Efficient Partial Enumeration for Timing Analysis of Asynchronous Systems

Eric Verlind, Gjalt de Jong and Bill Lin

IMEC, Kapeldreef 75, B-3001 Leuven, Belgium

Abstract — *This paper presents an efficient method for the timing verification of concurrent systems, modeled as labeled* **Timed Petri nets***. The verification problems we consider require us to analyze the system's reachable behaviors under the specified time delays. Our* **geometric timing analysis** *algorithm improves over existing ones by enumerating the state space only* **partially***. The algorithm relies on a concept called* **pre-mature firing** *and a new, extended notion of clocks with a* **negative age***. We have tested the fully automated procedure on a number of examples. Experimental results obtained on highly concurrent Petri nets with more than 6000 nodes and 10^{210} reachable states show that the proposed method can drastically reduce computational cost.*

I Introduction

Efficient timing verification algorithms are essential in the development of correctly working concurrent systems. Our work is mainly motivated by the need to verify asynchronous circuits where correctness of a design may depend on both functional and timing aspects. For example, some design methods, such as those for *timed circuits* [9], directly use timing information for optimization. Other design methods rely on delay information for the removal of hazards [8] or to ensure a fundamental mode of operation [14]. Timing verification can be a computationally expensive task due to exponential factors introduced by state enumeration and timing considerations.

The verification problems that we consider require timed *reachability analysis*. An approach to this is *geometric timing analysis (GTA)*. GTA algorithms have been studied by Berthomieu [2], Dill [5], Alur [1], among others. These methods can be relatively efficient in practice, but for highly concurrent systems, they can still be prohibitively expensive due to *state explosion* exponential in the concurrency parameter.

The GTA approach of Rokicki [10] improves on the basic procedure, but it suffers still from significant complexity problems, since it also traverses the complete state space. Related work by Hulgaard and Burns [7] is very efficient, but does not address verification problems that require reachability analysis and thus can not be directly compared with GTA methods.

To address the complexity problems in GTA, we propose an efficient state space traversal algorithm for the timing analysis of concurrent systems, modeled using a labeled timed Petri net (TPN). The TPN model used may combine different combinations of *choice* and *concurrency*, within a class of n-safe nets.

Our method offers a key improvement compared to existing GTA work by taking into account and exploiting independence information such that it suffices to traverse the state space only partially. It relies on the *pre-mature firing* concept, using a modified geometric representation, which incorporates an extended notion of clocks with a *negative age*. The canonization of the geometric regions required during each step of the enumeration is also extended to account for these two concepts. Since our GTA algorithm relies on partial enumeration of the state space, it requires path selection. Hence, we have developed several path selection heuristics.

Due to space limitation this paper focuses only on the analysis aspects. However we would like to stress that our formalism also includes notions of specification, circuit composition, and refinement pre-orders, required for timed circuit verification.

Experiments using our fully automated method show that for problems involving a high degree of concurrency, our approach indeed offers significant improvement over existing methods. Petri nets with more than 6000 nodes and 10^{210} reachable states have been analyzed using the proposed method.

Section II gives the formal terminology. Section III examines GTA and its complexity problems in detail. Section IV presents our analysis method, while section V presents experimental results.

II Model

We model a concurrent system as a Petri net extended with timing information: a **timed Petri net (TPN)** Σ is a tuple $\langle P, T, B, F, \tau\rangle$. Here, P is a finite set of *places*, T is a finite set of *transitions*, $B \subseteq P \times T$ is the *consumption flow relation*, and $F \subseteq T \times P$ is the *production flow relation*. Timing information is specified by a *static timing interval mapping* $\tau : P \rightarrow \mathbf{Q}^* \times (\mathbf{Q}^* \cup \{\infty\})$, where $\tau(p) = [\delta_p, \Delta_p]$, such that $\delta_p \leq \Delta_p \leq \infty$.

Note that the timing information is represented as interval time delays specified on the *places* of the net, rather than on transitions, as for example in Berthomieu's work [2]. **Presets** $\bullet t$ and $\bullet p$ and **postsets** $t\bullet$ and $p\bullet$ are defined as usual.

