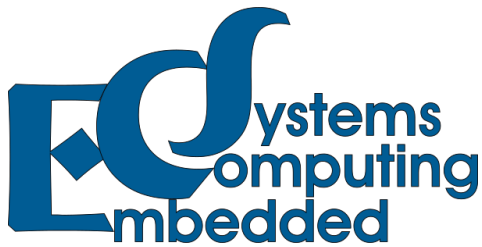
SeAL: A Provably Complete Fault-Injection Tool for Navigating Fault Tolerance in QDI Circuits

Raghda El Shehaby

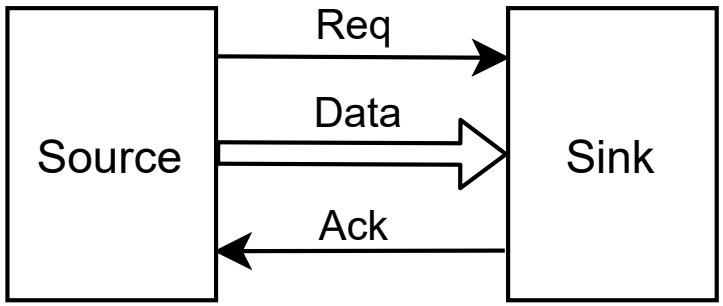
TU Wien
Embedded Computing Systems

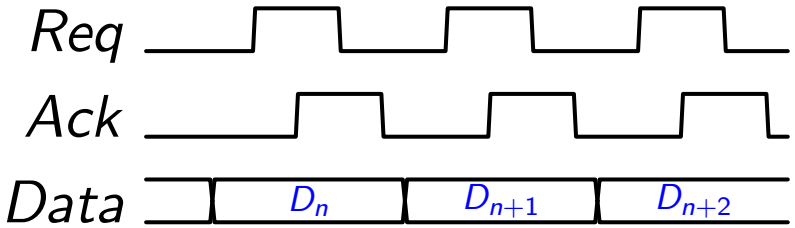


Outline of the Talk

Outline of the Talk

Asynchronous Design Style





because...

- Local handshake
- Low power consumption*
- Average-case performance
- **Inherent robustness against timing variations**

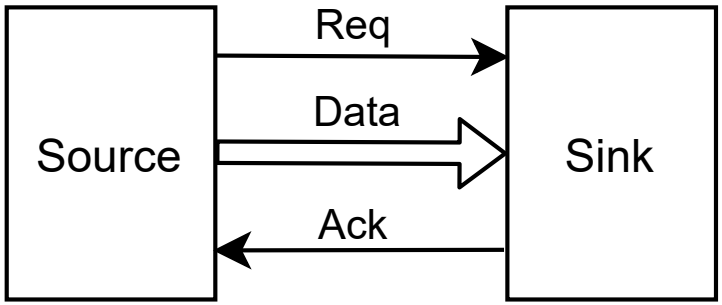
* Depends on the design style

however...

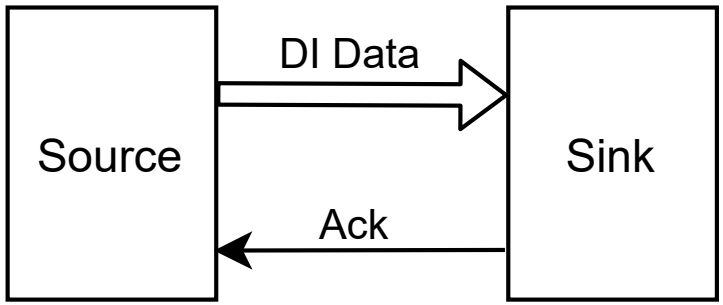
- More complex
- Area overhead
- Data encoding
- Limited tool-support

Asynchronous Design Style

- Closed-loop control
- Classified according to the delay model they follow



Bundled Data



Delay Insensitive

Fault Tolerance in Digital Circuits

- Synchronous circuits $\rightarrow$ state-based
  - Natural resilience against transient faults
  - Sensitivity against delay-related faults
- Asynchronous circuits $\rightarrow$ event-based
  - Strong robustness against timing variations
  - Sensitivity to extra transitions

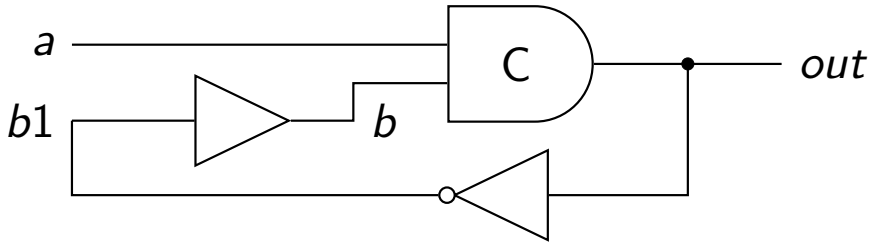
Fault Tolerance in Digital Circuits

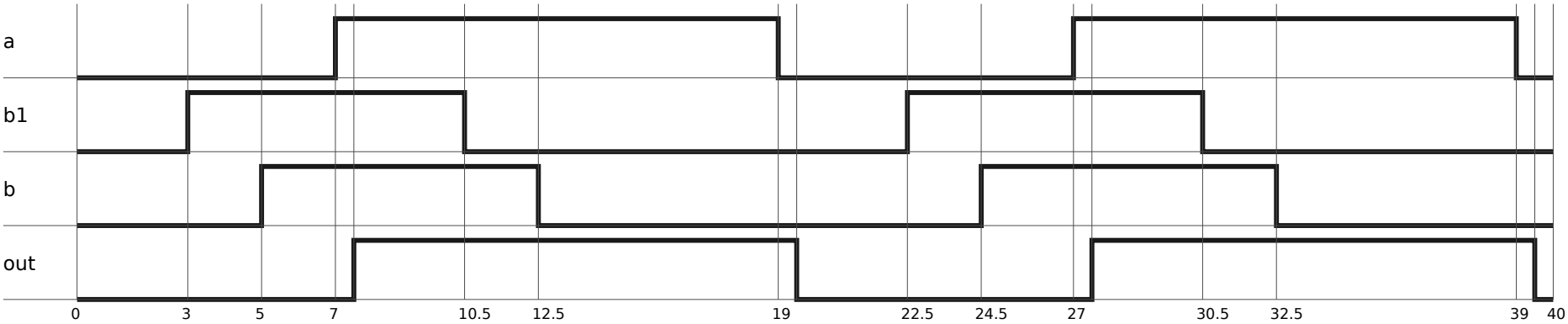
- Synchronous circuits $\rightarrow$ state-based
  - Natural resilience against transient faults
  - Sensitivity against delay-related faults
- Asynchronous circuits $\rightarrow$ event-based
  - Strong robustness against timing variations
  - Sensitivity to extra transitions

Which signals and at which times is an asynchronous circuit susceptible to a transient/permanent fault?

