Synthesizing Parametric Constraints on Various Case Studies Using IMITATOR II

Étienne André

December 2010

Research report LSV-10-21

Laboratoire Spécification & Vérification

École Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France

Synthesizing Parametric Constraints on Various Case Studies Using IMITATOR II

Étienne André

Chapter 1

Introduction

Timed automata [1] are finite control automata equipped with *clocks*, which are real-valued variables which increase uniformly. This model is useful for reasoning about real-time systems with a dense representation of time, because one can specify quantitatively the interval of time during which the transitions can occur, using timing bounds. However, the behavior of a system is very sensitive to the values of these bounds, and it is rather difficult to find their correct values. One can check the correctness of the system for one particular timing value for each timing bound (using model checkers such as, e.g., UPPAAL [31]), but this does not give any information for other values. Actually, testing the correctness of the system for all the timing values, even in a bounded interval, would require an infinite number of calls to the model checker, because those timing bounds can have real (or rational) values.

It is therefore interesting to reason *parametrically*, by considering that these bounds are unknown constants, or parameters, and try to synthesize a *constraint* (i.e., a conjunction of linear inequalities) on these parameters which will guarantee a correct behavior of the system. Such automata are called *parametric timed automata* (PTA) [2].

The Good Parameters Problem for Timed Automata. In order to find correct values of the parameters, we are interested in solving the following *good parameters problem*, as defined in [22] in the framework of linear hybrid automata: "Given a PTA $\mathcal{A}$ and a rectangular parameter domain V_0, what is the largest set of parameter values within V_0 for which $\mathcal{A}$ is safe?"

The parameter design problem for timed automata (and more generally, for linear hybrid automata) was formulated in [25], where a straightforward solution is given, based on the generation of the whole parametric state space until a fixpoint is reached. Unfortunately, in all but the most simple cases, this is is prohibitively expensive due, in particular, to the brute exploration of the whole parametric state space.

In [22], the authors propose an extension based on the *counterexample guided abstraction refinement* (CEGAR, [18]). When finding a counterexample, the system obtains constraints on the parameters that *make* the counterexample infeasible. When all the counterexamples have been eliminated, the resulting constraints describe a set of parameters for which the system is safe.

The tool IMITATOR II, presented in [8], is based on the *inverse method* [5],

which supposes given a "good valuation" π_0 of the parameters that one wants to generalize. More precisely, IMITATOR II synthesizes a constraint K_0 on the parameters that corresponds to an infinite dense set of valuations such that, for all valuation π of parameters in this set, the behavior of the timed automaton $\mathcal{A}$ is *(time-abstract) equivalent* to the behavior of $\mathcal{A}$ under π_0, in the sense that they have the same trace sets. This is useful to relax timing bounds, and gives a criterion of *robustness*.

Moreover, IMITATOR II implements the *behavioral cartography algorithm* [6], which synthesizes a constraint on the parameters ("tile") by calling the inverse method for each integer point located within a given rectangle V_0. This algorithm allows us to partition the parametric space into a subset of "good" tiles (which correspond to "good behaviors") and a subset of "bad" ones. Often in practice, what is covered is not the *bounded* and *integer* subspace of the parameter rectangle, but two major extensions: first, not only the integer points but a major part of the dense set of *real-valued* points of the rectangle is covered by the tiles; second, the tiles are often unbounded w.r.t. several dimensions (hence are infinite), and cover most of the parametric space beyond V_0, thus giving a solution to the good parameters problem.

IMITATOR II is a new version of IMITATOR [7], a prototype written in Python implementing the inverse method, and calling the model checker HYTECH [24]. IMITATOR II has been entirely rewritten and is a now standalone tool, making use of the APRON library [27] and the Parma Polyhedra Library [11]. Compared to IMITATOR, the computation timings of IMITATOR II have dramatically decreased. Moreover, IMITATOR II offers new features, such as the implementation of the behavioral cartography algorithm, the generation of the trace sets of the models, and a graphical output. This tool is being developed at LSV, ENS Cachan, France. The tool can be downloaded on its Web page[1], as well as a bunch of case studies.

We present in this report a range of case studies.

[1] `http://www.lsv.ens-cachan.fr/~andre/IMITATOR2/`

Contents

Chapter 2

SR-Latch

We consider in this section a SR "NOR" latch, which is one of the most fundamental latches. S and R stand for set and reset. This latch (described in, e.g., [23]) is depicted in Figure 2.1 left. This circuit is made of two "NOR" gates. There are two input signals R and S, and two output signals Q and $\overline{Q}$. The stored bit is present on the output Q.

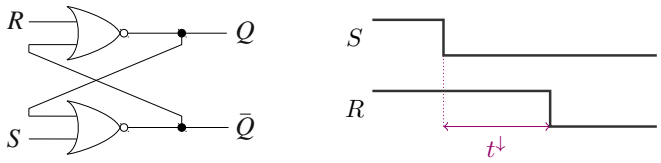


Figure 2.1: SR latch (left) and environment (right)

The possible configurations of the latch are the following ones:

S	R	Q	$\overline{Q}$
0	0	latch	latch
0	1	0	1
1	0	1	0
1	1	0	0

We consider an initial configuration with $R = S = 1$ and $Q = \overline{Q} = 0$. As depicted in Figure 2.1 right, the signal S first goes down. Then, the signal R goes down after a time $t^{\downarrow}$.

We consider that the gate NOR_1 (resp. NOR_2) has a punctual parametric delay δ_1 (resp. δ_2). Moreover, the parameter $t^{\downarrow}$ corresponds to the time duration between the fall of S and the fall of R.

Each location of the PTA $\mathcal{A}$ modeling this SR-latch corresponds to a different configuration of the signals R, S, Q and $\overline{Q}$. We give in Table 2.1 the correspondence between the name of the location q_i, for $i = 0, \ldots, 5$, and the value of the four signals (only the locations that are actually reachable from the initial state using our environment are depicted).

We consider the following reference valuation π_0 of the parameters:

$$\delta_1 = 2 \qquad \delta_2 = 2 \qquad t^{\downarrow} = 1$$

Location	S	R	Q	$\overline{Q}$
q_0	1	1	0	0
q_1	0	1	0	0
q_2	0	0	0	0
q_3	0	1	0	1
q_4	0	0	0	1
q_5	0	0	1	0

Table 2.1: Value of the signals for each of the locations of the SR-latch

Under π_0, it can be shown (e.g., using IMITATOR II in reachability mode) that the corresponding trace set is the one depicted in Figure 2.2.

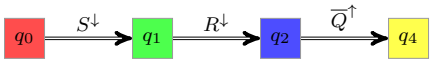


Figure 2.2: Trace set for the SR latch under K_0

Synthesis of Parameters. Our goal is to synthesize a set of parameters guaranteeing the following good behavior: "the system always ends in a state where $\overline{Q} = 1$". This behavior corresponds to trace sets such that, for any trace of the trace set, the last location of the trace is such that $\overline{Q} = 1$. From Table 2.1, such locations are q_3 or q_4. One can see that the trace set of $\mathcal{A}[\pi_0]$, which is made of a single trace, satisfies this requirement, because the last location of the trace is q_4. As a consequence, $\mathcal{A}$ has a good behavior under π_0.

2.1 Inverse Method

Let us now synthesize other parameter valuations corresponding to this behavior, by applying the inverse method to $\mathcal{A}$ and π_0. IMITATOR II synthesizes the following constraint K_0:

$$\delta_2 > t^{\downarrow} \quad \wedge \quad t^{\downarrow} + \delta_1 > \delta_2$$

From the correctness of the inverse method, the trace set corresponding to the system under any $\pi \models K_0$ is equal to the one given in Figure 2.2. It can be shown that this constraint K_0 is not maximal, i.e., there exist other parameter valuations having the same good behavior. It will be the purpose of Section 2.3 to synthesize the maximal constraint.

2.2 Parametric Reachability Analysis

Considering this environment, the trace set of this system is given in Figure 2.3, where the value of the signals corresponding to each location is given in Table 2.1.

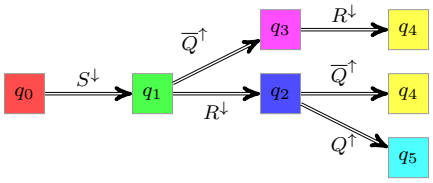


Figure 2.3: Parametric reachability analysis of the SR latch

2.3 Behavioral Cartography Algorithm

Using IMITATOR II, we now perform a behavioral cartography of this system. We consider the following rectangle V_0 for the parameters:

$$\begin{aligned} t^{\downarrow} &\in [0, 10] \\ \delta_1 &\in [0, 10] \\ \delta_2 &\in [0, 10] \end{aligned}$$

We get the following six behavioral tiles. For each of those tiles, we will give the corresponding trace set, where the value of the signals corresponding to each location is given in Table 2.1 page 6.

Tile 1. This tile corresponds to the values of the parameters verifying the following constraint:

$$t^{\downarrow} = \delta_2 \quad \wedge \quad \delta_1 = 0$$

The trace set of this tile is given in Figure 2.4.

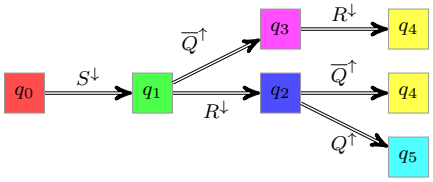


Figure 2.4: Trace set of tile 1 for the SR latch

Since $t^{\downarrow} = \delta_2$, $R^{\downarrow}$ and $\overline{Q}^{\uparrow}$ will occur at the same time. Thus, the order of those two events is unspecified, which explains the choice between going to q_2 or q_3. When in state q_2, either $Q^{\uparrow}$ can occur (since $\delta_1 = 0$), in which case the system is stable, or $\overline{Q}^{\uparrow}$ can occur, which also leads to stability.

Tile 2. This tile corresponds to the values of the parameters verifying the following constraint:

$$t^{\downarrow} = \delta_2 \quad \wedge \quad \delta_1 > 0$$

The trace set of this tile is given in Figure 2.5.

Since $t^{\downarrow} = \delta_2$, $R^{\downarrow}$ and $\overline{Q}^{\uparrow}$ will occur at the same time. Thus, the order of those two events is unspecified, which explains the choice between going to q_2 or

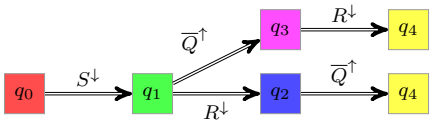


Figure 2.5: Trace set of tile 2 for the SR latch

q_3. When in state q_2, $Q^{\uparrow}$ can not occur (since $\delta_1 > 0$), so $\overline{Q}^{\uparrow}$ occurs immediately after $R^{\downarrow}$, which leads to stability.

Tile 3. This tile corresponds to the values of the parameters verifying the following constraint:

$$\delta_2 > t^{\downarrow} + \delta_1$$

The trace set of this tile is given in Figure 2.6.

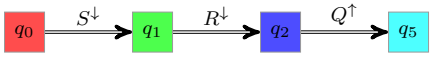


Figure 2.6: Trace set of tile 3 for the SR latch

In this case, since $\delta_2 > t^{\downarrow} + \delta_1$, $S^{\downarrow}$ will occur before the gate Nor_2 has the time to change. For the same reason, $Q^{\uparrow}$ will change before Nor_1 has the time to change. With $Q = 1$, the system is now stable: Nor_1 does not change.

Tile 4. This tile corresponds to the values of the parameters verifying the following constraint:

$$t^{\downarrow} + \delta_1 = \delta_2 \quad \wedge \quad \delta_2 \geq \delta_1 \quad \wedge \quad \delta_1 > 0$$

The trace set of this tile is given in Figure 2.7.

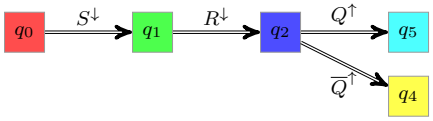


Figure 2.7: Trace set of tile 4 for the SR latch

Since $t^{\downarrow} + \delta_1 = \delta_2$, both $Q^{\uparrow}$ or $\overline{Q}^{\uparrow}$ can occur. Once one of them occured, the system gets stable, and no other change occurs.

Tile 5. This tile corresponds to the values of the parameters verifying the following constraint:

$$\delta_2 > t^{\downarrow} \quad \wedge \quad t^{\downarrow} + \delta_1 > \delta_2$$

Note that this constraint is equal to K_0. The trace set of this tile is given in Figure 2.8.

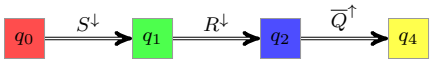


Figure 2.8: Trace set of tile 5 for the SR latch

Since $\delta_2 > t^{\downarrow}$, the gate Nor_2 can not change before $R^{\downarrow}$ occurs. However, since $t^{\downarrow} + \delta_1 > \delta_2$, the gate Nor_2 changes before $Q^{\uparrow}$ can occur, thus leading to event $\overline{Q}^{\uparrow}$.

Tile 6. This tile corresponds to the values of the parameters verifying the following constraint:

$$t^{\downarrow} > \delta_2$$

The trace set of this tile is given in Figure 2.9.

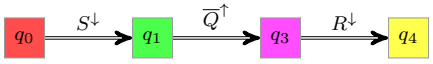


Figure 2.9: Trace set of tile 6 for the SR latch

Since $t^{\downarrow} > \delta_2$, $\overline{Q}^{\uparrow}$ occurs before $S^{\downarrow}$. The system is then stable.

Cartography. We give in Figure 2.10 the cartography of the SR latch example. For the sake of simplicity of representation, we consider only parameters δ_1 and δ_2. Therefore, we set $t^{\downarrow} = 1$.

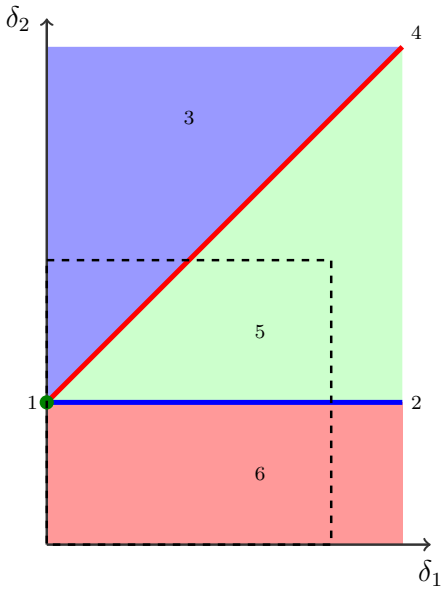


Figure 2.10: Behavioral cartography of the SR latch according to δ_1 and δ_2

The rectangle V_0 is represented with dashed lines. Note that tile 1 corresponds to a point, and tiles 2 and 4 correspond to lines. Note also that all tiles (except tile 1) are unbounded. As a consequence, the cartography covers, not only V_0, but all the positive real-valued parametric space. Constraints synthesized using our algorithm in order to guarantee a given behavior will thus necessarily be maximal for this case study.

Verification of Properties. Recall that we aim at synthesizing the maximal set of parameters guaranteeing the following behavior: "the system always ends in a state where $\overline{Q} = 1$", i.e., each trace ends either in state q_3 or in state q_4. One can easily infer from the six trace sets that tiles 2, 5 and 6 are good tiles, and the other tiles are bad tiles. As a consequence, the maximal set of parameters corresponding to all the good behaviors is the union of the constraints associated to the three good tiles, i.e.:

$$\begin{array}{lllll} & t^{\downarrow} = \delta_2 & \wedge & \delta_1 > 0 \\ \vee & \delta_2 > t^{\downarrow} & \wedge & t^{\downarrow} + \delta_1 > \delta_2 \\ \vee & t^{\downarrow} > \delta_2 & & \end{array}$$

It can be shown that this constraint is actually equivalent to $t^{\downarrow} + \delta_1 > \delta_2$. Note that this constraint is *maximal*, because our cartography algorithm covers the whole parametric space.

If one now considers another property, we will get a different partition between good and bad tiles, and thus a different constraint. For example, if one wants to synthesize parameter valuations such that "the system always ends in a state where $\overline{Q} = 0$" (i.e., each trace ends in state q_5), only tile 3 is a good tile, leading to the maximal constraint $\delta_2 > t^{\downarrow} + \delta_1$.

Comparison with other methods. Due to the simplicity of this example, it is possible to apply the method introduced in [25], consisting in computing the whole set of reachable states, and then intersect it with the bad states. We first consider the property "the system always ends in a state where $\overline{Q} = 1$". We introduce one more PTA in parallel with the others, which plays the role of an observer. This PTA goes into a "good" location when synchronizing with action $\overline{Q}^{\uparrow}$, and into a "bad" location when synchronizing with action $Q^{\uparrow}$. Using the HyTech model checker, we can compute the whole set of reachable states, project the constraint onto the parameters, and intersect with "the bad" locations, i.e., keep only the states where the observer is in the bad location. The constraint on the parameters associated to the bad states is $t^{\downarrow} + \delta_1 \leq \delta_2$. Thus, by negating it, one finds back the constraint found by our algorithm, i.e., $t^{\downarrow} + \delta_1 > \delta_2$.

Similarly for the second property (i.e., "the system always ends in a state where $\overline{Q} = 0$"), we slightly modify the observer (i.e., swap the good and the bad location), and this method also allows to synthesize the same constraint as our algorithm.

Chapter 3

Flip-flop Circuit

3.1 Description

Consider an asynchronous "D flip-flop" circuit described in [17] and depicted in Figure 3.1. It is composed of 4 gates (G_1, G_2, G_3 and G_4) interconnected in a cyclic way, and an environment involving two input signals D and CK. The global output signal is Q.

Each gate G_i has a delay in the parametric interval $[\delta_i^-, \delta_i^+]$, with $\delta_i^- \leq \delta_i^+$. There are 4 other parameters (viz., $T_{HI}, T_{LO}, T_{Setup}$, and T_{Hold}) used to model the environment. The output signal of a gate G_i is named g_i (note that $g_4 = Q$). The rising (resp. falling) edge of signal D is denoted by $D^{\uparrow}$ (resp. $D^{\downarrow}$) and similarly for signals $CK, Q, g_1, \ldots, g_4$.

Each gate is modeled by a PTA, as well as the environment. We consider a bi-bounded inertial model for gates (see [13, 32]), where any change of the input may lead to a change of the output (after some delay). The (network of) PTAs $\mathcal{A}$ modeling the system results from the composition of those 5 PTAs. Each location of $\mathcal{A}$ corresponds to a different value of the signals D, CK, g_1, g_2, g_3 and g_4 (recall that $g_4 = Q$).

3.2 First environment

We first consider an environment where initially $D = CK = Q = 0$ and $g_1 = g_2 = g_3 = 1$, with the following ordered sequence of actions for inputs D and

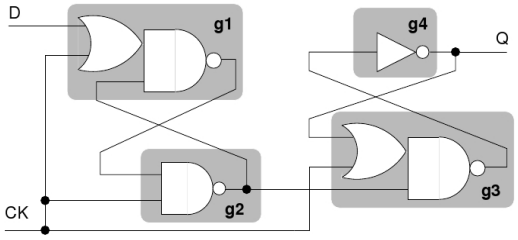


Figure 3.1: Flip-flop circuit

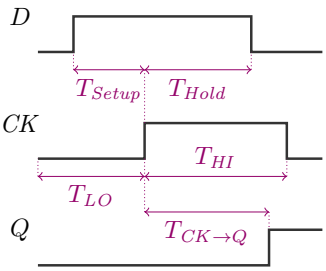


Figure 3.2: Environment for the flip-flop circuit with $D = 1$

CK: $D^{\uparrow}$, $CK^{\uparrow}$, $D^{\downarrow}$, $CK^{\downarrow}$, as depicted in Figure 3.2. Therefore, we have the implicit constraint $T_{Setup} \leq T_{LO} \wedge T_{Hold} \leq T_{HI}$.

The initial location q_0 corresponds to the initial levels of the signals according to the environment. The initial constraint K_0 corresponds to:

$$T_{Setup} \leq T_{LO} \wedge T_{Hold} \leq T_{HI} \wedge \bigwedge_{i=1,..,4} \delta_i^- \leq \delta_i^+$$

3.2.1 Definition of the Good Behavior

We consider that the flip-flop circuit has a *good behavior* if every trace contains both $Q^{\uparrow}$ and $CK^{\downarrow}$, and $Q^{\uparrow}$ occurs before $CK^{\downarrow}$.

