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

R. Chevallier, E. Encrenaz, L. Fribourg, W. Xu

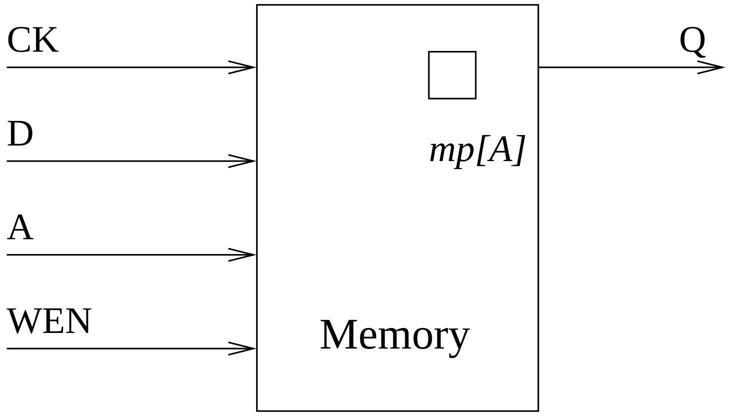
Partially supported by MEDEA+ BLUBERRIES project.
Originally presented at FORMATS 06.

Outline

- SPSMALL Embedded Memory: Architecture and Properties
- Verification Method
- Parametric Analysis
- Application to two instances of SPSMALL

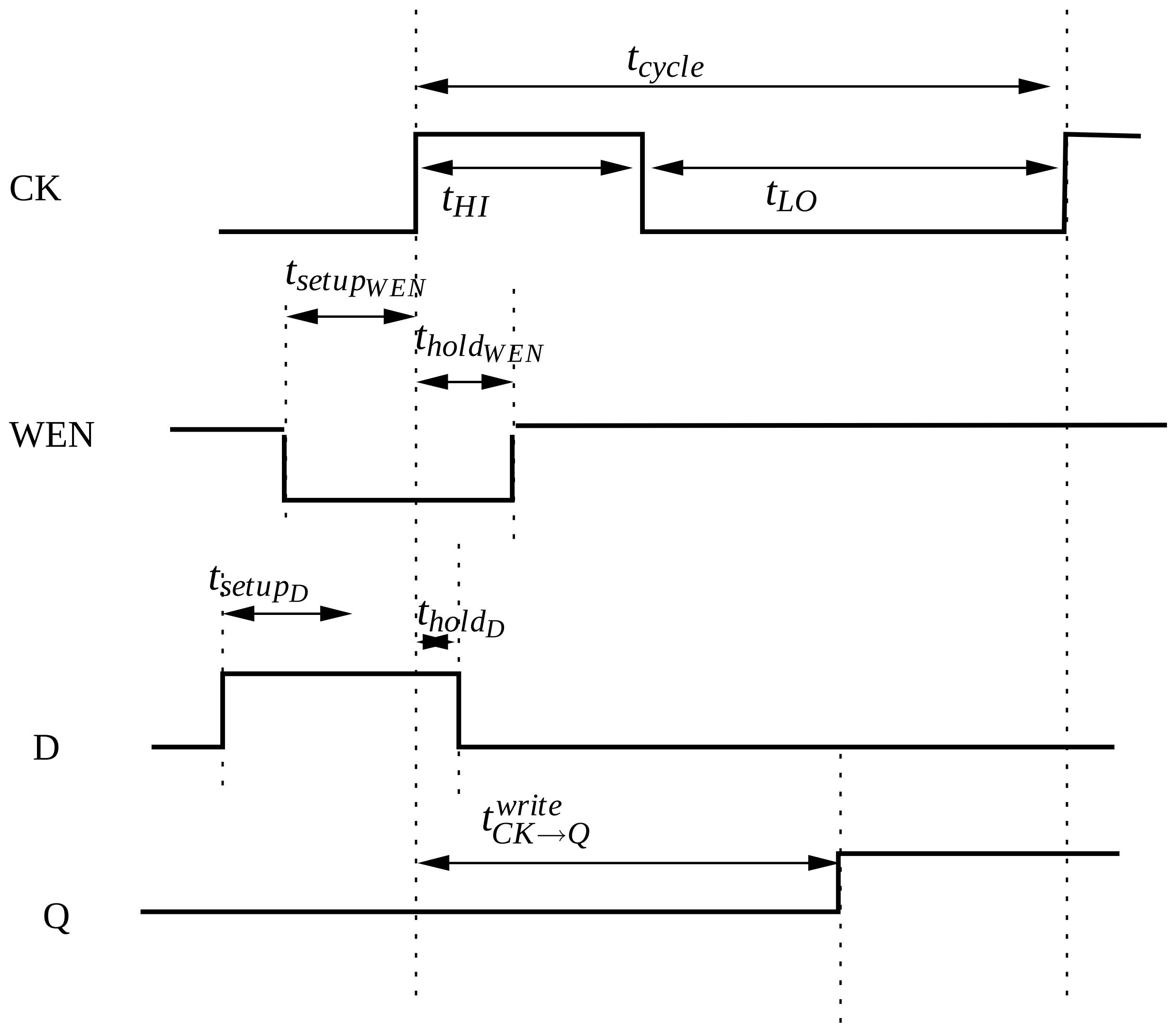
Memory's interface

Memory's purpose : To *store* data and to *supply* previously stored data.



