Reducing Probabilistic Timed Petri Nets for Asynchronous Architectural Analysis [†]

Sangyun Kim, Sunan Tugsinavisut
University of Southern California
Los Angeles, CA 90089, USA
{sangyunk,tugsinav}@eiger.usc.edu

Peter Beerel [‡]
Fulcrum Microsystems
Calabasas Hills, CA 91301
pabeerel@fulcrummicro.com

ABSTRACT

This paper introduces structural reductions of probabilistic timed Petri nets that preserve a large class of performance measurements. In particular, the paper proposes a class of reductions that preserve efficiently computable bounds of statistics of time-separation of events (TSEs). It identifies two specific reductions within this class. It demonstrates the utility of these reductions by reducing a detailed Petri net describing the four-phase protocol of a well-known asynchronous pipeline template into a simpler two-phase architectural-level Petri net model. The benefit of this reduced model is that the run-time of subsequent TSE analysis can be greatly improved.

Categories and Subject Descriptors

B.8.2 [**Hardware**]: Performance and Reliability—*Performance Analysis and Design Aids*

General Terms

Algorithms, Design, Performance, Theory

1. INTRODUCTION

Recently, asynchronous designs have demonstrated potential benefits in low power, high performance, composability, and improved noise immunity. (e.g. [6, 1, 12, 14]). For high-speed applications, many fine-grain pipelining techniques are developed [7, 13, 11].

Estimation and optimization of their performance, however, remains somewhat of a stumbling block due to the complex interaction of various handshaking protocols. Traditionally, performance analysis has been limited to simulation of detailed design models which suffers from long run-times and results that are subject to the reliability of the input vector statistics. For this reason, many analytical methods to analyze the performance of asynchronous circuits efficiently have been proposed [17]. Many of these techniques use timed Petri net with fixed delay [3], interval delay [4, 5, 10], and/or stochastic delay [16, 15] as circuit models. Numerous methods of analyzing these Petri nets have been proposed but most are limited to a small subclass of Petri nets or suffer from severe capacity limitations. For a large class of stochastic Petri nets (ones that do not involve arbitrarion), a recently developed approach using statistical simulation to obtain bounds on TSE statistics has been shown to be computationally efficient, able to handle models with hundreds of places and transitions. Nevertheless, real system models may have tens of thousands of places and transitions and increasing the capacity of analysis and reducing run-time is very important, particularly if analysis is used internal to a synthesis or design loop. It is this problem that this paper addresses.

The goal of this paper is to introduce simple reductions of Petri nets that preserve performance properties. In particular, we identify structural reductions that preserve bounds on TSE statistics, as computed using the algorithms in [16, 15]. This means that the reduced Petri nets can be analyzed instead of their more complicated counterparts. One intended use of this reduction algorithm is to simplify the performance model of small asynchronous cells which make up the building blocks of large asynchronous systems. These cells may be as small a single pipeline stage or consist of a collection of small cells that make up a larger re-usable library component. The reduced performance model of such cells can hide internal handshaking details and highlight the key performance parameters of the cells along with the critical interactions with its environment. The reduced performance models also greatly simplifies the combined performance model of any larger system comprised of these building blocks, making system-level performance analysis much more efficient. In this way, this work may form the basis to supplement any building-block based asynchronous design flow with efficient performance analysis.

In particular, the paper proposes two specific reductions, projection and redundancy removal, and proves that these reductions are in the identified class of reductions that preserve TSE statistics. The paper demonstrates the utility of these reductions by reducing a detailed Petri net describing the four-phase protocol of a linear pipeline comprised of well-known pre-charged half-buffer (PCHB) pipeline buffers [7]. The result is a much simpler model of the linear pipeline at

[†]This work was partially supported by NSF ITR Award No. CCR-00-86036.

[‡]Professor Beerel is currently on a leave of absence from the University of Southern California.

TAU'02, December 2–3, 2002, Monterey, California, USA.

more of an architectural level, highlighting the critical paths through the pipeline buffers.

The remainder of this paper is organized as follows. Section 2 describes the class of probabilistic timed Petri nets (PTPNs) we address and defines a notion of their timed execution. Section 3 reviews the bounding algorithm for TSE statistics. Section 4 and 5 introduce a class of reduction operators that preserves TSE bounds and two specific reduction algorithms within this class. Section 6 and 7 consist of our PCHB case study and some conclusions.

2. PTPN MODELS OF ASYNCHRONOUS SYSTEMS

This section reviews the subclass of Petri nets on which our proposed reduction techniques operate. More detailed review of Petri nets can be found in a survey paper [9].

A Petri net is a triple $N = (P, T, F)$ where P is the set of places, T the set of transitions, and $F \subseteq (P \times T) \cup (T \times P)$ the flow relation. N is a *marked graph* (also known as *event graph*) if every place has at most one input and one output transition. A place p is a *choice* place if $|p\bullet| > 1$. N is a *free-choice* (FC) net if $\forall p_1, p_2 \in P$, $p_1\bullet \cap p_2\bullet \neq \emptyset \implies |p_1\bullet| = |p_2\bullet| = 1$. Equivalently, in a FC net, if two transitions share an input place, they do not have any other input places. A Petri net is *extended free-choice* (EFC) if $\forall p_1, p_2 \in P$, $p_1\bullet \cap p_2\bullet \neq \emptyset \implies p_1\bullet = p_2\bullet$. An EFC net can be translated into a FC net [9]. A choice place is asymmetric if it is neither FC nor EFC. An asymmetric choice is *unique-choice* if at most one of its output transitions can be enabled at a time. Otherwise, it is an *arbitration* choice.

The class of underlying (untimed) nets of the performance models we consider is the PNs with (extended) *free-choice* and/or *unique-choice*, i.e., PNs without arbitration choice.

