Compositional Timing Analysis:
Power Plant Protection System Case Study

Simon Bliudze
Laboratory for Rigorous System Design (RiSD)
EPFL IC, Station 14, CH-1015, Lausanne, Switzerland
Simon.Bliudze@epfl.ch

Worst-case Traversal Time workshop
November 29th, 2011, Vienna

This work was carried out while the author was with CEA LIST (France)

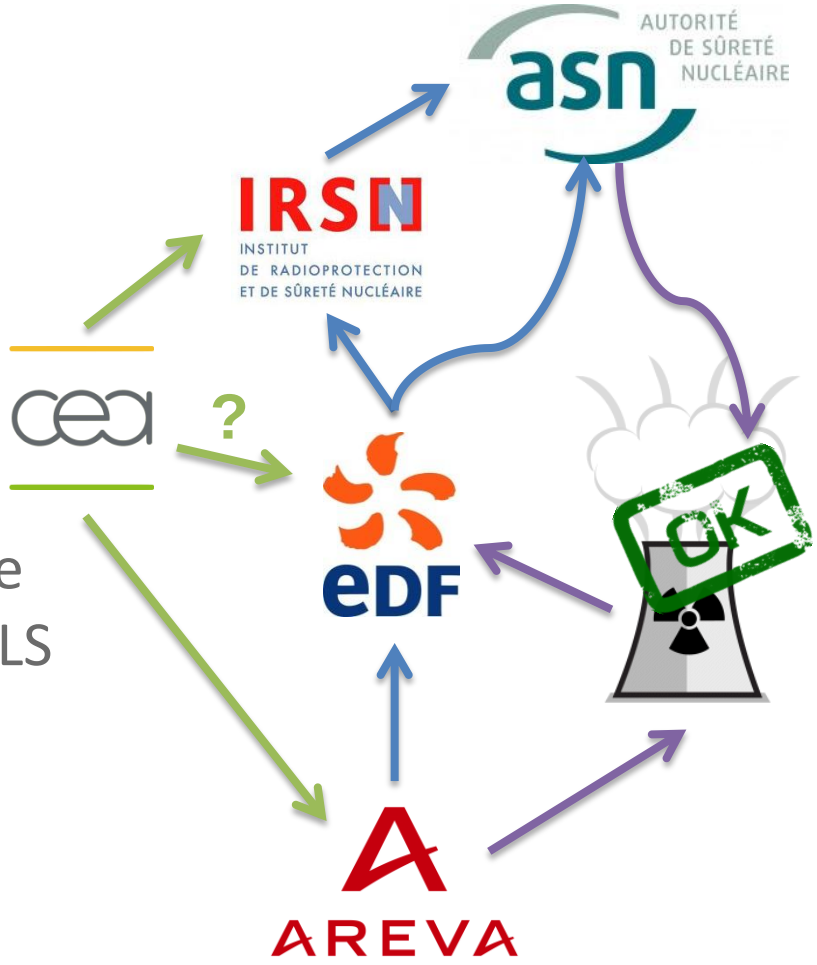
Presentation outline

- Context
- Case study system
- Methodology
- Application
- Conclusion

Nuclear industry in France

Partial overview:

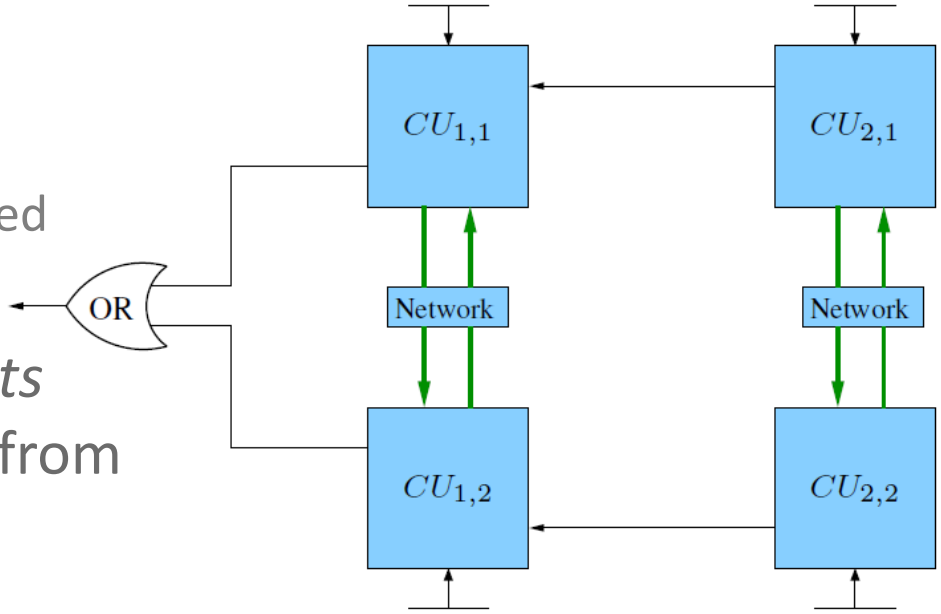
- Green arrows – research
- Blue arrows – workflow
- Violet arrows – product



- This work has been realised in the context of a CEA-IRSN project GALS
  - IF/TCA help from Verimag
- Double objective
  - Analyse the system behaviour
  - Present the results in a clear form