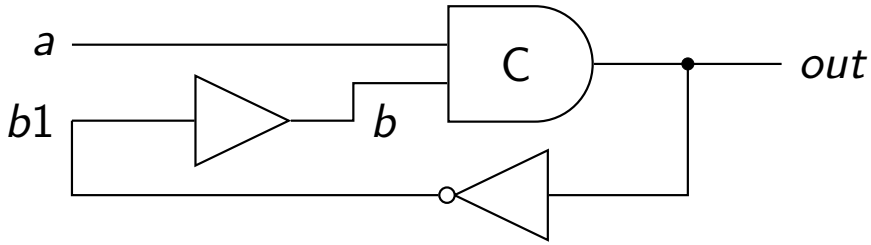
Outline of the Talk

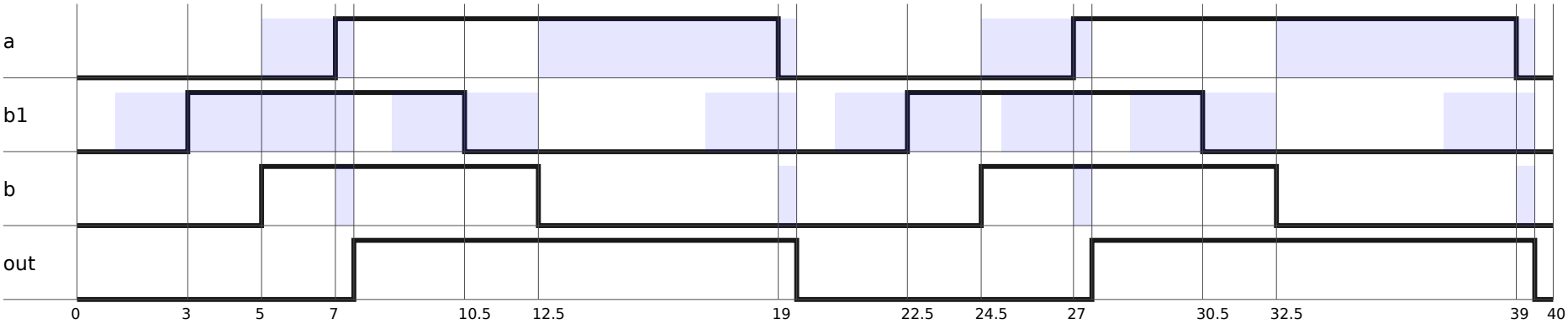
Transient Fault Effects



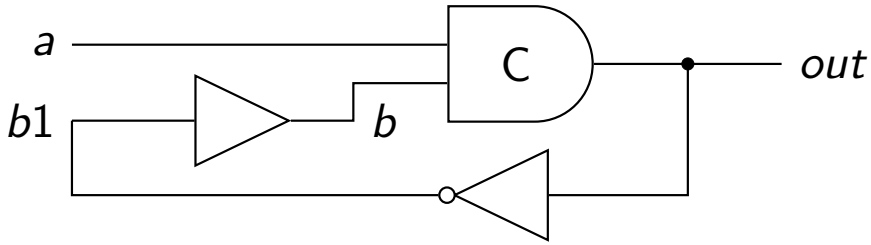


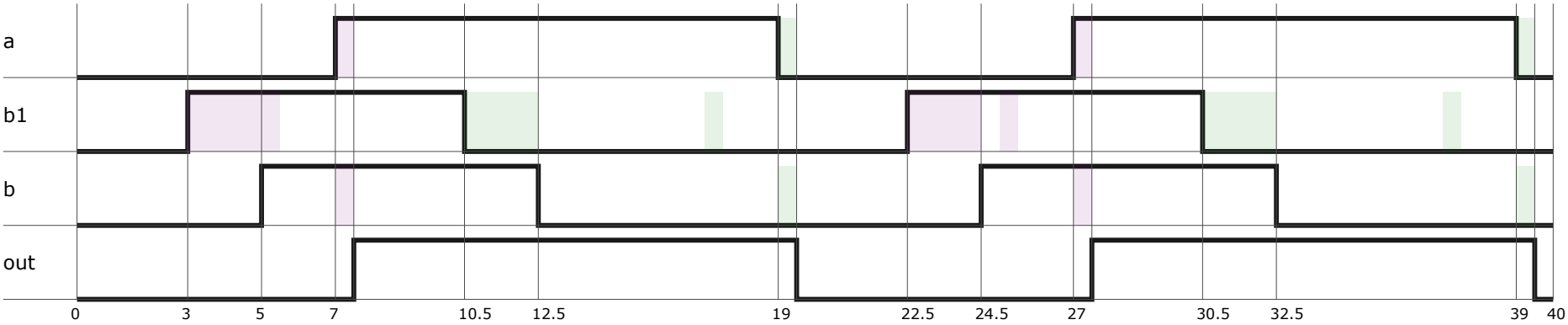
Transient Fault Effects



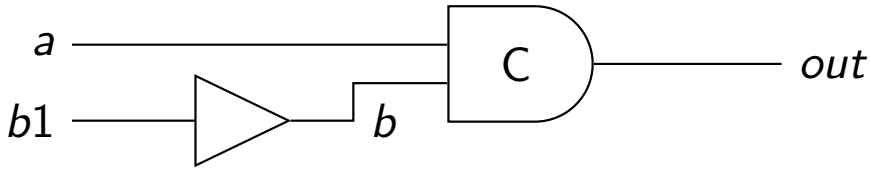


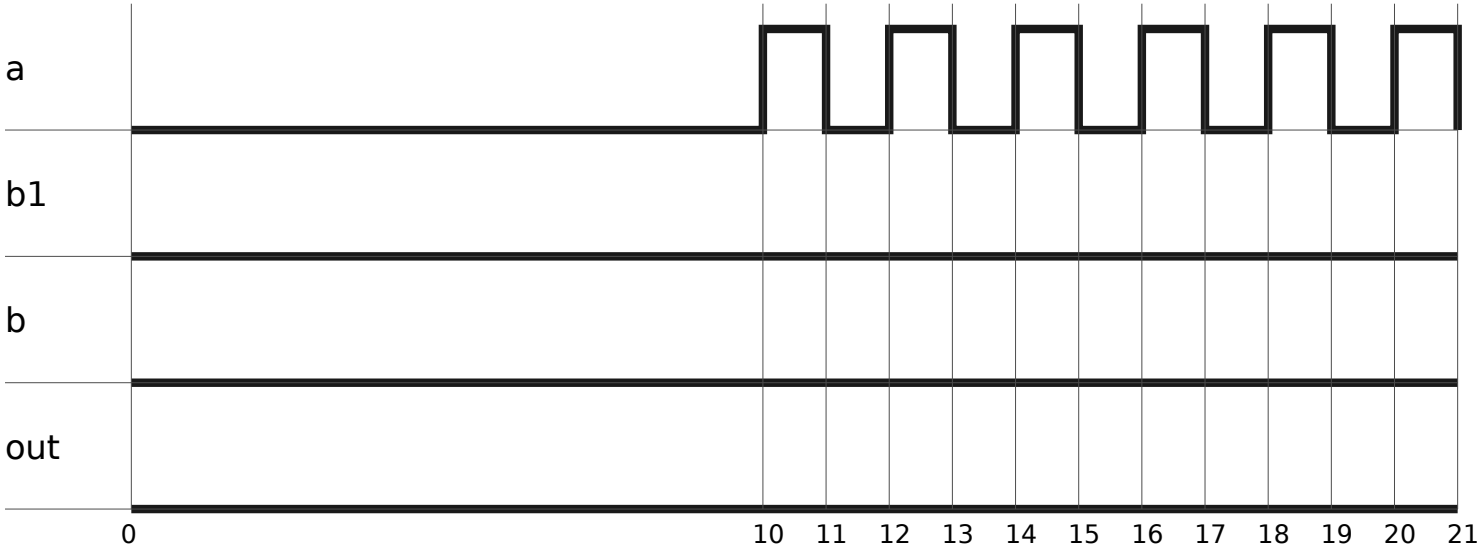
Stuck-at Fault Effects



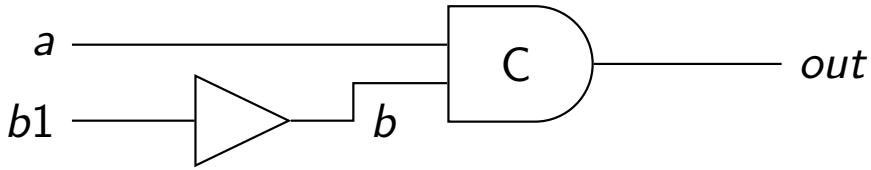


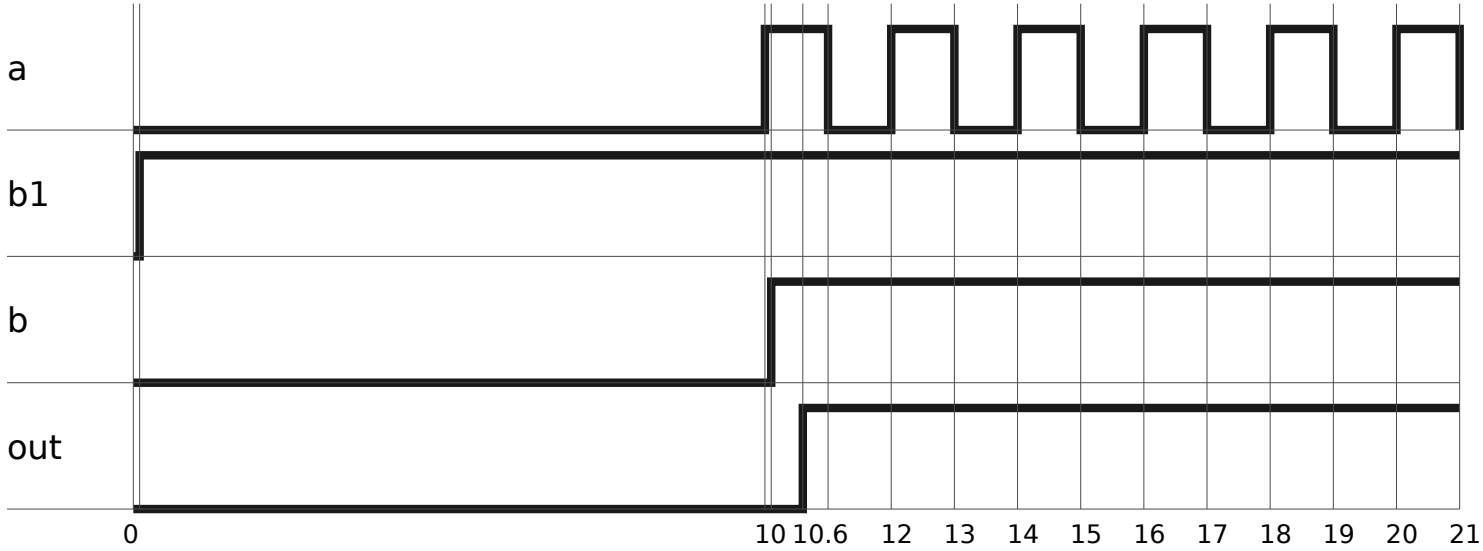
Stuck-at Fault Effects



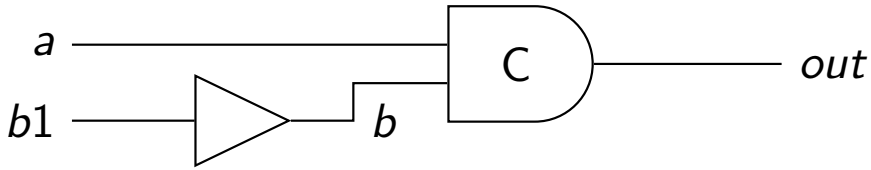


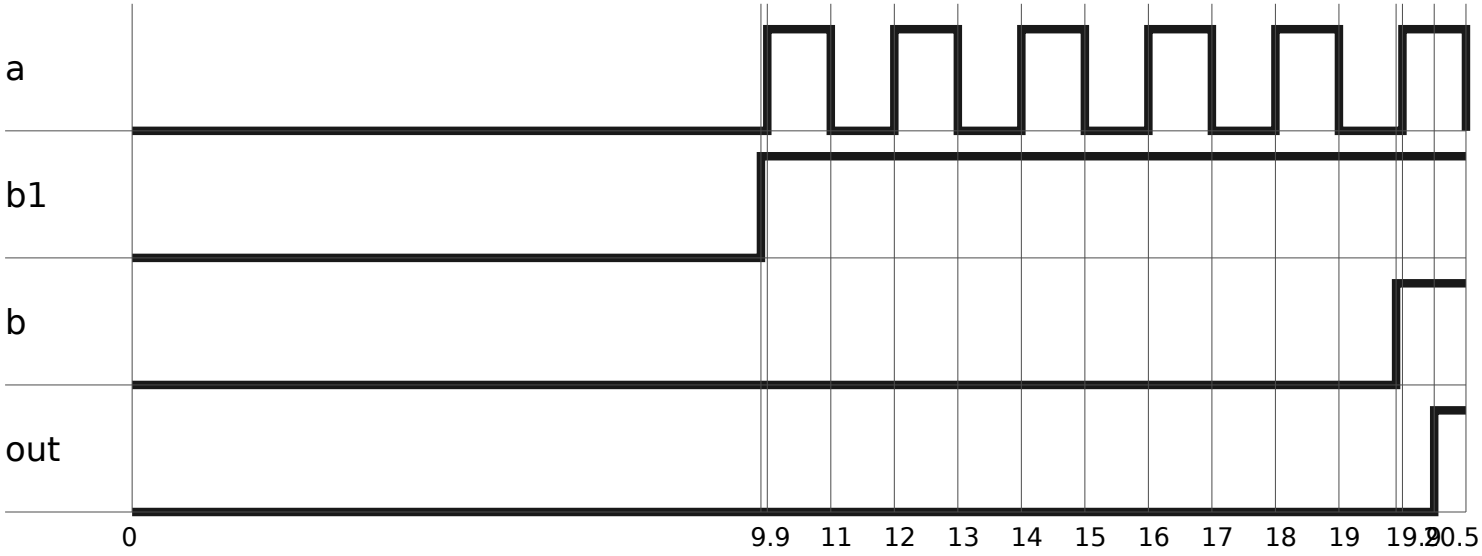
Stuck-at Fault Effects



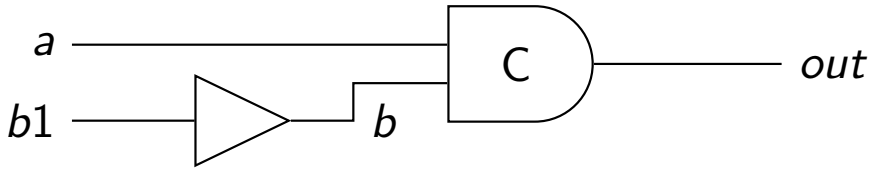


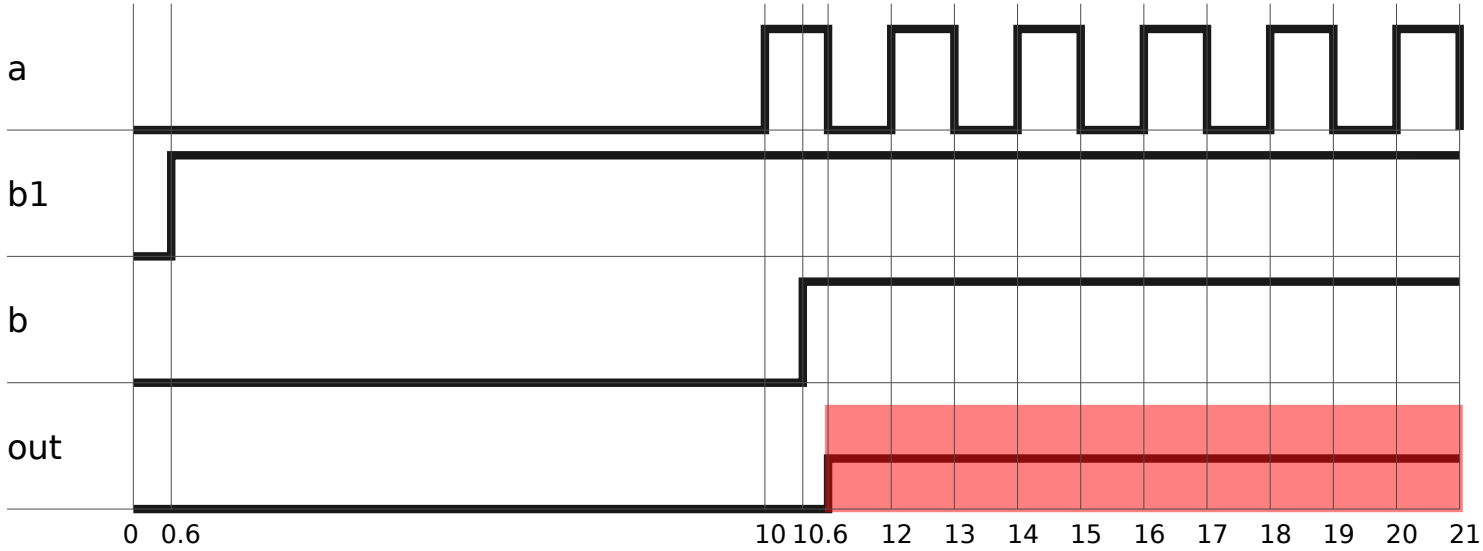
Stuck-at Fault Effects



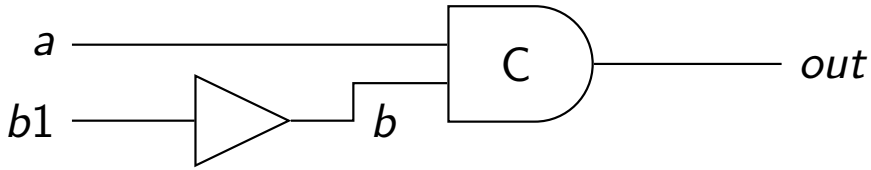