A marking is a mapping $M : P \rightarrow \{0, 1, 2, \cdots\}$ where $M(p)$ denotes the number of tokens in place p. A transition t is *enabled* at marking M if $M(p) \geq 1, \forall p \in \bullet t$. An enabled transition may fire. The firing of t removes one token from each place in its preset and deposits one token to each place in its poset, leading to a new marking M', denoted by $M[t\rangle M'$. A sequence of transitions $\sigma = t_0 t_1 \cdots t_{m-1}$ is a firing sequence from a marking M_0 iff $M_k[t_k\rangle M_{k+1}$ for $k = 1, \cdots, m-1$. In this case, we write $M_0[\sigma\rangle M_m$ and say σ has a length of m denoted by $|\sigma|$. We write $\vec{\sigma}$ to be the firing counter vector of σ, indicating the numbers of times transitions are fired along σ. The vector is called a *T-invariance* if there is a marking M such that $M[\sigma\rangle M$.

A *marked* net Σ is a tuple (N, M_0), where N is a net and M_0 is its initial marking. The set of reachable markings of Σ is denoted by $R(M_0)$. Σ is *live* iff all transitions will eventually be enabled from every $M \in R(M_0)$. It is *k-bounded* if $M(p) \leq k$ for $\forall M \in R(M_0), \forall p \in P$. A 1-bounded net is also called *safe*. A live and bounded (LB) marked net has no source or sink places and no source or sink transitions (e.g.,[9]). Thus, a LB net can be partitioned into a set of strongly connected components each evolving independently of others. Below, we assume the net is strongly connected. In particular, we restrict ourselves to LB nets with free-choice and unique-choice places.

2.1 Timing and choice probabilities

In this paper, we associate time with places and adopt the *fixed delay model* to model the passage of time. That is, a token flowing into a place p experiences a specified fixed delay associated with p denoted by $d(p)$, before it can be consumed by an output transition of p. The actual firing of a transition is assumed to be *instantaneous*. For Petri nets with free- or unique- choices, these assumptions imply there is no race condition among transitions in structural conflict, i.e., the poset of a choice place.

For each choice place p, we assume there is a probability mass function (p.m.f) $\mu(p, \cdot)$ to resolve the choice. That is, if $t \in p\bullet$, $\mu(p, t)$ is the probability that t consumes the token each time p is marked. Of course, $\sum_{t \in p\bullet} \mu(p, t) = 1$. P_c, P_{nc} denote a set of all choice places, and a set of all places without choice respectively. We call this subset of Petri nets, probabilistic timed Petri nets (PTPN), and note that they are a subset of stochastic timed Petri nets introduced in [16]. Figure 1(a) shows an example of such a net where P_8 is free-choice place with probability mass function.

We say ψ is a deterministic path in Petri net N if it is a sequence of non-choice places connected by transitions. The set of all deterministic paths leading from x to y is denoted by $\Psi(x, y)$ where $x, y \in P_{nc} \cup T$. Let $\Delta(\psi)$ be sum of delay along that deterministic path, that is, $\Delta(\psi) = \Sigma_{p \in \psi} d(p)$. Let $M(\psi)$ be sum of tokens along that deterministic path, that is, $M(\psi) = \Sigma_{p \in \psi} M(p)$.

2.2 Timed executions

We call a possible run of a PTPN a *timed execution* where choices are resolved and places are assigned delay values. In particular, we call a firing of a transition an *event*. A timed execution can be described as a sequence of events and their occurrence times. Alternatively, it can be depicted as a timed event graph which describes the causality among events.

For example, Figure 1(b) shows the event graph of a timed execution of the PTPN in Figure 1(a). The numbers along the (instanced) places denote the delay values. For convenience, we write t^k and p^j to denote the k-th event due to the firing of t and the j-th instance of place p, respectively.

Formally, a timed execution π of Σ is a triple (N_π, d_π, ℓ) where $N_\pi = (P_\pi, T_\pi, F_\pi)$ is an acyclic event graph, d denotes the delay value of each place in P_π and a labeling function $\ell : P_\pi \cup T_\pi \rightarrow P \cup T$ that maps each places and transitions of N_π to their corresponding ones in Σ. We use a function τ (called timing function) to denote the occurrence times of events. For a given timed execution, the occurrence time of event $t^{(k)}$ is determined as follows:

$$\tau(t^{(k)}) = \max_{\substack{(s^{(j)}, p) \in F_\pi, \\ p \in \bullet t^{(k)}}} \tau(s^{(j)}) + d(p) \quad (1)$$

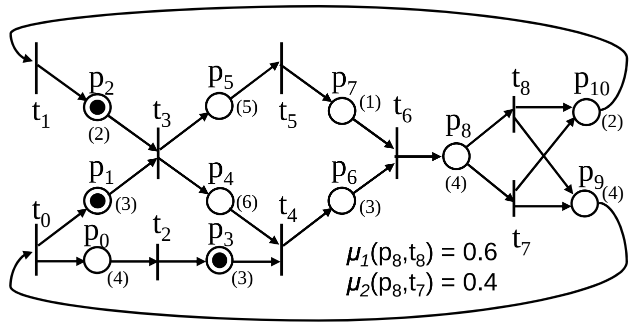
where the term $\tau(s^{(j)}) + d(p)$ reduces to $d(p)$ if p is a source place of π.

2.3 TSEs and their statistics

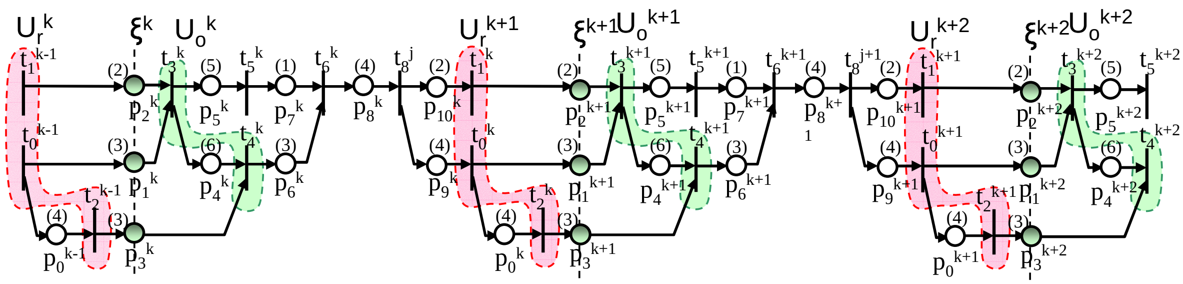
Given a timed execution π, the time separation of an event pair (TSE) is the time distance between the two events. More precisely, the TSE of event pair $(s^{(k)}, t^{(k+\varepsilon)})$, denoted by, $\gamma^{(k)}(s, t, \varepsilon)$, is:

$$\gamma^{(k)}(s, t, \varepsilon) = \tau(t^{(k+\varepsilon)}) - \tau(s^{(k)}) \quad (2)$$

where τ is the timing function of π, and (s, t, ε) is called the corresponding *separation triple*. When π is viewed as a random timed execution, the TSE $\gamma^{(k)}(s, t, \varepsilon)$ is a random



