HSE
Occupational Health & Safety
and Environmental Protection unit

Introduction to Formal Methods in Digital Electronics Design and Verification

Application to complex highly-parametrizable, continuously operating PLDs

Hamza Boukabache, **Katharina Ceesay-Seitz**

Daniel Perrin, Sarath Kundumattathil Mohanan, Gael Ducos, Markus Widorski, Michel Pangallo, Doris Forkel-Wirth, Stefan Roesler

9th December 2021

Why do we need functional verification?

"Does this design do what is intended to do ?"

Goal :

Find **systematic** failures

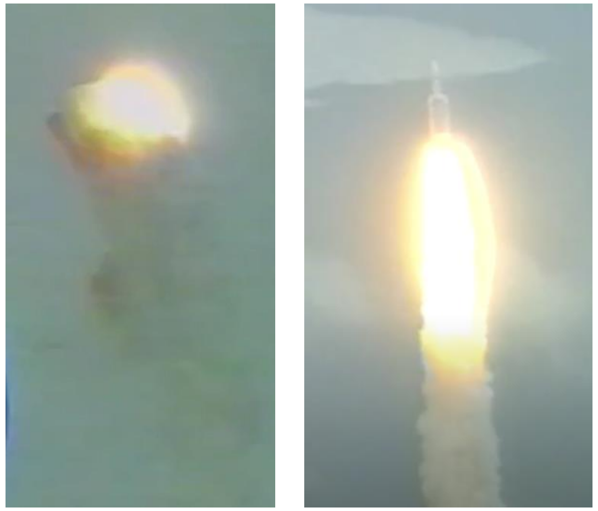
Methods :

Simulation, Formal, Emulation and Prototyping

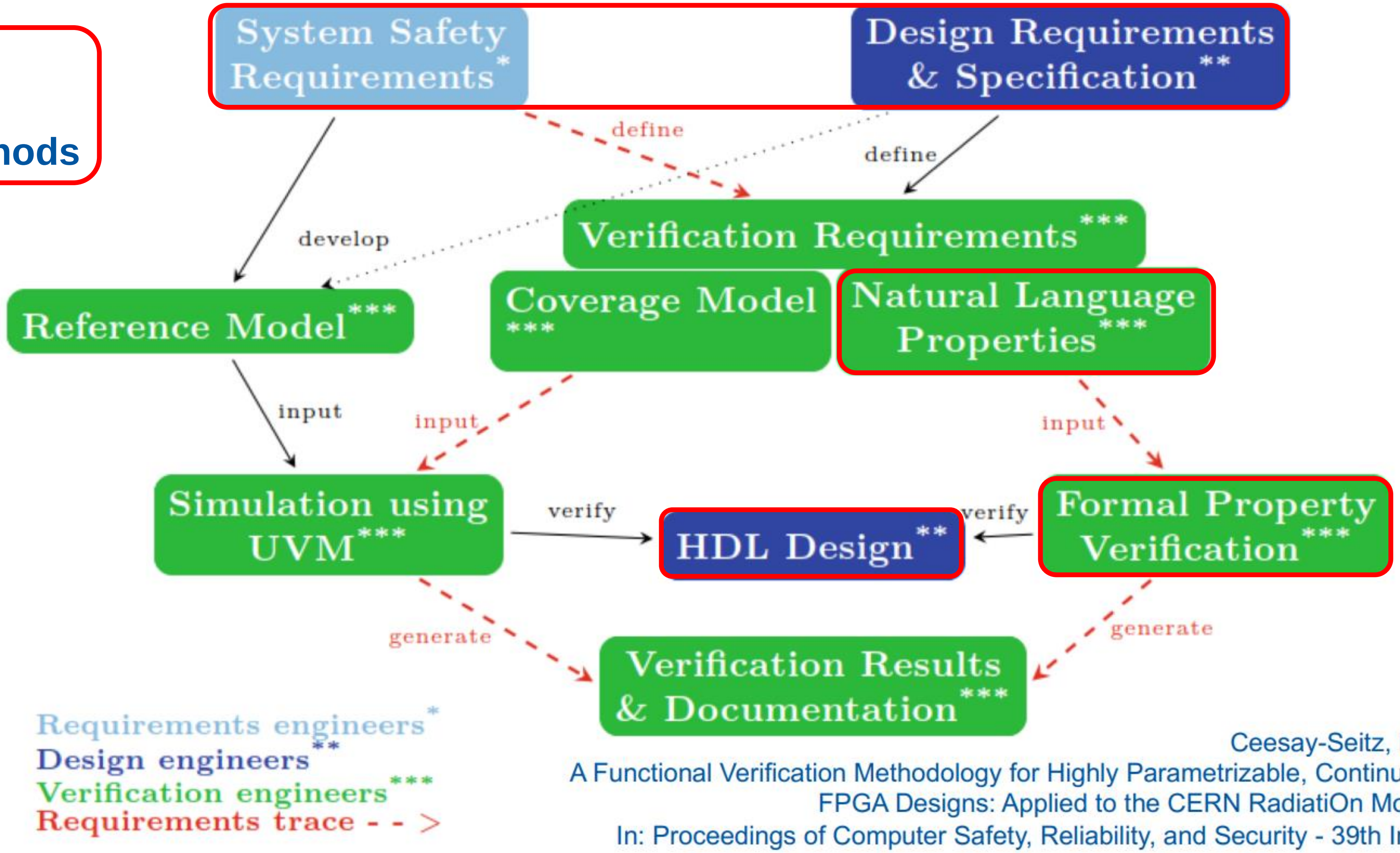
However :

No one of these methods can be used to completely verify an entire design or chip

- Formal, Simulation/Emulation and Prototyping complement each others
- Formal will find bugs that are missed by simulation and vise versa - They work very much together

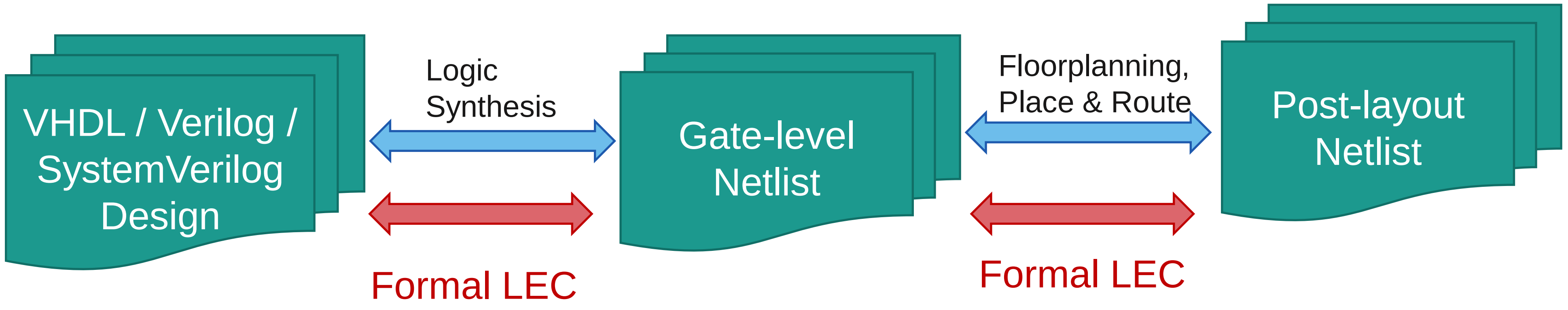

Flight 501, Ariane 5

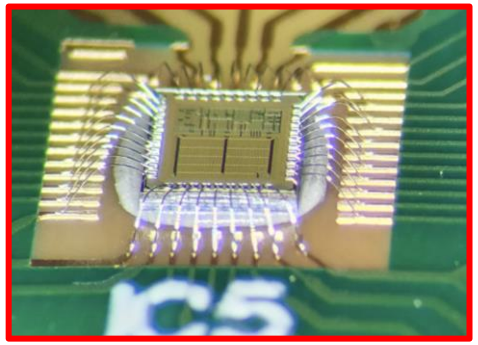
Application areas for **formal methods**


System Safety Requirements*
Design Requirements & Specification**
define
define
develop
Verification Requirements***
Reference Model***
Coverage Model ***
Natural Language Properties***
input
input
input
Simulation using UVM***
verify
HDL Design**
verify
Formal Property Verification***
generate
generate
Verification Results & Documentation***
Requirements engineers*
Design engineers**
Verification engineers***
Requirements trace - - >

Ceesay-Seitz, K., Boukabache, H., Perrin, D.:
A Functional Verification Methodology for Highly Parametrizable, Continuously Operating Safety-Critical FPGA Designs: Applied to the CERN RadiatiOn Monitoring Electronics (CROME).
In: Proceedings of Computer Safety, Reliability, and Security - 39th International Conference (2020)

Formal methods in electronics design – Automatic tools




ACCURATE 2 ASIC

Logic Equivalence Checking (LEC)/ Combinational Equivalence Checking

- Proof logical equivalence between models of different level of abstraction
- Map state elements (registers) of 2 versions of the design.
- Proof that combinational logic between any pair of states is equivalent in both models.

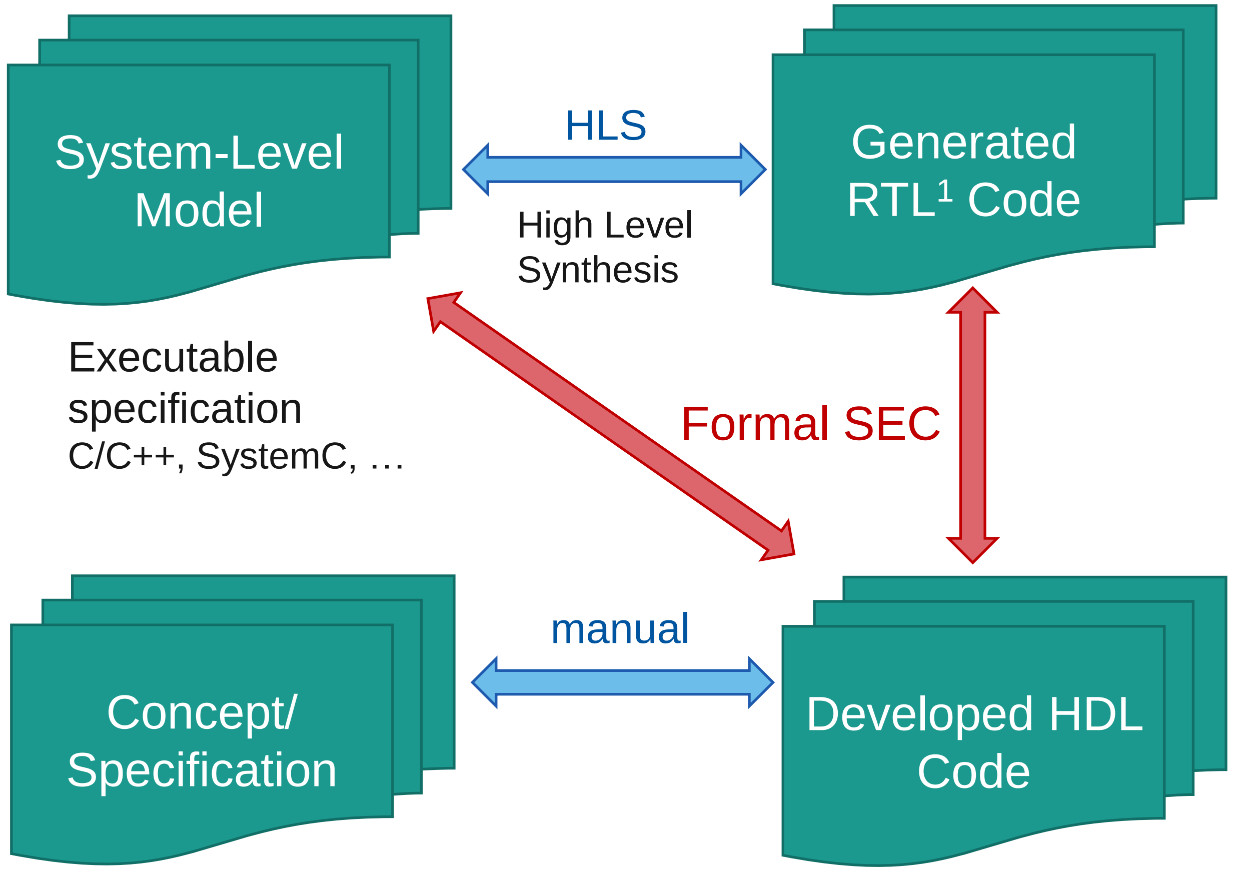
Commercial tools :

