



Rémy Chevallier,  
Émmanuelle Encrenaz-Tiphene,  
Laurent Fribourg and Weiwen Xu

Timed Verification  
of the Generic Architecture  
of a Memory Circuit  
Using Parametric Timed Automata

Research Report LSV-06-14

July 2006

Laboratoire  
Spécification  
et  
Vérification



CENTRE NATIONAL  
DE LA RECHERCHE  
SCIENTIFIQUE

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



# Timed Verification of the Generic Architecture of a Memory Circuit Using Parametric Timed Automata <sup>\*</sup> <sup>\*\*</sup>

Remy Chevallier<sup>1</sup>, Emmanuelle Encrenaz-Tiphene<sup>2</sup>,  
Laurent Fribourg<sup>2</sup> and Weiwen Xu<sup>2</sup>

<sup>1</sup> STMicroelectronics, FTM, Central R&D, Crolles, France

<sup>2</sup> LSV - CNRS, ENS de Cachan, France

**Abstract.** Using a variant of Clariso-Cortadella's parametric method for verifying asynchronous circuits, we analyse some crucial timing behaviors of the architecture of SPSMALL memory, a commercial product of STMicroelectronics. Using the model of parametric timed automata and model checker HYTECH, we formally derive a set of linear constraints that ensure the correctness of the response times of the memory. We are also able to infer the constraints characterizing the optimal setup timings of input signals. We have checked, for two different implementations of this architecture, that the values given by our model match remarkably with the values obtained by the designer through electrical simulation.

**Keywords.** Memory Circuit, Timed Automata, Model Checking.

---

<sup>\*</sup> Partially supported by project MEDEA+ Blueberries.

<sup>\*\*</sup> A preliminary version appeared in the Proceedings of 4th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS'06), Sept. 2006.

## 1 Introduction

The designer of a memory circuit guarantees the correct behavior of its product and some quantitative performance (called “(guaranteed) response times”), assuming that the circuit is embedded into an environment satisfying some quantitative requirements (called “(external) requirements”). The response times represent an upper bound on the time taken by the circuit to produce the result of a (write or read) operation. The external requirements specify the lower bounds on the clock cycle times and the stability times of input signals. These values of the parameters of the specification form the *datasheet* of the circuit provided by the manufacturer to the customer. They are determined by *electrical simulation*. In order to perform such a simulation, each component of the memory is modelled at the transistor level as a set of differential equations, which represent the Kirshoff laws associated with the electric current traversing the component. These differential equations depend on the physical characteristics of the transistors and wires (capacitors, resistors, ...). The full memory is thus represented as a big system of differential equations. The values of the datasheet are computed by resolution of these differential equations, using, e.g., tool HSIM [1]. Actually, such a simulation process is much too long to be performed in a complete manner. Sensitive portions of the circuit, which are supposed to contain the longest paths of traversal, are therefore identified by hand. Electrical simulations are performed only for such limited portions of circuit, which are assumed to contain the critical paths. Such an assumption of ‘criticality’ is risky: it is very difficult to identify by hand relevant sensitive portions of the circuit (especially when the complexity of the circuit increases). The need for *formal methods* to verify the timing values of the datasheet is therefore widely recognized.

In [9, 8], Clariso and Cortadella propose a formal method for verifying the timings of asynchronous circuits. The approach infers a set of sufficient linear constraints relating the delays of the internal gates of the circuit to the external delays of the circuit specification that guarantee the correct behavior of the circuit. The method is based on the reachability analysis of a timed model of the circuit (with additional abstract interpretation techniques [10]). As pointed out in [9], such parametric constraint sets are very informative for the designer, as they identify sensitive parts of the circuits (e.g., “critical paths”) and inter-relations between various data of the specification. Moreover, many technology mappings can be tested immediately (by mere instantiation of the parameters). We follow here a similar approach for formally verifying some generic properties of a commercial memory designed by STMicroelectronics, called SPSMALL. Our timing analysis method derives a set of sufficient linear constraints relating the external parameters of the datasheet to the internal gate delays that guarantee the correctness of the circuit’s behavior. In particular, these constraints can be seen as sufficient conditions for certain paths of the circuit to be “critical” (i.e. those along which the propagation delay is the longest). Using the model of parametric timed automata (see [3]) and tool HYTECH [13] for reachability analysis, we are able to generate a set of linear constraints that ensures the correctness

of some crucial timing behaviors of the memory. This method is applicable to several instances of SPSMALL memories implemented with different transistor technologies (corresponding to different sets of parameter values). We actually applied the method to two instances, called *SP1* and *SP2*, which correspond to a high-speed and low-power implementations respectively.

**Comparison with Related Work.** As pointed out above, our work is adapted from Clariso-Cortadella’s method [9, 8]. However, [9, 8] focus on a particular form of linear constraints (linear inequalities with coefficients always equal to  $\pm 1$ ) and represent them as a particular form of convex polyhedra, called “octahedra” [8]. In contrast, we use here linear constraints and their classical form of convex polyhedra in their full generality [12]. Other differences with [9, 8] are:

- a different level of modelling: the components of the memory are represented here at the “latch” level instead of gate level. At this level of representation, the flow of input signals traverses the circuit in a linear manner (without loops), while the flow is cyclic at the gate level (the output of one gate can be an input of another gate and the converse can be simultaneously true),
- the use of mere forward reachability analysis rather than techniques of fixpoint computation of abstract interpretation (using, e.g., widening operators),
- the use of a decompositional approach, which splits the global system into three smaller parts,
- the use of a *step-by-step refinement* of the reachability analysis process: we start with the most general form of constraints on parameters; we then refine them progressively, via *iterative* reachability analysis, detecting, at each run, some erroneous generated states until complete elimination.

Besides [9, 8], our work is along the lines of [15, 5, 17] where timed automata have been used extensively to model and check timing properties of asynchronous circuits (cf. [11]). Reachability analysis is there performed via tool KRONOS [18]. Our work is also a continuation of [4, 7] where the SPSMALL memory is modelled as a timed automaton and some of its timing properties are proven by reachability analysis (using tool UPPAAL [14]). The crucial difference here, with respect to these previous works, is that we use the model of *parametric* timed automata (performing reachability analysis with HYTECH [13]).

**Plan of the paper.** In Sect. 2, we present the general objectives of our verification process. In Sect. 3, we give a general description of our method. In Sect. 4, we explain how we apply it on SPSMALL memory after having split the model into 3 parts, in the case of the write property. We give the counterpart results for the read property in Sect. 5. Final remarks are given in Sect. 6.

## 2 Objectives of Verification

### 2.1 Architecture of a Memory Circuit

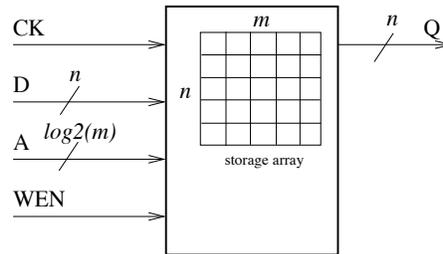
A memory circuit aims at storing data at some addressed locations. It is associated with two operations: ‘write’ and ‘read’. The memory circuit has several

input ports and one output port (see Fig. 1). The signals driven by input ports are:

- $CK$ , the signal of the periodic clock;
- $D$ , the  $n$ -bits width signal representing the data to be stored;
- $A$ , the  $\log_2(m)$ -bits 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  $n$ -bits 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. This is depicted on Fig. 1. 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  $Q$ . 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  $A$ .



**Fig. 1.** Simplified view of a memory (Symbol  $\text{---}/\text{---} \rightarrow$  represents a port of  $k$ -bits width)

## 2.2 Simplification

As a simplification, we will consider the case where the ports of  $D$  are reduced to a single bit. The propagation of all the bits of  $D$  follow identical paths. Each path propagates either a rising edge or a falling edge. Let us point out that the propagation time of a rising edge *differs* from that of a falling edge. However, the propagation of all the rising (resp. falling) edges takes the same time. This justifies the reduction of our model to a datapath of one bit for  $D$ , with *two* cases depending on whether  $D$  is rising or falling. Accordingly, the same simplification is done for  $Q$ .

A similar reduction applies to  $A$ : first, all the bits of  $A$  follow identical paths before reaching the address decoder; then, all the paths in the address decoder

are abstracted as the “slowest” path (the one taking the longest time to propagate the signal).

Likewise, we are naturally led to focus to a unique memory point, denoted by  $mp$ , instead of the original  $m \times n$  storage array. Note that the operations involving the original  $m \times n$  memory points may take different times, depending on their physical location. Our abstraction is justified because we will assume that  $mp$  corresponds to the point of the storage array from which the *slowest* path to  $Q$  originates (See Sect. 5).

A closer view of the storage mechanism is given in Appendix 0. It highlights the reduction of parallel paths for  $D$ ,  $A$  and  $Q$ . For more details about memory architecture, see [16].

### 2.3 Timings

A memory is embedded into a synchronous environment scheduled by a periodic signal, named ‘clock’ ( $CK$ ). The period of a cycle,  $t_{cycle}$ , decomposes itself into a high period ( $t_{HI}$ ) and a low period ( $t_{LO}$ ). In order to be taken into account, each input signal  $I$  has to remain stable for a given amount of time, before the rising edge of the clock. This delay is called *setup* time for  $I$ , and denoted by  $t_{setup_I}$ <sup>3</sup>.

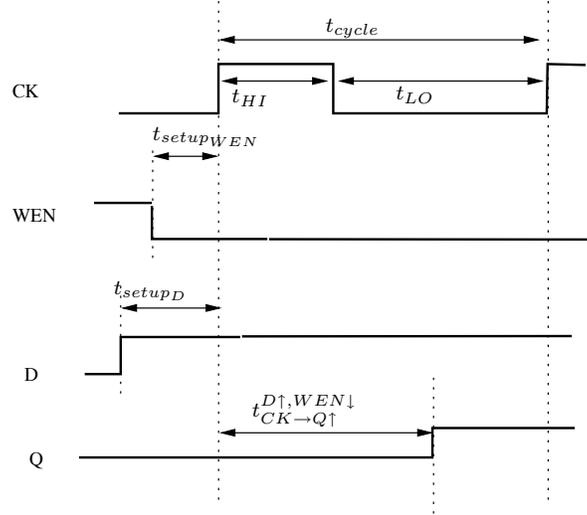
**Traversal of internal components.** The circuit is made of a number elementary internal components (logical gates, latches, wires,...). The logical gates (or blocks) are assumed to be traversed instantaneously<sup>4</sup>. The only components with non-null delay are the wires and the latches. Each wire and latch component is associated with an interval  $[l^\uparrow, u^\uparrow]$  (resp.  $[l^\downarrow, u^\downarrow]$ ), which gives lower and upper bounds of the component traversal delay when the attacking signal is rising (resp. falling)<sup>5</sup>. We adopt the “*inertial* delay” interpretation (see [6]): changes that do not persist for  $l$  time are filtered out.