- Array of memory points $mp[A]$ where data are stored
- interface signals :
  - ⇝ D : Data to be stored
  - ⇝ A : Address of selected memory point
  - ⇝ WEN : Operation to be performed *read* or *write*
  - ⇝ CK : Synchronous clock
  - ⇝ Q : Output (Data "given" by the memory)

Timing diagram of a *write* operation



Specification of the memory (*datasheet*)

The *datasheet* : a set of timing constraints provided by the manufacturer.

- ▶ Requierements for the environment.
  - $\leadsto$ global timings : t_{cycle}, t_{HI} and t_{LO}
  - $\leadsto$ stability timings : forall input signal i : t_{setup_i} and t_{hold_i}
- ▶ Guarantied *Nominal* end-to-end timings : t_{max}^{read} and t_{max}^{write}.

This set *heavily* depends on the technology in which the memory is implemented. It is usually computed by *electrical simulation* at transistor level.

Questions

- Can we formally compute the timings $t^{read}_{CK\rightarrow Q}$, $t^{write}_{CK\rightarrow Q}$ and $t_{CK\rightarrow mp}$ of a memory *at a higher level of abstraction* ?
- How are they related to the timings of the specification (t_{cycle}, t_{HI}, t_{LO}, t^{read}_{max}, t^{write}_{max}, and t_{setup_i} and t_{hold_i} forall input i) ?
- Can we *reduce* some t_{setup_i} timings while preserving the functionality ?

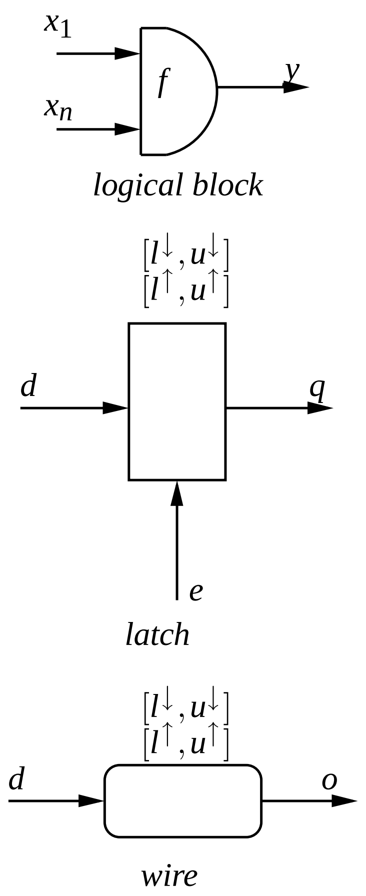
More precisely, we focus on properties of the form: $t^{write}_{CK\rightarrow Q} \leq t^{write}_{max}$

Our choices

- *Timings and logical functionality must be related.* differs from static analysis tools - [Lester-Dioury+ 00].
- Abstraction level : *The Latch level.* differs from [Baclet-Chevallier 04], [Clariso-Cortadella 04].
- Relating timing information to the logical model :
  - ⇝ *Extracted from transistor-level model or* ***parameters*****.**
  - ⇝ differs from [Baclet-Chevallier 04], [Bogza-Maler 03].
  - ⇝ particularization of [Clariso-Cortadella 04]'s parametric approach.

Abstract Functional and Timing Graph

The circuit is modelled by an Abstract Functional and Timing Graph.

