$  contains the input transition  $t$ ,  $y(a, \tau) \in F_0$  holds from C2 for  $\text{lp}(t, y) < \tau \leq \tau'$ , where  $a = \text{wire}_0(t)$  and  $\tau'$  is the smallest  $\text{lp}(u', y)$  among the enabled output transitions  $u'$  in modules  $M_1, \dots, M_{n-1}$ .  $y(a, \tau) \in P_i$  for modules  $M_i$  with  $1 \leq i \leq n-1$  from  $\tau \leq \tau'$ . Hence,  $y(a, \tau)$  is a failure of  $\mathcal{T}(\mathcal{M})$ . In this example,  $\text{lp}(a(in), \varepsilon) = \text{lp}(b(out), \varepsilon) = 5$  hold in  $M_0$ , and  $\text{lp}(a(out), \varepsilon) = 10$ . Thus, for  $\tau = 6$ ,  $(a, 6) \in F_0$  and  $(a, 6) \in P_1$ , and so  $(a, 6)$  is a failure in  $\mathcal{T}(M_0) \parallel \mathcal{T}(M_1)$ . Thus, the condition 3.2 correctly handles this case by modifying the equality in  $\{\ddot{u} > \text{Lft}(t)\}$  of the condition 3.1 to  $\{\ddot{u} \geq \text{Lft}(t)\}$ .

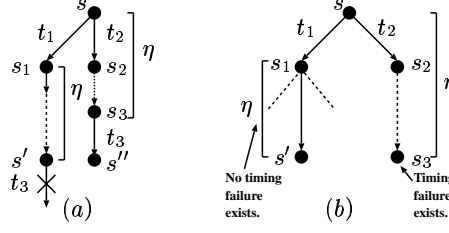
## 4 Partial Order Reduction for Timed Trace Theoretic Verification

In this section, the idea of partial order reduction for timed trace theory is proposed. Also, the difference between the proposed idea and [6] is discussed.

The concept of partial order reduction is to generate some subset of possible successor states as long as the correctness is not affected. We call a state space generated according to this principle the *reduced state space*. For timed trace theoretic verification, the reduced state space  $G_r = (S_r, R_r)$  should be constructed such that it satisfies the following conditions with respect to the full state space  $G_f = (S_f, R_f)$ , where  $S_r$  and  $S_f$  are the sets of reachable states for a set of (semi)modules, and  $R_r$  and  $R_f$  are the transition relations between states ( $R_f^*$  denotes the transitive closure of  $R_f$ ). A transition relation is a set of tuples  $(s, t, s')$  such that  $s'$  is obtained from  $s$  by firing an output transition  $t$  and its corresponding input transitions.

[PT1]  $S_r$  includes the initial state, and for  $s \in S_r$ , if  $s$  has successors in  $G_f$ , then  $s$  has at least one successor in  $G_r$ .

[PT2] For  $s \in S_r$ , if



**Fig. 3.** (a)  $t_1$  disables  $t_3$ . (b)  $t_1$  hides a timing failure.

- $(s, t_1, s_1) \in R_f$ , and
  - there exists a sequence  $\eta$  such that  $(s, \eta, s_3) \in R_f^*$ ,  $(s_3, t_3, s'') \in R_f$ ,  $(s_1, \eta, s') \in R_f^*$ ,  $\eta t_3 = t_2 \dots$  (if  $\eta$  is empty, then  $t_2 = t_3$ ), and the fi ring of  $t_2$  is necessary for the fi ring of  $t_3$ , but
  - $t_3$  is not enabled in  $s'$  (see Fig. 3(a)),
- then  $(s, t_1, s_1) \in R_r$  implies  $(s, t_2, s_2) \in R_r$ .

- [PT3] For  $s \in S_r$ , if
- $(s, t_1, s_1) \in R_f$ , and
  - there exists a sequence  $\eta$  such that  $(s, \eta, s_3) \in R_f^*$ ,  $\eta = t_2 \dots$ , and  $s_3$  has a timing failure, and
  - $(s_1, \eta, s') \in R_f^*$ , but
  - states from  $s_1$  to  $s'$  along  $\eta$  have no timing failure (see Fig. 3(b)),
- then  $(s, t_1, s_1) \in R_r$  implies  $(s, t_2, s_2) \in R_r$ .