- Cadence Conformal EC
- Siemens FormalPro-LEC
- Synopsis Formality

Automatic verification Apps :

- X check
- Overflow checks
- Security/Safety checks
- …

Formal methods in electronics design – Formal Equivalence Checking



Executable specification
C/C++, SystemC, …

Requirements
Textual, model, …

Sequential Equivalence Checking (SEC)

- Proof sequential equivalence between models of different level of abstraction
- Assuming the inputs receive the same values, proof equivalence of outputs of both models at any time.
- Commercial tools
  - Cadence Jasper SEC App
  - Siemens SLEC
  - Synopsis VC Formal

[1]RTL = Register Transfer Level, written in HDL (Hardware Description Language), e.g. VHDL, Verilog

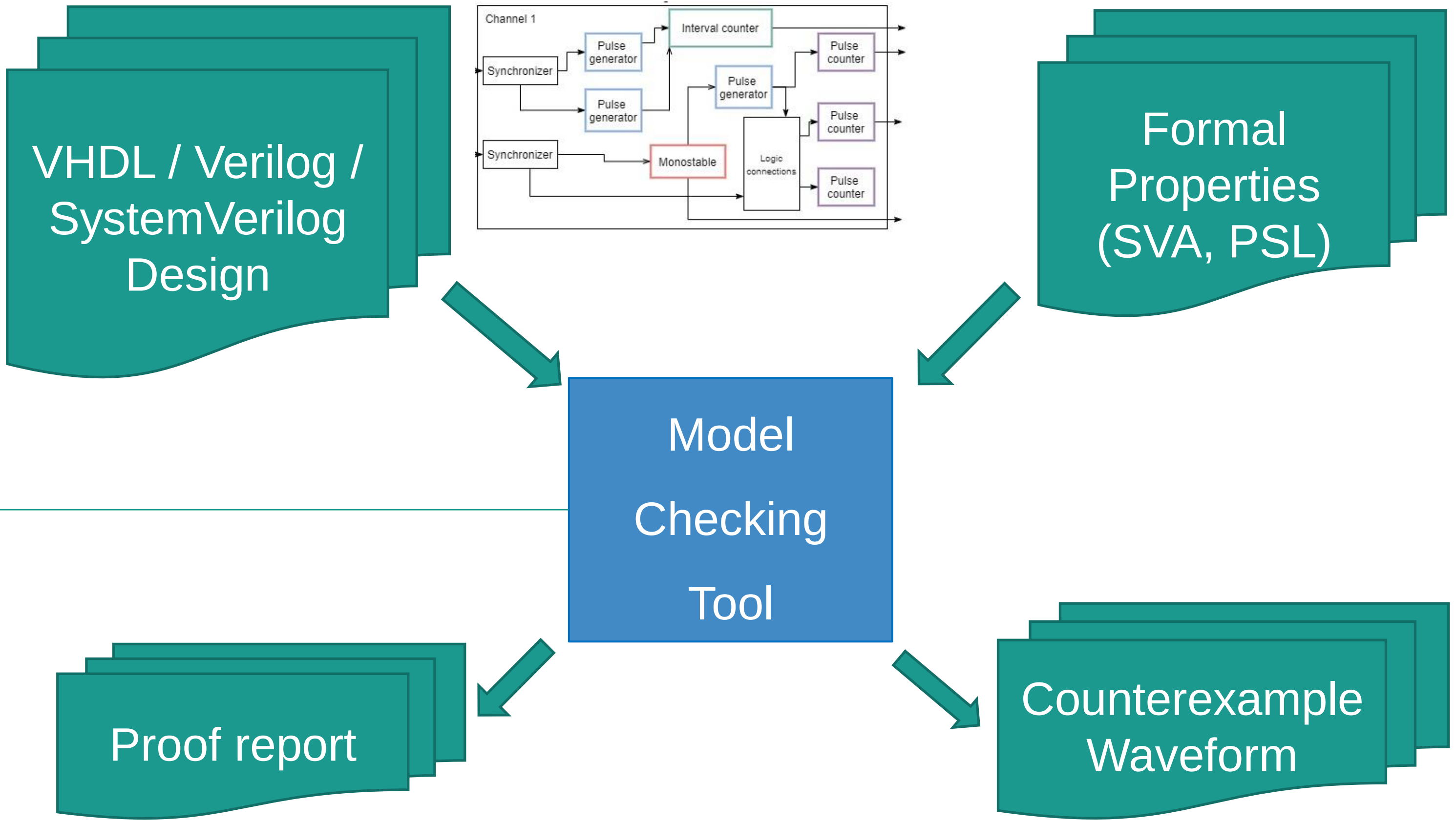
Formal Property Verification – Model Checking – User Perspective

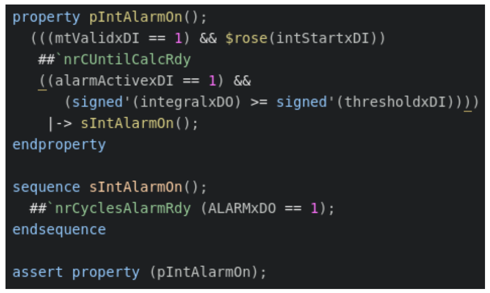
Commercial tools:

- Cadence Jasper Gold
- Siemens PropCheck
- Synopsis VC Formal
- …

Open-source tools:

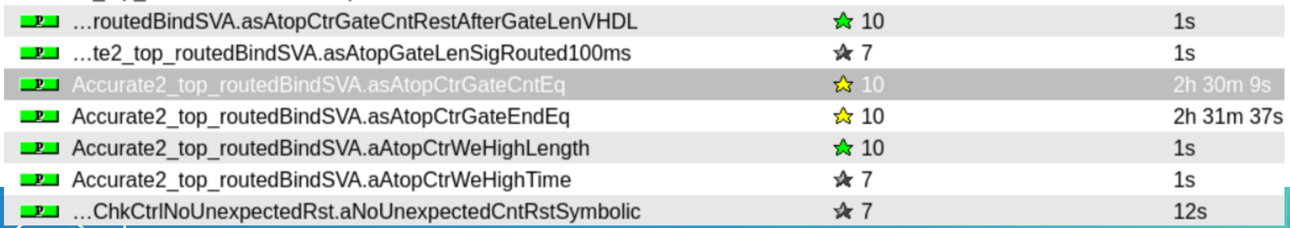
- SymbiYosys,
- EBMC, …

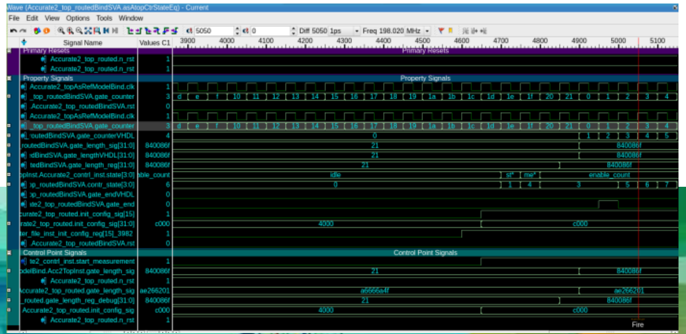



```
property pIntAlarmOn();
  (((mtValidxDI == 1) && $rose(intStartxDI))
   ##`nrCUntilCalcRdy
   ((alarmActivexDI == 1) &&
      (signed'(integralxDO) >= signed'(thresholdxDI))))
    |-> sIntAlarmOn();
endproperty

sequence sIntAlarmOn();
  ##`nrCyclesAlarmRdy (ALARMxDO == 1);
endsequence

assert property (pIntAlarmOn);
```


P	…routedBindSVA.asAtopCtrGateCntRestAfterGateLenVHDL	☆ 10	1s
P	…te2_top_routedBindSVA.asAtopGateLenSigRouted100ms	☆ 7	1s
P	Accurate2_top_routedBindSVA.asAtopCtrGateCntEq	☆ 10	2h 30m 9s
P	Accurate2_top_routedBindSVA.asAtopCtrGateEndEq	☆ 10	2h 31m 37s
P	Accurate2_top_routedBindSVA.aAtopCtrWeHighLength	☆ 10	1s
P	Accurate2_top_routedBindSVA.aAtopCtrWeHighTime	☆ 7	1s
P	…ChkCtrlNoUnexpectedRst.aNoUnexpectedCntRstSymbolic	☆ 7	12s



Formal Property Verification – Model Checking – Behind the Scene

Verification engineer states properties in :

- Linear Temporal Logic (LTL), e.g. as SystemVerilog Assertions or in Property Specification Language,
- Computational Tree Logic (CTL), …

User

Formal models can be created manually

- Petri nets, state machines, (timed) automata, …

Formal Tool

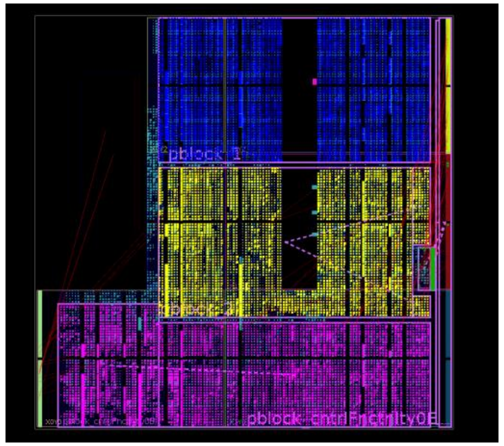
Formal model created by tool from HDL design

- Kripke structures, Binary Decision Diagrams (BDDs), …

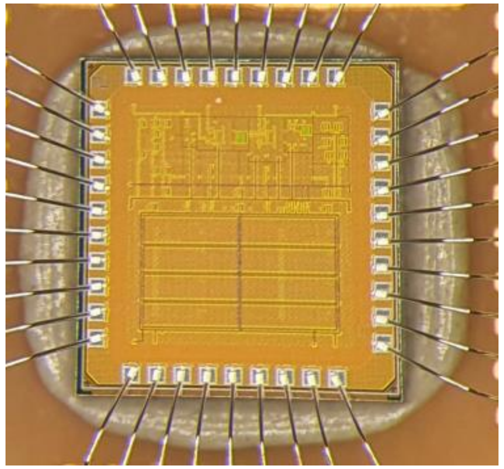
Model Checking **Algorithms inside the tools :**

