Génération de tests basés sur les modèles pour des systèmes sur puce avec cohérence de caches

Massimo Zendri & Abderahman Kriouile

STMicroelectronics DCG / IP dev / FVS

Model based test generation for cache coherent Systems On Chips

Massimo Zendri & Abderahman Kriouile

STMicroelectronics DCG / IP dev / FVS

Where you find us

Our MEMS & Sensors
are augmenting
the consumer experience

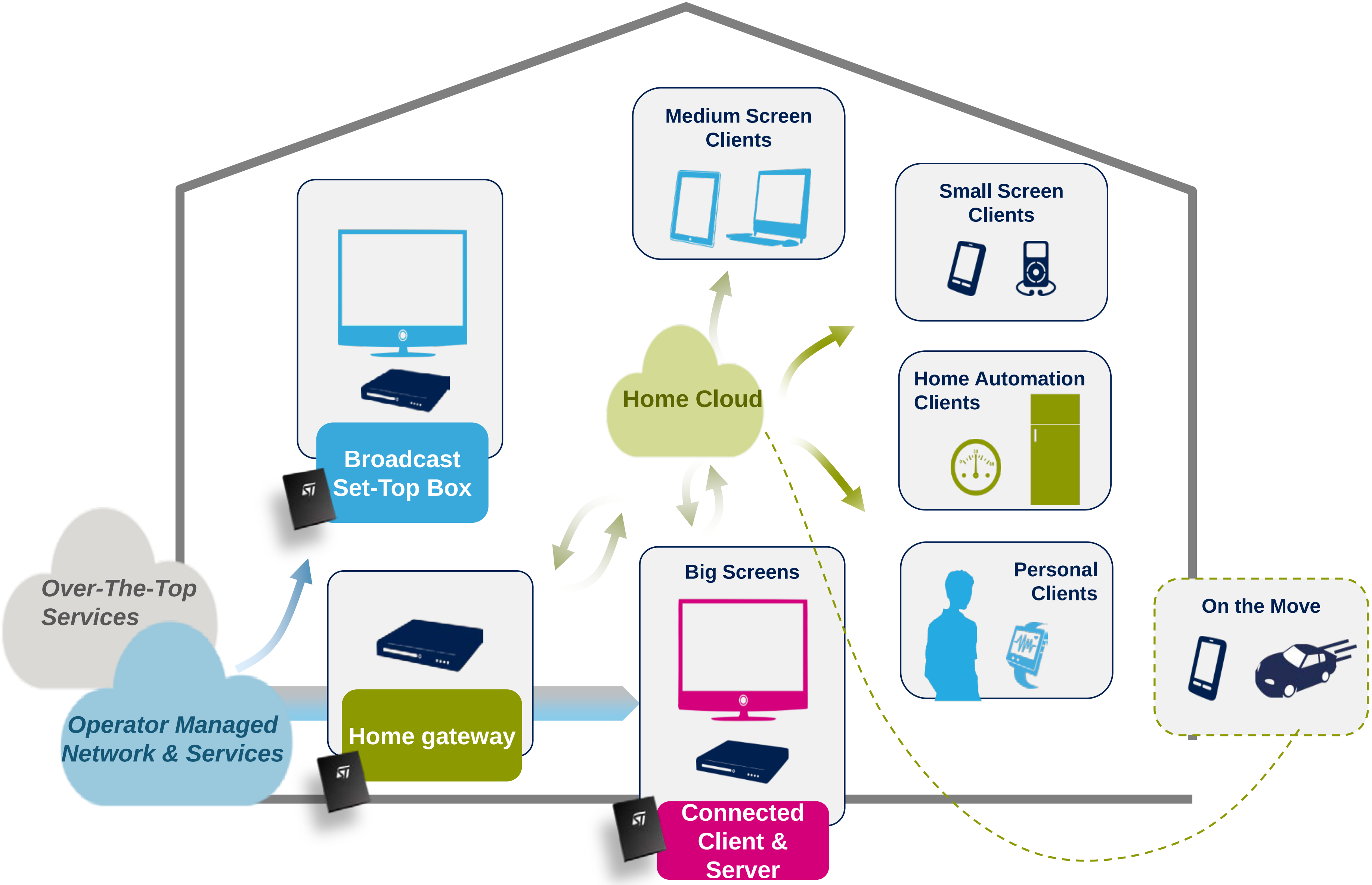
Our digital consumer products
are powering the augmented
digital lifestyle

Our automotive products
are making driving safer,
greener and more
entertaining

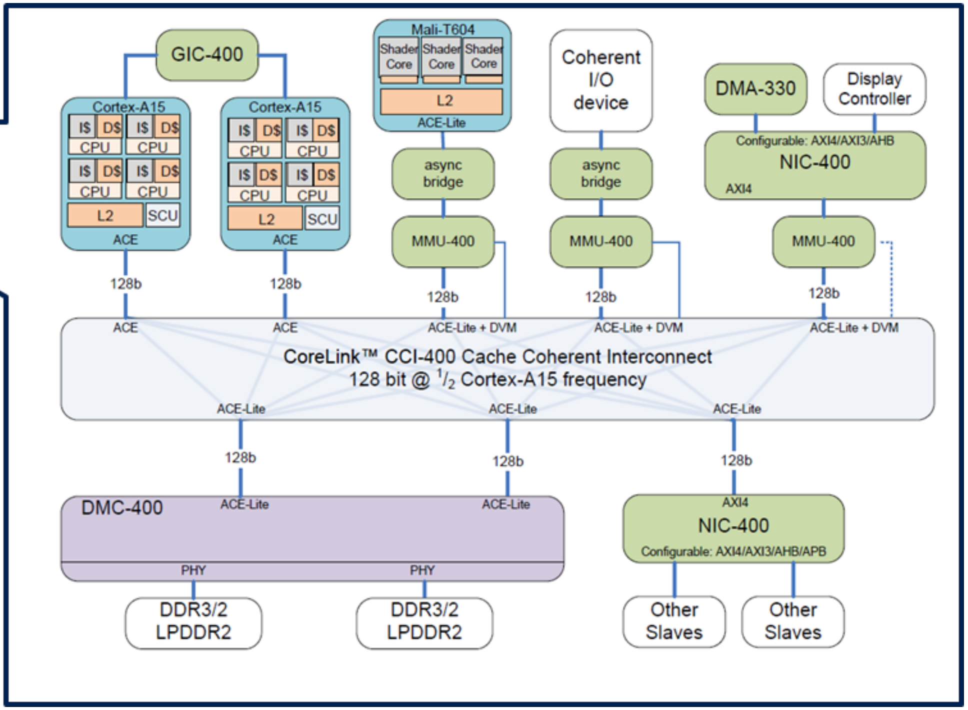
Our Microcontrollers
are everywhere
making everything smarter
and more secure

