Compositional Timing Analysis: Power Plant Protection System Case Study *

Simon Bliudze
CEA, LIST, Embedded Real Time System Foundations Laboratory
Boîte Courrier 94, Gif-sur-Yvette, F-91191 France
Simon.Bliudze@cea.fr

ABSTRACT

We introduce a method for studying temporal behaviour of the so-called Globally Asynchronous, Locally Synchronous (GALS) systems, that is systems consisting of synchronous computing elements communicating over asynchronous channels. Our method is based on the combined use of transitional logics and timed automata. The former is used to compute, by abstract interpretation, an over-approximation of the shape of the output signal, whereas the latter provide the time-stamps for the edges. Both are applied iteratively to a hierarchical model of the system in order to avoid state space explosion. We use the IF/TCA tool-chain developed at Verimag to apply this method to a case study based on the software protection system of a P4 nuclear reactor.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]; C.3 [**SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS**]: Real-time and embedded systems

General Terms

Reliability

1. INTRODUCTION

In the context of the nuclear power plant control, certification authorities have to analyse third-party protection systems in order to validate a number of safety properties. CEA LIST is involved in the development of tools to assist such analysis. This paper presents a case study performed in view of the development of a tool for static analysis of the temporal behaviour of asynchronous circuits built from synchronous components.

Timing analysis of embedded systems is a rather broad domain. A number of methods and tools have been developed based on different techniques, such as, for example, the Real-Time Calculus for performance analysis [3] and abstract interpretation for worst- and best-case execution time analysis [5]. Timed automata [1] are used in several tools [2, 6, 11] for modelling of timed systems.

We have applied the compositional approach implemented in the TCA (Timed Circuits Analyzer) extension of the IF tool-chain [8]. In addition to the compositional approach, TCA implements an advanced forward reachability algorithm that allows to considerably reduce the exploration space, as well as several behaviour preserving abstractions reducing the number of clocks used in the model. The TCA circuit specification language is well suited for the systems such as the case study presented in this paper.

Timed automata allow us to model the system with precision sufficient to obtain exploitable results. The main obstacle for the timing analysis is the state space explosion, where the notion of "state" comprises both control locations and time-stamps (cf. Sect. 4.1). On the other hand, in the context of highly-critical systems—such as nuclear power plants—the basic safety rule of thumb is to keep things simple. This applies to the control system, but also (and by consequence) to the properties involved, which are, in fact, properties of the signal produced by the circuit, given a pulse signal as an input (cf. Sect. 2). We use transitional logics [9] in order to precompute an estimation of the logical shape of the output signal before analysing the timed automaton computed by TCA to time-stamp its edges. This allows us to deduce certain key properties, such as the minimal duration the alarm signal has to be maintained to guarantee that the appropriate reaction is triggered.

The above methodology consists in decomposing the system under analysis into a hierarchy of components, then applying the following procedure in a bottom-up manner:

1. Model each constituent component as an IF process. The IF process is either obtained by modelling the component as a TCA circuit element or generated by TCA at a previous stage of the hierarchical analysis.
2. Using TCA, generate the IF interconnection model of the composed component and insert the models of the constituent components obtained in the previous steps.
3. Using TCA, generate the reduced product timed automaton for the composed component. (The abstraction techniques used for the reduction in TCA are briefly presented in Sect. 4.2.)
4. Using transitional logics, compute the shape of the output signal and time-stamp the key edges with the data from the automaton generated in the previous step.

*This work was carried out as part of a CEA LIST–IRSN project DSR/SAMS/BASEC/CDC_GALS_2010_V1.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
WCTT '11, November 29 2011, Vienna, UNK, Austria
Copyright 2011 ACM 978-1-4503-1008-6/11/11 ...$10.00.

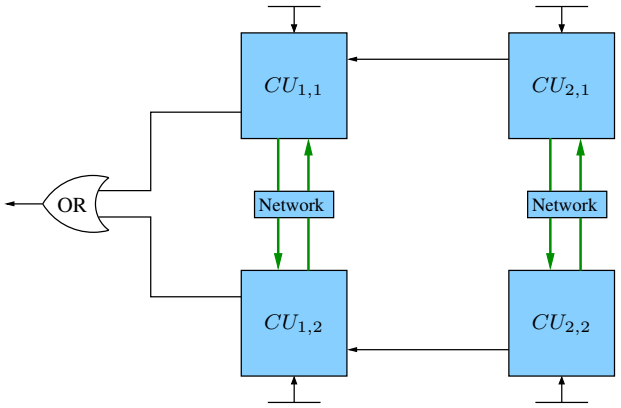


Figure 1: Architecture of the case study system

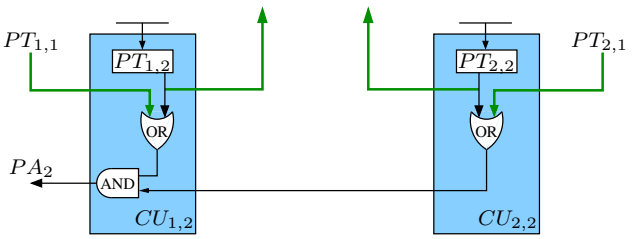


Figure 2: Architecture of the computing units

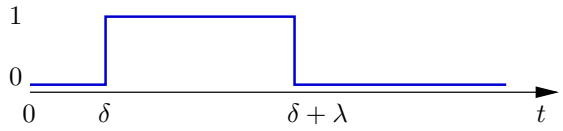


Figure 3: Pulse signal on the input of the circuit

The rest of the paper is structured as follows. In Sect. 2, we present the case study system and the properties that we wish to analyse. In Sect. 3 and Sect. 4, we briefly recall the necessary background notions. Finally, we present the analysis results in Sect. 5 and conclusions in Sect. 6.