[PT1] is vital, because a new deadlock state must not be introduced in  $G_r$ . [PT2] is for handling conflicting transitions. This is depicted by Figure 4(a). In Figure 4(a), since a fi ring of  $t_1$  immediately disables  $t_3$ , if  $G_r$  has only a successor by  $t_1$  in  $s$ , then we miss the fi ring of  $t_3$ . Thus, [PT2] fi rest $_2$  which is equal to  $t_3$  in this case ( $\eta = \varepsilon$ ). Moreover, a conflict may occur indirectly as shown in Figure 4(b). Suppose that  $t_1$  is in conflict with  $t_3$ , but  $t_3$  is not enabled in  $s$ . In this case, the fi ring of  $t_1$  does not disable  $t_3$  directly. However, the fi ring of  $t_3$ , which will become possible when fi ring  $t_2$  earlier than  $t_1$ , may be missed, if  $G_r$  contains only the fi ring of  $t_1$  in  $s$ . Thus, it must also contain the fi ring of  $t_2$  in order to retain the possibility that the fi ring of  $t_3$  occurs. [PT3] is for handling transitions hiding timing failures. Such a transition is an enabled output transition that has the larger latest fi ringtime point than the others. For example, consider the modules illustrated in Figure 5. If  $b(out)$  fi res at 10, then a timing failure occurs in the resultant state, because  $a(out)$  can fi relater than  $\text{lp}(a(in), (b, 10)) = 20$ . On the other hand, the current state is not a failure state, because  $b(out)$  always fi res earlier than  $\text{lp}(a(in), \varepsilon) = 20$ . Also note that  $a(out)$  is fi rable in this state. Therefore, if  $a(out)$  is fi red in this state, the above timing failure is never detected. In other words,  $a(out)$  hides the timing failure, and it corresponds to  $t_1$  of Figure 3(b). Hence, [PT3] forces  $b(out)$  to fi re, if  $a(out)$  is chosen for fi ring.

Due to the ignoring problem [1], we require that time certainly passes in any loop structure in any time Petri net in the module set and the latest fi ringtime of each output transition is bounded. This is necessary to prove the correctness of the partial order reduction algorithm.

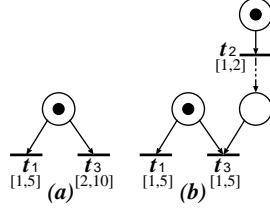


Fig. 4.  $t_1$  disables  $t_3$ .

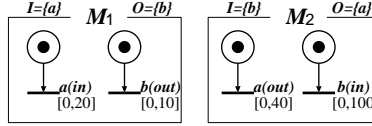


Fig. 5.  $a(out)$  hides a timing failure.

## 5 State Enumeration

This section briefly describes how to traverse the reduced state space of a set of (semi)-modules. The main idea is similar to that of [6]. Recall that we defined a state of a set of (semi)modules by  $S = (\sigma_0, \sigma_1, \dots, \sigma_{n-1})$ , where  $\sigma_i = (\mu_i, K_i)$  is a state of  $N_i$ . Here, we modify this definition a little for easier presentation. Let  $\{N_0, \dots, N_{n-1}\}$  denote a set of time Petri nets that compose the set of (semi)modules, where  $N_i = (P_i, T_i, F_i, lb_i, ub_i, \mu_i^0)$ . We define the state of  $\{N_0, \dots, N_{n-1}\}$  by  $(\hat{\mu}, K)$ , where  $\hat{\mu} \subseteq \bigcup_{i=0}^{n-1} P_i$  is a marking of all the nets and  $K$  is a set of inequalities.

The initial state of  $\{N_0, \dots, N_{n-1}\}$  is  $(\hat{\mu}_0, K_0)$ , where

$$\begin{aligned} - \hat{\mu}_0 &= \bigcup_{i=0}^{n-1} \mu_i^0, \\ - K_0 &= \{ \text{"}lb(t) \leq \ddot{t} - v \leq ub(t)\text{"} \mid t \text{ is an output transition such that } t \in \text{enabled}(\hat{\mu}_0) \} \end{aligned}$$

