



Advanced methodologies used for top-level verification of mixed signal products

Roger Witlox/Marcel Oosterhuis

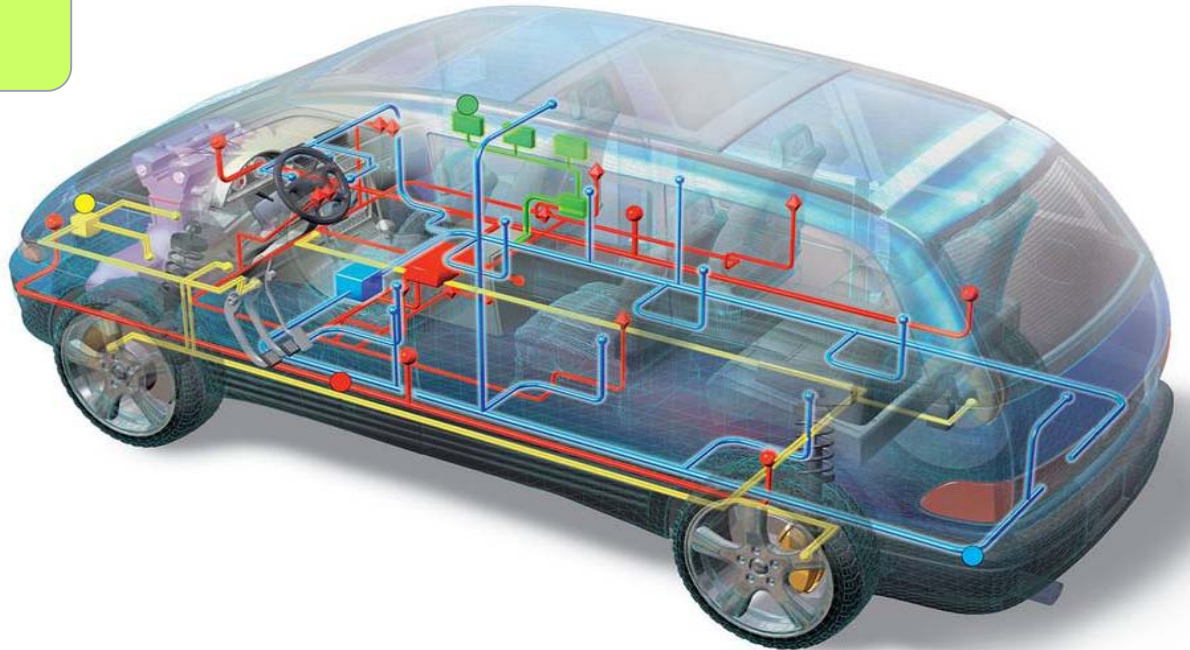
NXP, BU Automotive, PL Integrated IVN & Flexray