We are interested in synthesizing a constraint on the parameters of the system preventing any bad behavior, i.e., ensuring that the system will always behave well.

For the sake of simplicity (to bound the number of different traces), we consider from now that the time in the model stops after the event $CK^{\downarrow}$, since it is sufficient to consider the system up to $CK^{\downarrow}$ to study its correctness.

3.2.2 Reference Valuation

We consider the following valuation π_0 of the parameters:

$T_{HI} = 24$	$T_{LO} = 15$	$T_{Setup} = 10$	$T_{Hold} = 17$
$\delta_1^- = 7$	$\delta_1^+ = 7$	$\delta_2^- = 5$	$\delta_2^+ = 6$
$\delta_3^- = 8$	$\delta_3^+ = 10$	$\delta_4^- = 3$	$\delta_4^+ = 7$

Let us study the behavior of the flip-flop circuit under π_0. The trace set of $\mathcal{A}[\pi_0]$ is depicted in Figure 3.3, where the meaning of each location in terms of signals is given in Table 3.1. Recall that we do not depict each trace separately, but depict the trace set under the form of a tree or a graph. However, this graph structure is only used for the sake of simplicity of representation of the possible traces, and does not contain any information on the possible branching behavior of the system.

Each of the two traces depicted in this trace set contains both $Q^{\uparrow}$ and $CK^{\downarrow}$, and $Q^{\uparrow}$ occurs before $CK^{\downarrow}$. As a consequence, this trace set is a *good* trace set according to the property we want to verify.

We are now interested in studying the evolution of the behavior of the system if one changes some of the values of the parameters. More precisely, we are

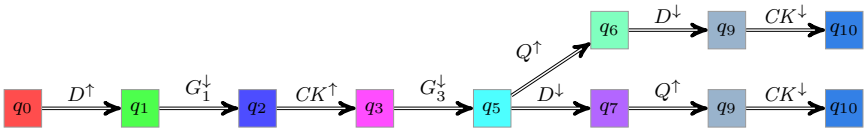


Figure 3.3: Trace set of the flip-flop circuit under π_0

Location	D	CK	g_1	g_2	g_3	g_4
q_0	0	0	1	1	1	0
q_1	1	0	1	1	1	0
q_2	1	0	0	1	1	0
q_3	1	1	0	1	1	0
q_4	0	1	0	1	1	0
q_5	1	1	0	1	0	0
q_6	1	1	0	1	0	1
q_7	0	1	0	1	0	0
q_8	0	0	0	1	0	0
q_9	0	1	0	1	0	1
q_{10}	0	0	0	1	0	1

Table 3.1: Locations of the flip-flop circuit

interested in identifying parameter valuations for which the system has exactly the same (good) behavior, i.e., exactly the same trace set.

3.2.3 Parametric Reachability

We first use a reachability method in order to infer good values for parameters δ_3^+ and δ_3^+. As a consequence, we instantiate the other parameters of the system, and use the following method:

1. Compute the whole parametric trace set;
2. Look for the bad states (i.e., belonging to trace violating the property of correctness);
3. Perform the disjunction of the constraints on the parameters associated to those bad states.

The parametric trace set is given in Figure 3.4. It is easy to see that states 10, 12, 16 and 18 are bad states, since they belong to traces where $Q^{\downarrow}$ does not occur before $CK^{\downarrow}$.

The constraint on the parameters associated to states 12 and 16 is the following one:

$$\delta_4^+ \geq 7 \quad \wedge \quad \delta_3^+ \geq 8 \quad \wedge \quad \delta_3^+ + \delta_4^+ \geq 24$$

The constraint on the parameters associated to states 10 and 18 is the following one:

$$\delta_4^+ \geq 3 \quad \wedge \quad \delta_3^+ \geq 17 \quad \wedge \quad \delta_3^+ + \delta_4^+ \geq 24$$

Since the zone $\delta_3^+ \leq 7$ or $\delta_4^+ \leq 3$ correspond to an undefined behavior according to the model, one can thus infer that the constraint corresponding to a good

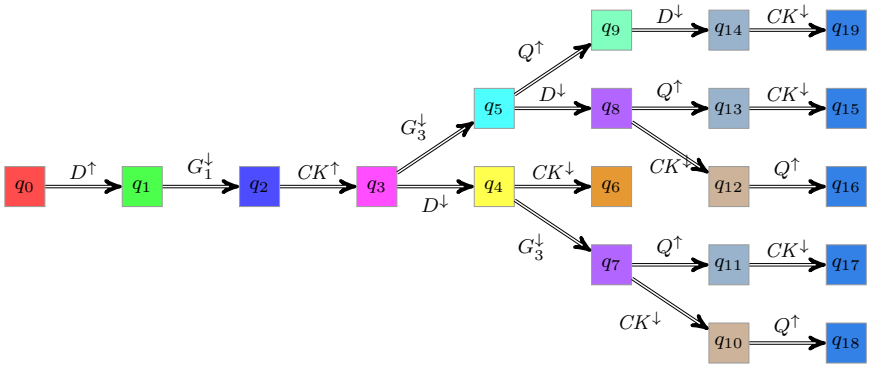


Figure 3.4: Parametric reachability analysis of the flip-flop circuit

behavior is:

$$\delta_3^+ \geq 7 \quad \wedge \quad \delta_4^+ \geq 3 \quad \wedge \quad \delta_3^+ + \delta_4^+ \leq 24$$

Note that this computation has been presented for only two parameters (viz., δ_3^+ and δ_4^+) for the sake of simplicity, but is also possible for all parameters. The computation, although quite long, terminates in practice using the reachability mode of IMITATOR II.

3.2.4 Inverse Method

Using our program IMITATOR II applied to the PTA $\mathcal{A}$ modeling the system and the reference valuation π_0, the following constraint K_0 is computed after 9 iterations in 0.122 second (11 reachable states with 10 transitions):

$$\begin{array}{rrcrr} & \delta_3^+ + \delta_4^+ \geq T_{Hold} & \wedge & \delta_1^- > 0 \\ \wedge & T_{Hold} \geq \delta_3^- + \delta_4^- & \wedge & T_{Hold} > \delta_3^+ \\ \wedge & T_{HI} > \delta_3^+ + \delta_4^+ & \wedge & T_{Setup} > \delta_1^+ \end{array}$$

3.2.5 Clarisó and Cortadella's Constraints

In [17], a constraint Z is synthesized in order to prevent bad system behaviors. The bad state is defined as the case where $CK^{\downarrow}$ occurs before $Q^{\uparrow}$. This constraint Z is the following:

$$\begin{array}{rrcrr} & T_{CK \to Q} \leq \delta_2^+ + \delta_3^+ + \delta_4^+ & \wedge & T_{Setup} > \delta_1^+ + \delta_2^+ - \delta_2^- \\ \wedge & T_{Hold} > \delta_2^+ + \delta_3^+ & \wedge & T_{HI} > \delta_2^+ + \delta_3^+ + \delta_4^+ \\ \wedge & T_{HI} > T_{Hold} & \wedge & T_{LO} > T_{Setup} \\ \wedge & \delta_1^- > \delta_2^+ \end{array}$$

Projection onto T_{Setup} and T_{Hold}:

$$\begin{array}{rrcrr} & T_{Setup} > 8 & \wedge & T_{Hold} > 16 \\ \wedge & T_{Hold} < 24 & \wedge & T_{Setup} < 15 \end{array}$$

Projection onto δ_3^+ and δ_4^+:

$$\delta_3^+ < 11 \quad \wedge \quad \delta_3^+ + \delta_4^+ < 18$$

3.2.6 Behavioral Cartography

We are interested in studying the correctness of the circuit according to parameters δ_3^+ and δ_4^+. Using the behavioral cartography algorithm, we compute the following cartography of the flip-flop circuit according to δ_3^+ and δ_4^+.

In the following, when giving constraints on all the parameters, we will omit the following "trivial inequalities" for the sake of clarity:

$$\begin{array}{rrcrr} & \delta_1^- \leq \delta_1^+ & \wedge & \delta_2^- \leq \delta_2^+ \\ \wedge & \delta_3^- \leq \delta_3^+ & \wedge & \delta_4^- \leq \delta_4^+ \\ \wedge & \delta_1^- \geq 0 & \wedge & \delta_2^- \geq 0 \\ \wedge & \delta_3^- \geq 0 & \wedge & \delta_4^- \geq 0 \\ \wedge & T_{Setup} \leq T_{LO} & \wedge & T_{Hold} \leq T_{HI} \end{array}$$

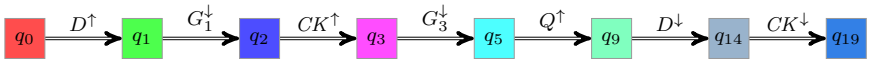
Tile 1. This tile corresponds to the values of the parameters verifying the following constraint:

$$\begin{array}{rrcrr} & \delta_1^- > 0 & \wedge & T_{Setup} > \delta_1^+ \\ \wedge & T_{Hold} > \delta_3^+ + \delta_4^+ \end{array}$$

After instantiation of all parameters except δ_3^+ and δ_4^+:

$$\delta_3^+ + \delta_4^+ < 17 \quad \wedge \quad \delta_3^+ \geq 8 \quad \wedge \quad \delta_4^+ \geq 3$$

The trace set of this tile is given below:



Tile 2. This tile corresponds to the values of the parameters verifying the following constraint:

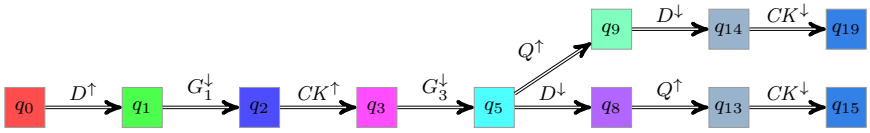
$$\begin{array}{rrcrr} & \delta_3^+ + \delta_4^+ \geq T_{Hold} & \wedge & \delta_1^- > 0 \\ \wedge & T_{Hold} \geq \delta_3^- + \delta_4^- & \wedge & T_{Hold} > \delta_3^+ \\ \wedge & T_{HI} > \delta_3^+ + \delta_4^+ & \wedge & T_{Setup} > \delta_1^+ \end{array}$$

Note that this constraint is the same as K_0, which makes sense since π_0 models this constraint.

After instantiation of all parameters except δ_3^+ and δ_4^+:

$$\begin{array}{rrcrr} & \delta_3^+ + \delta_4^+ < 24 & \wedge & \delta_3^+ < 17 \\ \wedge & \delta_3^+ \geq 8 & \wedge & \delta_4^+ \geq 3 \\ \wedge & \delta_3^+ + \delta_4^+ \geq 17 \end{array}$$

The trace set of this tile is given below:



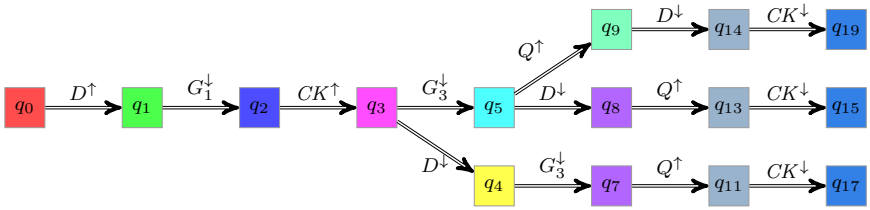
Tile 3. This tile corresponds to the values of the parameters verifying the following constraint:

$$\begin{array}{rrcrr} & \delta_3^+ \geq T_{Hold} & \wedge & T_{Hold} \geq \delta_3^- + \delta_4^- \\ \wedge & T_{Setup} > \delta_1^+ & \wedge & T_{HI} > \delta_3^+ + \delta_4^+ \end{array}$$

After instantiation of all parameters except δ_3^+ and δ_4^+:

$$\delta_3^+ + \delta_4^+ < 24 \quad \wedge \quad \delta_3^+ \geq 17 \quad \wedge \quad \delta_4^+ \geq 3$$

The trace set of this tile is given below:



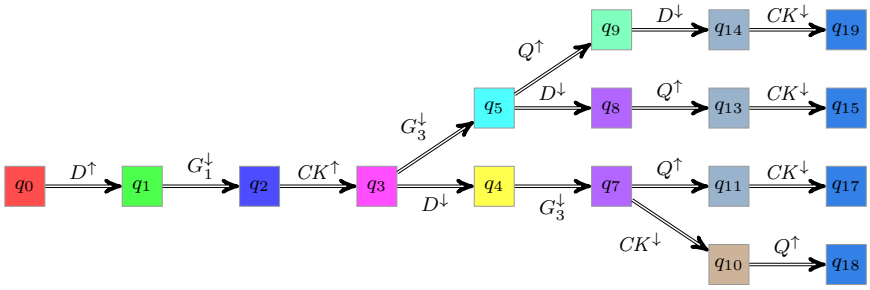
Tile 4. This tile corresponds to the values of the parameters verifying the following constraint:

$$\begin{array}{rrrr} & \delta_3^+ + \delta_4^+ \geq T_{HI} & \wedge & \delta_1^- > 0 \\ \wedge & T_{Hold} \geq \delta_3^- + \delta_4^- & \wedge & \delta_3^- > 0 \\ \wedge & T_{HI} > \delta_4^+ + T_{Hold} & \wedge & T_{HI} > \delta_3^+ \\ \wedge & T_{Setup} > \delta_1^+ & & \end{array}$$

After instantiation of all parameters except δ_3^+ and δ_4^+:

$$\begin{array}{rrrr} & \delta_3^+ + \delta_4^+ \geq 24 & \wedge & \delta_3^+ < 24 \\ \wedge & \delta_4^+ \geq 3 & \wedge & \delta_4^+ < 7 \end{array}$$

The trace set of this tile is given below:



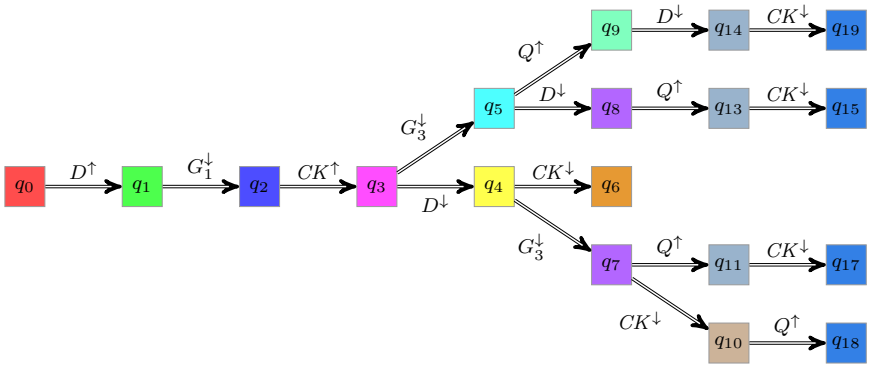
Tile 5. This tile corresponds to the values of the parameters verifying the following constraint:

$$\begin{array}{rrrr} & T_{Hold} \geq \delta_3^- + \delta_4^- & \wedge & \delta_1^- > 0 \\ \wedge & T_{HI} > \delta_4^+ + T_{Hold} & \wedge & \delta_3^- > 0 \\ \wedge & T_{Setup} > \delta_1^+ & \wedge & \delta_3^+ \geq T_{HI} \end{array}$$

After instantiation of all parameters except δ_3^+ and δ_4^+:

$$\delta_3^+ \geq 24 \quad \wedge \quad \delta_4^+ \geq 3 \quad \wedge \quad \delta_4^+ < 7$$

The trace set of this tile is given below:



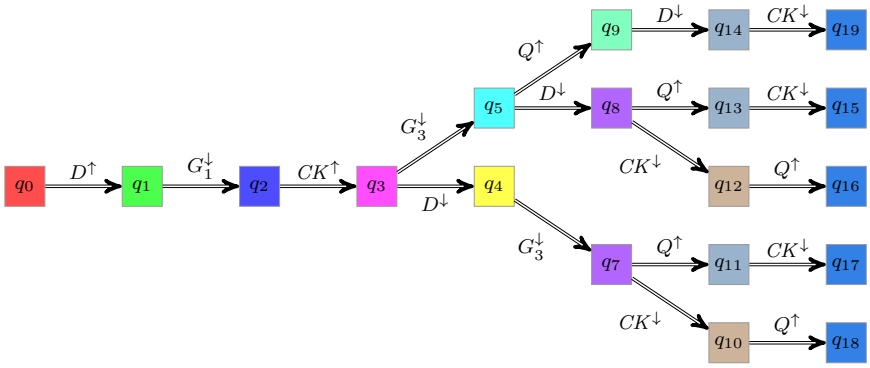
Tile 6. This tile corresponds to the values of the parameters verifying the following constraint:

$$\begin{array}{rrcrr} & \delta_4^+ + T_{Hold} \geq T_{HI} & \wedge & \delta_1^- > 0 \\ \wedge & T_{HI} \geq \delta_4^- + T_{Hold} & \wedge & \delta_3^- > 0 \\ \wedge & T_{Hold} \geq \delta_3^- + \delta_4^- & \wedge & \delta_3^+ \geq T_{Hold} \\ \wedge & T_{Setup} > \delta_1^+ & \wedge & T_{HI} > \delta_3^+ \end{array}$$

After instantiation of all parameters except δ_3^+ and δ_4^+:

$$\delta_3^+ \geq 17 \quad \wedge \quad \delta_3^+ < 24 \quad \wedge \quad \delta_4^+ \geq 7$$

The trace set of this tile is given below:



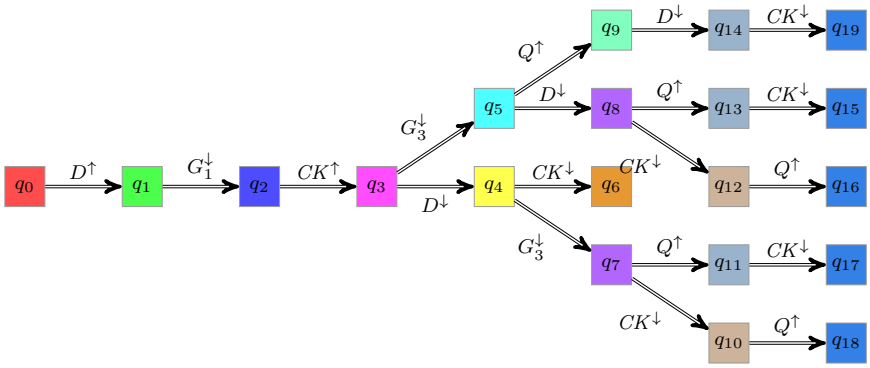
Tile 7. This tile corresponds to the values of the parameters verifying the following constraint:

$$\begin{array}{rrcrr} & \delta_4^+ + T_{Hold} \geq T_{HI} & \wedge & \delta_1^- > 0 \\ \wedge & T_{HI} \geq \delta_4^- + T_{Hold} & \wedge & \delta_3^- > 0 \\ \wedge & T_{Hold} \geq \delta_3^- + \delta_4^- & \wedge & T_{Setup} > \delta_1^+ \\ \wedge & \delta_3^+ \geq T_{HI} \end{array}$$

After instantiation of all parameters except δ_3^+ and δ_4^+:

$$\delta_3^+ \geq 24 \quad \wedge \quad \delta_4^+ \geq 7$$

The trace set of this tile is given below:



Note that this trace set corresponds to the whole set of reachable states, computed using the parametric reachability analysis (Figure 3.4).

