Verification of asynchronous circuits using Time Petri Net unfolding

Alexei Semenov* Alexandre Yakovlev†
Department of Computing Science, University of Newcastle
Newcastle upon Tyne NE1 7RU, England
e-mail: {alex.semenov, alex.yakovlev}@newcastle.ac.uk

Abstract— This paper describes a novel approach to timing analysis and verification of asynchronous circuits with bounded delays. The method is based on the *time-driven unfolding* of a time Petri net model of a circuit. Each reachable state, together with its timing constraints is represented implicitly. Our method is used to verify freedom from hazards in asynchronous circuits consisting of micropipeline components and logic gates.

I. Introduction

Verification of functional correctness is widely accepted today as an important step of the asynchronous circuit design process. Circuits are often designed in an *ad hoc* manner, It is impossible to guarantee correct operation of a circuit under all possible conditions. Most of the current research in asynchronous circuit verification is aimed at the design and verification of speed-independent (SI) or (quasi) delay insensitive ((Q)DI) circuits.

Additional circuitry for pure SI or (Q)DI design often discourages the designers from using self-timed models. Instead, designers often choose some short-cuts, implement circuits so that the delays in gate switching prevent the circuit from malfunctioning. As a result, bounded delay circuits need to be verified by means of efficient timing analysis.

Several methods for analysis of timed models of asynchronous circuits are known to date, e.g. Alur and Dill [2], Rokicki [8], Hulgaard et al [5], Yoneda et al [12] and others. The timed automaton model [2], due to its generality, can provide a very fine detail description of its behaviour but for obvious reasons of complexity it may be impractical. The approach in [12] builds a Time State Graph (TSG) of an asynchronous circuit from its Time PN description and attempts to use a partial order technique. An approach used in [8] is based on constructing a TSG from a special class of Timed PNs, orbital nets, using a partial order approach to calculate the states reachable by firing of the transitions in non-conflicting run of underlying PN. Timing analysis introduced by [5] is based on an untimed unfolding of underlying PN and uses an iterative algorithm to determine separation times between events.

We propose an algorithm for efficient implicit "construction" of the TSG for a Time PN. The latter is represented in the form of the Time PN unfolding segment. Our algorithm unfolds a Time PN creating only those transitions that are really instantiated within given timing bounds. An active role in this construction is played by timing constraints re-calculated dynamically for the transition instances in the unfolding. That latter feature differentiates our approach from [5], where the net unfolding is built separately from its timing analysis.

We consider Time PNs with time independent choice and apply our method to the verification of asynchronous control circuits built in two different design styles: micropipeline components and logic gates.

II. Basic definitions

Time Petri Nets — A *marked Petri Net (PN)*[1] is a tuple $N = \langle P, T, F, m_0 \rangle$ where P and T are non-empty finite sets of places and transitions respectively, F is a flow relation and m_0 is the initial marking. A *Labelled PN* is a tuple $N_l = \langle N, A, L \rangle$ where N is a marked PN, A is a set of actions and $L : T \rightarrow A$ is a labelling function. If $|T| > |A|$ then some transitions of LPN are labelled with the same action a. A special case of LPN, used for low level descriptions of asynchronous circuits, is called *Signal (Transition) Graph* (STG) [9]. In STG, the set of actions represents changes of the signals. Two signals are said to be *persistent* at some marking if transition of one signal does not disable transition of another. A PN is said to be *safe* if at any reachable marking the number of tokens in any place is not greater than 1. Further in this paper we will consider only safe PNs.

A *time Petri Net* **(TPN)** (introduced in [7]) is a tuple $\mathcal{N} = \langle N, \mathcal{E}, \mathcal{L} \rangle$, where N is a marked PN, $\mathcal{E}$, $\mathcal{L} : T \rightarrow Q^+$ are functions assigning each transition with earliest and latest firing times. For each transition $t \in T$ $\mathcal{E}(t) \leq \mathcal{L}(t)$ is satisfied. A *state of TPN* is a pair (m, cl_m). Function $cl_m : T \rightarrow Q^+$ associates a rational number with each transition representing the time elapsed from the moment when this transition became enabled. A transition t of a TPN is said to be *time-enabled* at some state s iff it has all its input places ($\bullet t$) marked (untimed enabled) at marking m and $\mathcal{E}(t) \leq cl_m(t) \leq \mathcal{L}(t)$. The state of a TPN can change due to two reasons:

- A transition fires, which takes no time but changes the marking (removing tokens from $\bullet t$ and adding to $t\bullet$) and updates the *cl* function excluding disabled transitions and adding newly enabled ones, or
- Some time (bounded by the latest firing times of enabled transitions) passes which does not change the marking but updates the clock values for all transitions.

