HSE
Occupational Health & Safety
and Environmental Protection unit

Simulation & Formal Property Verification
Application to complex highly-parametrizable, continuously operating PLDs

Hamza Boukabache, **Katharina Ceesay-Seitz, Jonas Bodingbauer**

7th October 2021

Why do we need functional verification

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

Goal :

Find **systematic** failures

Methods :

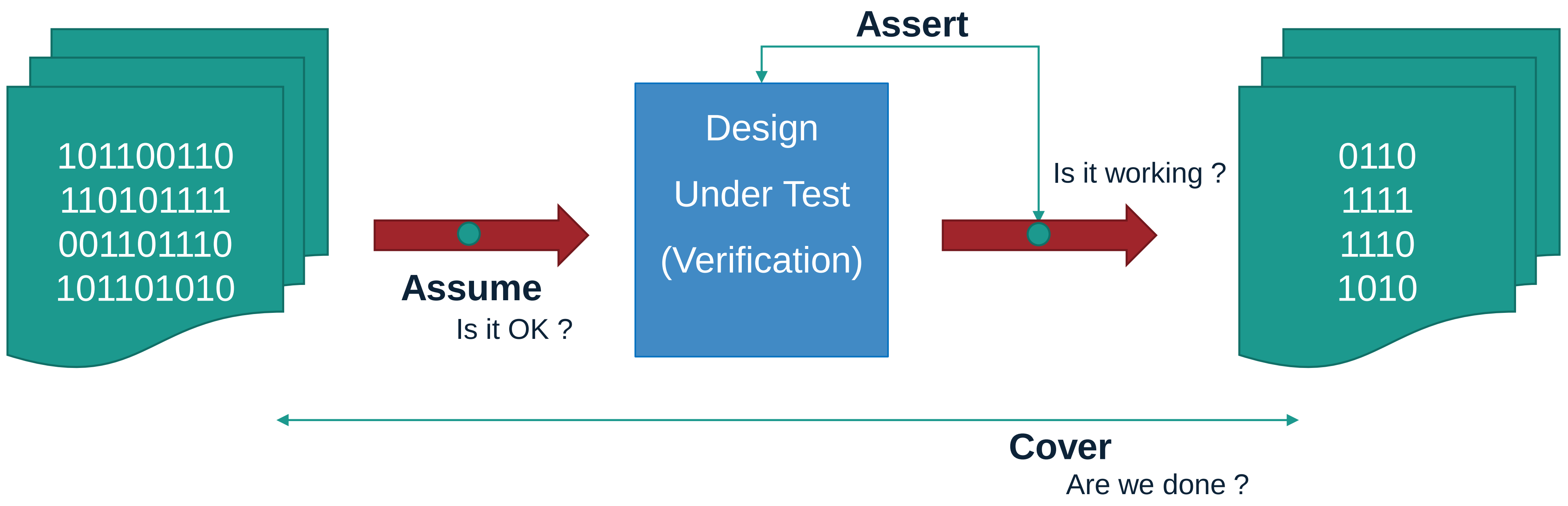
Simulation, Formal, Emulation and Prototyping

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

What is Formal Verification ? How does it work ?



Is Formal Verification easy to use ?

RTL Code → Write Properties → Lunch the tool → Results for every property

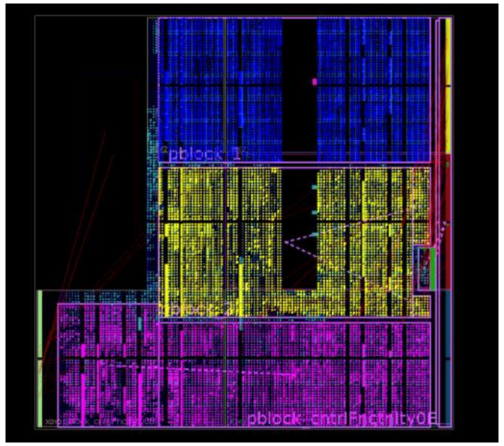
Lunch the tool
- script

Results for every property
- Proof for the assertion
- Counter-Example
- Inconclusive proof

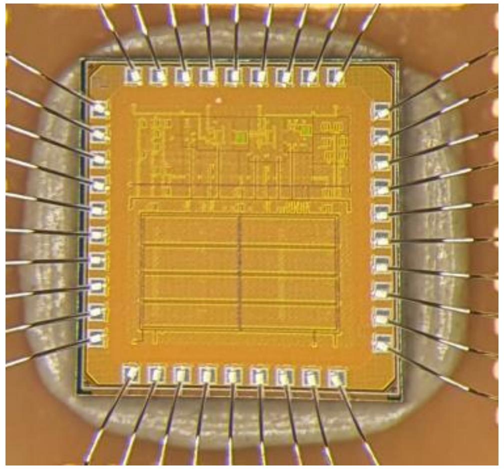
```
pMatOut1: assert property (@(posedge clk)$rose(startMatrixEvaluationxDI)|-> (##`nrCMatrixEval (Out1 == outputsxDO[0])));
pMatOut2: assert property (@(posedge clk)$rose(startMatrixEvaluationxDI)|-> (##`nrCMatrixEval (Out2 == outputsxDO[1])));
pMatOut3: assert property (@(posedge clk)$rose(startMatrixEvaluationxDI)|-> (##`nrCMatrixEval (Out3 == outputsxDO[2])));
pMatOut4: assert property (@(posedge clk)$rose(startMatrixEvaluationxDI)|-> (##`nrCMatrixEval (Out4 == outputsxDO[3])));
pMatAu1: assert property (@(posedge clk)$rose(startMatrixEvaluationxDI) |-> (##`nrCMatrixEval AU1 == AUxDO[0]));
pMatAu2: assert property (@(posedge clk)$rose(startMatrixEvaluationxDI) |-> (##`nrCMatrixEval AU2 == AUxDO[1]));
pMatAu3: assert property (@(posedge clk)$rose(startMatrixEvaluationxDI) |-> (##`nrCMatrixEval AU3 == AUxDO[2]));
```

```
--------------------------------------
Property Summary                 Count
--------------------------------------
Assumed                             18
Proven                              46
Covered                              8
Inconclusive                         0
Fired                                0
Uncoverable                          0
--------------------------------------
Total                               72
--------------------------------------
```

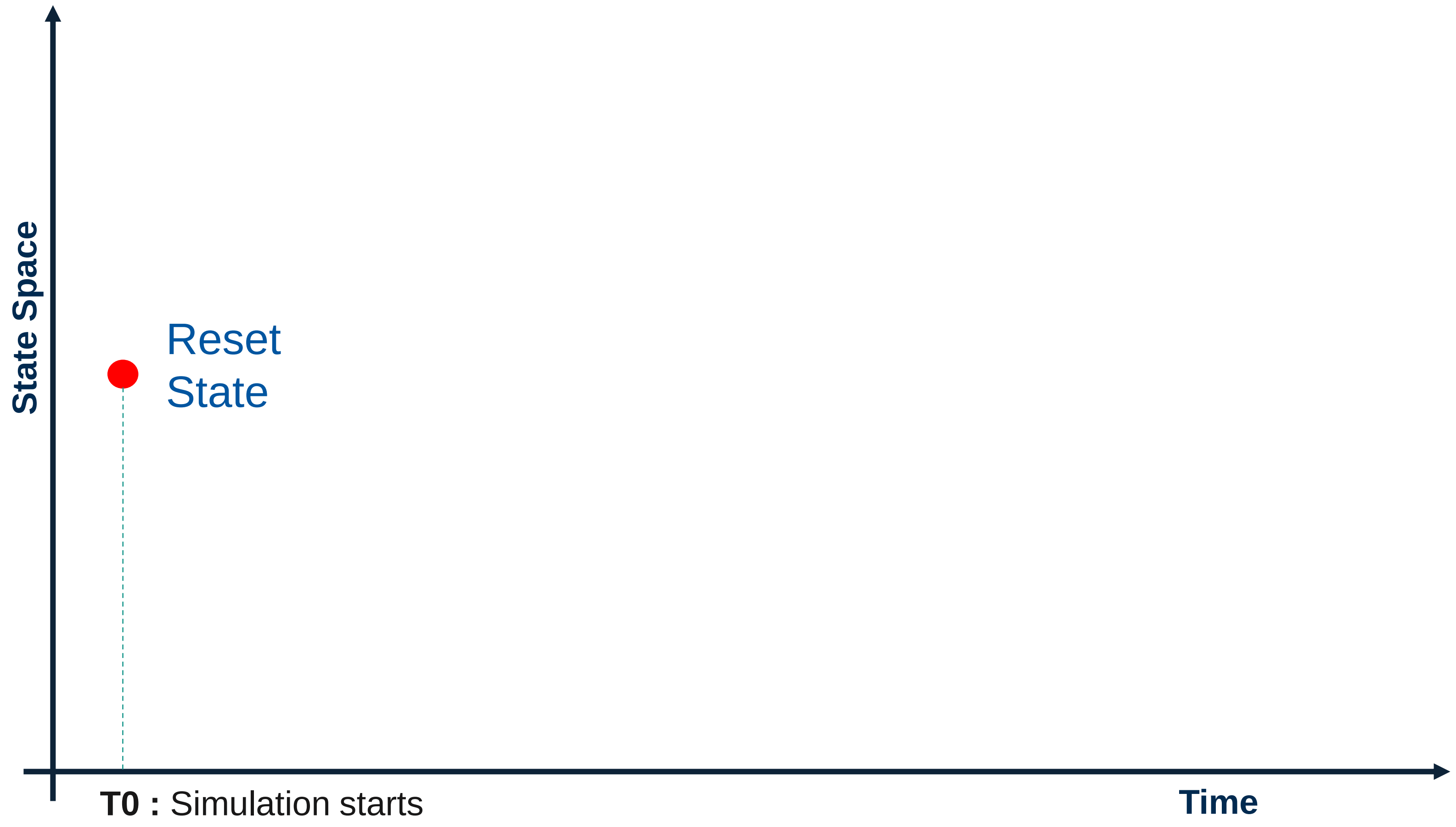
Simulation or formal ?