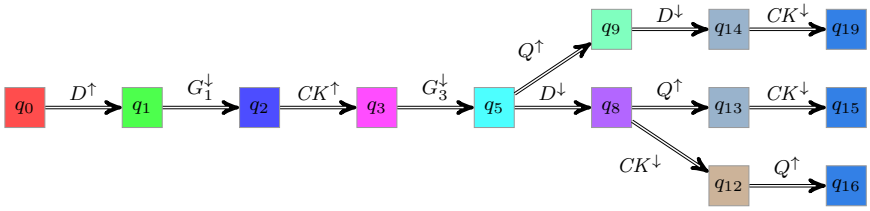
Tile 8. This tile corresponds to the values of the parameters verifying the following constraint:

$$\begin{array}{lll} \delta_3^+ + \delta_4^+ \geq T_{HI} & \wedge & \delta_1^- > 0 \\ \wedge \quad T_{Hold} \geq \delta_3^- + \delta_4^- & \wedge & \delta_3^- > 0 \\ \wedge \quad T_{Setup} > \delta_1^+ & \wedge & T_{Hold} > \delta_3^+ \end{array}$$

After instantiation of all parameters except δ_3^+ and δ_4^+:

$$\delta_3^+ + \delta_4^+ \leq 24 \quad \wedge \quad \delta_3^+ \geq 8 \quad \wedge \quad \delta_3^+ < 17$$

The trace set of this tile is given below:



Cartography. The cartography is depicted in Figure 3.5.

According to the nature of the trace sets, we can easily partition the tiles into good and bad tiles. Tiles 1 to 3 are good tiles, and tiles 4 to 8 are bad tiles.

After partition into good and bad tiles, we can infer the constraint on δ_3^+ and δ_4^+ corresponding to all the good behaviors:

$$\delta_3^+ + \delta_4^+ \leq 24 \quad \wedge \quad \delta_3^+ \geq 8 \quad \wedge \quad \delta_4^+ \geq 3$$

3.3 Second Environment

We now consider a variant of this case study, using the same model, as depicted in Figure 3.1 page 11, the same timing parameters, and the new environment depicted in Figure 3.6.

This new environment starts from $D = g_2 = Q = 1$ and $CK = g_1 = g_3 = 0$, with the following ordered sequence of actions for inputs D and CK: $D^\downarrow$, $CK^\uparrow$,

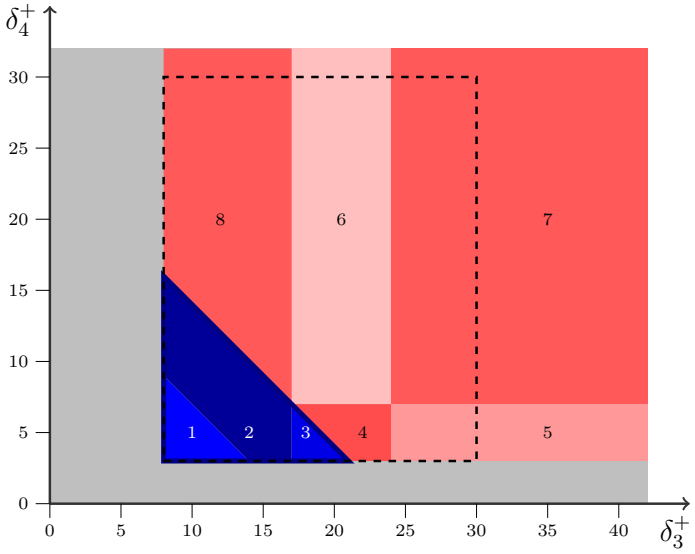


Figure 3.5: Behavioral cartography of the flip-flop according to δ_3^+ and δ_4^+

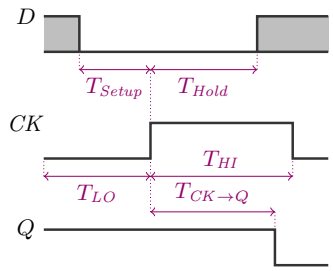


Figure 3.6: Environment for the flip-flop circuit with $D = 0$

$D^{\uparrow}$, $CK^{\downarrow}$. Therefore, we have the implicit constraint $T_{Setup} \leq T_{LO} \wedge T_{Hold} \leq T_{HI}$.

The initial location q_0 corresponds to the initial levels of the signals according to the environment. The initial constraint K_0 corresponds to:

$$T_{Setup} \leq T_{LO} \wedge T_{Hold} \leq T_{HI} \wedge \bigwedge_{i=1,..,4} \delta_i^- \leq \delta_i^+$$

As in [17], we now consider that the circuit has a *good behavior* if every trace contains both $Q^{\downarrow}$ and $CK^{\downarrow}$, and $Q^{\downarrow}$ occurs before $CK^{\downarrow}$. We are interested in identifying parameter valuations for T_{Hold} and δ_2^+ for which the system has such a good behavior. As a consequence, we perform a behavioral cartography of the system according to parameters T_{Hold} and δ_2^+. We consider the following V_0:

$$T_{Hold} \in [0, 50] \quad \text{and} \quad \delta_2^+ \in [5, 40].$$

The other parameters are instantiated as follows (note that this reference valuation is not the same as in the previous section):

$$T_{HI} = 40 \quad T_{LO} = 20 \quad T_{Setup} = 19 \quad \delta_1^- = 18 \quad \delta_1^+ = 18$$
$$\delta_2^- = 5 \quad \delta_3^- = 8 \quad \delta_3^+ = 10 \quad \delta_4^- = 3 \quad \delta_4^+ = 7$$

The cartography is computed automatically by IMITATOR II. We then partition the tiles into good and bad. This partition is depicted under a graphical form in Figure 3.7, where the light red (resp. dark blue) zones correspond to the bad (resp. good) values of the parameters.

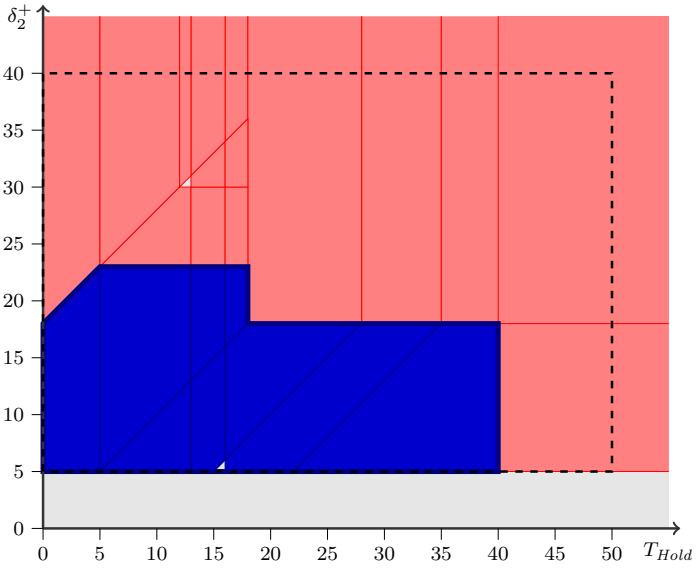


Figure 3.7: Behavioral cartography of the flip-flop for parameters T_{Hold} and δ_2^+

First note that all outer zones are infinite: as a consequence, the cartography covers the whole[1] dense real-valued set of parameters outside V_0. However, there are two holes within V_0, i.e., zones not covered by any tile. The full coverage can be achieved using two different methods:

1. by calling manually the inverse method on one (non-integer) point within each of the two holes, or
2. by performing again the cartography using a tighter grid than integers (actually calling the inverse method on rational points multiple of $1/3$ is enough in this case).

Both methods allow us to get similarly the full coverage of the parametric space within V_0. We do not redraw here the cartography again. The hole in the bad zone turns out to correspond to a bad behavior; similarly, the hole in the good zone turns out to correspond to a good behavior.

As a consequence, one is now able to infer the following constraint corresponding to the set of parameters for which the flip-flop circuit behaves well:

$$\begin{array}{rrcl} & 5 \leq \delta_2^+ \leq 18 & \wedge & 0 \leq T_{Hold} \leq 40 \\ \vee & 18 \leq \delta_2^+ \leq 23 & \wedge & \delta_2^+ - 18 \leq T_{Hold} \leq 18 \end{array}$$

[1] Apart from the irrelevant zone originating from the model ($\delta_2^+ < 5$).

This constraint corresponds to the maximal constraint solving the good parameters problem for parameters T_{Hold} and δ_2^+ for this case study, because the whole parameter domain has been covered by the tiles. Also note that this constraint is not under convex form.

Chapter 4

And–Or Circuit

4.1 Presentation

This example deals with an "AND–OR" circuit described in [16] and depicted in Figure 4.1 (left). It is composed of 2 gates (one "AND" gate and one "OR" gate) which are interconnected in a cyclic way. The environment, depicted in Figure 4.1 (right), corresponds to 2 input signals a and b, with cyclic alternating rising edges and falling edges.

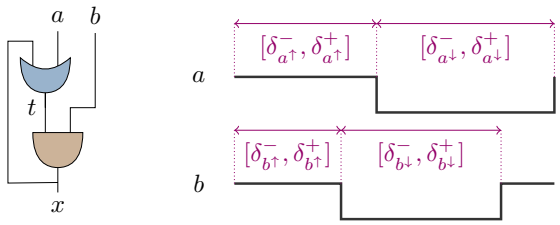


Figure 4.1: AND–OR circuit (left) and its environment (right)

Each rising (resp. falling) edge of signal a, is denoted by $a^\uparrow$ (resp. $a^\downarrow$), and similarly for b, t, x. The delay between the rising edge $a^\uparrow$ and the falling edge $a^\downarrow$ (resp. between $a^\downarrow$ and $a^\uparrow$) of signal a is in $[\delta^-_{a\uparrow}, \delta^+_{a\uparrow}]$ (resp. $[\delta^-_{a\downarrow}, \delta^+_{a\downarrow}]$), and similarly[1] for b. The traversal of the gate "OR" gate takes also a delay in $[\delta^-_{Or}, \delta^+_{Or}]$, and likewise for the "AND" gate. Those 12 timing parameters are bound by the following implicit constraint:

$$\begin{array}{ccccccc} & \delta^-_{And} \leq \delta^+_{And} & \wedge & \delta^-_{Or} \leq \delta^+_{Or} & \wedge & \delta^-_{a\downarrow} \leq \delta^+_{a\downarrow} \\ \wedge & \delta^-_{a\uparrow} \leq \delta^+_{a\uparrow} & \wedge & \delta^-_{b\downarrow} \leq \delta^+_{b\downarrow} & \wedge & \delta^-_{b\uparrow} \leq \delta^+_{b\uparrow} \end{array}$$

Each of the 2 gates is modeled by a PTA, as well as the environment. We consider an inertial model for gates, where any change of the input may lead to a change of the output (after some delay). The PTA $\mathcal{A}$ modeling the system results from the composition of those 3 PTAs.

[1] Note however that the interval $[\delta^-_{b\uparrow}; \delta^+_{b\uparrow}]$ has a slightly different meaning, because it corresponds to the interval of delays between the rise of a and the fall of b, as shown in Figure 4.1 (right). This choice allows an easier modeling, and a more frequent termination of the analysis.

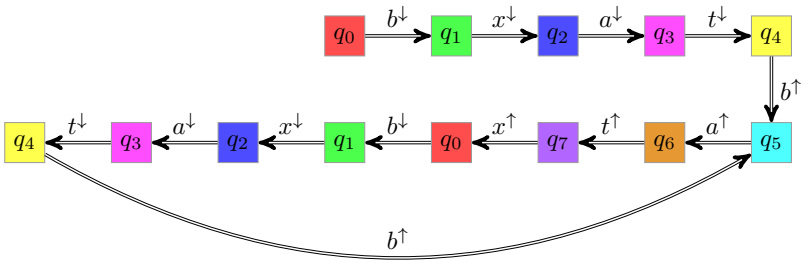


Figure 4.2: Trace of the AND–OR circuit under π_0

Location	a	b	t	x
q_0	1	1	1	1
q_1	1	0	1	1
q_2	1	0	1	0
q_3	0	0	1	0
q_4	0	0	0	0
q_5	0	1	0	0
q_6	1	1	0	0
q_7	1	1	1	0

Table 4.1: Locations of the AND–OR circuit

A bad state expresses the fact that the rising edge of output signal x occurs before the rising edge of a within the same cycle. We set the parameters to the following values, ensuring that the bad state is not reachable:

$$\begin{array}{llll} \delta^-_{a^\uparrow} = 13 & \delta^+_{a^\uparrow} = 14 & \delta^-_{a^\downarrow} = 16 & \delta^+_{a^\downarrow} = 18 \\ \delta^-_{b^\uparrow} = 7 & \delta^+_{b^\uparrow} = 8 & \delta^-_{b^\downarrow} = 19 & \delta^+_{b^\downarrow} = 20 \\ \delta^-_{And} = 3 & \delta^+_{And} = 4 & \delta^-_{Or} = 1 & \delta^+_{Or} = 2 \end{array}$$

We consider an environment starting at location q_0 with $a = b = x = t = 1$, and the following repeated cycle of alternating rising and falling edges of a and b: $b^\downarrow, a^\downarrow, b^\uparrow, a^\uparrow$. For the given environment and the valuation π_0, the set of traces of the system is depicted in Figure 4.2 under the form of an oriented graph, where q_i, $0 \leq i \leq 7$, are locations of $\mathcal{A}$. The value of the signals of the system for each location q_i is given in Table 4.1. We can check that, in this graph, the bad state is not reached, i.e., the rising edges and falling edges of a, b, x alternate properly.

4.2 Synthesis of Constraints

4.2.1 Reference Valuation

Using IMITATOR II applied to the PTA $\mathcal{A}$ modeling the system and the reference instantiation π_0, the following constraint K_0 is computed after 14 iterations:

$$\begin{array}{llll} & \delta^-_{b^\downarrow} + \delta^-_{b^\uparrow} > \delta^+_{Or} + \delta^+_{a^\uparrow} & \wedge & \delta^-_{b^\uparrow} > \delta^+_{And} + \delta^+_{Or} \\ \wedge & \delta^+_{a^\downarrow} + \delta^+_{a^\uparrow} \geq \delta^-_{b^\downarrow} + \delta^-_{b^\uparrow} & \wedge & \delta^-_{a^\uparrow} > \delta^+_{And} + \delta^+_{b^\uparrow} \end{array}$$

Under any instantiation of the parameters $\pi \models K_0$, the set of traces under π is guaranteed to be identical to the set of traces under π_0 given in Figure 4.2 and, therefore, does not reach any bad state. In [16], the synthesized constraint is not given.

This constraint gives a criterion of robustness for this system by guaranteeing that, for values of the parameters around the reference valuation, the system will still behave well. It is in particular interesting to note that several parameters do not appear in the constraint synthesized (and are actually only bound by the implicit constraint given earlier). This is the case of parameters $\delta^-_{a^\downarrow}$, $\delta^+_{b^\downarrow}$, δ^-_{And} and δ^-_{Or}. This means that, for the considered environment, the value of these parameters has no influence on the behavior of the system.

4.3 Inverse Method

4.3.1 Reference Valuation

Using our program IMITATOR II applied to the PTA $\mathcal{A}$ modeling the system and the reference valuation π_0, the following constraint is computed after 14 iterations in 0.15 second (13 reachable states with 13 transitions):

$$\begin{array}{llll} & \delta^-_{b^\downarrow} + \delta^-_{b^\uparrow} > \delta^+_{Or} + \delta^+_{a^\uparrow} & \wedge & \delta^-_{b^\uparrow} > \delta^+_{And} + \delta^+_{Or} \\ \wedge & \delta^+_{a^\downarrow} + \delta^+_{a^\uparrow} \geq \delta^-_{b^\downarrow} + \delta^-_{b^\uparrow} & \wedge & \delta^-_{a^\uparrow} > \delta^+_{And} + \delta^+_{b^\uparrow} \end{array}$$

Under any valuation of the parameters $\pi \models K_0$, the set of traces under π is guaranteed to be identical to the set of traces under π_0 given in Figure 4.2 and, therefore, does not reach any bad state. We notice that, whatever the algorithm randomly chooses at each iteration for the inequality $\neg J$, the constraint finally computed always remains equivalently the same. In [16], the constraint synthesized is not given.

4.3.2 Other Valuations

Note that a reachability analysis loops when applied to the following reference valuation (differences with π_0 are depicted in **bold**):

$$\begin{array}{llll} \delta^-_{a^\uparrow} = 13 & \delta^+_{a^\uparrow} = 14 & \delta^-_{a^\downarrow} = 16 & \delta^+_{a^\downarrow} = 18 \\ \delta^-_{b^\uparrow} = \mathbf{2} & \delta^+_{b^\uparrow} = \mathbf{3} & \delta^-_{b^\downarrow} = 19 & \delta^+_{b^\downarrow} = 20 \\ \delta^-_{And} = 3 & \delta^+_{And} = 4 & \delta^-_{Or} = 1 & \delta^+_{Or} = 2 \end{array}$$

Chapter 5

Latch Circuit

5.1 Description

We consider in this section a latch circuit studied in the case of ANR project VALMEM. This circuit, depicted in Figure 5.1, contains 5 elements: 2 "NOT" gates (viz., Not_1 and Not_2), one "XOR" gate (viz., element Xor), one "NAND" gate (viz., element And), as well as one "latch" element (viz., element $Latch$).

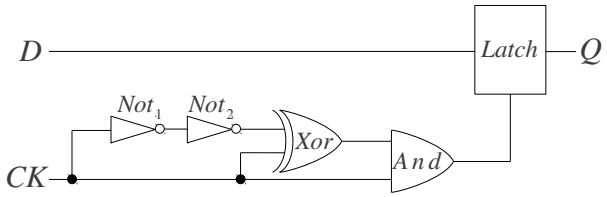


Figure 5.1: A latch circuit

Each of the four gates has a constant delay for a change of an input leading to a rising edge in the output, and another constant delay for a change of an input leading to a falling edge of the input. For example, when the input of the gate Not_1 is equal to 0 and raises, then the output will change after delay $\delta_{Not1^{\uparrow}}$. If the input is equal to 1 and falls, the delay before the output changes is $\delta_{Not1^{\downarrow}}$, and similarly for the other 3 gates. The latch has a single constant delay $\delta_{Latch^{\uparrow}}$ corresponding to the time needed between a change of its inputs and the raise of Q. There are 4 other parameters (viz., $T_{HI}, T_{LO}, T_{Setup}$, and T_{Hold}) used to model the environment. The rising (resp. falling) edge of signal D is denoted by $D^{\uparrow}$ (resp. $D^{\downarrow}$), and similarly for signals CK and Q.

We consider an environment starting from $D = CK = Q = 0$, with the following ordered sequence of actions for inputs D and CK: $D^{\uparrow}$, $CK^{\uparrow}$, $D^{\downarrow}$, $CK^{\downarrow}$, as depicted in Figure 5.2. Therefore, we have the implicit constraint $T_{Setup} \leq T_{LO} \wedge T_{Hold} \leq T_{HI}$.

Each gate is modeled by a PTA, as well as the environment. The PTA $\mathcal{A}$ modeling the system results from the composition of those 6 PTAs.

The following valuation π_0 of the 13 parameters (in ps) was extracted from the circuit description by simulation computed in the VALMEM project:

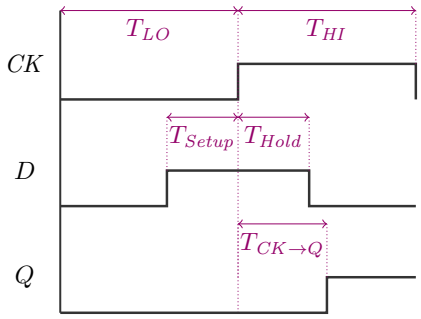


Figure 5.2: Environment for the latch circuit

$$\begin{array}{llll} T_{HI} = 1000 & T_{LO} = 1000 & T_{Hold} = 350 & T_{Setup} = 1 \\ \delta_{Not1\uparrow} = 219 & \delta_{Not1\downarrow} = 147 & \delta_{Not2\uparrow} = 155 & \delta_{Not2\downarrow} = 163 \\ \delta_{Xor\uparrow} = 147 & \delta_{Xor\downarrow} = 416 & \delta_{And\uparrow} = 80 & \delta_{And\downarrow} = 155 \\ \delta_{Latch\uparrow} = 240 & & & \end{array}$$

A bad state corresponds to the fact that the output signal Q has not changed before the end of the cycle of signal CK. Under this valuation, we can show that the system does not reach any bad state.