Recall that  $\ddot{t}$  in  $K_0$  is the future variable used to represent the next firing time of an output transition  $t$ , and  $v$  is a virtual variable to synchronize the nets at the initial state. Note that we only consider the output transition variables for the state space enumeration, because an input transition  $t$  of  $M_j = (I_j, O_j, N_j, \text{wire}_j)$  synchronizes with  $\text{out\_trans}(t)$ , where  $\text{out\_trans}(t)$  is the output transition that corresponds to  $t$ , i.e.,  $\text{out\_trans}(t) = t'$  such that  $\text{wire}_i(t') = \text{wire}_j(t)$ ,  $t' \in T_i$ ,  $N_i = (P_i, T_i, F_i, lb_i, ub_i, \mu_i^0)$ ,  $\text{wire}_i(t') \in O_i$ , and  $M_i = (I_i, O_i, N_i, \text{wire}_i)$ . Furthermore, we extend this notation by defining  $\text{out\_trans}(t) = t$  for an output transition  $t$ .  $\text{enabled}$  is extended for  $\hat{\mu}$ , that is,  $\text{enabled}(\hat{\mu}) = \bigcup_{i=0}^{n-1} \text{enabled}(\mu_i)$ .

Then, the set of successor states is obtained from a state  $s = (\hat{\mu}, K)$  by firing an output transition  $t$ . The output transition  $t$  is chosen from a set  $\text{ready}(s)$ . The  $\text{ready}(s)$  is the possibly smallest set of fireable transitions that satisfies [PT1],[PT2] and [PT3]. The ready set construction is described in the next section.

Sometimes more than one successor state  $s' = (\hat{\mu}', K')$  can be produced from a state  $s$  and an output transition  $t \in \text{ready}(s)$ . Those successors have the same marking

part  $\hat{\mu}'$  satisfying  $\hat{\mu}' = \bigcup_{i=0}^{n-1} \mu'_i$  such that

$$\mu'_i = \begin{cases} (\mu_i - \bullet t') \cup t' \bullet & \text{if } N_i \text{ has } t' \in \text{sync\_trans}(s, t), \\ \mu_i & \text{otherwise,} \end{cases}$$

where  $\text{sync\_trans}(s, t)$  be the set of enabled transitions that fire synchronously with  $t$ , i.e.,  $\{t' | t' \in \text{enabled}(\hat{\mu}), \text{out\_trans}(t') = t\}$ .

To obtain the inequality part  $K'$  of each successor, we first define  $J_1$  as follows.

$$J_1 = K \cup \{\ddot{t} \leq \ddot{t}' | t' \in \text{ready}(s)\} \cup \{t = \ddot{t}\}$$

This expresses the constraint that the firing time of  $t$  is smaller than those of any other transitions in  $\text{ready}(s)$ . Since  $t$  fires, its future variable  $\ddot{t}$  is copied to its past variable  $t$ .

Let  $\text{conflict}(u)$  be the set of transitions conflicting with  $u$ , i.e.,  $\{u' | \bullet u \cap \bullet u' \neq \emptyset\}$ . Furthermore, for a set  $U$  of variables and a set  $K$  of inequalities,  $K' = \text{delete}(K, U)$  is a unique set of inequalities over  $\text{var}(K) - U$ , such that the solution set of  $K'$  is equal to the solution set of  $K$ , projected on  $\text{var}(K) - U$ , where  $\text{var}(K)$  denotes the set of variables appearing in  $K$ . Since the transitions  $u$  in  $\text{sync\_trans}(s, t)$  also fire at the firing of  $t$ , the enabled output transitions  $t' \in \text{conflict}(u)$  for  $u \in \text{sync\_trans}(s, t)$  are disabled by the firing of  $t$ . Thus, the variables for those disabled transitions are no longer necessary, and the following  $J_2$  is obtained by deleting those variables for the disabled transitions.

$$J_2 = \text{delete}(J_1, \{\ddot{t}' | t' \text{ is an output transition in } \text{enabled}(\hat{\mu}) \cap \text{conflict}(u) \\ \text{for } u \in \text{sync\_trans}(s, t)\})$$