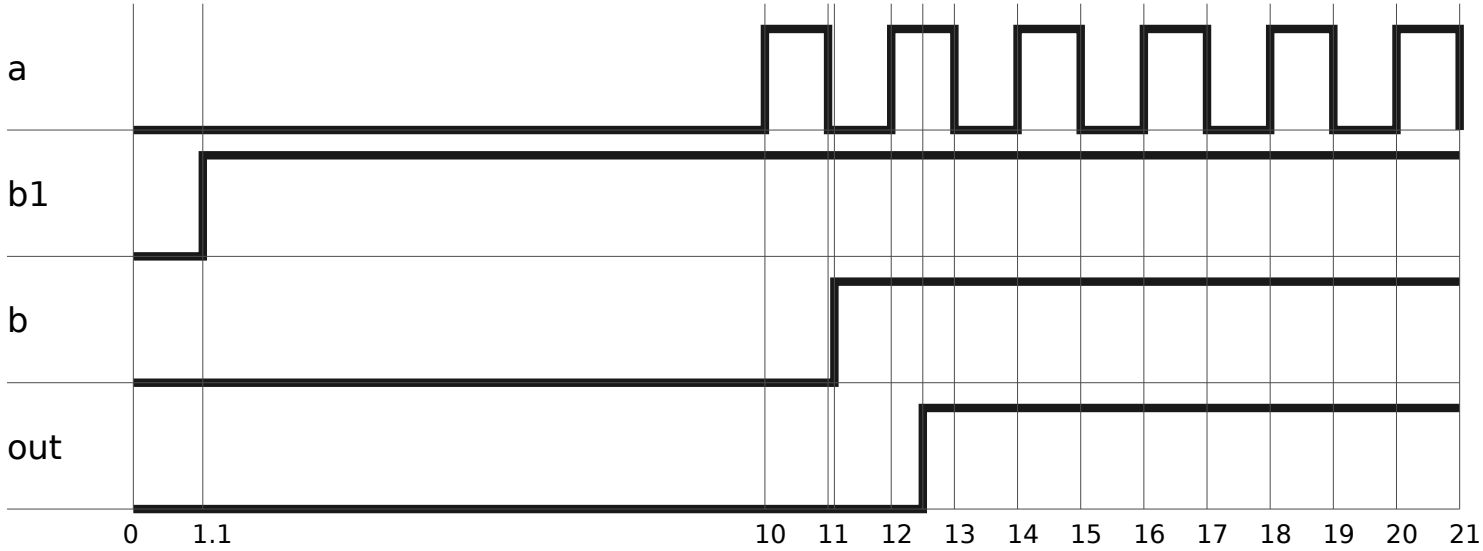
Stuck-at Fault Effects



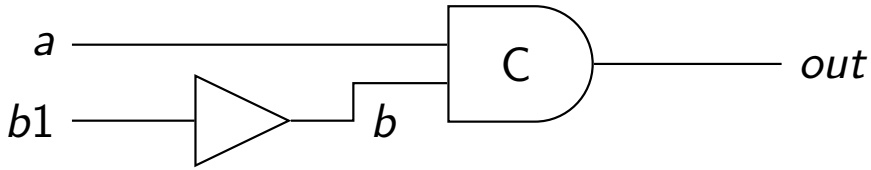


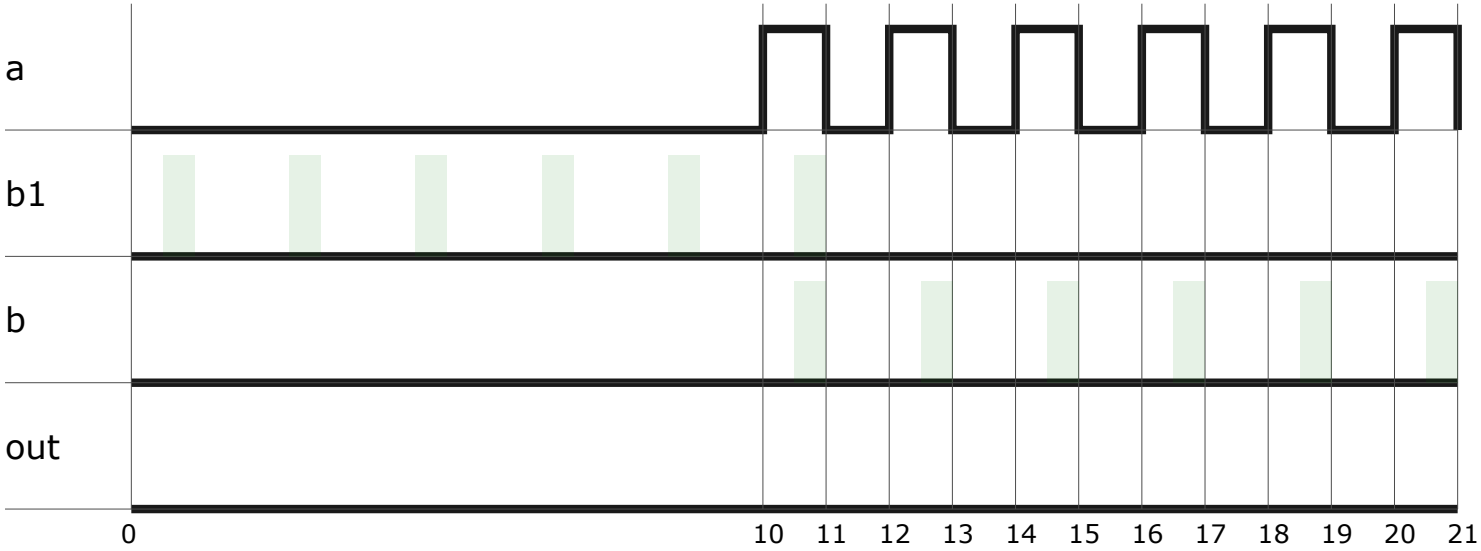
Stuck-at Fault Effects





Stuck-at Fault Effects





Goal

Build a formal framework to comprehensively identify vulnerable windows against transient and permanent faults

Goal

Build a formal framework to comprehensively identify vulnerable windows against transient and permanent faults

(E., Függer, Steininger, FORMATS'23)
(E., Függer, Huemer, Steininger, ASYNC'25)

Outline of the Talk

Modeling the Circuit and its Environment

- Gates are specified in terms of production rules

$$G \rightarrow s = 1\ [d] \quad \text{and} \quad G \rightarrow s = 0\ [d]$$

Modeling the Circuit and its Environment