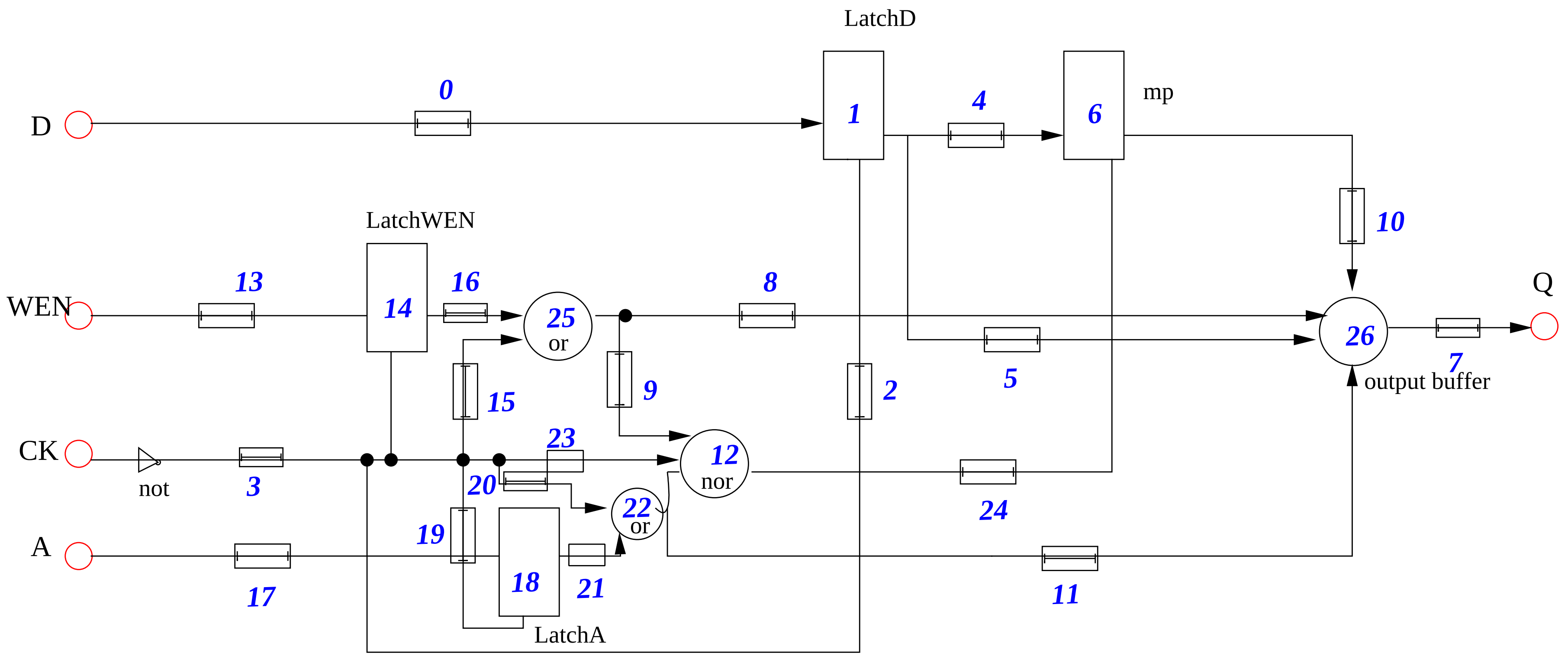


x_1
x_n
f
y
logical block
$[l^{\downarrow}, u^{\downarrow}]$
$[l^{\uparrow}, u^{\uparrow}]$
d
q
e
latch
$[l^{\downarrow}, u^{\downarrow}]$
$[l^{\uparrow}, u^{\uparrow}]$
d
o
wire

- node :
  - ⇝ logical block (a logical gate or a component realizing a logical function).
  - ⇝ latch (a “barrier” breaking the propagation flow of edges).
- arc : wire connection.
- delays $[l_i^{\downarrow}, u_i^{\downarrow}]$ and $[l_i^{\uparrow}, u_i^{\uparrow}]$ are assigned to each latch and wire i. delays of logical block are incorporated into input wires.