Case study system: P4 reactor

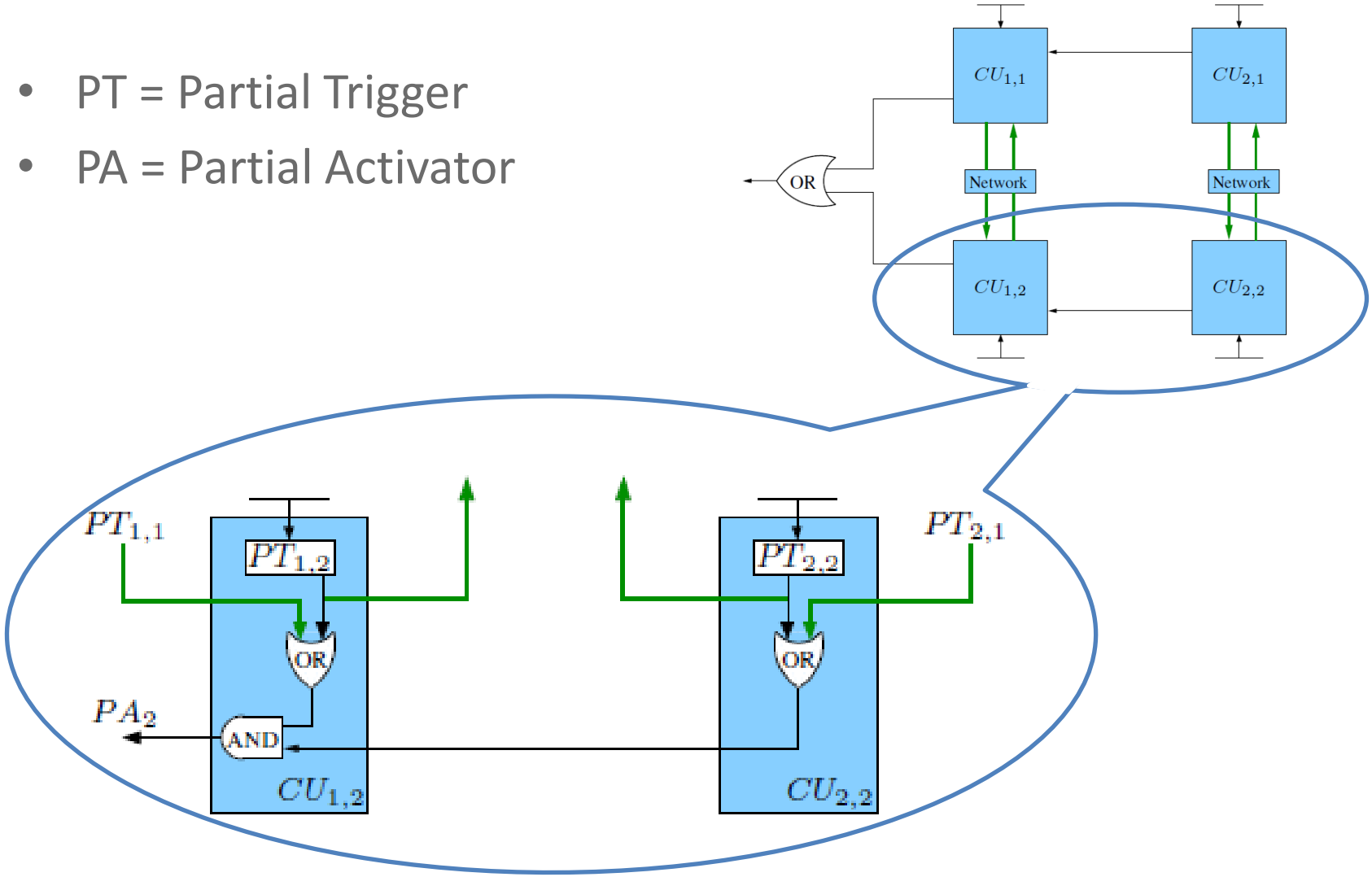
- Redundancy protection mechanism
  - Ensure that alerts are perceived
  - Filter out false ones
- A network of *computing units* (CU) taking on input signals from physical sensors
  - Synchronous operation
  - Asynchronous communication



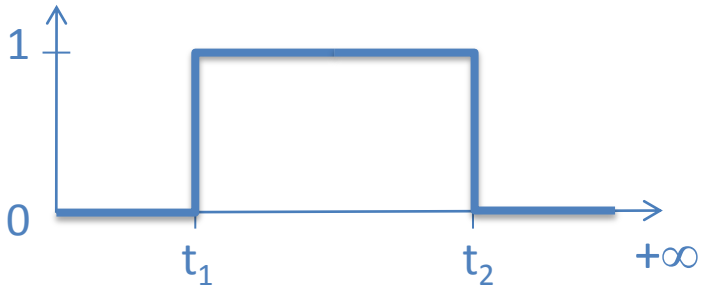
- Two values to determine:
  1. Minimal alarm duration that guarantees raising the Activator signal
  2. The worst case response time of the circuit

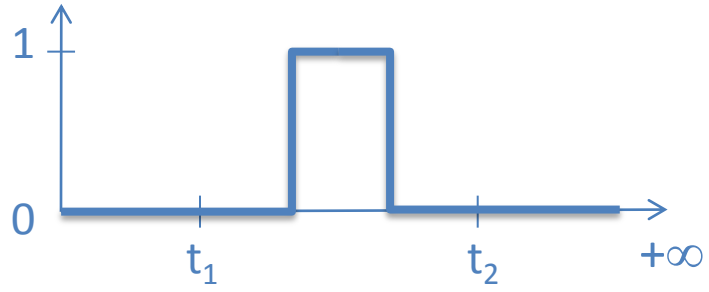
Zoom on the CUs

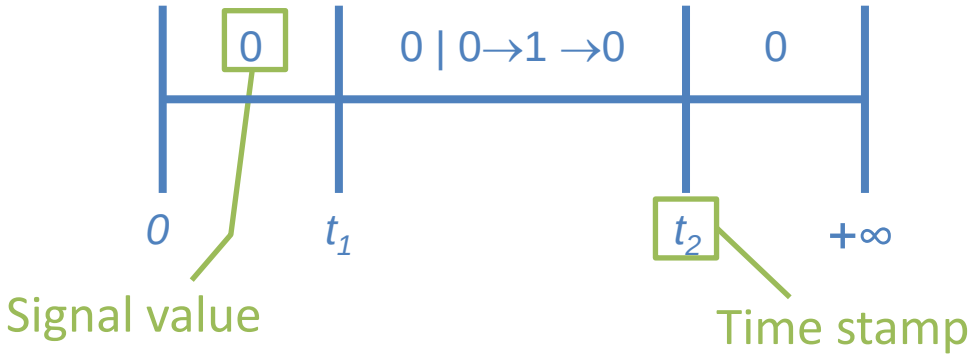
- PT = Partial Trigger
- PA = Partial Activator