- Gates are specified in terms of production rules

$$G \to s = 1\ [d] \quad \text{and} \quad G \to s = 0\ [d]$$

- A signal can have a value of the three-valued set $\mathbf{B}_{\mathsf{X}} = \{0, \mathsf{X}, 1\}$
  - X models non-binary, unstable or metastable values

Modeling the Circuit and its Environment

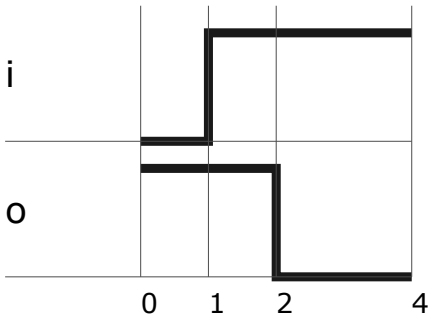
- Gates are specified in terms of production rules

$$G \rightarrow s = 1\ [d] \quad \text{and} \quad G \rightarrow s = 0\ [d]$$

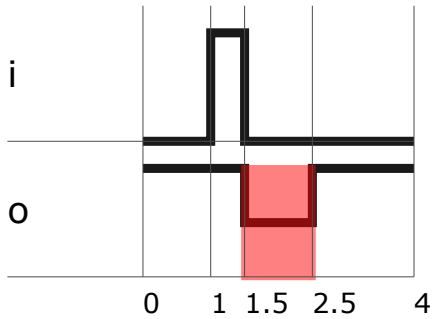
- A signal can have a value of the three-valued set $\mathbf{B}_X = \{0, X, 1\}$
  - X models non-binary, unstable or metastable values

Example: Inverter, with output initial value '1'