We denote the TPN's **marking** or **untimed state** by μ, being a mapping $P \rightarrow \mathbf{N}$, where μ_i is a shorthand for $\mu(p_i)$. $\mu[t\rangle$ denotes the *untimed enabling* of transition t. The untimed firing rule is defined as in ordinary Petri nets.

Besides a marking, a TPN state has also a timing component. A state ν for a TPN Σ is a tuple $\langle\mu, \gamma\rangle$, with μ a marking and γ a mapping $\gamma : P \times \mathbf{N} \rightarrow \mathbf{Q}^*$, which is defined for (p_i, s) for $0 \leq s \leq \mu_i$.

Mapping γ associates a *local time* with each token, set to 0 when the token is produced. In the sequel, we use the shorthand γ_{i_s} to denote $\gamma(p_i, s)$ and call it a *clock variable* or simply a *clock*. Furthermore we will omit the second subscript for clocks γ_{i_1} (γ_i denotes γ_{i_1}).

A (timed) Petri net is **n-safe** iff for all reachable markings μ, $\forall p_i \in P : \mu_i \leq n$. A 1-safe net is also just called **safe**. A transition t is enabled in timed state $\nu = \langle\mu, \gamma\rangle$, denoted by $\nu[t\rangle$, iff $\mu[t\rangle$ and $\forall p_i \in \bullet t : \delta_i \leq \gamma_i$ and $\exists p_i \in \bullet t : \gamma_i \leq \Delta_i$.

While first designed only to operate on 1-safe nets, our verifier now operates on an extended model called **SFTPN (transition Single-enabledness FIFO TPN)**, which is an n-safe TPN with a FIFO firing rule and transitions that are not multiply enabled.

The SFTPN model we propose is comparable to, but more general than the *orbital net* model used in Rokicki's work [10], in the sense that it does not have the significant restriction of allowing only a single (behavioral) input place for a transition. Furthermore, our model contains a class of *n-safe* nets, instead of just 1-safe. This allows to analyze a wider class of problems. In the above discussion, we have not dealt with verification of timing properties. However, in our model we provide means for checking and measuring these.

III Geometric Timing Analysis

A GTA: principle and complexity problems

During GTA, we deal with sets of states ν, which can be represented in various ways. The basic representation entity is a *geometric region*, representing for a marking μ with corresponding set of clocks, a system of inequalities of the following form:

$$\forall (p_i, s), s \le \mu_i : \varphi_i \le \gamma_{i_s} \le \Phi_i$$

$$\forall (p_i, s), (p_j, v) \in \mu_i, s \le \mu_i, v \le \mu_j : \varphi_{ij} \le \gamma_{i_s} - \gamma_{j_v} \le \Phi_{ij}$$

The first set of inequalities describes restrictions on single clocks γ_i, while the second set describes mutual restrictions between pairs of clocks $(\gamma_{i_s}, \gamma_{j_v})$. The inequalities describe a class of *convex regions* in $\mathbf{Q}^{|P|}$, in terms of clock variables γ_i. For such geometric regions, calculations can be done by efficient graph algorithms. The contiguity of the static timing intervals in the model often yields a natural and efficient representation in terms of these regions during analysis. However, as this section will show, analysis can still be prohibitively complex, depending on the problem.

The basic GTA method is described in the approaches of [2, 10], which, given a particular timed Petri net model, performs a reachability analysis, continuing along paths until a stop criterion is met. The core of the algorithm is formed by the following operations:

- advancement of time as much as possible;
- determination of all fireable transitions;
- firing of the transitions, leading to new timed regions.

The above basic GTA procedure gets prohibitively expensive for two reasons. First, there can be a blow-up in number of traversed *markings*, like in ordinary PNs. Additional to this, since representation of timing information introduces an extra level of complexity, a blow-up in the number of *regions* associated with individual markings can occur.