Signal representation



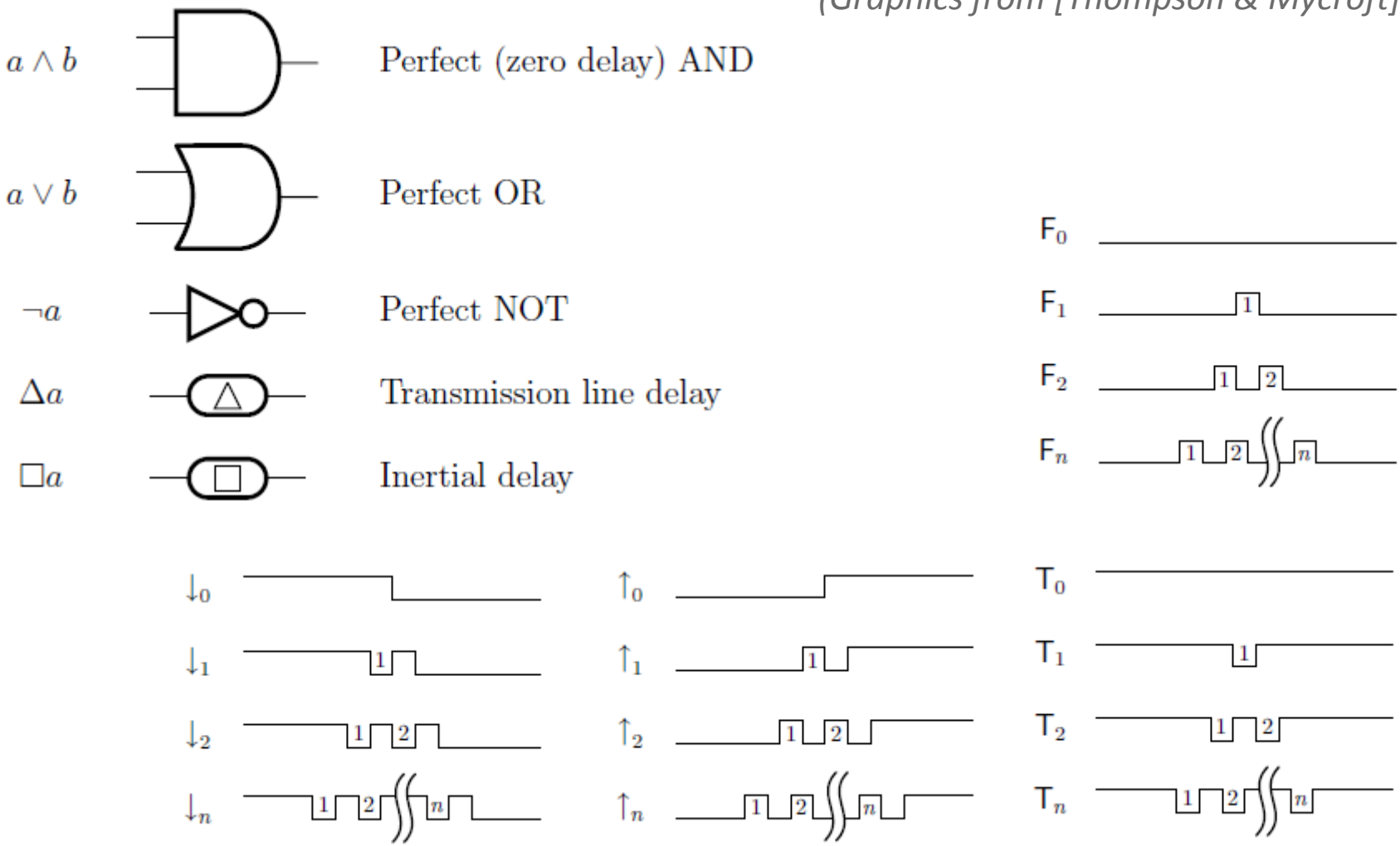




- Temporal graphs
  - Intuitive visualisation of the shape of the signal
  - Natural representation of non-determinism due to abstraction (transitional values, e.g. 0→1)

Transitional logics (1/2)

(Graphics from [Thompson & Mycroft])



Transitional logics (2/2)

(Graphics from [Thompson & Mycroft])

$\wedge$	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	$T_{1\ldots m+n}$	$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

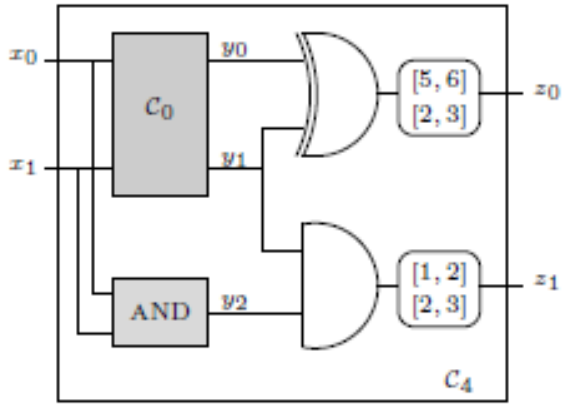
$\Delta^\sharp$	
F_0	F_0
F_n	F_n
T_0	T_0
T_n	T_n
$\uparrow_0$	$\uparrow_0$
$\uparrow_n$	$\uparrow_n$
$\downarrow_0$	$\downarrow_0$
$\downarrow_n$	$\downarrow_n$