(a) A PTPN



(b) A timed execution of the PTPN in (a)

Figure 1: A probabilistic timed Petri net and its timed execution.

variable, and the sequence $\{\gamma^{(k)}(s,t,\varepsilon) : k = 1, 2, \cdots\}$ is a random process.

DEFINITION 1. *The average TSE due to separation triple (s,t,ε), denoted by $\overline{\gamma}(s,t,\varepsilon)$, is the average of the corresponding TSEs of an infinite timed execution of the PTPN. That is,*

$$\overline{\gamma}(s,t,\varepsilon) = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} \gamma^{(k)}(s,t,\varepsilon). \tag{3}$$

Many system performance metrics such as average throughput and latency can be directly expressed as the average TSEs of some indicating event pairs. In addition, extensions to variance and other higher order statistics of the TSE can be similarly defined [15].

3. BACKGROUND: BOUNDING TSES

The key idea of the bounding approach is to partition an infinite timed execution into a sequence of so called *segments*. These segments are independent and identically distributed (iid). The targeted TSEs are then analyzed within individual segments independent of remaining segments. Since the analysis is done using limited history, it yields upper and lower bounds on the TSEs (via longest path analysis) instead of exact values of the TSEs. These bounds are iid, which facilitates the estimation of the statistics of the bounds with well-known statistical methods [16].

3.1 Partitioning infinite timed executions

Let $\pi = (N_\pi, d, \ell)$ be a timed execution of Σ. It is known that every (untimed) reachable marking of Σ induces a cut ξ (a set of instanced places) which partitions the event graph N_π [2]. The portion of the event graph in between two different cuts due to the same reachable marking M is a *segment*. Formally, if σ is a firing sequence that starts from a cut ξ and ends at another cut ξ' such that $\ell(\xi) = \ell(\xi') = M$, then the portion of N_π between ξ and ξ' is a segment, denoted by $S(\xi,\sigma)$. A segment $S(\xi,\sigma)$ can also be denoted by $S(\xi,\theta)$ such that $\theta = \{(s,t) | s \in T_s, t \in T_\pi\}$ is a strict partial order of event pairs where T_s is a set of all events in the segment. The segment is *minimal* if it does not contain any other segment starting from ξ. For example, in Figure 1(b), the k^{th} minimal segment is defined over $T_s = \{t_0^k, t_1^k, t_2^k t_3^k, t_4^k, t_5^k, t_6^k, t_8^j\}$ and is denoted by $S(\xi^k, \theta^k)$ where $\xi^k = \{p_1^k, p_2^k, p_3^k\}$ and θ^k is the strict partial order of the events on T_s projected from the partial order implied by T_π. This partial order is specified by an infinite set of pairs of events that includes $\{(t_3^k, t_5^k), (t_3^k, t_4^k), (t_3^k, t_6^k), (t_5^k, t_6^k)\}$.

One simple property of a random time execution of a Petri net considered in this section is that the structures of its segments are independent of each other. This is because the structure of a segment is determined by the choices made on the places inside the segment and choices made in different segments are independent. As a result, the sequence of segments generated by a random timed execution has the property that their structures are not determined by the location of the segment in the sequence. In fact, they are independent and identically distributed (iid). This fact allows us to reason about an infinite execution by considering all possible finite executions of as little as one segment in length.

3.2 Bounding a single TSE instance

We obtain bounds on a TSE instance belonging to a particular segment by ignoring the history of the segment. In other words, our bounds on a TSE instance are defined by assuming tokens in the source places of the segment can be available at any time within $(-\infty, \infty)$. This subsection describes a method to compute the upper bound using longest path analysis. The lower bound can be computed using duality.

To compute the upper bound, we identify a set of *reference events* that serve the synchronization points for the targeted events. By assuming each of the synchronization points to be critical, one obtains a set of time separations of the event pair. The upper bound is simply the largest separation obtained. To detail this idea, consider upper bounding the TSEs due to separation triple $(s, t, \varepsilon = 0)$. Extension to the case where $\varepsilon \neq 0$ is not difficult.

Let ρ be a path in the event graph of a timed execution $\pi = (N_\pi, d, \ell)$. The set of all paths leading from x to y is denoted by $\mathcal{P}(x, y)$ where $x, y \in P_\pi \cup T_\pi$. A *reference set* for event e of π is a subset of events of π such that every path from a source place of N_π to e contains at least one event in R, and every event in R has a path to e.

It follows from the timing relation (1) that if x has a path to y, then y must occur after x by at least the the sum of place delays along any path from x to y. That is, whenever $\mathcal{P}(x, y) \neq \emptyset$,

$$\tau(x) + \max_{\rho \in \mathcal{P}(x,y)} \delta(\rho) \leq \tau(y). \quad (4)$$

where $\delta(\rho) = \sum_{p \in \rho} d(p)$. In particular, we say event x is *critical* for event y if (4) holds in equality. Further, if R be a reference set for event y. Then, the occurrence time of y is uniquely determined by the occurrence times of the events of R plus the delay values of places following these events. That is,

$$\tau(y) = \max_{x \in R} [\tau(x) + \max_{\rho \in \mathcal{P}(x,y)} \delta(\rho)]. \quad (5)$$

The term $\max_{\rho \in \mathcal{P}(x,y)} \delta(\rho)$ in (5) measures the maximum delay on any path from event x to y. For convenience, we write:

$$\delta^*(x, y) = \max_{\rho \in \mathcal{P}(x,y)} \delta(\rho), \quad (6)$$

where $\delta^*(x, y) \stackrel{\triangle}{=} -\infty$ if there is no path from x to y, i.e., $\mathcal{P}(x, y) = \emptyset$. For completeness, we define an event e itself to be a path of delay 0, and consequently, $\delta^*(e, e) = 0$.