**Actual response times.** A write operation occurs when input signal  $WEN$  is low. Due to the time of traversal of the elementary components of the memory, this operation takes a delay, called “actual response time for write”, denoted by  $t_{CK \rightarrow Q}^{D, WEN \downarrow}$ . This is depicted on Fig. 2. Due to the one-bit abstraction of  $D$ , this delay now just depends on whether  $D$  is rising or falling. A rising (resp. falling) edge of  $D$  induces a rising (resp. falling) edge of  $Q$ . Accordingly, this delay will be denoted by  $t_{CK \rightarrow Q \uparrow}^{D \uparrow, WEN \downarrow}$  or  $t_{CK \rightarrow Q \downarrow}^{D \downarrow, WEN \downarrow}$ . The delay  $t_{CK \rightarrow Q}^{D, WEN \downarrow}$  is just the

<sup>3</sup> There exists also a required delay of stability, called “hold” time, required *after* the rising edge of the signal, but we will not consider it in this paper.

<sup>4</sup> If the delay of traversal of the gate is non-null, we backpropagate it to the input wires of the gate (cf. [15]).

<sup>5</sup> This is a straightforward generalization of “*bi-bounded* delay” model (see [6]), taking into account the rising or falling nature of input signal.



**Fig. 2.** A write operation corresponding to a rising edge of  $D$  ( $D \uparrow$ ).

maximum of these two values.

Similarly, when a read operation occurs ( $WEN = 1$ ), the “actual response time for read”, denoted by  $t_{CK \rightarrow Q}^{A, WEN \uparrow}$ , can be defined along the same lines. This delay is itself the maximum of the values corresponding to the falling and rising edges of  $A$ , denoted by  $t_{CK \rightarrow Q \uparrow}^{A \uparrow, WEN \uparrow}$  and  $t_{CK \rightarrow Q \downarrow}^{A \downarrow, WEN \uparrow}$  respectively.

**Response times guaranteed by the datasheet.** The datasheet states the maximum delay, denoted by  $t_{max}^{write}$ , after which a write operation is guaranteed to have produced its result. A similar delay, denoted by  $t_{max}^{read}$ , is also given for the read operation. The values  $t_{max}^{write}$ ,  $t_{max}^{read}$  form the (*guaranteed*) response times part of the datasheet.

The setup and cycle timing values ( $t_{HI}$ ,  $t_{LO}$ ,  $t_{setup_D}$ ,  $t_{setup_A}$ ,  $t_{setup_{WEN}}$ ) form the (*external*) requirements of the datasheet.

## 2.4 Verified properties

We suppose that we are given an implementation of the memory, and assume that its environment satisfies the external requirements of the datasheet. We are interested in verifying that the actual response times are no greater than the corresponding response times guaranteed by the datasheet.

**Write property.** The result of a write command is produced on output port  $Q$  within  $t_{max}^{write}$ . This will be expressed as:  $t_{CK \rightarrow Q}^{D \uparrow, WEN \downarrow} \leq t_{max}^{write}$ .

**Read property.** Similarly, we will express the correctness of a read command as:  $t_{CK \rightarrow Q}^{A, WEN \uparrow} \leq t_{max}^{read}$ .

**Optimal setup timings.** Besides, our analysis will allow us to infer an optimal value for setup timing  $t_{setup_D}$  (resp.  $t_{setup_A}$ ): it corresponds to the lowest value of  $t_{setup_D}$  (resp.  $t_{setup_A}$ ) which still guarantees the read and write properties, while letting invariant the values of all the other parameters of the datasheet.

### 3 Method

We suppose that we are given a high-level description of the memory circuit under the form of an abstract (timed) functional graph (see Fig. 3). Given such a graph, the process consists to:

- construct a model of the memory under the form of a timed automaton,
- generate the set of reachable states (within a finite number of cycles, *viz.* 2 cycles),
- infer (by stepwise refinement) a set of timing constraints ensuring the response time property,
- verify that, for a given implementation, the response time property holds by checking the instantiated constraints.

#### 3.1 Modelling SPSMALL

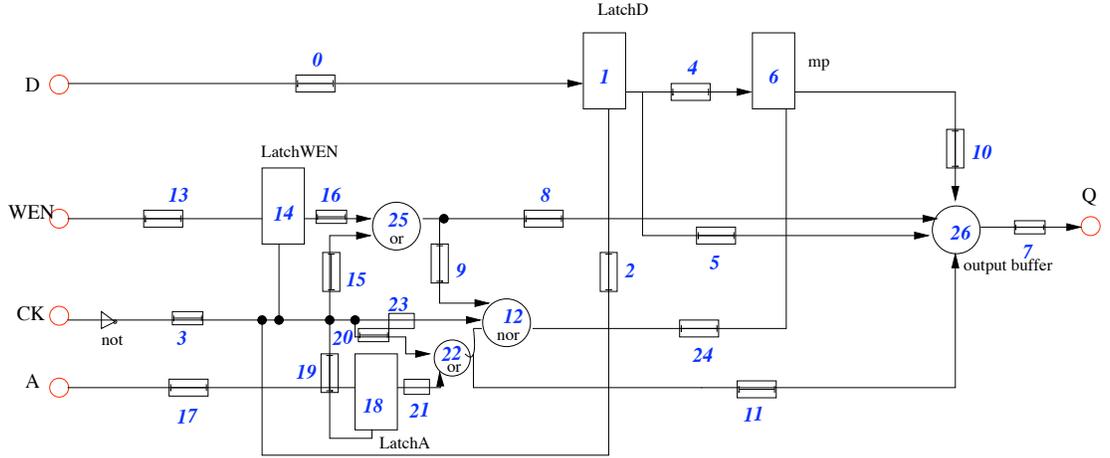
**Level of Modelling.** A memory can be modelled at different levels of complexity, e.g., in a increasing order: at the functional block level, at the “latch” level, at the gate level, or at the transistor level.

For the SPSMALL memory, the model can thus be implemented using: 3 main components, at the block level (cf [4]), 30 components at the latch level, 70 components at the gate level, or 200 components at the transistor level. There is a tradeoff in finding the appropriate level of modelling. The lower the level of modelling is, the more faithful to the reality the model is, but the more difficult the verification process is.

In this work, we chose to represent the memory at the latch level. The advantage is to limit the number of components (around 30) at a reasonable size, and to have a “schematics” describing the architecture of the memory at this level, which closely corresponds to (VHDL) code automatically produced with commercial tool TLL [2]. This is depicted on Fig. 3.

The schematics naturally decomposes into two relevant subparts, depending on whether one studies the circuit on a write or read operation. These parts are depicted on Fig. 7 and Fig. 8 for write and read respectively. Each part is made of three kinds of components: wires, latches and logical blocks:

- A *wire* carries an input signal to its output with a delay in  $[l^\uparrow, u^\uparrow]$  (resp.  $[l^\downarrow, u^\downarrow]$ ).



**Fig. 3.** Abstract graph for SPSMALL

- A *latch* is a component that can store one bit of data. It has two inputs  $d$  (*data to be latched*) and  $e$  (*enable*), and one output  $q$ . It propagates the input data  $d$  to its output  $q$  (with a delay in  $[l^\uparrow, u^\uparrow]$  or  $[l^\downarrow, u^\downarrow]$ ) as long as  $e$  is high. While  $e$  is low,  $q$  keeps its value (even if  $d$  changes). The latch is said to be “closed” (resp. “open”) when  $e = 0$  (resp.  $e = 1$ ).
- Logical blocks are used to output a signal functionally dependent of the inputs of the block. (Recall that we assume in this paper that the delays associated with such blocks are null.)

**SPSMALL as a Timed Automaton.** The model of timed automata [3] is especially well-suited to represent asynchronous circuits (see, e.g., [15, 5, 17]). Roughly speaking, a timed automaton is a finite state automata enriched with (*symbolic*) *clocks* that evolve at the same uniform rate, and can be reset to zero. A *state* is a pair  $(\ell, v)$  where  $\ell$  is a *location* (or “control state”), and  $v$  a clock valuation. Each location is associated with a conjunction of linear constraints over clocks, called *invariant*. A state  $(\ell, v)$  has a *discrete transition*, labelled  $e$ , to  $(\ell', v')$  if  $v$  satisfies a constraint, called *guard*, associated to  $e$ , and  $v'$  is obtained from  $v$  by resetting certain clocks to 0. The state  $(\ell, v)$  has a *time transition* of duration  $t$  to  $(\ell, v')$  if  $v' = v + t$  and for all  $t'$  ( $0 \leq t' \leq t$ ),  $v + t'$  satisfies the invariant associated to  $\ell$ . States can be expressed under the symbolic form of conjunctions of linear constraints. Such states are classically represented as convex polyhedra (see, e.g., [13]). Sets of states correspond to union of polyhedra. The (*forward*) *reachability analysis* consists in generating iteratively the “successors” of sets of states via the system transitions, using elementary manipulation of polyhedra.

The SPSMALL memory is modelled as a timed automaton, resulting from the composition of all the timed automata corresponding to the input signals and the internal components. More precisely:

- Each input signal of the memory (synchronous signal  $CK$ , input signal  $D$ , read/write command  $WEN$ ) is represented as a timed automaton.
- Each component of the memory (latch, wire, ...) is also represented as a timed automaton,
- The transmission of an internal signal between two components is modelled by synchronizing the two corresponding timed automata via a shared discrete transition.