In order to be able to have a finite representation of its time state space the notion of classes is introduced. A class is a tuple $s = (m, I)$, where m is some marking of

*Supported by the grants of Newcastle University's from the Research Committee and CVCP.
†Partially supported by ESPRIT LTR Project DeVa (Grant 20072).

[1] We assume that the reader is familiar with the basics of PN theory.

33rd Design Automation Conference ®

DAC 96 - 06/96 Las Vegas, NV, USA

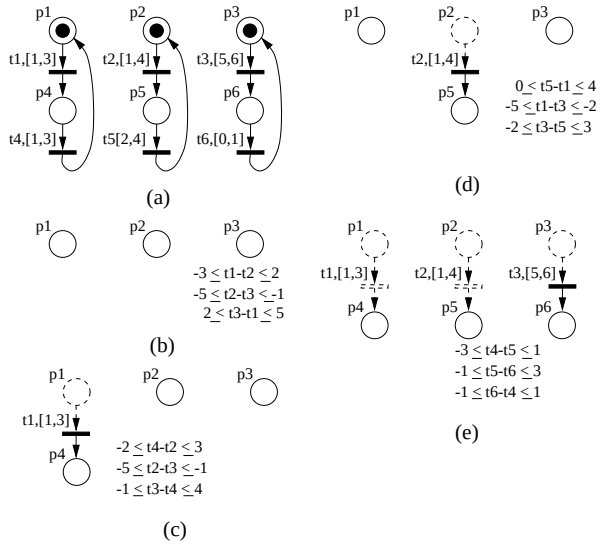

Fig. 1. An example of a Time Petri Net and its unfolding.

the underlying PN and I is a set of inequalities relating firing times of enabled transitions. Abusing the notation we will call classes of time states simply *time states* (TS).

An algorithm for generating a Time State Graph (TSG) [3] has been adapted for analysis of asynchronous circuits in [12]. The algorithm constructs TSG iteratively by applying Floyd's algorithm to the system of inequalities represented in the form of a matrix.

Analysis of the TSG hits the problem of state explosion. An attractive method of battling this problem was suggested in [6] which we extend to analysis of TPNs.Here we only briefly introduce the unfolding; with some useful notions and notation associated with it.

Unfoldings — Formally, an unfolding obtained from a PN, N, is an *occurrence* net $N' = \langle P', T', F', L' \rangle$ where P', T' and F' are its sets of places, transitions and flow relation respectively, and $L' : (P' \cup T') \rightarrow (P \cup T)$ is the labelling function which labels every unfolding element as an occurrence of the corresponding element of the original PN.

Using structural properties of the unfolding we can determine relations between instances and define the notions of (local) configuration and its final state[2].

For any reachable marking m of the original PN there exists a configuration in the unfolding such that its final state is equal to m and vice versa [6].

An algorithm constructing a truncated unfolding representing all reachable states has been suggested in [6]. The possible redundancy of the truncated unfolding has been discussed and solved in [10] and [4]. However, in this work, we are not concerned with the cutoff condition itself and will assume that the right cutoff condition is used for our examples.

Choice in TPNs — There are two paradigms of choice interpretation in TPN. The first assumes that, for any two enabled transitions that are in structural conflict choice is resolved instantly and then the winner's firing is delayed, i.e. the choice is resolved *independently of time*. In the second, the clock of each transition is started when it is PN (untimed) enabled and transitions compete for the token(s) in the conflict place(s) resolving the conflict in time. Thus the allowed firing times of a particular transition may be different from the assigned ones.

From a behavioural point of view, there are three basic types of choice in PNs:(Extended) Free Choice[3] (FC), Unique Choice (UC) and Arbitration choice (AC). In FC PNs, the choice between two transitions is always nondeterministic but fair to all transitions. In UC PNs, the choice is controllable; there is only one of structurally conflicting transitions enabled at a time. In AC PN, the choice is uncontrollable; transitions may be in structural choice, but at different markings may be in conflict with different transitions. This is the most difficult type of choice for analysis.