2. CASE STUDY DESCRIPTION

The case study system considered in this paper is based on the power plant control system used in the P4 reactors [10], it implements a redundancy protection mechanism aiming to filter out false alerts in the alarm activation circuit. The system consists of two groups of two computing units ($CU_{i,j}$ with $i, j = 1, 2$) connected as shown in Fig. 1.

CUs within the same group (e.g., $CU_{i,1}$ for $i = 1, 2$) communicate through a cabled connection (black arrows) with negligible latency (less than 1 ms). CUs from different groups (e.g., $CU_{1,j}$ for $j = 1, 2$) communicate over the network (thick green arrows) with the latency bounded by a given interval $[L_m, L_M]$.

CUs receive on input boolean signals from sensors monitoring a certain physical process in the reactor. When an abnormal behaviour is detected the signal is raised to 1.

Figure 2 shows a more detailed architecture of the computing units. For each CU, the input signal is first processed by a switch (Sect. 5.1), which produces the corresponding *Partial Trigger* signals $PT_{i,j}$ (for $i, j = 1, 2$). The computing time of each switch is bounded by a given interval $[T_m, T_M]$.

In each CU, the Partial Trigger signal is disjunctively combined with the Partial Trigger produced by the corresponding CU of the other group. Subsequently, primary CUs of each group ($CU_{1,j}$ for $j = 1, 2$) compute the respective *Partial Activator* signals PA_j by taking the conjunction of the results computed in both CUs of the group. Finally, the *Activator* signal produced by the circuit is the disjunction of the two Partial Activators (see again Fig. 1).

In the general case, the input signals of the computing units (arrows with bars) correspond to different sensor units and, therefore, need not necessarily be synchronised. However, for the sake of clarity, we will assume that these input signals are synchronised. This hypothesis is not necessary for the application of the techniques in this paper, but considerably simplifies presentation.

As mentioned above, the input signals are boolean alarms produced by physical sensors. By introducing redundancy and voting, the circuit architecture reduces the number of false alarms due to sensor malfunctions. The high-level system specification consists then in triggering an emergency action whenever the Activator signal is raised. Due to the asynchronous nature of the circuit behaviour, when the alarm is raised for a short duration, it can be propagated to different computing units at different moments in time and, for example, lost when two such propagated signals are combined by conjunction. Hence, it is not *a priori* guaranteed that the Activator signal is raised even if all sensors raise their respective alarms, but for a duration that is not sufficiently long. Hence, we have to answer two essential questions:

1. What is the minimal duration, for which the alarm signals must be maintained by the sensors, that guarantees that the Activator signal is raised by the circuit?
2. Provided the alarms are maintained sufficiently long, what is the worst case response time of the circuit?[1]

More generally, we are interested in exhibiting the *time-stamped shape* of the Activator signal in response to a pulse of duration λ and phase shift δ (Fig. 3).

Below we assume the bounds on the network latency to be $L_m = 24\ ms$ and $L_M = 52\ ms$ and the bounds on the computing time of the switches, respectively, $T_{m_1} = 27\ ms$ and $T_{M_1} = 33\ ms$, for the primary CUs of each group ($CU_{1,j}$ for $j = 1, 2$), and $T_{m_2} = 72\ ms$ and $T_{M_2} = 81\ ms$, for the secondary CUs of each group ($CU_{2,j}$ for $j = 1, 2$).

3. SIGNALS

Several formal definitions of the notion *signal* can be given, depending on the particularities of the physical context and the amount of the detail that has to be captured.

Definition 1. A *timed signal* over a value domain D is piece-wise constant function $s : \mathbb{R}_+ \to D$.

The pulse signal in Fig. 3 is then defined by the function $p(t) : \mathbb{R}_+ \to \{0, 1\}$, such that

$$p(t) \quad = \quad \begin{cases} 1 & \text{for } t \in (\delta, \delta + \lambda]\,, \\ 0 & \text{otherwise.} \end{cases} \qquad (1)$$

[1]Notice that the worst case response time is necessarily greater than or equal to the minimal duration in question 1.

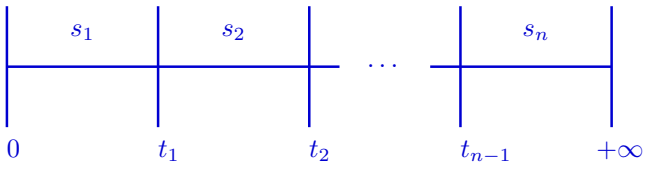


Figure 4: Graphical representation of a signal

In order to guide our analysis, we will also perform computations on abstract models of signals, only representing the sequence of values taken by the signal and omitting the temporal information.

Definition 2. An *untimed signal* over a value domain D is a sequence $s : \mathbb{N} \to D$.

An untimed pulse signal of Fig. 3 is then the sequence $0 \to 1 \to 0$.

Signals (timed or untimed) over the value domain $D = \{0, 1\}$ are called *boolean*.