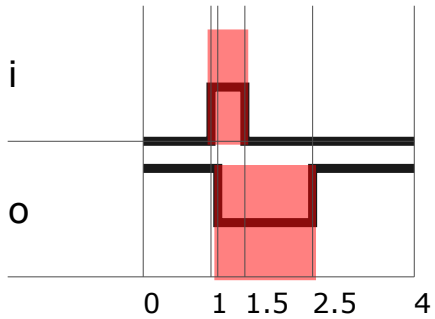
$$i \rightarrow o = 0\ [1.0]$$
$$\neg i \rightarrow o = 1\ [1.0]$$



(a) fault-free



(b) X-generation



(c) X-propagation

Modeling the Circuit and its Environment

- Fault equivalence
  2 faults are said to be equivalent when the sequence of circuit states is the same in both executions

Outline of the Talk

SeAL Fault Insertion Tool

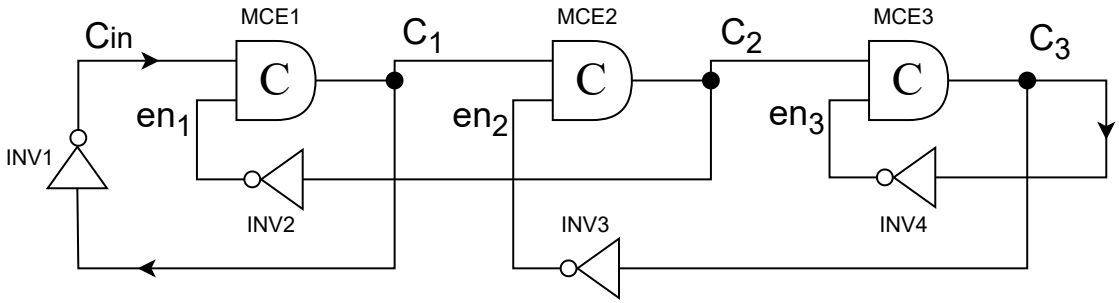
Sensitivity analyzer for Asynchronous Logic (SeAL), publicly available on GitHub, extended implementation to include:

- Both transient and stuck-at fault injection
- Plotting and reporting features
- Larger asynchronous circuits with pipeline stages and logic functions in between, synthesized and translated from pypr
  `https://gitlab.ecs.tuwien.ac.at/eda/pypr`
- High-level tokens as input
  - data value for each input in circuit
  - broken down to dual-encodings and further to single-rail signals

Supporting 4-phase communication protocol with completion detection controlling token flow through pipeline

SeAL Fault Insertion Tool

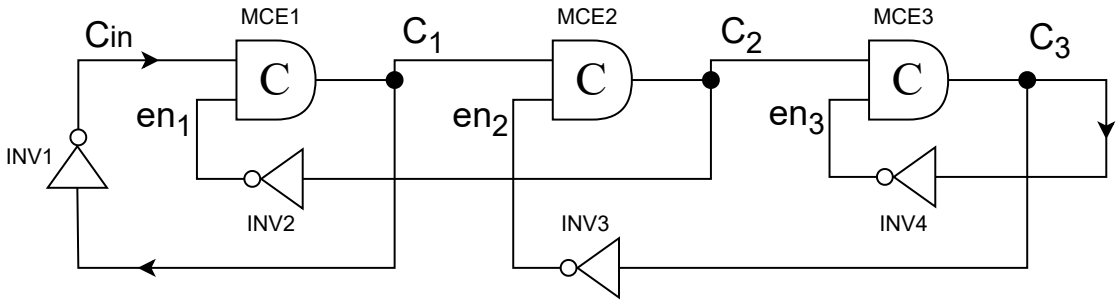
Case Study: Linear Muller Pipeline

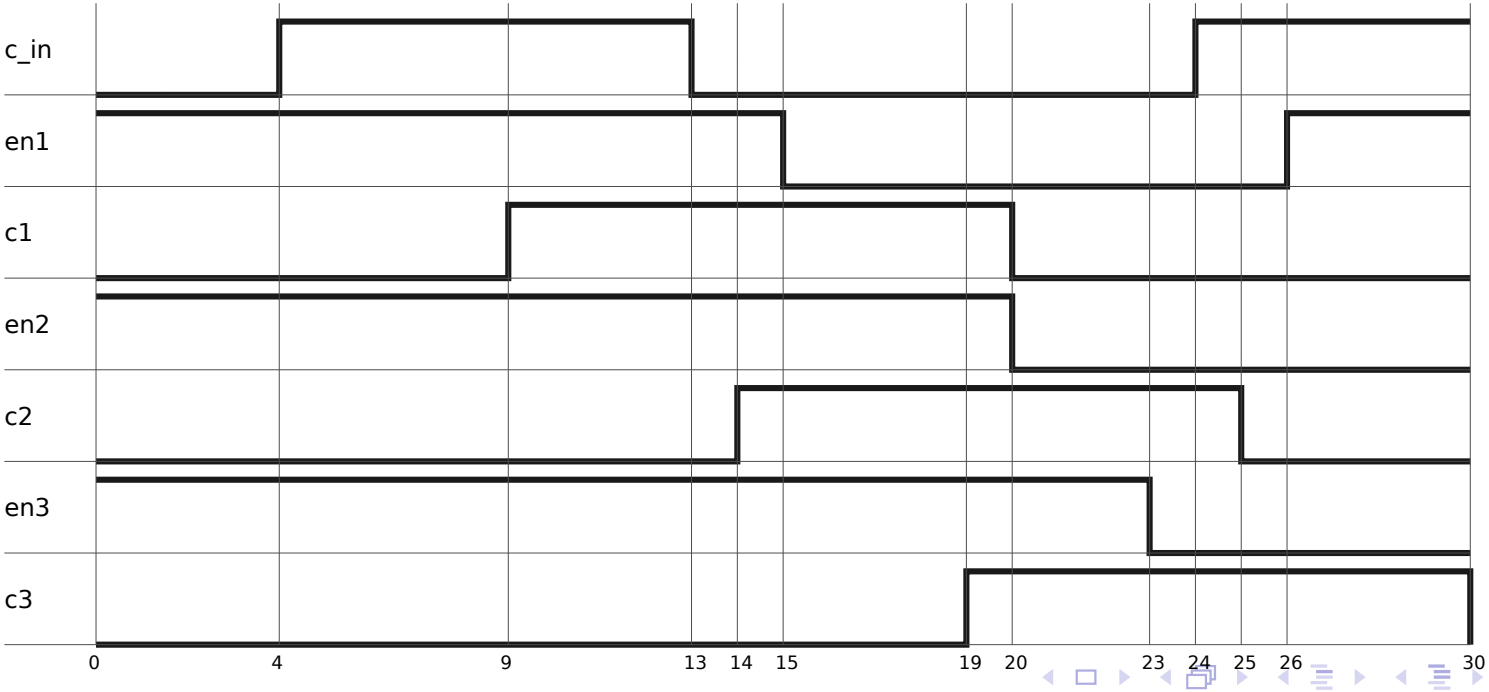


$$c1 \rightarrow cin = 0\ [4] \quad \text{and} \quad \neg c1 \rightarrow cin = 1\ [4]$$
$$c2 \rightarrow en1 = 0\ [1] \quad \text{and} \quad \neg c2 \rightarrow en1 = 1\ [1]$$
$$cin \wedge en1 \rightarrow c1 = 1\ [5] \quad \text{and} \quad \neg cin \wedge \neg en_1 \rightarrow c1 = 0\ [5]$$
$$c3 \rightarrow en2 = 0\ [1] \quad \text{and} \quad \neg c3 \rightarrow en2 = 1\ [1]$$
$$c1 \wedge en2 \rightarrow c2 = 1\ [5] \quad \text{and} \quad \neg c1 \wedge \neg en2 \rightarrow c2 = 0\ [5]$$
$$c3 \rightarrow en3 = 0\ [4] \quad \text{and} \quad \neg c3 \rightarrow en3 = 1\ [4]$$
$$c2 \wedge en3 \rightarrow c3 = 1\ [5] \quad \text{and} \quad \neg c2 \wedge \neg en3 \rightarrow c3 = 0\ [5]$$

