Testing Logical Diagrams in power plants: a tale of LTL model checking

Authors:

- **Aziz SFAR**
- **David CARRAL**
- **Dina IROFTI**
- **Madalina CROITORU**

Laboratory of Computer Science, Robotics and Microelectronics of Montpellier

Électricité de France

FMICS 2023

Formal Methods for Industrial Critical Systems

email: masfargandoura@lirmm.fr

Background

Fukushima Daiichi Nuclear Disaster March 2011

New Safety and Security norms for nuclear power plants.

Implement New Safety Requirements

- **Sensors**
- **Agents**
- **Information**

inputs

Operating parameters exceed the limits

Logical Controller

outputs

- **Actuators**

Shutdown of nuclear reactor

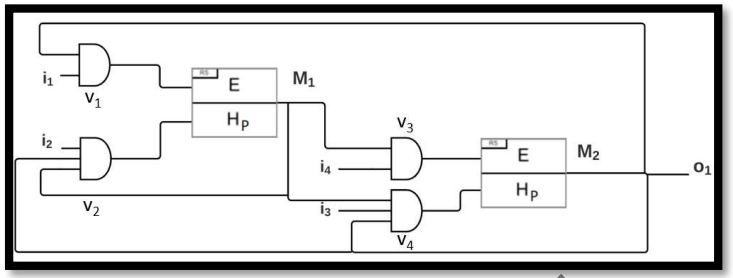
Reactor trip or scram

- Security and protection
- Control of the process

email: masfargandoura@lirmm.fr

Background

Specifications: Logical Diagrams



Modifications

Implementation

Post Fukushima Safety Requirements

Logical Controller

➔ Test newly implemented functions.

➔ Test non-modified functions (regression).

Done manually. Let's make it automatic.

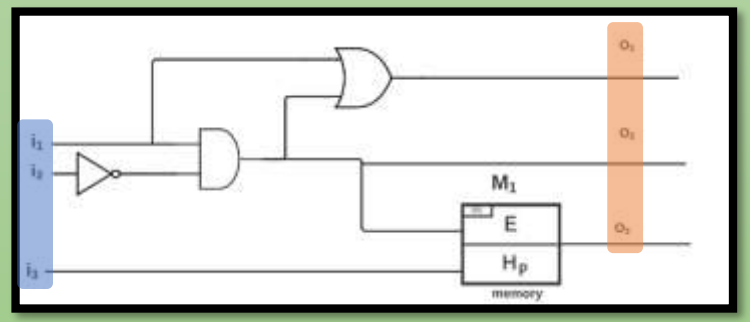
email: masfargandoura@lirmm.fr

Outline of the presentation

- I. Logical Diagrams: what are the challenges ?
- II. Formal definition of Logical Diagrams.
- III. From Logical Diagrams to LTL formalism.
- IV. Conclusions and future work

I. Logical Diagrams: what are the challenges ?

Logical Diagrams: test scenarios generation



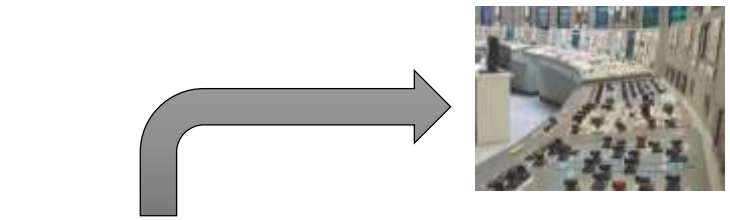
Functional Specification: Logical Diagram

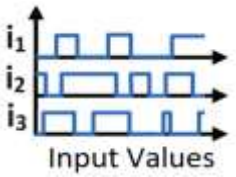
Test Generation

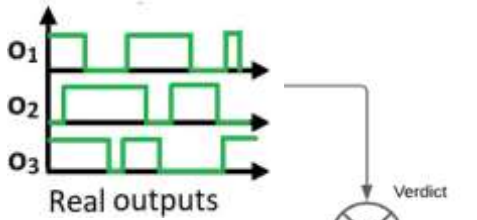
Output (o_1, o_2, o_3): 0 0 0 → 1 0 1 **?**