5.2 Inverse Method

Applying IMITATOR II to the PTA version of this model and the reference valuation π_0, the following constraint K_0 is synthesized after 12 iterations in 0.345 second (18 reachable states with 17 transitions):

$$\begin{array}{lr} \wedge & \delta_{Xor\downarrow} + \delta_{Not2\uparrow} + \delta_{Not1\downarrow} > T_{Hold} \\ \wedge & \delta_{Latch\uparrow} + \delta_{And\uparrow} > \delta_{Not2\uparrow} + \delta_{Not1\downarrow} \\ \wedge & \delta_{Not1\downarrow} > \delta_{And\uparrow} \\ \wedge & T_{Hold} > \delta_{Latch\uparrow} + \delta_{And\uparrow} \\ \wedge & T_{LO} \geq T_{Setup} \\ \wedge & T_{HI} \geq \delta_{And\downarrow} + \delta_{Xor\downarrow} + \delta_{Not2\uparrow} + \delta_{Not1\downarrow} \end{array}$$

Under this constraint, the system has the same behavior as under the reference valuation, and therefore guarantees a good behavior of the system. Suppose now that we are interested in minimizing the T_{Hold} value, provided the system keeps its good behavior. Such a minimization has the following practical interest: since T_{Hold} corresponds to the duration during which the input signal D should be stable before the rise of the clock, minimizing this value allows to integer this portion of circuit in a *faster* environment. By instantiating all the parameters in K_0 with their value in π_0, except T_{Hold}, we get the following constraint:

$$320 < T_{Hold} < 718$$

So we can minimize the value of T_{Hold} to 321, and we guarantee that the system will have exactly the same behavior as before.

Chapter 6

SPSMALL Memory

We consider in this section the SPSMALL memory, which is a memory circuit sold by ST-Microelectronics. This memory has been first studied in the MEDEA+BLUEBERRIES (T126) European project involving ST-Microelectronics and the LSV laboratory (École Normale Supérieure de Cachan). It was then studied in the ANR VALMEM project involving, besides ST-Microelectronics and LSV, the LIP 6 laboratory (Université Pierre et Marie Curie).

6.1 Description

The SPSMALL memory actually corresponds to a *class* of small memories with a maximum total capacity of 64 kbits. Each instance of the memory is built by a parametrized compiler, where the number of words and the size of the words are parameters[1]. The number of words is ranking from 3 to 512 words, and the number of bits from 2 to 256 bits. We consider throughout this section the smallest memory consisting in 3 words of 2 bits (or abstractions of it), which leads to a netlist of 305 transistors.

The SPSMALL is manually built directly at the transistor level. Indeed, in order to be able to optimize the memory array part of the circuit, one must tune it manually. Moreover, the control logic and the decoder logic uses hand-made cells, and these complex structures cannot be automatically generated.

Our Approach. Before describing the memory and the model considered in this section, we first give the methodology used in the VALMEM project.

As depicted in Figure 6.1, a description of the memory under the form of a transistor netlist is given by ST-Microelectronics. Then, a functional abstraction generates a description of the memory in the functional description language VHDL, using automatic techniques developed by LIP 6 [35]. At the same time, timings are extracted under the form of traversal delays of the elements. The next step is the translation by LIP 6 of the VHDL code into a network of (instantiated) timed automata, using the tool Vhdl2Ta [12] developed in

[1] This notion of parameter is not anyhow linked to the *timing parameters* (set P) mentioned throughout this document. We will study this memory for a given instance of these words and size parameters, and the parameters that we will consider correspond to internal timing delays, as for the other case studies.

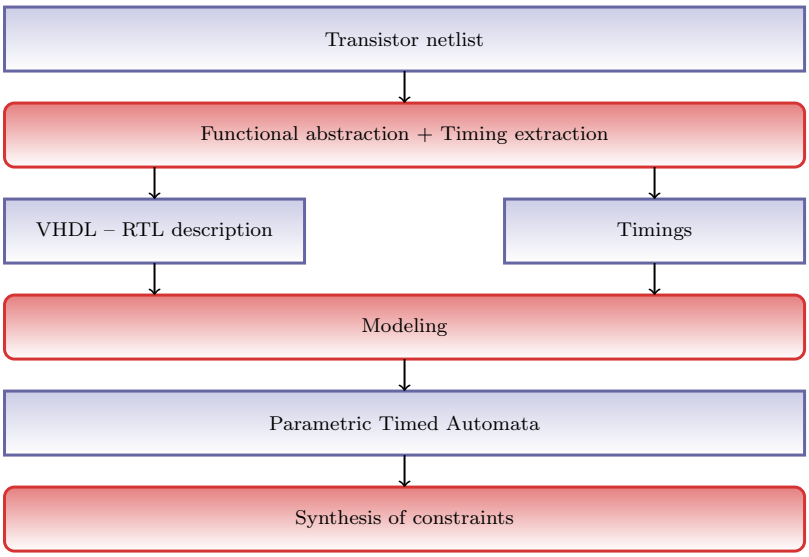


Figure 6.1: Methodology of the VALMEM project

the framework of the VALMEM project. Finally, using a parametric version of those timed automata, and the reference instantiation of the parameters, we synthesize using IMITATOR II constraints guaranteeing a good behavior for the memory. Although we will mostly focus on this latter task in this section, we recall various information on the global process for the sake of comprehension.

Level of Modeling. We borrow part of the following description to [10, 14]. A memory circuit aims at storing data at some addressed locations, and is associated with two operations: *write* and *read*. A memory can be modeled at different levels of complexity, e.g., in an increasing order: at the functional block level, at the "latch" level, at the gate level, or at the transistor level. For the SPSMALL memory, the model can thus be implemented using 3 main components at the block level (see [10]), a few dozens of components at the latch level, about one hundred components at the gate level, or 305 components at the transistor level. There is a tradeoff in finding the appropriate level of modeling. The lower the level of modeling is, the more faithful to the reality the model is, but the more difficult the verification process is. In [14] and in this section, we choose to represent the memory at the latch level. The advantage is to limit the number of components at a reasonable size, and to have a "schematics" describing the architecture of the memory at this level, which closely corresponds to the VHDL code automatically produced.

In order to better illustrate the complexity of this memory, we give in Figure 6.2 a graphical representation of the memory at the transistor level.

Inputs and Outputs. The SPSMALL memory circuit has several input ports and one output port. The signals driven by input ports are:

- CK, the signal of the periodic clock;

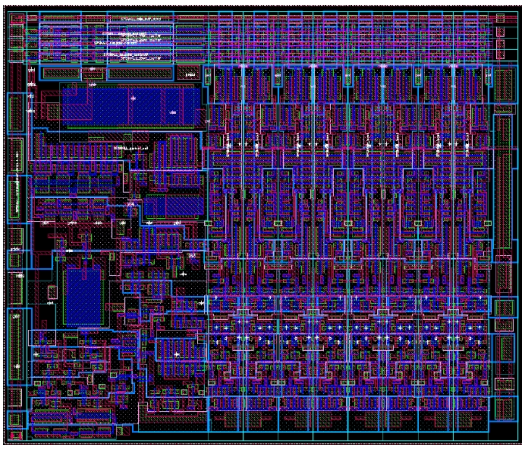


Figure 6.2: Transistor representation of the SPSMALL memory

- D, the n-bit width signal representing the data to be stored;
- A, the $\log_2(m)$-bit width signal representing the address of an internal memory location;
- WEN, the 1-bit width signal representing either a write or a read operation.

The signal driven by the output port is Q (of n-bit width). The data are stored in a memory array composed of $m \times n$ memory points. A memory location is a collection of n memory points. The write operation ($WEN = 0$ when CK is rising) writes the value of D in the internal memory location selected by A, and propagates D on output port Q. Such a memory is called a *write-through* memory. The read operation ($WEN = 1$ when CK is rising) outputs on port Q a copy of the data stored in the memory location selected by A.

Timing Parameters. We consider here the *write* operation. The environment for this operation is depicted in Figure 6.3.

The duration of the clock cycle is parameterized by T_{HI} (duration of the high edge) and T_{LO} (duration of the low edge). We study this operation for two clock cycles. The parameter t^{WEN}_{setup} corresponds to the time during which the WEN signal should be stable before the beginning of the second clock cycle, i.e., the second rise of CK. Similarly, the parameter t^{D}_{setup} corresponds to the time during which the D signal should be stable before the beginning of the second clock cycle. Finally, the parameter $T_{CK \to Q}$ corresponds to the maximal time between the beginning of the second clock cycle and the rise of the output signal Q. Besides these 5 parameters, the SPSMALL memory is characterized by other parameters corresponding to the traversal delays of the gates and latches of the circuit.

Each of those parameters is given a valuation. Parameter valuations corresponding to the environment (viz., T_{HI}, T_{LO}, t^{WEN}_{setup}, t^{D}_{setup}) are taken from the datasheet of the memory given by ST-Microelectronics. Parameter valuations

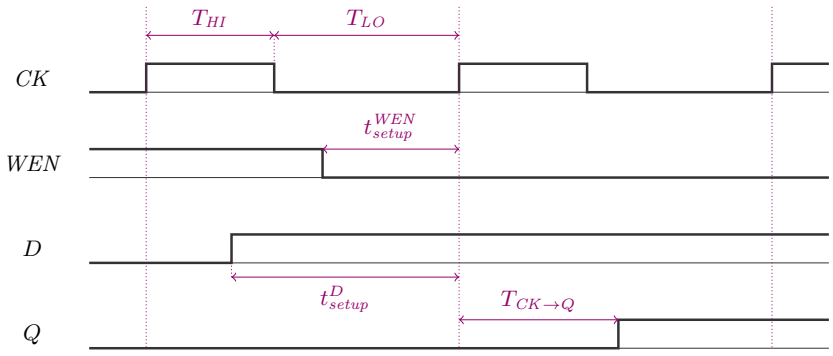


Figure 6.3: Environment for the *write* operation of SPSMALL

corresponding to internal delays are synthesized as follows. In the BLUEBERRIES project, they were manually computed by electrical simulation for a single configuration of the environment. In the VALMEM project, there are automatically retrieved using the transistor netlist (see Figure 6.1 page 28): from all possible inputs and outputs for a given component, only two values are kept, namely the lower and the upper bounds of the traversal time taken on all the possible configurations. Although this gives suitable results for the gates, the bounds are sometimes far from each other for memory points, thus weakening the precision of the verification. Those internal delays depend on the size of the transistors and on the technology used. This is the value of those internal delays which induces the possible values of the environment parameters.

Actually, this memory circuit has *two* different implementations for the same architecture. In other words, for the same schematics of gates and latches, there are two different sets of valuations of the parameters (i.e., environment parameters and traversal delays). The first implementation (SP1) corresponds to a fast component with a high power consumption, whereas the second implementation (SP2) corresponds to a slower component with a lower power consumption.

6.2 A Short History

The SPSMALL memory was first studied in [10], where the authors verify this memory component modeled by timed automata, using the real-time model checkers HyTech and Uppaal. In particular, the authors take into account the electrical propagation delays through gates and along wires. The authors propose an abstraction of the memory sufficiently small to be (manually) described in the model-checker Uppaal. Then they verify that, for some internal timings given by ST-Microelectronics, the read and write access timings are correct. Moreover, they verify that those access timings (viz., $T_{CK\to Q}$ for the write operation) are *optimal* by showing that the memory model has correct behaviors with those timings, whereas incorrect behaviors occur when choosing smaller timings. This is done by manually decreasing those timings, and checking that the behavior remains correct. Note that the authors consider here only *integer* timings, and do not investigate what is happening between two integer values.

The SPSMALL memory was then studied in [15], where the authors propose a high-level formalism, called Abstract Functional and Timing Graph (AFTG), for describing the memory. This formalism allows in particular to combine logical functionality and timing. After translation of the AFTG into the form a timed automaton, the authors are able to compute the response times of the modeled memory, and check their consistency with the values specified in the datasheet. The authors then go one step further by showing not only that the access timings are correct, but they also give the *optimal* input setup and hold timings such that the access timings remain correct. This is done by manually decreasing those input timings, and check that the access timings remain correct. Note that the authors also consider here only *integer* timings, and do not investigate what is happening between two integer values.

In [14], the authors then manually synthesize constraints on the setup and internal timings seen as *parameters* guaranteeing that the response times to a write command or a read command lie between certain bounds. Those constraints, derived using the SP1 implementation of the memory, can be immediately applied to other instances of the parameters to verify the behavior of other versions of the memory, such as SP2. Contrarily to the first two approaches, this work allows to consider dense (i.e., real) values for the timings, and give a criterion of robustness to the timings of the memory.

Our aim is to *automatically* derive constraints on the internal timings seen as parameters, such that the memory behaves well. We study in the forthcoming sections several abstractions of the SPSMALL memory.

6.3 Inverse Method

6.3.1 Manually Abstracted Model

Description. We consider here a model manually abstracted, close to the model considered in [15]. We recall the model considered in [15] in Figure 6.4 under the form of an AFTG. This model was abstracted in order to consider that only one bit is stored. As a consequence, D becomes a 1-bit signal. Furthermore, we consider only the portion of the circuit relevant to the *write* operation.

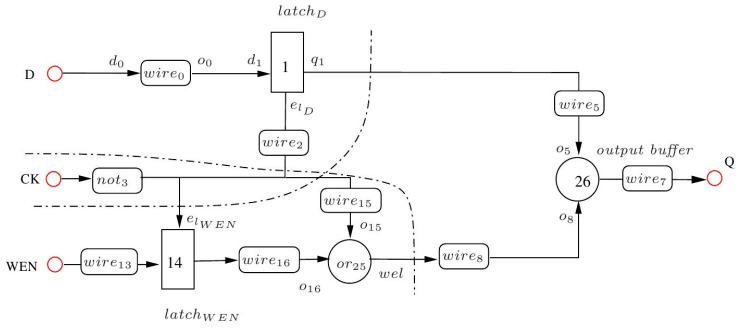


Figure 6.4: Abstract model of the SPSMALL memory (write operation)

Although the model we consider here is close to the model considered in [15], a major difference with the model of [15] though is that delays are not only asso-

ciated to latches and wires anymore, but to latches, wires and *gates*, depending on the components. This model has been designed partially automatically from the VHDL code, using abstractions. This VHDL source code (available in [37]) was itself manually written.

This model, depicted in Figure 6.5, results in 9 components. Components $delay_D$ and $delay_{WEN}$ are delays (i.e., the logical functionality is the identity), components NOT_1, NOT_2 and NOT_3 are "NOT" gates, WEL is an "OR" gate, and components $delay_{WEN}$, $latch_D$ and net_{27} are latches. A further difference with the model considered in [15] is that several components have been grouped together in order to avoid the state-space explosion problem[2]. For example, several delays associated to wires have been incorporated into the previous elements: this is the case, e.g., of component $wire_5$ from Figure 6.4, the delay of which has been incorporated into the element $latch_D$, resulting in only one component ($latch_D$) in our model depicted in Figure 6.5.

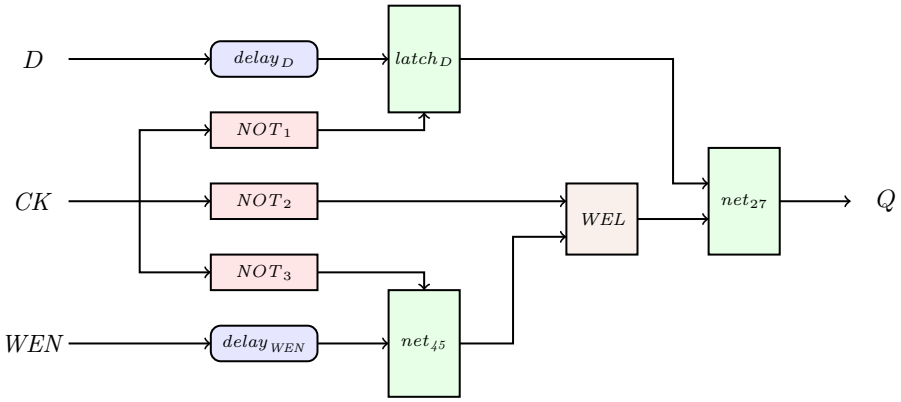


Figure 6.5: PTAs modeling the write operation of SPSMALL

Each of the components depicted in Figure 6.5 (wires, gates, latches) is modeled using a PTA. The translation of the gates into PTAs has been performed automatically using a preliminary version of VHDL2TA. The other components were manually written, and so was the composition of all components together. The environment is also modeled using a PTA. This results in a model containing 10 automata, 10 clocks and 26 parameters corresponding to the traversal delays of the components and the environment. Contrary to [15], the PTAs modeling the gates are actually *complete*, in the sense that all possible configurations and transitions are modeled, not only the configurations that will be met for a precise environment, as it was the case in [15]. This is in particular due to the automatic generation of the PTAs.

Implementation SP1. We give below the set of parameter valuations (say, π_1) coming from the implementation SP1 and adapted to this first model (timings are given in tens of pico-seconds).

[2]This model was actually first designed to be analyzed using HYTECH, which can hardly accept more than 10 components modeled by PTAs in parallel. However, analyzing this model using IMITATOR II is performed easily in a couple of seconds.

$$
\begin{array}{rrr}
d\_up\_q\_0 = 21 & d\_dn\_q\_0 = 20 & d\_up\_net27 = 0 \\
d\_dn\_net27 = 0 & d\_up\_d\_inta = 22 & d\_dn\_d\_inta = 45 \\
d\_up\_wela = 0 & d\_dn\_wela = 22 & d\_up\_net45a = 5 \\
d\_dn\_net45a = 4 & d\_up\_net13a = 19 & d\_dn\_net13a = 13 \\
d\_up\_net45 = 21 & d\_dn\_net45 = 22 & d\_up\_d\_int = 14 \\
d\_dn\_d\_int = 18 & d\_up\_en\_latchd = 28 & d\_dn\_en\_latchd = 32 \\
d\_up\_en\_latchwen = 5 & d\_dn\_en\_latchwen = 4 & d\_up\_wen\_h = 11 \\
d\_dn\_wen\_h = 8 & d\_up\_d\_h = 95 & d\_dn\_d\_h = 66 \\
T_{HI} = 45 & T_{LO} = 65 & t^{D}_{setup} = 108 \\
t^{WEN}_{setup} = 48 & &
\end{array}
$$

Constraint. Applying IMITATOR II to this model and the reference valuation π_1 (corresponding to the SP1 implementation), one synthesizes the following constraint K_1 after 32 iterations (31 reachable states with 30 transitions):

$$
\begin{array}{ll}
 & T_{HI} + d\_up\_net13a > d\_dn\_net13a + d\_dn\_wela + d\_up\_net27 + d\_up\_q\_0 \\
\wedge & T_{LO} > d\_up\_en\_latchd + d\_up\_d\_int + d\_up\_d\_inta \\
\wedge & t^{D}_{setup} + d\_dn\_en\_latchd > d\_up\_d\_h + d\_up\_d\_int + d\_up\_d\_inta \\
\wedge & t^{WEN}_{setup} + d\_up\_d\_h > t^{D}_{setup} + d\_dn\_wen\_h + d\_dn\_net45 + d\_dn\_net45a + d\_up\_wela \\
\wedge & T_{LO} + d\_dn\_wen\_h > t^{WEN}_{setup} + d\_up\_net13a + d\_up\_wela \\
\wedge & T_{HI} > d\_dn\_net13a + d\_dn\_wela \\
\wedge & T_{LO} > t^{WEN}_{setup} + d\_up\_en\_latchwen \\
\wedge & t^{D}_{setup} > d\_up\_d\_h \\
\wedge & t^{D}_{setup} \geq T_{LO} \\
\wedge & T_{LO} + T_{HI} \geq t^{D}_{setup} \\
\wedge & d\_dn\_en\_latchwen \geq 0 \\
\wedge & d\_up\_en\_latchwen \geq 0 \\
\wedge & t^{WEN}_{setup} + d\_up\_en\_latchd > T_{LO} + d\_dn\_wen\_h \\
\wedge & d\_dn\_net13a > d\_dn\_en\_latchwen \\
\wedge & t^{WEN}_{setup} + d\_up\_net13a > T_{LO} \\
\wedge & d\_up\_en\_latchwen + d\_up\_net45 + d\_up\_net45a > d\_up\_en\_latchd \\
\wedge & d\_dn\_net13a + d\_dn\_wela > d\_dn\_en\_latchd \\
\wedge & d\_up\_wela \geq 0 \\
\wedge & t^{D}_{setup} + d\_up\_en\_latchd + d\_dn\_d\_int + d\_dn\_d\_inta > T_{LO} + d\_up\_d\_h \\
\wedge & d\_up\_en\_latchd + d\_up\_d\_int + d\_up\_d\_inta > d\_up\_en\_latchwen + d\_dn\_net45 + d\_dn\_net45a \\
\wedge & d\_up\_d\_h + d\_up\_d\_int + d\_up\_d\_inta > t^{D}_{setup} + d\_dn\_net13a \\
\wedge & d\_dn\_net13a + d\_dn\_wela + d\_up\_net27 + d\_up\_q\_0 > T_{HI} + d\_up\_en\_latchwen
\end{array}
$$

Interpretation. The main advantage of the constraint synthesized by IMITATOR II is that it allows to show the link between the internal timing delays and the external values of the environment. Indeed, the timing parameters corresponding to the environment are *constrained* by the internal traversal delays of the gates, wires and latches. Despite the complex form of the constraint synthesized, it is possible to give an interpretation for some of the inequalities.