SeAL Fault Insertion Tool

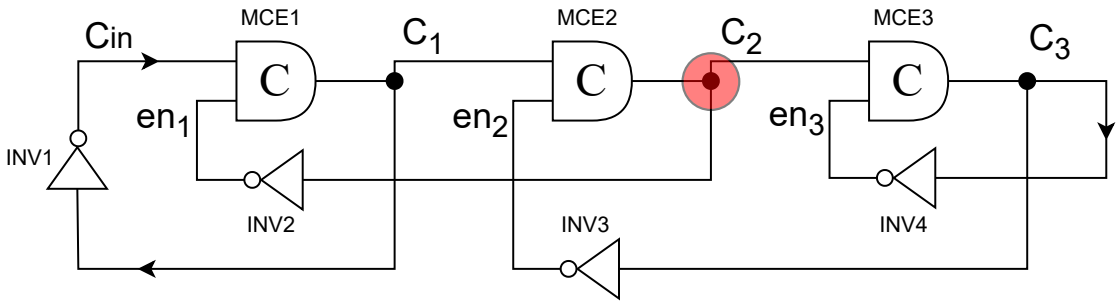
Case Study: Linear Muller Pipeline





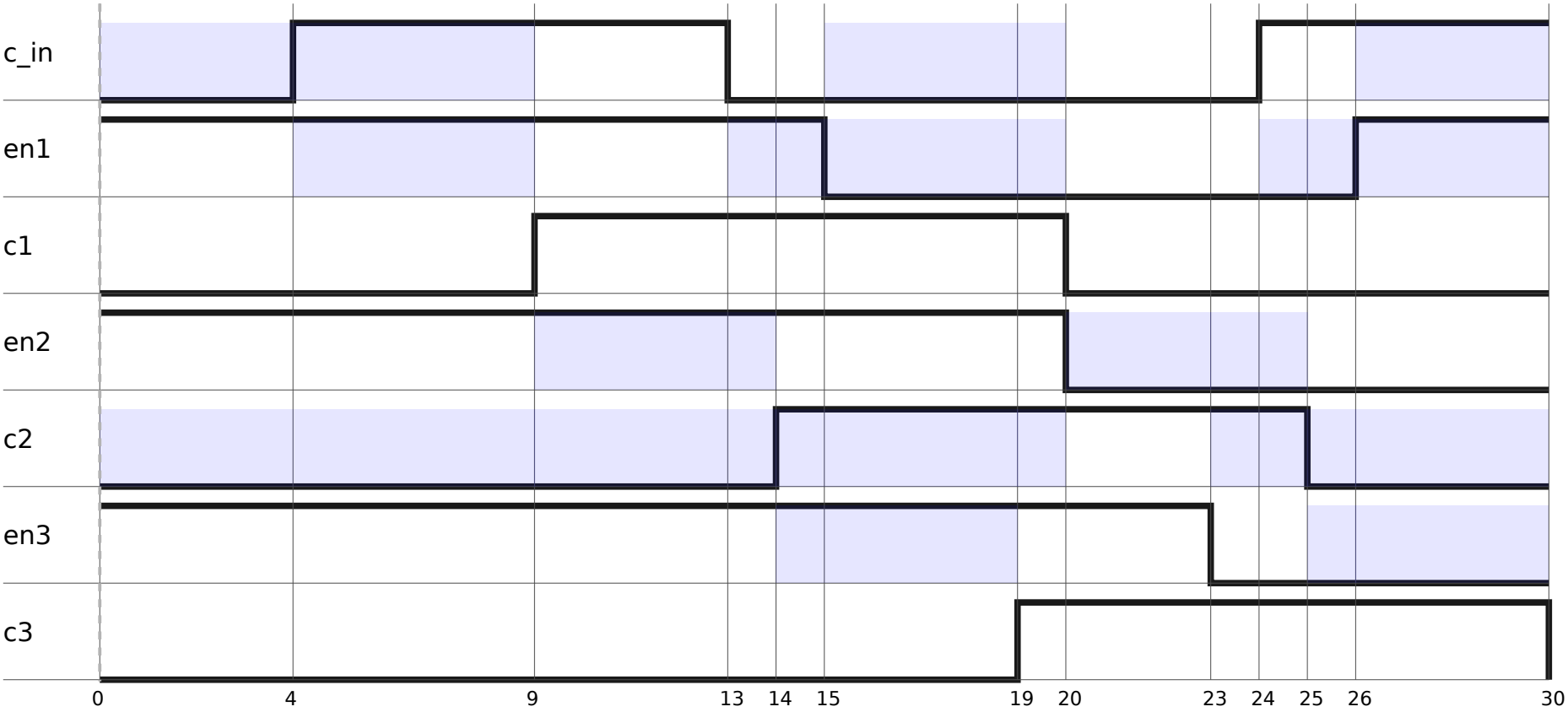
SeAL Fault Insertion Tool

Case Study: Linear Muller Pipeline



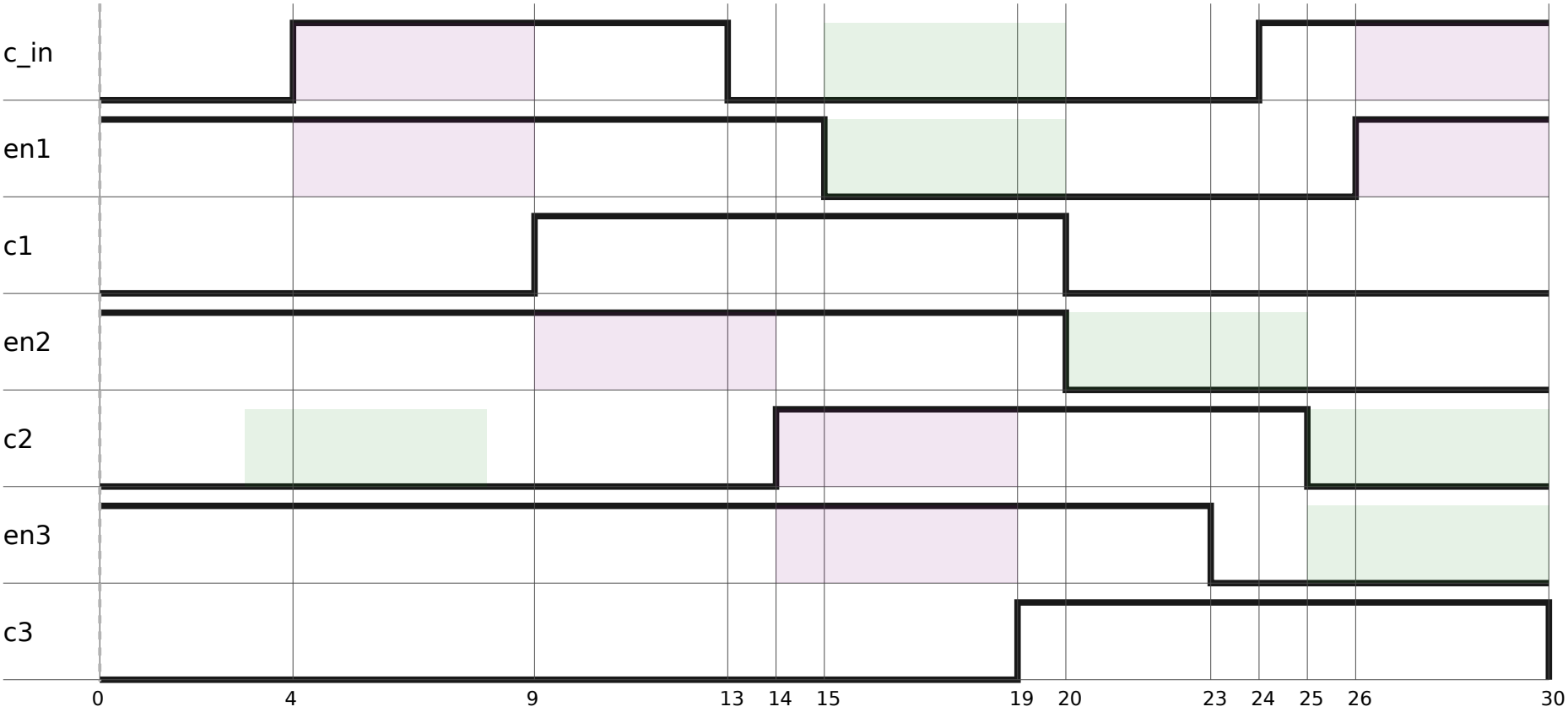
Equivalence of Transient Faults

Case Study: Linear Muller Pipeline