Next, the true parents of newly enabled transitions are determined. Let  $E(s, s', t) = (t_1, t_2, \dots, t_m)$  denote the set of newly enabled output transitions in a successor  $s'$  of  $s$  by  $t$ . Note that it includes the transitions that become enabled by the firings of input transitions in  $\text{sync\_trans}(s, t)$ . Consider a transition  $t_i \in E(s, s', t)$ . If  $t_i$  has more than one source place, i.e.  $|\bullet t_i| > 1$ , we need to decide among the candidates  $X(t_i)$  of the true parent of  $t_i$  which transition really enables  $t_i$ , where  $X(t_i) = \{x | x \in \text{out\_trans}(\bullet \bullet t_i) \cap \text{var}(J_2)\}$ . Since we have choices of a true parent for each newly enabled transition in  $E(s, s', t)$ , we have successors which correspond to the possible combinations of true parents. Here, consider one of such combinations  $(u_1, u_2, \dots, u_m)$  for  $E(s, s', t) = (t_1, t_2, \dots, t_m)$ , where  $u_i \in X(t_i)$  is the true parent of  $t_i$ . The timing constraint needed for it is that  $u_i$  can fire later than the other true parent candidates. That is,

$$J_3 = J_2 \cup \bigcup_{1 \leq i \leq m} \{x \leq u_i | x \in X(t_i)\}$$

Finally,

$$J_4 = J_3 \cup \bigcup_{1 \leq i \leq m} \{lb(t_i) \leq \ddot{t}_i - u_i \leq ub(t_i)\}$$

decides the firing timing of  $t_i$  based on  $u_i$ ,  $lb(t_i)$ , and  $ub(t_i)$ . This  $J_4$  has all necessary information for the inequality part of  $s'$ . It, however, still contains some unnecessary past variables, and deleting them is necessary to make the state space finite. A past

variable  $u$  is necessary, if the transition  $u$  has some chance to become a true parent of some currently disabled transition  $t$  such that  $u \bullet \cap \bullet t \neq \emptyset$ , or the transition  $u$  is a true parent of some enabled transition. The former condition is considered in [6], but the latter is a new condition needed in our method because a transition which is a true parent of some enabled transition must be kept<sup>5</sup> in order to detect the safety and timing failures as mentioned before and construct the ready set which is described in the next section. Therefore, the set  $D$  of unnecessary past variables can be defined as follows.

$$\begin{aligned}
D &= \{v \mid \text{for every } u \in Y(v) \text{ and } x \in Z(u), \text{ either (1) } u \bullet \cap \bullet x \not\subseteq \hat{\mu}', \text{ or} \\
&\quad (2) x \notin \text{enabled}(\hat{\mu}') \text{ and } \text{no\_parent}(u, x, \hat{\mu}', J_4), \text{ or} \\
&\quad (3) x \in \text{enabled}(\hat{\mu}') \text{ and } \text{ub}(x) = \infty \text{ and } \text{no\_safety\_failure}(x, \hat{\mu}', J_4)\}, \\
Y(v) &= \{u \mid v = \text{out\_trans}(u)\}, \\
Z(u) &= \{x \mid x \text{ is an output transition such that } u \bullet \cap \bullet x \neq \emptyset\},
\end{aligned}$$

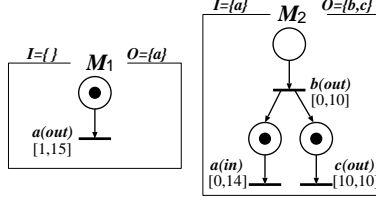
where  $\text{no\_parent}(u, x, \hat{\mu}, J)$  is a predicate representing that  $\text{out\_trans}(u)$  can never become a true parent of the transition  $x$  under  $\hat{\mu}$  and  $J$ . Formally, it is true, iff for some  $p \in \bullet x - u \bullet - \hat{\mu}$ , and for every pair  $(y, w)$  with  $w \in \bullet p$  and  $y \in \text{enabled}(\hat{\mu})$ ,  $\text{Solution}(\{\text{out\_trans}(u) \geq \ddot{y} + \text{diff}(y, w)\} \cup J) = \emptyset$ , where  $\text{diff}(t_1, t_2)$  represents the minimal values of sum of earliest firing times in all paths (input transitions are linked to their corresponding output transitions and the earliest firing times of output transitions are used.) from an output transition  $t_1$  to another output transition  $t_2$ , with  $lb(t_1)$  not included. And,  $\text{no\_safety\_failure}(x, \hat{\mu}, J)$  is a predicate representing that a safety failure can no longer be occurred by an input transition  $x$ , and so, it is not necessary to keep the true parent of  $x$  in order to check the safety failure. Formally, it is true iff  $\forall t \in \text{enabled}(\hat{\mu}) \cap O, \text{Solution}(\{\text{Eft}(x) > \dot{t}\} \cup J) = \emptyset$  holds. Hence, the inequality part  $K'$  of a successor state of  $s$  by  $t$  is obtained as  $K' = \text{delete}(J_4, D)$ .