A *time independent choice* (**TIC-PN**) as a TPN in which any choice is resolved independently of time.

If conflicting transitions of a FC TPN have ranges of their delays which are either equal or non-overlapping then such PN is a TIC-PN. If the delay ranges are different but are overlapping, then this choice can be made time independent by either using the first paradigm or by introducing additional transitions with zero delay range which will resolve the choice. UC PNs are included into TIC-PNs. The choice in AC is time independent only if all input places are marked at the same time or are marked mutually exclusive. In the first case this type of conflict can be reduced to FC, in the second – to UC.

III. Time Petri Net Unfolding

The unfolding algorithm for untimed PNs cannot be applied directly to the analysis of TPNs. Unlike in untimed PNs, two interleavings of concurrent transitions may lead into two different time states.

Consider a TPN shown in Figure 1. An invisible action resets the whole system inserting tokens into all places of the initial marking at the same time. Using method explained in [12] we compute the initial set of inequalities I_0, represented as a matrix where each element I_{ij} is an upper bound of firing transition t_i earlier than t_j. All diagonal elements are set to 0. To check if an instance of t_1 is time-enabled we check that $I_{0_1,i} \geq 0$ for all t_i enabled at m_0. The new set of inequalities is computed using Floyd's algorithm as in [12]. For the purposes of our verification we restrict ourselves to the class of TIC-PNs.

Similar to the untimed unfolding we define the following "cornerstone" notions of TPN unfolding.

- A *TPN unfolding* is an occurrence net $\mathcal{N}' = \langle N', \mathcal{I} \rangle$ build from a TPN so that all transition firings satisfy timing constraints imposed on them.
- A *time configuration*[4] is a non-conflicting backwards closed set of transitions of a TPN unfolding. The least time configuration including transition t' is called *time local configuration* (TLC) of t'. It may correspond to some configuration of the unfolding of underlying PN if any "untimed" concurrent to t' transition is time-preceding t'. For example, TLC of t'_3 (Figure 1(e)) in TPN unfolding of TPN from Figure 1(a) includes both t'_1 and t'_2.

[2] See [6] for full definitions and explanations.

[3] FC is also a structural property.

[4] We will simply reuse notation of untimed unfolding inscribing it with ^{t}.

```
while QUEUE is not empty do
  for each transition t in PN N do
    find an untried subset of independent occurrences of •t
    compute a set of pre-firing time states of t'
    for each pre-firing timed state F_s^t(t') do
      if t' is time-enabled then do
        insert F_s^t(t') into the Queue in order of Lo(t')
      end do
    end do
  end do
  pull the first time state F_s^t(t') from the QUEUE
  compute new set of inequalities I.
  make a copy t' of transition t in the UNFOLDING
  if t' is a cutoff then do
    mark transition t' and its t'• as cutoff point
  end do
end do
```

Fig. 2. Algorithm for TPN unfolding segment.

A particular instance t' can be associated with a set of TLCs.

• A *timed final state* of configuration C^t is a pair (m, I), where m is a marking reached by firing transitions in C^t and I is a set of inequalities associated with it. The set of inequalities depends on the order in which transitions in C^t are fired.

A TLC represents a (possibly infinite) set of timed firing sequences which all lead to the states of TPN reachable through the first possible firing of the transition t' in every sequence. Each transition of the TPN unfolding is associated with two time bounds: $Lo(t')$ and $Hi(t')$ representing the earliest and latest times respectively at which transition t can fire from the start of the system.

Proposition 1 Two transitions t_1 and t_2 of TIC-TPN are in conflict if there exist two of their instances t'_1 and t'_2 in TPN unfolding such that instances of some transitions $t'_3 \in \lceil t'_1 \rceil^t$ and $t'_4 \in \lceil t'_2 \rceil^t$ are in structural conflict. □

From the above proposition and structural precedence of the unfolding elements we can determine conflict, concurrency and precedence relations between any two instances. Thus we can check the safeness property of TPNs on their TPN unfolding by simply checking if any two occurrences of the same place of the original net are independent.

Due to non-conflictness property of a local configuration any sequence of transitions in a TLC leads to the same PN marking. Moreover, this marking is always reachable through some firing sequence represented in this TLC because the timing constraints imposed on transition firings can only restrict reachable markings by ordering concurrent events.