- Boolean Satisfiability Problem (SAT), Satisfiability Modulo Theories (SMT) solvers,
- Symbolic Trajectory Evaluation (STE), …

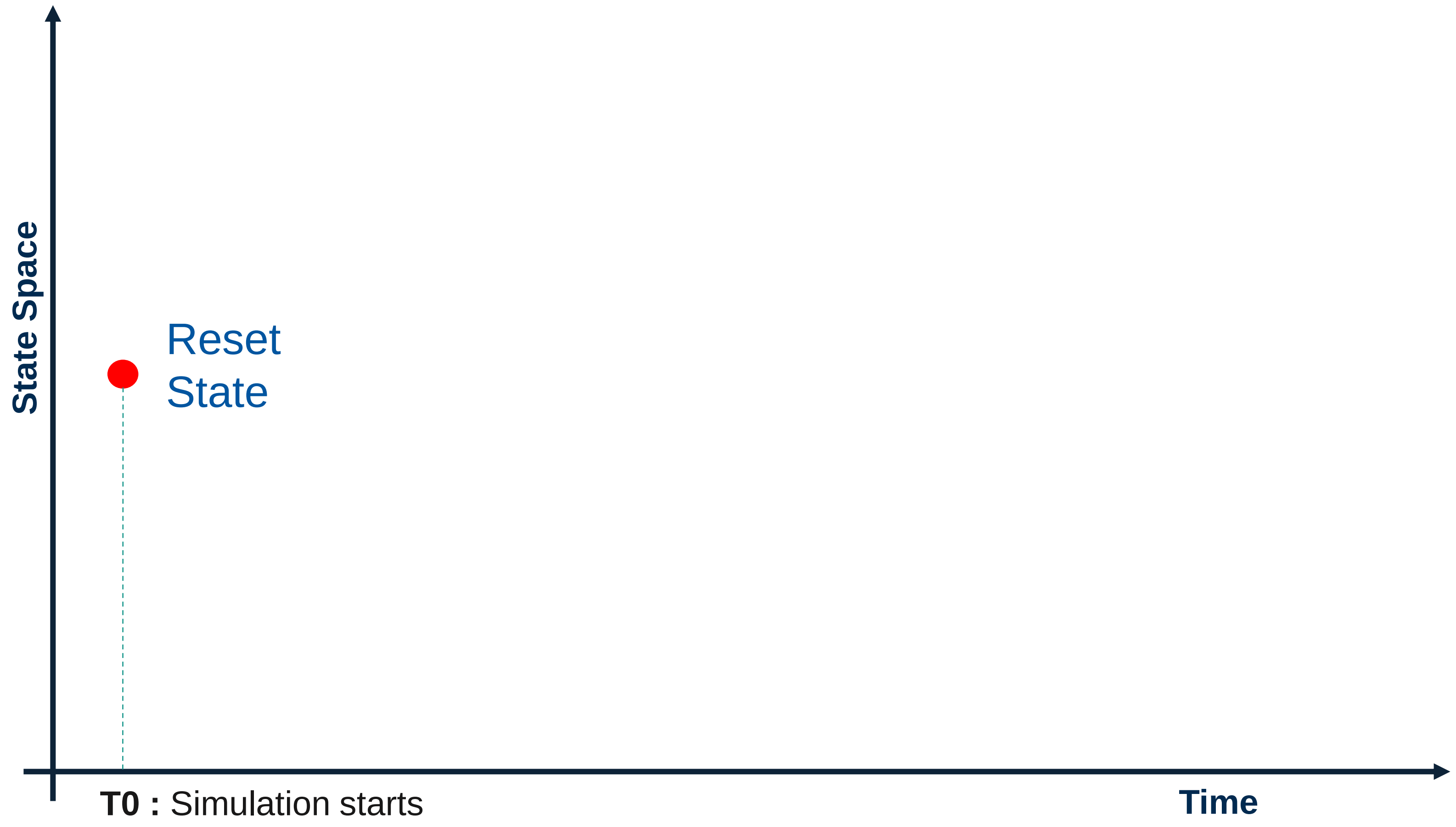
Simulation or formal ?



Floor planning of CROME FPGA



ACCURATE 2 ASIC

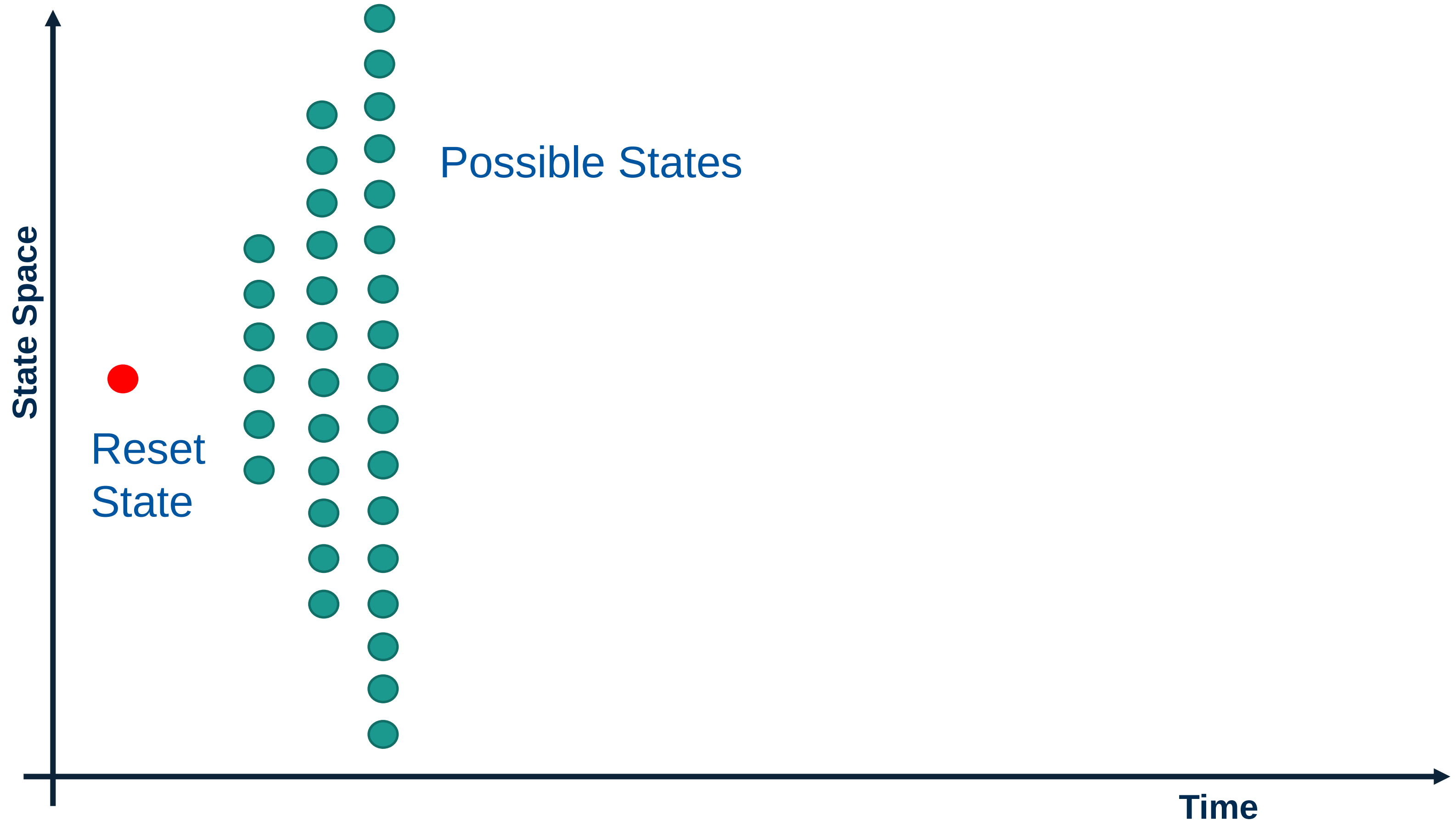


Possible States

Simulation based verification

As the simulation progress :

→ Every clock cycle the number of states explodes

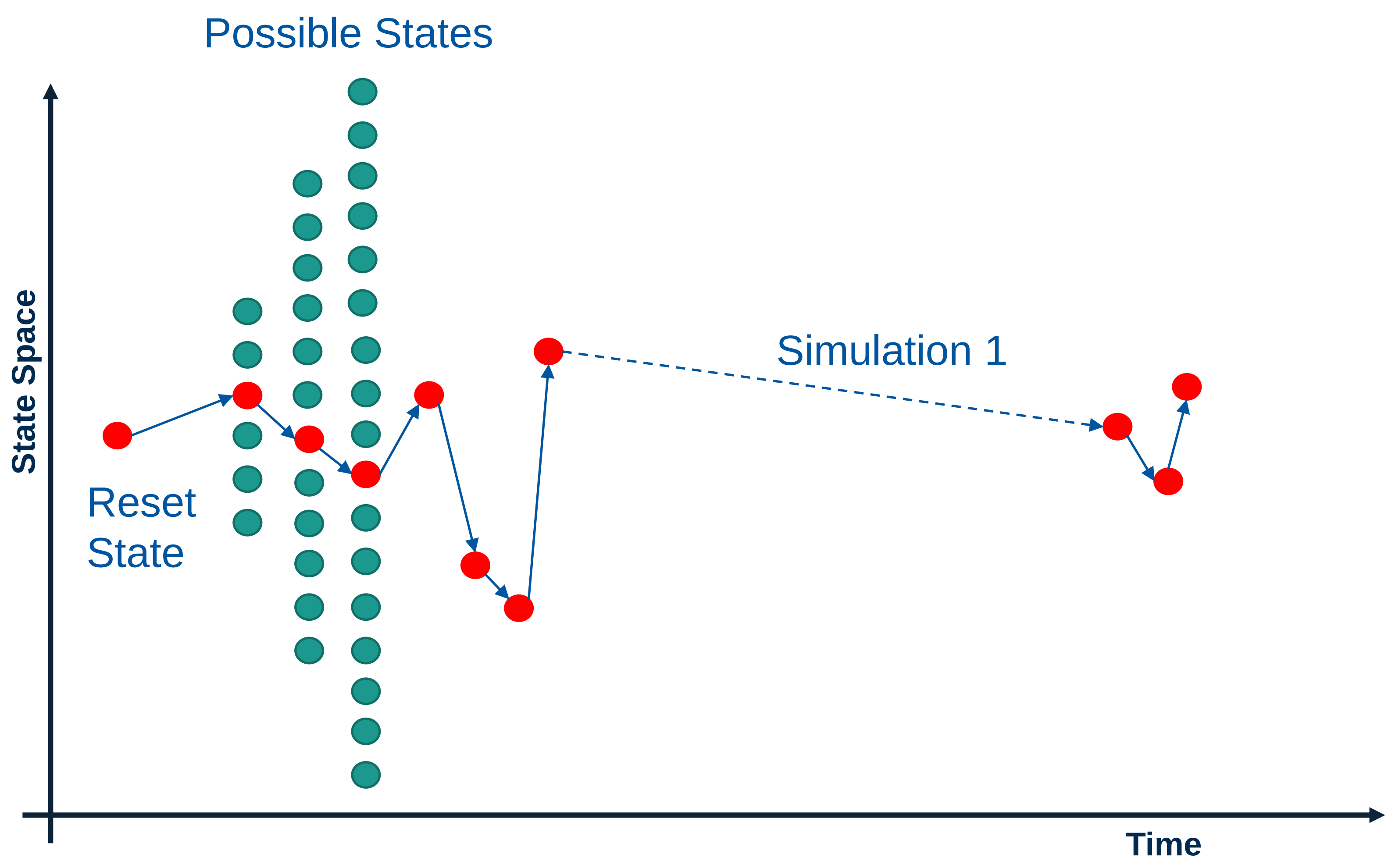


Simulation based verification

As the simulation progress :