Our smart power products
are allowing our mobile products to operate longer
and making more of our energy resources

Towards the Home Cloud

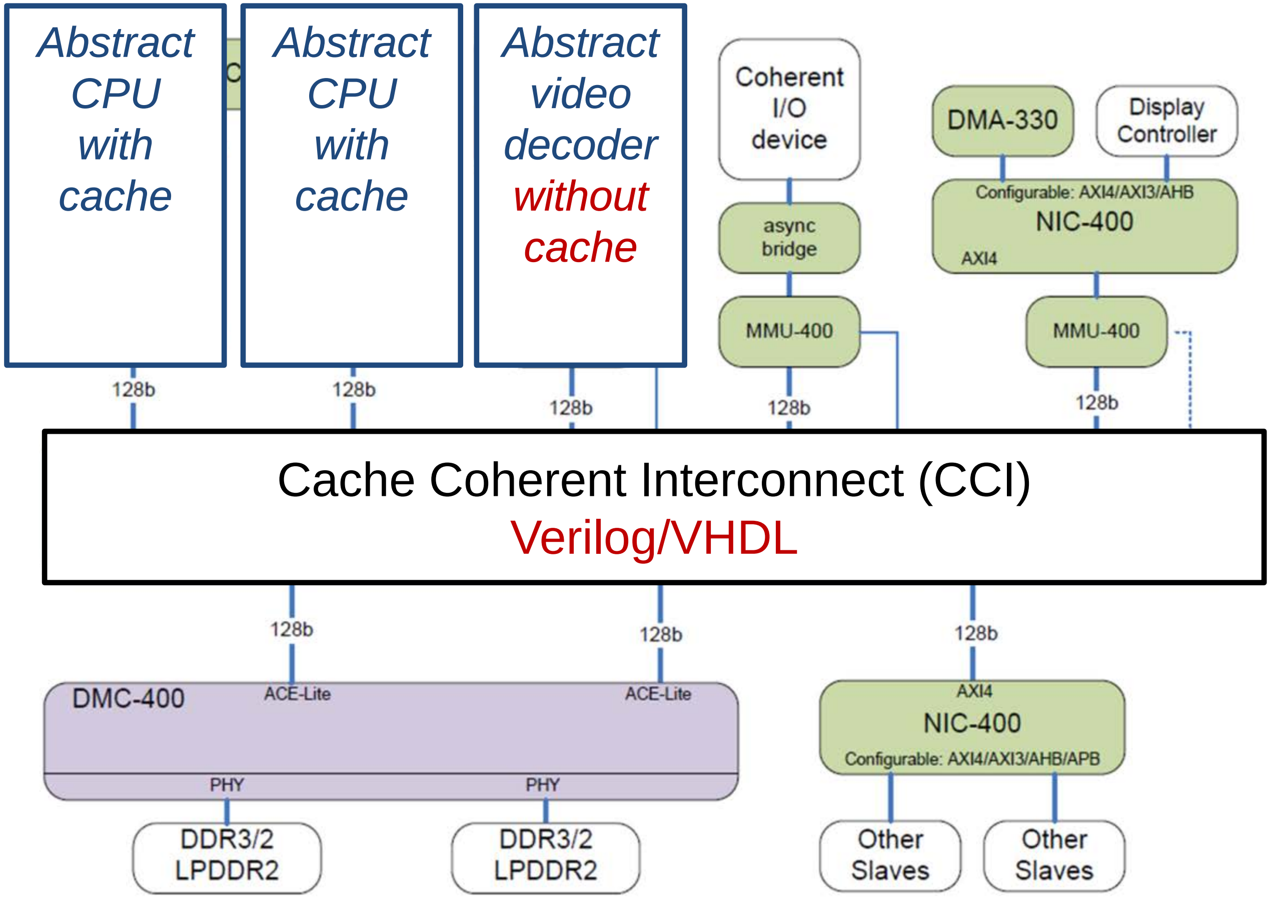


Heterogeneous System-on-Chip (SoC)