$\Box^\sharp$	
F_0	F_0
F_n	$F_{0..n}$
T_0	T_0
T_n	$T_{0..n}$
$\uparrow_0$	$\uparrow_0$
$\uparrow_n$	$\uparrow_{0..n}$
$\downarrow_0$	$\downarrow_0$
$\downarrow_n$	$\downarrow_{0..n}$

- Galois connections between Boolean sequences and multi-value logics

IF/TCA tool-chain

(Graphics from [Ben Salah])



C4.tc :

```
input     {x0; x1}
output    {z0; z1}

(y0, y1)     :  C0 (x0, x1);
y2           :  AND (x0, x1);
z0           :  [5,6][2,3] ((~y0*y1)+(y0*~y1));
z1           :  [1,2][2,3] (y1*y2);
```

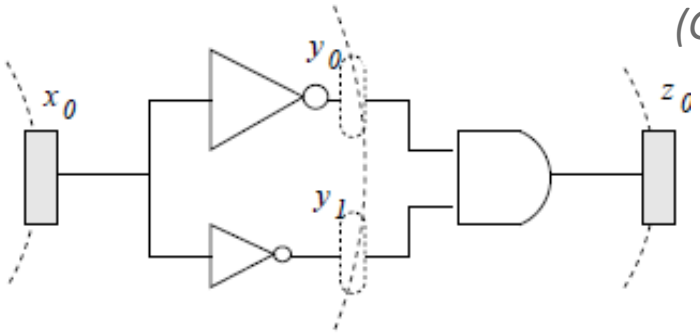
- Timed Circuit Analyzer tool box
  - Research prototype (Verimag)
- Two stages
  1. Transformation of a structural model of a circuit into a set of timed automata
     - IF (Intermediate Format)
  2. Computation of the automaton representing the entire circuit
     - Synchronous product (IF) + abstractions (IF/TCA)

Proposed methodology

- Combine the two techniques within a *hierarchical bottom-up approach*
  1. Model constituent components as IF processes
     - Translated TCA circuit elements, or
     - Generated by TCA at a previous stage
  2. Generate the IF interconnection model and insert the models of the constituent components
  3. Generate the *reduced* product automaton for the composed component
  4. Compute the shape of the output signal and time-stamp the edges

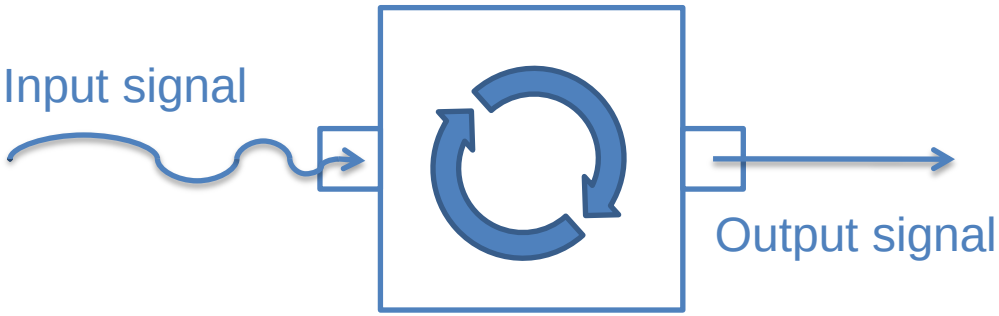
Abstractions in TCA

(Graphics from [Ben Salah])



- Elimination of redundant clocks
  - Each sub-component model has a certain nomber of clocks (0 or 1 for the automata generated from logical gates)
  - These clocks are preserved during the synchronous product computation, but eliminated by this abstraction
- Projection on the input/output variables
  - eliminate the intermediate signals
- Control-state space factorisation by bisimilation
  - Functional behaviour is preserved, temporal one is over-approximated

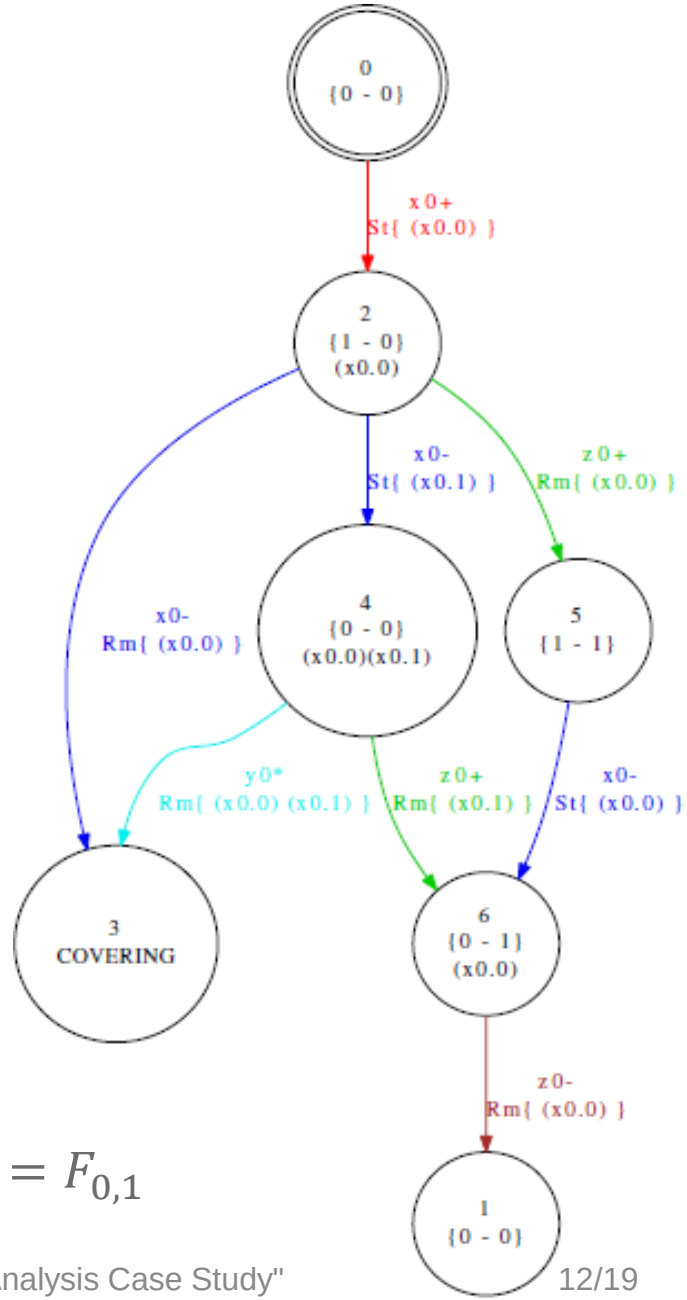
Switch element (1/2)