The above notions allow us to develop an algorithm for construction of a TPN unfolding of a TIC-PN. The algorithm constructs a possibly infinite time unfolding of a TIC-PN. The candidate transitions are ordered according to the their earliest time of firing. Unlike in [6], the algorithm orders time final states ($F_s^t(\lceil t' \rceil^t)$). In an untimed unfolding, every instance had a uniquely defined final state whereas in a TPN unfolding each transition may have several time final states associated with it.

An important step of the algorithm is computing the pre-firing TS for t', which is considered in more detail below.

• Suppose for simplicity that each transition in $\bullet(\bullet t')$ has only one TLC[5]. Pre-firing configuration $\bullet\lceil t' \rceil^t$ is found as a union of local configurations of $t'_i \in \bullet(\bullet t')$. The pre-firing marking of t is $F_s^t(\bullet\lceil t' \rceil^t)$.

• Firing time bounds of a newly generated transition are computed from the firing bounds of the maximal transitions of $\bullet\lceil t' \rceil^t$ as: $Lo(t') = max(Lo(t'_m)) + \mathcal{E}(t')$ and $Hi(t') = max(Hi(t'_m)) + \mathcal{L}(t')$ $\forall t'_m \in \bullet\lceil t' \rceil$.

From the pre-firing configuration, using its structural information and the firing bounds of each transition a matrix M, corresponding to the pre-firing configuration set of inequalities is created.

• By applying Floyd's algorithm obtain the set of inequalities $\bullet I$ for the state covering timed states from which t' can be fired. Let k be the row corresponding to t. If $\forall i : M_{k,i} \geq 0$, then the t is time-enabled. Otherwise, find a transition t_e with minimum $M_{k,e}$ and check if its occurrence exists in the time unfolding. Intuitively, this corresponds to firing the transition with the most "time penalty", i.e. earliest needed to be fired to time enable t. If several transitions have equal time penalties, choose the one which has looser bounds w.r.t. t'_m. If no instance of this transition exists in the unfolding or if t is no longer enabled after firing t_e, then the algorithm proceeds to analysis of the next time final state.

Note, that by construction of TPN unfolding, any TLC has only one time final state explicitly associated with it. However, in the case when several concurrent branches of one process are synchronised by one transition, called *synchronising transition* a final state corresponds to a union of all timed states reachable through transitions of this TLC. Such a final state is called *covering final state* and has the following properties:

• If a transition is not time-enabled at the covering final state, then it is not enabled in any of the TS which are covered by this final state.

• If a transition is enabled at any of the covered TS, then it is enabled at the covering final state.

The covering final states represent the possible future traces of the net from some particular marking. It is also possible to show that if any two transitions are enabled at two different TS covered by one covering state, then there exists a TS at which both these transitions are enabled.

Proposition 2 A time state $s = (m, I)$ is reachable in the TSG of TIC-PN iff there exists a time configuration C^t such that marking component of $F_s^t(C)$ is equal to marking m and its associated set of inequalities has an isomorphic to I set of solutions. □

We develop a termination condition, called *cutoff condition*, to stop the construction of possibly infinite TPN unfolding. Two time states are said to be *equivalent* if their markings are equal and their sets of inequalities have isomorphic set of solutions. A transition of TPN unfolding is said to be a *cutoff transition* if all its time final states are equivalent to time final states of some other

[5] If transitions have several TLCs, then each combination of those should be considered.

Benchmark	Timed unfolding			TSG
	Trans.	Places	Fin. St.	States
Example	134	137	133	1249
TransCont.	21	29	21	37
FastF_Usafe	30	42	28	292
FastF_safe	21	30	21	125
Blocking	21	29	21	29
Ctrl_haz.	12	37	11	17
Ctrl_haz_free	10	32	10	13
Correct_ctrl	11	38	11	16

TABLE I
Experimental results

transitions added earlier to the unfolding. The algorithm producing TPN unfolding segment is given in Figure 2. It can be shown (similar to the proof in [6]) that for each time state $s = (m, I)$ reachable in the TSG there exists a time configuration C^t in segment of TPN unfolding such that the $F_s^t(C^t)$ is equal to m and the set of inequalities associated with C^t has an isomorphic to I set of solutions.