In the following, we will represent signals graphically as shown in Fig. 4. This graphical representation shows the consequent values $s_1, s_2, \ldots, s_n \in D$ of the signal along with the dates $t_1, t_2, \ldots, t_{n-1}$ of the corresponding edges. For untimed signals, it is sufficient to omit the dates. This representation has an advantage over the function-style one in Fig. 3, since it allows a compact representation of signals over non-boolean domains.

For untimed boolean signals, Thompson and Mycroft [9] have proposed a number of *transitional logics* that provide further abstractions of signals. For example, one can notice that, for a boolean signal, it is sufficient to know its initial value and the total number of oscillations: the signal starting with the value 0 and making n oscillations before stabilising at 1 is denoted $\uparrow_n = 0 \to 1 \to 0 \to \cdots \to 1 \to 0 \to 1$ (where the "$1 \to 0$" pair is repeated n times). Similarly $\downarrow_n = 1 \to 0 \to 1 \cdots \to 0 \to 1 \to 0$ denotes the signal that starts at 1 and stabilises at 0 after n oscillations; F_n and T_n denote the signals that start respectively at 0 or 1 then make n oscillations before going back to the initial value.[2]

The circuits considered in [9] are constructed from four basic building blocks: (perfect—zero delay) AND-gates ($\wedge$), (perfect) NOT-gates ($\neg$), *transmission line* ($\Box$) and *inertial* ($\triangle$) delays. The difference between transmission line and inertial delays is that, in the former, the number of transitions on its input and output are equal, whereas, in the latter, a short-duration pulse may be removed entirely from the output. Corresponding operations are defined by truth tables, e.g.,

$\wedge$	F_0	F_n	T_0	T_n	$\ldots$
F_0	F_0	F_0	F_0	F_0	$\ldots$
F_m	F_0	$F_{0\ldots m+n-1}$	F_m	$F_{0\ldots m+n}$	$\ldots$
T_0	F_0	F_n	T_0	T_n	$\ldots$
T_m	F_0	$F_{0\ldots m+n}$	T_m	$T_{1\ldots m+n}$	$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Here, $F_{a\ldots b} = F_a|F_{a+1}|\ldots|F_b$, with '|' denoting the nondeterministic choice, is similar to F_n, except that it oscillates any number of times between a and b. Complete presentation of the transitional logics is out of the scope of this paper. In [9], the authors provide a hierarchy of such logics linked by Galois connections and therefore allowing static analysis by abstract interpretation [4] for the detection of glitches. Such logics can also be used as signal value domains, where the precise shape of a signal is irrelevant on a given segment of its domain (that is $\mathbb{R}_+$ for timed and $\mathbb{N}$ for untimed signals).

[2]For the sake of coherence with our sources, T_m will denote either the minimal computation duration in the case study or a value of a transitional logic. The exact meaning will always be clear from the context.

4. TIMED CIRCUITS ANALYZER

Timed Circuits Analyzer (TCA) toolbox [7] is an extension of the IF tool-chain [2] with the two following main contributions:

- A language and the associated IF model generator for modelling of asynchronous circuits;
- An integrated engine with an efficient forward reachability algorithm implementing a technique for merging of clock valuation zones.

The latter contribution is particularly important, since it allows a considerable reduction of the state space to be explored at analysis stages.

The underlying formalism for the IF/TCA tool-chain is that of the timed automata extended with signal variables. In the following section, we briefly present the basic timed automata formalism. The extension with signal variables is rather straightforward.

4.1 Timed Automata

Timed automata were introduced in [1]. An extended presentation is also available in [7] as the background for the TCA extension of IF. Here, we only provide a brief overview.

Definition 3. Let C be a finite set of clocks. Let Ψ_C and Γ_C be the sets of respectively constraints and update operations (resetting, deactivation and copying) on clocks from C.

A *timed automaton* is a tuple $(Q, q_0, C, \Sigma, I, \Delta)$, where

- Q is a finite set of *control states*,
- $q_0 \in Q$ is the *initial state*,
- Σ is a finite set of *labels*,
- $I : Q \to \Psi_C$ is a function mapping each state to a *state invariant* (the automaton is only allowed to stay in a state q as long as the invariant $I(Q)$ is satisfied),
- $\Delta \subseteq Q \times \Psi_C \times \Sigma \times \Gamma_C \times Q$ is a *transition relation*.

A transition $(q, g, a, \gamma, q') \in \Delta$ consists of the source and target states (q and q' respectively), a label a, a guard g (a condition on clocks that must be satisfied for the transition to be enabled) and a function γ associating update operations to each clock.

We speak of *control states* in order to distinguish them from the the global states or *configurations* of the automaton. Configurations are pairs (q, v), where $q \in Q$ is a control state and $v : C \to (\mathbb{R}_+ \cup \{\perp\})^{|C|}$ is a *valuation* of clocks. In the initial configuration $(q_0, 0)$, all active clocks are set to zero.

The semantics of timed automata is given in terms of *runs*, that is sequences of steps from one configuration to another.

Definition 4. Let $\mathcal{A} = (Q, q_0, C, \Sigma, I, \Delta)$ be a timed automaton.

- A *discrete step* of $\mathcal{A}$ is a transition of the form $(q, v) \xrightarrow{a} (q', v')$, with $(q, g, a, \gamma, q') \in \Delta$ such that $v \models g$ (the guard condition is satisfied) and $v' = \gamma(v)$.
- A *time step* of $\mathcal{A}$ is a transition of the form $(q, v) \xrightarrow{d} (q, v + d)$, with $d \in \mathbb{R}_+$ such that $\forall d' \in [0, d], v + d' \models I(q)$ (the state invariant in the state q is satisfied).
- A *compound step* is a transition of the form $(q, v) \xrightarrow{(a,d)} (q', v' + d) \stackrel{def}{=} (q, v) \xrightarrow{a} (q', v') \xrightarrow{d} (q', v' + d)$.