- Cyclic computation
  - (synchronous operation)
  - Cycle duration bounded by T_m et T_M
  - If input signal changes several times during one computation cycle, intermediate values are lost

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

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



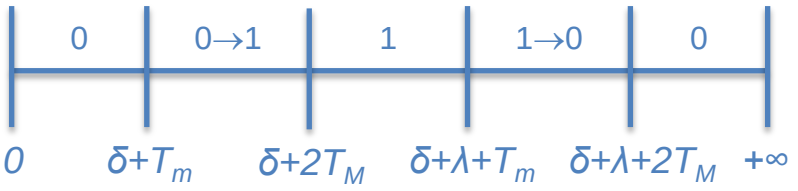
Switch element (2/2)

T_m = 27 ms and T_M = 33 ms

```
TRANSITION      :   2 -> 5
Event           :   z0+
Waves killed    :   (x0.0)
Time            :   (x0.0)[27,66]
Time detail     :
   1 -> 4   :  (x0.0)[27,66]
```

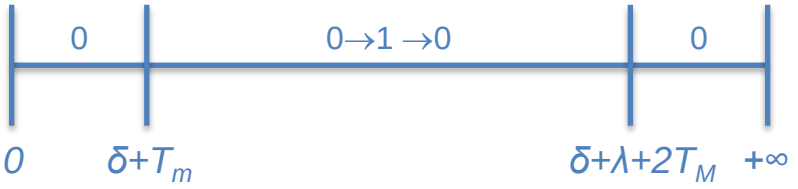
```
TRANSITION      :   4 -> 6
Event           :   z0+
Waves killed    :   (x0.1)
Time            :   (x0.0)[27,66] (x0.1)[0,33] (x0.1)-(x0.0)[-66,0]
Time detail     :
   3 -> 5   :  (x0.0)[27,66] (x0.1)[0,33] (x0.1)-(x0.0)[-66,0]
```

$\lambda + T_m > 2T_M$



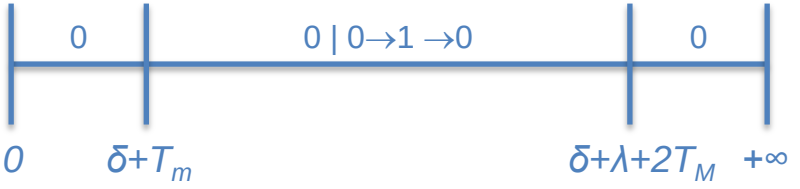

```
TRANSITION      :   6 -> 1
Event           :   z0-
Waves killed    :   (x0.0)
Time            :   (x0.0)[27,66]
Time detail     :
   5 -> 6   :  (x0.0)[27,66]
```

$T_M < \lambda \leq 2T_M - T_m$



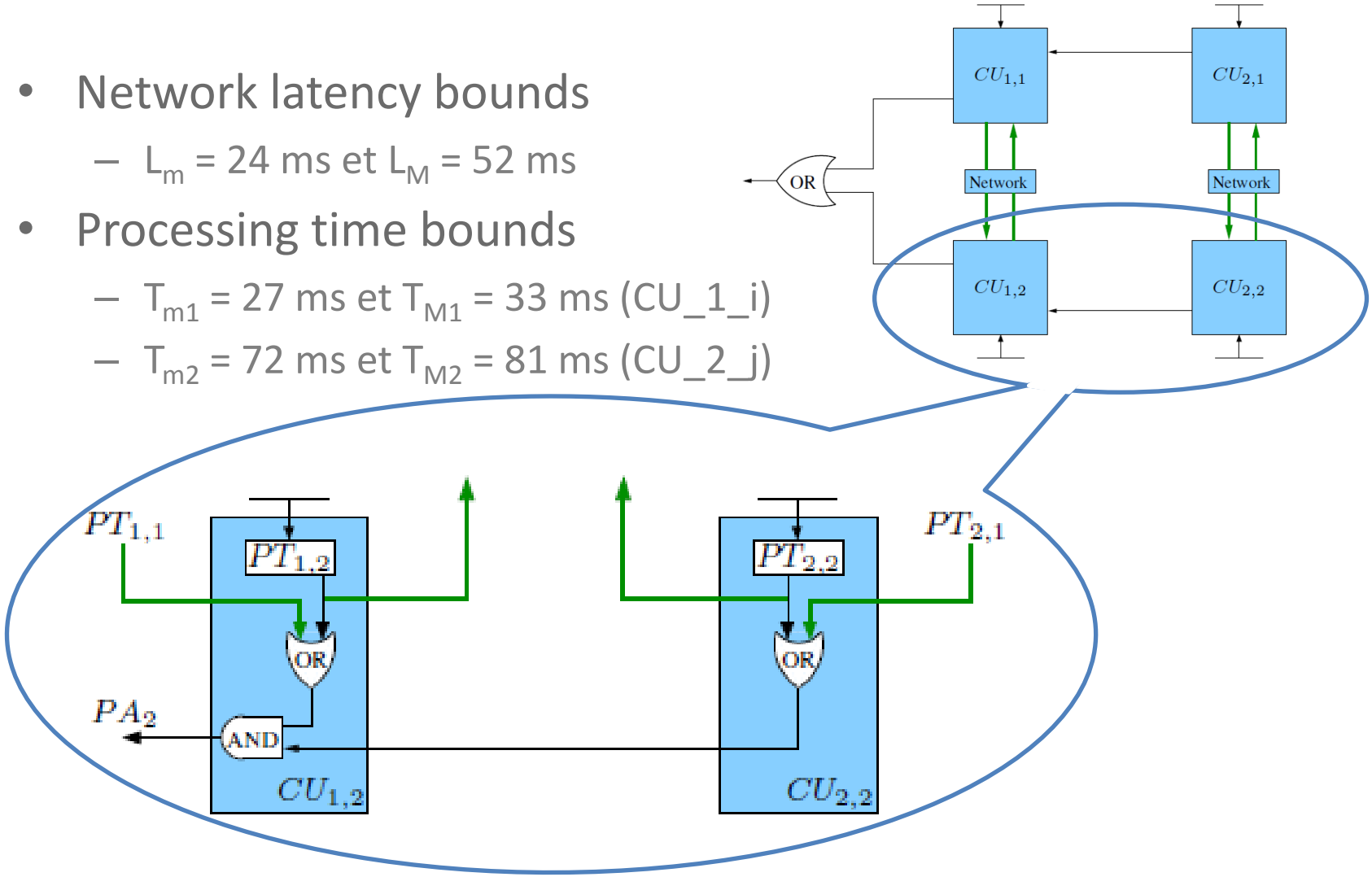

```
TRANSITION      :   2 -> 3
Event           :   x0-
Waves killed    :   (x0.0)
Time            :   (x0.0)[0,33]
Time detail     :
   1 -> 2   :  (x0.0)[0,33]
```