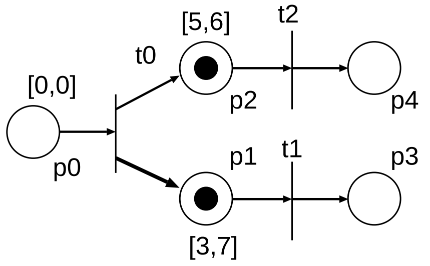

Figure 1: TPN with two parallel execution paths

Figure 1 shows a TPN to illustrate these problems. In the depicted TPN, concurrent transitions t_1 and t_2 are (untimed) enabled. Their independent firings produce a final marking μ^{34} of one token in both p_3 and p_4. Standard analysis evaluates all possible firing interleavings, in this case two, as t_1 could fire before t_2 or vice-versa.

Before arriving at μ^{34}, each interleaving passes a different *intermediate marking*. Besides of this, in μ^{34}, each interleaving has a different set of timing values, so they end up in two different *regions*, as shown by figure 2. It is clear from this, that in a complex system, we are very likely to see a state explosion effect because of the above problems.

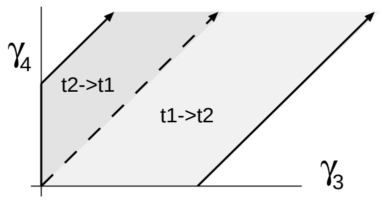

Figure 2: Firing interleavings/ pre-mature

B Improvement by Process Enumeration

To reduce the complexity problems of the basic GTA, Rokicki [10] proposes a partial order method using the concept of *process enumeration*. Its prime achievement is the reduction of the effects of the extra complexity caused by timing information, as compared to untimed analysis. As [9] shows, for a number of analysis examples, the number of different timing regions associated with a particular untimed state is on average very close to one, which is a drastic improvement over the standard method. However, if a parameterizable system has a number of reachable untimed states that is growing exponentially in function of its parameter n, the analysis will still show state explosion. The cause of the problem is the analysis procedure still visiting every reachable untimed state in the system. To alleviate these problems, we propose our method, as discussed in the remainder of the paper.

IV Partial Geometric Timing Analysis

In order to tackle the problems discussed in section III, we propose a partial enumeration approach related to *stubborn sets* [11], *partial orders* [6] and *anticipation* [4]. By traversing the state space *partially*, we aim at avoiding a combinational explosion for highly concurrent systems.

Figure 3 shows a simple, yet extreme case of a concurrent system, with n independent transitions with mutually overlapping firing intervals. The untimed state graph is a hypercube of dimension n, as figure 4 shows for $n = 3$.

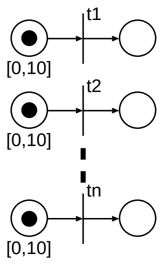

Figure 3: Parallel enabled transitions

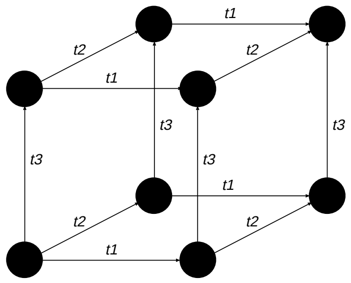

Figure 4: Untimed state space of highly parallel system

Standard GTA traverses $n!$ paths through the system, with $n.n!$ edges and $n.n! + 1$ visited regions. The method of [10] traverses 2^n edges and visits 2^n regions, still showing exponential complexity. Our approach, aimed at highly concurrent systems, however, reduces the number of traversed paths for this example to a single one, thus only traversing n edges and reaching $n + 1$ regions. Since for many complex problems, concurrency is an essential feature, our method will in many such cases show significantly better performance.