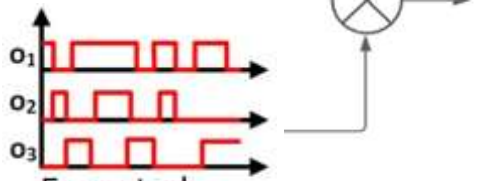
Input (i_1, i_2, i_3): 0 0 1

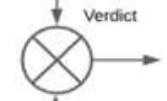
Logical Controller







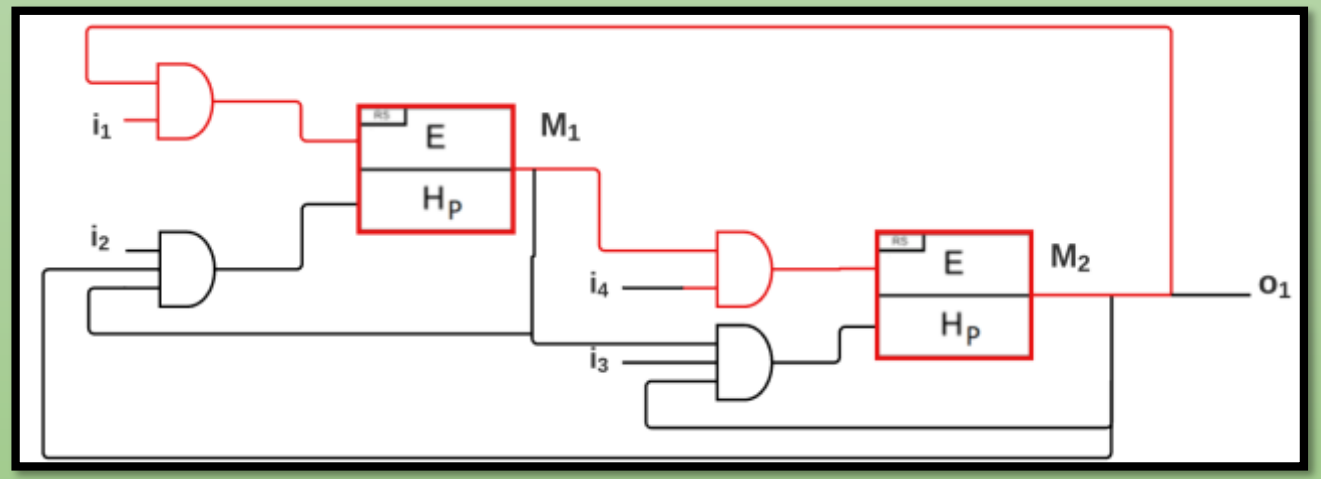




Black Box Testing

email: masfargandoura@lirmm.fr

Logical Diagrams: verification of stability



Functional Specification: Logical Diagram

Fixed input (i_1, i_2, i_3, i_4): 1 1 1 1

Output o_1: 1 → 0 → 1 → 0 → 1 …

➔ Output o_1 does not converge when inputs are set to 1 1 1 1

Verify that the behavior described by the logical diagram is always convergent

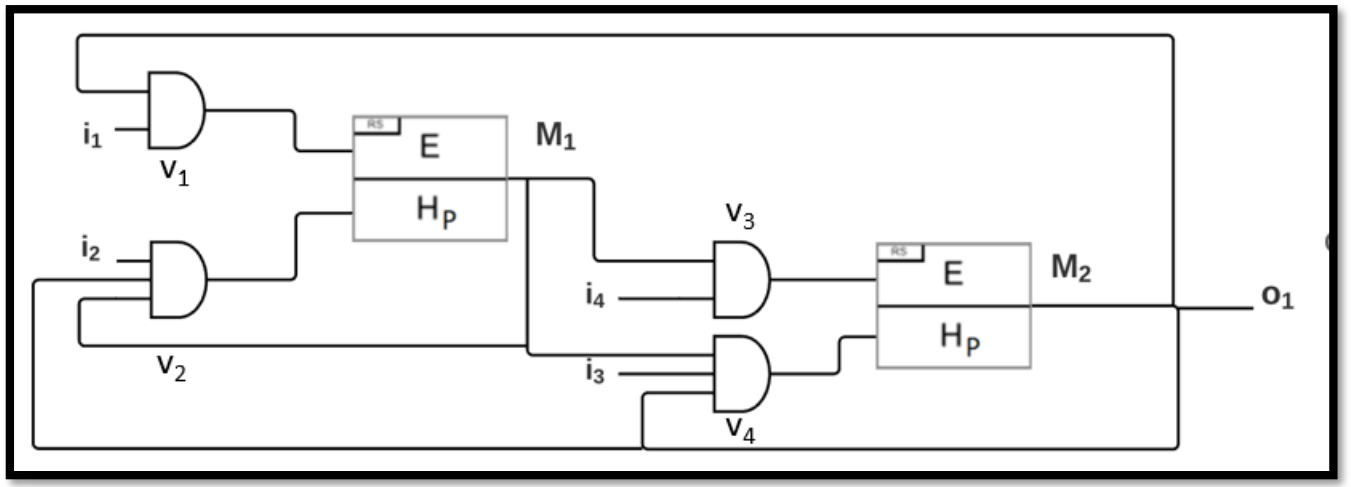
email: masfargandoura@lirmm.fr

Logical Diagrams: challenges

- Lack of formal definition and analysis methods.

➔ Generating test scenarios is done through manual simulation of the Logical Diagram.

- Verifying the stability of all ouputs in all cases: how?



Functional Specification

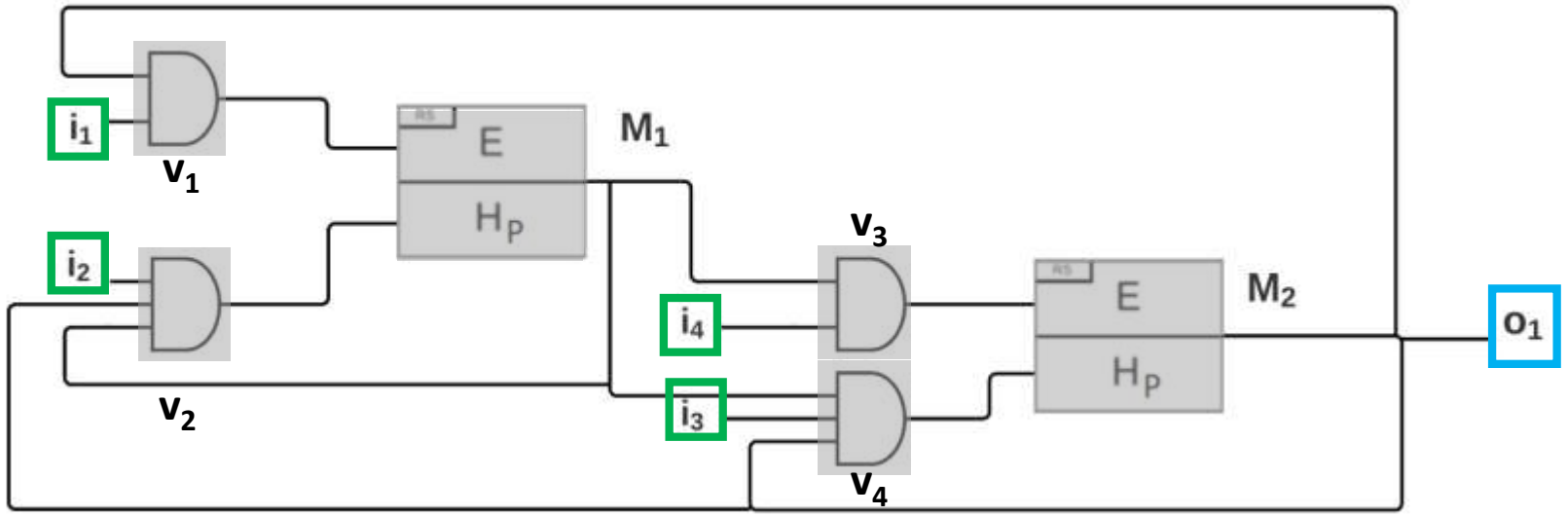
Manually generating test scenarios is a hard error-prone task.