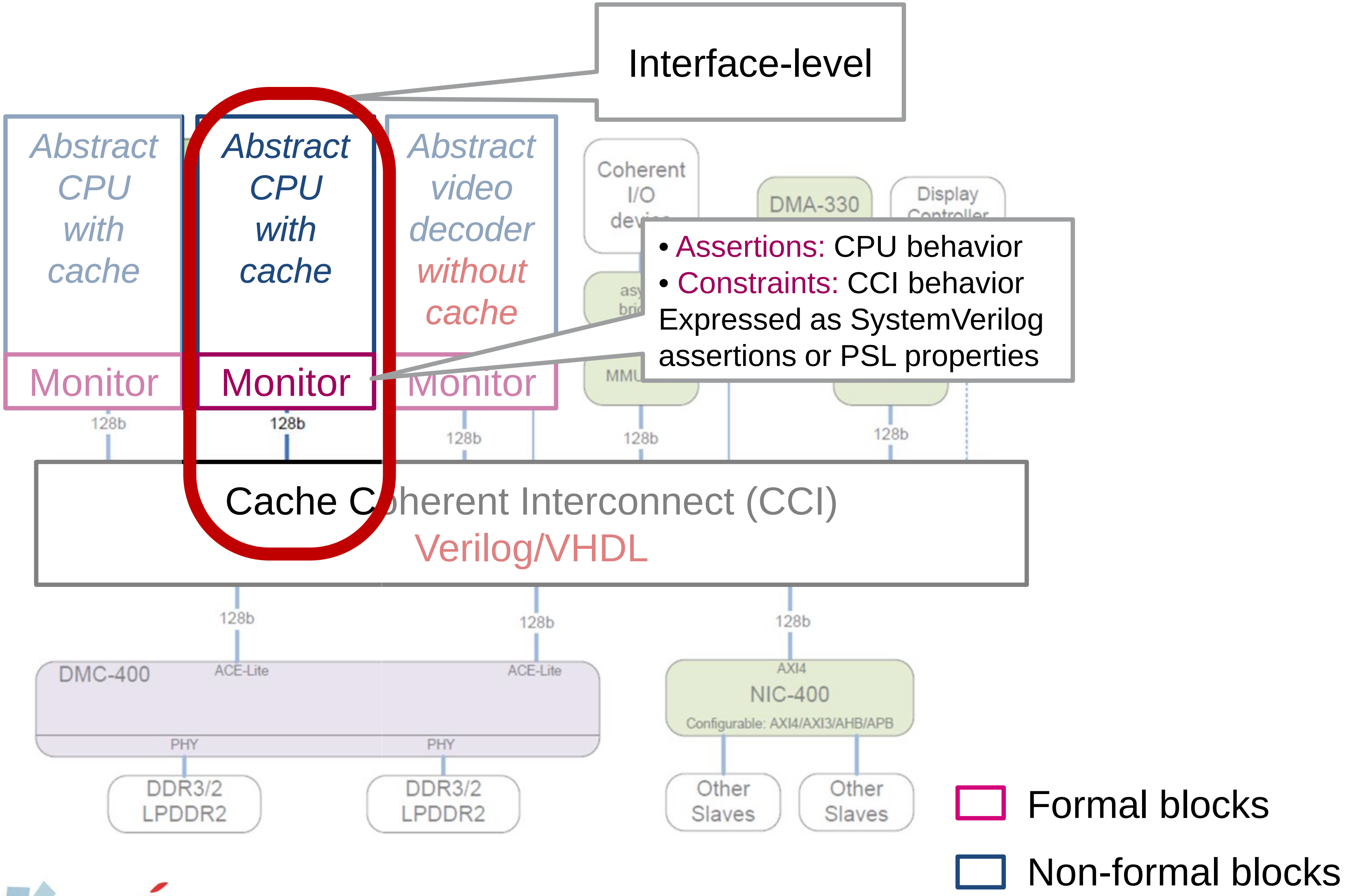


Need for System-Level Cache Coherency
ARM proposed ACE specification:
standard for system level cache coherency