Suppose the m-th TSE instance starts in the l-th segment of π denoted by $S^{(l)}$. Since the set of source places of a segment is a cut of N_π, its poset must contain a reference set for every event e of segment $S^{(l)}$ (in fact, for every event in segments $S^{(l')}$ if $l' \geq l$). For convenience, if event $e \in S^{(l')}(l' \geq l)$, let us denote by $R(e, l)$ such a reference set. An upper bound ${U_\gamma}^{(m)}(s, t, 0)$ on TSE $\gamma^{(m)}(s, t, 0)$ is determined by (7) ([16]).

$${U_\gamma}^{(m)}(s, t, 0) = \max_{e \in R(t^{(m)}, l)} [\delta^*(e, t^{(m)}) - \delta^*(e, s^{(m)})] \quad (7)$$

Note that the above upper bound ${U_\gamma}^{(m)}(s, t, 0)$ is independent of the occurrence times of the events in the reference set of $s^{(m)}$. In other words, it does not depend on the history of the timed execution prior to segment $S^{(l)}$. Applying a longest path analysis from a fixed event e as outlined above, the term $\delta^*(e, t^{(m)})$ in (7) is computed in $O(|T(S)| + |P(S)|)$ time where $|P(S)|$ is the number of places in S. Thus, ${U_\gamma}^{(m)}(s, t, 0)$ is computed in $O((|T(S)| + |P(S)|) * |R|)$ time where $|R|$ is the size of the referent set of e. This way, ${U_\gamma}^{(m)}(s, t, 0)$ is computed in $O(|T(S)| + |P(S)|)$ time.

3.3 Bounding and evaluating TSE statistics

As pointed out earlier, the structure of segments are independent. However, multiple TSEs due to the same separation triple may start in one segment. Thus, these TSEs can be dependent on each other. To overcome this dependency, we treat all the TSE instances starting from one segment as a *group* and translate the problem of bounding the average TSE to that of bounding the average grouped TSE [16, 15]. In addition, extensions to variance and other higher order statistics of the TSE bounds can also be similarly proved. For the purposes of this paper, we refer to the derived bound of any statistic (average, variance, etc..) of a TSE $\gamma(s, t, e)$ as $\mathcal{S}_\gamma(s, t, e)$. Evaluation of a statistic $\mathcal{S}$ can be performed via Monte-Carlo simulation (e.g.,[8]) in which each independent segment yields one sample of $\mathcal{S}_\gamma(s, t, e)$.

4. REDUCED PETRI NET AND ITS TSE STATISTICS

One of the goals of this work is to take a PTPN N and reduce it to a smaller net N'. The key requirement is that N' should preserve all TSE statistics related to those signals also in N. We now introduce a class of reduced Petri nets for which we will prove that this performance-preservation property is guaranteed.

Definition 2. Reduced Petri Net: *A reduced Petri net $N' = (P', T', F')$ is a reduced Petri net of $N = (P, T, F)$ if,*

1. $T' \subseteq T$
2. $P'_c \equiv P_c$
3. $M'_0(p) = M_0(p)$, $\forall p \in P_c$
4. $(t, p) \in F \Leftrightarrow (t, p) \in F'$, $\forall p \in P_c$ $\forall t \in T'$
5. $(p, t) \in F \Leftrightarrow (p, t) \in F'$, $\forall p \in P_c$ $\forall t \in T'$
6. $\forall (s, t) \in (T', T')$, $\forall k$,

$$\max_{\psi' \in \Psi_{N'}(s,t), M(\psi')=k} \Delta(\psi') = \max_{\psi \in \Psi_N(s,t), M(\psi)=k} \Delta(\psi)$$

As an example, the reduced Petri net of the net in Figure 1(a) is depicted in 2(a). The key property of a reduced Petri net is the last property in Definition 2. It states that the maximum delay along any path between two transitions in the reduced net that has k tokens must be identical to the maximal delay of any path between those transitions in the original net that has the same number of tokens.

To prove the performance-preservation property, we must introduce a few definitions. For a Petri net N and its reduced Petri net N', we call N_π and N'_π *corresponding event graphs* if the sequence of decisions is identical in N_π and N'_π, i.e., $p^k\bullet \in T_\pi = p^k\bullet \in T'_\pi$, $\forall p \in P_c$. A segment $S'(\xi', \theta')$ is a *corresponding segment* of the segment $S(\xi, \theta)$ if the partial order of S' is preserved in the partial order of S, i.e., $\theta' \subseteq \theta$.

Next, we need to define the following partial order operators.

Definition 3. *For a set of events E and an event e,*

$$e \preceq E \Leftrightarrow [(e \in E) \vee (\exists (e, e_i), e_i \in E)] \wedge [\neg(\exists (e_j, e), e_j \in E)]$$
$$e \succeq E \Leftrightarrow [(e \in E) \vee (\exists (e_i, e), e_i \in E)] \wedge [\neg(\exists (e, e_j), e_j \in E)]$$
$$e \prec E \Leftrightarrow (e \preceq E) \wedge (e \notin E)$$
$$e \succ E \Leftrightarrow (e \succeq E) \wedge (e \notin E)$$

Definition 4. *For a pair of sets of events E_1 and E_2,*

$$E_1 \preceq E_2 \Leftrightarrow (\forall e_i \in E_1, e_i \preceq E_2) \wedge (\forall e_j \in E_2, e_j \succeq E_1)$$
$$E_1 \succeq E_2 \equiv E_2 \preceq E_1$$
$$E_1 \prec E_2 \Leftrightarrow (E_1 \preceq E_2) \wedge (E_1 \cap E_2 = \emptyset)$$
$$E_1 \succ E_2 \equiv E_2 \prec E_1$$

To prove the theorem we establish that each statistical sample of the TSE obtained from the reduced net has an equally likely sample in the original net. To do this we show that there is one-to-one correspondence of partitioning segment sequences in the original and reduced nets in Lemma 1. We then show that the samples of the upper and lower bounds obtained from the sequences in the reduced net can be no tighter than one that can be obtained from the corresponding segment sequences of the original net in Lemma 2. The proof relates the computation of these bounds to the delay of critical paths in the reduced net and uses the last property of Definition 2 to guarantee that their delay is equal to the delay of corresponding paths in the original net. In particular, the critical paths consist of those paths from transitions of corresponding reference events to the pairs of transitions in the TSE.