$\lambda \leq T_M$



P4 Reactor Protection System

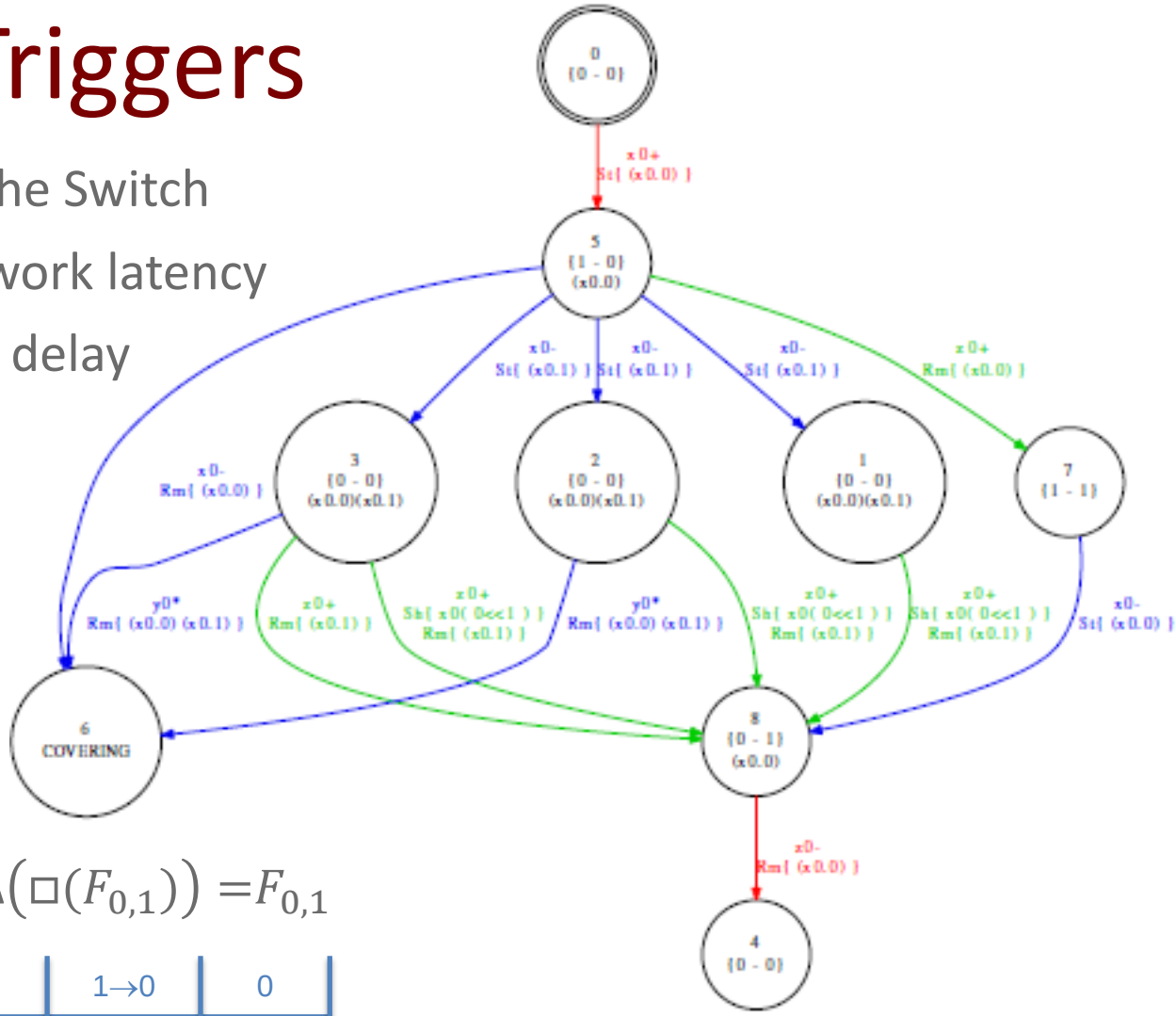
- Network latency bounds
  - L_m = 24 ms et L_M = 52 ms
- Processing time bounds
  - T_{m1} = 27 ms et T_{M1} = 33 ms (CU_1_i)
  - T_{m2} = 72 ms et T_{M2} = 81 ms (CU_2_j)