Simulation-Based Testing



Simulation-Based Testing



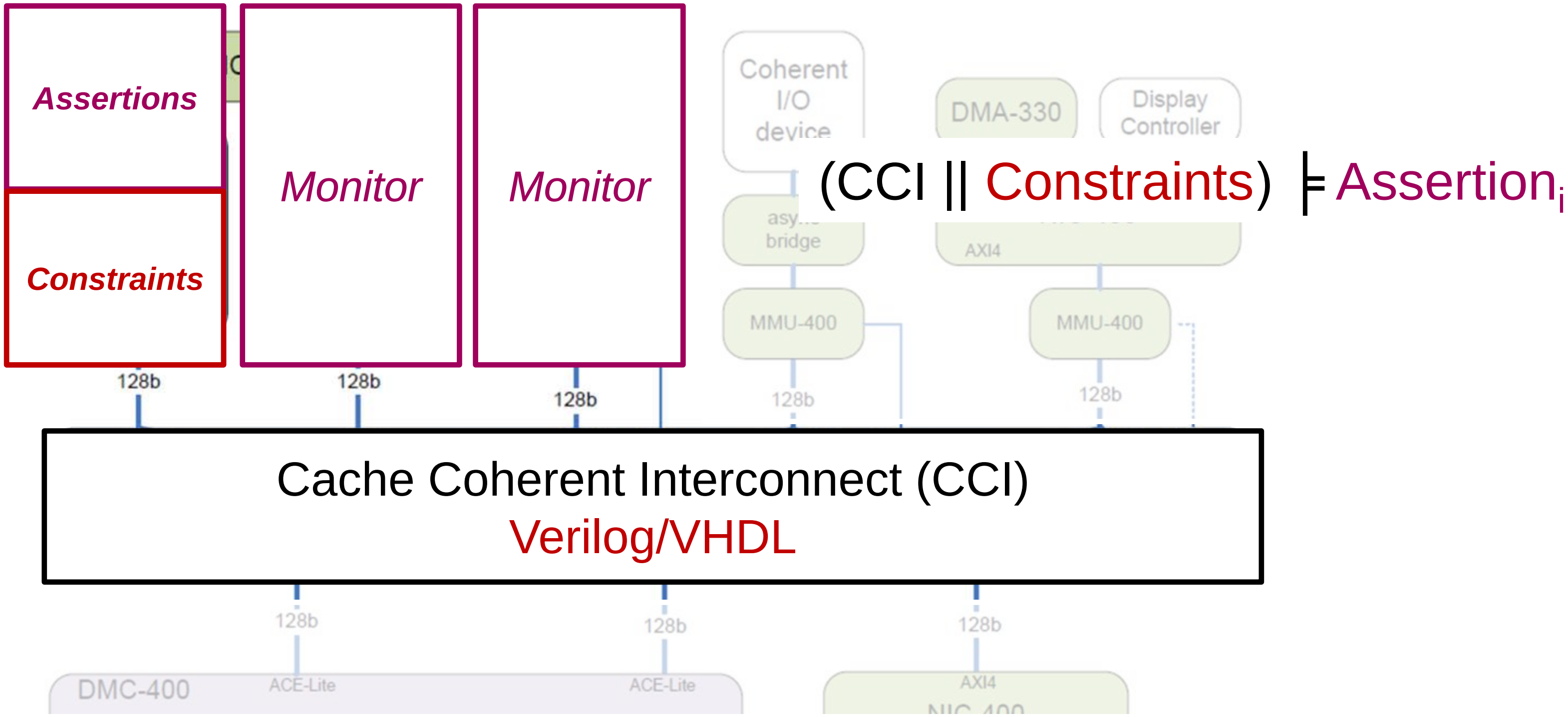
Interface-level

- Assertions: CPU behavior
- Constraints: CCI behavior

Expressed as SystemVerilog assertions or PSL properties

Model Checking

(without running any test)

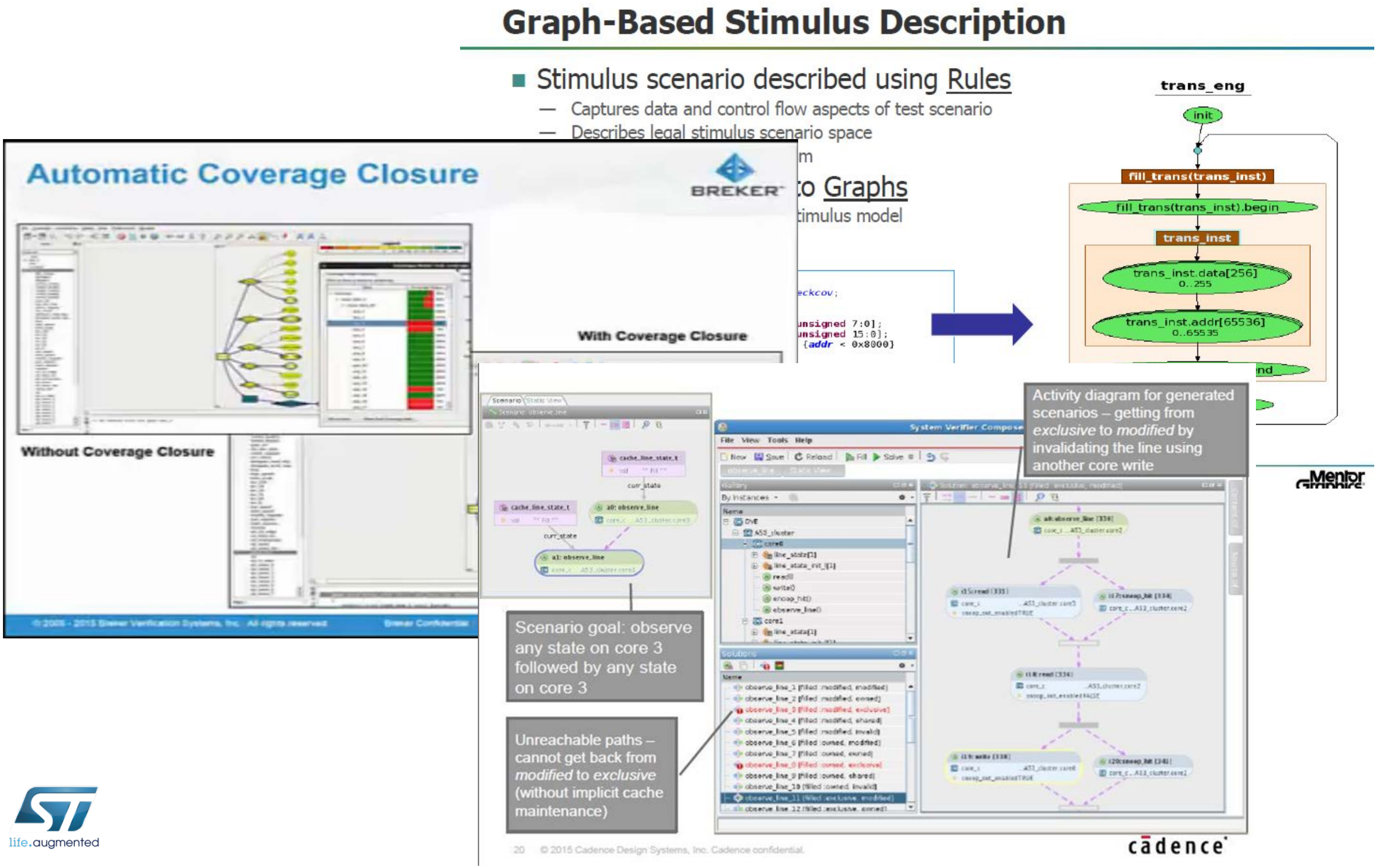


$(\text{CCI} \parallel \text{Constraints}) \models \text{Assertion}_i$

- Applying restrictions for more exploration
- Limitation due to state-space explosion problem