Lemma 1. *For any sequence of segments which partitions N'_π, there exists a sequence of segments which partitions N_π such that the k-th segment in N'_π corresponds to the k-th segment in N_π where N' is the reduced Petri net of N, and N'_π is the corresponding event graph of N_π.*

Proof (Sketch) Let ξ'^k be the k-th cut and θ'^k be the partial order of the k-th segment where the k-th segment, $S'^k(\xi'^k, \theta'^k)$, is a segment between ξ'^k and ξ'^{k+1}. We have $\theta'^k = \{(s,t) | s \in T'^k \wedge t \in T'_\pi$ where T'^k is a set of events in a segment $S'^k\}$. Let $\bigcup_o^k$ and $\bigcup_o'^k$ be $\forall t \in \bigcup_{\forall p \in \xi^k} p\bullet$ and $\forall t \in \bigcup_{\forall p \in \xi'^k} p\bullet$ and $\bigcup_r^k$ and $\bigcup_r'^k$ be $\forall t \in \bigcup_{\forall p \in \xi^k} \bullet p$ and $\forall t \in \bigcup_{\forall p \in \xi'^k} \bullet p$. Figure 1(b) and 2(b) shows examples of $\bigcup_o^k$, $\bigcup_r^k$, $\bigcup_r'^k$ and $\bigcup_o'^k$.

For any ξ'^k of N'_π, there exists a ξ^k of N_π such that $\bigcup_r'^k \preceq \bigcup_r^k \prec \bigcup_o^k \preceq \bigcup_o'^k$ because N_π and N'_π are corresponding event graphs and $T'_\pi \subseteq T_\pi$. Thus, for any k-th segment of N'_π, there exists the k-th segment of N_π which yields a one-to-one mapping between segments in N'_π and N_π.

From the definition of the corresponding segments, if $\theta'^k \subseteq \theta^k$, then the one-to-one mapping maps corresponding segments. Since $\bigcup_o^k \preceq \bigcup_o'^k \preceq T'_k \preceq \bigcup_r'^{k+1} \preceq \bigcup_r^{k+1}$ and partial order operators are transitive in set-to-set operations, we have that $T'^k \subseteq T^k$. From Definition 2, we have $\theta'_\pi \subseteq \theta_\pi$. $\theta'^k = \{(e_i, e_j) | e_i \in T'^k, e_j \in T'_\pi\}$ and $\theta^k = \{(e_i, e_j) | e_i \in T^k, e_j \in T_\pi\}$. Thus, $\theta'^k \subseteq \theta^k$ since $\theta'_\pi \subseteq \theta_\pi$, $T'^k \subseteq T^k$ and $T'_\pi \subseteq T_\pi$.

As an example, the segment $S'^k(\xi'^k, \theta'^k)$ of the reduced Petri net shown in Figure 2(b) is the corresponding segment of the segment $S^k(\xi^k, \theta^k)$ of the original Petri net shown in Figure 1(b).

Lemma 2. *Let S'^k and S^k be the k-th segments of corresponding event graphs N'_π, N_π, respectively. Then, any upper (lower) bound ${U'_\gamma}^{(k)}(s,t,\epsilon)$ (${L'_\gamma}^{(k)}(s,t,\epsilon)$) on a TSE instance of a pair of events (s,t) belonging to the segment S'^k is a bound on the TSE instance $\gamma^{(k)}(s,t,\epsilon)$ belonging to the segment S^k.*

Proof (Sketch)

For any reference set, R'^k, of a target event t in S'^k, there exists a reference set, R^k, of the target event t in S^k such that $R^k \preceq R'^k$ because $\bigcup_o^k \preceq \bigcup_o'^k \preceq R'^k$. For example, we can choose $R^k = \bigcup_o^k$. Thus, for any events $r \in R^k$, we have

$$(r \in R^k) \Leftrightarrow (r \in R'^k) \vee \exists (r, r') \in \theta^k, r \in R, r' \in R'$$

Case 1. $r \in R'^k$

$$\delta^*(r,t) = \delta'^*(r,t)$$
$$\delta^*(r,s) = \delta'^*(r,s)$$
$$\delta^*(r,t) - \delta^*(r,s) \leq {U'_\gamma}^{(k)}(s,t,\epsilon) \quad (8)$$

Case 2. $\exists (r, r') \in \theta^k, r \in R, r' \in R'$

$$\delta^*(r,t) = \max_{r' \in R_s'^k} (\delta^*(r,r') + \delta^*(r',t)) \quad (9)$$
$$\delta^*(r,s) = \max_{r' \in R_s'^k} (\delta^*(r,r') + \delta^*(r',s)) \quad (10)$$

Let r'_j and r'_i be such that,

$$\delta^*(r,t) = \delta^*(r,r'_j) + \delta^*(r'_j,t) \quad (11)$$
$$\geq \delta^*(r,r'_i) + \delta^*(r'_i,t) \quad (12)$$
$$\delta^*(r,s) = \delta^*(r,r'_i) + \delta^*(r'_i,s) \quad (13)$$
$$\geq \delta^*(r,r'_j) + \delta^*(r'_j,s) \quad (14)$$

From equations (11) and (14)