First of all, some inequalities are actually synthesized because of the environment that we consider. Inequalities such as $t^{D}_{setup} \geq T_{LO}$ or $T_{LO} + T_{HI} \geq t^{D}_{setup}$ come from the way we modeled the environment, and are bound by the *model* more than the system.

Moreover, other inequalities can be interpreted as a guarantee on the *order* of the events. Recall that our inverse method guarantees the same trace sets and, as a consequence, the same ordering of events. For example, the inequality $t^{WEN}_{setup} + d\_up\_en\_latchd > T_{LO} + d\_dn\_wen\_h$ implies that the (timed) path through wire $delay_{WEN}$ is greater than the path through gate NOT_3. In other words, the upper input of latch net_{45} must change before its left input.

Optimization. By replacing within K_1 every parameter except t^D_{setup} and t^{WEN}_{setup} by its valuation as defined in π_1, one gets the following constraint on t^D_{setup} and t^{WEN}_{setup}:

$$46 < t^{WEN}_{setup} < 54 \quad \wedge \quad 99 < t^D_{setup} \leq 110 \quad \wedge \quad t^D_{setup} < t^{WEN}_{setup} + 61$$

It is then interesting to *minimize* those setup timings. Indeed, if one minimizes the setup duration of the input signals without changing the overall behavior of the system, then this means that the memory can be inserted in a faster environment where the input signals change faster. One can thus minimize t^D_{setup} and t^{WEN}_{setup} according to K_1 as follows:

$$t^{WEN}_{setup} = 47 \quad \wedge \quad t^D_{setup} = 100$$

By comparison with the original parameter valuation π_1 (viz., $t^D_{setup} = 108$ and $t^{WEN}_{setup} = 48$), this results in a decreasing of the setup timing of signal D (resp. WEN) of 7.4 % (resp. 2.1 %).

In [15], the authors compute a minimum value of 95 for t^D_{setup}, and a minimum value of 29 for t^{WEN}_{setup}. As a consequence, our values may still be improved. Improving those values for this model will be the purpose of Section 6.4.

6.3.2 Automatically Generated Model

This second version of the SPSMALL memory is a more complete model of the memory, representing not only the portion of the memory corresponding to the *write* operation, but the complete architecture. As in the previous section, this model was abstracted in order to consider that only one bit is stored. As a consequence, D becomes a 1-bit signal. We give in Figure 6.6 the schematics from [15] depicting the wires, gates and latches under the form of an Abstract Functional and Timing Graph, and corresponding to the complete architecture of SPSMALL.

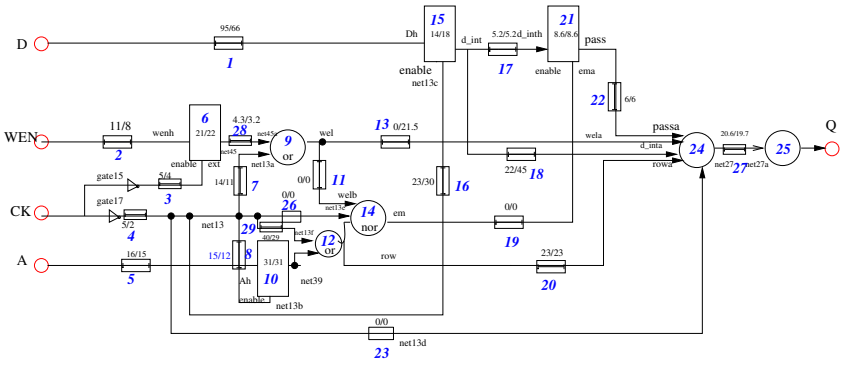


Figure 6.6: Abstract model of the SPSMALL memory

A further major difference with the manual model described in the previous section is that the PTAs are here fully automatically generated. Recall that, in the previous section, the PTAs were written in a partially manual way, and the

model was then simplified by grouping together several automata. Here, we first manually wrote the VHDL code corresponding to the different elements of the memory (which is much quicker and less error-prone than describing the PTAs), and then automatically synthesized the PTAs using the tool VHDL2TA [12]. This leads to more parameters, including a slightly richer environment, involving explicitly signal A, characterized by its setup value, viz., t^A_{setup}. This technique results in a model containing 28 automata, 28 clocks, 32 discrete variables and 62 parameters.

We give below the reference valuation π'_1 of the full set of parameter corresponding to this model:

$$
\begin{array}{ll}
 & delta0\_reg\_mux\_output = 0 \quad delta1\_reg\_mux\_output = 0 \\
\wedge & delta0\_passa = 6 \quad delta1\_passa = 6 \\
\wedge & delta0\_reg\_mem\_point = 9 \quad delta1\_reg\_mem\_point = 9 \\
\wedge & delta0\_rowa = 23 \quad delta1\_rowa = 23 \\
\wedge & delta0\_d\_int\_h = 6 \quad delta1\_d\_int\_h = 6 \\
\wedge & delta0\_d\_inta = 45 \quad delta1\_d\_inta = 22 \\
\wedge & delta0\_ema = 0 \quad delta1\_ema = 0 \\
\wedge & delta0\_row = 0 \quad delta1\_row = 0 \\
\wedge & delta0\_welb = 0 \quad delta1\_welb = 0 \\
\wedge & delta0\_wela = 22 \quad delta1\_wela = 0 \\
\wedge & delta0\_net45a = 4 \quad delta1\_net45a = 5 \\
\wedge & delta0\_net13f = 31 \quad delta1\_net13f = 45 \\
\wedge & delta0\_net13e = 2 \quad delta1\_net13e = 5 \\
\wedge & delta0\_net13d = 2 \quad delta1\_net13d = 5 \\
\wedge & delta0\_net13a = 13 \quad delta1\_net13a = 19 \\
\wedge & delta0\_reg\_latch\_a = 31 \quad delta1\_reg\_latch\_a = 31 \\
\wedge & delta0\_reg\_latchwen = 22 \quad delta1\_reg\_latchwen = 21 \\
\wedge & delta0\_reg\_latchd = 18 \quad delta1\_reg\_latchd = 14 \\
\wedge & delta0\_en\_latchd = 32 \quad delta1\_en\_latchd = 28 \\
\wedge & delta0\_en\_latcha = 14 \quad delta1\_en\_latcha = 20 \\
\wedge & delta0\_en\_latchwen = 4 \quad delta1\_en\_latchwen = 5 \\
\wedge & delta0\_a\_h = 15 \quad delta1\_a\_h = 16 \\
\wedge & delta0\_wen\_h = 8 \quad delta1\_wen\_h = 11 \\
\wedge & delta0\_d\_h = 66 \quad delta1\_d\_h = 95 \\
\wedge & delta0\_q\_0 = 20 \quad delta1\_q\_0 = 21 \\
\wedge & T_{HI} = 45 \quad T_{LO} = 65 \\
\wedge & t^D_{setup} = 108 \quad t^A_{setup} = 58 \\
\wedge & t^{WEN}_{setup} = 48
\end{array}
$$

We now give below the set of parameter valuations corresponding to the three input timings we are interested in optimize (timings are given in tens of pico-seconds):

$$t^D_{setup} = 108 \qquad t^{WEN}_{setup} = 48 \qquad t^A_{setup} = 58$$

Applying IMITATOR II to this model and this reference valuation π'_1, one synthesizes the following constraint:

$$
\begin{array}{ll}
 & T_{HI} + delta1\_net13a > delta1\_reg\_mux\_output + delta0\_wela + delta0\_net13a + delta1\_q\_0 \\
\wedge & T_{HI} > delta1\_reg\_mux\_output + delta0\_wela + delta0\_net13a \\
\wedge & delta0\_en\_latchd > delta1\_ema + delta0\_row + delta0\_net13f
\end{array}
$$

$$
\begin{array}{ll}
\wedge & delta1\_reg\_latchwen + delta1\_en\_latchd > delta1\_row + delta1\_net13f \\
\wedge & t^{WEN}_{setup} + delta1\_d\_h > t^{D}_{setup} + delta0\_net45a + delta0\_reg\_latchwen + delta0\_wen\_h \\
\wedge & delta1\_d\_int\_h + delta1\_reg\_latchd + delta1\_d\_h > t^{D}_{setup} + delta0\_en\_latchwen \\
\wedge & delta1\_d\_inta + delta1\_reg\_latchd + delta1\_d\_h > t^{D}_{setup} + delta0\_en\_latcha \\
\wedge & delta0\_en\_latcha > delta0\_welb + delta0\_net13a \\
\wedge & delta1\_en\_latcha > delta1\_welb + delta1\_net13a \\
\wedge & T_{LO} > delta1\_net45a + delta1\_reg\_latchwen + delta1\_en\_latchd \\
\wedge & delta1\_reg\_latchd + delta1\_d\_h > t^{D}_{setup} \\
\wedge & delta1\_net13f > delta1\_en\_latchd \\
\wedge & t^{WEN}_{setup} + delta1\_net13a > T_{LO} \\
\wedge & T_{LO} + delta0\_en\_latchwen > delta1\_rowa + delta1\_net13f \\
\wedge & delta1\_net13e \geq 0 \\
\wedge & t^{WEN}_{setup} + delta1\_en\_latchd > T_{LO} + delta0\_wen\_h \\
\wedge & delta1\_net45a \geq 0 \\
\wedge & delta0\_net45a \geq 0 \\
\wedge & delta1\_welb \geq 0 \\
\wedge & delta0\_wela + delta0\_net13a > delta0\_en\_latchd \\
\wedge & delta0\_welb \geq 0 \\
\wedge & delta1\_row \geq 0 \\
\wedge & delta1\_reg\_mux\_output + delta0\_wela + delta0\_net13a + delta1\_q\_0 > T_{HI} + delta1\_net13e \\
\wedge & delta0\_row \geq 0 \\
\wedge & delta1\_ema \geq 0 \\
\wedge & delta1\_reg\_mux\_output \geq 0 \\
\wedge & T_{LO} + delta0\_reg\_latchwen + delta0\_wen\_h > t^{WEN}_{setup} + delta1\_row + delta1\_net13f \\
\wedge & t^{A}_{setup} + delta0\_wen\_h > t^{WEN}_{setup} + delta1\_a\_h \\
\wedge & t^{A}_{setup} > delta1\_reg\_latch\_a + delta1\_a\_h \\
\wedge & t^{A}_{setup} + delta0\_rowa + delta0\_net13f > T_{HI} + T_{LO} \\
\wedge & T_{LO} + delta1\_a\_h > t^{A}_{setup} + delta1\_en\_latcha \\
\wedge & t^{D}_{setup} + delta1\_reg\_latch\_a + delta1\_a\_h > t^{A}_{setup} + delta1\_d\_h \\
\wedge & t^{D}_{setup} + delta0\_net13a > delta1\_d\_int\_h + delta1\_reg\_latchd + delta1\_d\_h \\
\wedge & t^{D}_{setup} + delta0\_net13f > delta1\_d\_inta + delta1\_reg\_latchd + delta1\_d\_h \\
\wedge & T_{LO} > t^{A}_{setup} + delta1\_net13e \\
\wedge & t^{D}_{setup} + delta1\_rowa + delta1\_net13f > T_{HI} + 2 * T_{LO} \\
\wedge & T_{HI} + T_{LO} > t^{WEN}_{setup} + delta0\_rowa + delta0\_net13f \\
\wedge & T_{HI} + T_{LO} > delta1\_reg\_latchd + delta1\_d\_h \\
\wedge & delta0\_ema = 0 \\
\wedge & delta1\_welb = delta1\_wela \\
\wedge & T_{HI} + T_{LO} = t^{D}_{setup} + delta0\_net13e \\
\wedge & T_{HI} + T_{LO} = t^{D}_{setup} + delta0\_net13d \\
\wedge & delta1\_net13e = delta1\_net13d \\
\wedge & delta1\_net13e = delta1\_en\_latchwen \\
\wedge & T_{LO} + delta1\_wen\_h = t^{WEN}_{setup} + delta1\_en\_latchd
\end{array}
$$

We project below this constraint onto t^{A}_{setup}, t^{D}_{setup} and t^{WEN}_{setup}.

$$t^{D}_{setup} = 108 \quad \wedge \quad t^{WEN}_{setup} = 48 \quad \wedge \quad 56 < t^{A}_{setup} < 60$$

This constraint is an "interesting" (though unfortunate) example of constraint for which the output parameter domain is (almost) reduced to a single point. Thus, it is not possible to optimize values of t^{D}_{setup} and t^{WEN}_{setup} according to this constraint. Nevertheless, the cartography algorithm introduced in [6] will allow us to overcome this shortcoming, and synthesize a dense set of parameters allowing us to minimize those input timing parameters (see Section 6.4).

6.3.3 Larger Models

Two other versions of the SPSMALL memory have been considered. The first one is actually the full SPSMALL memory with 1 memory point of 2 bits. The model described as a network of PTAs has been automatically generated from the VHDL code using VHDL2TA. The VHDL code itself was also automatically generated from the transistor netlist given by ST-Microelectronics. This

chain of analysis has been performed in the framework of the VALMEM project (see Figure 6.1 page 28). Unfortunately, because of the high size of this model (101 automata, 101 clocks, 200 parameters, 130 discrete variables, which result in more than 6000 lines of code described in the IMITATOR II syntax), IMITATOR II does not succeed to synthesize a constraint after several hours.

The second version corresponds to a larger version of the SPSMALL memory, with 3 memory points of 2 bits. Due to the even larger size of this model (more than 130 automata), IMITATOR II does not succeed to synthesize a constraint either.

Improving IMITATOR II so that it can synthesize constraints for such large systems is the subject of future work. It is also interesting to note that non-parametric analyses of these two models have been successfully performed using the UPPAAL model checker [31], allowing to verify several properties.

6.4 Behavioral Cartography

We will consider here two versions of the memory: the manually abstracted model (described and analyzed using the inverse method in Section 6.3.1), and the automatically generated model (described and analyzed in Section 6.3.2).

6.4.1 Manually Abstracted Model

We first consider here the model manually abstracted, described in Section 6.3.1. We are interested in minimizing the values of the setup timing parameters, viz., t^D_{setup} and t^{WEN}_{setup}, so that they still verify the following good property mentioned in [15]: "the response time of the memory must be smaller than 56" (recall that units are given in tens of *ps*). This response time corresponds to the value $T_{CK \to Q}$ depicted in Figure 6.3 page 30, and represents the time between the second rise of input signal CK and the rise of the output signal Q. Note that this good property does not strictly speaking correspond to a property on traces. As a consequence, we make use of an *observer* (as in [10] and [15]), i.e., an additional PTA which waits for the rise of Q and, depending on the time of this action, goes into a good location or into a bad location. Locations are observable within traces, thus this property is now a property on traces.

We perform a behavioral cartography of the SPSMALL memory, for the following V_0:

$$t^D_{setup} \in [65; 110] \quad \wedge \quad t^{WEN}_{setup} \in [0; 66].$$

The other parameters are instantiated like in π_1. We give in Figure 6.7 the cartography of the SPSMALL memory, as automatically output by IMITATOR II. The dashed rectangle corresponds to V_0. The red zone above t^{WEN}_{setup} is infinite, and corresponds to a bad behavior.

Recall that each different colored zone corresponds to a different behavior[3]. Note that the cartography actually contains a few holes, i.e., zones (depicted in white) covered by no tile. We manually "filled" those zones by calling again the inverse method on one point in each zone, which allowed us to cover the whole rectangle V_0.

[3]Recall that this cartography has been automatically output by IMITATOR II which can only represent a few colors (due to the use of an external plot tool). As a consequence, different zones depicted using the same color do not necessarily have the same trace set.

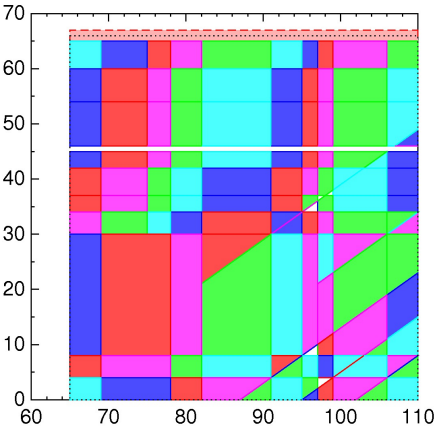


Figure 6.7: Cartography of the SPSMALL memory

We then partition the tiles into good and bad. This partition is depicted under a graphical form in Figure 6.8, where the light red (resp. dark blue) zone corresponds to the bad (resp. good) values of the parameters. After partitioning the tiles into good and bad, one is able to infer the following constraint corresponding to the set of parameters for which the memory circuit behaves well:

$$99 < t^{D}_{setup} \leq 110 \quad \wedge \quad 30 < t^{WEN}_{setup} \leq 65$$

This constraint corresponds to the maximal constraint solving the good parameters problem for the SPSMALL memory within V_0, because the whole rectangle has been covered by the tiles.

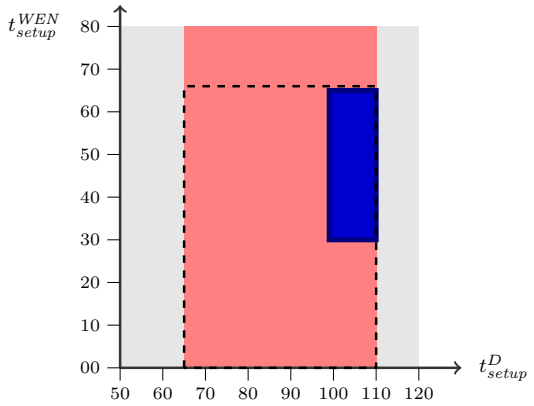


Figure 6.8: Cartography of the SPSMALL memory (after partition)

Due to the way we modeled the system (in particular the environment), values such that $t^{D}_{setup} < 65$ or $t^{D}_{setup} > 110$ do not correspond to any proper behavior. As a consequence, the constraint synthesized corresponds to the maximal constraint for the whole parameter space of this model.

One can thus minimize t^D_{setup} and t^{WEN}_{setup} according to the cartography as follows:

$$t^D_{setup} = 100 \quad \wedge \quad t^{WEN}_{setup} = 31$$

By comparison with the original datasheet π_1 (viz., $t^D_{setup} = 108$ and $t^{WEN}_{setup} = 48$), this results in a decreasing of the setup timing of signal D of 7.4 %, and a decreasing of the setup timing of signal *WEN* of 35.4 %.

Comparison with Other Methods. In [15], the authors synthesize a minimum for these setup timings, by iteratively decreasing the setup timings until the system does not behave well anymore, i.e., until the response time is not guaranteed anymore. When compared to our approach, the approach of [15] has the following limitation: they test only the *integer* points, and do not have any guarantee for the dense set of parameters between two integer points. In [15], a minimum value of 95 is given for t^D_{setup}. However, our approach indicates that the value of 95 corresponds to a bad behavior, and therefore shows a discrepancy between our respective models. A minimum value of 29 is given for t^{WEN}_{setup}, which is slightly smaller as ours. Again, this indicates a discrepancy between our respective models.