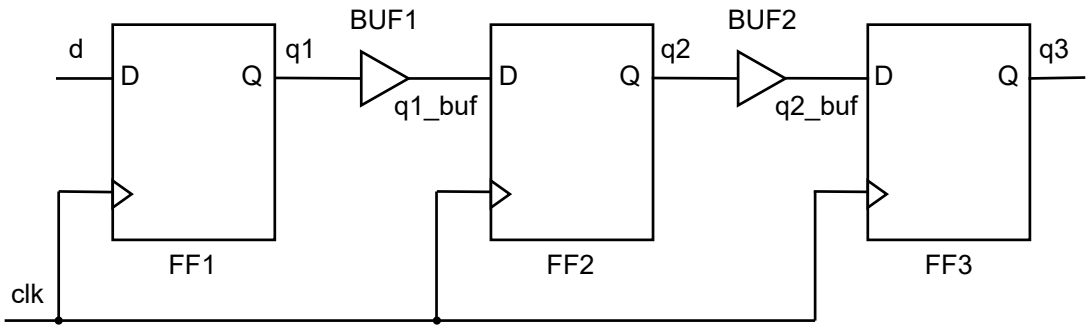
Equivalence of Stuck-at Faults

Case Study: Linear Muller Pipeline



SeAL Fault Insertion Tool

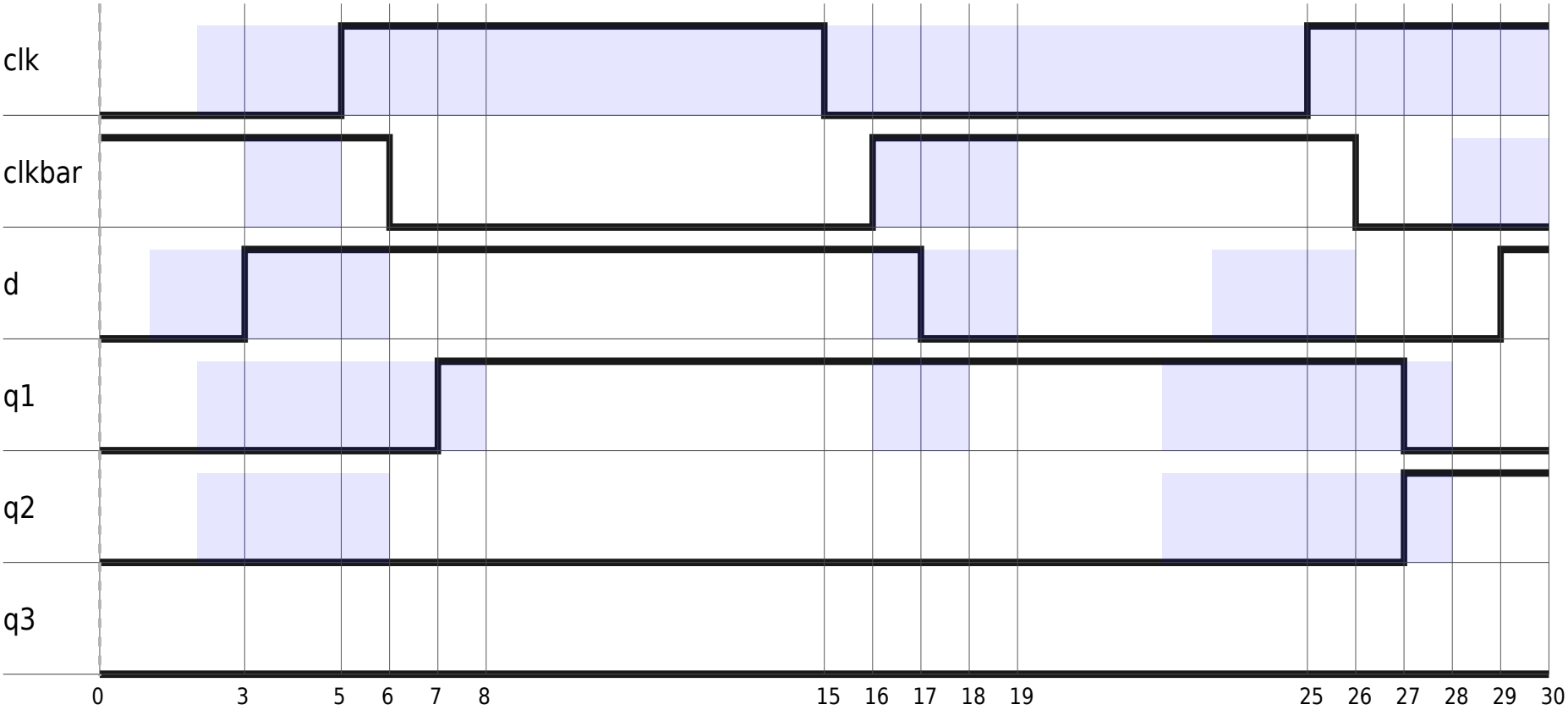
Case Study: Linear Synchronous Pipeline



$en \wedge d \rightarrow q = 1$ [2] and $en \wedge \neg d \rightarrow q = 0$ [2]