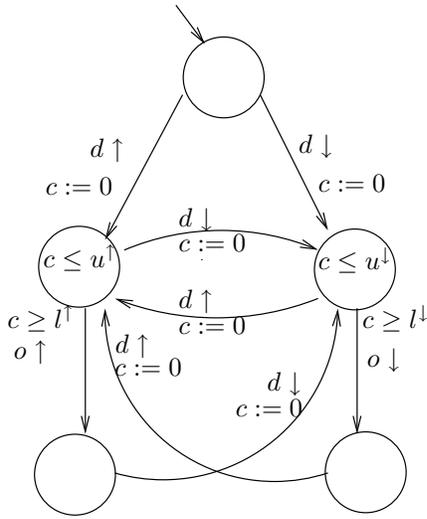
Let us explain how to model a wire as a timed automaton. The wire component is enabled  $l^\uparrow$  (pico)secs after the rising edge of the input signal, and the output signal is fired after a delay lying in  $[l^\uparrow, u^\uparrow]$  (parameters  $l^\uparrow$  and  $u^\uparrow$  are used to propagate a falling edge). Such a wire component is naturally modelled as the timed automaton depicted in Fig. 4. It makes use of 1 internal clock  $c$  and 5 locations. The symbol  $d \uparrow$  (resp.  $d \downarrow$ ) corresponds to a rising edge (resp. falling edge) of input internal signal  $d$ , and similarly for symbol  $o \uparrow$  (resp.  $o \downarrow$ ) with output signal  $o$ . Each edge corresponds to a discrete transition labelled with the name of input signals ( $d \uparrow$  or  $d \downarrow$ ) or output signals ( $o \uparrow$  or  $o \downarrow$ ). The medium-level locations are associated with invariants  $c \leq u^\uparrow$  or  $c \leq u^\downarrow$ . The guard associated with edges outgoing downwards from these locations is  $c \geq l^\uparrow$  or  $c \geq l^\downarrow$ .

Likewise, a latch is modelled as a timed automaton depicted on Fig. 5 (using the same conventions as for the wire automaton). There are 6 locations and 1 clock  $c$ . The name of the locations  $e_i d_j B$  reflects the values  $i$  and  $j$  of  $e$  and  $d$  respectively; besides  $e_1 d_j B$  indicates that an output  $q \uparrow$  (resp.  $q \downarrow$ ) has already been output if  $j = 1$  (resp.  $j = 0$ ).

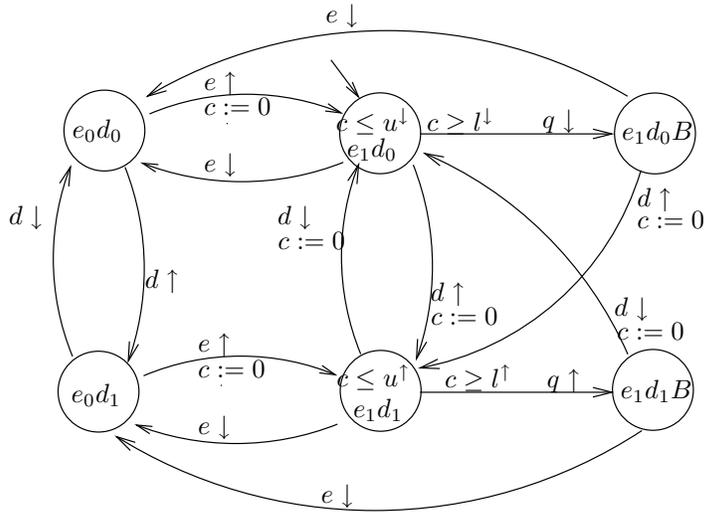
### 3.2 Reachability Analysis

A “global clock”  $s$  (which is initialized to 0, and never reset) is used to measure the evolution of time. The states generated by reachability analysis correspond to linear constraints which relate the values of input and output signals, depending on timing values of the datasheet, the values of the internal delays, as time  $s$  evolves. In order to verify property:  $t_{CK \rightarrow Q \uparrow}^{D \uparrow, WEN \downarrow} \leq t_{max}^{write}$ , we model the behavior of the memory along two cycles:

- a 1st cycle where the values of  $D$  and  $WEN$  are set  $t_{setup_D}$  and  $t_{setup_{WEN}}$  time before the 2nd rising edge of  $CK$  (corresponding to the write operation); input signal  $WEN$  is modelled as a falling edge, followed by a low level (selection of a write command), and input signal  $D$  is modelled as a rising edge, followed by a high level (in keeping with the stabilization requirement  $setup_D$ ).



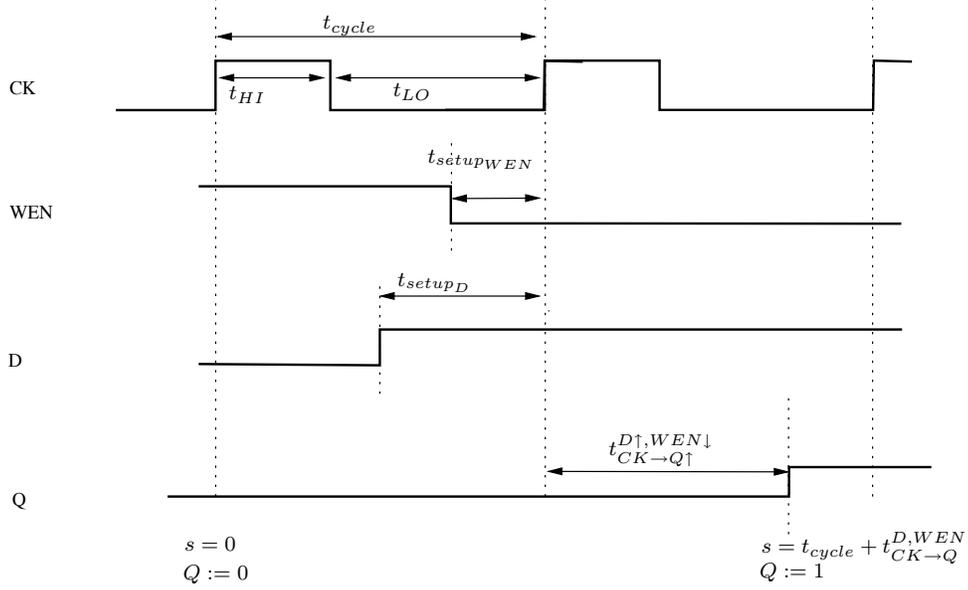
**Fig. 4.** Timed automaton of a wire component with delays  $[l^\uparrow, u^\uparrow]$  and  $[l^\downarrow, u^\downarrow]$  to propagate an edge from  $d$  to  $o$ .



**Fig. 5.** Timed automaton of a latch component with delays  $[l^\uparrow, u^\uparrow]$  and  $[l^\downarrow, u^\downarrow]$  to propagate an edge from  $d$  to  $q$ .

- a 2nd cycle where the write operation is performed (the  $D$  value is propagated on  $Q$  port).

Accordingly, the observation of the generated states is done along two cycles (see Fig. 6).



**Fig. 6.** The *write* operation used in our experiment.

In the next subsection, we will explain how the verification method applies without instantiation of the parameters. Let us first explain how the method works given a specific implementation of the memory: in the rest of this section, we assume all the parameters to be instantiated with the values of the datasheet (for  $t_{HI}$ ,  $t_{LO}$ ,  $t_{setup_D}$ ,  $t_{setup_{WEN}}$ ,  $t_{max}^{write}$ ,  $t_{max}^{read}$ ) and those given by simulation (for the intervals  $[l_i^\uparrow, u_i^\uparrow]$  and  $[l_i^\downarrow, u_i^\downarrow]$ , for each internal component  $i$ ).

We use a flag (initialized to 0) which stores the fact that the rising edge of input signal  $D$  has reached  $Q$  port. Without loss of understanding, this flag is also denoted by  $Q$ . The value of  $s$  when  $Q$  is set to 1, corresponds to the sought value  $t_{CK \rightarrow Q}^{D \uparrow, WEN \downarrow}$ . Starting from  $s = 0 \wedge Q = 0$ , the set of reachable states is iteratively computed until, either:

- the switch of  $Q$  occurs before 2 cycles, or
- 2 cycles are run out without any switch of  $Q$ .

This yields a set of final states, denoted by  $Post^{2t_{cycle}}$ , which can be decomposed into:

- *good* states, i.e., states, for which the switch of  $Q$  has occurred ( $Q = 1 \wedge s = t_{cycle} + t_{CK \rightarrow Q \uparrow}^{D \uparrow, WEN \downarrow} \leq 2t_{cycle}$ ).
- *bad* states, i.e., states, for which 2 cycles has run out without any switch of  $Q$  ( $Q = 0 \wedge s = 2t_{cycle}$ ).

The property holds iff:

- all the final states of  $Post^{2t_{cycle}}$  are good (no bad state), and
- for each final state, the value  $t_{CK \rightarrow Q \uparrow}^{D \uparrow, WEN \downarrow}$  of  $s$  is at most equal to  $t_{max}^{write}$  (i.e.:  $t_{CK \rightarrow Q \uparrow}^{D \uparrow, WEN \downarrow} \leq t_{max}^{write}$ ).

The symmetrical property:  $t_{CK \rightarrow Q \downarrow}^{D \downarrow, WEN \uparrow} \leq t_{max}^{write}$  has to be also verified. This is done along the same principles as described above.

### 3.3 Timing Constraints Extraction by Stepwise Refinement

Let us now explain how to perform the reachability analysis at a *generic* level, without setting the parameters to some specific values of a given implementation. We denote by  $Assumption(D \uparrow)$ , a set of symbolic constraints, relating some parameters of the external requirements of the datasheet (*viz.*,  $t_{HI}$ ,  $t_{LO}$ ,  $t_{setup_D}$ ) and the internal component delays  $\{[l_i^\uparrow, u_i^\uparrow]\}$ ,  $\{[l_i^\downarrow, u_i^\downarrow]\}$  together. At the beginning of the process,  $Assumption(D \uparrow)$  is *True*, which means that we start with the most general parameters ( $l_i^\uparrow$  (resp.  $l_i^\downarrow$ ) is just assumed to be less than or equal to  $u_i^\uparrow$  (resp.  $u_i^\downarrow$ )). Given an initial set of states characterized by  $Assumption(D \uparrow)$ , we perform reachability analysis on 2 cycles. We thus generate a set of constraints, denoted by  $Post^{2t_{cycle}}(Assumption(D \uparrow))$ . The first run of  $Post^{2t_{cycle}}(Assumption(D \uparrow))$ , with  $Assumption(D \uparrow) = True$ , usually contains bad (final) states. The *refinement* process consists to eliminate such bad states, by restricting the possible values of the parameters, as follows:

1. select a bad state of  $Post^{2t_{cycle}}(Assumption(D \uparrow))$ ;
  2. detect a “suspect” constraint;<sup>6</sup>
  3. add the negation of this subconstraint to  $Assumption(D \uparrow)$ .
  4. Recompute  $Post^{2t_{cycle}}(Assumption(D \uparrow))$  after having reset  $s$  and  $Q$  to 0.
- And so on, perform iteratively 1-2-3-4 until no bad state is generated.