→ Every clock cycle the number of states explodes

→ We progress through a specific path among the huge number of states in the state space



Simulation based verification

Golden Path 1

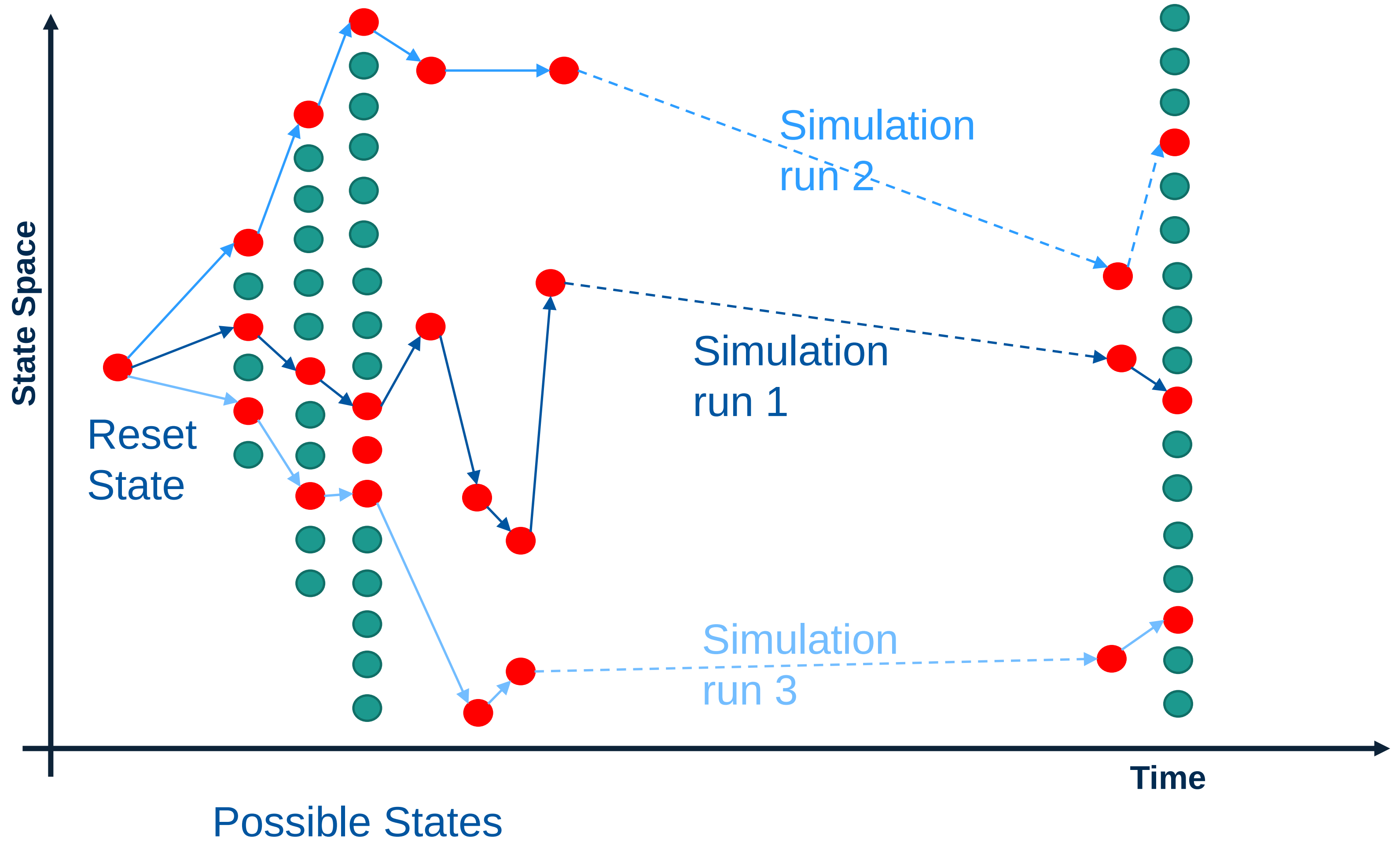
Golden Path 2

Golden Path 3

⋮

Golden Path n

where our design would work (with no assertion violation)