Abstract Functional and Timing Graph of SPSMALL

One memory location of one bit-width



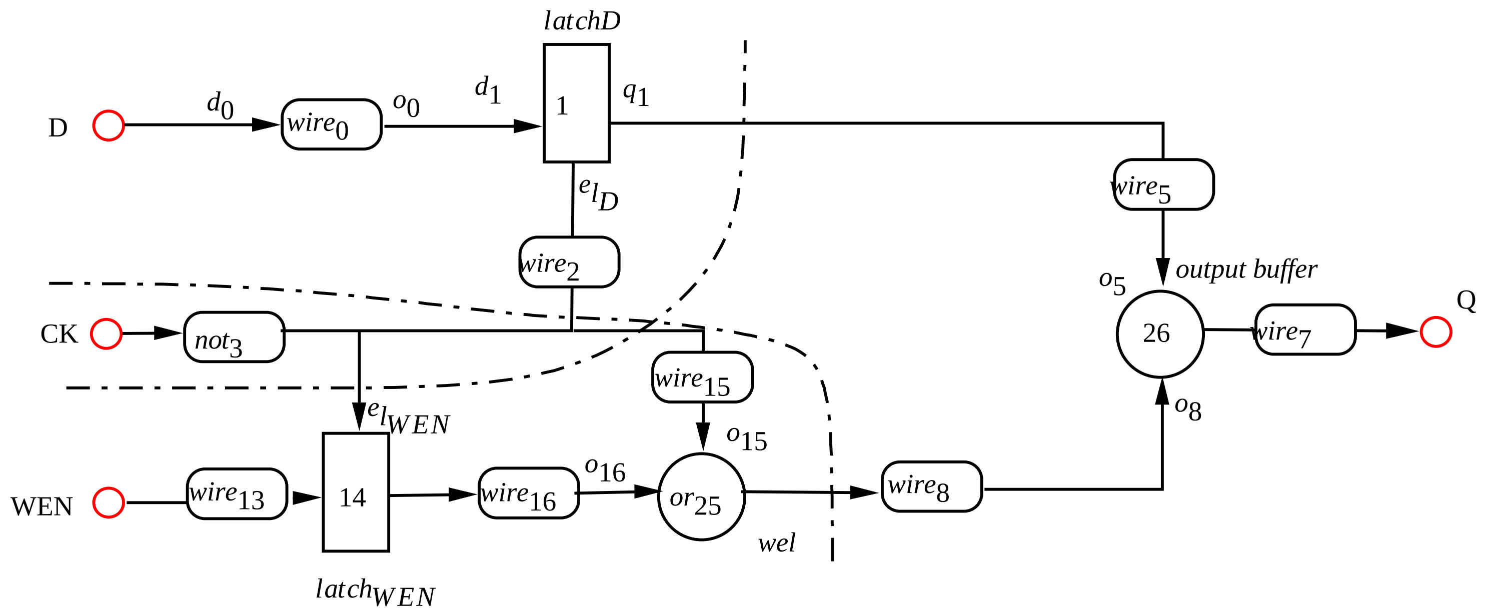
Verification method

Define an experiment : components involved, k number of cycles requiered, waveform of input signals, goal to reach.

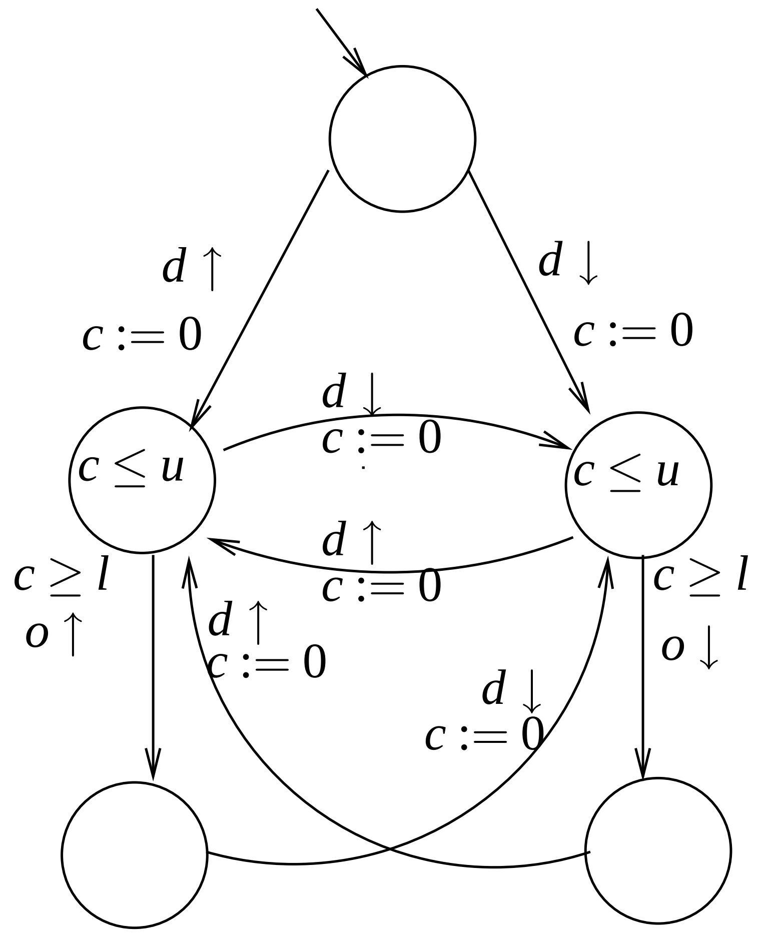
- ▶ Build G, the Abstract Functional and Timing graph of the relevant portion of the circuit,
- ▶ For each node n of G, build the timed automaton representing the functionality and delays of n,
- ▶ for each input signal, build a timed automaton mimicking its waveform along k cycles,
- ▶ Compute the product of timed automata,
- ▶ Extract timing constraints by stepwise refinement of the reachability set.

Verification of correct access time for a write operation

step 1. Building the AFTG.