At each run,  $Assumption(D \uparrow)$  is a conjunction of linear constraints with an increasing number of conjuncts. Accordingly,  $Post^{2t_{cycle}}(Assumption(D \uparrow))$  decreases at each run. At the end, the refinement process outcomes two formulas,  $Assumption(D \uparrow)$  and  $Final(D \uparrow) \equiv Post^{2t_{cycle}}(Assumption(D \uparrow))$ :

- $Assumption(D \uparrow)$  is a conjunctive constraint binding parameters of the external requirements of the datasheet (cycle and setup timings) to the internal component delays  $\{[l_i^\uparrow, u_i^\uparrow]\}$ ,  $\{[l_i^\downarrow, u_i^\downarrow]\}$ .

<sup>6</sup> In our context, a *suspect* constraint is an inequality that is violated by the values of an SPSMALL available instance, *viz.*  $SP1$  (see Sect. 4.3).

- $Final(D \uparrow)$  is a set of conjunctive constraints relating the actual response time  $t_{CK \rightarrow Q \uparrow}^{D \uparrow, WEN \downarrow}$  to the internal component delays  $\{[l_i^\uparrow, u_i^\uparrow]\}, \{[l_i^\downarrow, u_i^\downarrow]\}$ .

Typically,  $Final(D \uparrow)$  is of the form:

$$f(\bar{l}, \bar{u}) \leq t_{CK \rightarrow Q \uparrow}^{D \uparrow, WEN \downarrow} \leq g(\bar{l}, \bar{u}),$$

where  $f(\bar{l}, \bar{u})$  and  $g(\bar{l}, \bar{u})$  are linear expressions over the delays  $l_i^\downarrow, u_i^\downarrow, l_i^\uparrow, u_i^\uparrow$  associated to all the components  $i$  of the system.

By construction,  $Final(D \uparrow)$  contains good states only (the writing always occurs within two cycles.). Given an implementation of the memory,  $Assumption(D \uparrow)$  and  $Final(D \uparrow)$  can now be used to show that  $t_{CK \rightarrow Q \uparrow}^{D \uparrow, WEN \downarrow} \leq t_{max}^{write}$  is satisfied. This is done by checking that, after instantiation of all the parameters with the values of the datasheet and the values of the internal delays of the implementation:

1.  $Assumption(D \uparrow)$  holds;
2.  $Final(D \uparrow)$  entails  $t_{CK \rightarrow Q \uparrow}^{D \uparrow, WEN \downarrow} \leq t_{max}^{write}$ .

Note that, in the case where  $Final(D \uparrow)$  is of the form  $f(\bar{l}, \bar{u}) \leq t_{CK \rightarrow Q \uparrow}^{D \uparrow, WEN \downarrow} \leq g(\bar{l}, \bar{u})$ , checking item 2 reduces to check:  $g(\bar{l}, \bar{u}) \leq t_{max}^{write}$ .

In an analogous way, in order to verify the counterpart property for the falling edge of  $D$ , one will infer  $Assumption(D \downarrow)$  and  $Final(D \downarrow)$ , and will check:  $t_{CK \rightarrow Q \downarrow}^{D \downarrow, WEN \downarrow} \leq t_{max}^{write}$ .

From  $t_{CK \rightarrow Q \uparrow}^{D \uparrow, WEN \downarrow} \leq t_{max}^{write}$  and  $t_{CK \rightarrow Q \downarrow}^{D \downarrow, WEN \downarrow} \leq t_{max}^{write}$ , one infers that the implementation satisfies:  
 $t_{CK \rightarrow Q}^{D, WEN \downarrow} \leq t_{max}^{write}$ .

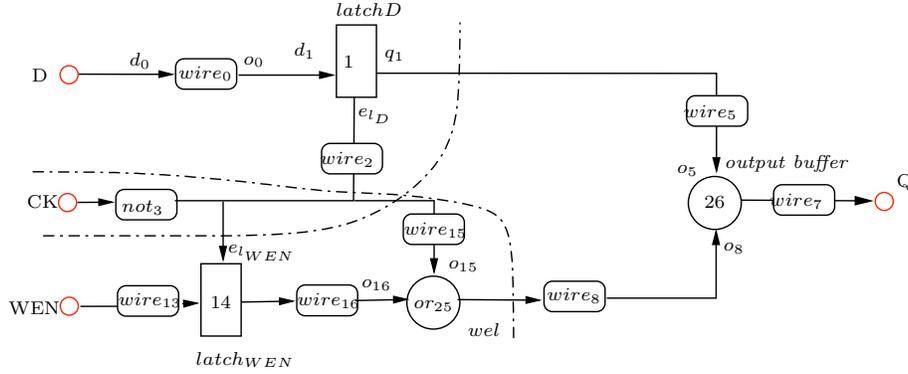
Sets  $Assumption(D \uparrow)$  and  $Assumption(D \downarrow)$  can also be used to find the constraints associated to the *optimal* setup timing  $t_{setup_D}$ : it corresponds to the lowest value of  $t_{setup_D}$  which still guarantees the read and write properties, while letting invariant the values of all the other parameters of the datasheet. This is illustrated in Sect. 4.2.

## 4 Verification of the Write Property of SPSMALL

We apply the method for proving the correctness of the timings of the datasheet of SPSMALL for the write operation. The relevant portion of this schematics for the write property is described on Fig. 7. Note that, somehow surprisingly, the memory point  $mp$  is absent of the schematics: the write property concerns an end-to-end propagation of information from  $D$  to  $Q$ , which actually *bypasses* the memory point (where  $D$  is independently stored)<sup>7</sup>. The schematics contains 12

<sup>7</sup> Such a bypass mechanism is a feature of so-called “write-through” memories.

components: 7 “wire” components, which transmits an input; one “not” component, which transmits the negation of its input ; one “or” component which computes the logical “or” of its two inputs, and two “latches”,  $latch_D$  and  $latch_{WEN}$ .



**Fig. 7.** Schematics of the relevant portion of SPSMALL for write.

We now apply the above method for deriving the sets  $Assumption(D \uparrow)$  and  $Final(D \uparrow)$  of constraints associated with memory SPSMALL, in the case of a rising edge of  $D$ . A similar approach applies for the case of a falling edge.

#### 4.1 Decomposition

The relevant portion of the circuit for the write command from  $D$  to  $Q$  has been depicted on Fig. 7.

In practice, we cannot apply directly the method described above, because we cannot perform reachability analysis with HYTECH due to the high number (34) of parameters. The model is therefore decomposed into three parts (see the dashed lines on Fig. 7):

- The 1st part represents the  $D$ 's edge propagation through  $latch_D$ : The input signals are  $D$  and  $CK$ . The output is the output of  $latch_D$ , denoted by  $q_1$ . A rising edge of  $D$  induces a rising edge on  $q_1$ . The goal is to compute lower and upper bounds of the occurrence time of the rising edge of  $q_1$ , starting from the rising edge of  $CK$ . This delay will be denoted by  $t_{CK \rightarrow q_1 \uparrow}^{D \uparrow}$ .
- The 2nd part represents the propagation of the  $WEN$  falling edge through  $latch_{WEN}$  (among other components). The input signals are  $WEN$  and  $CK$ . The output is denoted by  $wel$ . A falling edge of  $WEN$  induces a falling edge on  $wel$ . The goal is to compute the lower and upper bounds of the occurrence time of the falling edge of  $wel$ , starting from the rising edge of  $CK$ . This delay will be denoted by  $t_{CK \rightarrow wel \downarrow}^{WEN \downarrow}$ .

- The 3rd part represents the propagation of  $q_1$ 's rising edge through  $Q$ . The input signals are  $q_1$  and  $wel$ . The output is  $Q$ . A falling edge of  $wel$  and a rising edge of  $q_1$ , induce a rising edge on  $Q$ . The goal is to compute the lower and upper bounds of the occurrence of the rising edge of  $Q$ , starting from the rising edge of  $CK$ . This will be denoted by  $t_{CK \rightarrow Q \uparrow}^{q_1 \uparrow, wel \downarrow}$ . Using the bounds  $t_{CK \rightarrow wel \downarrow}^{WEN \downarrow}$  and  $t_{CK \rightarrow q_1 \uparrow}^{D \uparrow}$  found in the two first parts, this will allow us finally to determine the constraints on  $t_{CK \rightarrow Q \uparrow}^{D \uparrow, WEN \downarrow}$ .

## 4.2 Generic Constraints

We analyze separately each part, thus obtaining constraints binding intermediate input and output parameters (see Appendix 1,2,3). The “suspect” constraints discarded during the refinement process (see Sect. 3.3) are those incompatible with the values of the delay intervals  $[l_i^\uparrow, u_i^\uparrow]$  and  $[l_i^\downarrow, u_i^\downarrow]$  of instance  $SP1$  (see Sect. 4.3). By recombination of these separate sets of constraints, we obtain constraints relating the inputs and outputs of the whole memory, that are given below. For the case of a rising edge of  $D$ , we have:

$$\begin{aligned}
& \textit{Assumption}(D \uparrow): \\
& t_{setup_D} + u_2^\downarrow + u_3^\downarrow < l_0^\uparrow + t_{LO} \wedge u_2^\downarrow + u_3^\downarrow + u_1^\uparrow < t_{LO} \\
& \wedge u_3^\downarrow + t_{setup_{WEN}} < t_{LO} + u_{13}^\downarrow \wedge u_{13}^\downarrow + u_{14}^\downarrow < t_{setup_{WEN}} + l_3^\downarrow \wedge u_{14}^\downarrow < t_{HI} \\
& \wedge u_{13}^\downarrow + u_{14}^\downarrow + u_{16}^\downarrow < t_{setup_{WEN}} + l_3^\downarrow + u_{15}^\downarrow \wedge t_{setup_D} + u_3^\downarrow + u_{15}^\downarrow \leq l_5^\uparrow + l_0^\uparrow + l_1^\uparrow \\
& \wedge u_5^\uparrow + u_0^\uparrow + u_1^\uparrow \leq l_8^\downarrow + l_3^\downarrow + l_{15}^\downarrow + t_{setup_D}
\end{aligned}$$

$$\begin{aligned}
& \textit{Final}(D \uparrow): \\
& l_3^\downarrow + l_{15}^\downarrow + l_8^\downarrow + l_7^\uparrow \leq t_{CK \rightarrow Q \uparrow}^{D \uparrow, WEN \downarrow} \leq u_3^\downarrow + u_{15}^\downarrow + u_8^\downarrow + u_7^\uparrow
\end{aligned}$$

In the case of a falling edge of  $D$ , the propagation of  $D$  induces a falling edge of  $q_1$ , then of  $Q$ . Accordingly  $\textit{Assumption}(D \downarrow)$  (resp.  $\textit{Final}(D \downarrow)$ ) can be deduced from  $\textit{Assumption}(D \uparrow)$  (resp.  $\textit{Final}(D \uparrow)$ ) by changing each  $l_i^\uparrow$  (resp.  $u_i^\uparrow$ ) of  $\textit{Assumption}(D \uparrow)$  (resp.  $\textit{Final}(D \uparrow)$ ) into  $l_i^\downarrow$  (resp.  $u_i^\downarrow$ ).