Floor planning of CROME FPGA



ACCURATE 2 ASIC

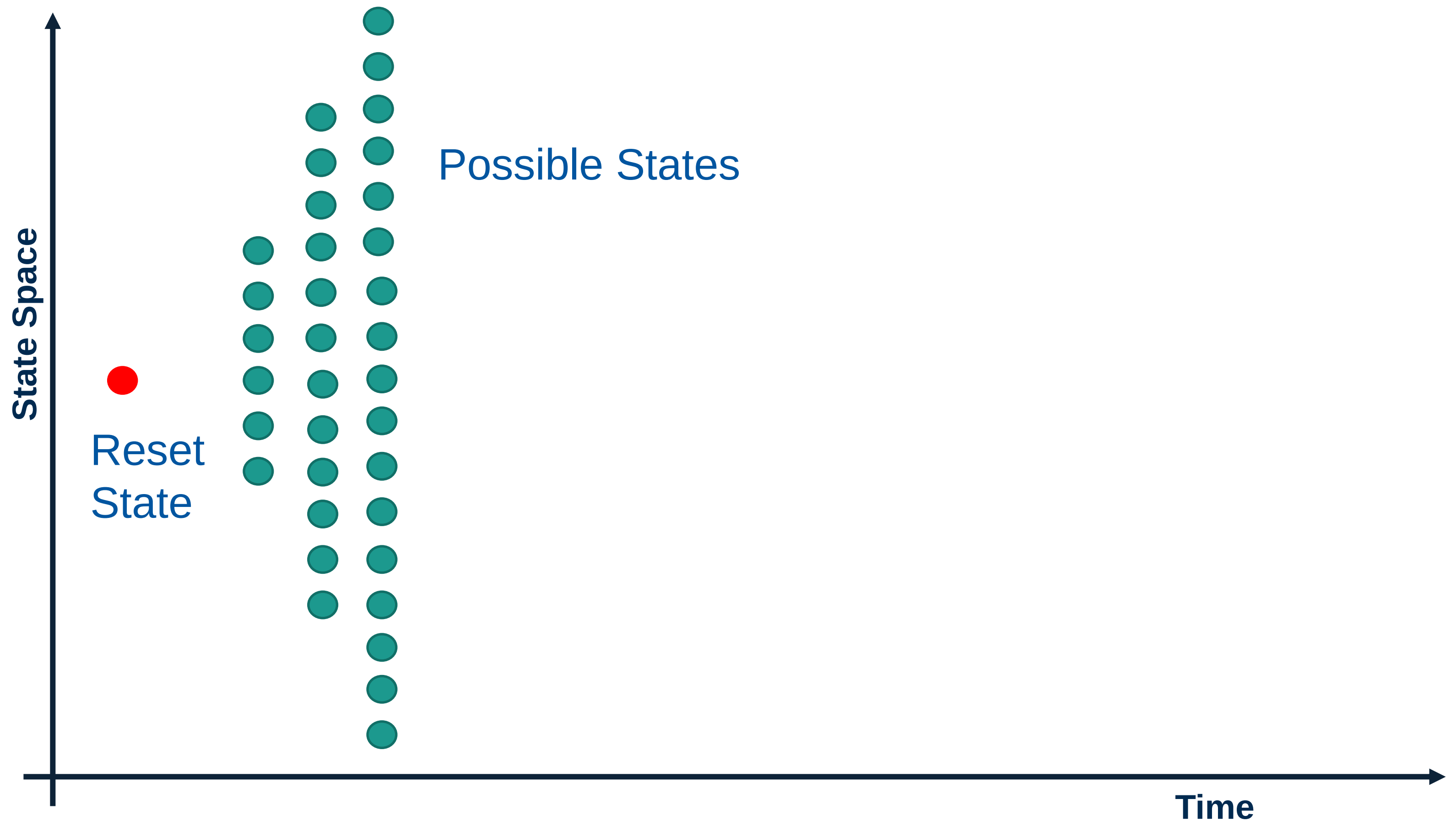


Possible States

Simulation based verification

As the simulation progress :