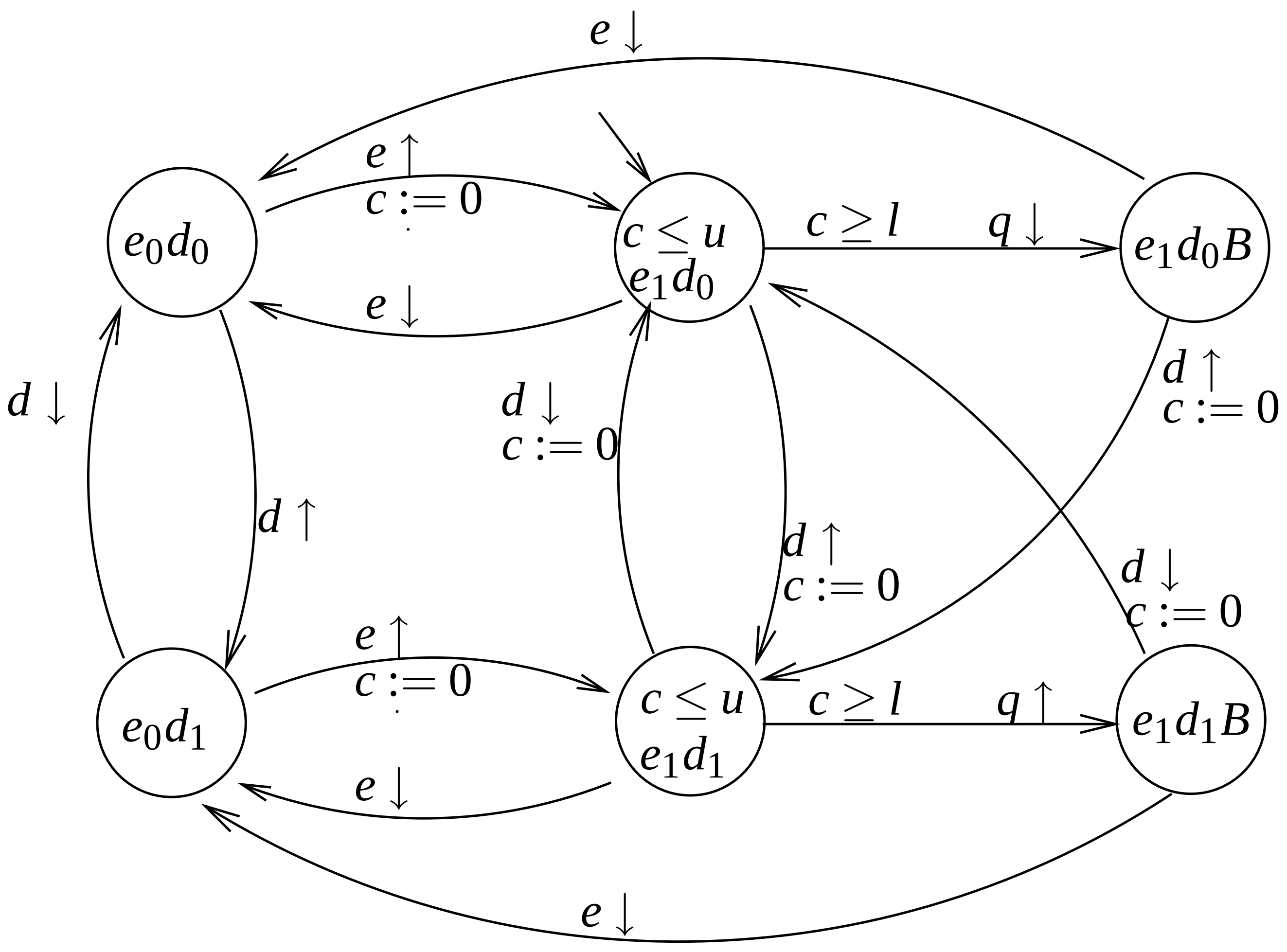


Timed automaton for a wire



A rising (resp. falling) edge is delayed by $\delta\in[l,u]$.