From  $\textit{Final}(D \uparrow)$  and  $\textit{Final}(D \downarrow)$ , we infer the following constraint, guaranteeing property  $t_{CK \rightarrow Q}^{D, WEN \downarrow} \leq t_{max}^{write}$ :

$$\boxed{u_3^\downarrow + u_{15}^\downarrow + u_8^\downarrow + \max\{u_7^\uparrow, u_7^\downarrow\} \leq t_{max}^{write}.} \quad (*)$$

Constraints of  $\textit{Assumption}(D \uparrow)$  and  $\textit{Assumption}(D \downarrow)$  can be used to determine lower and upper bounds for  $t_{setup_D}$ :

$$\begin{aligned}
& \max\{u_0^\uparrow + u_1^\uparrow + u_5^\uparrow - l_8^\downarrow - l_3^\downarrow - l_{15}^\downarrow, u_0^\downarrow + u_1^\downarrow + u_5^\downarrow - l_8^\downarrow - l_3^\downarrow - l_{15}^\downarrow\} \leq t_{setup_D} \\
& \wedge t_{setup_D} \leq \min\{l_0^\uparrow + l_1^\uparrow + l_5^\uparrow - u_3^\downarrow - u_{15}^\downarrow, l_0^\downarrow + l_1^\downarrow + l_5^\downarrow - u_3^\downarrow - u_{15}^\downarrow, t_{LO} + l_0^\uparrow - u_2^\downarrow - u_3^\downarrow\}
\end{aligned}$$

We thus infer an *optimal* value of  $t_{setup_D}$ , denoted by  $t_{setup_D}^{opt}$ , which corresponds to its lower bound<sup>8</sup>:

$$t_{setup_D}^{opt} = \max\{u_0^\uparrow + u_1^\uparrow + u_5^\uparrow, u_0^\downarrow + u_1^\downarrow + u_5^\downarrow\} - l_8^\downarrow - l_3^\downarrow - l_{15}^\downarrow$$

A similar expression can be obtained for the optimal value of  $t_{setup_{WEN}}$ .

### 4.3 Application to Instance SP1

The above sets of constraints now allow us to give a formal justification of the correctness of the instance SP1. The values of the datasheet are (in tens of picoseconds):

$$t_{HI} = 36, t_{LO} = 74, t_{setup_D} = 108, t_{setup_{WEN}} = 48, t_{max}^{write} = 56.$$

The internal delays are (in tens of picoseconds):

	$[l_i^\uparrow, u_i^\uparrow]$	$[l_i^\downarrow, u_i^\downarrow]$
$(l_0, u_0)$	(95, 95)	(66, 66)
$(l_1, u_1)$	(14, 14)	(18, 18)
$(l_2, u_2)$	(23, 30)	(23, 30)
$(l_3, u_3)$	(5, 5)	(2, 2)
$(l_5, u_5)$	(22, 22)	(45, 45)
$(l_7, u_7)$	(21, 21)	(20, 20)
$(l_8, u_8)$	(0, 0)	(22, 22)
$(l_{13}, u_{13})$	(11, 11)	(8, 8)
$(l_{14}, u_{14})$	(21, 22)	(21, 22)
$(l_{15}, u_{15})$	(14, 14)	(11, 11)
$(l_{16}, u_{16})$	(24, 24)	(0, 0)

We check that all the constraints of *Assumption*( $D \uparrow$ ) and *Assumption*( $D \downarrow$ ) are satisfied. We also check constraint (\*). This shows that SP1 satisfies  $t_{CK \rightarrow Q}^{D, WEN \uparrow} \leq t_{max}^{write}$ . Furthermore, we find the value 96 for  $t_{setup_D}^{opt}$ , which matches very well with the optimal value 95 found by simulation by the designer.

### 4.4 Application to Instance SP2

As mentioned above, the values of the datasheet and internal delays of SP1 have been used in the refinement process in order to derive the appropriate set of generic constraints. Therefore, the derived sets of constraints *Assumption* and *Final* of Sect. 4.2 has not been produced independently from SP1, and the correctness of SP1 has been checked *a posteriori*. The constraints are however available now once for all, and can be reused to check immediately any other instance of SPSMALL. This is what has been done with instance SP2. The values of the datasheet are (in tens of picoseconds):

$$t_{HI} = 72, t_{LO} = 170, t_{setup_D} = 241, t_{setup_{WEN}} = 109, t_{max}^{write} = 142.$$

The internal delays are (in tens of picoseconds):

<sup>8</sup> Actually, we checked that  $t_{setup_D}^{opt}$  also satisfies constraints, coming from other parts of the circuit, which are not limitative in the case of instances SP1 and SP2.

	$[l_i^\uparrow, u_i^\uparrow]$	$[l_i^\downarrow, u_i^\downarrow]$
$(l_0, u_0)$	(197, 197)	(140, 140)
$(l_1, u_1)$	(60, 60)	(58, 58)
$(l_2, u_2)$	(66, 66)	(43, 43)
$(l_3, u_3)$	(8, 8)	(4, 4)
$(l_5, u_5)$	(61, 61)	(63, 63)
$(l_7, u_7)$	(47, 47)	(52, 52)
$(l_8, u_8)$	(0, 0)	(42, 42)
$(l_{13}, u_{13})$	(23, 23)	(23, 23)
$(l_{14}, u_{14})$	(35, 35)	(36, 36)
$(l_{15}, u_{15})$	(56, 56)	(43, 43)
$(l_{16}, u_{16})$	(24, 24)	(0, 0)

As in the case of *SP1*, we check that *SP2* satisfies  $t_{CK \rightarrow Q}^{D,WEN} \leq t_{max}^{write}$ . Besides, we find the value 229 for  $t_{setup_D}^{opt}$ , which matches exactly with the optimal value found by electrical simulation by the designer.

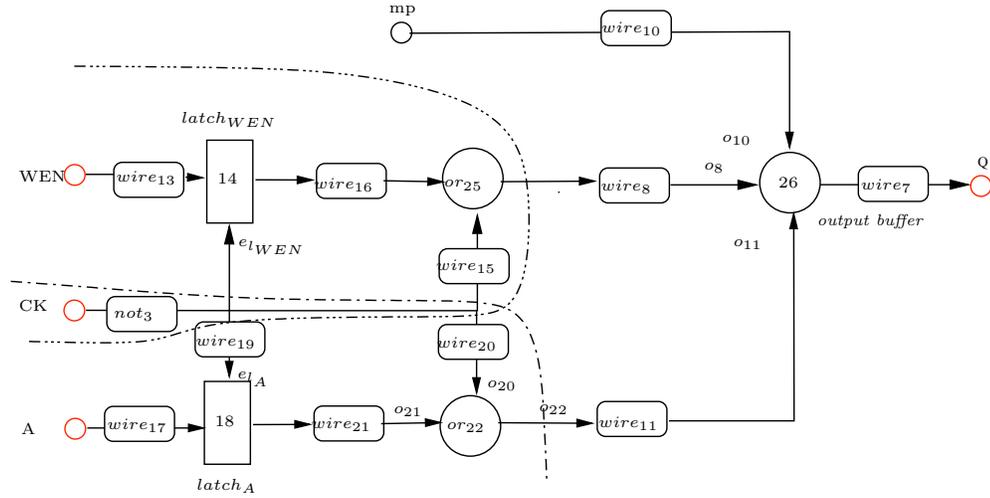
#### 4.5 Interpretation of *Assumption* and *Final* constraints

Let us explain how certain constraints of *Assumption* and *Final* can be interpreted. As mentioned above, the constraints of *Final* is of the form  $f(l, u) \leq t_{CK \rightarrow Q}^{D,WEN \downarrow} \leq g(l, u)$ , where  $g(l, u)$  is a sum of internal delays (viz.  $u_3^\downarrow + u_{15}^\downarrow + u_8^\downarrow + \max\{u_7^\uparrow, u_7^\downarrow\}$ ), and similarly for  $f(l, u)$ . This expression corresponds to the delay associated to the path which starts at input *CK*, goes through inverter *not3*, then wires *wire15*, *wire8*, *wire7*, and ends at output port *Q*. Such a path is *critical*, as it corresponds to the longest traversal from an input to the output port *Q*, thus conditioning the end-to-end delay to *Q*. This feature is guaranteed by the set of constraints *Assumption*, which gives conditions, among others, enquiring the other paths to be shorter. More precisely, the constraints in the set *Assumption* state:

- the order along which signals arrive at junction points (logical gates or blocks). For instance, inequality  $u_5^\uparrow + u_0^\uparrow + u_1^\uparrow \leq l_8^\downarrow + l_3^\downarrow + l_{15}^\downarrow + t_{setup_D}$  says that, at junction point *output buffer*, the input  $o_5$  (issued from a rising edge of *D*, occurring  $t_{setup_D}$  before the rising edge of *CK*, and following path *wire0-latchD-wire5*) arrives before input  $o_8$  (issued from a rising edge of *CK*, and following path *not3-wire15-wire8*); in other words, this inequality says that the data to be written in the output buffer arrives before the writing command.
- the stability conditions sufficient to ensure that an input signal correctly propagates throughout a component. For instance, inequality  $u_0^\uparrow + u_1^\uparrow \leq t_{setup_D} + l_2^\downarrow + l_3^\downarrow$  says that input  $d_1$  (issued from the rising edge of *D*, occurring  $t_{setup_D}$  before the rising edge of *CK*) arrives at least  $u_1$  before input  $e_{l_D}$  of *latchD* (issued from the rising edge of *CK*, and following path *not3-wire2*); in other words, the inequality says that the data to be latched arrives early enough before the latch closes.