Verifying stability directly on the logical diagram is even harder.

email: masfargandoura@lirmm.fr

II. Logical Diagrams: a formal definition.

Logical Diagrams: what are they ?

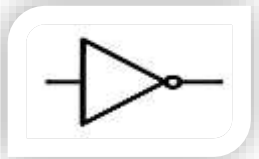


A logical diagram is specification model used for EDF Logical Controllers.

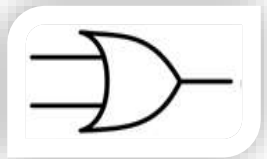
A set of logical inputs and outputs linked through Interconnected logical blocks.

- Inputs
- Logical functions
- Outputs

Logic gates



NOT

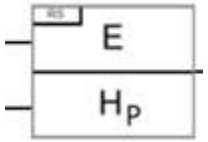


OR

AND

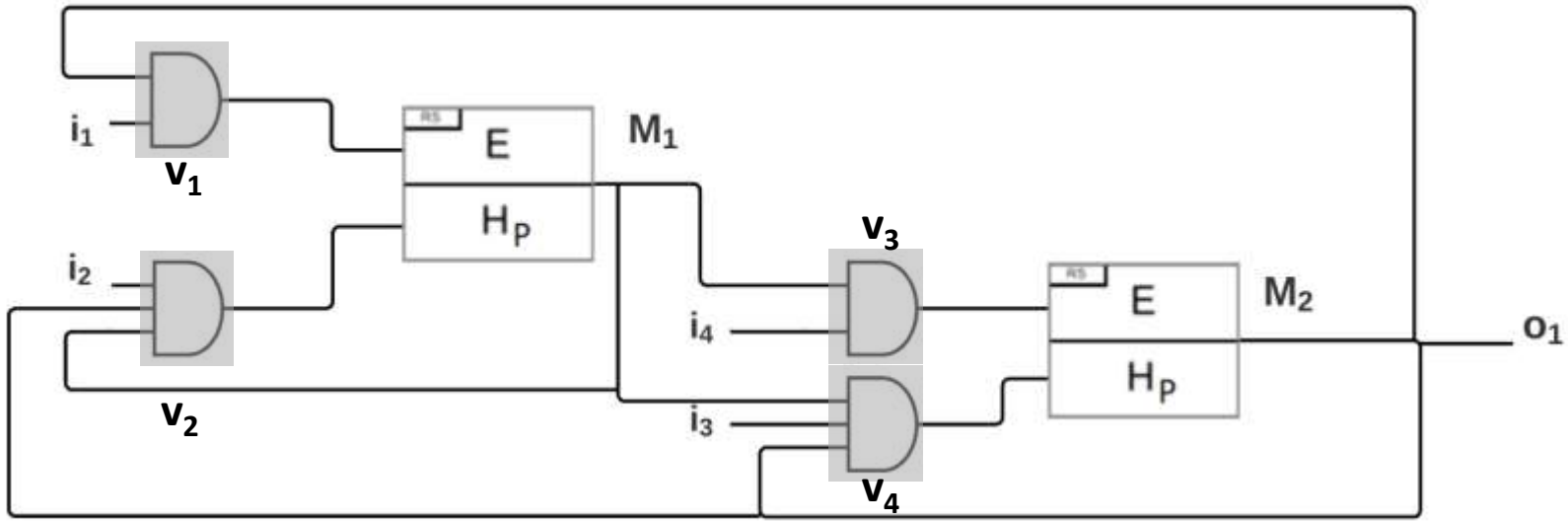
Memory blocks

M_E

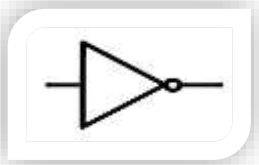


M_H

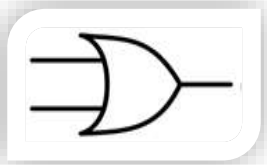
Logical Diagrams: how do they function ?