→ Every clock cycle the number of states explodes

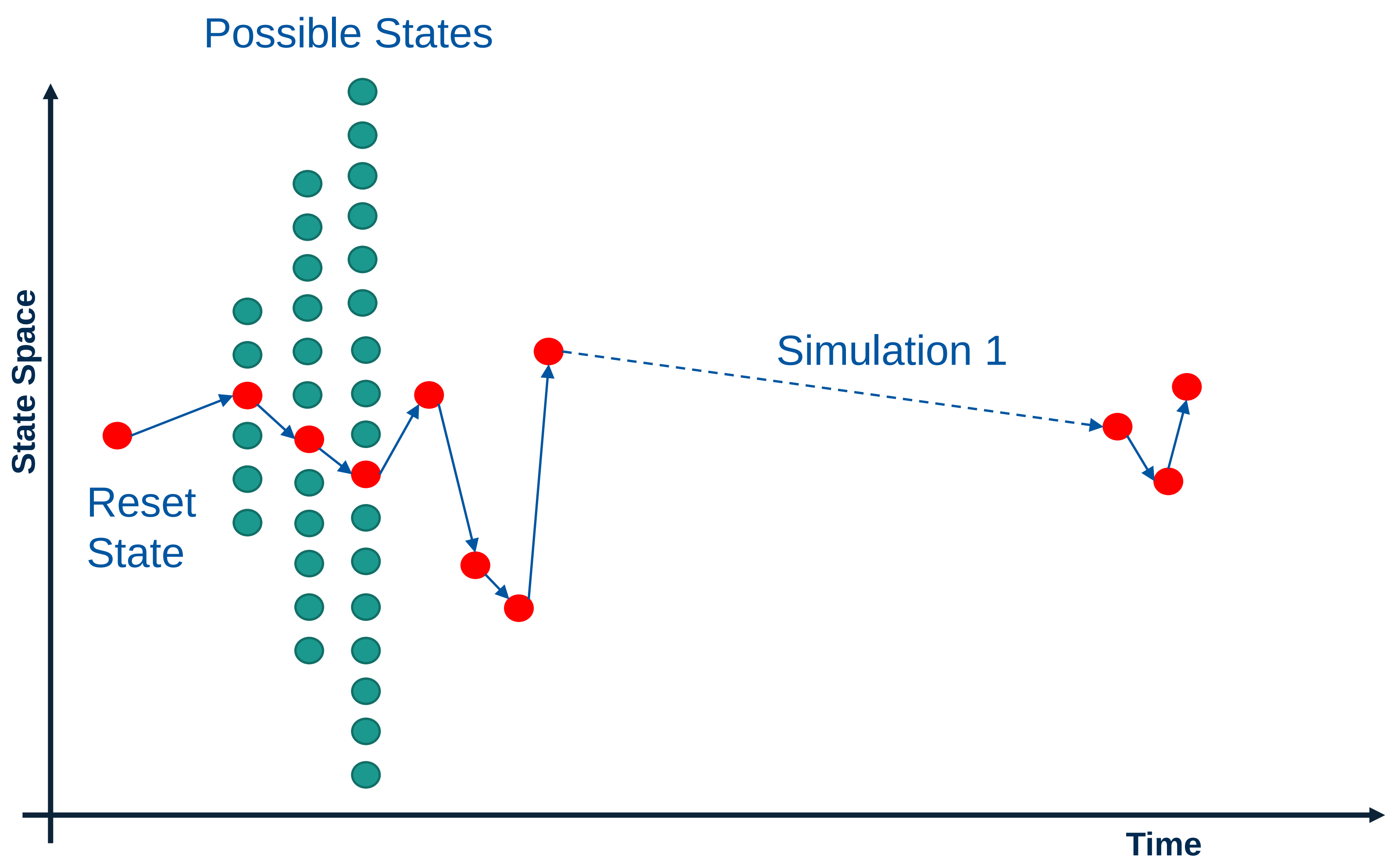


Simulation based verification

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

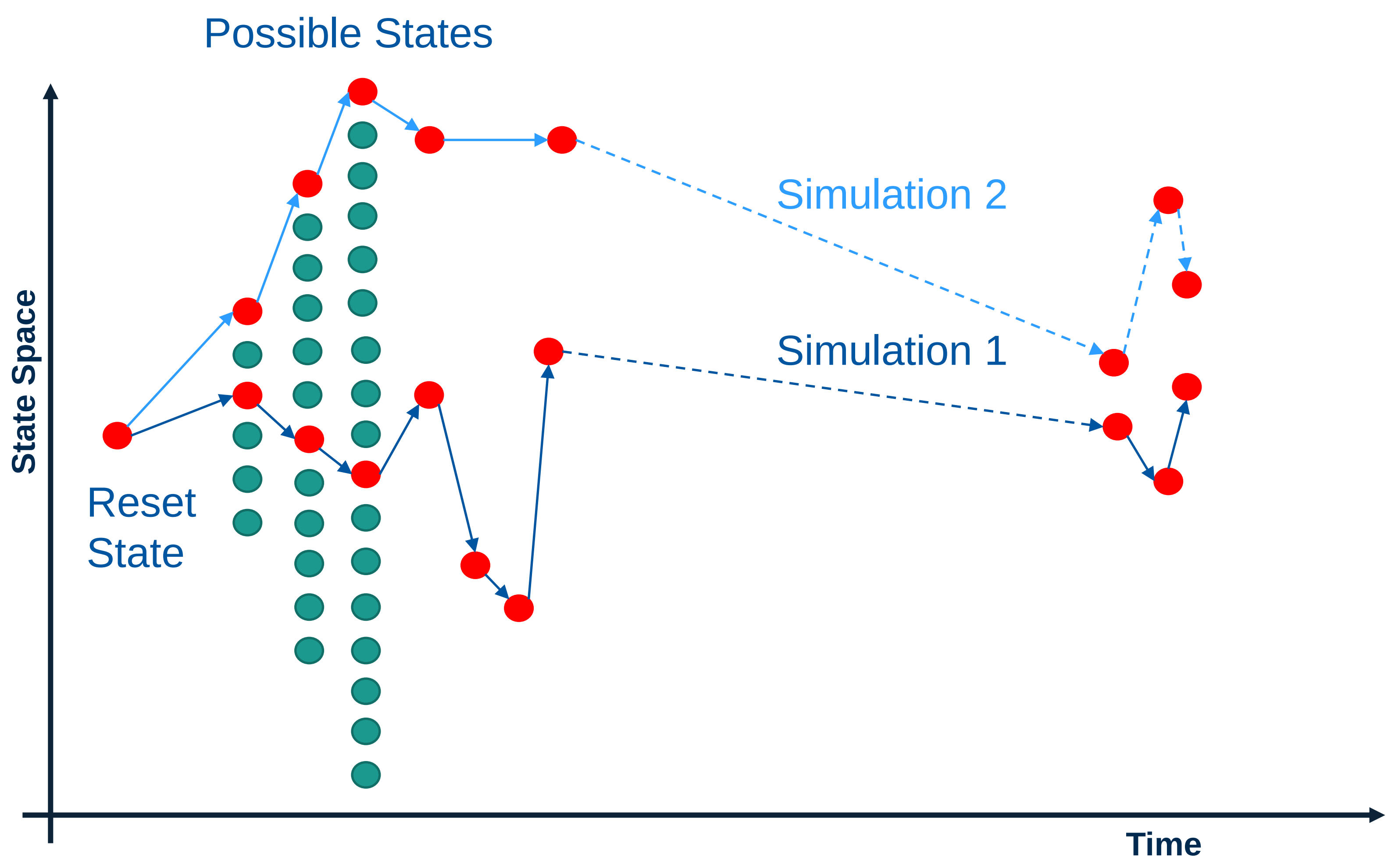


Simulation based verification

Golden Path 2

Golden Path 1

where our design would work
(with no assertion violation)