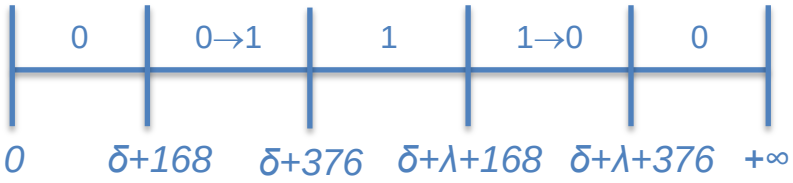
P4: Partial Triggers

- $PT_{i,1}$ exactly as for the Switch
- $PT_{i,2} = PT_{i,1}$ + network latency
  + input delay

```
input {x0}
output {z0}

y0 : [0,81] x0;
y1 : [72,81] y0;

y2 : [24,52] y1;

y3 : [0,81] y2;
z0 : [72,81] y3;
```



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



λ > 199 ⇒ the signal cannot be lost

P4: ½ vote in the CU

- $CU_{i,1}$ computes the disjunction $PT_{i,1}$ + $PT_{i,2}$
- Two steps:
  - Generate the structure with TCA
  - Inject generated IF code for Partial Triggers

```
input {x0}
output {z0}

y0 : [0,0] x0;
y1 : [0,0] x0;

z0 : [0,0] (y0+y1);
```

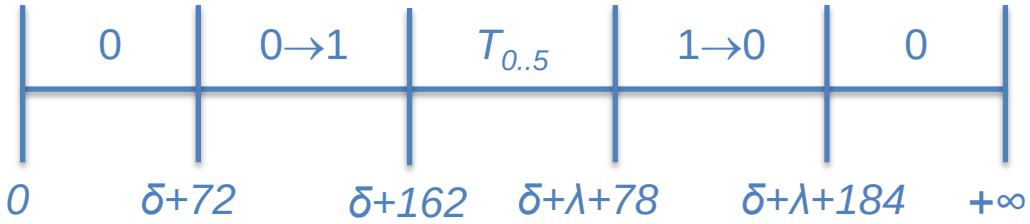
```
process x0(1);
...
endprocess;

process z0(1);
...
endprocess;

process y0(1);
...
endprocess;

process y1(1);
...
endprocess;
...
```

```
process x0(1);
...
endprocess;

process z0(1);
...
endprocess;

process PT_1(1);
...
endprocess;

process PT_2(1);
...
endprocess;
...
```

P4: The End

- Applying the presented techniques iteratively, we obtain, for the Activator signal

$$F_{0..3} \vee F_{0..3} = F_{0..6} = 0 \rightarrow T_{0..5} \rightarrow 0 = (0 \rightarrow 1) \rightarrow T_{0..5} \rightarrow (1 \rightarrow 0)$$



- And the answers to the initial questions:
  1. $\lambda > 162 \Rightarrow$ The Activator signal is guaranteed to be raised
  2. The worst case reaction time is 162 (whenever the signal is not lost)

Conclusion

- Automata-based solutions should not be discarded right away
- The enabling factor seems to be the simplicity of the properties under analysis
- Bottom-up hierarchical approach
  - Mostly manual
  - Synchronous product, forward reachability and abstractions by TCA
- Some limitations
  - Properties can be derived only with respect to input event dates (due to clock projection)
  - Explicit modelling of transmission line delays in TCA would simplify the analysis

Perspectives

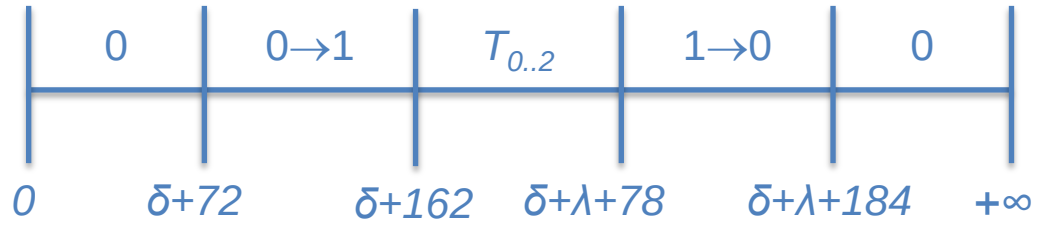
- Application of other tools and approaches (UPPAAL…)
- Implementation of tools for the presented methodology
- Study a tighter integration of transitional logics with TCA
  - Can information about the signal shape guide abstractions, notably the projection on the input/output variables