State of the Art

HW Model Based Test Generator



Graph-Based Stimulus Description

- Stimulus scenario described using Rules
  - Captures data and control flow aspects of test scenario
  - Describes legal stimulus scenario space
  - m
- to Graphs
  - timulus model

```
eckcov;

unsigned 7:0];
unsigned 15:0];
 {addr < 0x8000}
```

Automatic Coverage Closure

With Coverage Closure

Without Coverage Closure

Breker Confidential

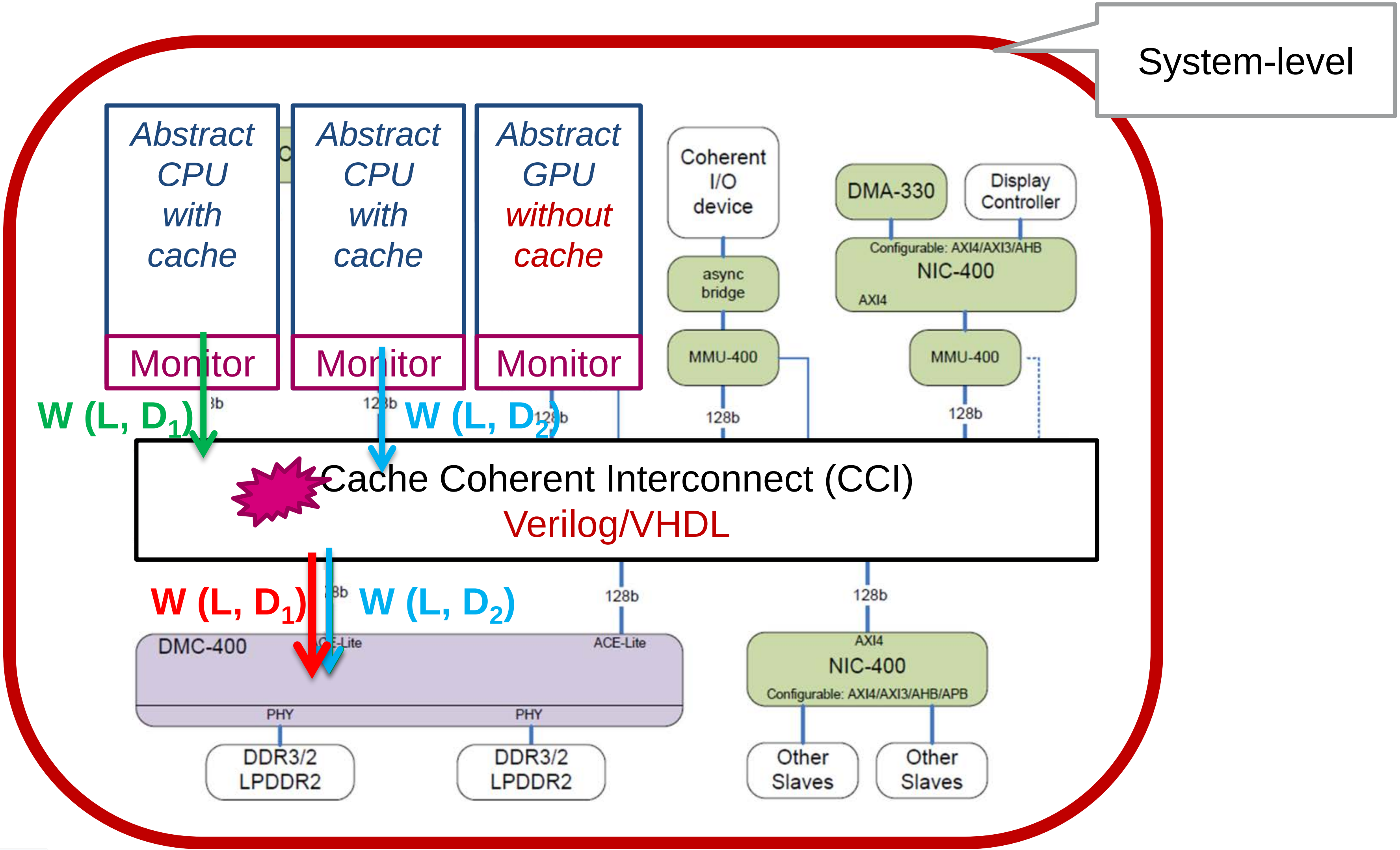
```
observe_line_1 [filled :modified, modified]
observe_line_2 [filled :modified, owned]
observe_line_3 [filled :modified, exclusive]
observe_line_4 [filled :modified, shared]
observe_line_5 [filled :modified, invalid]
observe_line_6 [filled :owned, modified]
observe_line_7 [filled :owned, owned]
observe_line_8 [filled :owned, exclusive]
observe_line_9 [filled :owned, shared]
observe_line_10 [filled :owned, invalid]
observe_line_11 [filled :exclusive, modified]
observe_line_12 [filled :exclusive, owned]
```

Activity diagram for generated scenarios – getting from *exclusive* to *modified* by invalidating the line using another core write