Agenda

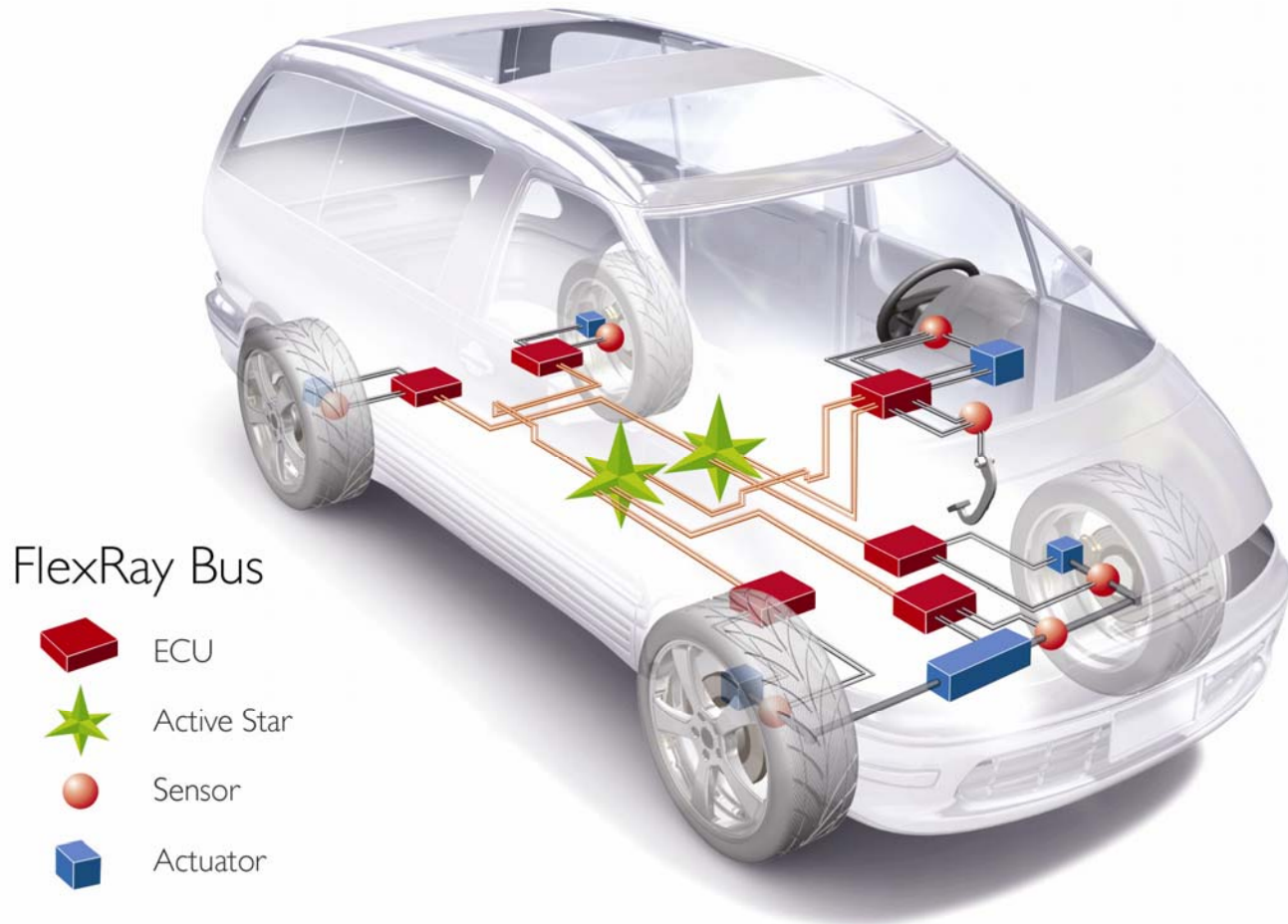
- ▶ Our products (DUT)
- ▶ Second time right strategy
- ▶ Design Trends and Verification Scope
- ▶ Architecture
- ▶ Used methods
 - Self Checking (vmonitor)
 - Checkers
 - Modelling
 - Coverage
- ▶ Summary

Our products (DUT)