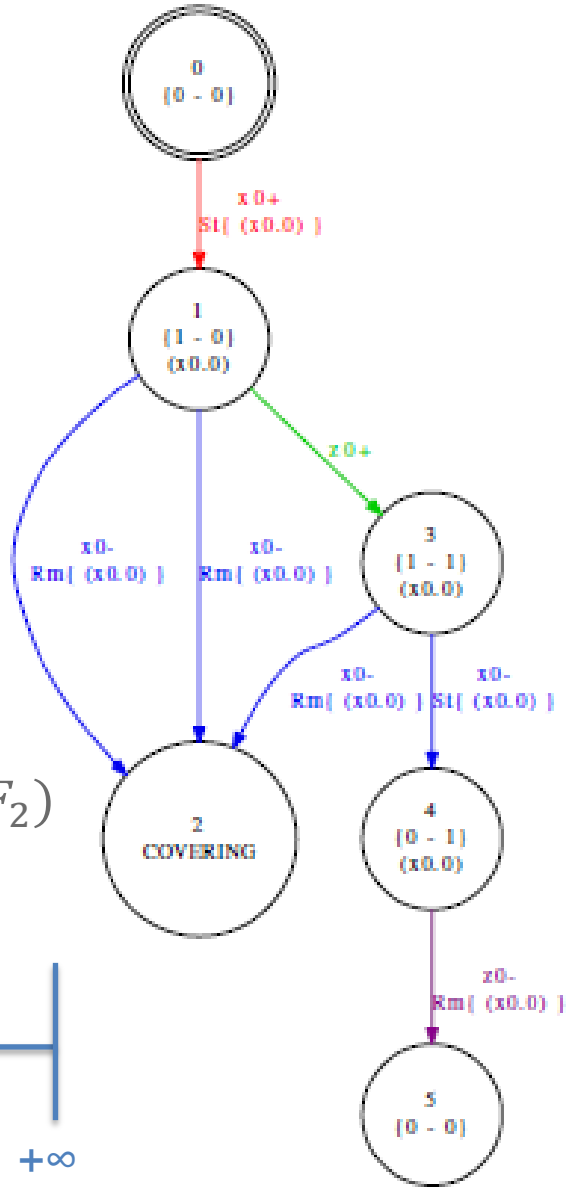
P4 : Partial Activator

- $CU_{1,j}$ computes the conjunction of the ½ votes in $CU_{1,j}$ and $CU_{2,j}$

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

$$F_{0..2} \wedge F_{0..2} = (F_0|F_1|F_2) \wedge (F_0|F_1|F_2)$$
$$= (F_0 \wedge F_{0..2})|(F_1 \wedge F_1)|(F_1 \wedge F_2)|(F_2 \wedge F_2)$$
$$= F_0|F_{0,1}|F_{0..2}|F_{0..3} = F_{0..3}$$

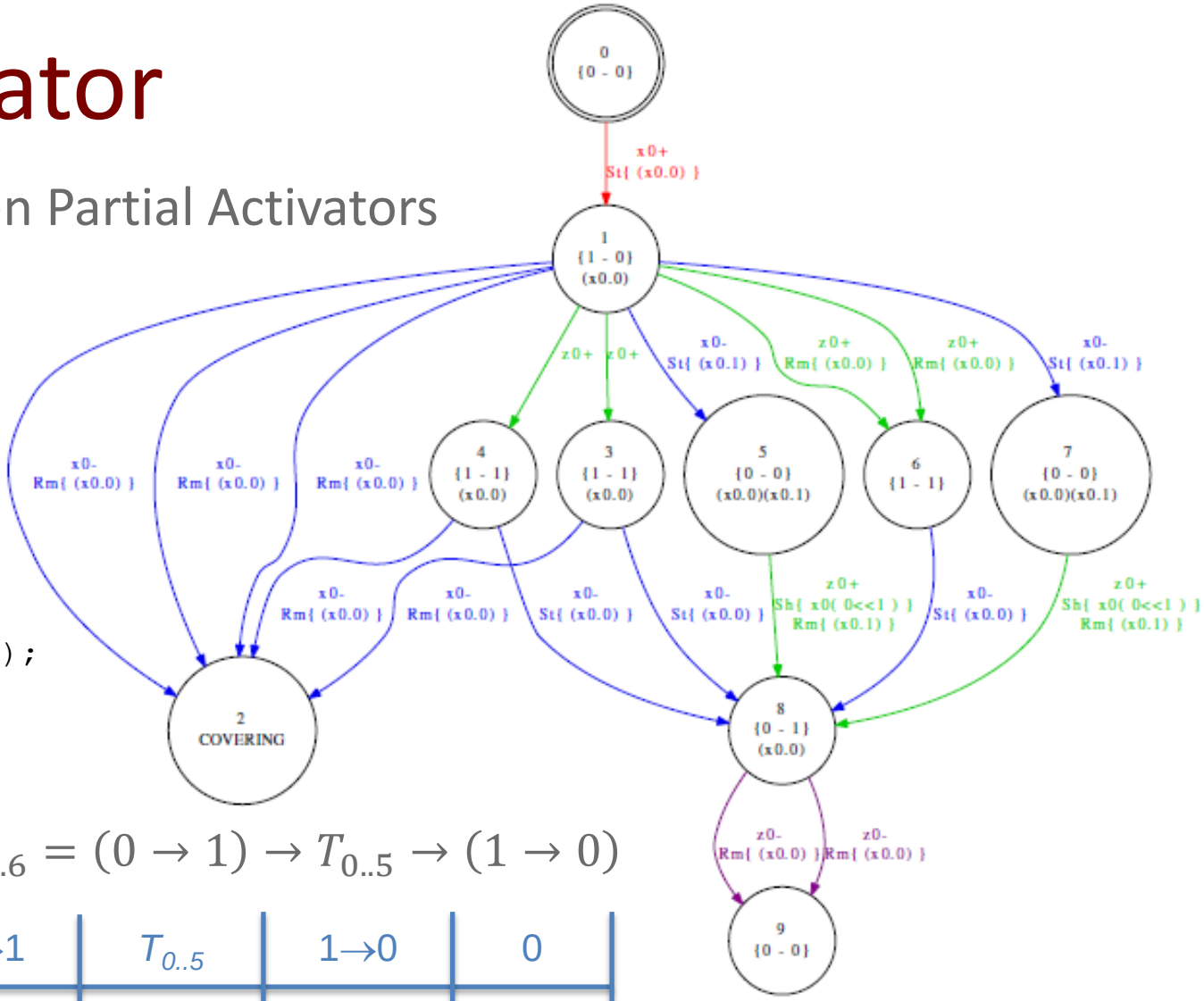




P4 : Activator

- ½ vote between Partial Activators PA_1 and PA_2

```
input {x0}
output {z0}

y0 : [0,0] x0;
y1 : [0,0] x0;

z0 : [0,0] (y0+y1);
```



$$F_{0..3} \vee F_{0..3} = F_{0..6} = (0 \rightarrow 1) \rightarrow T_{0..5} \rightarrow (1 \rightarrow 0)$$