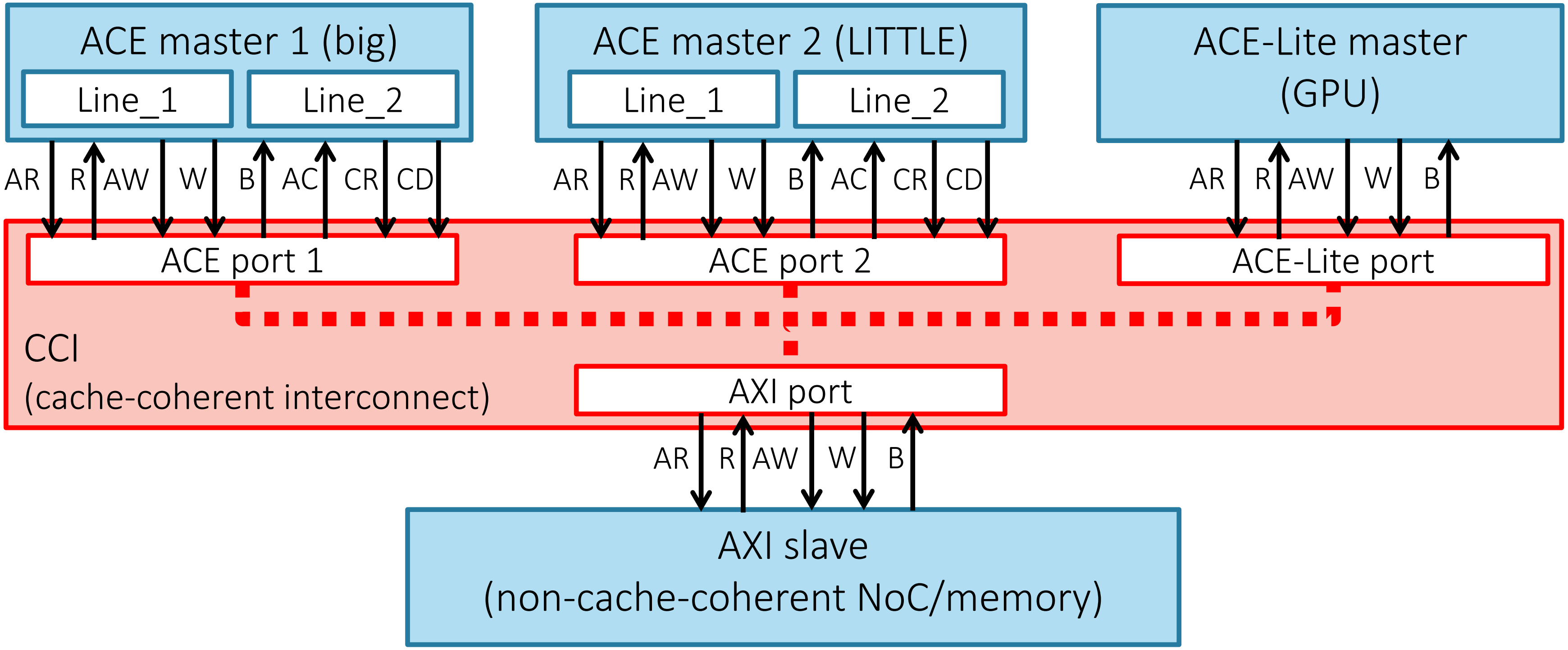
Scenario goal: observe any state on core 3 followed by any state on core 3

Unreachable paths – cannot get back from *modified* to *exclusive* (without implicit cache maintenance)

Need for System-Level Verification



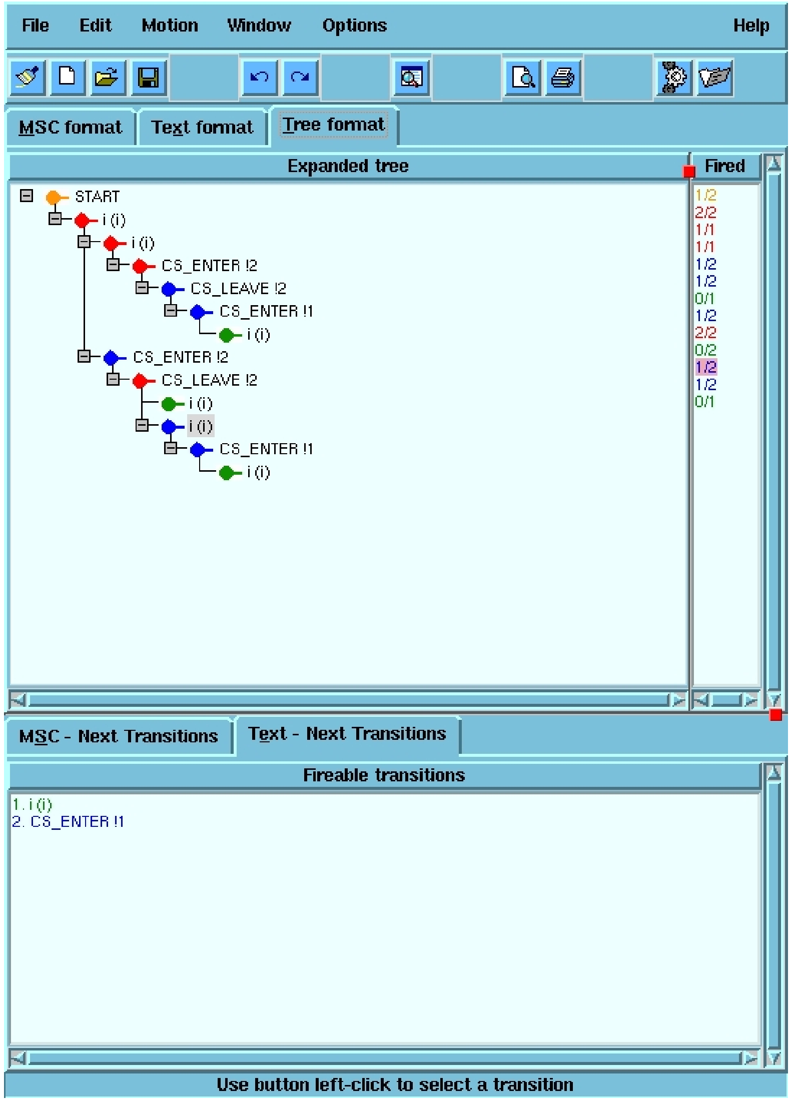
Formal Model of an ACE-based SoC



- Interface transfers modeled by rendezvous
- 3400 lines of LNT code derived from ACE specification
- Parametric: #masters, forbidden ACE transactions, …
- [Kriouile-Serwe-13] Formal Analysis of the ACE Specification for Cache Coherent Systems-on-Chip, FMICS, LNCS 8187, 2013