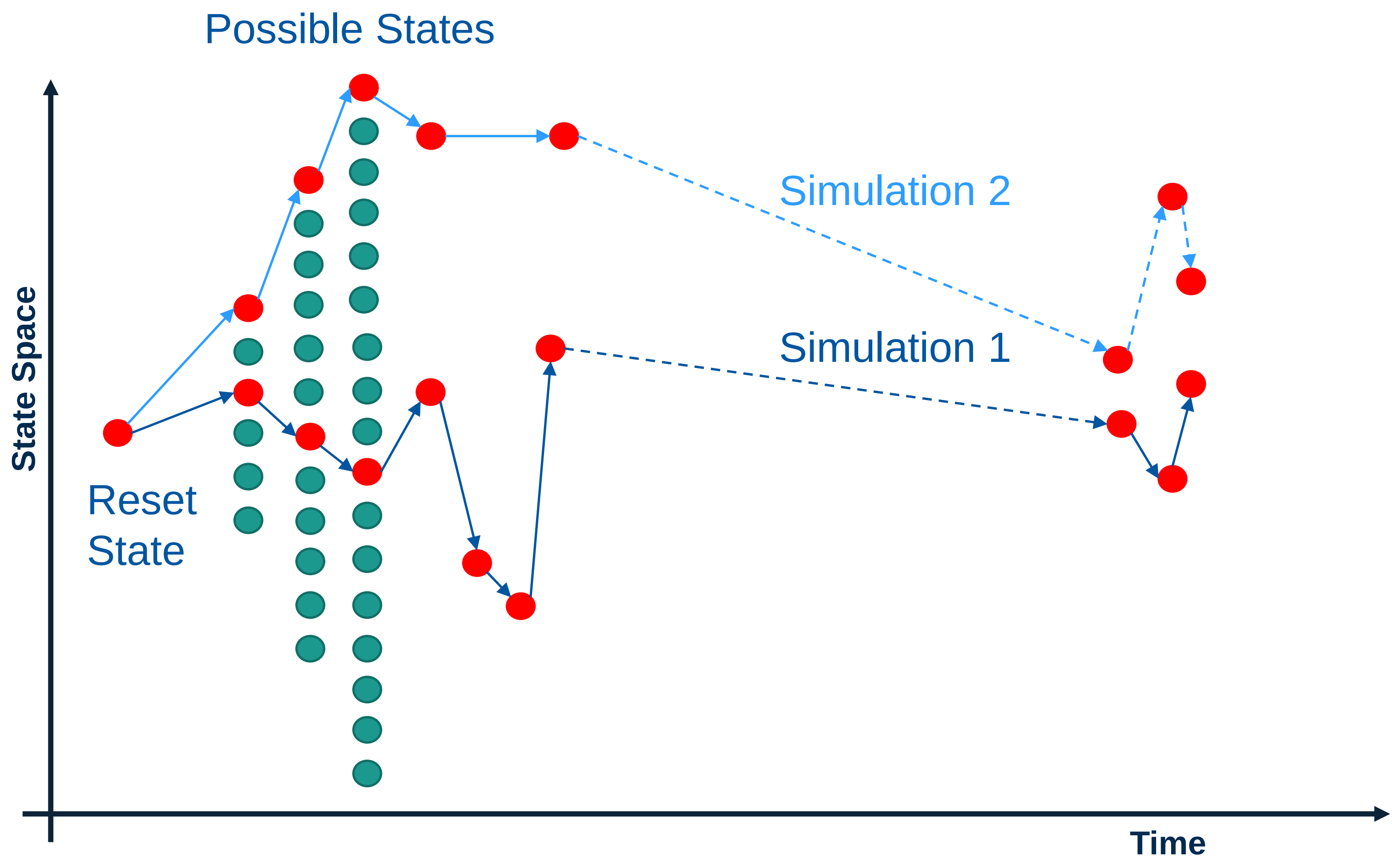


Simulation based verification

Simulation enumerate one state every cycle

☹ Requires input stimulus

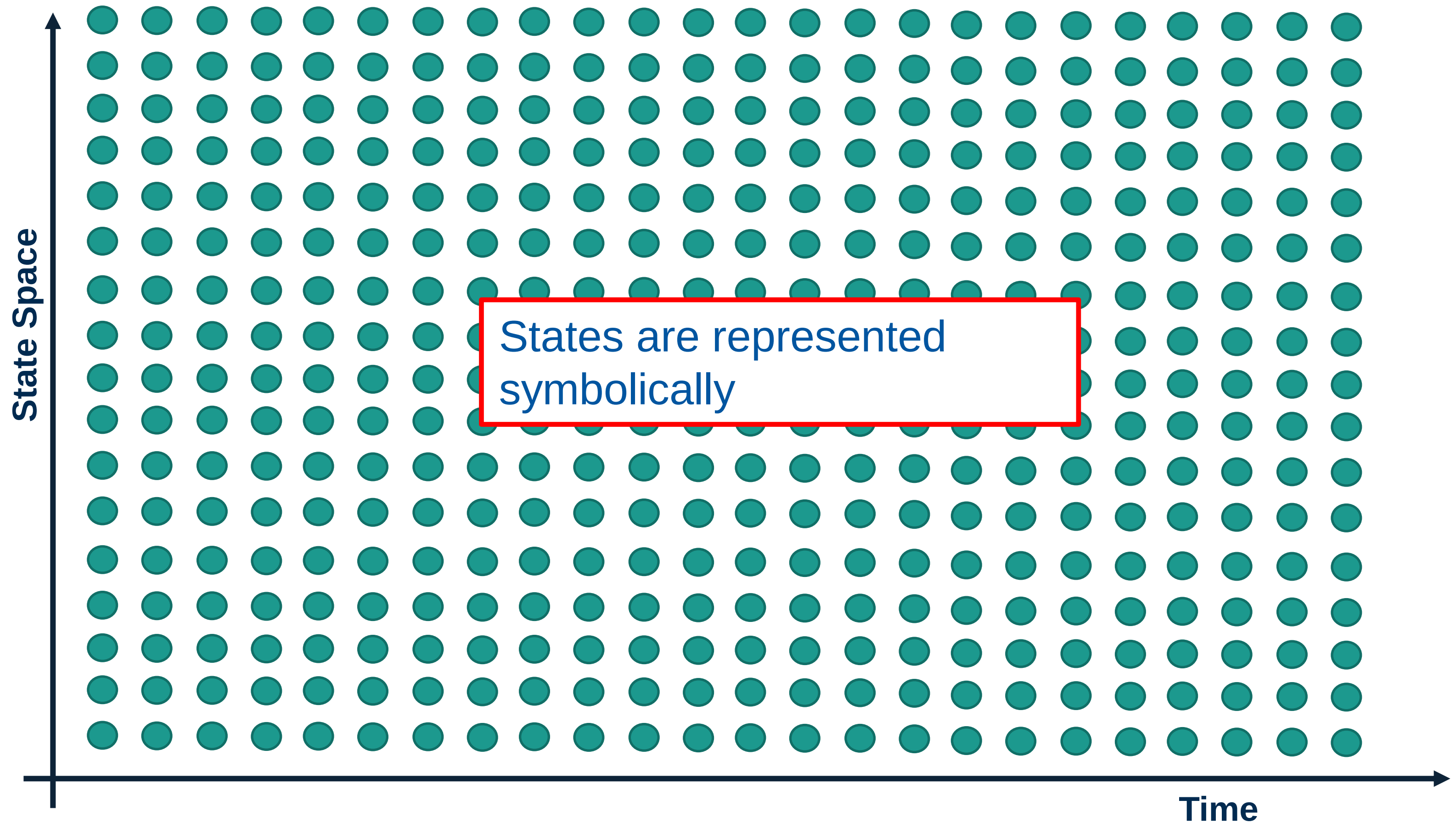
☹ Subject to time explosion



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

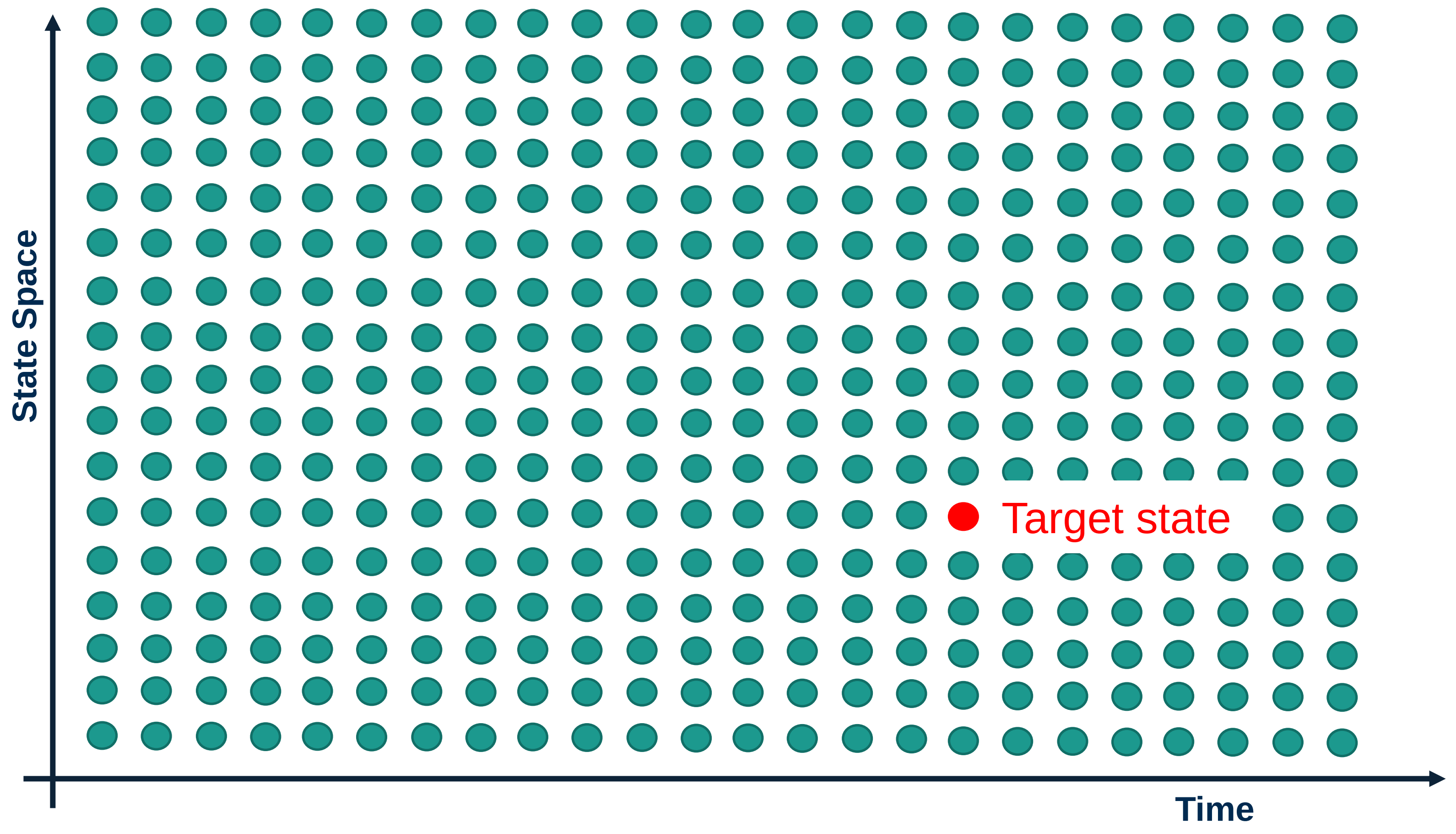


Formal Verification