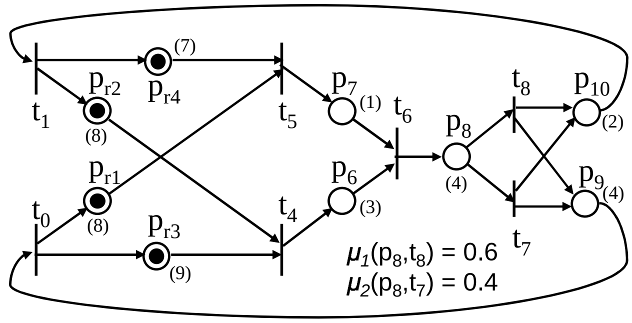
$$\delta^*(r,t) - \delta^*(r,s) \leq \delta^*(r,r'_j) + \delta^*(r'_j,t) - \delta^*(r,r'_j) - \delta^*(r'_j,s)$$
$$\leq \delta^*(r'_j,t) - \delta^*(r'_j,s) \quad (15)$$

From (15),

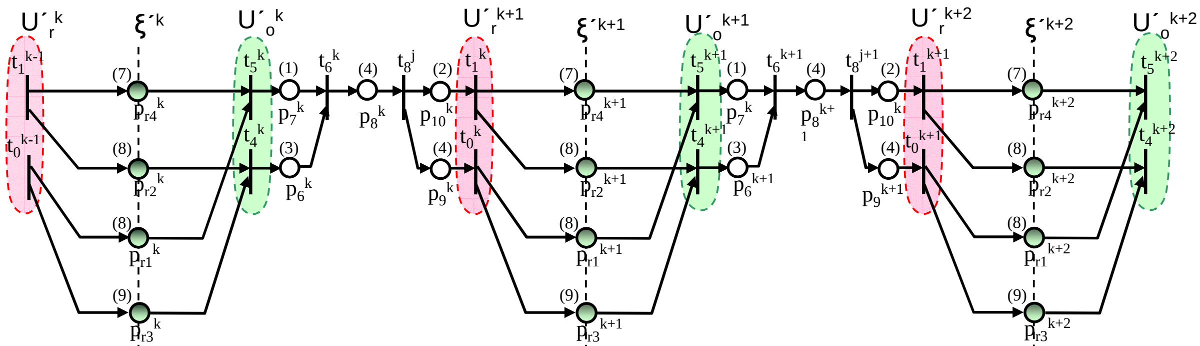
$$\delta^*(r,t) - \delta^*(r,s) \leq \delta^*(r'_j,t) - \delta^*(r'_j,s) \leq {U'_\gamma}^{(k)}(s,t,\epsilon) \quad (16)$$

Since the above equation is true for every $r \in R^k$, ${U_\gamma}^{(k)}(s,t,\epsilon) \leq {U'_\gamma}^{(k)}(s,t,\epsilon)$. Similarly, the following inequality, $L^k(s,t) \geq L'^k(s,t)$, can easily be proved through the duality of bounds. Thus, any bound on a TSE instance of a pair of events belonging to the segment S'^k is a bound on the TSE instance belonging to the segment S^k.

To demonstrate a concrete example of this proof, consider a TSE instance ${\gamma'}^{(k)}(t_6, t_6, 1)$ obtained by sampling from the sequence of segments $S'^k(\xi'^k, \theta'^k)$ and $S'^{k+1}(\xi'^{k+1}, \theta'^{k+1})$ in the reduced net shown in Figure 2(b). Repetitive application of Equation 1 yiels ${U'_\gamma}^{(k)}(t_6, t_6, 1) = 20$. The proof argues that this upper bound is also valid for the original net shown in Figure 1(b). Repetitive application of Equation 1 produces the same value, 20, which proves that the obtained value in the reduced net is valid.



(a) A reduced Petri net



(b) A timed execution of the Petri net in (a)

Figure 2: A reduced Petri net of the net in Figure 1(a) and its timed execution.

Theorem 1. *Let N' be a reduced Petri net of N. Then, any bound of a TSE statistic in N' defined by $\mathcal{S}'(s,t,e)$ is a bound of the corresponding TSE statistic in N.*

Proof (Sketch) From Lemma 1, there exists a sequence of segments which partitions N_π such that k-th segment in N'_π is a corresponding k-th segment in N_π. From extensions of Lemma 2 to grouped TSEs, we know that each sample of $\mathcal{S}'(s,t,e)$ is also a bound for the corresponding TSE statistic in N. Thus, the resulting probabilistic bound obtained by the Monte-Carlo simulation for $\mathcal{S}'(s,t,e)$ is also valid for N.[1]

5. PETRI NET REDUCTION OPERATIONS

In this section, we propose two reduction operations and prove that the modified Petri net through those operations is a reduced Petri net of the original net.

5.1 Projection

Reduction Rule 1.(Projection)
Precondition: There exist two sets of transitions T_i, T_o, two sets of places P_i, P_o, and a transition t_d.

1. $|\bullet p| = |p\bullet| = 1$, $\forall p \in P_i$ and P_o.
2. $(s,p) \in F$ and $(p,t_d) \in F$ where $s \in T_i$, $\forall p \in P_i$
3. $(t_d,p) \in F$ and $(p,t) \in F$ where $t \in T_o$, $\forall p \in P_o$

Rule: For every pair of places (p_i,p_o), $p_i \in P_i, p_o \in P_o$, create new place $p_n \in P_n$ such that $d(p_n) = d(p_i) + d(p_o)$, $M(p_n) = M(p_i) + M(p_o)$, $\bullet p_n = \bullet p_i$ and $p_n\bullet = p_o\bullet$. Remove $p_i \in P_i, p_o \in P_o$ and t_d.

After applying projection operation to the Petri net N, the modified Petri net N' is a reduced Petri net of N. Since only transition t_d is removed from N, $T' \subseteq T$ and the places with choice are not considered in this operation, hence properties 1-5 of the reduced Petri net are satisfied. Furthermore, for any pair of transitions (t_i,t_o) where $t_i \in T_i$ and $t_o \in T_o$, there exists a path from t_i to t_o with equal path delay and number of initial markings. Thus all properties of the reduced Petri net are satisfied. Figure 3 illustrates an example of the projection operation.