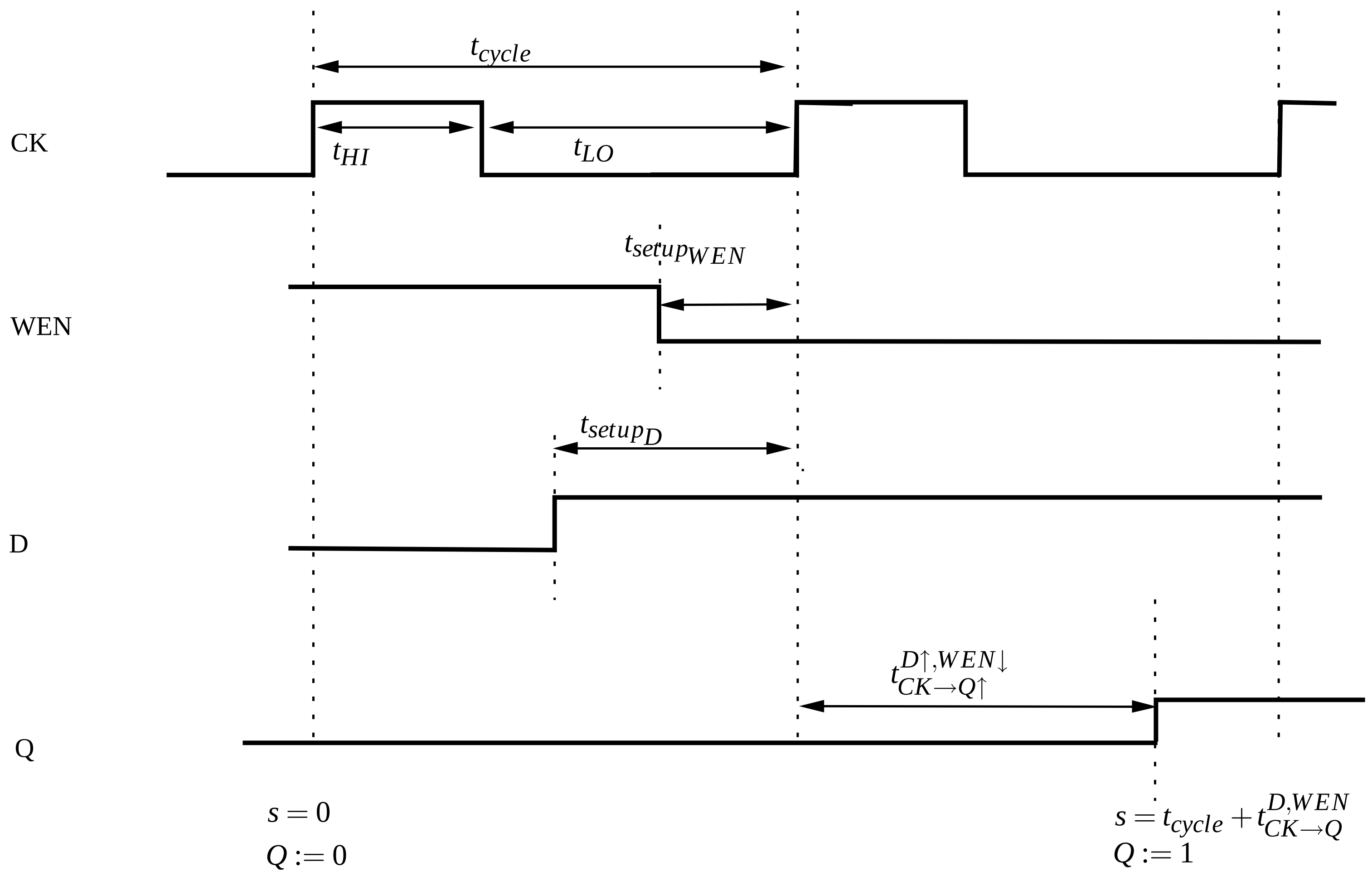
Timed automaton for a latch



A value on d is propagated up to the output with a delay $\delta \in [l, u]$ when e is high.

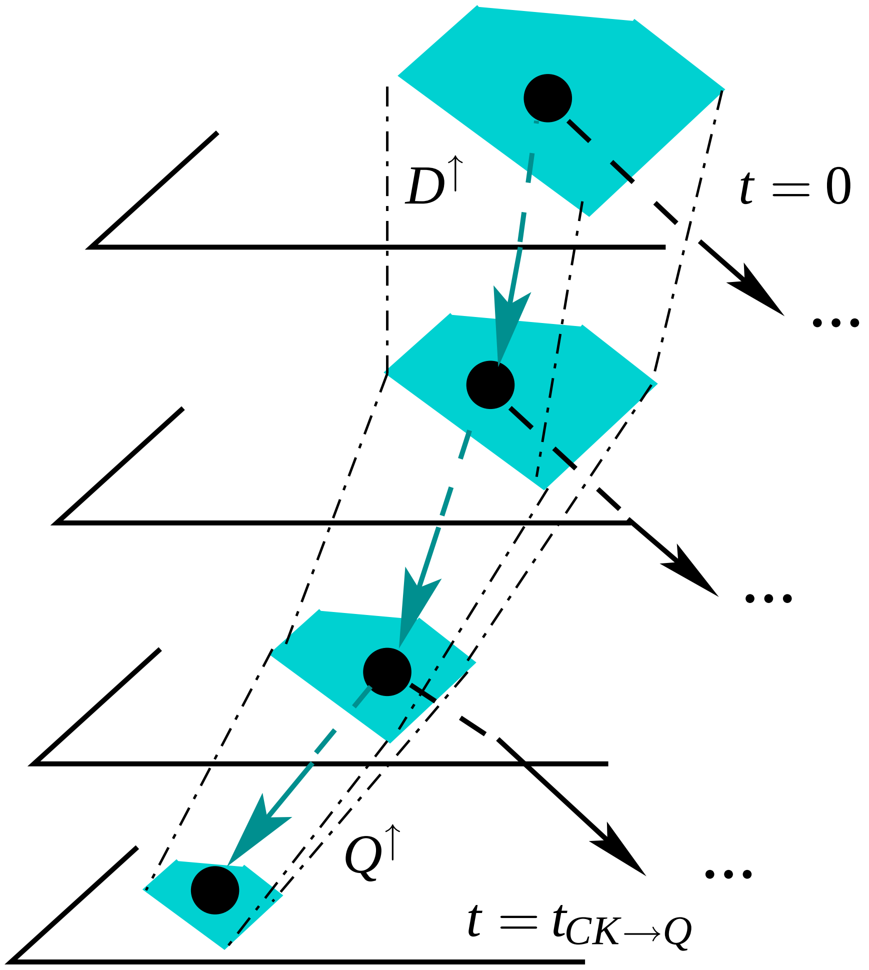
Verification of correct access time for a write operation