Notice that, during a discrete transition, time "does not advance", that is the only possible clock operations are resetting, deactivation and copying.

Any run of a timed automaton is equivalent to a run consisting only of compound steps, except potentially a first time step. In the following, we will only consider runs of this form.

The original decidability proof for verification of timed automata [1] was based on partitioning the space of clock valuations into a finite number of equivalence classes called *regions*. TCA relies on a more advanced notion of *zones*, which are convex polytopes composed of a number of regions. The more clocks there are in the system, the higher is the dimension of the clock valuation space. Hence, it is crucial for efficient analysis of timed automata to keep the number of clocks involved to the minimum.

4.2 Abstraction techniques in TCA

In this section, we present the abstraction techniques implemented in TCA [7, Section 7.1.2], on which we rely in step 3 of the procedure described in the introduction.

The abstraction starts with an automaton $\mathcal{A}_X$ operating on the set X of signals and involves the following steps: $\mathcal{A}_X \rightsquigarrow \mathcal{A}_X^{+\hat{C}} \rightsquigarrow \mathcal{A}_X^r \rightsquigarrow \mathcal{A}_X^{\hat{C}} \rightsquigarrow \mathcal{A}_{X_{io}} \rightsquigarrow \mathcal{A}_{X_{io}}^m$, which are explained below.

1. $\mathcal{A}_X \rightsquigarrow \mathcal{A}_X^{+\hat{C}}$: the $\mathcal{A}_X$ automaton is extended with a set $\hat{C}$ of auxiliary clocks that do not participate in transition guards, nor in the state invariants. These clocks only *observe* the behaviour of the obtained automaton $\mathcal{A}_X^{+\hat{C}}$ and measure the time elapsed since the corresponding input events.
2. $\mathcal{A}_X^{+\hat{C}} \rightsquigarrow \mathcal{A}_X^r$: the forward reachability algorithm is applied to obtain an equivalent timed automaton $\mathcal{A}_X^r$, where all paths correspond to a realizable qualitative behaviour.
3. $\mathcal{A}_X^r \rightsquigarrow \mathcal{A}_X^{\hat{C}}$: timing constraints in $\mathcal{A}_X^r$ are projected on auxiliary clocks $\hat{C}$.
4. $\mathcal{A}_X^{\hat{C}} \rightsquigarrow \mathcal{A}_{X_{io}}$: the automaton is projected on the input and output signal variables thus rendering some transitions unobservable.
5. $\mathcal{A}_{X_{io}} \rightsquigarrow \mathcal{A}_{X_{io}}^m$: the automaton is reduced by merging bisimilar states that produced by the previous step.

4.3 Propagation Waves

The auxiliary clocks $\hat{C}$ added to the automaton $\mathcal{A}_X$ in the first step of transformation in the previous section correspond to input events. Each input event triggers a *propagation wave* that consists of the set of ports in an excited state caused by this event. When an input event occurs, a clock in $\hat{C}$ is initialised. This clock is deactivated when the wave triggered by the corresponding event disappears, that is the last port is removed from the wave. Thus, auxiliary clocks measure the time elapsed since the occurrence of the corresponding input event. Furthermore, auxiliary clocks and waves are in a one-to-one correspondence. Waves are modelled in TCA by dedicated IF processes and can be reused once the triggering event has disappeared from the system. When the number of events present in the system is bounded—which is the case in the context of this paper—a finite number of waves and auxiliary clocks is sufficient to analyse the temporal behaviour of the system.

In TCA models, identifiers of the wave processes are communicated among the processes modelling the ports along with the signal values. Thus, when a port reacts to an event, it is automatically removed from the corresponding wave and replaced by ports that take on input the corresponding produced signal. Each port of the circuit can only be part of one wave at a time. When an excited port receives a signal associated to a different wave, one speaks of the *wave covering*, modelled in the generated product automaton by a transition to the state "COVERING". In the following sections, we will show how we exploit the information associated to such transitions.

4.4 TCA output

TCA takes on input a `.tc` file with the circuit description consisting of three parts:

1. declaration of input signals,
2. declaration of output signals,
3. a list of gate declarations, each comprising the name of the signal computed by the gate, the lower and the upper delay bounds and the boolean expression the input signals of the gate.

Based on the `.tc` file, the TCA front-end generates a `.if` file with the IF model of the circuit comprising one IF process for each input signal and each gate of the circuit, as well as a generic IF process modelling the propagation waves. This model is then used to generate C code for the model exploration with the IF/TCA engine. Finally, this simulator generates a number of automata representing the entire circuit and corresponding to the abstractions presented in Sect. 4.2. For each automaton, several files can be generated, among which we use the following four:

`.states` — this file provides information about all states of the automaton: values of input and output signals, active waves and the state invariant;

`.trans` — this file provides information about all transitions of the automaton: associated event (signal edge), wave operations and the guard;

`.pdf` — the graphical representation of the automaton;

`.pif` — a template IF process that can be used as a component in the IF models of higher hierarchical levels of the circuit.