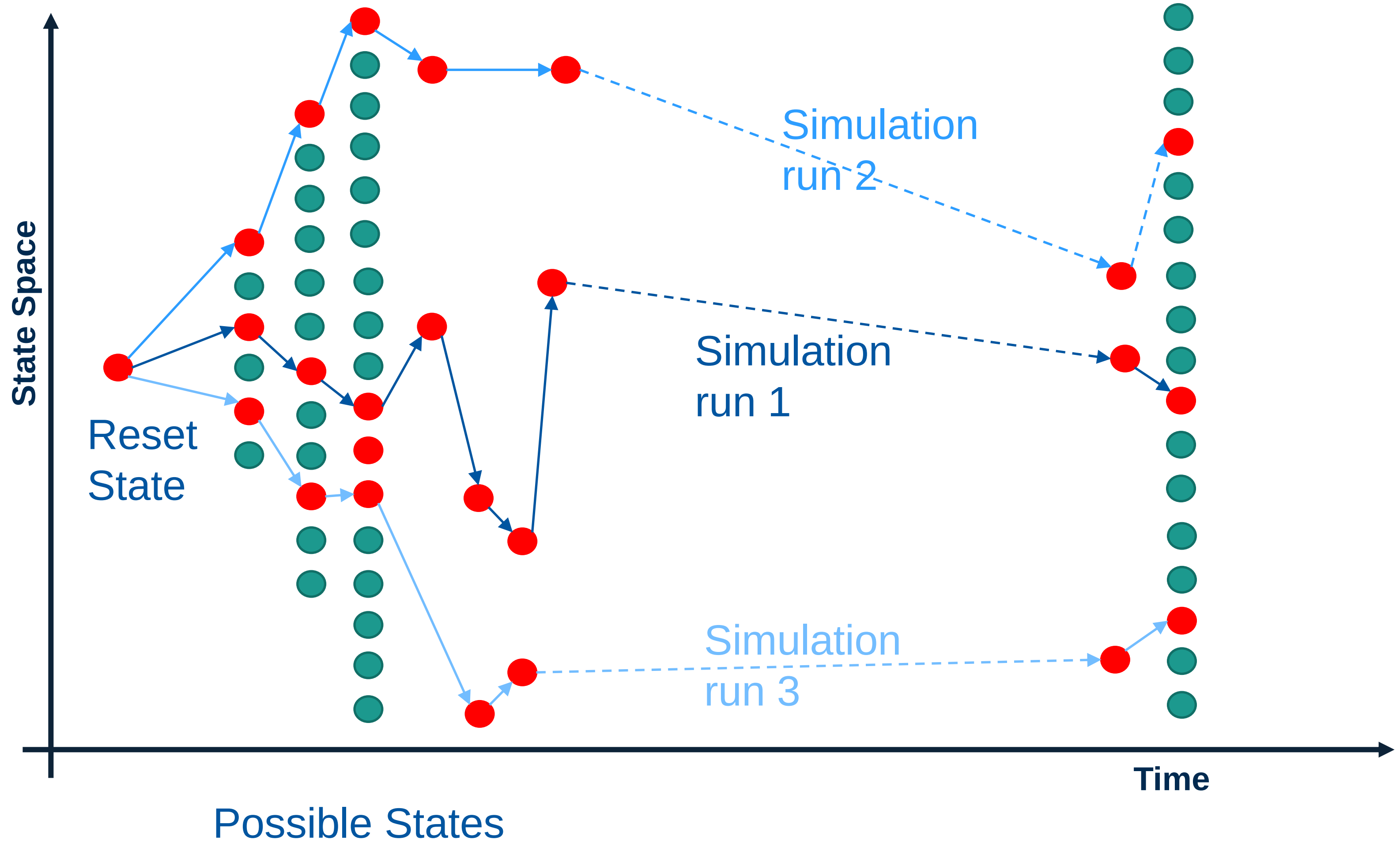


Simulation based verification

Simulation enumerate one state every cycle

☹ Design is only verified for the applied input stimulus

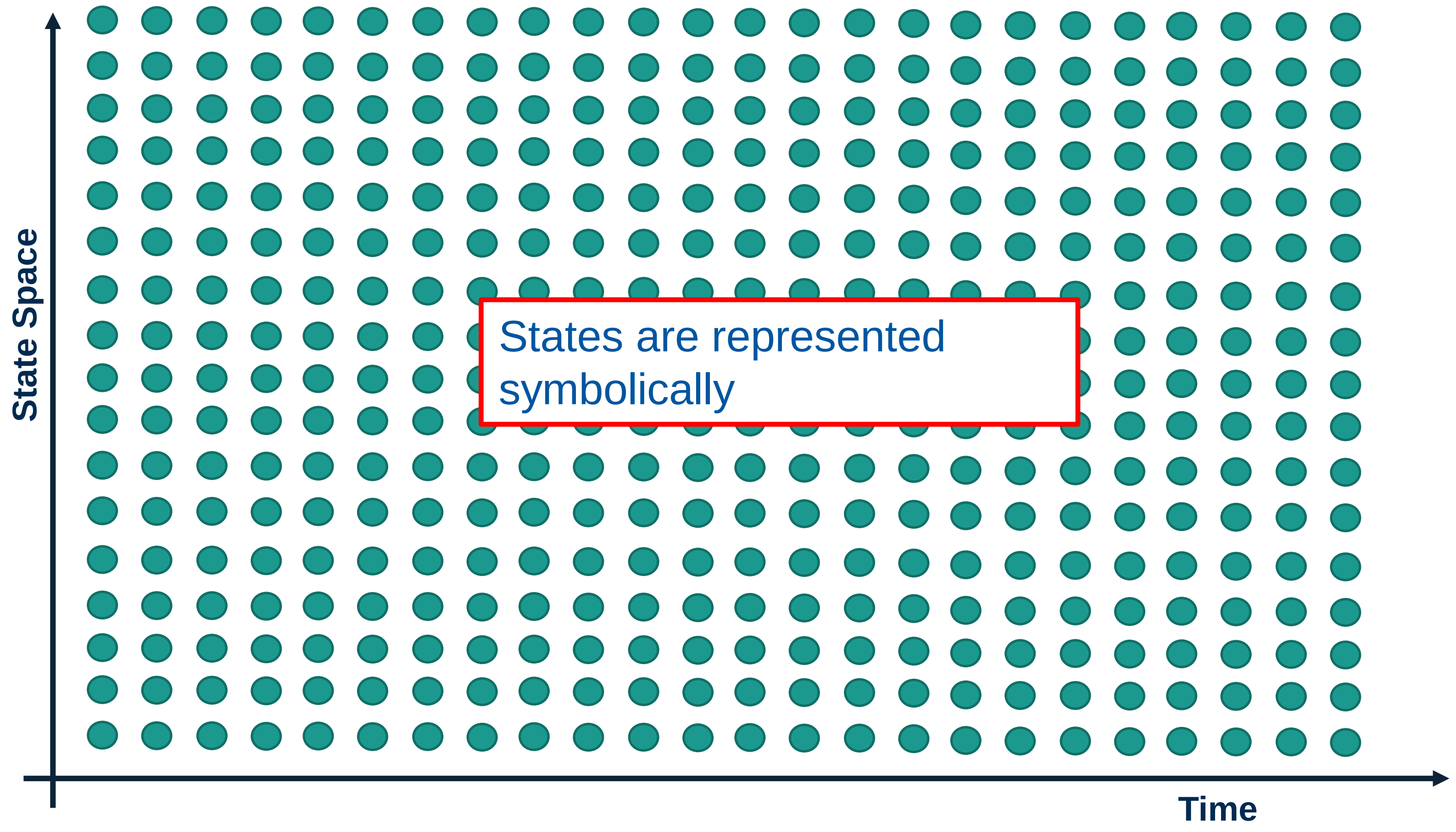
☹ Subject to time explosion



Formal Verification

The formal tool will not list all the states of our design

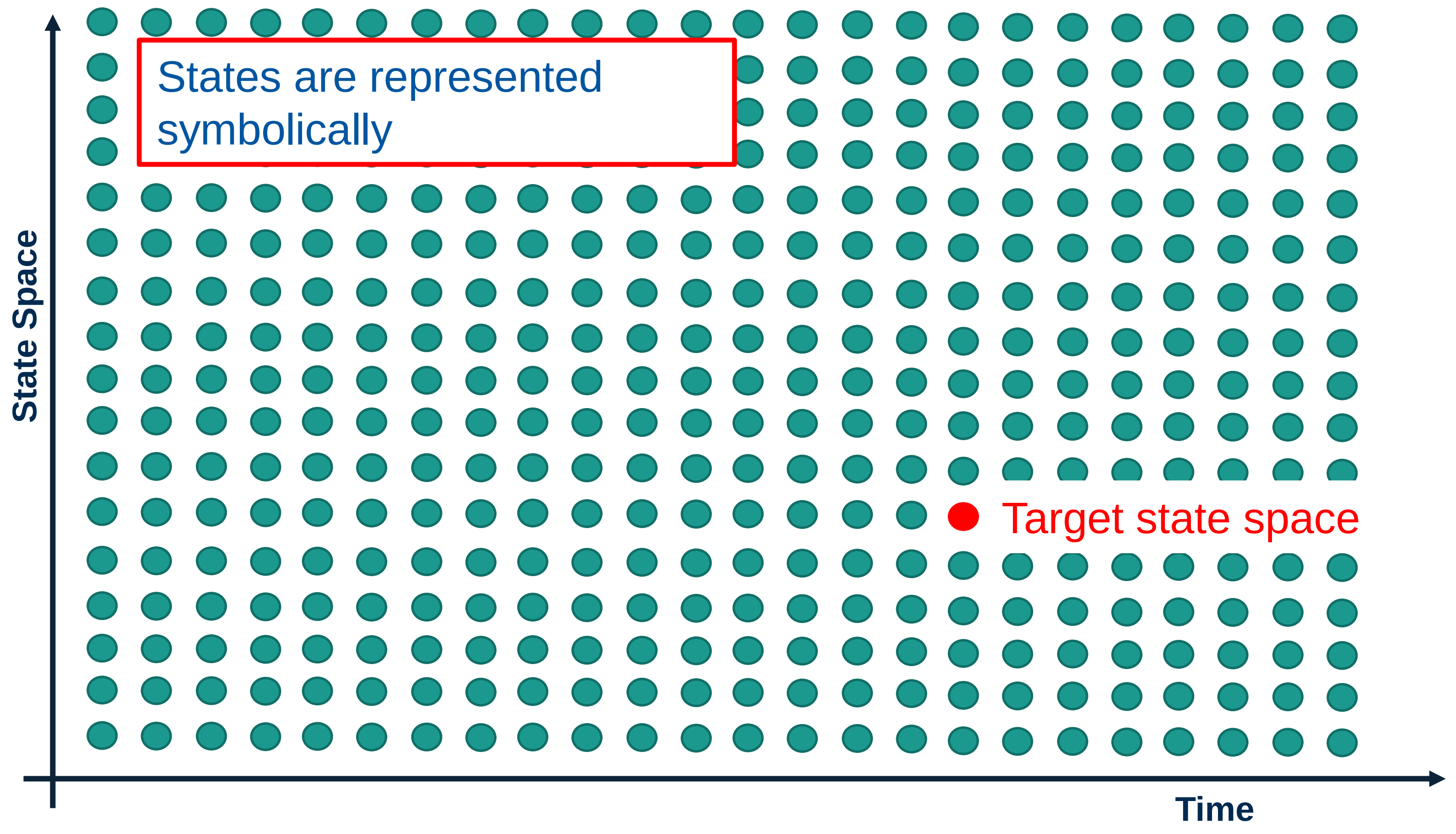
→ It will instead represent the state of our design with a mathematical formalism (e.g. ROBDD)



Formal Verification

We identify a target state space

→ The tool tries to demonstrate that the negation of the target state can be reached



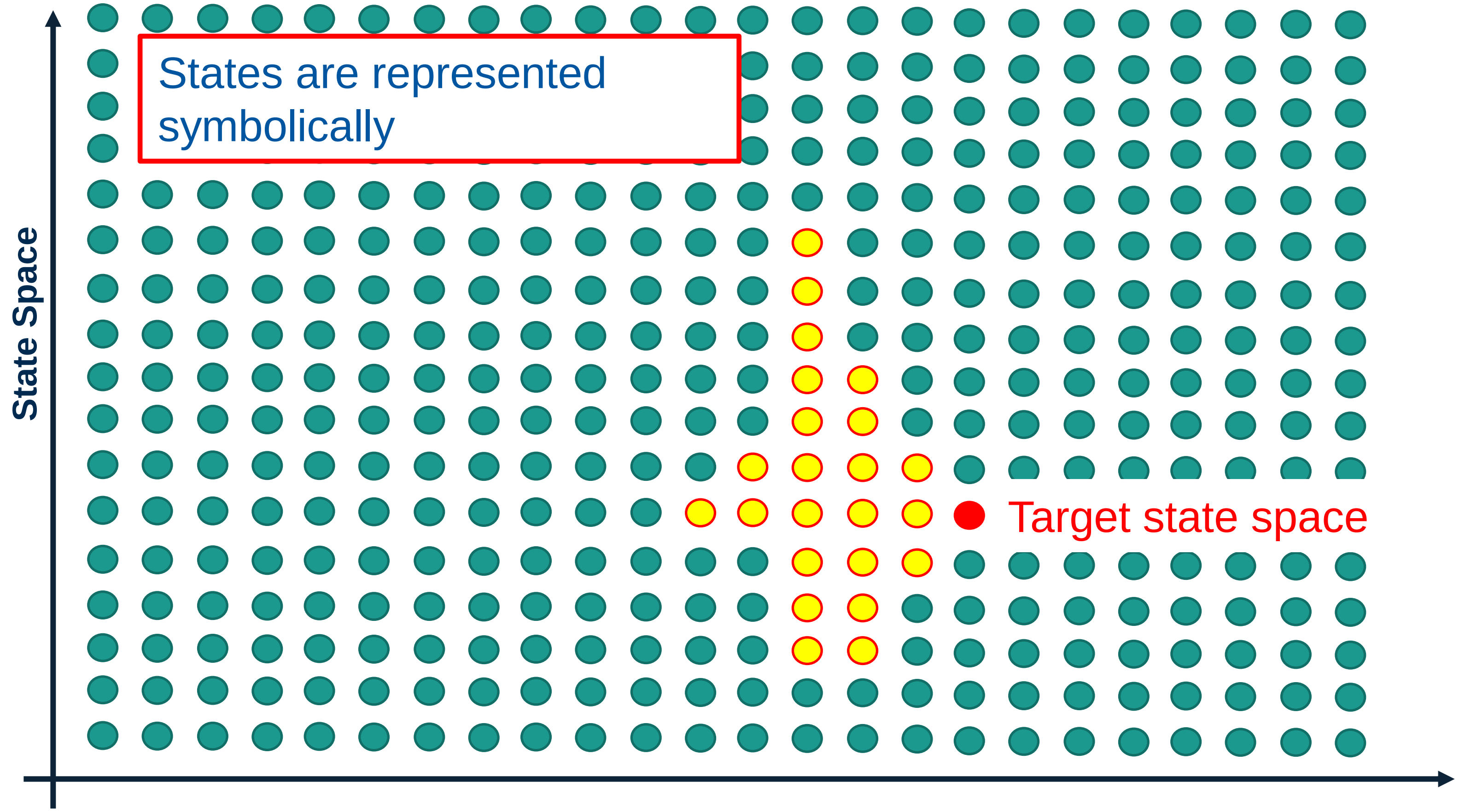
Formal Verification

We identify a target state space

→ The tool tries to demonstrate that the negation of the target state can be reached