Logic gates

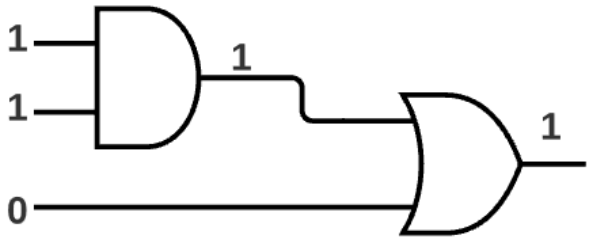


NOT



OR

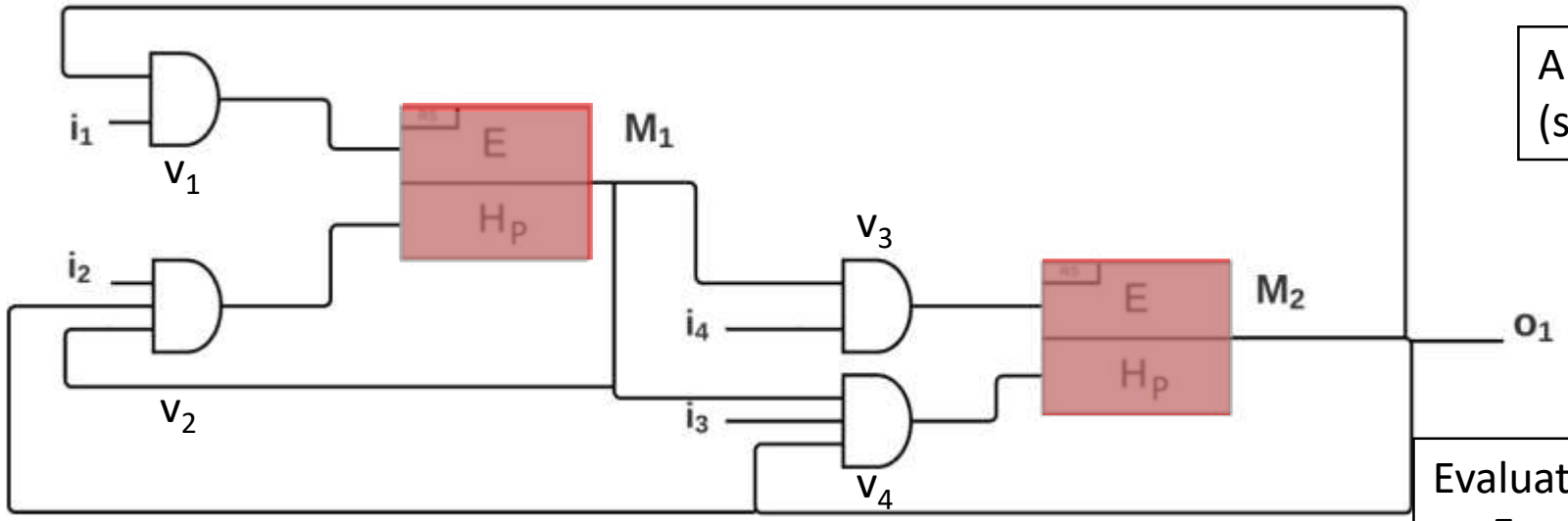
AND



Synchronous execution from left to right

email: masfargandoura@lirmm.fr

Logical Diagrams: how do they function ?



A memory block has 1 output and 2 inputs E (set) and H (reset).

Two types:
- M_E priority for the set (E)
- M_H priority for the reset (H)

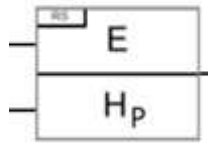
Evaluation of a memory block:
- E set to 1 activates the output.
- H set to 1 deactivates the output.
- Both set to 0 maintains the last output value.
- Both set to 1 activates the output in the case of M_E and deactivates the output in the case of M_H.

Memory block are evaluated in a sequence in accordance to a given order.

Example: (M1, M2)

Memory blocks

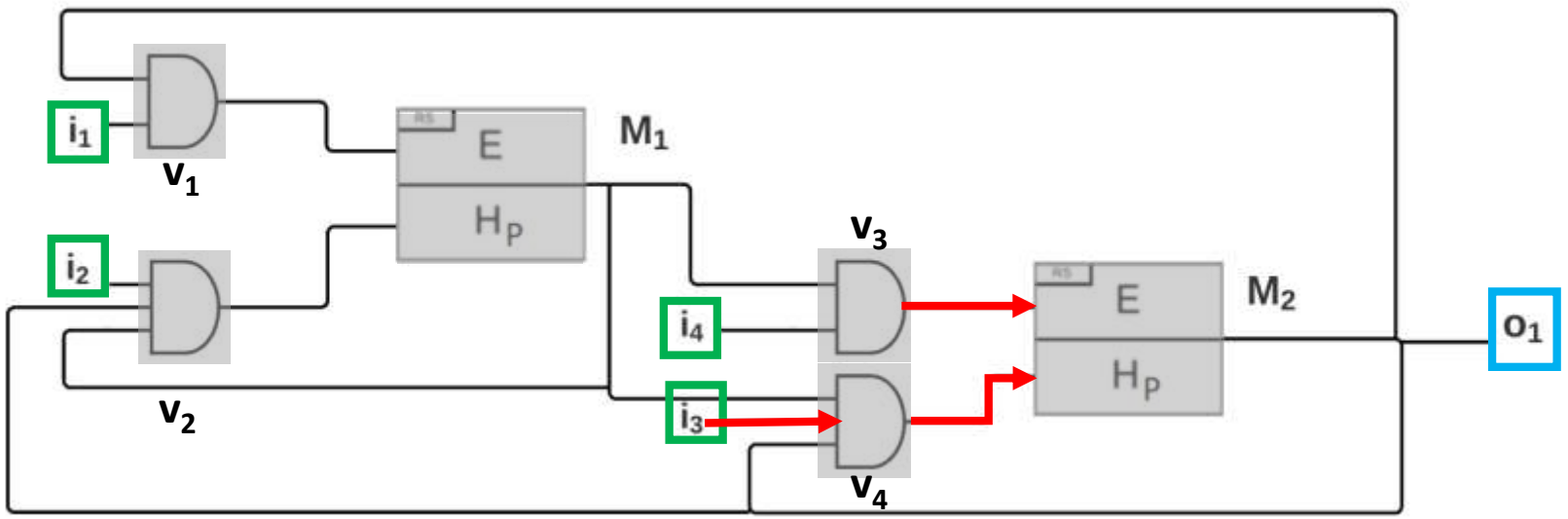
M_E



M_H

email: masfargandoura@lirmm.fr

Logical Diagrams: formal definition



A Logical Diagram $LD = \langle V, E, O \rangle$:

- V: the set of vertices
- E: the set of edges
- O: memory evaluation order

Vertices *V:*

- *input,*
- *output,*
- *or, and, not,* M_E, M_H

Edges *E:*

- $u \rightarrow v$
- $u \overset{s}{\rightarrow} v$
- $u \overset{r}{\rightarrow} v$

- $i_3 \rightarrow v_4$
- $v_3 \overset{s}{\rightarrow} M_2$
- $v_4 \overset{r}{\rightarrow} M_2$

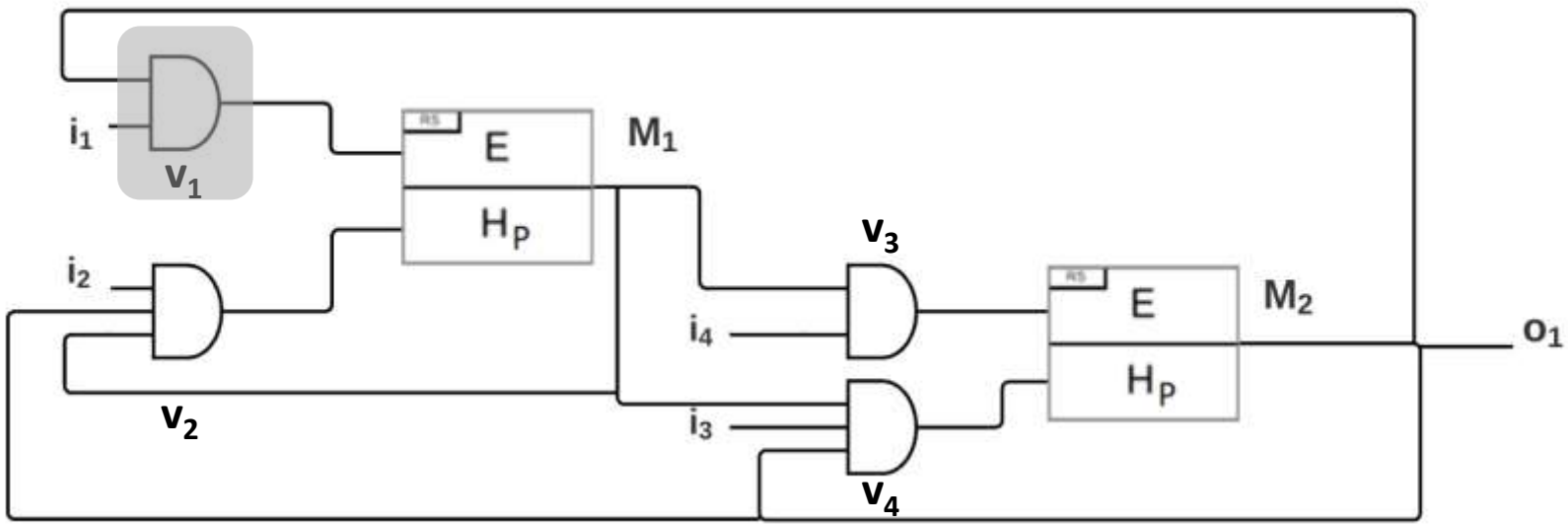
Memory evaluation order *O:*

A bijection $V(M) \rightarrow \{1, \dots, |V(M)|\}$

Gives an evaluation order to each memory vertex M_E and M_H

email: masfargandoura@lirmm.fr

Logical Diagrams: formal definition



A logical Diagram $LD = (V, E, O)$:

- V: the set of vertices
- E: the set of edges
- O: memory evaluation order

Evaluation:

- f: the initializing function
- $LD_f(v, k)$: output function

The initializing function f:

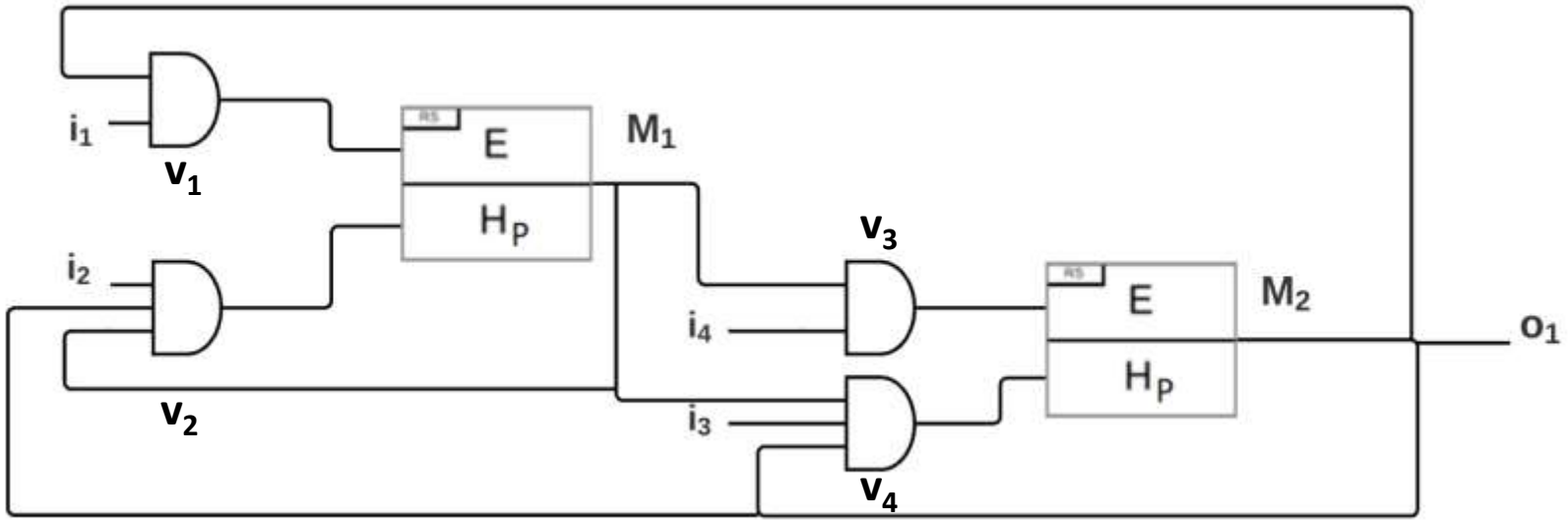
Associates a logical value
- to each input vertex
- to each memry vertex