For timing analysis, we take into account implicitly the effects of firing path interleavings that were not selected, using only the traversal of a single selected interleaving. Figure 5 shows procedure "*ra_dfs*", which implements this depth-first GTA algorithm. Since we have to calculate the final timing region for a set of interleavings out of one interleaving, we have devised a special firing computation, different from the one used in standard geometric timing analysis. Therefore, we introduce the notion of *pre-mature firing* (section A) which enables us to perform these computations. Besides this, our pruning method is heuristic in nature and therefore requires a *transition selection strategy*, discussed in section B.

```
ra_dfs($\Sigma$)
{
 $\mathcal{R} = \emptyset$;
 $Q = \{\nu_0\}$;
 while ($Q \neq \emptyset$) {
  $R$ = get_head($Q$);
  advance_time($R$);
  if ($R \in \mathcal{R}$)
   continue;
  $\mathcal{R} = \mathcal{R} \cup \{R\}$;
  det_untimed_enabled_trans($R$);
  det_timed_enabled_trans($R$);
  $C$ = select_trans_set($R$);
  foreach $t_f \in C$ {
   foreach $p_i \in \bullet t_f$ {
    $Rc_i$ = strip_convex($R, p_i$);
    if ($Rc_i \neq \emptyset$) {
     $R_{new}$ =
      fire_premature($Rc_i, t_f$);
     add_head($Q, R_{new}$);
 }}}}}
```

```
fire_premature($Rc, t_f$)
{
 // $v'$ denotes quantity in
 // created region $R'c$
 foreach $p_i \in \mu$
  $\psi_i = \max_{j \in \bullet t}(\max(\varphi_j, \delta_j) + \varphi_{ij})$;
 foreach $p_k \in t_f \bullet$ {
  $\Phi'_k = \Phi_{i_{min}} - \delta_{i_{min}}$;
  $\varphi'_k = \varphi_{i_{max}} - \Delta_{i_{max}}$;
  foreach $p_j \in \mu' - t_f \bullet$ {
   $\Phi'_{kj} = -\psi_j$;
   $\varphi'_{kj} = \varphi_{ij} - \Delta_{i_{max}}$;
  }
 foreach $p_i, p_j \in t_f \bullet$
  $\Phi'_{ij} = 0$; //$= -\varphi'_{ji}$
 foreach $p_i, p_j \in \mu' - t_f \bullet$
  $\Phi'_{ij} = \Phi_{ij}$; //$= -\varphi'_{ji}$
 return $R'c$;
}
```

Figure 5: Reachability analysis and pre-mature firing

A Pre-mature firing calculation

When a transition firing produces a token in place p_i, clock γ_i (assuming 1-safe net for simplicity) is initialized to 0. A new conceptual idea that we propose here is to allow a clock also to have a *negative* age, with $\gamma_i < 0$ in fact meaning that it will take $|\gamma_i|$ time before the token is produced. The idea here is that of a *coordinate transformation*, removing clocks γ_i for $p_i \in \bullet t$ and adding clocks γ'_i for $p'_i \in t\bullet$. This firing is called *pre-mature firing*, which is done for an enabled transition that is sure to be fired because of *independence* of any other transitions.

The *pre-mature firing* procedure fires a transition t also for timed states in a region for which t is not enabled yet, before it could have fired and takes account of this implicitly by recording a negative value in the clock of the produced token. Thus, we take account of the various interleavings of independently evolving parallel computations in a rather natural way.

Consider figure 1 with marking $\mu_{12} = \{p_1, p_2\}$, with $\gamma_1 = \gamma_2 = 0$. Pre-mature firing of t_1 would yield $\mu_{32} = \{p_3, p_2\}$ with $-7 \leq \gamma_3 \leq -3 \wedge \gamma_2 = 0$. Consecutively executing a pre-mature firing of t_2 yields $\mu_{34} = \{p3, p4\}$, with $-7 \leq \gamma_3 \leq -3 \wedge -6 \leq \gamma_4 \leq -5 \wedge -2 \leq \gamma_3 - \gamma_4 \leq 3$. Figure 2 shows the result of the calculation, after advancing time to the positive quadrant (assuming tokens stay in p_3 and p_4), being the union of the regions after normal firing.

The set of inequalities contains the information about the extreme relative difference in the production times of the tokens in p_3 and p_4. The information kept thus, is sufficient for our verification purposes. It would otherwise have been scattered over the two regions resulting from firing each interleaving, section III. Thus, by performing the pre-mature firing procedure, we have brought down the number of evaluated paths to just one. Procedure "*ra_dfs*" (figure 5) calls "*fire_premature*" which implements the calculation of a new region, following the pre-mature firing of transition t_f in Rc. Because of the nonlinear character of maximum firing time enabling semantics, sometimes region splitting occurs, but we will not go into detail about it in this paper. Procedure "*strip_convex*" implements this by selecting part of a region for which p_i contains the limiting maximum timing bound. In "*fire_premature*", index i_{min} refers to a clock $\gamma_{i_{min}} \in \bullet t_f$, which is limiting as far as the minimum firing time of t_f is concerned, while i_{max} refers to $\gamma_{i_{max}} \in \bullet t_f$ limiting the maximum firing time.