CADP: OCIS *(Open/Cæsar Interactive Simulator)*

- language-independent
- tree-like scenarios
- save/load scenarios
- source code access
- dynamic recompile

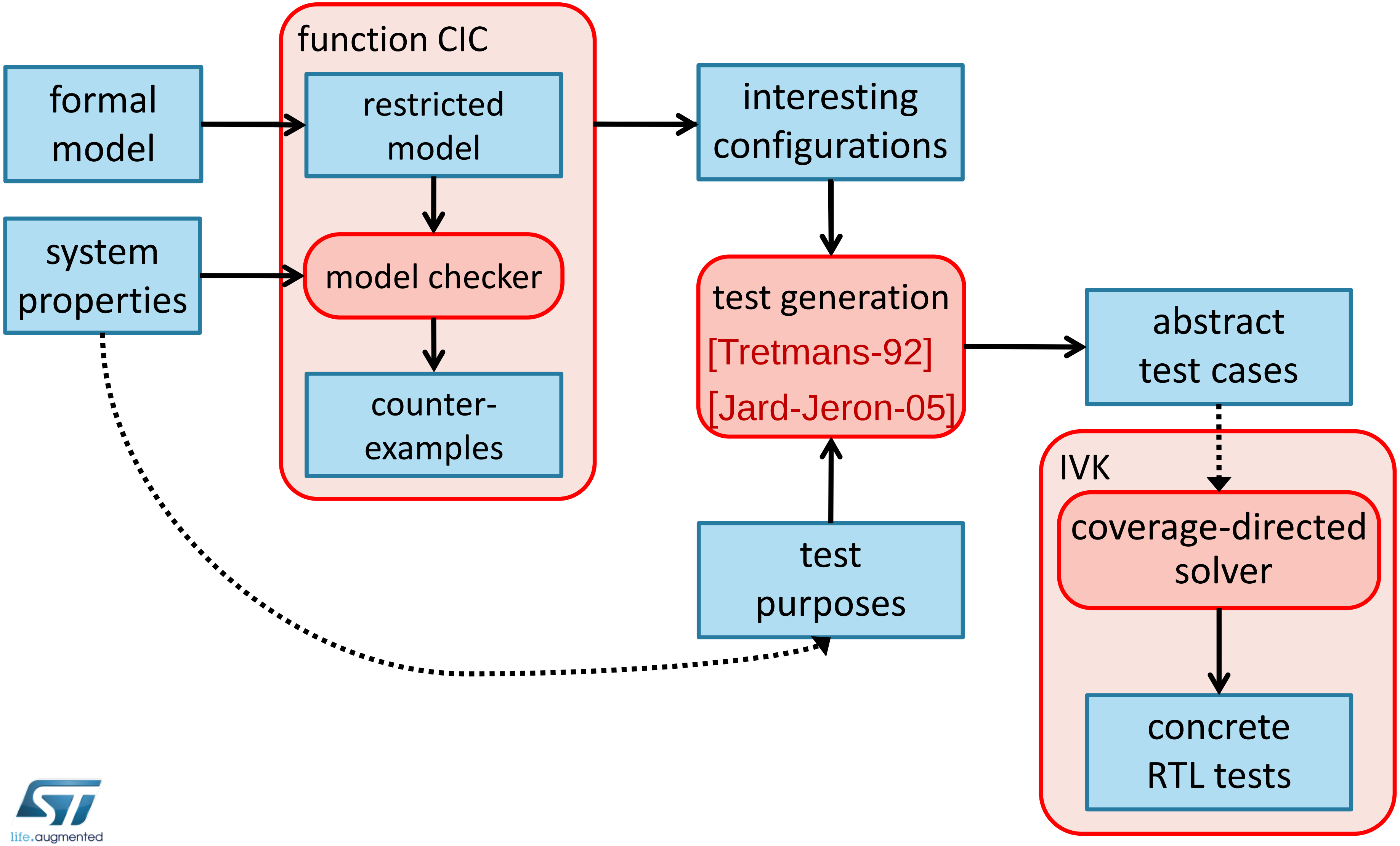


```
START                              1/2
  i (i)                            2/2
    i (i)                          1/1
      CS_ENTER !2                  1/1
        CS_LEAVE !2                1/2
          CS_ENTER !1              1/2
            i (i)                  0/1
    CS_ENTER !2                    1/2
      CS_LEAVE !2                  2/2
        i (i)                      0/2
        i (i)                      1/2
          CS_ENTER !1              1/2
            i (i)                  0/1
```

```
1. i (i)
2. CS_ENTER !1
```