5. CASE STUDY ANALYSIS

In this section, we present two examples: first, we model and analyse the temporal behaviour of a *switch*—a basic element of the circuits in the context of this paper—; then we proceed to the analysis of a case-study based on the protection system of the P4 reactors.[3]

5.1 Switch

The *switch* is a circuit element with the following cyclic behaviour:

1. The current value of the input signal is consulted;
2. A processing with the duration bounded by a given interval $[T_m, T_M]$ is performed;
3. The value is propagated to the output.

For circuits, this type of behaviour, is called *synchronous*, since subsequent computation cycles are clearly separated. Although strictly speaking, it cannot be called *periodic*, since the duration of each cycle can vary, it can be considered *periodic with jitter*, for example by taking the period to be T_m and the jitter $(T_M - T_m)$.

Whenever the input signal of a switch changes, the propagation of the new value is subject to two delays:

1. The new value is perceived by the switch when the previous processing cycle terminates. This delay is bounded by $[0, T_M]$.
2. Once the new value is perceived, its propagation is further delayed for the duration of the processing. This delay is bounded by $[T_m, T_M]$.

When a second change of the input signal occurs while the previous one is not yet perceived by the switch, the previous value is lost. We have to compute the possible shapes of the output signal of the switch in response to a pulse of duration λ (Fig. 3).

In TCA, we model the switch by specifying explicitly the two delays:

```
input {x0}
output {z0}
y0 : [0, T_M] x0;
z0 : [T_m, T_M] y0;
```

An internal signal y_0 models the perception delay, whereas the delay associated to the output signal z_0 models the processing delay in the switch. TCA does not implement symbolic parameter manipulation. In this example we will use values from the P4 case study: $T_m = 27\ ms$ and $T_M = 33\ ms$.

Figure 5 shows a view of the automaton generated by TCA. This view does not contain time information. For each control state, it shows values of the signals x_0 and z_0 and the list of active clocks. For each transition, this view shows the

[3]For the obvious confidentiality reasons, this example does not represent the actual reactor protection system, but is only inspired by it.

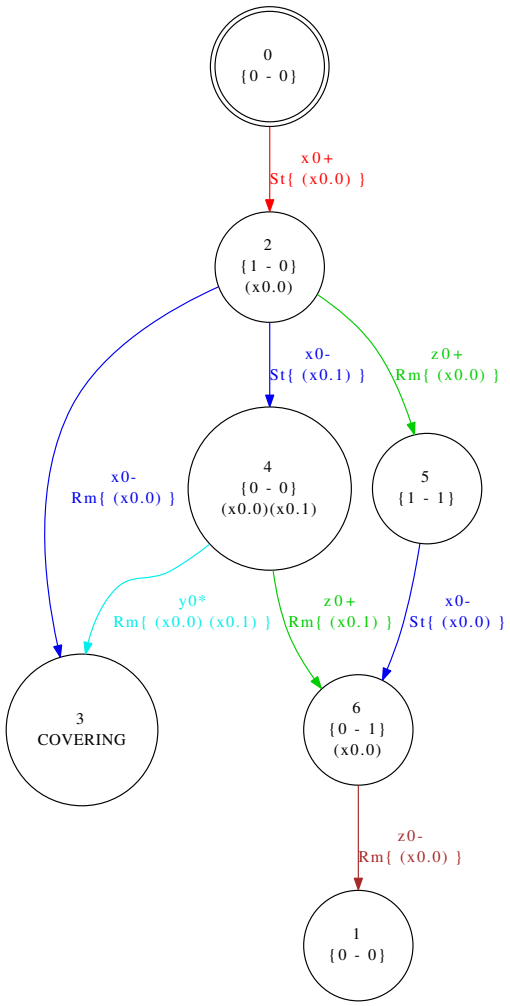


Figure 5: Switch model generated by TCA

event (signal edge) and the associated clock operations. For example, the transition $0 \to 2$ is associated with the raising edge of the input signal x_0 (denoted $x0+$ in the diagram) and starts the clock $x0.0$. Transition $2 \to 3$ (resp. $4 \to 3$) models wave covering (see Sect. 4.3) when two edges of the signal x_0 (resp. y_0) occur before the corresponding reaction has been produced. The event $y0*$ denotes a sequence of a raising edge of y_0 followed a falling one—this information is abstracted away, since y_0 is an internal signal (cf. Sect. 4.2).

The transition $2 \to 3$ models the phenomenon discussed above: the loss of the input signal value when a second change occurs while the previous one is not yet perceived by the switch. The transition $4 \to 3$ is a modelling artefact: a change of the internal signal y_0 models the perception of the input signal by the switch and can only occur at the end of the processing cycle, when the previous change of y_0 is propagated to the output. In other words, the first delay in the switch model is inertial, whereas the second one is a transmission line delay. Although TCA does not make this distinction, knowledge of the system allows one to systematically ignore artefacts as above during system analysis.

Before proceeding with the timing analysis of the switch, we compute the shape of the output using transitional log-

ics. The transitional logic value for the pulse signal is F_1; as discussed above, the switch can be modelled by a composition of an inertial delay $\Box$ and a transmission line delay $\triangle$. Thus the switch response to the pulse signal is

$$\triangle(\Box(F_1)) = \triangle(F_{0,1}) = F_{0,1}\,,$$

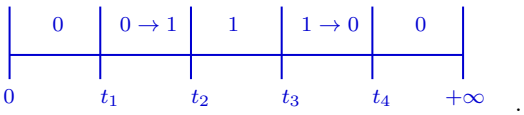
that is either a constant zero signal F_0 or a pulse F_1.