→ Tool tries to find a sequence that will violate the assertion

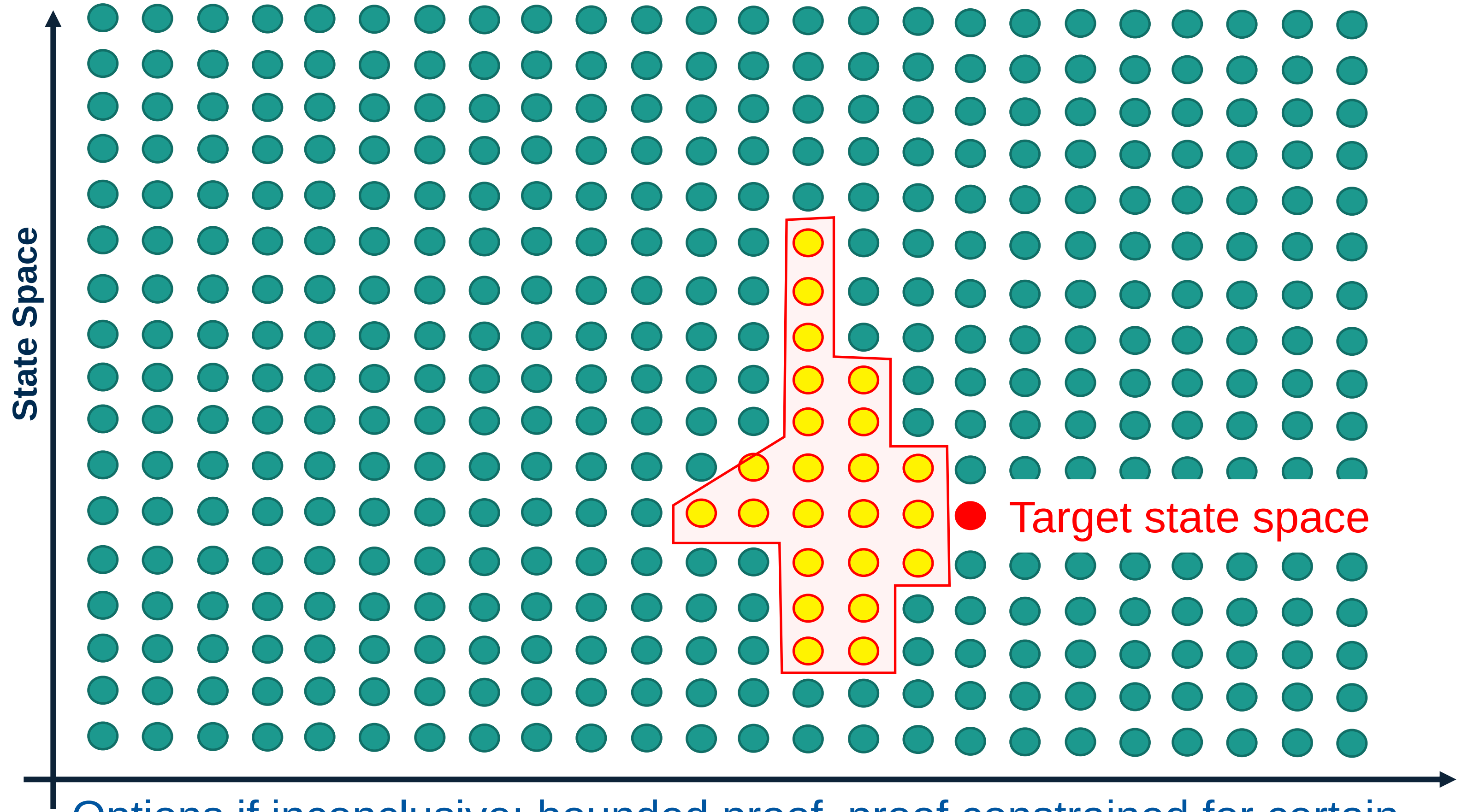
→ If no violation found over **full state space**, the assertion is proven



Formal Verification

Cone of Influence: The subset of the design states that can influence the target state

Assertion (target state/state sequence) proven for ALL input values and ALL points in time

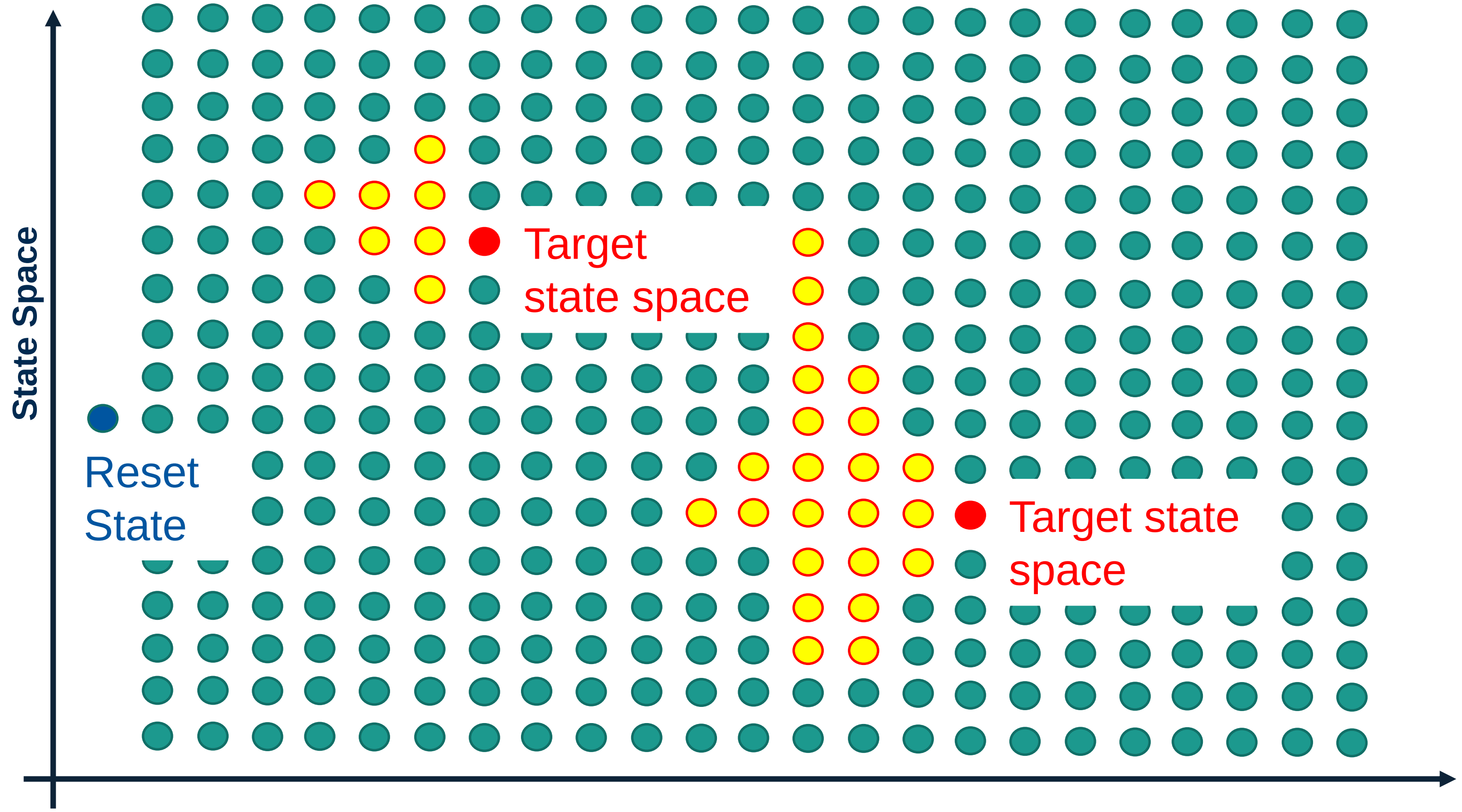


Options if inconclusive: bounded proof, proof constrained for certain input scenarios

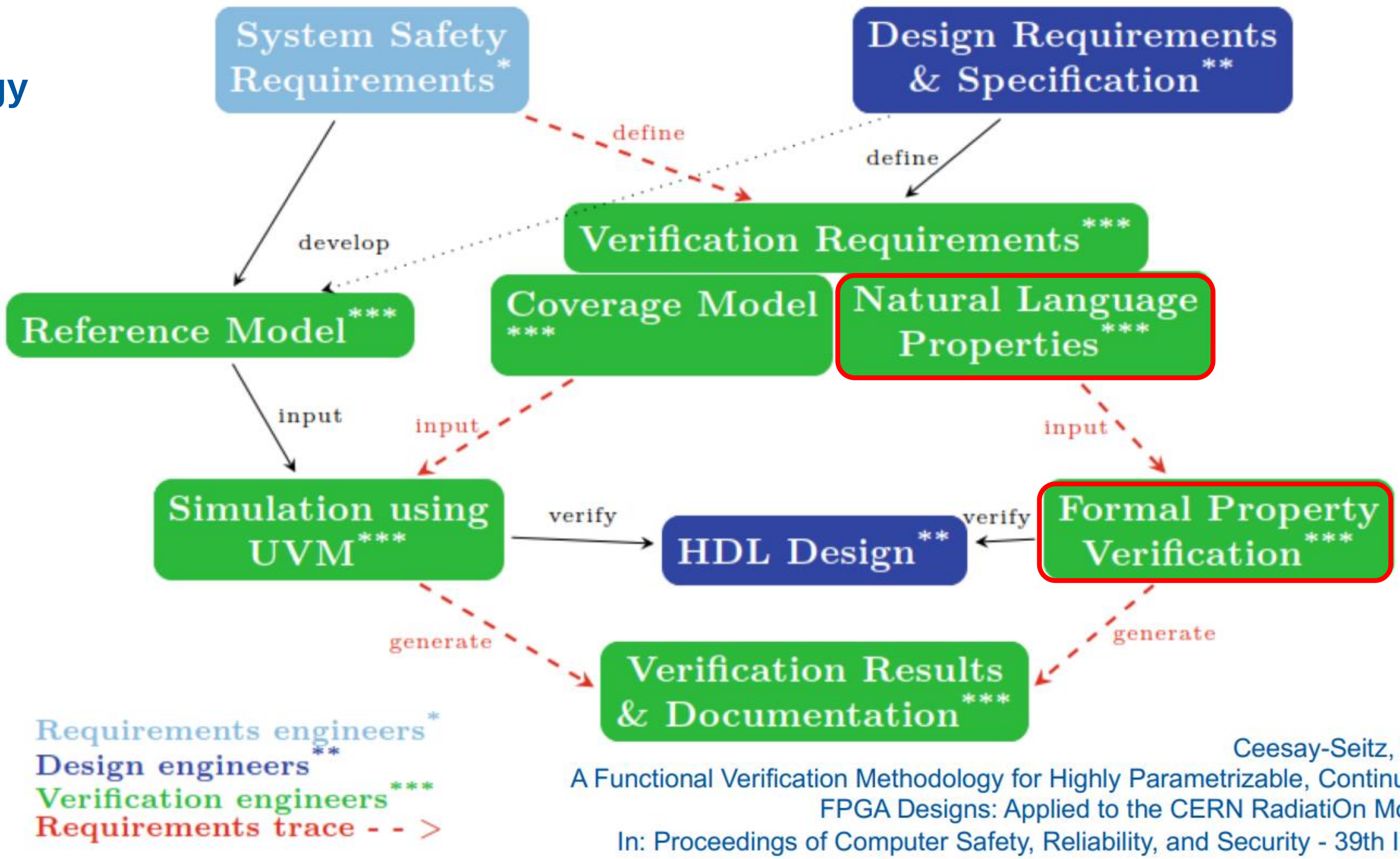
Formal Verification

States are represented symbolically

☹ Formal suffers from state space explosion



Our Verification Methodology


System Safety Requirements*
Design Requirements & Specification**
define
define
develop
Verification Requirements***
Reference Model***
Coverage Model***
Natural Language Properties***
input
input
input
Simulation using UVM***
verify
HDL Design**
verify
Formal Property Verification***
generate
generate
Verification Results & Documentation***
Requirements engineers*
Design engineers**
Verification engineers***
Requirements trace - - >

Ceesay-Seitz, K., Boukabache, H., Perrin, D.:
A Functional Verification Methodology for Highly Parametrizable, Continuously Operating Safety-Critical FPGA Designs: Applied to the CERN RadiatiOn Monitoring Electronics (CROME).
In: Proceedings of Computer Safety, Reliability, and Security - 39th International Conference (2020)

Natural Language Properties

- Requirement:

"It shall be possible to manually trigger a reset of a radiation dose alarm through the supervision software."