The output evaluation function $LD_f(v, k)$:

Gives the logical output value of a vertex v at an evaluation step k for an initilizing function f.

Example: $LD_f(v_1, 0) = LD_f(i_1, 0) \wedge LD_f(M_2, 0)$

Logical Diagrams: formal definition



A logical Diagram $LD = \langle V, E, O \rangle$:

- V: the set of vertices
- E: the set of edges
- O: memory evaluation order

Evaluation:

- f: the initializing function
- $LD_f(v, k)$: output function

Stability:

For every vertex v and every initlizing function f, there is some k' such that $LD_f(v, k) = LD_f(v, k-1)$; $k \geq k'$

Activation and deactivation of an output:

An output vertex O is activated (resp. deactivated) if there is some f *for which* $LD_f(o, k)$ converges to 1 (resp. 0)

III. From Logical Diagrams to LTL formulas.

From LD to LTL: what for?

A logical diagram lacks formal methods to:

- Check stability.
- Automatically generate scenarios that lead to wanted output values.

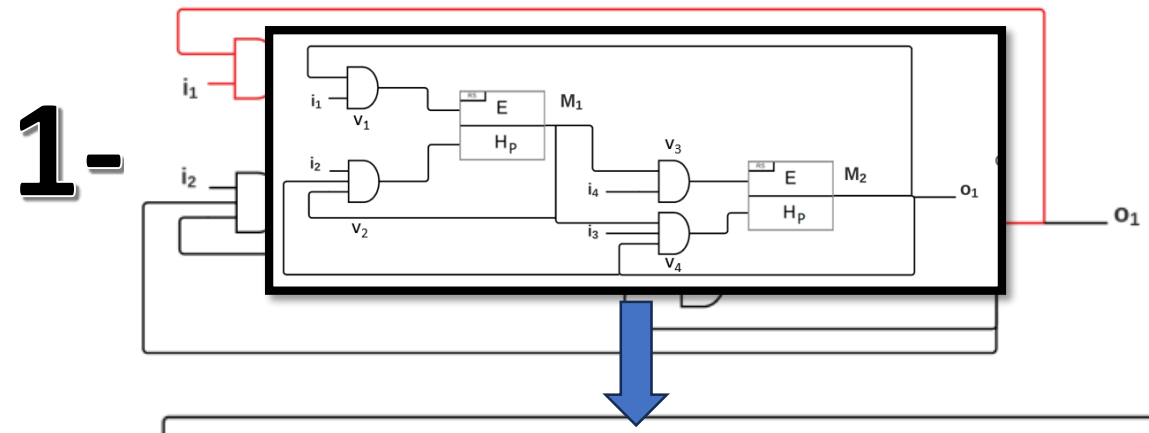
We formally define Logical Diagrams

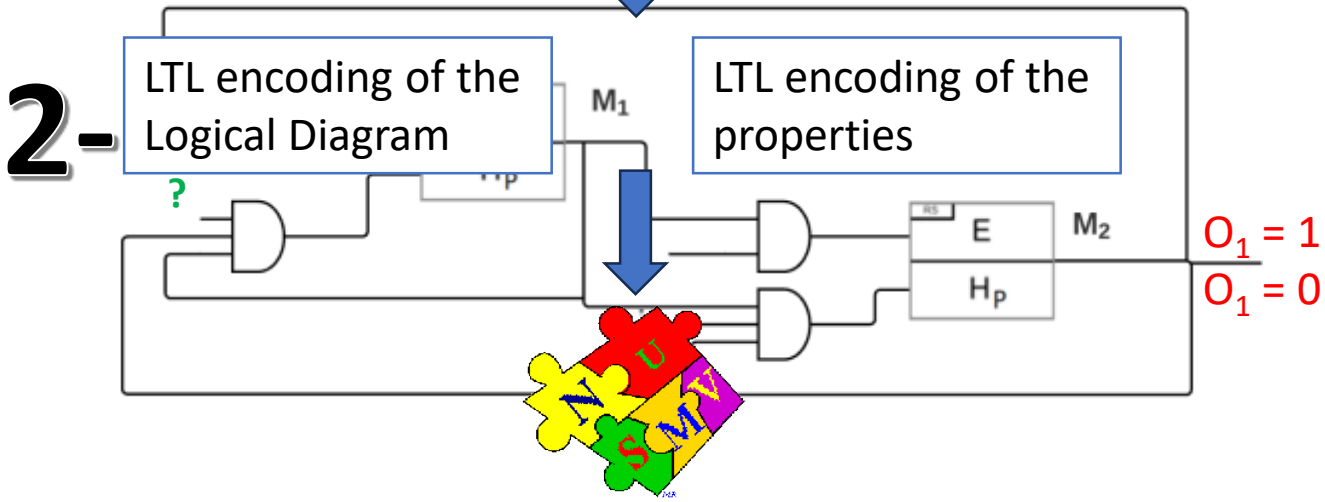
We establish a sound and complete LTL encoding of the Logical Diagrams.