States are represented symbolically

We define a target state

→ We try to demonstrate that this target state can be reached



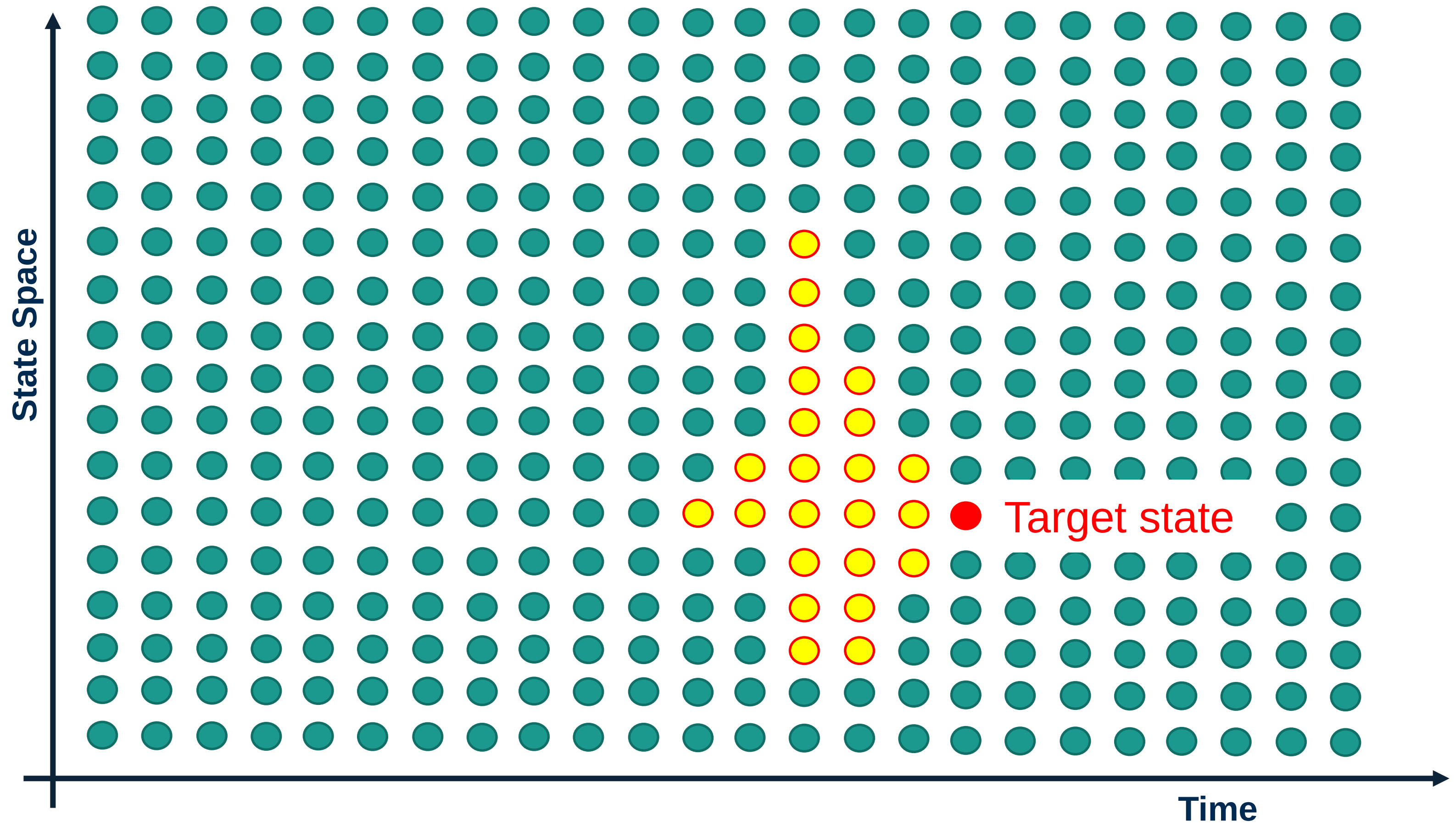
Formal Verification

States are represented symbolically

We define a target state

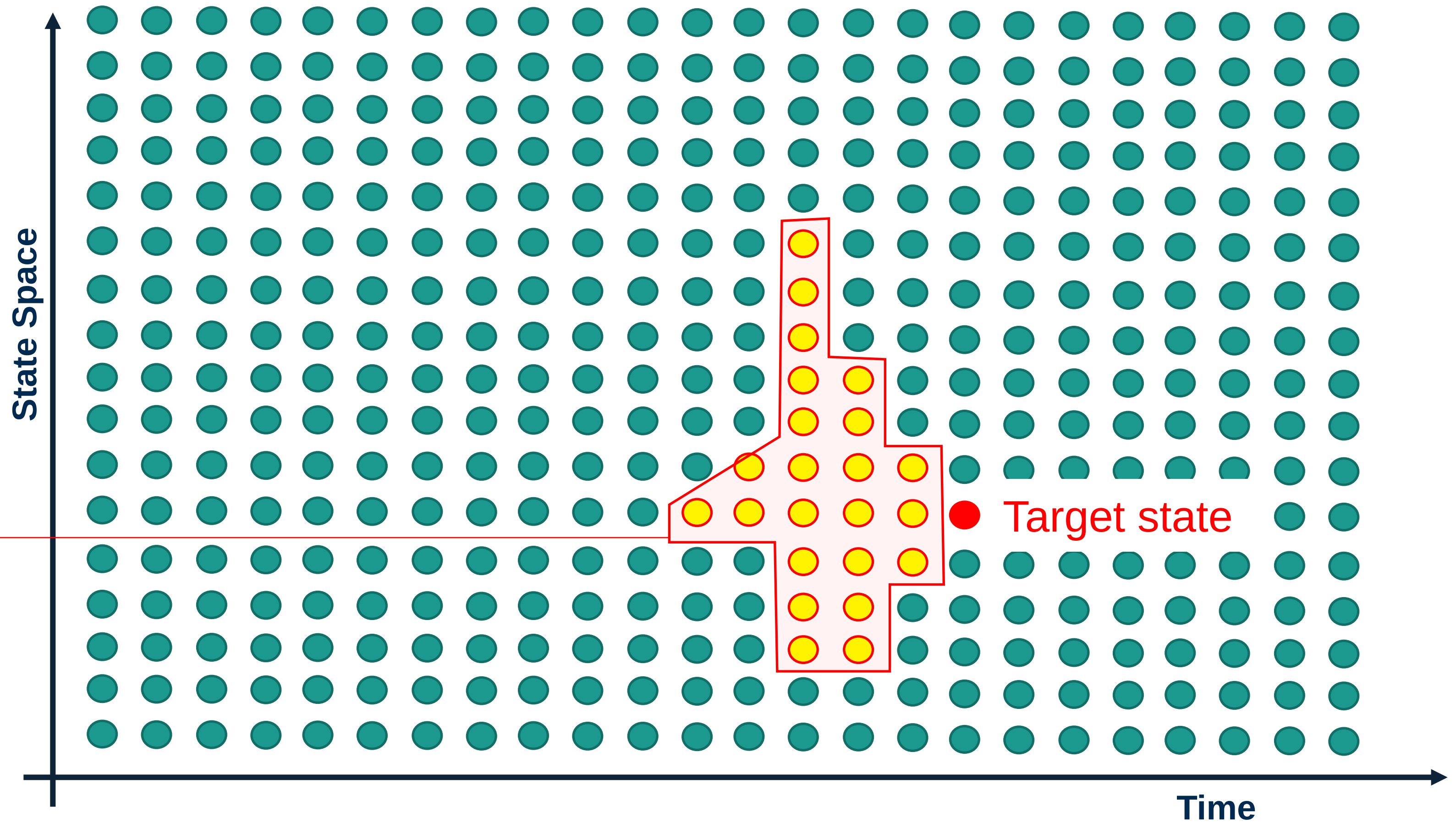
→ We try to demonstrate that this target state can be reached