The state space is traversed in the depth first search manner checking safety and timing failures as long as a new successor state that is not covered by any other reached states is obtained, where a state  $s = (\hat{\mu}, K)$  is covered by state  $s' = (\hat{\mu}', K')$ , if  $\hat{\mu} = \hat{\mu}'$  and the solution set of  $K$  is a subset of that of  $K'$ .

## 6 Ready Set Construction

A ready set must be constructed according to the idea of the partial order reduction for timed trace theory mentioned in Section 4. Thus, all transitions in the ready set must satisfy [PT1], [PT2], and [PT3]. For [PT1], some fireable output transition is included in the ready set, if it exists. Suppose that  $t_1 \in \text{ready}(s)$ . If  $t$  has no conflicting transitions, i.e.,  $\text{conflict}(t_1) = \{t_1\}$ , then  $\{t_1\}$  can be a ready set. Otherwise, for example, if there exists a disabled transition  $t_3$  which is in conflict with  $t_1$ , then in order to satisfy [PT2] the ready set must include a fireable output transition whose firing is necessary to enable  $t_3$  in the future. In the example shown in Figure 4(b),  $t_2$  is such a transition. Furthermore, if  $a(\text{out})$  is in the ready set in Figure 5, then  $b(\text{out})$  must be included in the ready

<sup>5</sup> In the actual implementation, in order to reduce the number of variables, input transitions with  $[0, \infty]$  are handled in the same way as [6] instead of keeping their true parents.



**Fig. 6.** No firable output transitions are chosen by  $\text{limiting}(s)$ .

set from [PT3]. We call such  $b(out)$  a *limiting transition*. That is, a limiting transition is a firable output transition such that its latest firing time point is smallest among the firable output transition of all (semi)modules.

The ready set construction starts from finding a limiting transition. First, consider

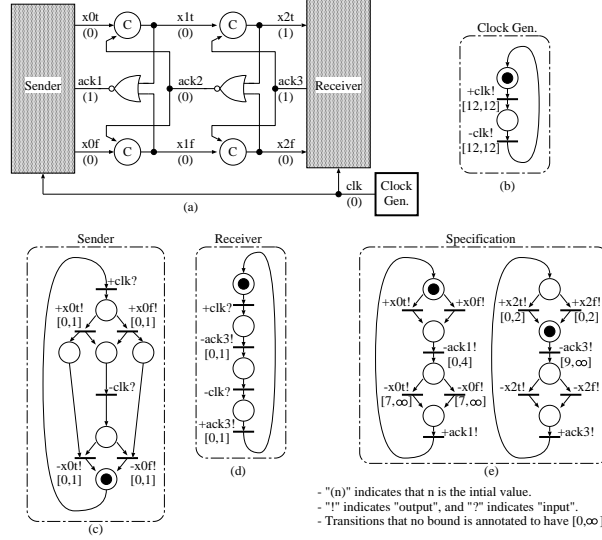
$$\text{limiting}(s) = \{t \mid \forall u \in \text{firable}(s) - \{t\},$$

$$\text{Solution}(\{\ddot{t} > \text{Lft}(\text{out\_trans}(u))\} \cup \bigcup_{i=0}^{n-1} K_i) = \emptyset\}.$$

The transitions in  $\text{limiting}(s)$  have certainly smaller latest firing time points than the other firable transitions. In the example of Figure 5,  $\text{limiting}(s_0) = \{b(out)\}$ . If it is nonempty, any singleton subset of  $\text{limiting}(s)$  can be used as a seed of the ready set, because firing such a transition does not hide the timing failure anyway. This condition is, however, sometimes too strong. For example, in the net shown in Figure 6, neither  $a(out)$  nor  $c(out)$  satisfies this condition, i.e.,  $\text{limiting}(s) = \emptyset$ . This is because the state  $s$  does not distinguish situations obtained by different firing times of  $b(out)$ , i.e., if  $b(out)$  fires before time 5, then  $c(out)$  has smaller latest firing time point than  $a(out)$ , and otherwise,  $a(out)$  has smaller latest firing time point than  $c(out)$ . We solve this problem in a conservative way such that if  $\text{limiting}(s)$  is empty, then the set of all firable output transitions is used as a seed of the ready set.