We encode stability, activation and deactivation of outputs into LTL formulas.

We use LTL model checkers to verify stability then generate the input scenarios we need.

Manual generation and verification





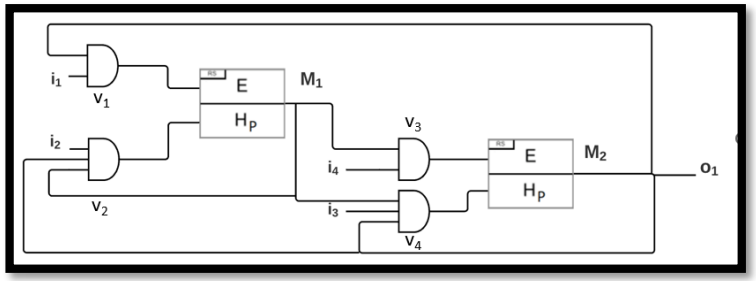
Automatic generation and verification

email: masfargandoura@lirmm.fr

From LD to LTL: the encoding

Logical Diagram

LTL Encoding



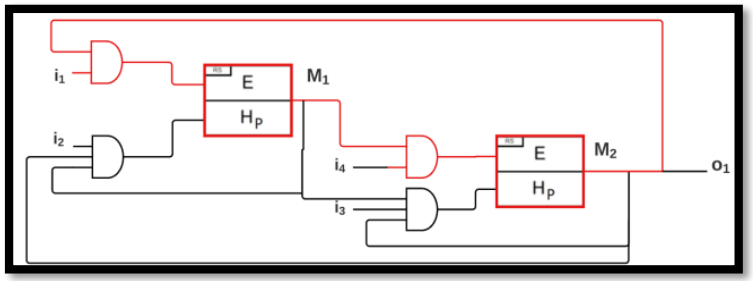
$LD = \langle V, E, O \rangle$

$LD_f(v, k)$

Theorem 1

- v → AP_{LD}
- $E; O; LD_f \rightarrow \tau_{LD}$
- $f \rightarrow \tau_f$

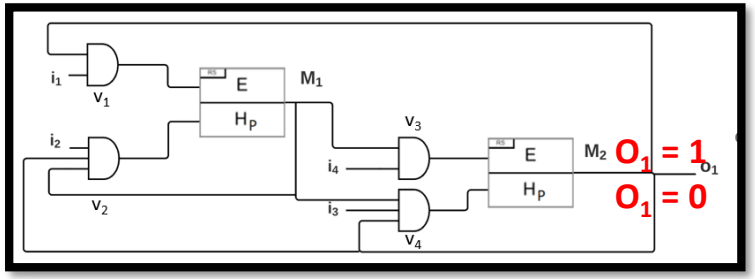
W = s_0, s_1, …



Stability

Theorem 2

- Convergence property → τ_{stable}
- Uniform stability → $\tau_{LD} \vDash \tau_{stable}$ P_{stab}



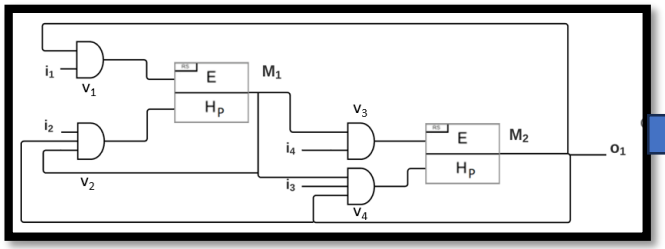
Activation
Deactivation
of outputs

Theorem 3

- Activation / Deactivation → $\tau^o_{act} / \tau^o_{deact}$
- An init f for act → $\tau_{LD} \cup \tau^o_{act} \vDash \bot$ $P_{act}(o)$
- An init f for deact → $\tau_{LD} \cup \tau^o_{deact} \vDash \bot$ $P_{deact}(o)$

email: masfargandoura@lirmm.fr

From LD to LTL: LTL Model checking



Logical Diagram

P_{stab}: $\tau_{LD} \vDash \tau_{stable}$

$P_{act}(o)$: $\tau_{LD} \cup \tau^{o}_{act} \vDash \bot$

$P_{deact}(o)$: $\rightarrow \tau_{LD} \cup \tau^{o}_{deact} \vDash \bot$

LTL encoding

LTL Model Checking

- Stability: True / False
- Input values
- Evaluation trace

A counter-example

$\mathcal{P}_{stab}$: True in 26 minutes using the k-liveness technique (k-MC)

A real case Logical Diagram:

- **16 inputs**
- **12 outputs**
- **19 memories**
- **77 logic gates**

Manual generation of test cases

	BDD	BMC	SBMC	k-MC
$\mathcal{P}_{act}(o)$	timeout	mean: 26s (33) min: 4.2s (22) max: 2m 11s (57)	mean: 3.4s (33) min: 1.3s (22) max: 11.3s (57)	mean: 6.2s (33) min: 3.9s (24) max: 12.9s (57)
$\mathcal{P}_{deact}(o)$	timeout	mean: 18.2s (32) min: 4s (22) max: 35s (39)	mean: 2.9s (32) min: 1.3s (22) max: 4.4s (39)	mean: 5.3s (32) min: 3.4s (22) max: 7.5s (35)