- Natural language property :

```
"(Cycle is no MC
 and (alarm was configured as latched at the previous MC)
 and alarm reset equals 1 and (dose value is less than (threshold at previous MC)
 or alarm function was deactivated at previous MC))

 implies that:
 (in one clock cycle, alarm is off)"
```

Ceesay-Seitz, K., Boukabache, H., Perrin, D.: Semi-formal reformulation of requirements for formal property verification. In: Proceedings of Design and Verification Conference and Exhibition Europe, DVCon Europe, Munich (2019)

Natural Language Properties

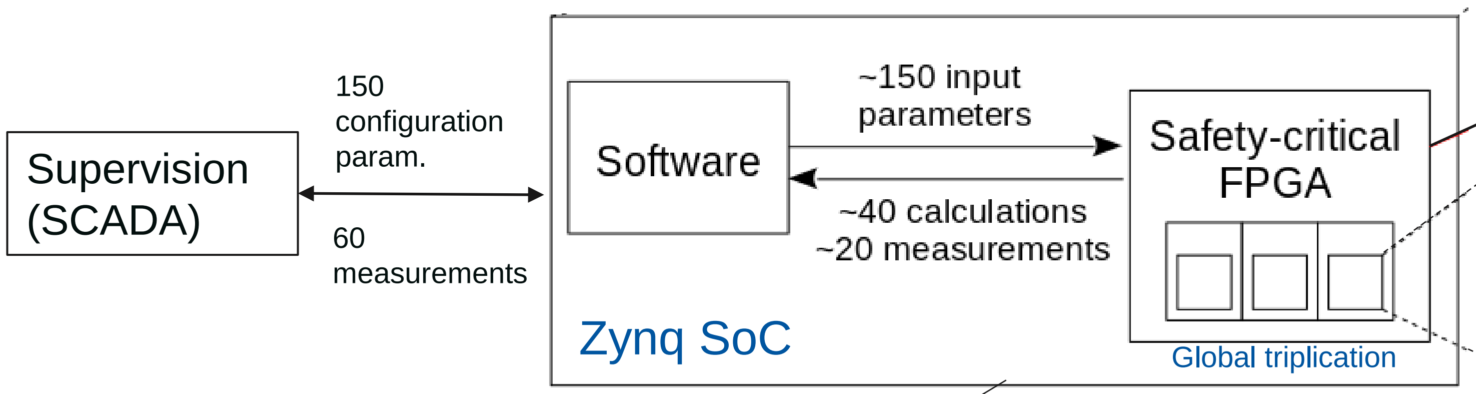
"(Cycle is no MC and (alarm was configured as latched at the previous MC) and alarm reset equals 1 and (dose value is less than (threshold at previous MC) or alarm function was deactivated at previous MC))
implies that:(in one clock cycle, alarm is off)“

- SystemVerilog property:

```
property pIntAlarmResetBetweenMT1();
     (mtValidxDI == 0 && latchedLastMC == 1 &&
integralAlarmResetxDI == 1 &&
     (signed’(integralxDO) < signed’(thresholdLastMc) ||
     alarmActiveLastMc == 0))
     |->
     ##1 (ALARMxDO == 0);
endproperty
```

Ceesay-Seitz, K., Boukabache, H., Perrin, D.:
Semi-formal reformulation of requirements for formal property verification.
In: Proceedings of Design and Verification Conference and Exhibition Europe, DVCon Europe, Munich (2019)

Verification Example – CERN RadiatiOn Monitoring Electronics (CROME)



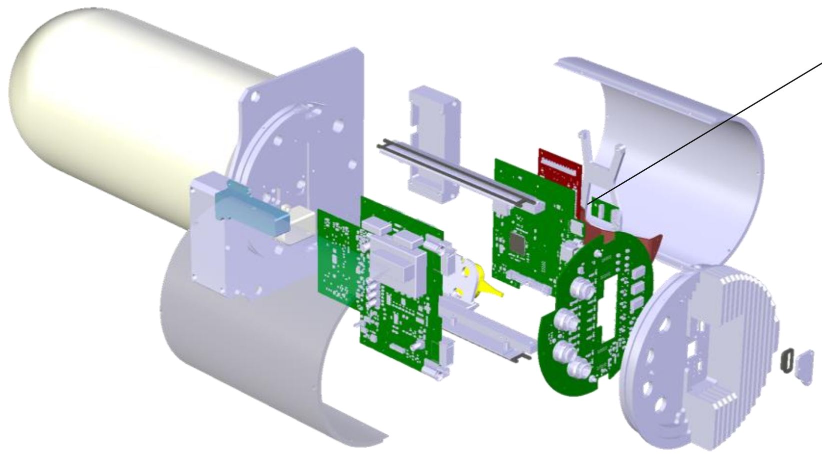
Alarm Units / Interlocking system

Alarm/Interlock Matrix:

Huge configurable logical formula
451 input bits (= 2^{451} configuration options)

Drives safety-critical outputs

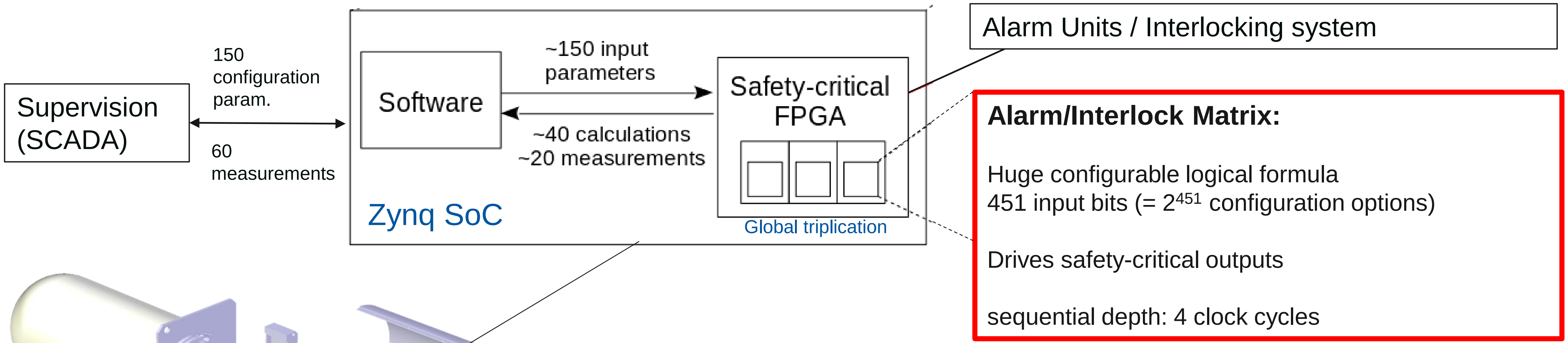
sequential depth: 4 clock cycles



Fault resilient FPGA design for 28 nm ZYNQ system-on-chip based radiation monitoring system at CERN
Microelectronics Reliability Journal
C Toner, H Boukabache, G Ducos, M Pangallo, S Danzeca, M Widorski, S Roesler, D Perrin

https://crome.web.cern.ch

Verification Example – CERN RadiatiOn Monitoring Electronics (CROME)

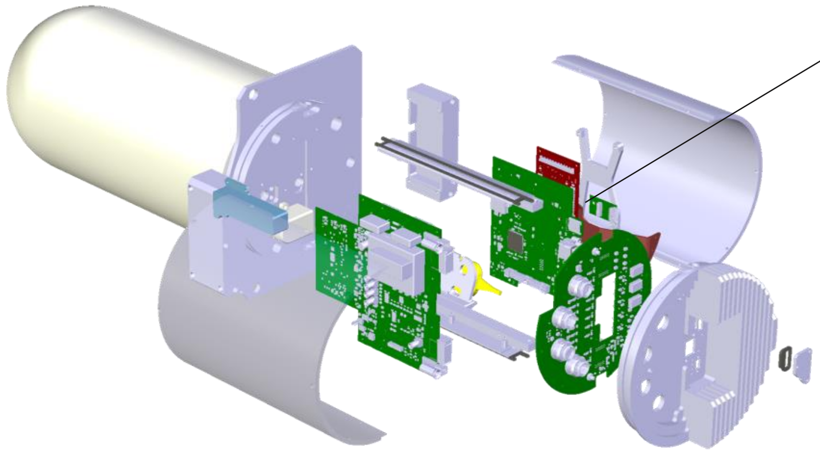


Alarm/Interlock Matrix:

Huge configurable logical formula
451 input bits (= 2^{451} configuration options)

Drives safety-critical outputs

sequential depth: 4 clock cycles

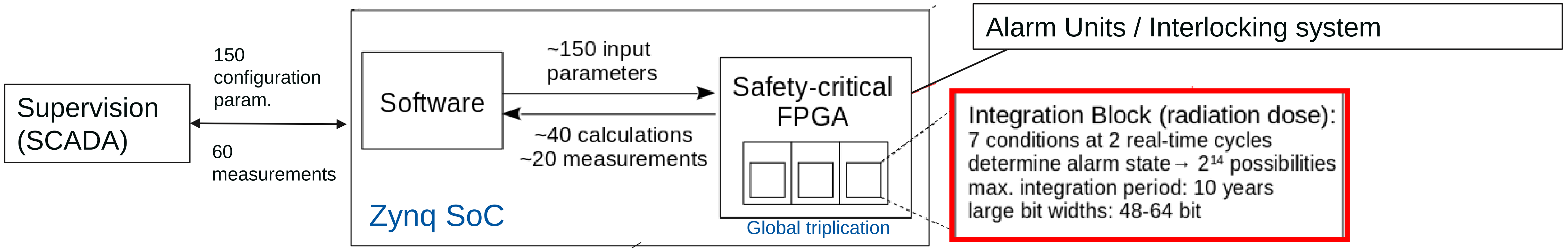


Estimated simulation time for all input combinations: $8*10^{137}$ years

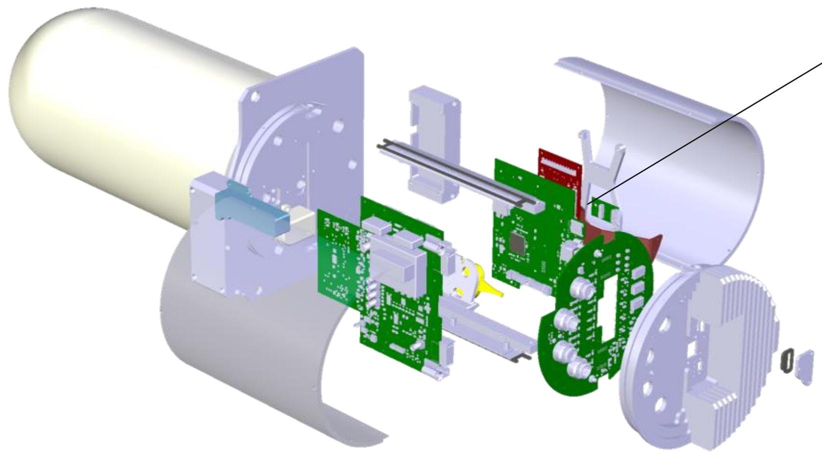
→ **46 properties proven in 33 seconds**

Fault example: In one particular configuration **radiation dose alert** was not triggered due to a wrong VHDL vector range

Verification Example – CERN RadiatiOn Monitoring Electronics (CROME)



Integration Block (radiation dose):
7 conditions at 2 real-time cycles
determine alarm state → 2^{14} possibilities
max. integration period: 10 years
large bit widths: 48-64 bit