→ We try to find a sequence that will fire the assertion



Formal Verification

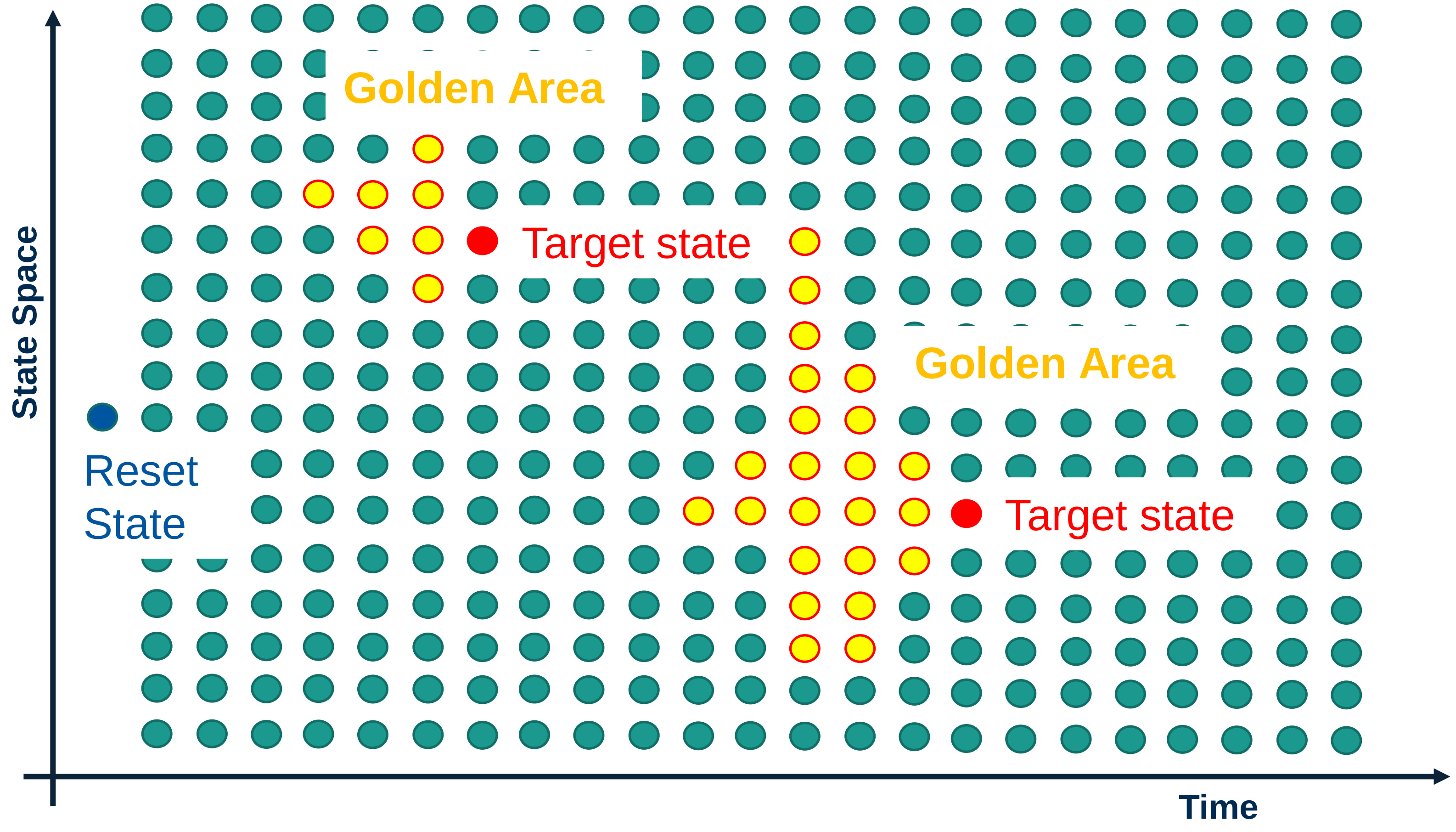
Golden Area



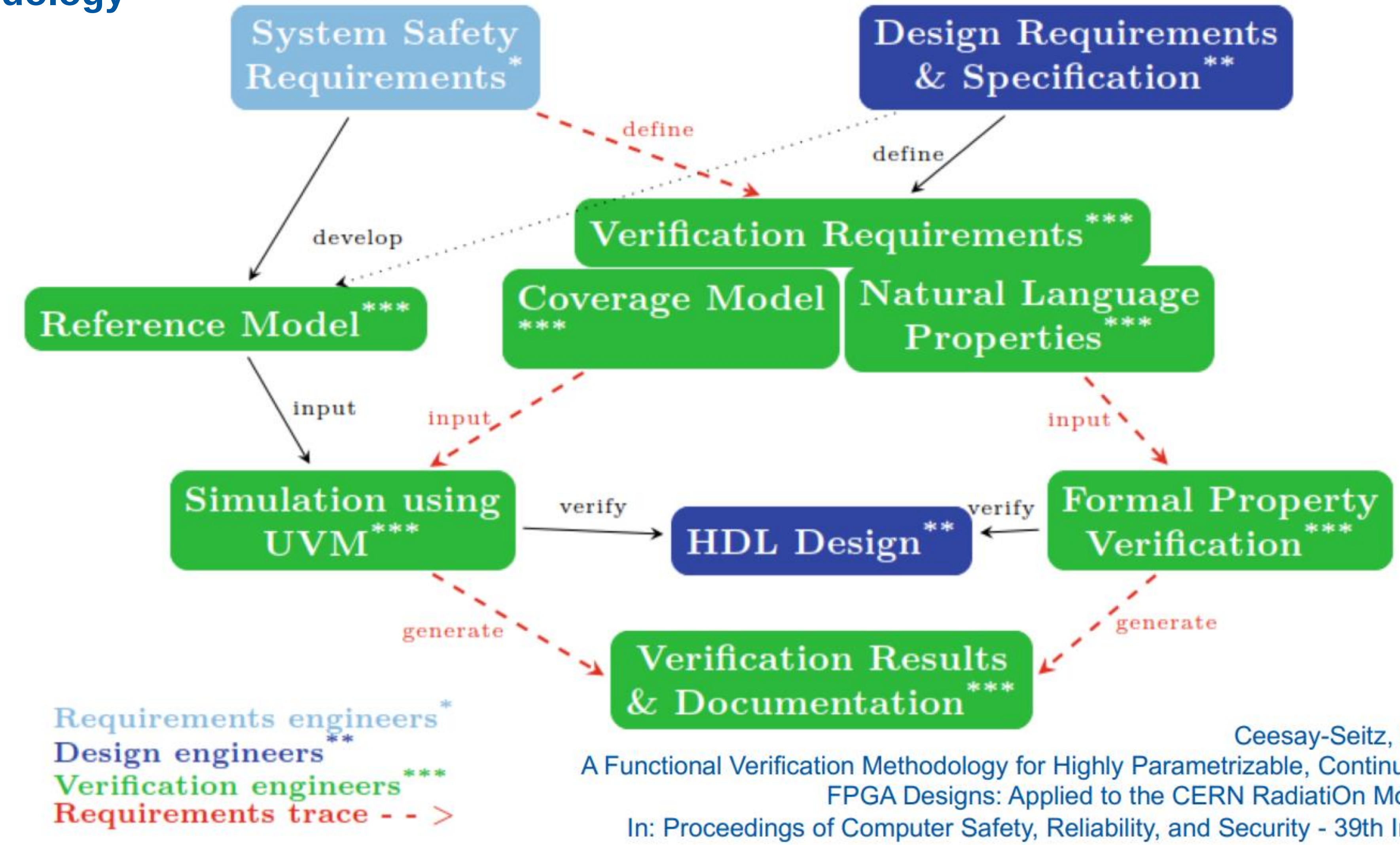
Formal Verification

States are represented symbolically

☹ Formal suffers from state space explosion

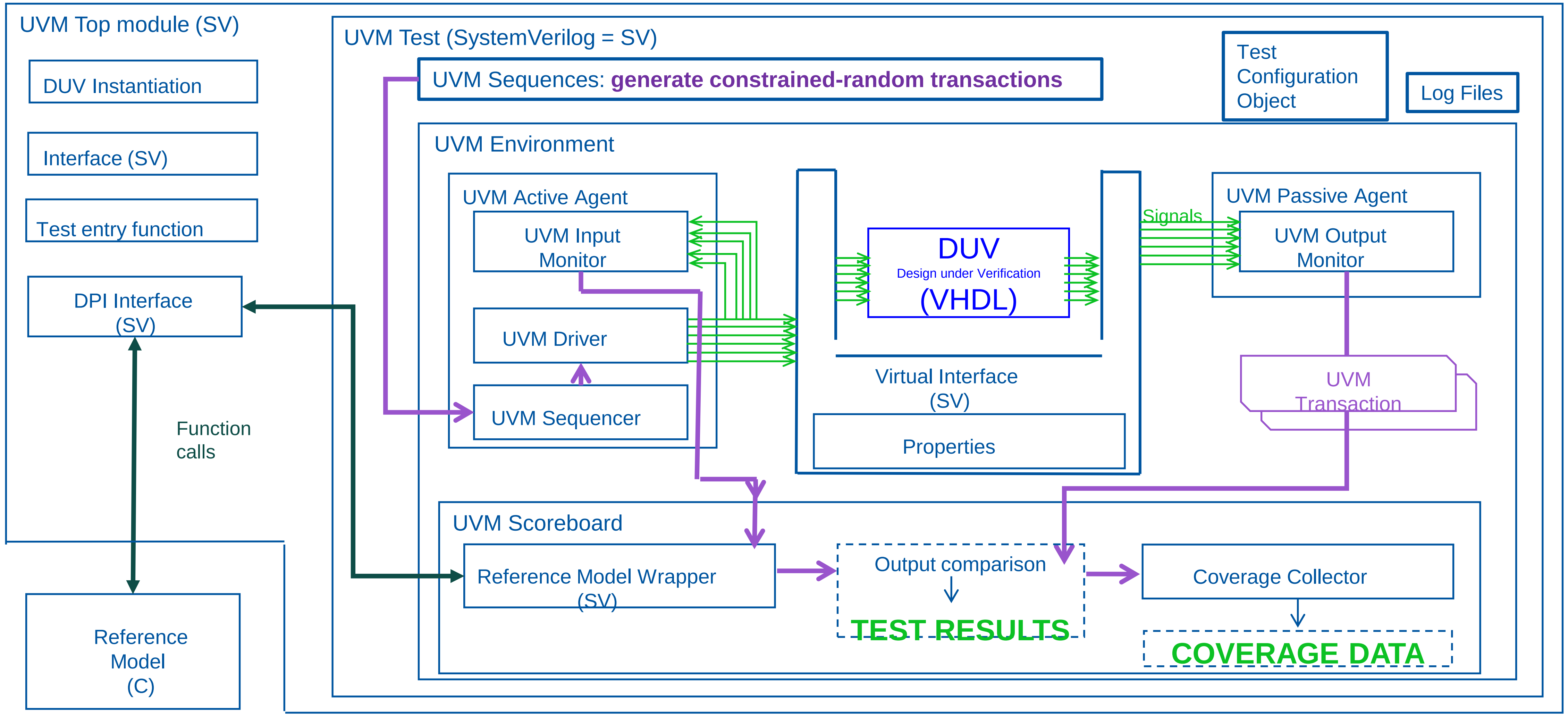


Our Methodology

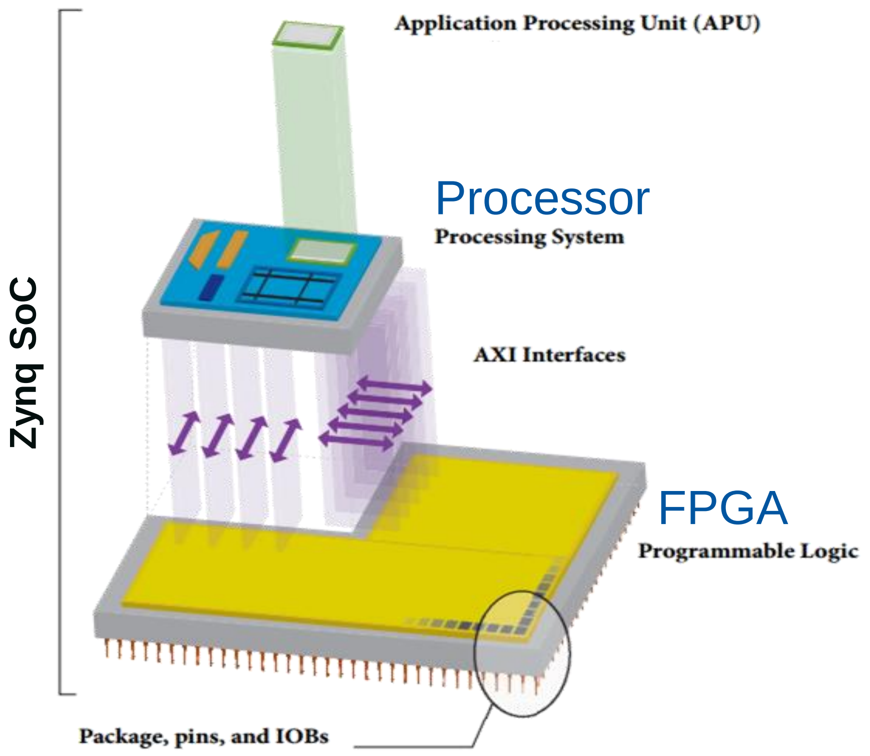

System Safety Requirements*
Design Requirements & Specification**
define
define
develop
Verification Requirements***
Reference Model***
Coverage Model***
Natural Language Properties***
input
input
input
Simulation using UVM***
verify
HDL Design**
verify
Formal Property Verification***
generate
generate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

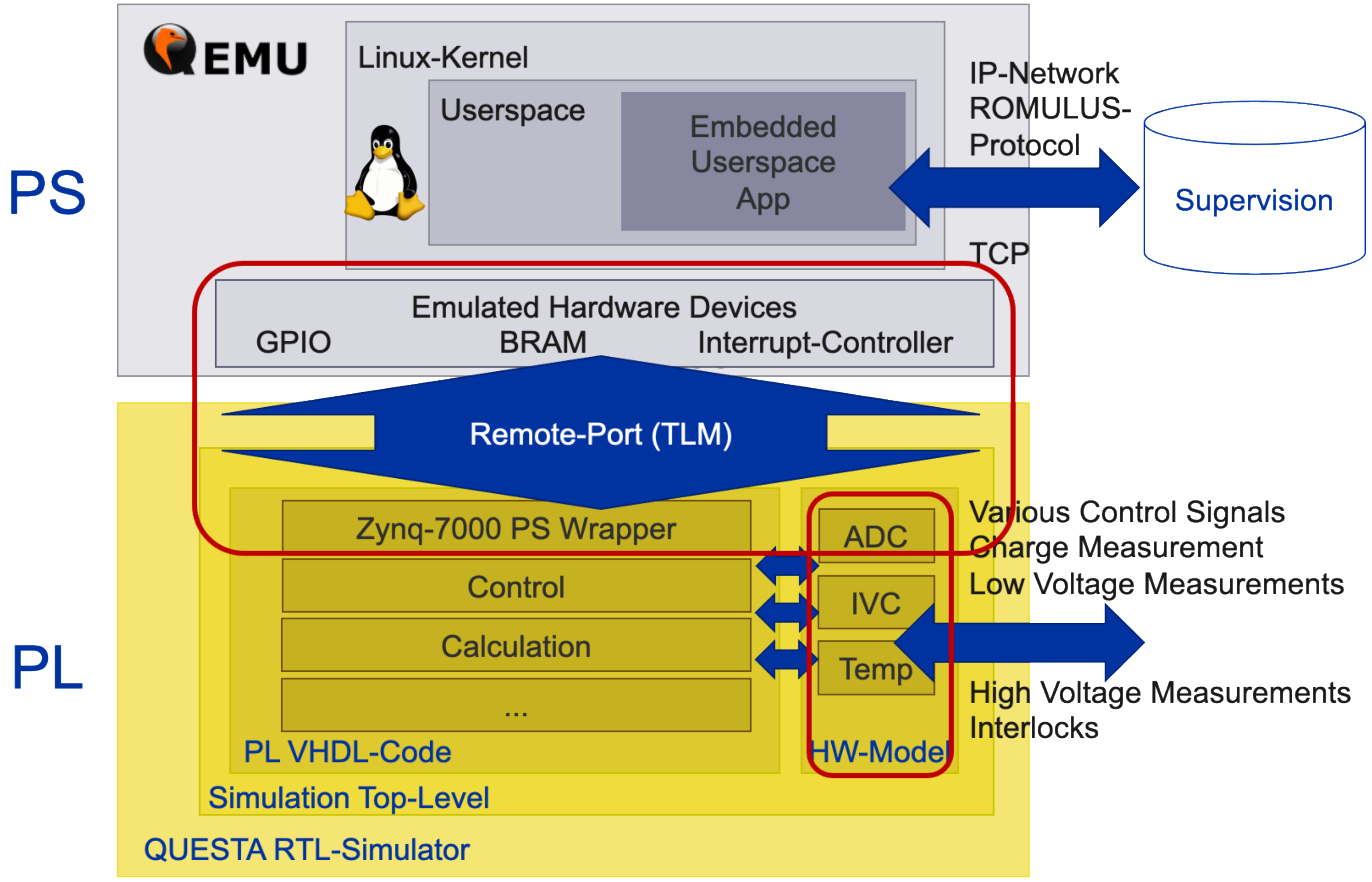
Ceesay-Seitz, K., Boukabache, H., Perrin, D.:
A Functional Verification Methodology for Highly Parametrizable, Continuously Operating Safety-Critical FPGA Designs: Applied to the CERN RadiatiOn Monitoring Electronics (CROME).
In: Proceedings of Computer Safety, Reliability, and Security - 39th International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