## 5 Verification of the Read Property

In this Section, we focus on the *read* operation (instead of *write*). The relevant portion of the circuit (in case of a falling edge of  $A$ ) is depicted on Fig. 8. Recall that we assume that the port of  $A$  is reduced to a single bit. Furthermore, the  $m \times n$  storage array is abstracted as a unique memory point  $mp$ . As mentioned before, the operations involving the memory points may take different times, according to the physical location of the memory point. We will assume that  $mp$  corresponds to the memory point taking the *longest* time (i.e., the one corresponding to the “slowest”  $wire_{e_{10}}$ )<sup>9</sup>. Moreover, input  $D$  is not relevant for this property.



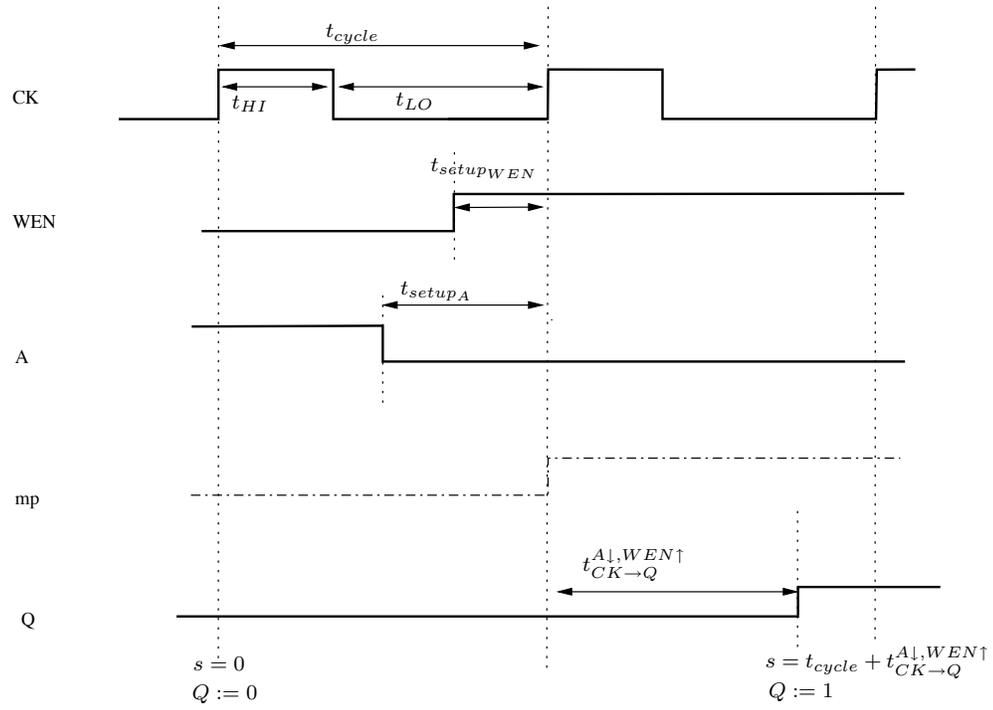
**Fig. 8.** Schematics of the relevant portion of SPSMALL for read.

The “response time” property for the read operation now expresses as follows:  
 $t_{CK \rightarrow Q}^{A \downarrow, WEN \uparrow} \leq t_{max}^{read}$  and  $t_{CK \rightarrow Q}^{A \uparrow, WEN \uparrow} \leq t_{max}^{read}$ .

We now focus on the case of a falling edge of  $A$ . (A similar approach applies for the rising edge case.) The timing parameters involved are depicted on Fig. 9.

As for the write property, the schematics decomposes itself into three subparts (delimited by the dashed line on Fig. 8). Likewise, the analysis is performed on each subpart, and the associated constraints recombined together. The resulting constraints (besides  $\{l_i^\downarrow \leq u_i^\downarrow\}$  and  $\{l_i^\uparrow \leq u_i^\uparrow\}$ ) are:

<sup>9</sup> Such a memory point generally corresponds to an “extreme” point of the topology of the memory array, and can be identified by electrical simulation.



**Fig. 9.** Timing diagram for the *read* operation (considering  $mp = 1$  at the beginning of the cycle of the read operation).

*Assumption*( $A \downarrow$ ):

$$\begin{aligned}
& \max\{u_{10}^\uparrow, u_{10}^\downarrow\} < l_3^\downarrow + l_{20}^\downarrow + l_{11}^\downarrow \\
& \wedge t_{LO} + l_{17}^\downarrow > u_3^\downarrow + u_{19}^\downarrow + t_{setup_A} \\
& \wedge u_{17}^\downarrow + u_{18}^\downarrow < t_{setup_A} + l_3^\downarrow + l_{19}^\downarrow \\
& \wedge u_{18}^\downarrow < t_{LO} \\
& \wedge u_{17}^\downarrow + u_{18}^\downarrow + u_{21}^\downarrow < t_{setup_A} + l_3^\downarrow + l_{20}^\downarrow \\
& \wedge t_{LO} + l_{13}^\downarrow > u_3^\downarrow + t_{setup_{WEN}} \\
& \wedge u_{13}^\downarrow + u_{14}^\downarrow < t_{setup_{WEN}} + l_3^\downarrow \\
& \wedge u_{14}^\downarrow < t_{HI} \\
& \wedge u_{13}^\downarrow + u_{14}^\downarrow + u_{16}^\downarrow < t_{setup_{WEN}} + l_3^\downarrow + l_{15}^\downarrow
\end{aligned}$$

*Final*( $A \downarrow$ ):

$$l_3^\downarrow + l_{20}^\downarrow + l_{11}^\downarrow + \max\{l_7^\downarrow, l_7^\uparrow\} \leq t_{CK \rightarrow Q}^{A \downarrow, WEN \uparrow} \leq u_3^\downarrow + u_{20}^\downarrow + u_{11}^\downarrow + \max\{u_7^\downarrow, u_7^\uparrow\}$$

Similarly, one can infer sets *Assumption*( $A \uparrow$ ) and *Final*( $A \uparrow$ ).

From *Final*( $A \downarrow$ ) and *Final*( $A \uparrow$ ), we infer the following constraint, guaranteeing property  $t_{CK \rightarrow Q}^{A, WEN \uparrow} \leq t_{max}^{read}$ :

$$\boxed{u_3^\downarrow + u_{20}^\downarrow + u_{11}^\downarrow + \max\{u_7^\uparrow, u_7^\downarrow\} \leq t_{max}^{read}} \quad (**)$$

Constraints of *Assumption*( $A \downarrow$ ) and *Assumption*( $A \uparrow$ ) can be used to infer an *optimal* value of  $t_{setup_A}$ , denoted by  $t_{setup_A}^{opt}$ <sup>10</sup>:

$$\boxed{t_{setup_A}^{opt} = \max\{\max\{u_{17}^\downarrow + u_{18}^\downarrow + u_{21}^\downarrow, u_{17}^\uparrow + u_{18}^\uparrow + u_{21}^\uparrow\} - l_3^\downarrow - l_{20}^\downarrow, \max\{u_{17}^\downarrow + u_{18}^\downarrow, u_{17}^\uparrow + u_{18}^\uparrow\} - l_3^\downarrow - l_{19}^\downarrow\}}$$

As in Sect. 4.5, the constraints of *Final* correspond to the delays associated with the critical path: it is issued from *CK*, goes through *not*<sub>3</sub>, *wire*<sub>20</sub>, *wire*<sub>11</sub> and *wire*<sub>7</sub>. Likewise, many constraints of *Assumption* can be viewed as sufficient conditions either for the other paths to be shorter than the critical one, or for the input signals to propagate through all the components to *Q*.

**Verification of *SP1*.** The relevant values of the datasheet *SP1* (in tens of picoseconds) :

$$t_{HI} = 36, t_{LO} = 74, t_{setup_A} = 58, t_{max}^{read} = 77$$

The internal delays are (in tens of picoseconds):

<sup>10</sup> Actually, we checked that  $t_{setup_A}^{opt}$  also satisfies constraints, coming from other parts of the circuit, which are not limitative in the case of instances *SP1* and *SP2*.

	$[l_i^\uparrow, u_i^\uparrow]$	$[l_i^\downarrow, u_i^\downarrow]$
$(l_3, u_3)$	(5, 5)	(2, 2)
$(l_7, u_7)$	(21, 21)	(20, 20)
$(l_{10}, u_{10})$	(6, 6)	(6, 6)
$(l_{11}, u_{11})$	(23, 23)	(23, 23)
$(l_{13}, u_{13})$	(11, 11)	(8, 8)
$(l_{14}, u_{14})$	(21, 22)	(21, 22)
$(l_{16}, u_{16})$	(24, 24)	(0, 0)
$(l_{17}, u_{17})$	(16, 16)	(15, 15)
$(l_{18}, u_{18})$	(31, 31)	(31, 31)
$(l_{19}, u_{19})$	(15, 15)	(12, 12)
$(l_{20}, u_{20})$	(40, 40)	(29, 29)
$(l_{21}, u_{21})$	(0, 0)	(0, 0)

We check that all the constraints of *Assumption*( $A \downarrow$ ) and *Assumption*( $A \uparrow$ ) are satisfied. We also check constraint (\*\*). This shows that *SP1* satisfies  $t_{CK \rightarrow Q}^{A, WEN \uparrow} \leq t_{max}^{read}$ . Furthermore, we find the value 30 for  $t_{setup_A}^{opt}$ , which matches exactly with the optimal value found by simulation by the designer.

**Verification of *SP2*.** For *SP2*, the relevant values of the datasheet are:  $t_{HI} = 72, t_{LO} = 170, t_{setup_A} = 110, t_{max}^{read} = 169$ . The internal delays are (in tens of picoseconds):

	$[l_i^\uparrow, u_i^\uparrow]$	$[l_i^\downarrow, u_i^\downarrow]$
$(l_3, u_3)$	(8, 8)	(4, 4)
$(l_7, u_7)$	(47, 47)	(52, 52)
$(l_{10}, u_{10})$	(14, 14)	(10, 10)
$(l_{11}, u_{11})$	(51, 51)	(55, 55)
$(l_{13}, u_{13})$	(23, 23)	(23, 23)
$(l_{14}, u_{14})$	(35, 35)	(36, 36)
$(l_{16}, u_{16})$	(24, 24)	(0, 0)
$(l_{17}, u_{17})$	(14, 14)	(9, 9)
$(l_{18}, u_{18})$	(79, 79)	(89, 89)
$(l_{19}, u_{19})$	(36, 36)	(22, 22)
$(l_{20}, u_{20})$	(74, 74)	(58, 58)
$(l_{21}, u_{21})$	(0, 0)	(0, 0)

As in the case of *SP1*, we check that *SP2* satisfies  $t_{CK \rightarrow Q}^{A, WEN \uparrow} \leq t_{max}^{read}$ . Besides, we find the value 75 for  $t_{setup_A}^{opt}$ , which matches well with the optimal value (74) found by electrical simulation by the designer.