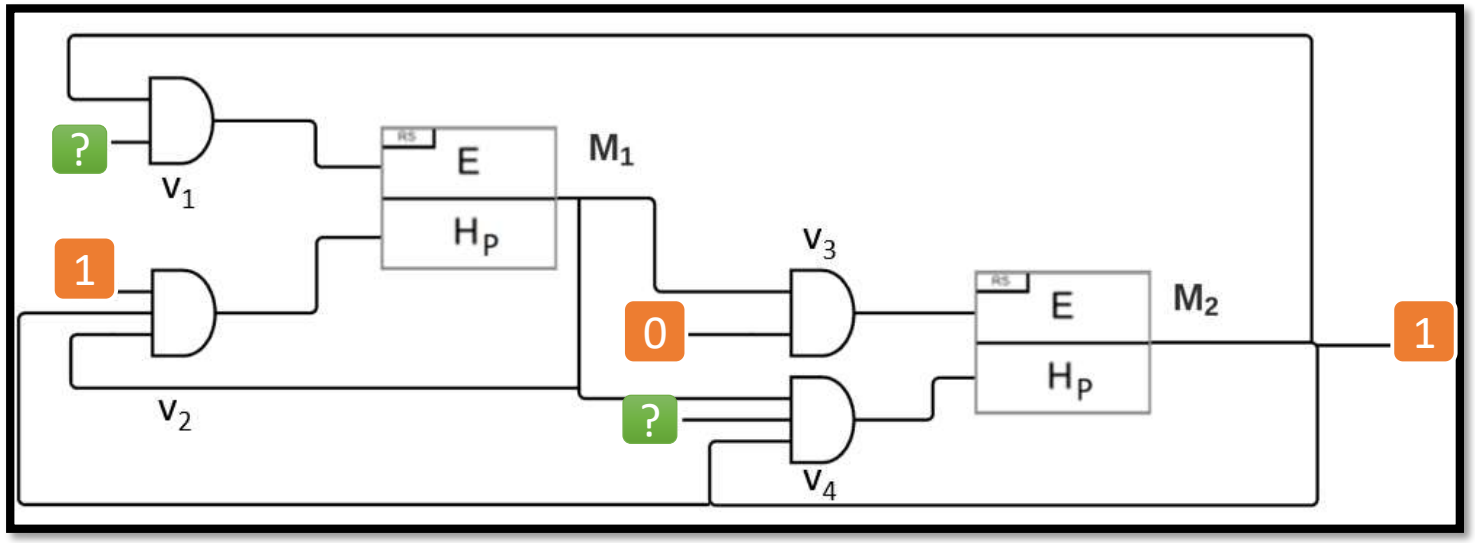
- Verify stability
- Generate a scenario of activation/deactivation for each of the 12 outputs

email: masfargandoura@lirmm.fr

IV. Conclusion and future work

Conclusions

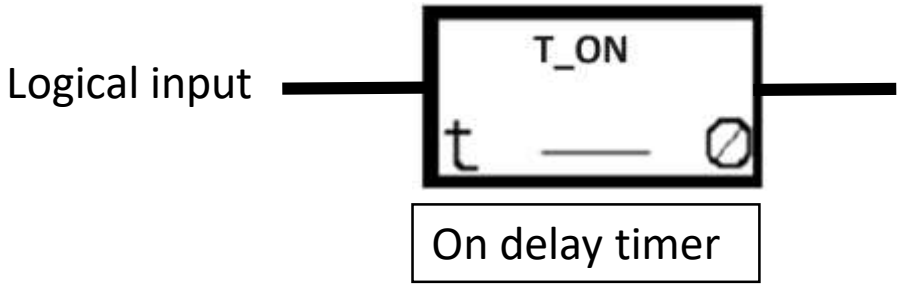
Électricité de France



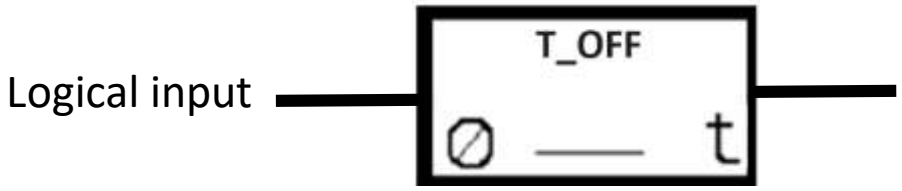
Generation of a test scenario

- We establish a sound and complete LTL encoding of the logical diagram.
- We use LTL model checking to verify the stability and generate the wanted test scenarios.
- We do a real case evaluation of our approach.

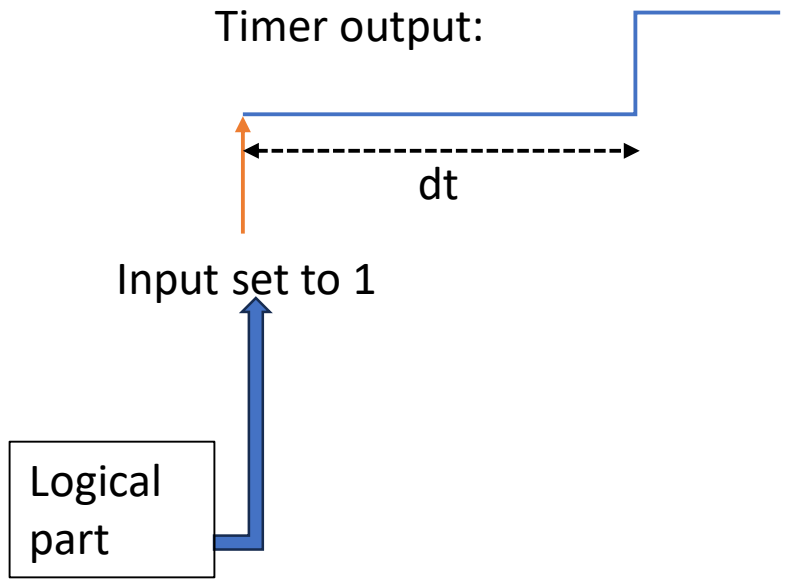
Future work



On delay timer



Off delay timer



Open question: How to encode timers into LTL formulas?

Thank you !

email: masfargandoura@lirmm.fr