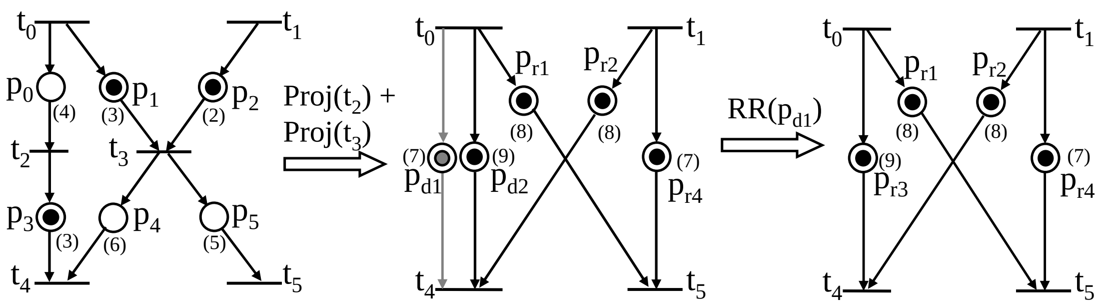
5.2 Redundancy removal

Reduction Rule 2.(Redundancy Removal)
Precondition: There exist a transitions t_i and a transition t_o and a place p_d.

1. $\bullet p_d = \{t_i\}$ and $p_d\bullet = \{t_o\}$
2. $\exists \psi \in \Psi(t_i,t_o)$ such that $p_d \notin \psi$, $\Delta(\psi) \geq d(p_d)$ and $M(\psi) = M(p_d)$.

Rule: Remove p_d.

[1]Note, however, that the obtained bounds of TSE statistics of N_π and N'_π may still be numerically different due to differences in cut and/or the reference set selection.



Proj(t2): $T_i = \{t_0\}$; $T_o = \{t_4\}$; $P_i = \{p_0\}$; $P_o = \{p_3\}$; $P_n = \{p_{d1}\}$
$d(p_{d1}) = d(p_0) + d(p_3)$; $M(p_{d1}) = M(p_0) + M(p_3)$;
Proj(t3): $T_i = \{t_0, t_1\}$; $T_o = \{t_4, t_5\}$; $P_i = \{p_1, p_2\}$; $P_o = \{p_4, p_5\}$; $P_n = \{p_{d2}, p_{r1}, p_{r2}, p_{r4}\}$
$d(p_{d2}) = d(p_1) + d(p_4)$; $d(p_{r1}) = d(p_1) + d(p_5)$; $d(p_{r2}) = d(p_2) + d(p_4)$; $d(p_{r4}) = d(p_2) + d(p_5)$
$M(p_{d2}) = M(p_1) + M(p_4)$; $M(p_{r1}) = M(p_1) + M(p_5)$; $M(p_{r2}) = M(p_2) + M(p_4)$; $M(p_{r4}) = M(p_2) + M(p_5)$
RR(p_{d1}): $t_i = t0$; $t_o = t4$ and $d(p_{d2}) > d(p_{d1})$, $M(p_{d2}) = M(p_{d1})$

Figure 3: Projection of t_2, t_3 as well as redundancy removal of p_{d1}.

After applying redundancy removal operation to the Petri net N, a modified Petri net N' is a reduced Petri net of N. Since all transitions remains the same and the places with choice are not considered in this operation, properties 1-5 of the reduced Petri net are satisfied. Furthermore, for a pair of transitions (t_i, t_o), there exists a path from t_i to t_o with equal path delay and number of initial markings. Thus, all properties of the reduced Petri net are satisfied. Figure 3 illustrates an example of projection operation.

6. CASE STUDY

We now demonstrate the power of the proposed reduction operations on a well-known 4-phase Petri net model of a 3-stage PCHB linear pipeline illustrated in Figure 4. To semi-automate reduction, we developed a C program that would check the validity of user-proposed projection and redundancy removal reductions and if valid return the reduced Petri net. Our future work includes developing heuristics to guide the application of these reduction operations, thereby creating a more fully-automated reduction tool.

Table 1 shows the number of places/transitions in the input model and the percentage of places/transitions that were removed compared to the original model due to the proposed reductions. As shown in Table 1, after the proposed series of reductions, the number of places were reduced by more than half and the number of transitions were reduced by a factor of 6. Thus, using this model in place of its more complicated orignal model for system-level performance analysis would yield a significant run-time improvement.

In addition, the reduced Petri net highlights performance characteristics and dependencies not obvious in the original, more complicated, net. For example, R_i is the typical forward latency, $B1_i$ is the typical reverse latency, and $B2_i$ is a special reverse latency typical of pipelines based on half-buffer templates. $B2_i$ highlights the dependency across three pipeline stages that was not obvious from the more detailed Petri nets or, for that matter, the original description of PCHB in [7].

	# places	reduc$_p$	# transitions	reduc$_t$
PCHB model	37	0%	24	0%
Intermediate I	26	35%	12	50%
Intermediate II	22	41%	8	67%
Reduced model	15	59%	4	83%

Table 1: An Example of Petri net reduction on 3-stage PCHB model.

7. CONCLUSION AND FUTURE WORK

This paper characterizes a class of Petri net reduction operations that preserve TSE statistics as well as two useful operators that fall into this class. We applied these two operations using a novel semi-automated Petri net reduction tool to a Petri net model of well-known asynchronous pipeline. The resulting reduced Petri net is both significantly and adds insight into its performance characteristics. Our tool can be used to obtain performance models for libraries of asynchronous cells on which system-level performance analysis tools can be applied. Potential future work includes the development of more reduction operations, the further automation of the tool, and the expansion of our work to more general stochastic Petri nets.

8. REFERENCES

[1] M. Benes, S. M. Nowick, and A. Wolfe. A fast asynchronous Huffman decoder for compressed-code embedded processors. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 43–56, 1998.

[2] E. Best. Partial order behavior and structure of Petri nets. *Formal Aspects of Computing*, 2:123–138, 1990.