As mentioned above, the loss of the input signal (output F_0) is modelled by the transition $2 \to 3$ (Fig. 5). TCA provides the following temporal annotation for this transition:

```
TRANSITION    : 2 -> 3
Event         : x0-
Waves killed  : (x0.0)
Time          : (x0.0)[0,33]
```

The last line represents the guard, that is the transition is enabled only when the value of the clock $x0.0$ is in the interval $[0, 33]$. The clock $x0.0$, here, is started on the occurrence of the raising edge $x0+$, which happens at the time δ (Fig. 3). The date of falling edge is $\delta + \lambda$. We conclude that the input signal cannot be lost provided $\delta + \lambda > \delta + 33$, i.e. $\lambda > 33$.[4]

Assuming that the signal is not lost, the response of the switch is necessarily the pulse F_1. However, we cannot precisely time-stamp its edges, therefore we consider an equivalent signal over a larger domain



.

Here, the raising (resp. falling) edge of the switch response signal is in the interval $[t_1, t_2]$ (resp. $[t_3, t_4]$). In particular, the signal is guaranteed to be true in the interval $[t_2, t_3]$.

In order to determine t_1 and t_2, we consider transitions $2 \to 5$ and $4 \to 6$ associated to the rising edge of the signal z_0. TCA provides the following information:

```
TRANSITION    : 2 -> 5
Event         : z0+
Waves killed  : (x0.0)
Time          : (x0.0)[27,66]

TRANSITION    : 4 -> 6
Event         : z0+
Waves killed  : (x0.1)
Time : (x0.0)[27,66] (x0.1)[0,33] ...
```

The guards of both transitions have the same interval $[27, 66]$ for the clock $x0.0$.[5] Hence, $z0$ cannot rise before $t_1 = \delta + 27$ and must have risen after $t_2 = \delta + 66$.

Before continuing with the analysis, one should observe the following. In order to maintain a finite number of auxiliary clocks, TCA normalises the clock usage when a wave disappears from the circuit. For example, in the transition $4 \to 6$, the rising edge of z_0 triggers the end of the wave started by the rising edge of x_0. The associated clock $x0.0$ is deactivated. However, in the state 4, the clock $x0.1$, activated by the transition $2 \to 4$, is active. Hence, this clock is copied $x0.0 := x0.1$ and deactivated. Thus, in the state 6, the clock $x0.0$ measures the time elapsed since the falling edge of x_0.

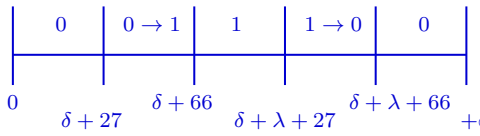
In order to determine t_3 and t_4, we consider the transition $6 \to 1$ associated to the falling edge of z_0. TCA provides the information below, which allows us to conclude $t_3 = \delta + \lambda + 27$ and $t_4 = \delta + \lambda + 66$.[6]

```
TRANSITION    : 6 -> 1
Event         : z0-
Waves killed  : (x0.0)
Time          : (x0.0)[27,66]
```

Finally, notice that $t_2 \leq t_3$ is equivalent to $\lambda \geq 39$.

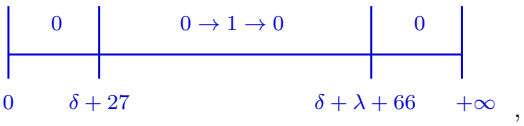
Summarising the above discussion, we conclude that three possibilities exist for the response of the switch to the pulse signal in Fig. 3:

If $39 \leq \lambda$, the switch response is a pulse with known bounds for the interval where the value 1 is maintained



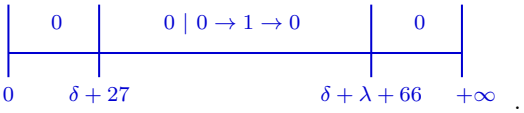
, (2)

if $33 < \lambda < 39$, the switch response is a pulse, but our approach does not provide any guarantees with respect to the interval where the value 1 is maintained



,

if $\lambda \leq 33$, the input signal can be lost



.

Notice that a simple manual analysis shows that, due to the synchronous nature of the switch, whenever the input signal is not lost, it is maintained on the output for at least $T_m = 27$. Since TCA does not distinguish between transmission line and inertial delays, this information cannot be included in the model.

5.2 Analysis of the P4 Protection System

In this section, we present an overview of the analysis of the P4 protection system focusing mainly on the compositional aspect rather than on the completeness of the analysis.

5.2.1 *Partial Triggers*

Each partial trigger signal is used twice in the CUs: once in the CU where it is produced and once in the corresponding CU of the other group, after the transmission over the network.

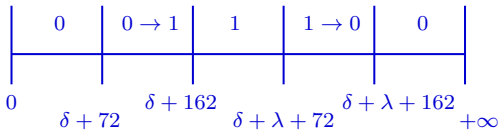
In the first case, partial trigger signals of the primary CUs ($PT_{1,j}$ with $j = 1, 2$) are modelled exactly as in Sect. 5.1, whereas, for the secondary CUs ($PT_{2,j}$ with $j = 1, 2$), the appropriate values of T_m and T_M must be substituted. Assuming $\lambda \geq 90$, we obtain the signal in (2) for the primary

[4] In this simple example, we can conclude that the sufficient condition for the signal to be preserved is $\lambda > T_M$.

[5] Again, this can be translated as $T_m \leq x0.0 \leq 2T_M$.