The seed of the ready set obtained in this way satisfies [PT1] and [PT3]. In order to satisfy [PT2], the seed should be extended such that for any transition  $t$  in the set, the set includes the dependent set of  $t$ , where the dependent set is defined below. Note that this process is the same as the one presented in [6], and so, only its intuitive idea is described here. The details are shown in [6].

The necessary set for a transition  $t$  is  $\{t\}$ , if  $t$  is enabled. Otherwise, it is a set of enabled transitions such that  $t$  can never be enabled if none of those transitions is fired. For example, the necessary set for  $t_3$  is  $\{t_3\}$  for Figure 4(a), and  $\{t_2\}$  for Figure 4(b). The dependent set  $\text{dependent}(s, t)$  for a transition  $t$  in  $s$  is a set of transition that satisfies (1)  $t \in \text{dependent}(s, t)$ , and (2) if  $u \in \text{dependent}(s, t)$ , then the necessary sets for all the transitions that conflict with both  $u$  and those synchronized with  $u$  are included in  $\text{dependent}(s, t)$ . This set contains transitions that should be fired when  $t$  is fired in  $s$ . For example, the dependent set for  $t_1$  is  $\{t_3\}$  for Figure 4(a), and  $\{t_2\}$  for Figure 4(b). In the latter case, if the path from  $t_2$  to  $t_3$  takes a long time, and  $t_3$  cannot actually conflict with  $t_1$ , then  $t_2$  does not have to be in the dependent set. Thus, the sum of the



**Fig. 7.** Specification and environment of STARI circuit.

earliest firing times of transitions in such a path can be used to decide this possibility. Also note that  $\text{dependent}(s, t)$  may contain nonfirable transitions. In such a case, all firable output transitions are used for  $\text{dependent}(s, t)$  again conservatively.

Moreover, if some (semi)module contains independent loop structure, our algorithm may not terminate [6]. This is because the time differences of concurrent transitions increase without converging. This situation can be detected by checking that the time differences exceed some constant value. In such a case, again the set of all firable transitions is used for the ready set, meaning that the algorithm temporally reverts to the full state space enumeration.

## 7 Experimental Results

To show the performance of the proposed method, we have implemented it based on a tool VINAS-P [6]. This section demonstrates the proposed method with the STARI circuits [10, 11].

The STARI circuit is composed of a number of FIFO stages. A two-stage STARI circuit is shown in Figure 7(a). These gates are modeled by the timed Petri nets. In [11], the verification of this circuit with respect to the following three properties is demonstrated: (1)  $ack1$  goes low (i.e. the current data is moved to the next stage.) earlier than the separator is given in  $(x0t, x0f)$ , (2) a new data is ready in  $(x2t, x2f)$  earlier than  $ack3$  goes low (i.e. the receiver samples the data.), and (3) no hazard occurs at any gate. In addition to these properties related to the event ordering, we verify the following timing properties for  $n$ -stage STARI circuits with  $n \geq 3$  in this experiment: (4)  $ack1$  goes low within 4 time units after a new data is sent from the sender and at least 7 time unit before the next separator is sent, (5) a new data is ready in  $(x2t, x2f)$  within

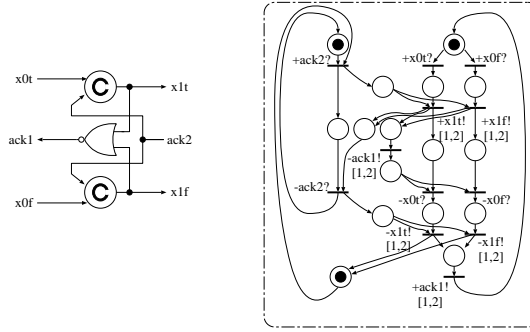


Fig. 8. Sub-circuit and its specification for hierarchical verification.