Using our algorithm we obtain a finite segment of TPN unfolding only examining the basic time states. For the example from Figure 1(a) the TPN unfolding contains 134 transitions (137 places) and examines 133 final states. For comparison, TSG constructed using algorithm of [12] has 1249 time states. Figures 1(b-e) show initial state and TLCs of the first instances of transitions.

IV. Application of TPN unfolding

PNs can readily model gates of asynchronous circuits (e.g. Circuit PNs [11]). In event-driven circuits the actual level of signal is not important and changes of one signal can be modelled with one transition. In level-driven circuits, the value of the signal plays important each signal is modelled by a set of transitions setting its level to high (e.g. $c+$) or to low ($c-$). Fragments of Circuit PN are composed together forming a PN for the whole circuit by simply merging the input and output arcs of the corresponding gates. Resulting net is then composed with a PN of the circuit environment.

Firstly, a circuit is verified for deadlock freedom. A deadlock exists in TIC-PN iff it exists in its underlying PN which follows from the fact that all choices in TIC-PN are time independent.

Analysis of event-driven circuits for hazards (unspecified changes of signals) is done by verifying that the composed net is safe. In TPN, these two input transitions can be PN enabled, but due to firing delays there may be no hazard.

The modelling of level-driven circuits needs more careful consideration. Often, when two gates are combined together, the resulting PN is no longer FC or UC PN.

If the choice is asymmetric (i.e. only one transition is disabling another), then this is interpreted as a hazard of the output signal of a gate and therefore can be analysed with our method. The method described in this paper is not capable of verifying other classes of circuits with non-TIC choice such as circuits with term takeover phenomenon and non-distributive circuits. However, we see it as a natural extension of the suggested method.

Latch control circuits from the AMULET microprocessor [1] were used as examples of event driven circuits. Verifying these circuits confirmed that the "fast forward" micropipeline control circuit has a potential hazard if the delay of the environment is short.

A 4-phase pipeline control circuit was taken as an example of a level driven circuit. Its verification showed that under certain delay assumptions this circuit would also be hazardous. A correct implementation of this circuit presented in [1] was shown to be correct.

In all examples our method explored less number of TS than in the TSG of the time PN model.

V. Conclusion

In this work we suggested a novel method of timing analysis based on the time unfolding. Our approach builds the time-driven unfolding constructing an implicit representation of the TSG. Although our method is exponential in the worst case, in practice it showed good results. We have demonstrated application of this method to the verification of hazard freedom of even-driven and level-driven circuits.

VI. Acknowledgements

Authours would like to aknowledge AMULET group, Tomohiro Yoneda and Luciano Lavagno for their help in carrying out this work.

References

[1] Amulet1 group workshop: Presentation materials. Technical report, Manchester University, Department of Computer Science, Amulet Group, Lake District, England, July 18-22 1994.

[2] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[3] B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE Transactions on Software Engineering*, 17(3):259–273, 1991.

[4] J. Esparza, S. Römer, and W. Vogler. An inprovement of mcmillan's unfolding algorithm. Technical Report TUM-I9599, Insitute Für Informatik, Technische Universität München, 1995.

[5] H. Hulgaard. *Timing analysis and verification of timed asynchronous circuits.* PhD thesis, University of Washignton, December 1995.

[6] K.L. McMillan. *Symbolic Model Checking.* Kluwer Academic Publishers, Boston, 1993.

[7] P. Merlin and D.J. Faber. Recoverability of communication protocols. *IEEE Transactions on Communications*, COM-24(9), 1976.

[8] T.G. Rokicki. *Representing and modeling digital circuits.* PhD thesis, Stanford University, 1993.

[9] L.Ya. Rosenblum and A.V. Yakovlev. Signal graphs: from self-timed to timed ones. In *Proceedings of International Workshop on Timed Petri Nets, Torino, Italy, July 1985*, pages 199–207.

[10] A. Semenov and A. Yakovlev. Event-based framework for verification of high-level models of asynchronous circuits. Technical Report 487, University of Newcastle upon Tyne, 1994.

[11] A. Semenov and A. Yakovlev. Combining partial orders and symbolic traversal for efficient verification of asynchronous circuits. In *Proceedings of CHDL'95, Chiba, Japan*, pages 567–573, 1995.

[12] T. Yoneda, I. Honma, and B.-H. Schlingloff. Verification of bounded delay asynchronous circuits with timed traces. Technical Report 94TR-0013, Tokyo Institute of Technology, August 1994.