[6] Idem, $t_3 = \delta + \lambda + T_m$ and $t_4 = \delta + \lambda + 2T_M$.

CUs, and



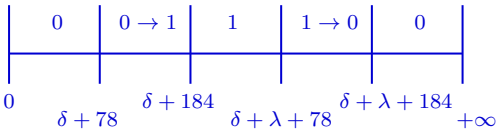
(3)

for the secondary CUs.

In the second case, we have to consider three additional delays: a transmission line delay for the network latency and a pair of delays to model a switch at the reception by the target CU. Transitional logic analysis gives the following shape of the received signal:

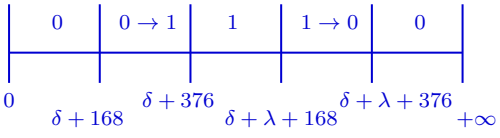
$$\triangle(\square(\triangle(\triangle(\square(F_1))))) = \triangle(\square(F_{0,1})) = F_{0,1}\,.$$

Analysis, similar to that of Sect. 5.1, of $PT_{1,j}$ $(j = 1, 2)$ shows that the signal cannot be lost provided that $\lambda > 100$. Furthermore, when $\lambda \geq 106$, the following bounds can be computed for the edge time-stamps:



.

Similarly, for $PT_{2,j}$ $(j = 1, 2)$, the signal cannot be lost provided that $\lambda > 199$ and, when $\lambda \geq 208$, the following bounds can be computed for the edge time-stamps:



.

5.2.2 *Disjunction of two partial triggers*

Before moving on with the analysis, one has to observe that for a given circuit TCA generates, firstly, an IF system with one process for each signal (input, output and internal) and one process for the waves. Secondly, the toolbox generates a single IF process encoding the product automaton modelling the entire system. For the compositional analysis, we use this latter process to replace one of the processes generated by TCA before the analysis of the compound system.[7]

More specifically, we start by modelling the structure of the compound system (a disjunction of two partial triggers within a CU) in TCA as follows:

```
input {x0}
output {z0}
y0 : [0,0] x0;
y1 : [0,0] x0;
z0 : [0,0] (y0+y1);
```

Here the delay between $x0$ and $y0$ (resp. $x0$ and $y1$) is a placeholder for the partial trigger generated within the CU (resp. received over the network). TCA generates two processes for each of the signals $y0$ and $y1$, which we manually replace by the process generated for the partial triggers at the previous analysis stage (Sect. 5.2.1). Hence, the associated delays are irrelevant and taken to be zero without loss of generality. Notice that we use the same input signal for both $y0$ and $y1$. This reflects the simplifying hypothesis that the input signals are synchronised (Sect. 2). The signal $z0$ models the ideal OR port (associated delay in the system is considered to be negligible).

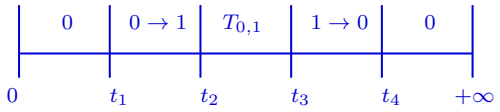
[7]These substitutions have been realised manually for the presented case study, but can be automated in the future.

For the transitional logic analysis, we combine the results obtained for the partial triggers—both $F_{0,1}$—with an ideal OR operator to obtain:

$$\begin{aligned} F_{0,1} \vee F_{0,1} &= (F_0|F_1) \vee (F_0|F_1) \\ &= (F_0 \vee F_0)|(F_0 \vee F_1)|(F_1 \vee F_1) \\ &= F_0|F_1|F_{1,2} \;=\; F_{0\ldots2}\,, \end{aligned}$$

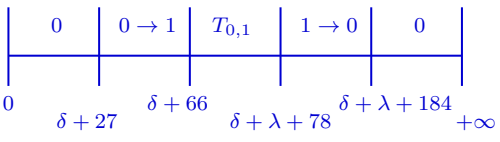
that is a constant zero signal, a pulse or a double pulse.

Since we are only interested in answering the two questions in Sect. 2, we only have to consider the situation where the input signal is not lost. For the disjunction of two signals, this is true provided that the input signal is not lost in at least one of them, which is the case when $\lambda > 33$ (resp. $\lambda > 82$) for the primary (resp. secondary) CUs. Thus the output signal is limited to $F_{1,2}$. Assuming stronger conditions $\lambda \geq 39$ (resp. $\lambda \geq 90$) for the primary (resp. secondary) CUs (see Sect. 5.2.1), the output signal can be represented as follows:

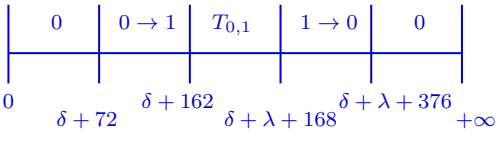


.

By considering the guards in the generated automata as in the previous sections, we obtain the following signals for primary and secondary CUs respectively:



,



.

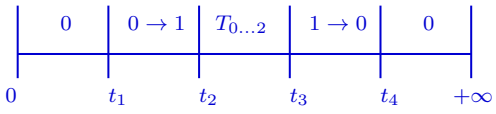
5.2.3 *Partial activators and activator*

The rest of the analysis of the case study consists in repeating iteratively the steps described in the previous sections. A partial activator is computed by taking the conjunction of the signals obtained in Sect. 5.2.2 for primary and secondary CUs of the corresponding group. Observe that the two groups are completely symmetrical, hence it is sufficient to analyse only one of them.