6.4.2 Automatically Generated Model

We now consider the model automatically generated, described in Section 6.3.2. As in the previous section, we are interested in minimizing the values of the setup timing parameters, viz., t^D_{setup} and t^{WEN}_{setup}, so that they still verify the following good property mentioned in [15]: "the response time of the memory must be smaller than 56" (recall that units are given in dozens of *ps*). Again, we make use of an *observer* in order to transform this property into a property on traces.

We perform a behavioral cartography of the SPSMALL memory, for the following V_0:

$$t^D_{setup} \in [89; 98] \quad \wedge \quad t^{WEN}_{setup} \in [25; 34].$$

Due to the complexity of this model, note that the rectangle V_0 is not as large as for the manual model. We give in Figure 6.9 the cartography of the SPSMALL memory, as automatically output by IMITATOR II. The dashed rectangle corresponds to V_0.

Recall that each different colored zone corresponds to a different behavior. This cartography, though interesting, contains many holes, i.e., zones (depicted in white) covered by no tile.

We then chose to launch again the analysis using a tighter grid, viz., by calling the inverse method on points multiple of 1/3 instead of integer points. This corresponds to the algorithm BC' sketched in [6]. The reason for the choice of 1/3 is that, with such a step, one is sure to cover any tile delimited by integer points. This is not the case of a step of 1 (or even 1/2), because tiles delimited by integer points may *exclude* those integer points in the case of strict inequalities.

This second cartography of the SPSMALL, with step 1/3, is given in Figure 6.10. This cartography is this time successful in the sense that the whole bounded parameter domain V_0 is covered by the tiles. Furthermore, a significant part of the parametric space outside V_0 is also covered.

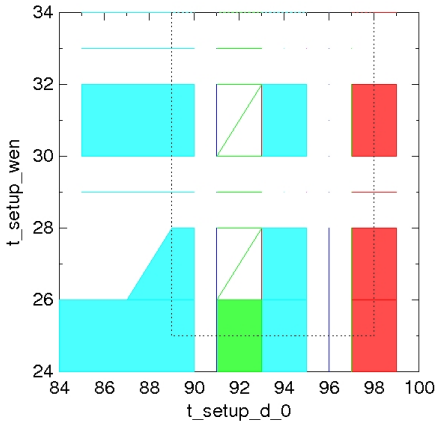


Figure 6.9: Cartography of the SPSMALL memory (generated model)

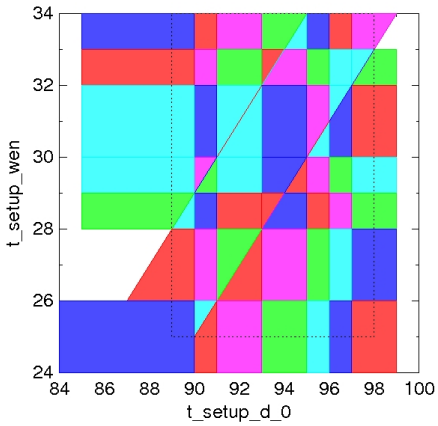


Figure 6.10: Cartography of the SPSMALL memory (full coverage)

We then partition the tiles into good and bad. This partition is depicted under a graphical form in Figure 6.11, where the light red (resp. dark blue) zone corresponds to the bad (resp. good) values of the parameters. From this partition, one is able to infer the following constraint corresponding to the set of parameters within V_0 for which the memory circuit behaves well:

$$96 \leq t^D_{setup} \leq 98 \quad \wedge \quad 29 \leq t^{WEN}_{setup} \leq 34$$

This constraint corresponds to the maximal constraint solving the good parameters problem for the SPSMALL memory within V_0, because the whole rectangle has been covered by the tiles. Also note that the cartography gives further information outside V_0.

One can thus minimize t^D_{setup} and t^{WEN}_{setup} according to the cartography as follows:

$$t^D_{setup} = 96 \quad \wedge \quad t^{WEN}_{setup} = 29$$

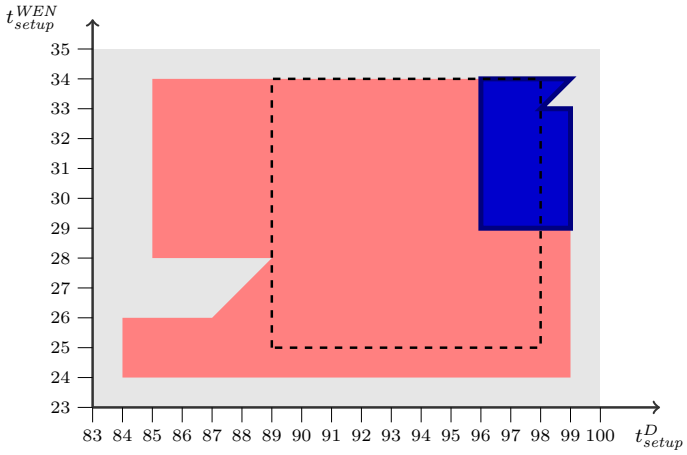


Figure 6.11: Cartography of the generated model of the SPSMALL memory (after partition)

By comparison with the original valuation for t^D_{setup} and t^{WEN}_{setup} (viz., $t^D_{setup} = 108$ and $t^{WEN}_{setup} = 48$), this results in a decreasing of the setup timing of signal D of 11.1 %, and a decreasing of the setup timing of signal WEN of 39.6 %. Such an important decreasing of some of the values of the environment show the interest of the cartography algorithm for the optimization of timing parameters.

Comparison with Other Methods. Recall that, in [15], the authors also synthesize a minimum for these setup timings, by iteratively decreasing the setup timings until the system does not behave well anymore. In [15], a minimum value of 95 is given for t^D_{setup}. However, our approach indicates that the value of 95 corresponds to a bad behavior, and therefore shows a slight discrepancy between our respective models. Also observe that the authors of [15] find a minimum value of 29 for t^{WEN}_{setup}, which is exactly the same as ours. This shows the interest of our method, which computes a constraint allowing to retrieve fully automatically the (manually computed) results from [15], with the advantages that we considered the full model of the memory (not only the write operation), that we give relations between the parameters (under the form of a constraint), and above all that we now give conditions of correctness on the *dense* space of parameters.

Due to the high size of this model (viz., an NPTA composed of 28 PTA containing 28 clocks, 32 discrete variables and 62 parameters) and to the practical interest of the constraint output, this case study can be considered as an extremely interesting application of IMITATOR II.

Remark 6.1 *In [15], values corresponding to simulation are given. Simulation is a technique based on an exact virtual version of the memory. It is usually extremely costly to perform (and is suitable for only one environment) but its results can be considered as exact for this particular case. For this case study, a simulation has been performed using the entire system (i.e., without cutting away some parts of the memory), for some (punctual) values of the*

input timings. For this environment and those values of the parameters, according to [15], the minimum possible value computed by simulation for t_{setup}^{WEN} is 36, and the minimum possible value for t_{setup}^{D} is 95. For t_{setup}^{D}, this means that the value we compute is suitable, because it is greater than the minimum possible value. Moreover, it is almost the optimal value, since our method allows to minimize t_{setup}^{D} to 96, whereas the minimum value is 95. For t_{setup}^{WEN}, however, our value is strictly smaller than the value computed using the simulation, which represents a minimum. This indicates that (at least) one delay assigned to a gate of our model (which has been automatically computed in the framework of the VALMEM project) is too approximative. Note that this limitation is of course not due to the methods developed here, but to the way the PTAs and the reference valuation were automatically generated, which is completely beyond the scope of this work.

Chapter 7

CSMA/CD Protocol

7.1 Description

Consider the CSMA/CD protocol, as studied in the context of probabilistic timed automata in [30]. We consider the case when there are two stations 1 and 2 trying to send data at the same time. The overall model is given by the parallel composition of three probabilistic timed automata representing the medium and two stations trying to send data.

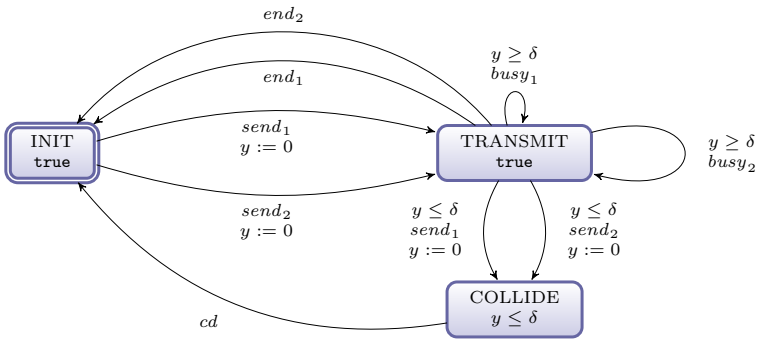


Figure 7.1: CSMA/CD Medium

The probabilistic timed automaton representing the medium is given in Figure 7.1. The medium is initially ready to accept data from any station (event $send_1$ or $send_2$). Once a station, say 1, starts sending its data there is an interval of time (at most δ), representing the time it takes for a signal to propagate between the stations. In this interval the medium can accept data from station 2 (resulting in a collision). After this interval, if station 2 tries to send data it will get the busy signal ($busy_2$). When a collision occurs, there is a delay (again at most δ) before the stations realize there has been a collision, after which the medium will become free (event cd). If the stations do not collide, then when station 1 finishes sending its data (event end_1) the medium becomes idle.

The probabilistic timed automaton representing a station i ($i = 1, 2$) is given in Figure 7.2. Station i starts by sending its data. If there is no collision, then,

after λ time units, the station finishes sending its data (event end_i). On the other hand, if there is a collision (event cd), the station attempts to retransmit the packet, where the scheduling of the retransmission is determined by a truncated binary exponential backoff process. The delay before retransmitting is an integer number of time slots (each of length $slot$). The number of slots that station i waits after the nth transmission failure is chosen as a uniformly distributed *random* integer in the range: $0, 1, 2, \ldots, 2^{bc_i+1} - 1$, where $bc_i = \min(n, bcmax)$, and $bcmax$ is a fixed upper bound for bc_i (initially: $bc_i = n = 0$). This random choice is depicted in Figure 7.2 by the assignment $backoff_i := \mathbf{RAND}(bc_i) * slot$. Once this time has elapsed, if the medium appears free the station resends the data (event $send_i$), while if the medium is sensed busy (event $busy_i$) the station repeats this process.

We consider in the following that $bcmax$ is a constant equal to 1, and that δ, λ and $slot$ are *parameters*. The reference valuation for these parameters, taken from the IEEE standard 802.3 for 10 Mbps Ethernet, is: $\delta = 26\mu\, s$, $\lambda = 808\mu\, s$, and $slot = 2\delta = 52\mu\, s$.

The method for inferring a constraint K_0 on the parameters, which is satisfied by the reference valuation and in which the behavior of the model remains the same, consists in transforming the system into a non-probabilistic parametric timed automaton. We replace the random choice $backoff_i := \mathbf{RAND}(bc_i) * slot$ with a *non-deterministic* choice, i.e., a set of 2^{bc_i+1} transitions associated with assignments of the form $backoff_i := j * slot$, for $j = 0, 1, 2, \ldots, 2^{bc_i+1} - 1$.

In the case where $bcmax = 1$, the application of the inverse method to the non-probabilistic parametric timed automaton infers for K_0 the following constraint: $(0 < \delta < slot) \wedge (15slot < \lambda < 16slot)$.

In particular, the minimum and maximum probabilities for a message sent by a station to be transmitted (i.e., to reach the location DONE) after having collided exactly k times with another message (action cd) are the same under the reference valuation and another parameter valuation satisfying K_0. This has two practical implications. Firstly, in order to compute the aforementioned minimum and maximum probabilities for $\delta = 26, \lambda = 808, slot = 52$, it suffices to compute the minimum and maximum probabilities for $\delta = 1, \lambda = 31, slot = 2$ (because both valuations satisfy the constraint $(0 < \delta < slot) \wedge (15slot < \lambda < 16slot)$ synthesized by the inverse method). Note that the valuation $\delta = 26, \lambda = 808, slot = 52$ results in a model with 5240 states using the discrete-time semantics construction, whereas the valuation $\delta = 1, \lambda = 31, slot = 2$ results in a model with 282 states. The second practical implication concerns the case in which the system designer wishes to understand the behavior of the system, in terms of minimum and maximum probabilities, for a number of parameter valuations. The approach of obtaining such information by changing manually the timing parameters and repeating model-checking analysis is potentially time consuming. Instead, the application of the inverse method shows that the minimum and maximum probabilities remain invariant for all parameter valuations satisfying the constraint K_0.

7.2 Inverse Method

We now synthesize constraints for various reference valuations.

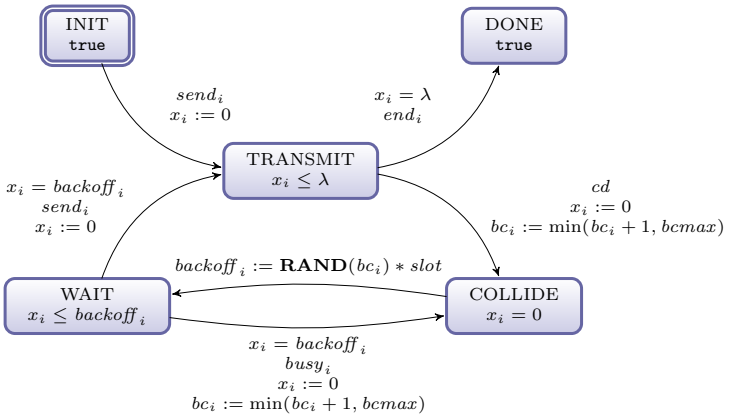


Figure 7.2: CSMA/CD Station i

7.2.1 First Valuation

We consider the following reference valuation π_0 taken from the IEEE standard 802.3 for 10 Mbps Ethernet:

$$\lambda = 808\mu\, s \qquad slot = 52\mu\, s \qquad \delta = 26\mu\, s.$$

The following constraint is synthesized after 19 iterations in 1.01 seconds (219 reachable states with 342 transitions):

$$16 * slot > \lambda \quad \wedge \quad slot > \sigma \quad \wedge \quad \sigma > 0 \quad \wedge \quad \lambda > 15 * slot$$

7.2.2 Second Valuation

We consider the following valuation (Prism):

$$\lambda = 96 \qquad \sigma = 3 \qquad slot = 6$$

The following constraint is synthesized after 21 iterations in 1.17 seconds (247 reachable states with 384 transitions):

$$16 * slot = \lambda \quad \wedge \quad \sigma > 0 \quad \wedge \quad \lambda > 16 * \sigma$$

Chapter 8

Root Contention Protocol

8.1 Presentation

This case study concerns the Root Contention Protocol of the IEEE 1394 ("FireWire") High Performance Serial Bus, considered in the parametric framework in [19, 26], and in the probabilistic framework in [29]. As described in [26], this protocol is part of a leader election protocol in the physical layer of the IEEE 1394 standard, which is used to break symmetry between two nodes contending to be the root of a tree, spanned in the network technology. The protocol consists in first drawing a random number (0 or 1), then waiting for some time according to the result drawn, followed by the sending of a message to the contending neighbor. This is repeated by both nodes until one of them receives a message before sending one, at which point the root is appointed.

We consider the following five timing parameters:

- f_min (resp. f_max) gives the lower (resp. upper) bound to the waiting time of a node that has drawn 1;
- s_min (resp. s_max) gives the lower (resp. upper) bound to the waiting time of a node that has drawn 0;
- *delay* indicates the maximum delay of signals sent between the two contending nodes.

Those timing parameters are bound by the following implicit constraint:

$$f\_min \leq f\_max \quad \wedge \quad s\_min \leq s\_max$$

The model we consider is a nonprobabilistic version of the probabilistic given in [29, 33], where the probabilistic distributions have been replaced with nondeterminism. We give in Figure 8.1 the PTA model of node i, and in Figure 8.2 the PTA model of wire i. We make use in Figure 8.1 of the notion of *urgent* locations: the semantics is that the time cannot pass inside these locations, and one must take a transition immediately after entering it. This is only syntactic sugar which is equivalent to the use of one more clock that is reset when entering the location, and that must be equal to 0 when leaving the location through any transition. Moreover, in both Figure 8.1 and Figure 8.2, we exceptionally integrate the invariant *in* the location, for the sake of readability. As usual, a location without any invariant is considered to have an invariant equal to `true`.

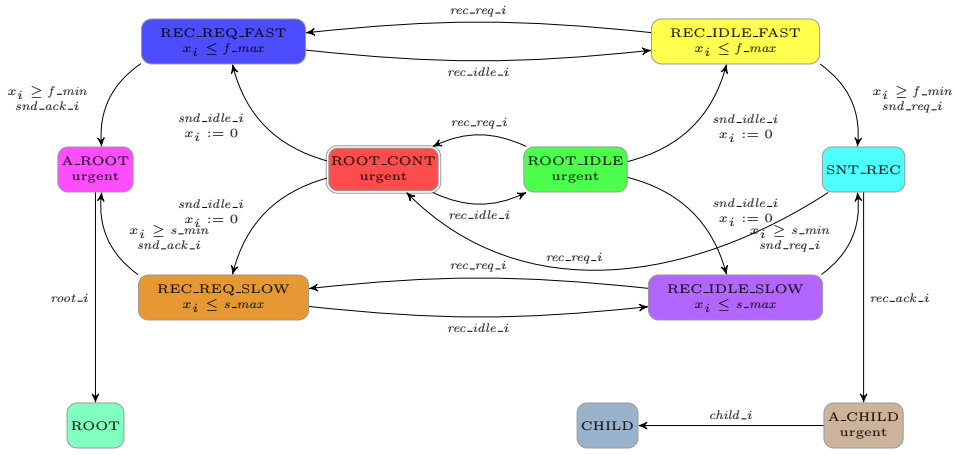


Figure 8.1: PTA modeling node i in the Root Contention Protocol

8.2 Inverse Method

We synthesize here constraints for various reference valuations.

IEEE Reference Valuation. We first aim at synthesizing a constraint for the following reference valuation π_0, which corresponds to the IEEE standard with wire length near to the maximum possible according to [26] (timings are given in ns):

$$f\_min = 760 \qquad f\_max = 850 \qquad delay = 360$$
$$s\_min = 1590 \qquad s\_max = 1670$$

By applying IMITATOR II to this model and the reference valuation π_0, we synthesize the following constraint K_0:

$$s\_min > 2 * delay + f\_max \wedge delay \geq 0 \wedge f\_min > 2 * delay$$

Observe that this constraint is exactly the same as the one synthesized in [26]. This constraint is also very similar to the one synthesized in [19]; the only difference is that our constraint is larger, because we do not constraint $delay$ to be strictly positive.

We give in Figure 8.3 the trace set of the protocol under any $\pi \models K_0$, as automatically output by IMITATOR II.

The main interest brought by the synthesis of this constraint is that it gives a criterion of robustness to the system. Similarly, it shows that this protocol is also correct for values of the parameters other than the ones given by the IEEE reference.

Further Valuations. We also get the same constraint for the following reference valuation:

$f\_min = 76$, $f\_max = 85$, $s\_min = 159$, $s\_max = 167$, and $delay = 36$.

And also for:

$f\_min = 76$, $f\_max = 85$, $s\_min = 159$, $s\_max = 167$, and $delay = 30$.