B State space pruning selection heuristic

Our partial state space traversal exploits the fact that particular concurrent fireable transitions are independent. During traversal, in a region only a subset of the enabled transitions is (pre-maturely) fired, while firing other ones is postponed. So we need two things: calculation of independence (in general dynamic) and a selection criterion. As calculating exact independence can be prohibitively expensive, we use approximations. A computationally cheap solution is to use asimple structural notion of independence in Petri nets (extended free choice) as well as unique choice [7]. Our analyzer also includes an approximation that takes into account timing dynamically. This is both more powerful and more costly. For the highly concurrent problems we have experimented with, the simple structural approximation combined with a greedy selection criterion used in procedure "select_trans_set" appeared to work rather well.

C Generation of complete state graph

For the verification of more global functional properties compared to the checks currently done and for synthesis applications, one might need the complete region graph under timing restrictions, after state minimization. Calculating it directly using full GTA is not possible because of complexity problems. We propose to follow a conceptually two-stage approach: first, perform pruned GTA to obtain a Petri net unfolding with timing information. Timing ordering between transitions is then encoded in the unfolding *structure* by additional ordering edges, i.e. timing ordering is now represented without timing information. Second, use efficient implicit BDD [3] traversal techniques to generate an implicit representation of the complete state graph.

V Experiments

To assess the viability of our geometric timing analysis approach, we have carried out a number of experiments[1] that suffer from state explosion when existing methods are used. We compare to Rokicki's approach [10], which in general significantly outperforms the basic GTA algorithm[2]. Table 1 shows some of the results obtained. Examples "mmu", "master-read", "vme-write" and "mux-1" are highly concurrent STGs. The other examples are parameterised examples constructed

[1] implementation in "C", 64 MB HP 9000/715

[2] We implemented his method in "C" to be more favorable to it (orig. "Scheme")

Examples	$\|P\|$	$\|T\|$	$\|B\|$ $+\|F\|$	Untimed states	Process[10] enumeration regions	Process[10] enumeration CPU[s]	Partial enumeration regions	Partial enumeration CPU[s]
mmu	20	16	40	174	47	0.10	17	0.04
master-read	36	26	72	8932	8097	207.51	89	0.19
master-read2	38	26	76	1882	2440	51.91	62	0.13
vme-write	38	26	76	236	183	0.72	26	0.04
mux-1	43	35	93	9392	4512	51.12	107	0.14
3 PAR	39	28	78	748	321	1.15	36	0.05
5 PAR	65	44	130	1.87×10^4	2407	24.16	52	0.08
7 PAR	91	60	182	4.69×10^5	12654	313.66	68	0.12
10 PAR	130	84	260	5.86×10^7	out of time		92	0.20
100 PAR	1300	804	2600	4.73×10^{70}	out of time		812	14.69
300 PAR	3900	2404	7800	2.95×10^{210}	out of time		2412	261.56
4 RDZ	24	14	48	128	137	0.78	20	0.05
8 RDZ	56	30	112	3.28×10^4	out of time		51	0.23
16 RDZ	120	62	240	2.15×10^9	out of time		126	2.32
32 RDZ	248	126	496	9.22×10^{18}	out of time		319	22.86
64 RDZ	504	254	1008	1.70×10^{38}	out of time		759	228.26
4 SEQ	42	28	84	6156	140	0.46	31	0.06
8 SEQ	98	60	196	4.61×10^7	697	6.05	63	0.15
16 SEQ	210	124	420	2.15×10^{15}	6516	731.72	127	0.49
32 SEQ	434	252	868	4.18×10^{30}	out of time		255	1.94
64 SEQ	882	508	1764	1.48×10^{61}	out of time		511	9.81
128 SEQ	1778	1020	3556	1.78×10^{122}	out of time		1023	60.58
256 SEQ	3570	2044	7140	2.51×10^{244}	out of time		2047	404.08