The transitional logic analysis provides us the shape of the resulting signal:

$$\begin{aligned} & F_{0\ldots2} \wedge F_{0\ldots2} \\ &= (F_0|F_1|F_2) \wedge (F_0|F_1|F_2) \\ &= (F_0 \wedge F_{0\ldots2})|(F_1 \wedge F_1)|(F_1 \wedge F_2)|(F_2 \wedge F_2) \\ &= F_0|F_{0,1}|F_{0\ldots2}|F_{0\ldots3} \;=\; F_{0\ldots3}\,, \end{aligned}$$

which can be represented as follows:



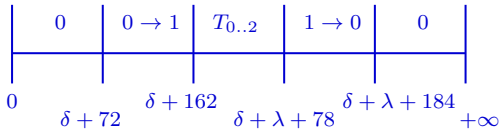
.

Similarly to Sect. 5.2.2, we start by modelling the structure of the compound system by the following TCA circuit:

```
input {x0}
output {z0}
y0 : [0,0] x0;
y1 : [0,0] x0;
z0 : [0,0] (y0*y1);
```

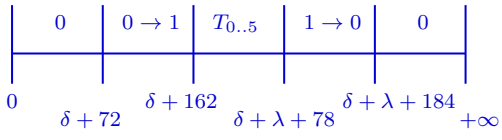
We generate the IF model with the TCA parser, replace the IF processes generated for the intermediate signals $y0$ and $y1$ by those generated following the analysis in the previous phase, and run the TCA forward reachability engine on the obtained model.

The wave covering analysis as in Sect. 5.1 shows that the input signal cannot be lost, provided $\lambda > 162$, whereas the analysis of the other relevant transitions results in the output signal



.

Finally, the Activator signal is the disjunction of the two Partial Activators. The transitional logic analysis provides us with the untimed signal shape $F_{0...3} \vee F_{0...3} = F_{0...6}$. As in Sect. 5.2.2, we conclude that the input signal cannot be lost provided $\lambda > 162$, since this guarantees that it is not lost in either of the components of the disjunction, and, assuming that the signal is not lost, the output Activator signal is



.

Among others, this result provides the answers to the two questions of Sect. 2:

1. The Activator signal is *guaranteed* to be raised, provided that the alarm signal is maintained by the sensors for at least 162 ms.

2. Whenever the Activator signal is raised by the circuit, this happens no longer than 162 ms after the alarm is raised by the sensors.

6. CONCLUSION

In this paper, we have applied a combination of two existing techniques, that is abstract interpretation with transitional logics [9] and compositional timing analysis with the IF/TCA tool chain, to a case study inspired by the software protection system of a P4 nuclear reactor. This case study belongs to the class of asynchronous circuits with synchronous components, a subclass of the so-called *Globally Asynchronous, Locally Synchronous* (GALS) systems. We have shown that the proposed methodology allows one to answer essential questions about temporal behaviour of such systems while avoiding the combinatorial explosion of the state space of the analysed systems.

The proposed methodology can be authomated and implemented as an applicative extension of the IF/TCA tool chain. Furthermore, this study allowed us to exhibit some improvements to the TCA tool chain, such as a possibility to explicitly distinguish inertial and transmission line delays in the TCA models. Future work will have to include a comparison with other analysis tools and techniques, particularly, UPPAAL and Real-Time Calculus.

Acknowledgements

The author would like to thank Ramzi Ben Salah, Marius Bozga and Oded Maler for their help with understanding the inner workings of both IF and TCA tools, as well as the anonymous reviewers for their comments and suggestions, even though some of these could not be implemented in the present version of the paper for technical reasons.

7. REFERENCES

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

[2] M. Bozga, S. Graf, I. Ober, I. Ober, and J. Sifakis. The IF toolset. In *Formal Methods for the Design of Real-Time Systems*, volume 3185 of *LNCS*, pages 237–267. Springer, 2004.

[3] S. Chakraborty, S. Künzli, and L. Thiele. A general framework for analysing system properties in platform-based embedded system designs. In *DATE 2003*, pages 190–195. IEEE Computer Society, 2003.

[4] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, POPL '77, pages 238–252, New York, NY, USA, 1977. ACM.

[5] C. Ferdinand, R. Heckmann, M. Langenbach, F. Martin, M. Schmidt, H. Theiling, S. Thesing, and R. Wilhelm. Reliable and precise WCET determination for a real-life processor. In *Embedded Software Workshop*, volume 2211, pages 469–485, Lake Tahoe, USA, October 2001.

[6] K. G. Larsen, P. Pettersson, and W. Yi. Model-Checking for Real-Time Systems. In *Proc. of Fundamentals of Computation Theory*, LNCS 965, pages 62–88, Aug. 1995.

[7] R. B. Salah. *On Timing Analysis of Large Systems*. PhD thesis, INPG, Grenoble, 2007.

[8] R. B. Salah, M. Bozga, and O. Maler. Compositional timing analysis. In S. Chakraborty and N. Halbwachs, editors, *EMSOFT*, pages 39–48. ACM, 2009.

[9] S. Thompson and A. Mycroft. Abstract interpretation of combinational asynchronous circuits. *Science of Computer Programming*, 64(1):166–183, 2007.

[10] Wikipedia. Nuclear power in France — Wikipedia, the free encyclopedia, 2011. [Accessed 29-July-2011].

[11] S. Yovine. KRONOS: A verification tool for real-time systems. *International Journal on Software Tools for Technology Transfer (STTT)*, 1:123–133, 1997. 10.1007/s100090050009.