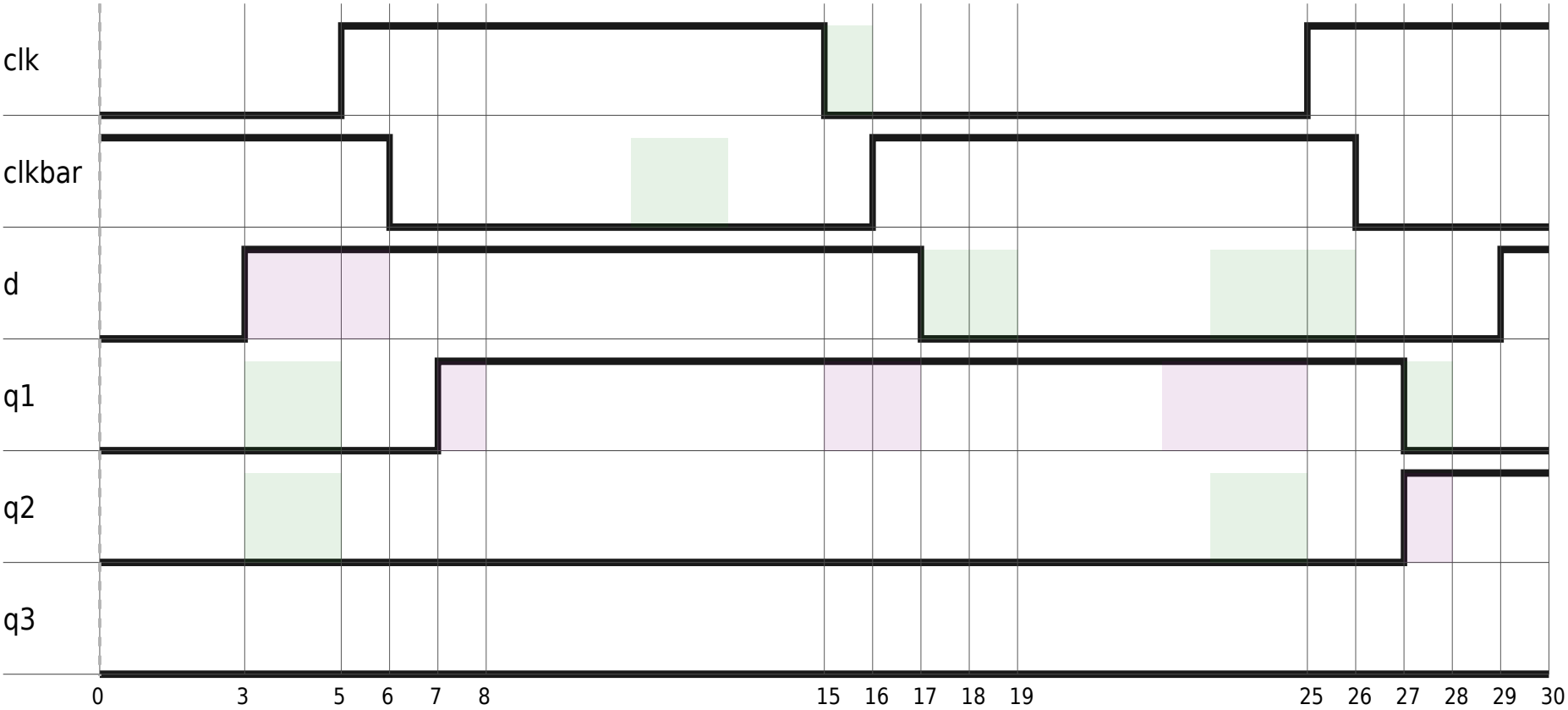
Equivalence of Transient Faults

Case Study: Linear Synchronous Pipeline



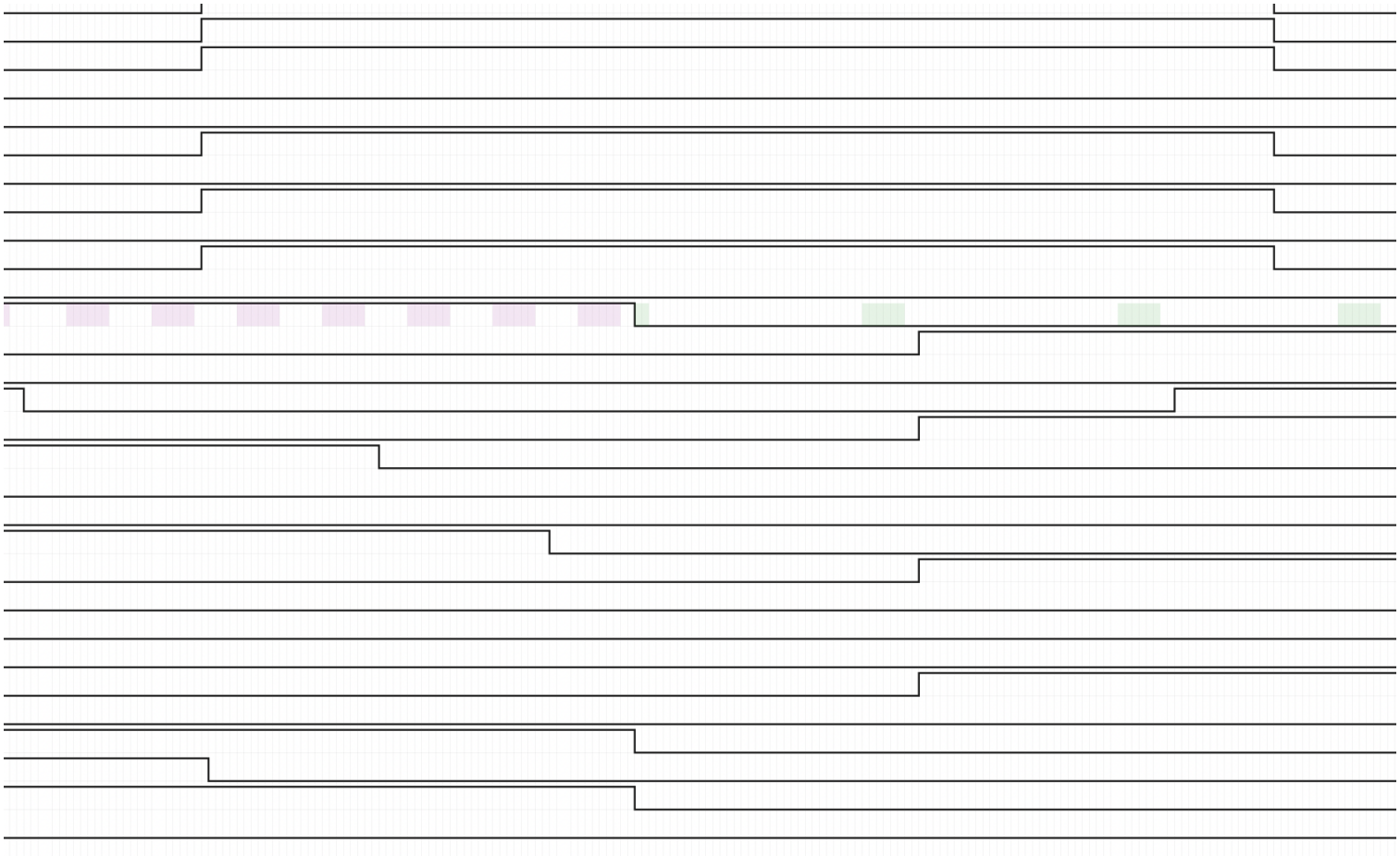
Equivalence of Stuck-at Faults

Case Study: Linear Synchronous Pipeline

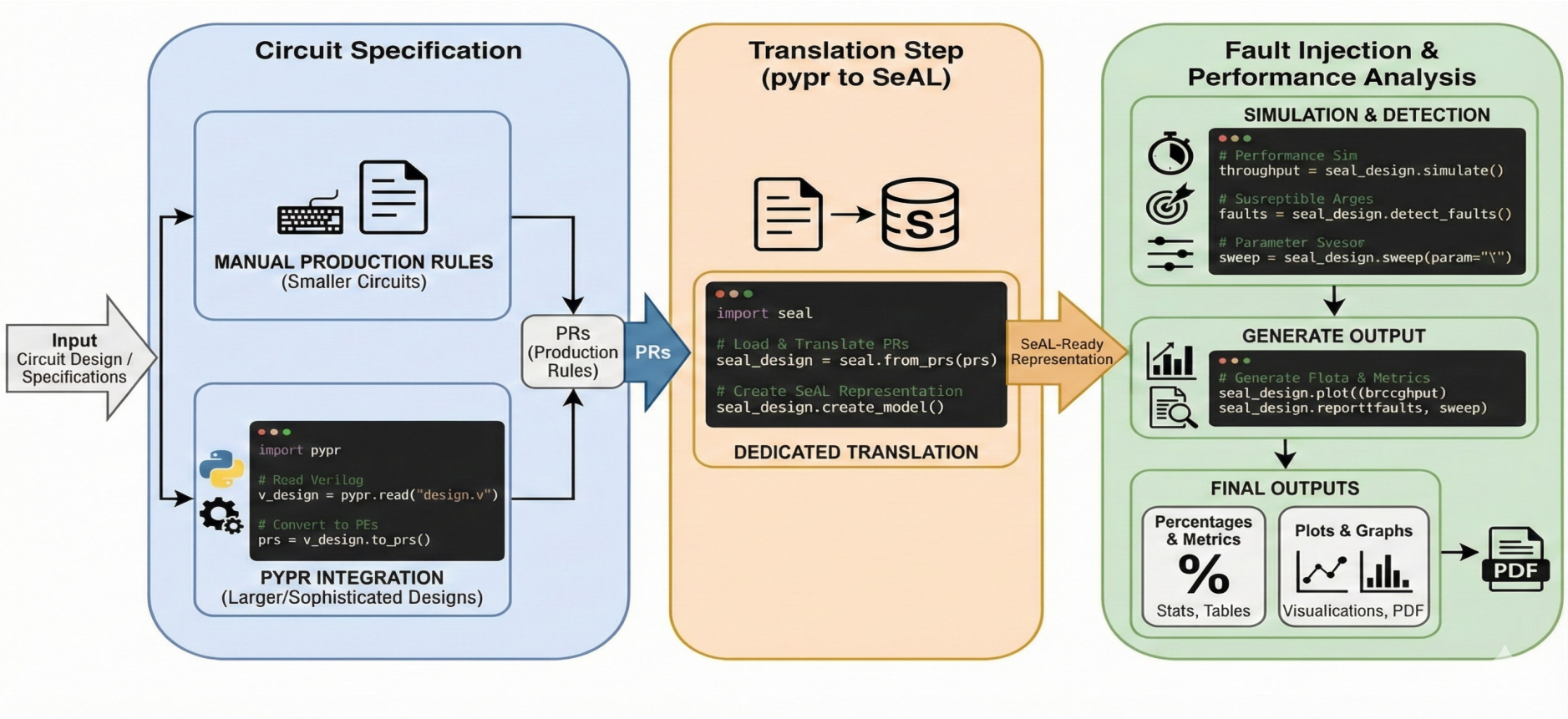


Equivalence of Stuck-at Faults: Non-Toy Circuit

Equivalence of Stuck-at Faults: Non-Toy Circuit



SeAL Workflow



Outline of the Talk

Summary

Python-based tool (SeAL) that systematically explores susceptibility windows

`https://github.com/mfuegger/SeAL`

Summary

Python-based tool (SeAL) that systematically explores susceptibility windows
`https://github.com/mfuegger/SeAL`

- Injects at least 1 fault into each equivalence region
- Exhaustive check, based on equivalence of transient and stuck-at faults
- Complete because of proof: guaranteed to find even the smallest sucsceptible windows
  (E., Függer, Steininger, FORMATS'23)
  (E., Függer, Huemer, Steininger, ASYNC'25)