Step 2 : Number of cycles, waveform of input signals, goal to reach


t_{cycle}
CK
t_{HI}
t_{LO}
$t_{setup_{WEN}}$
WEN
t_{setup_D}
D
$t_{CK\rightarrow Q\uparrow}^{D\uparrow,WEN\downarrow}$
Q
$s = 0$
$Q := 0$
$s = t_{cycle} + t_{CK\rightarrow Q}^{D,WEN}$
$Q := 1$

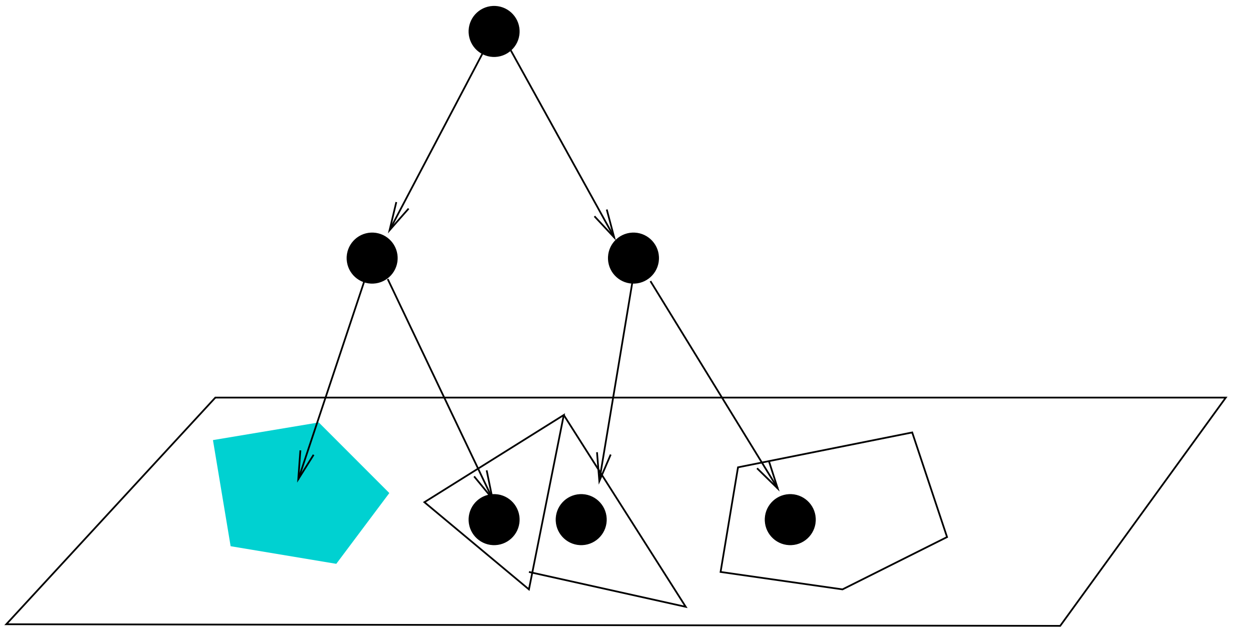
Computation of $Post*(Init)$ - 1

A trace represents *the history of choices* made on parameters constraints



Computation of $Post*(Init)$ - 2

The states at $t = 2.cycles$ represents the history of *all possible choices* made on parameters constraints (exponential).



Only some of them correspond to a *correct functioning* of the memory.

Computation of a set of constraints by stepwise refinement

1. $Assumption \leftarrow \top$
2. Compute $Post^*(Init \cap Assumption)$ up to the end of cycle k.
3. Select a bad state in $Final = Post^*(Init \cap Assumption) \cap \{t = k\ cycles\}$.
4. Detect a wrong subconstraint in this bad state (irrealistic or numerically wrong),
5. Strengthen $Assumption$, by adding the negation of the wrong subconstraint. goto 2.

Finally :
$Final$ contains a set of states at the end of k cycles that are functionally correct, and $Assumption$ is a set of constraints that guaranty the convergence up to $Final$ states.

Obtained constraints in *Final*

They bound the response time ($t^{write}_{CK \to Q}$) of the circuit.

$$t^{write}_{CK \to Q} \leq u_3^{\downarrow} + u_{15}^{\downarrow} + u_8^{\downarrow} + \max\{u_7^{\downarrow}, u_7^{\uparrow}\}$$

Hence, $t^{write}_{CK \to Q} \leq t^{write}_{max}$ is guarantied iff :

$$u_3^{\downarrow} + u_{15}^{\downarrow} + u_8^{\downarrow} + \max\{u_7^{\downarrow}, u_7^{\uparrow}\} \leq t^{write}_{max}$$

Obtained constraints in *Assumption*

In case of a rising edge of D ($D^{\uparrow}$):