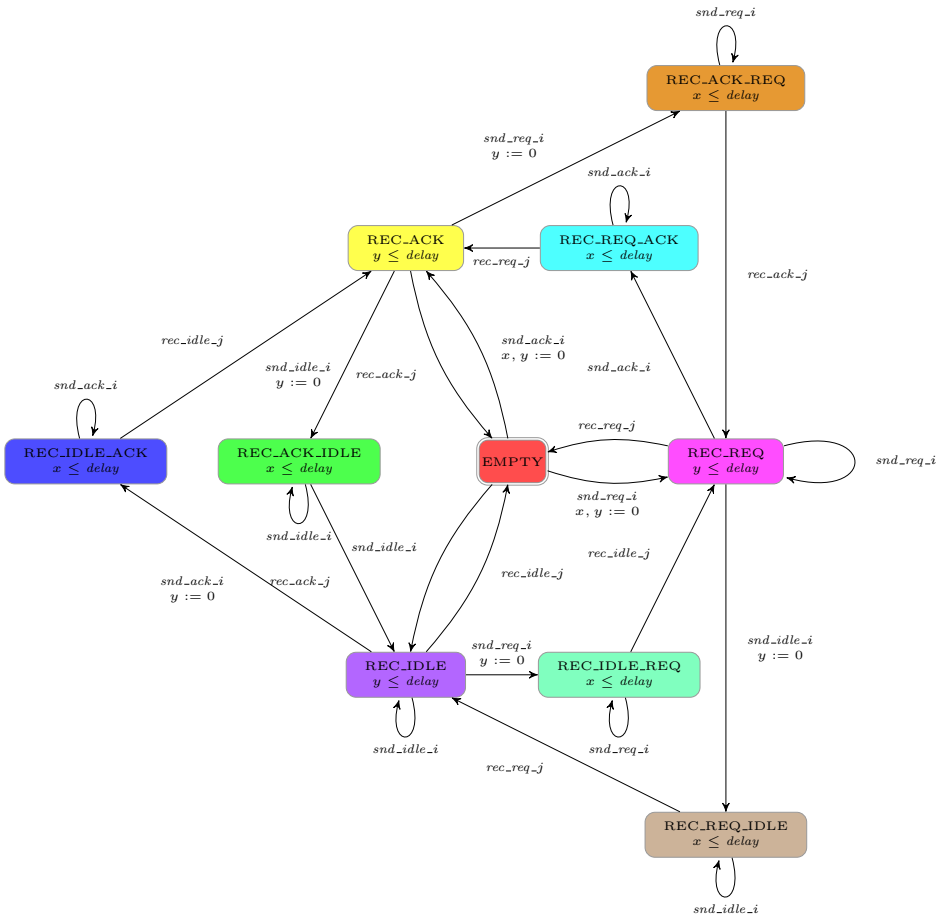


Figure 8.2: PTA modeling wire i in the Root Contention Protocol

And also for:
$f\_min = 76$, $f\_max = 85$, $s\_min = 159$, $s\_max = 167$, and $delay = 3$.

8.3 Cartography

We compute a cartography of the Root Contention Protocol using the following rectangle V_0:

$$s\_min \in [140, 200] \qquad s\_max \in [140, 200] \qquad delay \in [1, 50].$$

The two other parameters remain constant, i.e., $f\_min = 76$ and $f\_max = 85$. Note that, in order to reduce the number of points to be covered by the algorithm, we divided by 10 the reference valuation π_0 given above. This is equivalent to calling the inverse method only on the integer points which are multiples of 10 instead of on all integer points.

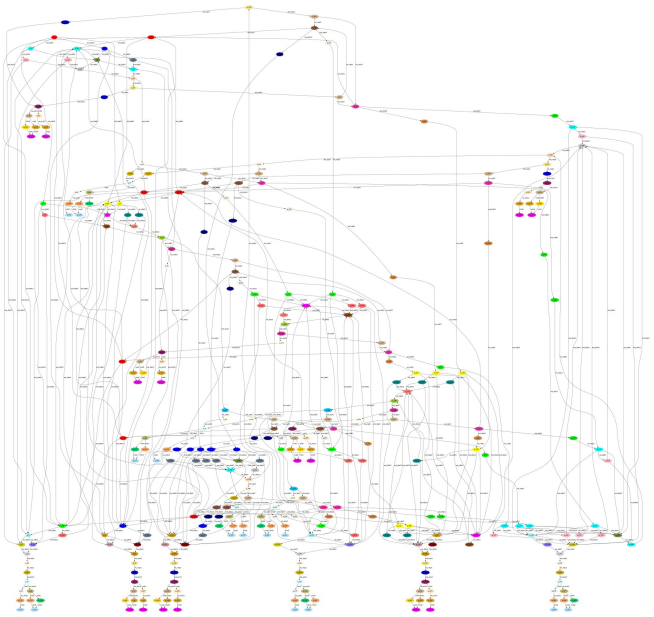


Figure 8.3: Trace set of the RCP output by IMITATOR II

Using IMITATOR II, we compute the cartography given in Figure 8.4. For the sake of clarity, we project onto *delay* and *s_min*. In each tile, the parameter *s_max* is only bound by the implicit constraint $s\_min \leq s\_max$.

It is interesting to note that the zones are delimited by two sets of lines, all centered on two different points. Those lines are depicted in Figure 8.5.

Tiles 1 and 6 are infinite towards dimension *s_min*, and all tiles are infinite towards dimension *s_max*. Moreover, although all the integer valuations within V_0 are covered (from the algorithm), the real-valued part of V_0 is not fully covered, because there are some "holes" (real-valued zones without integer valuations) in the lower right corner. An example of point which not covered by the cartography is $delay = 50$, $s\_min = 140.4$ and $s\_max = 141$.

Note that the computation of the whole set of reachable states is not possible in this example, because there exist traces of unbounded length (with incomparable constraints). Also note that it would not be possible to compute all the tiles outside V_0, because it can be shown that there exists an infinite number of tiles for this system.

8.4 Partition According to Properties

First property. We are first interested in computing the minimum probability pr_1 of satisfying the property that a leader is elected after *three* rounds or less. Using PRISM, we compute $pr_1 = 0.75$ for tile 1, $pr_1 = 0.625$ for tiles 2, 3 and 6, and $pr_1 = 0.5$ for the other tiles.

Let us suppose that a tile is good when the probability $pr_1 \geq 0.7$, and bad otherwise. In this case, the good subspace is only made of tile 1, depicted in

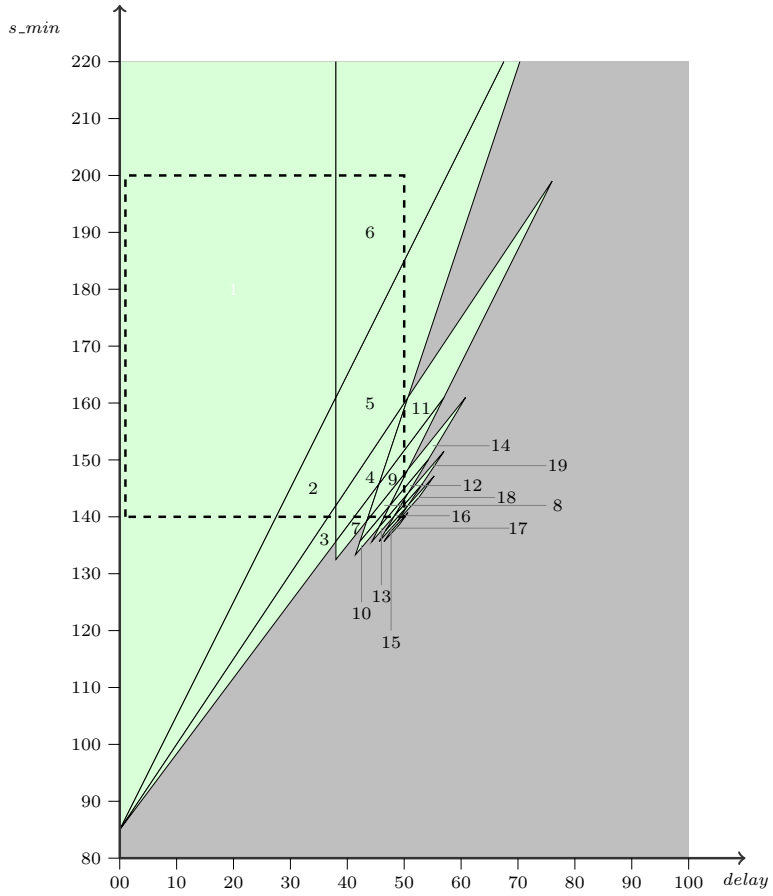


Figure 8.4: Behavioral cartography of the Root Contention Protocol according to *delay* and *s_min*

blue in Figure 8.6. Note that, in Figure 8.6, two tiles with the same color have the same probability pr_1.

Second property. We are now interested in computing the minimum probability pr_2 of satisfying the property that a leader is elected after *five* rounds or less. We do not need to compute the cartography again, but only the value of pr_2 for one valuation in each tile. We get $pr_2 = 0.936$ for tile 1, $pr_2 = 0.789$ for tiles 2 and 3, $pr_2 = 0.664$ for tile 6, and $pr_2 = 0.5$ for the other tiles.

Let us suppose that a tile is good when the probability $pr_2 \geq 0.7$, and bad otherwise. In this case, the good subspace is made of tiles 1, 2 and 3. (For the sake of concision, we do not give the new coloring of the cartography given in Figure 8.4.)

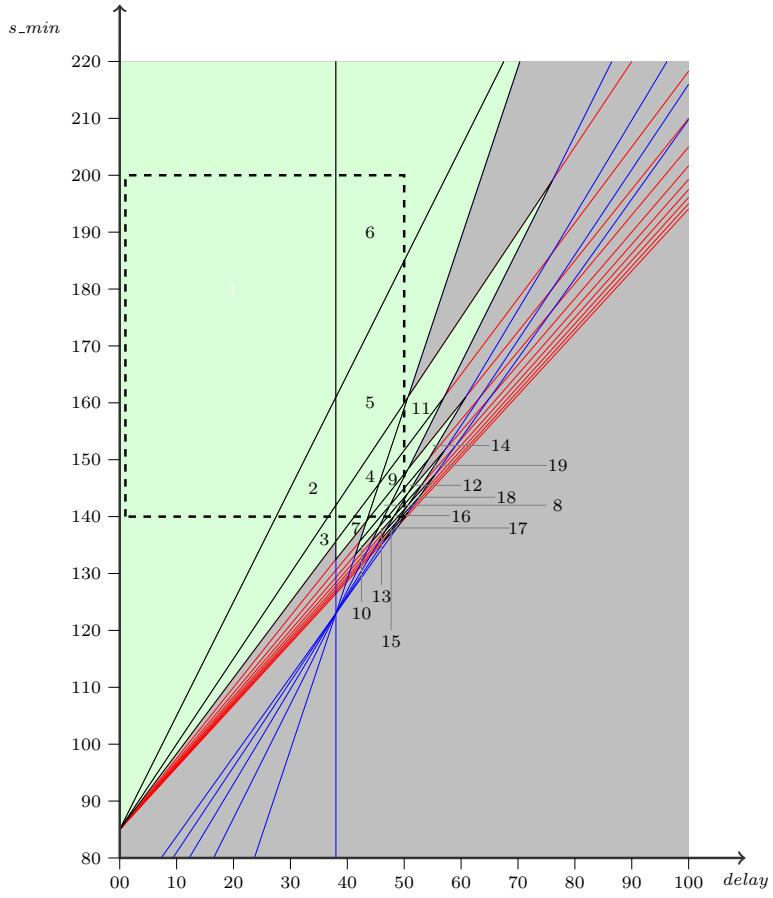


Figure 8.5: Behavioral cartography of the Root Contention Protocol according to *delay* and *s_min* (with lines)

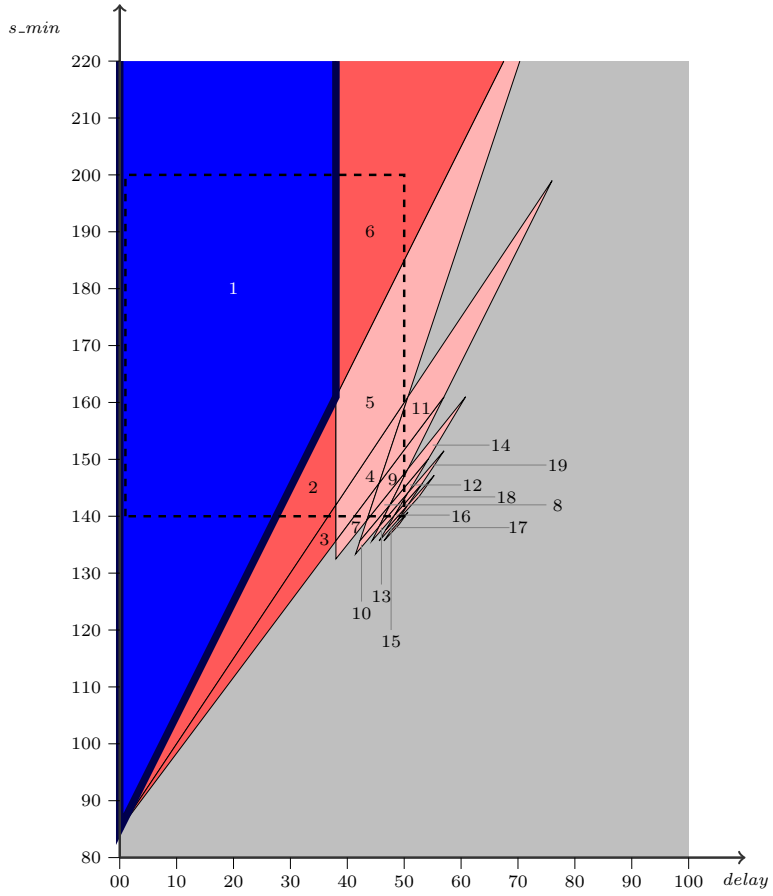


Figure 8.6: Behavioral cartography of the Root Contention Protocol according to *delay* and *s_min*, colored according to $P_{\leq 3}$

Chapter 9

Bounded Retransmission Protocol

9.1 Description

We study here the Bounded Retransmission Protocol described and modeled using timed automata in [20]. As said in [20], this protocol, used in one of Philips' products, is based on the well-known alternating bit protocol but is restricted to a bounded number of retransmissions of a chunk, i.e., part of a file. So, eventual delivery is not guaranteed and the protocol may abort the file transfer. Timers are involved in order to detect the loss of chunks and the abortion of transmission.

The protocol consists of a sender equipped with a timer, and a receiver equipped with another timer, which exchange data via two unreliable (lossy) channels.

The model considered here is a slightly simplified version of the model of [20]. In particular, a loop in the model of the sender has been discarded, implying the fact that the sender tries to send only one file.

As in [20], we consider the following five timing parameters for the model.

- N stands for the number of chunks of a file;
- $SYNC$ corresponds to the delay added after a failure in order to assure that the sender does not start transmitting a new file before the receiver has properly reacted to the failure;
- $T1$ corresponds to the time-out of the sender for initiating a retransmission when the sender has not received an acknowledgment from the receiver;
- TR corresponds to the time-out of the receiver for indicating failure when it has not received the last chunk of a file
- TD corresponds to the maximum delay in communication channels.

We consider the following valuation π_0 of the parameters of the system:

$$\begin{array}{ccc} MAX = 2 & N = 2 & TD = 1 \\ T1 = 3 & TR = 16 & SYNC = 17 \end{array}$$

We consider a slightly simplified model of the protocol, where the system stops when one file has been successfully sent.

9.2 Synthesis of Constraints

Using IMITATOR II applied to the PTA $\mathcal{A}$ modeling the system and the reference valuation π_0, the following constraint K_0 is computed:

$$\begin{array}{rrcrr} & N = 2 & \wedge & MAX = 2 \\ \wedge & TR + TD > 5 * T1 & \wedge & TR \geq 3 * TD + 4 * T1 \\ \wedge & 2 * TD + 5 * T1 > TR & \wedge & SYNC + TD \geq TR \\ \wedge & T1 > 2 * TD \end{array}$$

The corresponding trace set, automatically synthesized by IMITATOR II, is given in Figure 9.1. As for the Root Contention Protocol, the main interest brought by the synthesis of this constraint is that it gives a criterion of robustness to the system. Similarly, it shows that this protocol is also correct for values of the parameters other than the ones given by the reference valuation π_0.

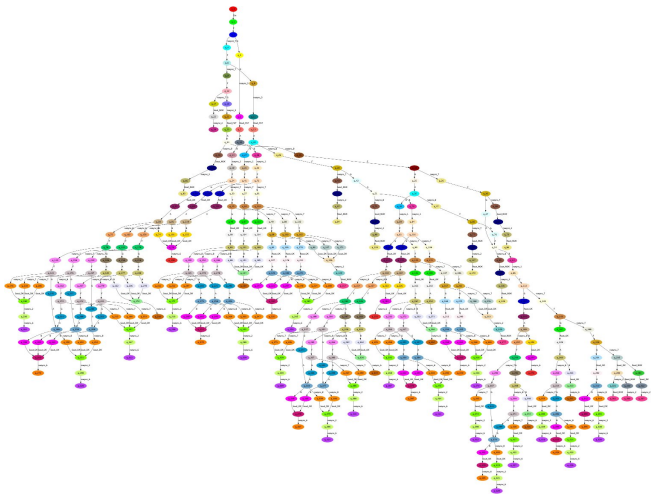


Figure 9.1: Trace set of the BRP automatically output by IMITATOR II

Comparison with other methods. In [20], the authors synthesize the following (non-linear) constraints guaranteeing that (1) premature time-outs are not possible, and (2) sender and receiver resynchronize after an abort.

$$Z: \quad T1 > 2 * TD \quad \wedge \quad SYNC \geq TR \geq 2 * MAX * T1 + 3 * TD$$

Note that, since $\pi_0 \models Z$, our constraint K_0 also guarantees that the assumptions of [20] are satisfied. It can be shown that our constraint K_0 is incomparable with this constraint Z.

The analysis of the complete model of the protocol (as described in [20]) has also been considered in [36] using the variant $IM^{\subseteq}$ implemented in IMITATOR II. The following constraint $K^{\subseteq}$ is computed using the same reference valuation π_0:

$$
\begin{array}{rrcrr}
 & N = 2 & \wedge & MAX = 2 & \\
\wedge & TR + TD \geq 5 * T1 & \wedge & TR \geq 3 * TD + 4 * T1 & \\
\wedge & 2 * TD + 5 * T1 \geq TR & \wedge & T1 \geq 2 * TD & \\
\wedge & SYNC + T1 \geq TR + TD & & &
\end{array}
$$

It can be shown that this constraint $K^{\subseteq}$ is incomparable both with our constraint K_0, and with the constraint Z of [20].

Chapter 10

IEEE 802.11 Wireless Local Area Network Protocol

10.1 Description

We also applied our inverse method algorithm to the IEEE 802.11 Wireless Local Area Network Protocol.

In order to apply the inverse method, we have to use for this chapter the first version of IMITATOR [7], and not IMITATOR II. Indeed, the new version IMITATOR II does not allow the use of *urgent actions* (or "as soon as possible" actions), i.e., actions that the system *must* take as soon as the guards of the different transitions synchronizing on this action are all verified. On the contrary, IMITATOR is based on HYTECH, which allows the use of such actions. The implementation of this feature to IMITATOR II is the subject of future work.

We consider here a model with a maximal exponential backoff of $BOFF = 1$, whereas the model given in [33, 28] considers that $BOFF = 6$. The inverse method and its implementations IMITATOR and IMITATOR II are in general not much sensitive to the size of the constants of the model. However, for this particular protocol, the waiting time before a retransmission is modeled by a nondeterministic choice between 2^{BOFF} transitions leading to 2^{BOFF} different loops. Each of those loops can be repeated up to $2^{BOFF} * ASLOTTIME$ times, thus leading to a dramatic explosion of the number of states. This, together with the fact that IMITATOR is by far less efficient than IMITATOR II (see comparison in Table 12.1 page 61), explains the much higher computation time (7 hours instead of a few seconds for the two other case studies) of K_0. For bigger values of $BOFF$, IMITATOR does not succeed to synthesize a constraint.

We also consider a maximal number of collisions of $MAXCOL = 3$.