Table 1: Experimental results
$|P|$, $|T|$ and $|B| + |F|$: TPN size parameters. "Untimed states": #states in untimed analysis. Then results obtained using method of [10] and the results obtained using our approach. Parametrized examples: size indicates nr. of elements for "PAR" system and nr. of 4-phase handshake input ports for "RDZ" and "SEQ" systems.

using basic elements from the synthesis approach of [12]. For instance, "RDZ" is a balanced tree of rendezvous C-elements, while the other examples are composed of other handshake blocks. The figures show that the method we propose successfully analyzes systems of significant size, while the analysis method from [10] could only analyze moderate problem instances.

VI Conclusion

The basic geometric timing analysis (GTA) method may suffer from combinational explosion both in the number of untimed states and in the timing information. The improved approach of [10] still suffers from state explosion for systems involving a high degree of concurrency. To tackle these problems, we have presented an approach that exploits the concurrency available in a system to arrive at a more efficient analysis.

Our partial enumeration GTA approach works with a relevant n-safe class of timed Petri nets. It relies on an extended geometric representation that is used by a "pre-mature firing" algorithm. The method also uses path selection heuristics. Experiments show the viability of the approach: it is highly efficient for a class of high concurrency problems, which are impossible to analyze with previous methods. The verifier is currently incorporated in the Assassin asynchronous interface compiler [13].

Acknowledgment - We would like to thank Tomas Rokicki, Chris Myers, Tilman Kolks and Steven Vercauteren for several insightful discussions.

References

[1] R. Alur, C.Courcoubetis, D. Dill, N.Halbwachs, and H.W.Toi. An implementation of three algorithms for timing verification based on automata emptiness. In *Proc. of the Real-Time Systems Symposium*. IEEE Computer Society Press, 1992.

[2] 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, March 1991.

[3] R.E. Bryant. Algorithmic Aspects of Symbolic Switch Network Analysis. *IEEE Transactions on Computer-Aided Design*, CAD-6(4):618–633, July 1987.

[4] G.G. de Jong. Verification of Data Flow Graphs using Temporal Logic. In L. Claesen, editor, *Applied Formal Methods For Correct VLSI Design*, volume 1, pages 301–310, November 1989.

[5] D. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Proc. Workshop on Automatic Verification Methods for Finite-State Systems*, June 1989.

[6] P. Godefroid. Using Partial Orders to Improve Automatic Verification Methods. In *LNCS vol. 531*, pages 176–185. Springer-Verlag, June 1990.

[7] H. Hulgaard and S.M. Burns. Efficient Timing Analysis of a Class of Petri Nets. In *Proc. Computer Aided Verification Workshop*, 1995.

[8] L. Lavagno, K. Keutzer, and A. Sangiovanni-Vincentelli. Algorithms for synthesis of hazard-free asynchronous circuits. In *Proc. 28th ACM/IEEE Design Automation Conference*, June 1991.

[9] C.J. Myers, T.G. Rokicki, and T.H.-Y. Meng. Automatic Synthesis and Verification of Gate-level Timed Circuits. Technical Report CSL-TR-94-652, Stanford Un., Computer Systems Lab, December 1994.

[10] T.G. Rokicki. *Representing and Modeling Digital Circuits.* PhD thesis, Stanford University, December 1993.

[11] A. Valmari. Stubborn sets for reduced state space generation. In *Int. Conference on Applications and Theory of Petri Nets*, pages 1–22, 1989.

[12] K. van Berkel. *Handshake circuits: an intermediary between communicating processes and VLSI.* PhD thesis, Technische Universiteit Eindhoven, 1992.

[13] C. Ykman-Couvreur, B. Lin, and H. De Man. ASSASSIN: A Synthesis System for Asynchronous Control Circuits. Technical report, IMEC, September 1994. User and Tutorial manual.

[14] K.Y. Yun. Synthesis of Asynchronous Controllers for Heterogeneous Systems. Technical Report CSL-TR-95-644, Stanford Un., Computer Systems Lab, August 1994.