[3] Steven M. Burns. *Performance Analysis and Optimization of Asynchronous Circuits*. PhD thesis, California Institute of Technology, 1991.

[4] Jo Ebergen and Robert Berks. Response time properties of linear asynchronous pipelines.

Proceedings of the IEEE, 87(2):308–318, February 1999.

[5] H. Hulgaard, S. M. Burns, T. Amon, and G. Borriello. An algorithm for exact bounds on the time separation of events in concurrent systems. *IEEE Transactions on Computers*, 44(11):1306–1317, November 1995.

[6] Joep Kessels and Paul Marston. Designing asynchronous standby circuits for a low-power pager. *Proceedings of the IEEE*, 87(2):257–267, February 1999.

[7] Andrew M. Lines. Pipelined asynchronous circuits. Master's thesis, California Institute of Technology, 1996.

[8] I. R. Miller, J. E. Freund, and R. Johnson. *Probability and Statistics for Engineers*. Prentice Hall, 1990.

[9] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77:541–580, April 1989.

[10] Chris J. Myers and Teresa H.-Y. Meng. Synthesis of timed asynchronous circuits. *IEEE Transactions on VLSI Systems*, 1(2):106–119, June 1993.

[11] Recep O. Ozdag and Peter A. Beerel. High-speed QDI asynchronous pipelines. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*. IEEE Computer Society Press, April 2002.

[12] Shai Rotem, Ken Stevens, Ran Ginosar, Peter Beerel, Chris Myers, Kenneth Yun, Rakefet Kol, Charles Dike, Marly Roncken, and Boris Agapiev. RAPPID: An asynchronous instruction length decoder. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 60–70, April 1999.

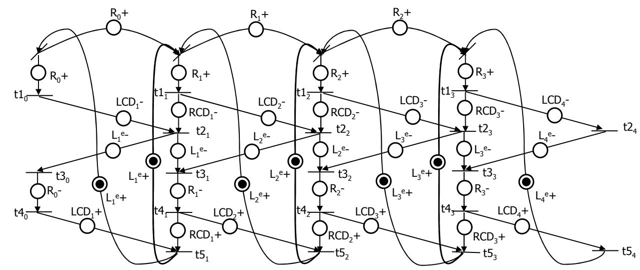
[13] Montek Singh and Steven M. Nowick. High-throughput asynchronous pipelines for fine-grain dynamic datapaths. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 198–209. IEEE Computer Society Press, April 2000.

[14] Hiroaki Terada, Souichi Miyata, and Makoto Iwata. DDMP's: Self-timed super-pipelined data-driven multimedia processors. *Proceedings of the IEEE*, 87(2):282–296, February 1999.

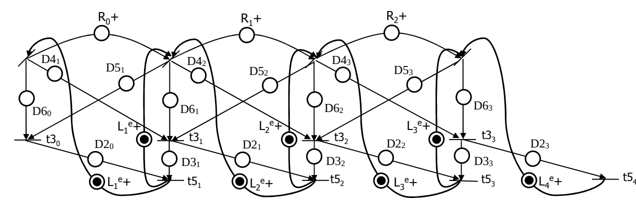
[15] A. Xie and P. A. Beerel. Performance analysis of asynchronous circuits and systems using stochastic timed Petri nets. In A. Yakovlev, L. Gomes, and L. Lavagno, editors, *Hardware Design and Petri Nets*, pages 239–268. Kluwer Academic Publishers, March 2000.

[16] Aiguo Xie, Sangyun Kim, and Peter A. Beerel. Bounding average time separations of events in stochastic timed Petri nets with choice. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 94–107, April 1999.

[17] A. Yakovlev and A. M. Koelmans. Petri nets and digital hardward design. In *Lectures on Petri nets II: Basic Models*, Lecture Notes in Computer Science. Springer-Verlag, 1998.



(a) Initial PCHB model



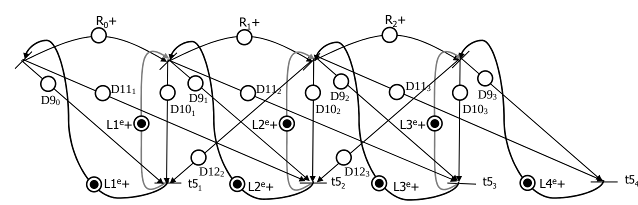
Proj(t4$_i$): $D2_i = R_i- + LCD_{i+1}+$; $D3_i = R_i- + RCD_i+$

Proj(t1$_i$): $D0_i = R_i+ + LCD_{i+1}-$; $D1_i = R_i+ + RCD_i-$;

Proj(t2$_i$): $D4_i, D6_i = D0_i + L_i^e-$; $D5_i, D7_i = D1_i+ + L_i^e-$;

RR(D7$_i$): $D6_{i+1} = \max(D7_i, D6_{i+1})$

(b) Intermediate model I

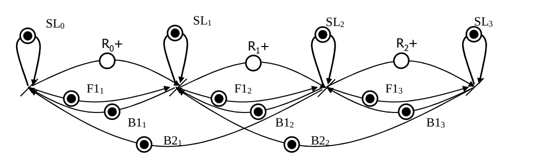


Proj(t3$_i$): $D9_i = D8_i + D2_i$; $D14_i = D4_i + D3_i$; $D10_i = D8_i + D3_i$; $D13_i = D5_{i+1} + D2_i$
$D11_i = D4_i + D2_i$; $D12_i = D5_i + D3_{i-1}$

RR(D14$_{i+1}$): $D9_i = \max(D9_i, D14_{i+1})$

RR(D13$_{i-1}$): $D10_i = \max(D10_i, D13_{i-1})$

(c) Intermediate model II



RR(L1^e+): Remove all gray edges since $L_i^e+ \le R_{i-1} + L_i^e+$

Proj(t3$_i$): $F1_i = D11_i + L_{i+1}{}^e+$; $B1_i = D10_i + L_i^e+$; $B2_i = D12_{i+1} + L_i^e+$; $SL_i = D9_i + L_{i+1}{}^e+$;

(d) Reduced PCHB model

Figure 4: Reduction of 3-stage PCHB pipeline model.