10.2 Inverse Method

10.2.1 First Valuation

We first consider the following valuation π_0 of the parameters, which is the original (non rescaled) valuation from the IEEE standard and considered in [33,

28][1] (timings are given in $\mu\, s$):

$$\begin{array}{cccc} ACK = 205 & ASLOTTIME = 50 & BOFF = 1 & DIFS = 128 \\ MAXCOL = 3 & SIFS = 28 & TTMIN = 224 & TTMAX = 15717 \\ VULN = 48 & & & \end{array}$$

Taking a parametric timed automaton version of the model and the parameter valuation π_0 as input, the tool IMITATOR computes the following constraint K_0 after 78 iterations and 30134 seconds (790 reachable states):

$$\begin{array}{rcl} SIFS + ACK < 6ASLOTTIME & \wedge & 0 < VULN \\ \wedge \quad DIFS < 3ASLOTTIME & \wedge & 0 < SIFS \\ \wedge \quad 2ASLOTTIME < DIFS & \wedge & 0 < ACK \\ \wedge \quad VULN < ASLOTTIME & \wedge & SIFS < DIFS \\ \wedge \quad 5ASLOTTIME < VULN + DIFS + TTMIN & \wedge & TTMIN \leq TTMAX \\ \wedge \quad 6ASLOTTIME = ACKTO & & \end{array}$$

10.2.2 Second Valuation

We consider the following rescaled valuation π_1 of the parameters given in [33, 28]. These parameters were obtained by rescaling the original timings from the IEEE 802.11 standard by $50\mu\, s$; because not all timing values are multiples of $50\mu\, s$, it was then necessary to round some constants either up or down. Note also that, in [28], $BOFF = 6$ and $TTMAX = 315$.

$$\begin{array}{cccc} ACK = 4 & ASLOTTIME = 1 & BOFF = 1 & DIFS = 3 \\ MAXCOL = 3 & SIFS = 1 & TTMIN = 4 & TTMAX = 7 \\ VULN = 1 & & & \end{array}$$

Taking a parametric timed automaton version of the model and the parameter valuation π_0 as input, the tool IMITATOR computes the following constraint K_0 after 78 iterations and 67584 seconds (1572 reachable states):

$$\begin{array}{rcl} 6ASLOTTIME = ACKTO & \wedge & 0 < SIFS \\ \wedge \quad 3ASLOTTIME = DIFS & \wedge & SIFS < 3ASLOTTIME \\ \wedge \quad ASLOTTIME = VULN & \wedge & 0 < ASLOTTIME \\ \wedge \quad ASLOTTIME < TTMIN & \wedge & 0 < ACK \\ \wedge \quad SIFS + ACK < 6ASLOTTIME & \wedge & TTMIN \leq TTMAX \end{array}$$

10.2.3 Bigger Valuations

For $BOFF = 1$ and $MAXCOL = 4$, the inverse method does not terminate (out of memory).

[1] Note however that, in [28], $BOFF = 6$.

Chapter 11

A Networked Automation System

In this section, we consider a Networked Automation System studied in the framework of the SIMOP project of Institut Farman (Fédération de Recherche CNRS, FR3311). This project is a joint work between two laboratories of École Normale Supérieure de Cachan, namely LSV and LURPA. The goal of this project was to define several good behavior zones for a distributed control system, using different techniques of timed verification.

11.1 Description of the Model

We are here interested in Networked Automation Systems (NAS). NAS with Ethernet-based fieldbuses (instead of traditional fieldbuses) are more and more often used in the industry, even for critical systems such as chemical or power plants. To ensure the reliability of such systems, not only the functionalities but also the timing performances must be validated.

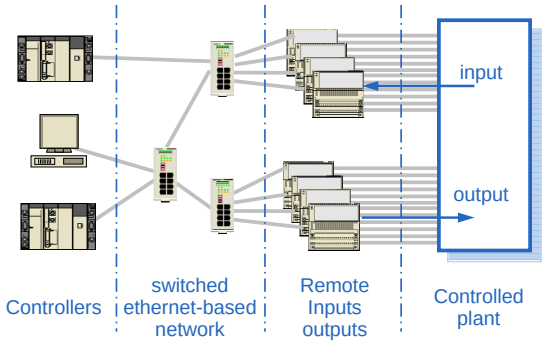


Figure 11.1: Example of Networked Automation Systems (NAS)

The main features of the physical components of these architectures is described in Figure 11.1 (see, e.g., [34, 21]):

- Programmable Logical Controllers (PLCs) are modular. Within each controller, a calculus processor runs a program cyclically, while a communication processor performs a periodic scanning of some Remote Input-Output Modules (RIOMs), termed I/O scanning. It matters to underline that the cycles of these two processors are asynchronous, data exchanges being made by means of a shared memory.
- The network includes Ethernet switches and Ethernet links and is dedicated only to communications between the PLCs and RIOMs; there is no other additional traffic.
- Inputs and outputs from/to the plant are gathered in RIOMs which are directly connected to the network. One RIOM may be shared by several PLCs.

In the following we will use a simple example of NAS, which includes an item of each component: one controller, one Ethernet switch, one RIOM and no particular behavior for the plant. Only one input signal is considered, producing a causal output signal after processing into the controller. Moreover, it will be assumed that there is no frame loss, which is a quite reasonable assumption for this kind of switched industrial Ethernet solution in the concerned operation conditions.

The full description of the model is available in [3].

In the design and the development process of NAS, engineers have to select and setup components that involves delay parameters. When setting the parameters, the engineer must preserve the expected performance of the NAS, i.e., the response time between the input signal and the output signal. This response time should remain below a maximum limit to get an assessed NAS. The assessment of a NAS is difficult because, for each valuation of the parameters and each input signal, the response time may be different.

The aim of the SIMOP project [3] is to propose an approach able to assist engineers to design, setup and/or reconfigure Networked Automation Systems, by synthesizing values for the parameters of the NAS guaranteeing a correct response time.

11.2 Definition of a Zone of Good Behavior

The system is modeled by a PTA $\mathcal{A}$ containing 7 parameters $COMct$, $COMd$, $NETd$, $PLCct$, $COMct$, $RIOd$, $SIGmrt$, corresponding to various timing delays of the system (see the description of the model in [3]). We consider the following reference valuation π_0 of the parameters

$$PLCct = 600 \quad COMct = 500 \quad SIGmrt = 2071 \quad PLCmtt = 100$$
$$RIOd = 70 \quad COMd = 25 \quad NETd = 10$$

It can be shown (e.g., using the model checker Uppaal [31] for timed automata) that the system under π_0 behaves well (this notion of good behavior corresponds here to the response time of the system being under a given value). The goal of the SIMOP project is to find *other* valuations of the parameters with a good behavior.

Using our program Imitator II applied to $\mathcal{A}$ and π_0, we infer the following constraint K_0 defining a good functioning zone.

$$
\begin{array}{ll}
 & 4 * COMct \geq NETd + COMd + 3 * PLCct + PLCmtt \\
\wedge & 4 * COMct \geq RIOd + 2 * NETd + 3 * PLCct + PLCmtt \\
\wedge & PLCct \geq COMct + PLCmtt \\
\wedge & PLCct < RIOd + NETd + COMct + COMd \\
\wedge & SIGmrt > RIOd + 4 * COMct \\
\wedge & NETd > 0 \\
\wedge & PLCmtt > NETd + COMd \\
\wedge & PLCmtt > RIOd + 2 * NETd \\
\wedge & 4 * COMct < RIOd + NETd + COMd + 3 * PLCct + PLCmtt
\end{array}
$$

Note that this constraint was synthesized using IMITATOR II with the "inclusion" mode. This mode is the implementation of the variant $IM^{\subseteq}$ of IM, which allows to terminate earlier by modifying the fixpoint of the algorithm (see [9]). This good behavior of the NAS is modeled by an observer which checks that the response time is correct, and then goes into a good or a bad state, depending on the response time. As a consequence, although this variant does not guarantee the equality of trace sets, the system under any $\pi \models K_0$ will not contain any bad state because this variant preserves the non-reachability.

By applying IMITATOR II in the standard mode, the analysis does not terminate, due to the explosion of the state space.

11.3 Comparison with Other Methods

In [3], we consider two different approaches: our inverse method, synthesizing a constraint on the parameters, and a dichotomy method, testing (using the UPPAAL model checker) the correctness of a great number of integer points. The dichotomy method synthesizes a cloud of "good" points, which is obviously much bigger than the zone defined by our constraint K_0 (see the graphical comparison in [3]). However, this discrete approach suffers from several limitations. First, only the *discrete* integer points are guaranteed to be correct, whereas our inverse method synthesizes a *dense* zone for which the behavior is guaranteed to be correct. This gives a criterion of robustness for the system, which is interesting in practice, where the real values of the timing delays may not always be exactly equal to the values specified by the designer. Second, only 3 dimensions (viz., *COMct*, *PLCct* and *SIGmrt*) have been considered in the discrete approach, whereas our constraint K_0 is given in 7 dimensions.

The final remarks of [3] suggests the idea to combine both approaches in order to synthesize a much larger dense zone in 7 dimensions: by iterating the inverse method on points synthesized by the dichotomy method, one gets a set of constraints guaranteeing a good behavior. This is actually the idea of the behavioral cartography developed in [6]. However, this idea has not been experimented in the SIMOP project for two reasons. First, the cartography algorithm did not exist at the time of the project. Second, the first version of the tool IMITATOR used in the framework of this project needed almost 7 hours to synthesize a constraint, and iterating it manually would have been highly time-consuming. It would be interesting to investigate this idea again using the new version IMITATOR II, implementing the cartography algorithm.

Chapter 12

Summary of the Experiments

Experiments were conducted on an Intel Core2 Duo 2.4 GHz with 2 Gb.

All those case studies can be found on IMITATOR II's Web page[1].

12.1 Inverse Method

The results of the application of the inverse method to various case studies are given in Table 12.1. We give from left to right the name of the example with its appropriate reference, the number of PTAs composing the global system $\mathcal{A}$, the lower and upper bounds on the number of locations per PTA, the number of clocks and parameters of $\mathcal{A}$, the number of iterations of the algorithm, the number of inequalities within K_0, the number of states and transitions, the computation time in seconds using IMITATOR, and the computation time in seconds using IMITATOR II.

Example	PTAs	loc./PTA	$\|X\|$	$\|P\|$	iter.	$\|K_0\|$	states	trans.	Time1	Time2
SR-latch	3	$[3, 8]$	3	3	5	2	4	3	0.11	0.007
Flip-flop [17]	5	$[4, 16]$	5	12	9	6	11	10	1.6	0.122
AND–OR [16]	3	$[4, 8]$	4	12	14	4	13	13	1.81	0.15
Latch	7	$[2, 5]$	8	13	12	6	18	17	14.4	0.345
CSMA/CD [30]	3	$[3, 8]$	3	3	19	2	219	342	41	1.01
RCP [29]	5	$[6, 11]$	6	5	20	2	327	518	64	2.3
SPSMALL_1 [14]	10	$[3, 8]$	10	26	32	23	31	30	4680	2.6
BRP [20]	6	$[2, 6]$	7	6	30	7	429	474	901	34
SIMOP [4]	5	$[5, 16]$	8	7	53	9	1108	1404	23455	67
SPSMALL_2 [14]	28	$[2, 11]$	28	62	94	45	129	173	-	461

Table 12.1: Summary of experiments for the inverse method

The SPSMALL_1 case study corresponds to the model manually written and described in Section 6.3.1. The SPSMALL_2 case study corresponds to the model automatically generated and described in Section 6.3.2. Both computation times

[1] `http://www.lsv.ens-cachan.fr/~andre/IMITATOR2/`

refer to the first implementation of the memory (SP1, with the reference valuation π_1). It is impossible to analyze the version automatically generated (SPSMALL_2) using the first version of IMITATOR because HYTECH runs out of memory when trying to statically compose the 28 automata in parallel.

When considering the cyclicity of the trace sets, note that, for the respective reference valuation considered, the trace sets of the following case studies are acyclic: SR-latch, flip-flop, latch, SPSMALL_1, BRP and SPSMALL_2. The other trace sets (viz., AND–OR, CSMA/CD, RCP and SIMOP) are cyclic and thus feature infinite behavior.

Note that the computation time using IMITATOR II has dramatically decreased compared to IMITATOR for all examples: the time has been divided at least by 10, and up to 2000 for the SPSMALL_1 memory. Explanations for this high improvement are the rewriting of the tool using a library of convex polyhedra instead of the call to HYTECH, the on-the-fly composition of the different PTAs, and the optimization of the algorithm described in [8].

12.2 Behavioral Cartography

The results of the application of the behavioral cartography algorithm to various case studies are given in Table 12.2. We give from left to right the name of the example with its appropriate reference, the number of PTAs composing the global system $\mathcal{A}$, the lower and upper bounds on the number of locations per PTA, the number of clocks of $\mathcal{A}$, the number of parameters varying in the cartography, the number of integer points within V_0, the number of tiles computed, the average number per tile of states and transitions of the trace set, and the computation time in seconds. Experiments were conducted on an Intel Core2 Duo 2.4 GHz with 2 Gb.

Example	PTAs	loc./PTA	$\lvert X\rvert$	$\lvert P\rvert$	$\lvert V_0\rvert$	tiles	states	trans.	Time
SR-latch	3	$[3,8]$	3	3	1331	6	5	4	0.3
Flip-flop [17]	5	$[4,16]$	5	2	644	8	15	14	3
Latch circuit	7	$[2,5]$	8	4	73062	5	21	20	96.3
AND–OR [16]	3	$[4,8]$	4	6	75600	4	64	72	118
CSMA/CD [30]	3	$[3,8]$	3	3	2000	140	349	545	269
SPSMALL_1 [14]	10	$[3,8]$	10	2	3149	259	60	61	1194
RCP [29]	5	$[6,11]$	6	3	186050	19	5688	9312	7018
SPSMALL_2 [14]	28	$[2,11]$	28	3	784	213	145	196	31641

Table 12.2: Summary of experiments for the cartography algorithm

Recall that the SPSMALL_1 case study corresponds to the model manually written, and the SPSMALL_2 case study corresponds to the model automatically generated. For the SPSMALL_2 case study, the statistics given correspond to the cartography performed using a step of 1/3 and allowing us to cover the whole parameter space within V_0: as a consequence, the number of points within V_0 correspond for this case study, not to the number of integer points, but of rational points multiple of 1/3.

For all those examples, the cartography covers 100 % of the real-valued space of V_0, except for the Root Contention Protocol, where "only" 99,99 % of V_0 is

covered. Moreover, a significant part of the real-valued space outside V_0 is also often covered.

Note that, in contrast to the inverse method (Section 12.1), no comparison with the computation time of the first version of IMITATOR is possible, because this first version did not feature the behavioral cartography algorithm.

Bibliography

[1] R. Alur and D. L. Dill. A theory of timed automata. *TCS*, 126(2):183–235, 1994.

[2] R. Alur, T. A. Henzinger, and M. Y. Vardi. Parametric real-time reasoning. In *STOC '93*, pages 592–601. ACM, 1993.

[3] S. Amari, É. André, T. Chatain, O. De Smet, B. Denis, E. Encrenaz, L. Fribourg, and S. Ruel. Timed analysis of networked automation systems combining simulation and parametric model checking. Research Report LSV-09-14, Laboratoire Spécification et Vérification, ENS Cachan, France, 2009. SIMOP Research Report. 49 pages.

[4] É. André, T. Chatain, O. De Smet, L. Fribourg, and S. Ruel. Synthèse de contraintes temporisées pour une architecture d'automatisation en réseau. In Didier Lime and Olivier H. Roux, editors, *MSR'09*, volume 43 of *Journal Européen des Systèmes Automatisés*, pages 1049–1064. Hermès, 2009.

[5] É. André, T. Chatain, E. Encrenaz, and L. Fribourg. An inverse method for parametric timed automata. *International Journal of Foundations of Computer Science*, 20(5):819–836, 2009.

[6] É. André and L. Fribourg. Behavioral cartography of timed automata. In *RP'10*, volume 6227 of *LNCS*, pages 76–90. Springer, 2010.

[7] Étienne André. IMITATOR: A tool for synthesizing constraints on timing bounds of timed automata. In Martin Leucker and Carroll Morgan, editors, *ICTAC'09*, volume 5684 of *LNCS*, pages 336–342. Springer, 2009.

[8] Étienne André. IMITATOR II: A tool for solving the good parameters problem in timed automata. In Yu-Fang Chen and Ahmed Rezine, editors, *INFINITY'10*, volume 39 of *Electronic Proceedings in Theoretical Computer Science*, pages 91–99, 2010.

[9] Étienne André. *An Inverse Method for the Synthesis of Timing Parameters in Concurrent Systems*. Ph.d. thesis, Laboratoire Spécification et Vérification, ENS Cachan, France, December 2010.

[10] M. Baclet and R. Chevallier. Timed verification of the SPSMALL memory. In *Proceedings of the 1st International Conference on Memory Technology and Design (ICMTD'05)*, pages 89–92, Giens, France, May 2005.

[11] R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 72(1–2):3–21, 2008.

[12] Abdelrezzak Bara. Vhdl2ta: A tool for automatic translation of vhdl programs plus timings into timed automata. Research report, LIP6, 2009. ANR-VALMEM Technical Report.

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

[14] R. Chevallier, E. Encrenaz, L. Fribourg, and W. Xu. Timed verification of the generic architecture of a memory circuit using parametric timed automata. *Formal Methods in System Design*, 34(1):59–81, 2009.

[15] R. Chevallier, E. Encrenaz-Tiphène, L. Fribourg, and W. Xu. Timing analysis of an embedded memory: SPSMALL. *WSEAS Transactions on Circuits and Systems*, 5(7):973–978, July 2006.

[16] R. Clarisó and J. Cortadella. Verification of concurrent systems with parametric delays using octahedra. In *ACSD '05*. IEEE Computer Society, 2005.

[17] R. Clarisó and J. Cortadella. The octahedron abstract domain. *Sci. Comput. Program.*, 64(1):115–139, 2007.

[18] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement. In *CAV '00*, pages 154–169. Springer-Verlag, 2000.

[19] A. Collomb–Annichini and M. Sighireanu. Parameterized reachability analysis of the IEEE 1394 Root Contention Protocol using TReX. In *RT-TOOLS '01*, 2001.

[20] P.R. D'Argenio, J.P. Katoen, T.C. Ruys, and G.J. Tretmans. The bounded retransmission protocol must be on time! In *TACAS '97*. Springer, 1997.

[21] B. Denis, S. Ruel, J.-M. Faure, G. Marsal, and G. Frey. Measuring the impact of vertical integration on response times in Ethernet fieldbuses. In *Proc. of ETFA'07*, 2007.

[22] G. Frehse, S.K. Jha, and B.H. Krogh. A counterexample-guided approach to parameter synthesis for linear hybrid automata. In *HSCC '08*, volume 4981 of *LNCS*, pages 187–200. Springer, 2008.

[23] D. Harris and S. Harris. *Digital Design and Computer Architecture*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[24] T. A. Henzinger, P. H. Ho, and H. Wong-Toi. Hytech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1:460–463, 1997.

[25] T. A. Henzinger and H. Wong-Toi. Using HyTech to synthesize control parameters for a steam boiler. In *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control, LNCS 1165*. Springer-Verlag, 1996.

[26] T.S. Hune, J.M.T. Romijn, M.I.A. Stoelinga, and F.W. Vaandrager. Linear parametric model checking of timed automata. *Journal of Logic and Algebraic Programming*, 2002.

[27] B. Jeannet and A. Miné. Apron: A library of numerical abstract domains for static analysis. In *CAV '09*, volume 5643 of *LNCS*, pages 661–667. Springer, 2009.

[28] M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of the IEEE 802.11 wireless local area network protocol. In *Proc. PAPM/PROBMIV'02*, volume 2399 of *LNCS*, pages 169–187. Springer, 2002.

[29] M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of deadline properties in the IEEE 1394 FireWire root contention protocol. *Formal Aspects of Computing*, 14(3):295–318, 2003.

[30] M. Kwiatkowska, G. Norman, J. Sproston, and F. Wang. Symbolic model checking for probabilistic timed automata. *Information and Computation*, 205(7):1027–1077, 2007.

[31] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1-2):134–152, 1997.

[32] O. Maler and A. Pnueli. Timing analysis of asynchronous circuits using timed automata. In *CHARME '95*, pages 189–205. Springer-Verlag, 1995.

[33] PRISM Web page. Prism web page.

[34] S. Ruel and J.-M. De Smet, O. Faure. Efficient representation for formal verification of time performances of networked automation architectures. In *Proc. of 17th IFAC World Congress*, pages 5119–5124, July 2008.

[35] P. Bazargan Sabet, P. Renault, and D. Le Dû. Prototype d'outil d'abstraction fonctionnelle, 2009. VALMEM Project deliverable 2.4.

[36] Romain Soulat. Analysis of the bounded retransmission protocol using IMITATOR II. 2010.

[37] VHDL2TA Web page. `http://www.lsv.ens-cachan.fr/~encrenaz/valmem/vhdl2hytech/`.