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

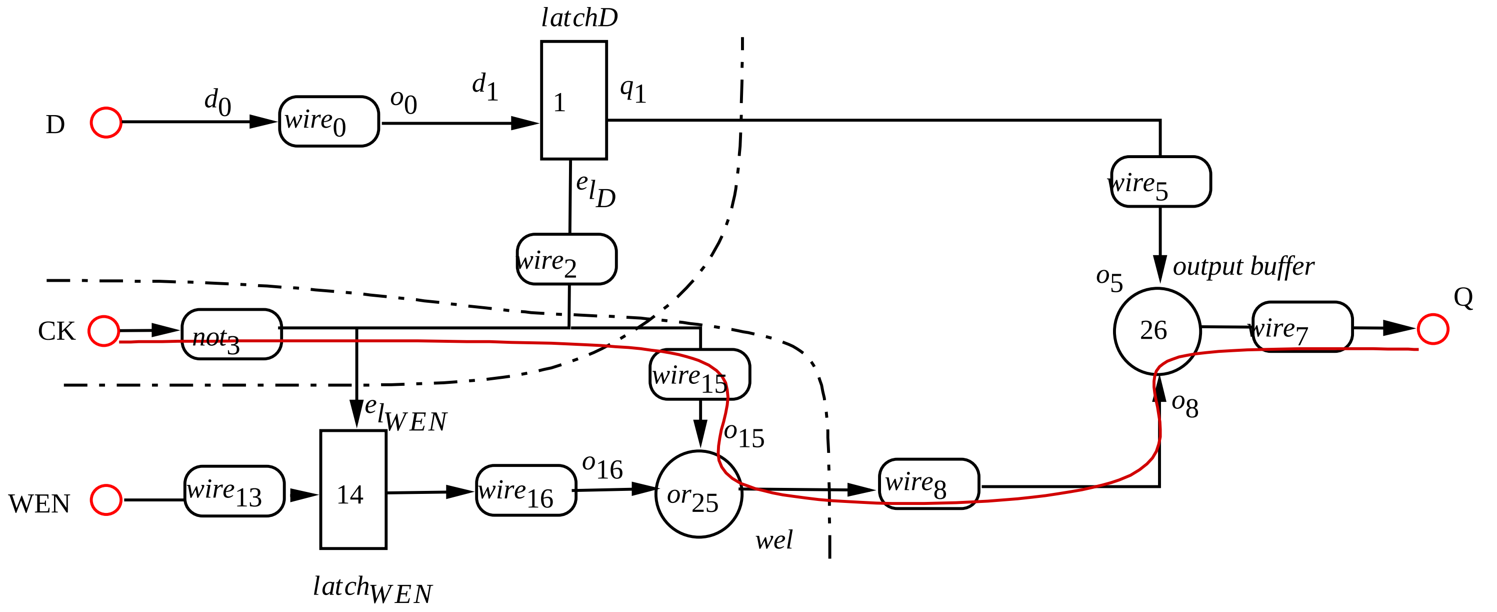
This is a minimal set of *sufficient* constraints.

Interpretation of the constraints

- The constraints in $Final$ characterize the *critical path* of the circuit.
- Some constraints in $Assumption$ characterize admissible edge ordering (from different inputs) arriving at a given node.

  $u_2^{\downarrow} + u_3^{\downarrow} \leq t_{LO} + l_0^{\uparrow} - t_{setup_D}$: when a rising edge arrives on its input, $latch_D$ is already open
- Some constraints in $Assumption$ characterize the functioning of latch.
  $u_{14}^{\downarrow} < t_{HI}$: the inertial delay of $latch_{WEN}$ is shorter than its enabling period.

Verification of two instances of SPSMALL



- The critical path is exhibited: $not_3 \longrightarrow wire_{15} \longrightarrow or_{10} \longrightarrow wire_8 \longrightarrow$ *output buffer* $\longrightarrow wire_7$
- The minimal response time: $t^{write}_{CK \rightarrow Q} = t^{write}_{max}$
- The setup time of D can be reduced (in the limits of *Assumption*) without changing the value of $t^{write}_{CK \rightarrow Q}$)

Conclusion

- Similar experiment for *read* operation and *write into the memory point* were successfully achieved.
- Obtention of a proof of consistency of timing parameters of the datasheet of SPSMALL (for two different implementations).

- Computation of the relations between timing parameters at an abstract level is *feasible*.
- However, it is *not automatic* (Abstract graph modelling and constraint selection).
- *Exponential Complexity* : applicable for small parts of the design only;
- Reduction strategies have been used in order to manage the complexity (case splitting /modular analysis).

Further developments

Questions:

- ▶ Is the bi-bounded delay model suitable for such an analysis ?
- ▶ From a practical point of view, is parametric approach worthwhile ? (vs. timed verification without parameters).
- ▶ Is automation possible ?
  - $\rightsquigarrow$ Combination of timing and functional abstractions
  - $\rightsquigarrow$ Resolution of the inverse problem is polynomial [EF07]

Finding answers to these questions is the purpose of *VALMEM* project !