Fault resilient FPGA design for 28 nm ZYNQ system-on-chip based radiation monitoring system at CERN
Microelectronics Reliability Journal
C Toner, H Boukabache, G Ducos, M Pangallo, S Danzeca, M Widorski, S Roesler, D Perrin

https://crome.web.cern.ch

Verification Example – CERN RadiatiOn Monitoring Electronics (CROME)

Exhaustively proved radiation dose **alarm generation**

Findings in integration/calculation algorithm :

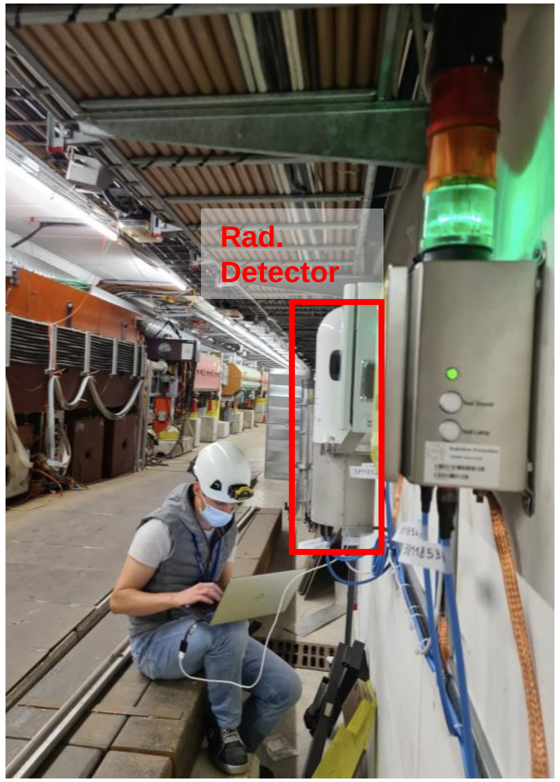
Undocumented design decision

→ **Fault** in rounding mechanism only if internal result was negative
→ **Scenario not covered by simulation** (400000 stimuli applied)

Fault that would happen **after 7 years of continuous operation**

→ Found after 1 second with formal
→ Would require > 7 years of simulation

CROME at transfer line (COMPASS)

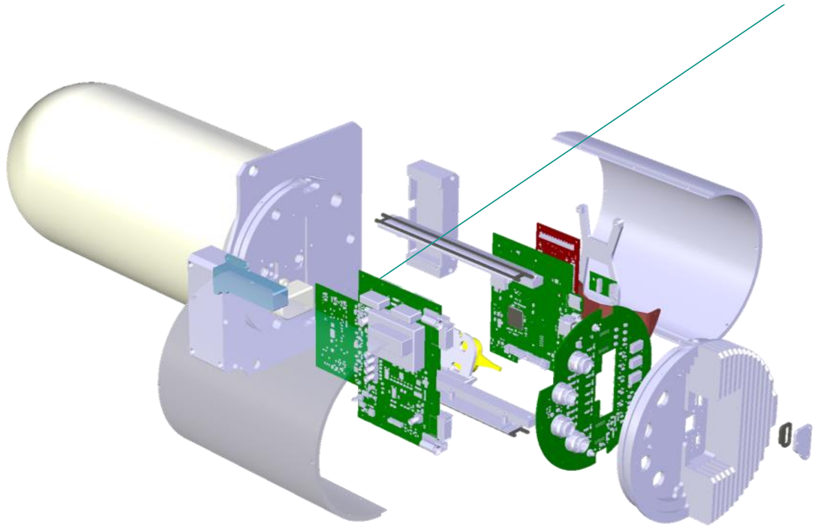


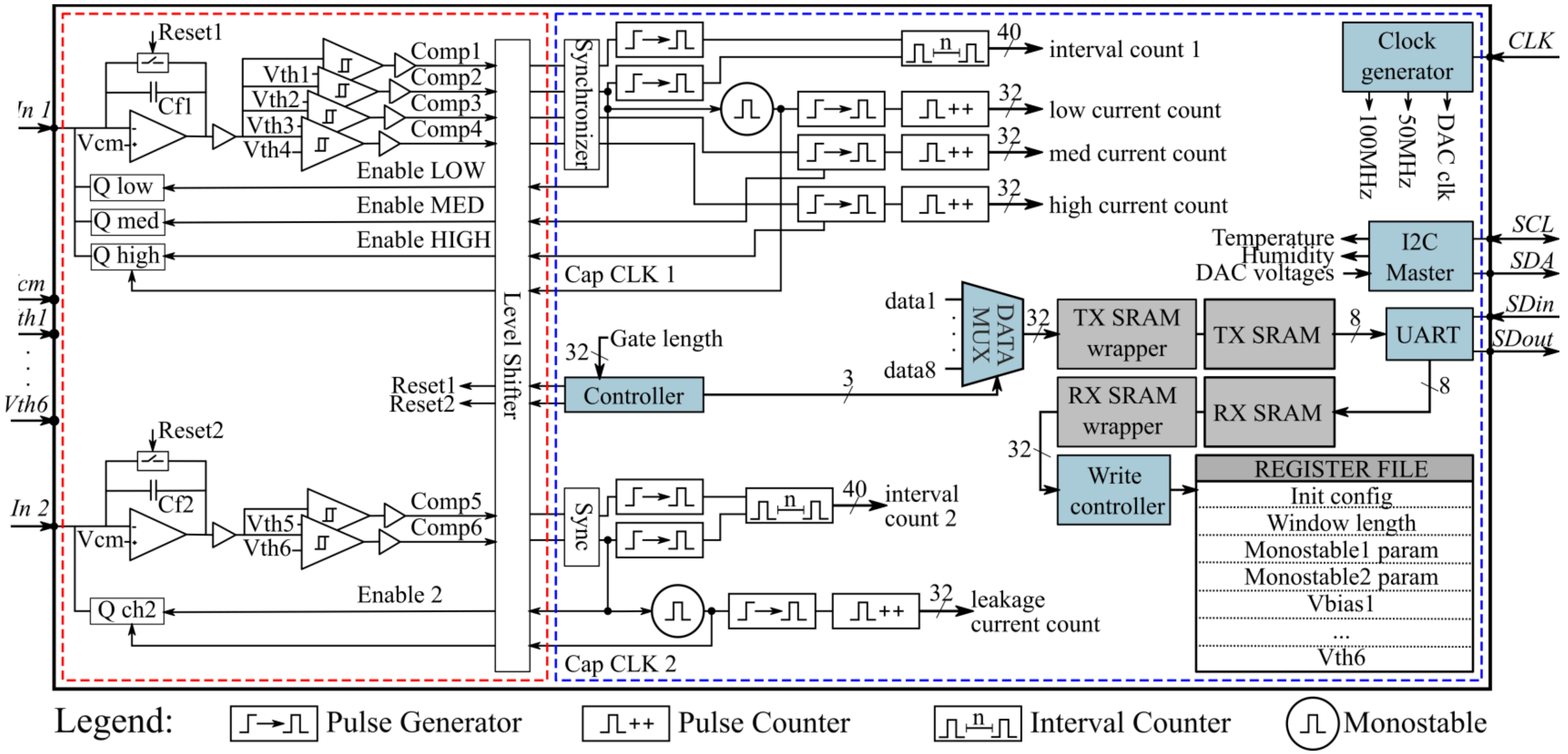
CROME at SM18

Verification Example – ACCURATE2 Mixed signal ASIC

Prototype for new read-out front end for CROME

- Several up to 40 bits wide counters
- Many corner cases





IP Core | Digital | Analog

S. K. Mohanan, H. Boukabache, V.Cruchet, D. Perrin, S.Roesler, and U. Pfeiffer, "*An Ultra Low Current Measurement Mixed-Signal ASIC for Radiation Monitoring Using Ionisation Chambers*", (IEEE sensors)

Verification Example – ACCURATE2 Mixed signal ASIC

```
assert property(
  // Normal counting
    counter == $past(counter) + 1 ||
  // Case 1: counting not started
    ( ($past(counter) == 0 ||
  // Case 2/3: counter should reset
     $past(counter) == $past(targetValue) ||
     otherCondition)
    && counter == 0 )
);
```

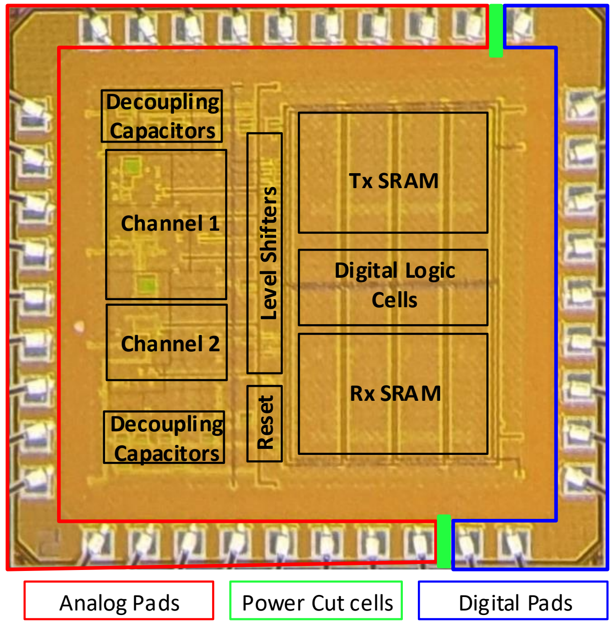
Exhaustively proved functionality of most blocks end-to-end

- Proved current measurement counters

Found and removed 33 faults, caused by:

- **Ambiguous specification: 20**
- Bug in Design Under Test: 10
- Contradicting verification requirements: 2
- Verification code: 1

Some proofs only concluded by manually finding invariants
End-to-end proofs of full design were not feasible



Ceesay-Seitz, K., Kundumattathil Mohanan, S. Boukabache, H., Perrin, D.:
Formal Property Verification of the Digital Section of an Ultra-Low Current Digitizer ASIC.
In: Proceedings of Design and Verification Conference and Exhibition Europe, DVCon Europe, Munich (2021)

Conclusion – Formal Methods

It is a **powerful tool** that can be applied

- During many stages of a development project (specification, model generation, verification),
- For many different systems (PLCs, FPGAs/ASICs, Software, …).

Challenges:

- State-space explosion: **not every design can be fully verified** within reasonable runtime
- Can be expensive in terms of engineering time for complex designs

Huge benefits for critical systems:

- **Unambiguous specifications** → less faults
- Automated tools → find bugs with little effort
- **Model checking covers a larger state space** than tests → find more faults
  - Proofs are valid for all input combinations over all time (within the chosen constraints)
- **Fast detection of corner case faults** → hard to find with simulation or tests

Contact – Formal Methods interest group

If you are interested in Formal Methods, please subscribe to the following egroup:

formal-methods-interest-group@cern.ch

(you will get updates about future events and presentations)

If you want to get started with formal methods or stablish a collaboration, you can send an email to the following egroup:

formal-methods-working-group-admin@cern.ch

We have created a *readthedocs page,* where useful information about formal methods and verification will be collected:

https://readthedocs.web.cern.ch/display/FMVWG/Formal+methods+interest+group+Home www.cern.ch