Our UVM (Universal Verification Methodology) Test Bench

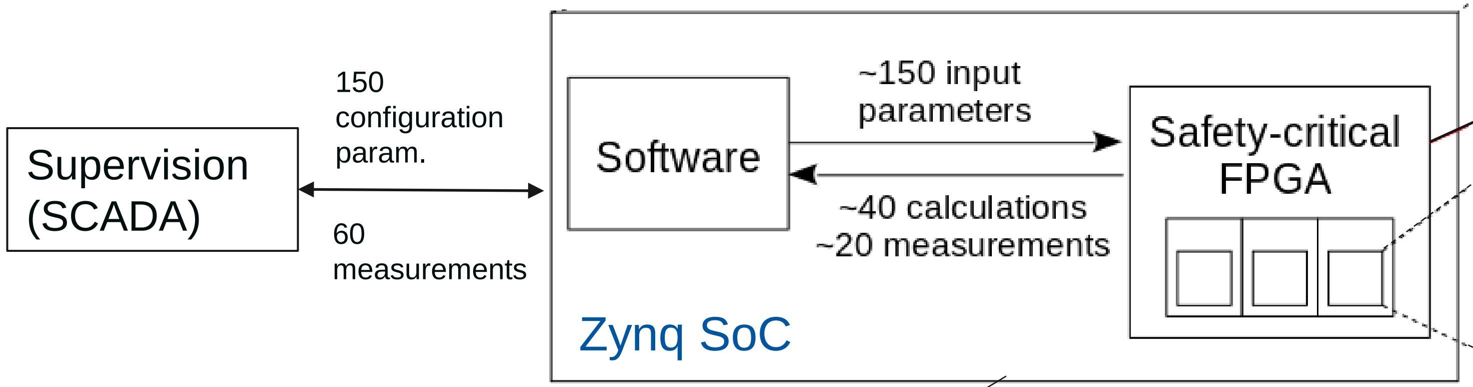


Our Co-Simulation environment





Verification examples



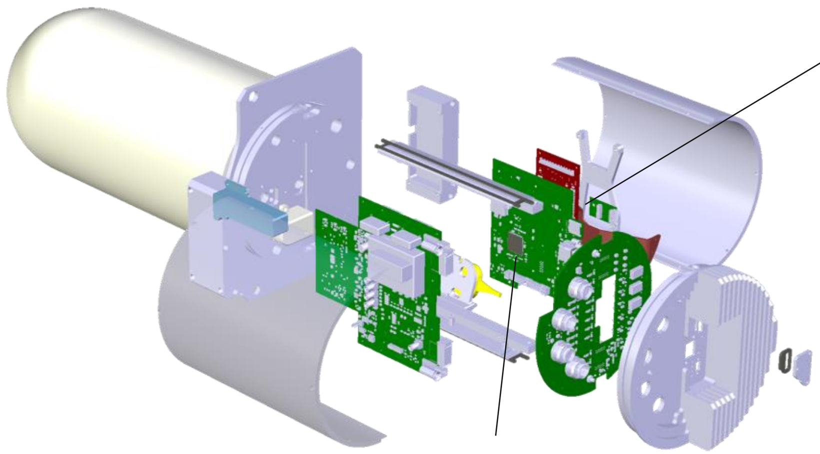
Alarm Units / Interlocking system

Alarm/Interlock Matrix:

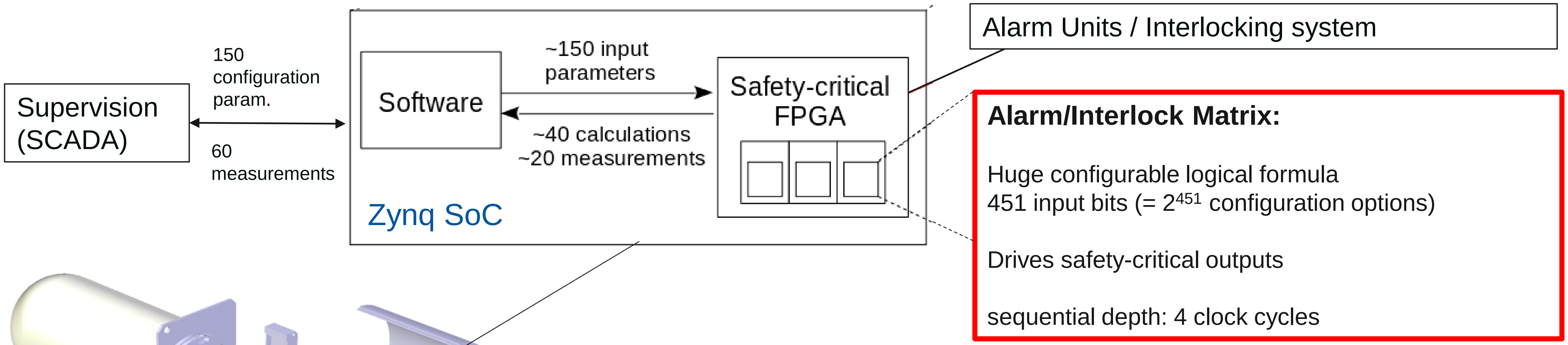
Huge configurable logical formula
451 input bits (= 2^{451} configuration options)

Drives safety-critical outputs

sequential depth: 4 clock cycles



Verification examples

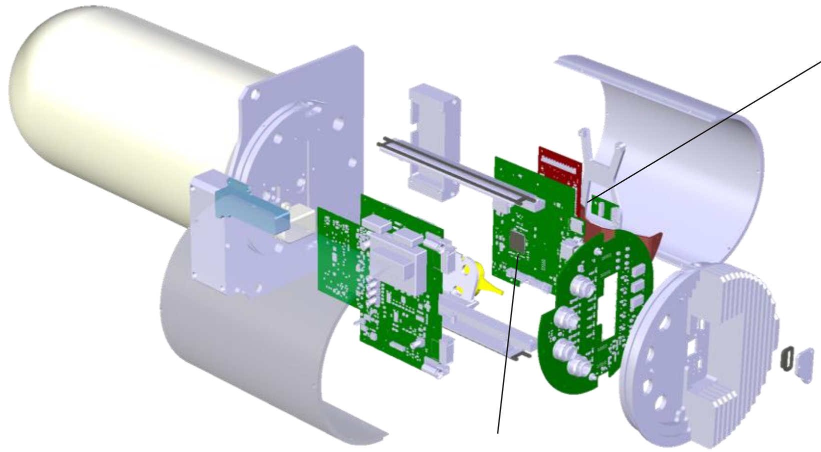


Alarm/Interlock Matrix:

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

Drives safety-critical outputs

sequential depth: 4 clock cycles



With a Reference model in SystemVerilog

(Only constraint: parameters do not change during 4 cycles of formula evaluation)

→ **46 properties proven in 33 seconds** (estimated simulation time: $8*10^{137}$ years)

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

Verification examples

Exhaustively Proved radiation dose **alarm generation**

Findings :

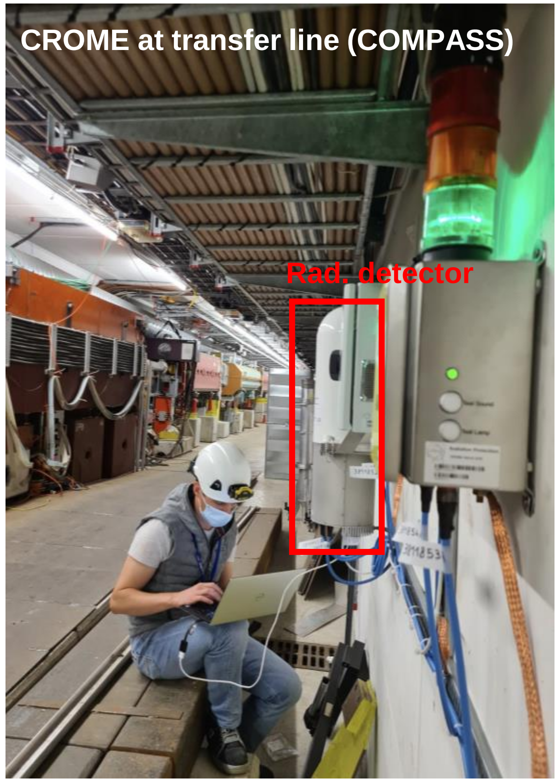
Undocumented design decision

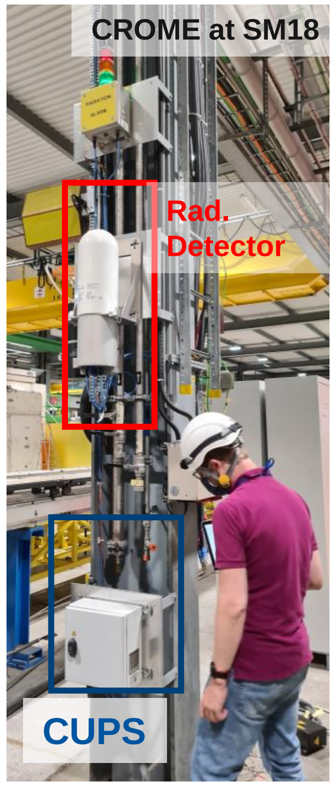
→ **Fault** in rounding mechanism only if internal result was negative
→ **Scenario not covered by simulation** (400000 stimuli applied)