## 6 Final Remarks

We have shown in this paper how to apply parametrized methods to verify timed properties of the generic architecture of a memory. We have thus found certain

sufficient conditions (under the form of linear inequalities between parameters) that ensure that the response times to a write command or a read command lie between certain bounds, and have checked these properties on an instance *SP1* of the memory. These linear inequalities have been also used to derive the optimal values of setup timings of input signals (*viz.*, setup timings for *D* and *A*). Such an analysis can be immediately applied to the verification of other instances of the SPSMALL memory, as exemplified here on instance *SP2*.

Other properties concerning the writing into the memory points, and concerning the “hold timings” of input signals (minimal stability duration after the rising edge of the clock) can be proved similarly, for the completion of the verification. Note that such properties are considered by the designer as less sensitive for the SPSMALL architecture.

Let us also point out that our verification process does not get rid completely of results given by simulation, since it makes use of values of *elementary* delays  $[l^\downarrow, u^\downarrow]$  and  $[l^\uparrow, u^\uparrow]$  for the internal components. This is in contrast with the traditional simulation process, which computes the values of the *end-to-end* propagation delays setting up the datasheet.

Our method has exploited the data available for instance *SP1* (datasheet and internal delay intervals), especially during the stepwise refinement process, when we took the refutation of selected constraints (the “suspect” ones). By negating such constraints, we have actually focused on a certain class of the circuit, disregarding other possible choices of implementation. In fact, the major outcome of our verification process is a general and formal description of the architecture principles along which SPSMALL has been designed. We have shown that these principles are correctly implemented by two different instances of the memory.

## References

1. HSIM Simulator Description. In <http://www.synopsys.com/products/>.
2. TLL Transistor Abstraction Tool description. In <http://www.transeda.com/products/>.
3. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science* 126, pages 183–235, 1994.
4. M. Baclet and R. Chevallier. Timed verification of the SPSMALL memory. In *1th International Conference Memory Technology and Development*, pages 1–2, Giens France, 2005.
5. M. Bozga, H. Jianmin, O. Maler, and S. Yovine. Verification of asynchronous circuits using timed automata. In *TPTS'02, ENTCS vol 65*, 2002.
6. J.A. Brzozowski and C-J.H. Seger. *Asynchronous Circuits*. Springer, 1994.
7. R. Chevallier, E. Encrenaz-Tiphène, L. Fribourg, and W. Xu. Timing analysis of an embedded memory: SPSMALL. In *10th WSEAS International Conference on Circuits, Athens, Greece*, 2006.

8. R. Clariso and J. Cortadella. The octahedron abstract domain. In *Proc. 11th Static Analysis Symposium (SAS), LNCS 3148, Springer, pp. 312-327, 2004.*
9. R. Clariso and J. Cortadella. Verification of timed circuits with symbolic delays. In *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 628-633, 2004.*
10. P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL, pp. 238-252, 1977.*
11. D. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Automatic Verification Methods for Finite State Systems, LNCS 407, Springer, 1989.*
12. N. Halbwachs. Delay analysis in synchronous programs. In *CAV'93, LNCS 697, Springer, pp. 333-346, 1993.*
13. T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. A User Guide to HYTECH. In *TACAS'95, LNCS 1019, Springer, pp.41-71, 1995.*
14. K. Larsen, P. Pettersson, and W. Yi. UPPAAL in a Nutshell. *International Journal on Software Tools for Technology Transfer*, 1:134–152, 1997.
15. O. Maler and A. Pnueli. Timing analysis of asynchronous circuits using timed automata. In *CHARME'95, LNCS 987, Springer, pp.189-205, 1995.*
16. D.A. Patterson and J.L. Hennessy. *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann, 1990.
17. R. Ben Salah, M. Bozga, and O. Maler. On timing analysis of combinational circuits. In *FORMATS'03, LNCS 2791, Springer, pp.204-219, 2003.*
18. S.Yovine. KRONOS: A verification tool for real-time systems. *International Journal on Software Tools for Technology Transfer*, 1:123–133, 1997.

## Appendix 0. Memory Storage mechanism

Fig. 10 gives a closer view of the memory storage mechanism. Thick lines in the figure represent the reduced number of paths on which our analysis focus on.

## Appendix 1. Computation of $t_{CK \rightarrow q_1}^D$

The experiment propagates an edge (either rising or falling) from D through  $q_1$ . Corresponding schematics is depicted on Fig. 11. Timing parameters are depicted on Fig. 12.

Applying the constraint extraction procedure of section 3.3, we obtain, in the case of a rising edge of  $D$  ( $D \uparrow$ ):

*Assumption:*

$$\begin{aligned}
 u_0^\uparrow + u_1^\uparrow &\leq t_{setup_D} + l_2^\downarrow + l_3^\downarrow \\
 \wedge u_2^\downarrow + u_3^\downarrow + u_1^\uparrow &< t_{LO} \\
 \wedge t_{setup_D} + u_2^\downarrow + u_3^\downarrow &< l_0^\uparrow + t_{LO}
 \end{aligned}$$

*Final:*

$$t_{HI} + t_{LO} - t_{setup_D} + l_0^\uparrow + l_1^\uparrow \leq t_{CK \rightarrow q_1}^{D\uparrow} \leq t_{HI} + t_{LO} - t_{setup_D} + u_0^\uparrow + u_1^\uparrow$$

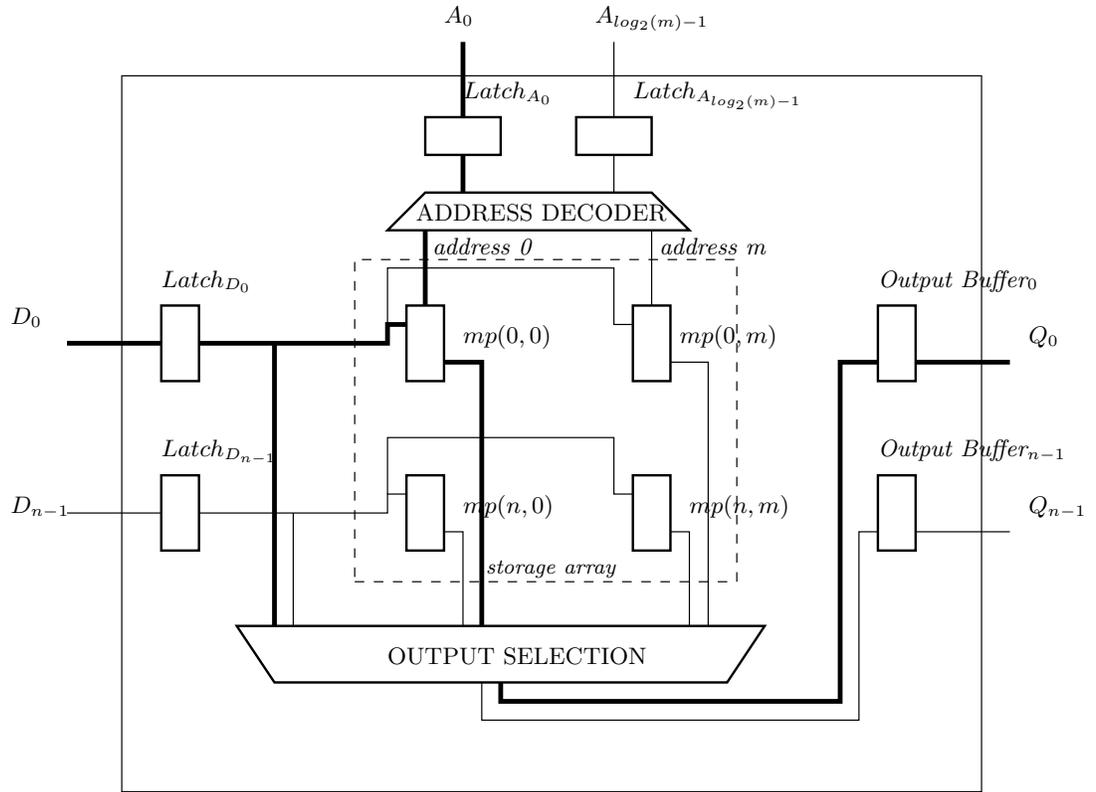


Fig. 10. Schematic view of parallel paths of  $D$ ,  $A$  and  $Q$ .