Use button left-click to select a transition

Generation of System-Level Test Cases



function CIC

formal model

restricted model

interesting configurations

system properties

model checker

counter-examples

test generation [Tretmans-92] [Jard-Jeron-05]

abstract test cases

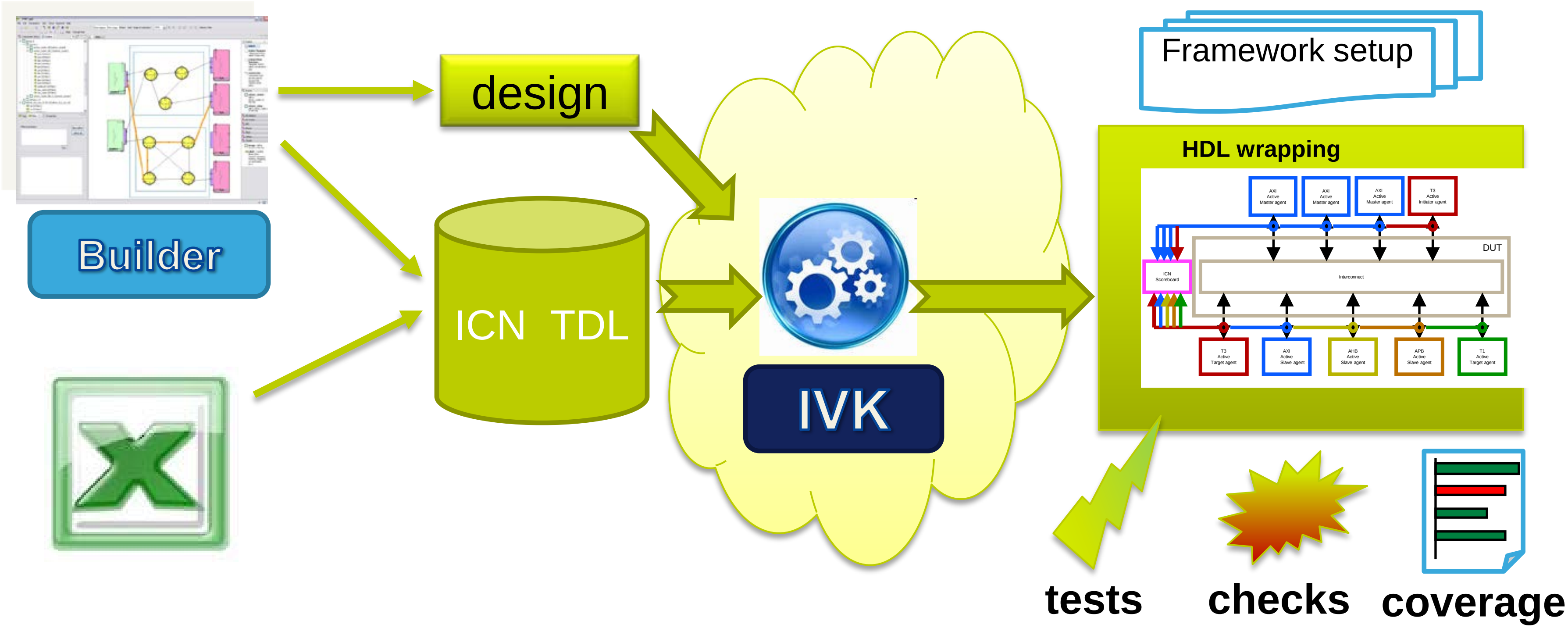
test purposes

IVK

coverage-directed solver

concrete RTL tests

IVK (Interconnect Verification Kit):
Automated Interconnect Testbench Generation



- **Inputs**

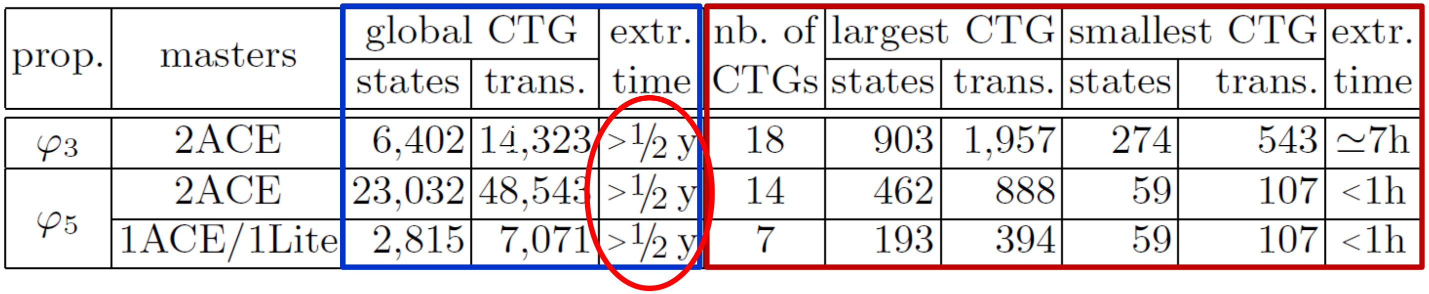
  Architectural description (TDL) either generated by interconnect designers GUI or through Excel flow

- **Outputs**

  Full Verification Environment, including sequences and coverage models

Several Kinds of Derived Tests

- 39 + 3 generated CTGs (Complete Test Graphs)



prop.	masters	global CTG		extr.	nb. of	largest CTG		smallest CTG		extr.
		states	trans.	time	CTGs	states	trans.	states	trans.	time
φ_3	2ACE	6,402	14,323	>½ y	18	903	1,957	274	543	≃7h
φ_5	2ACE	23,032	48,543	>½ y	14	462	888	59	107	<1h
	1ACE/1Lite	2,815	7,071	>½ y	7	193	394	59	107	<1h

- 296 simple system-level tests
  - for each correct initial state with two masters possibly sharing a memory line, initiate all permitted transitions
  - check correct behavior of the Cache Coherent Interconnect (e.g., generation of corresponding snoops)
- 10 sequence tests to recreate counter-examples
  - concurrency between transactions
  - conditioned by response of the Cache Coherent Interconnect

Results

- 300 IVK tests generated
- Many problems identified on the verification environment (VIP components)
- System level assertions to check system behavior
- 100% coverage of system level assertions
- Reproduction of 1 suspected architectural issue
- Used on 2 currently developed products (codenamed Orly3 and Barcelona)