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

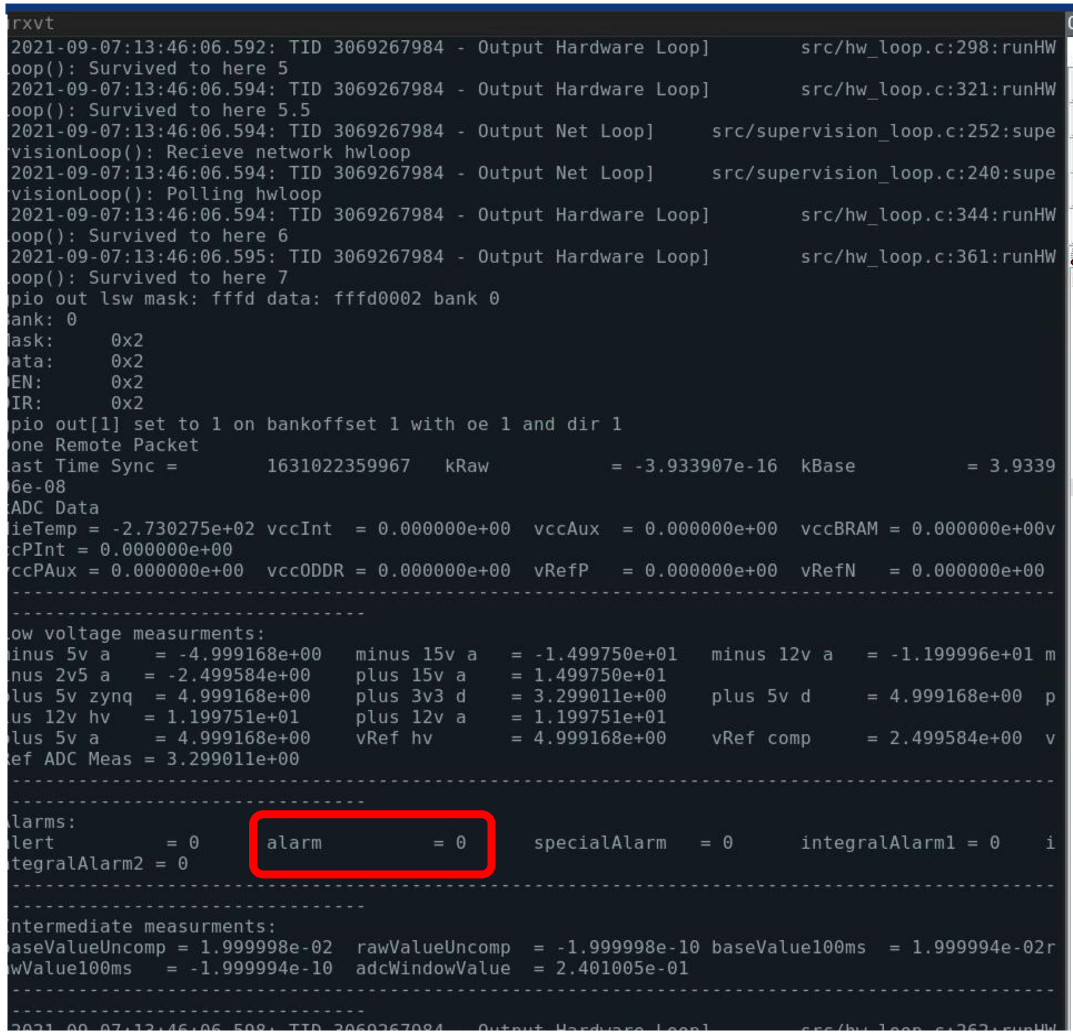
→ Found after 1 second with formal
→ Would require > 7 years of simulation

CROME Bulk - Wall-Mounted Version

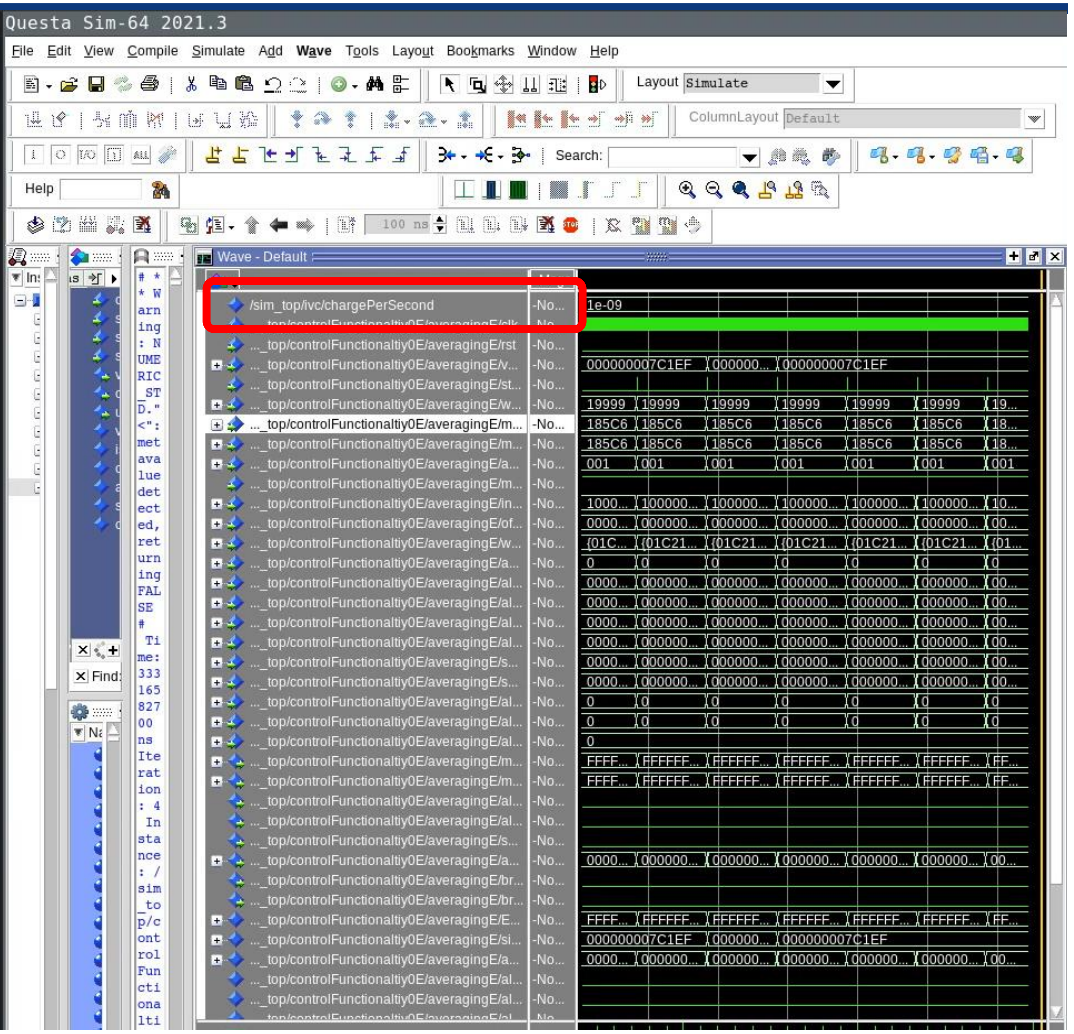




Co-simulation of a custom Linux distribution running user space apps, communicating with FPGA



```
rxvt
2021-09-07:13:46:06.592: TID 3069267984 - Output Hardware Loop]        src/hw_loop.c:298:runHW
oop(): Survived to here 5
2021-09-07:13:46:06.594: TID 3069267984 - Output Hardware Loop]        src/hw_loop.c:321:runHW
oop(): Survived to here 5.5
2021-09-07:13:46:06.594: TID 3069267984 - Output Net Loop]     src/supervision_loop.c:252:supe
visionLoop(): Recieve network hwloop
2021-09-07:13:46:06.594: TID 3069267984 - Output Net Loop]     src/supervision_loop.c:240:supe
visionLoop(): Polling hwloop
2021-09-07:13:46:06.594: TID 3069267984 - Output Hardware Loop]        src/hw_loop.c:344:runHW
oop(): Survived to here 6
2021-09-07:13:46:06.595: TID 3069267984 - Output Hardware Loop]        src/hw_loop.c:361:runHW
oop(): Survived to here 7
pio out lsw mask: fffd data: fffd0002 bank 0
ank: 0
ask:     0x2
ata:     0x2
EN:      0x2
IR:      0x2
pio out[1] set to 1 on bankoffset 1 with oe 1 and dir 1
one Remote Packet
ast Time Sync =        1631022359967   kRaw           = -3.933907e-16  kBase          = 3.9339
6e-08
ADC Data
ieTemp = -2.730275e+02 vccInt  = 0.000000e+00  vccAux  = 0.000000e+00  vccBRAM = 0.000000e+00v
cPInt = 0.000000e+00
ccPAux = 0.000000e+00  vccODDR = 0.000000e+00  vRefP   = 0.000000e+00  vRefN   = 0.000000e+00
----------------------------------------------------------------------------------------------
-------------------------------
ow voltage measurments:
inus 5v a     = -4.999168e+00   minus 15v a    = -1.499750e+01   minus 12v a    = -1.199996e+01 m
nus 2v5 a    = -2.499584e+00    plus 15v a     = 1.499750e+01
lus 5v zynq  = 4.999168e+00    plus 3v3 d     = 3.299011e+00    plus 5v d      = 4.999168e+00  p
us 12v hv    = 1.199751e+01    plus 12v a     = 1.199751e+01
lus 5v a     = 4.999168e+00    vRef hv        = 4.999168e+00    vRef comp      = 2.499584e+00  v
ef ADC Meas = 3.299011e+00
----------------------------------------------------------------------------------------------
-------------------------------
larms:
lert           = 0      alarm          = 0      specialAlarm   = 0      integralAlarm1 = 0    i
tegralAlarm2 = 0
----------------------------------------------------------------------------------------------
-------------------------------
ntermediate measurments:
aseValueUncomp = 1.999998e-02  rawValueUncomp   = -1.999998e-10 baseValue100ms   = 1.999994e-02r
wValue100ms    = -1.999994e-10  adcWindowValue   = 2.401005e-01
----------------------------------------------------------------------------------------------
-------------------------------
```


Questa Sim-64 2021.3
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
Layout Simulate
ColumnLayout Default
Search:
Help
Wave - Default
/sim_top/ivc/chargePerSecond
1e-09

www.cern.ch