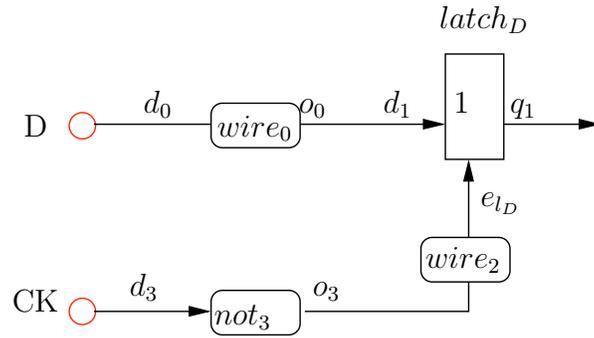
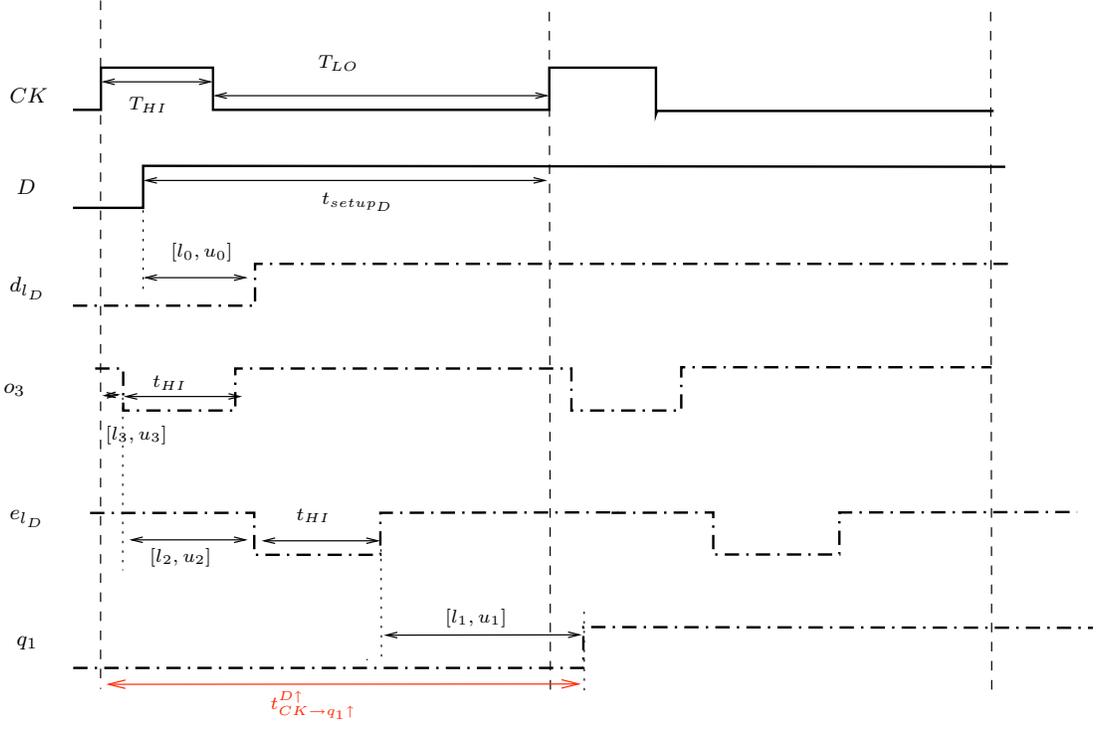


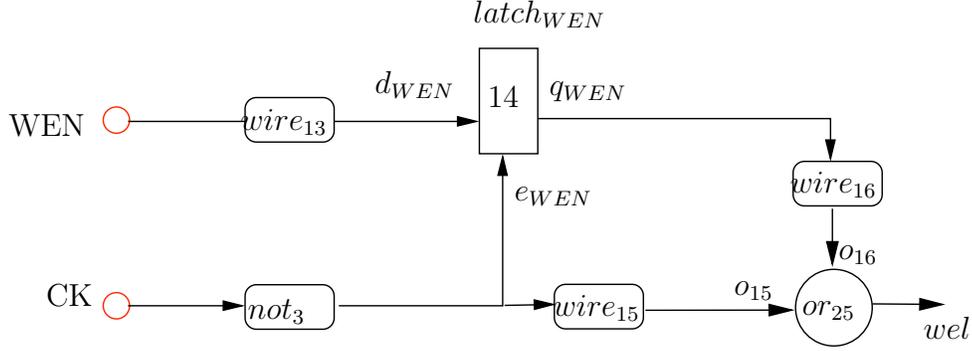
Fig. 11. Components for the computation of  $t_{CK \rightarrow q_1}^D$ .



**Fig. 12.** Timing parameters for the computation of  $t_{CK \rightarrow q_1}^{D\uparrow}$ .

## Appendix 2: Computation of $t_{CK \rightarrow wel}^{WEN \downarrow}$

This experiment gives analytical expressions of lower and upper bounds of  $t_{CK \rightarrow wel}^{WEN \downarrow}$ . It consists in propagating a falling edge from input port  $WEN$  up to internal signal  $wel$ . The schematics of the components is described on Fig. 13.



**Fig. 13.** Components involved in the computation of  $t_{CK \rightarrow wel}^{WEN \downarrow}$ .

The timing parameters are defined on Fig. 14.

Applying the constraint extraction procedure of section 3.3, we obtain:

*Assumption:*

$$\begin{aligned} t_{LO} + l_{13}^{\downarrow} &> u_3^{\downarrow} + t_{setup_{WEN}} \\ \wedge u_{13}^{\downarrow} + u_{14}^{\downarrow} &< t_{setup_{WEN}} + l_3^{\downarrow} \\ \wedge u_{14}^{\downarrow} &< t_{HI} \\ \wedge u_{13}^{\downarrow} + u_{14}^{\downarrow} + u_{16}^{\downarrow} &< t_{setup_{WEN}} + l_3^{\downarrow} + l_{15}^{\downarrow} \end{aligned}$$

*Final:*

$$l_3^{\downarrow} + l_{15}^{\downarrow} \leq t_{CK \rightarrow wel}^{WEN \downarrow} \leq u_3^{\downarrow} + u_{15}^{\downarrow}$$

## Appendix 3. Computation of $t_{CK \rightarrow Q}^{q_1, wel \downarrow}$

The experiment propagates a rising (resp. falling) edge from  $q_1$ , occurring at instant  $t_{CK \rightarrow q_1}^{D \uparrow}$  (resp.  $t_{CK \rightarrow q_1}^{D \downarrow}$ ), up to the output port  $Q$ . Signal  $wel$  is abstracted as a falling edge occurring at instant  $t_{CK \rightarrow wel}^{WEN \downarrow}$ . Corresponding schematics is depicted on Fig. 15.

We obtain the following sets of constraints :

*Assumption:*

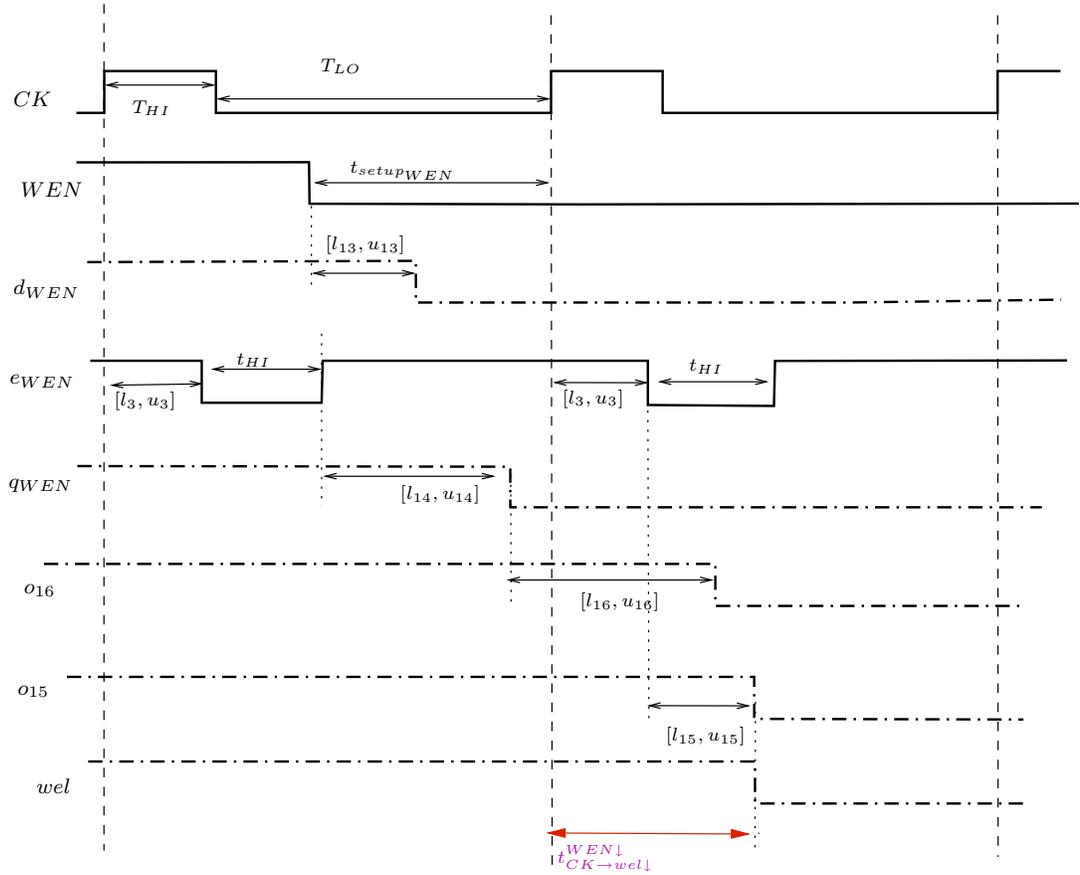


Fig. 14. Timing parameters for  $t_{CK \rightarrow wel \downarrow}^{WEN \downarrow}$ .

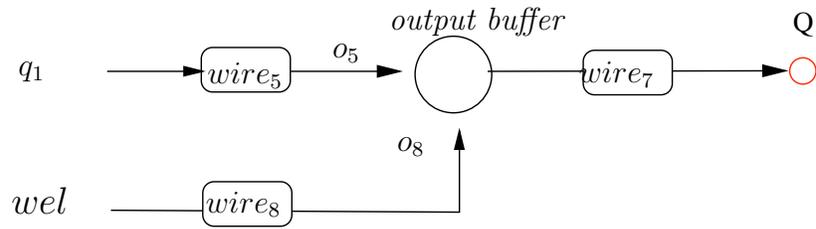


Fig. 15. Computation of  $t_{CK \rightarrow Q \uparrow}^{q_1 \uparrow, wel \downarrow}$ .

$$\begin{aligned}
t_{HI} + t_{LO} + t_{CK \rightarrow wel \downarrow}^{WEN \downarrow} &\leq l_5^\uparrow + t_{CK \rightarrow q_1 \uparrow}^{D \uparrow} \\
\wedge u_5^\uparrow + t_{CK \rightarrow q_1 \uparrow}^{D \uparrow} &\leq l_8^\downarrow + t_{CK \rightarrow wel \downarrow}^{WEN \downarrow} + t_{HI} + t_{LO}
\end{aligned}$$

The bounds for  $t_{CK \rightarrow Q \uparrow}^{q_1 \uparrow, wel \downarrow}$  in *Final*:

$$l_8^\downarrow + t_{CK \rightarrow wel \downarrow}^{WEN \downarrow} + l_7^\uparrow + t_{HI} + t_{LO} \leq t_{CK \rightarrow Q \uparrow}^{q_1 \uparrow, wel \downarrow} \leq u_8^\downarrow + t_{CK \rightarrow wel \downarrow}^{WEN \downarrow} + u_7^\uparrow + t_{HI} + t_{LO}$$