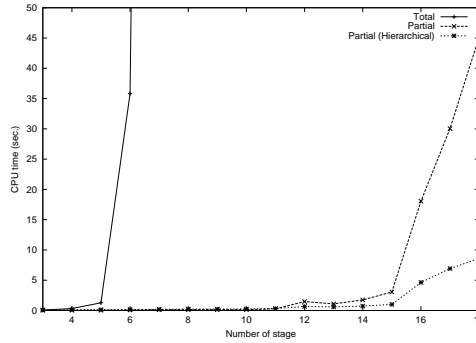


Fig. 9. CPU times for verifications of STARI circuits.

2 time units after ack3 goes high and at least 9 time unit before ack3 goes low again. To express these whole properties, we use the time Petri net shown in Figure 7(e).

Here, the experiments have been done on a 2.8 GHz Pentium 4 workstation with 4 gigabytes of memory. We have verified the STARI circuits by using the total order method and the partial order method to compare their performances, where in the total order method, the ready set contains all fireable output transitions. Moreover, we have hierarchically verified these circuits with the partial order method as well. In this experiment, we first verify a one-stage STARI circuit with its specification shown in Figure 8. Note that this specification is obtained by analyzing the behavior of the one-stage STARI circuit. The verification of such a sub-circuit and its specification should be done for stages with different initial markings. Once those verifications succeed, every stage-circuit is replaced by its corresponding specification, which is much smaller than the circuit model, and it is verified that the set of those sub-specifications conforms to the original specification (Figure 7(e)).

Figure 9 shows the CPU times for verifications of  $n$ -stage STARI circuits where the x-axis shows  $n$ . Note that "Partial(hierarchical)" includes the verification runs for sub-circuits. These results show that the performance improved by partial order reduction is significant, and the hierarchical verification is much more powerful. One disadvantage

tage of the hierarchical verification is that the sufficient sub-specifications should be prepared by users.

In addition to these experiments, we have run the XOR chain example in [12] to compare the proposed method with Minea's work. According to our results, the example is very sensitive with the delay bounds of the XOR gates, which is not specified in his thesis. Thus, fair comparison is not easy. One fact is that the total order method outperforms our partial order method in this example, although it has some amount of concurrency. This is probably because this example contains many almost independent loops, which makes the visited state checking difficult in the partial order method (the details can be found in [6]).

## 8 Conclusion

In this paper, we have proposed a partial order reduction algorithm for a timed trace theoretic verification. Our algorithm can hierarchically verify timed circuits and detect a kind of liveness failures (i.e. timing failures). Experimental results obtained from the STARI circuits by using the partial order reduction show the effectiveness of the proposed method.

We are planning to do a case study to verify a practical system for showing the usefulness of the proposed method.

## References

1. A. Valmari. Stubborn sets for reduced state space generation. *LNCS 483 Advances in Petri Nets*, pages 491–515, 1990.
2. P. Godefroid. Using partial orders to improve automatic verification methods. *Proc. of Computer Aided Verification Workshop*, 1990.
3. S. Katz and D. Peled. Defining conditional independence using collapses. *Semantics for concurrency, BCS-FACS Workshop, M. Kwiatkowska (ed.), Springer*, 1990.
4. K. L. McMillan. Trace theoretic verification of asynchronous circuits using unfoldings. *LNCS 939 Computer aided verification*, pages 180–195, 1995.
5. E. G. Mercer. *Correctness and Reduction in Timed Circuit Analysis*. PhD thesis, University of Utah, 2002.
6. Tomohiro Yoneda and Hiroshi Ryu. Timed trace theoretic verification using partial order reduction. *Proc. of Fifth International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 108–121, 1999.
7. B. Zhou, T. Yoneda, and H. Schlingloff. Conformance and mirroring for timed asynchronous circuits. *Proc. of ASP-DAC'01*, pages 341–346, 2001.
8. D. L. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits*. MIT press, 1988.
9. B. Zhou and T. Yoneda. Verification of asynchronous circuits with bounded delay model. *IEICE Trans. (in Japanese) Vol.J82-D-I, No.7*, pages 819–833, 1999.
10. S. Tasiran and R. Brayton. STARI: A case study in compositional and hierarchical timing verification. *LNCS 1254 Computer Aided Verification*, pages 191–201, 1997.
11. W. Belluomini and C. Myers. Verification of timed systems using POSETs. *LNCS 1427 Computer Aided Verification*, pages 403–415, 1998.
12. Marius Minea. *Partial order reduction for verification of timed systems*. PhD thesis, Carnegie Mellon University, 1999.