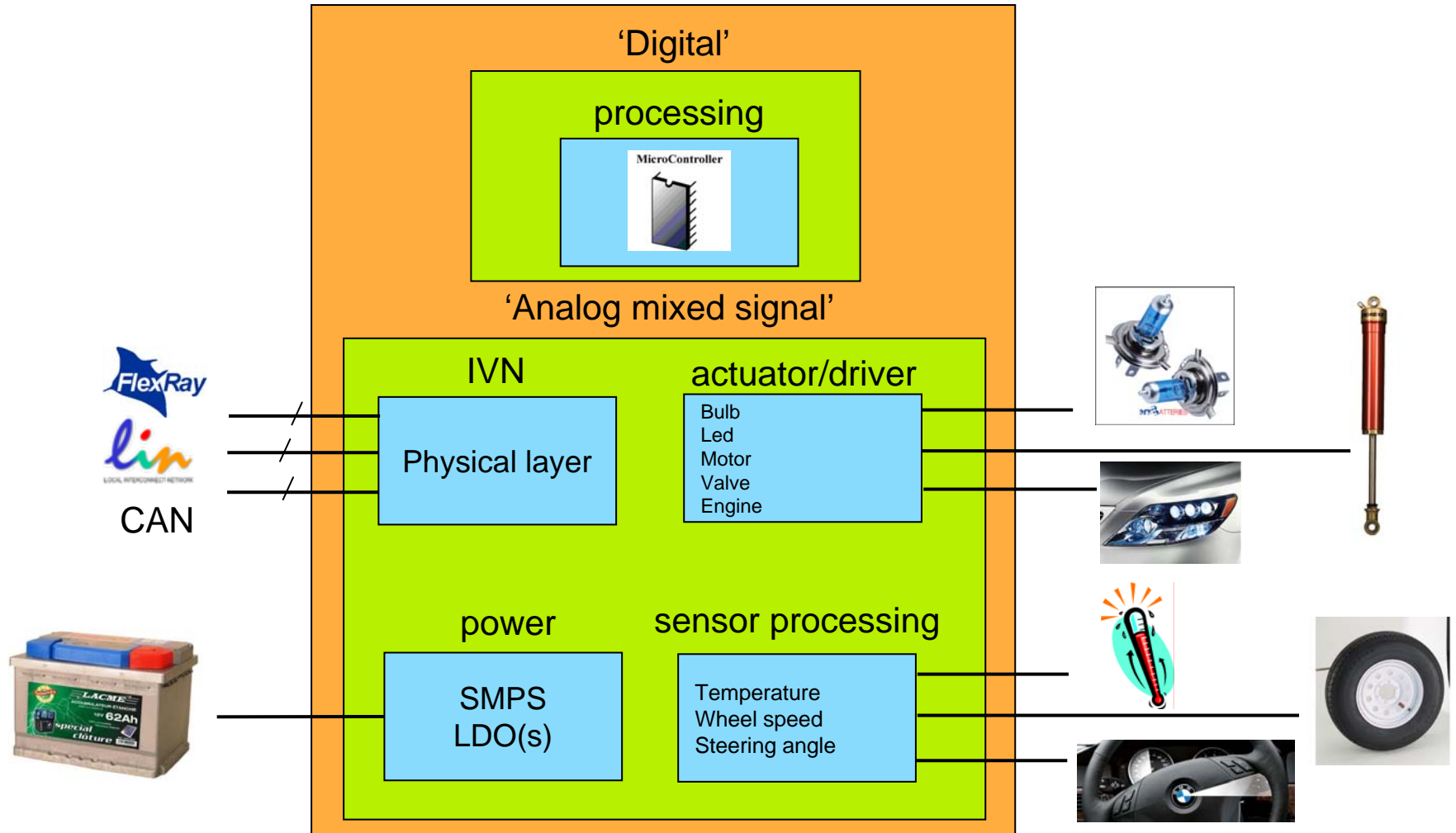
- Automotive Power
- In Vehicle Networking
 - LIN, CAN, FlexRay
- Sensors
- Tire Pressure Monitoring
- Car Access & Immobilizers
- Telematics (Road pricing & E-Call)



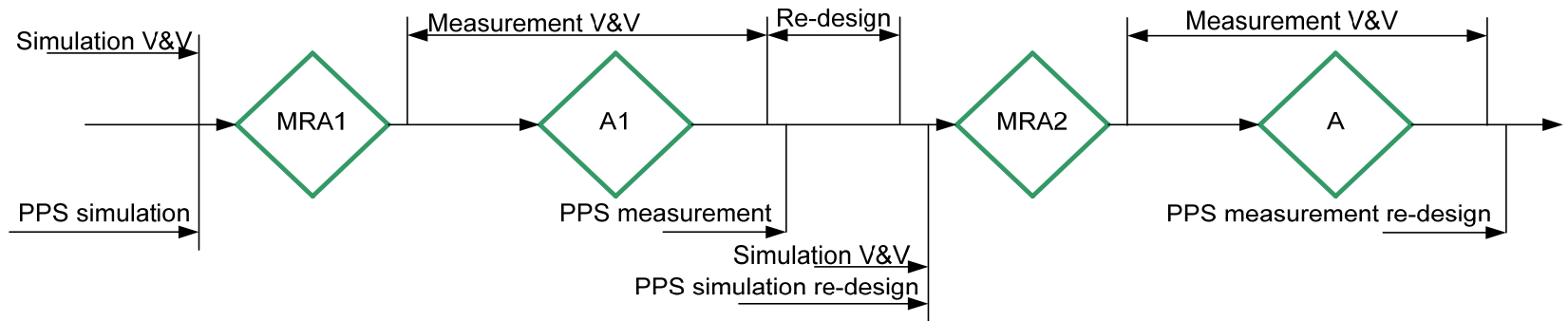
FlexRay Bus Topology



Electronic Control Unit (ECU)



Second time right strategy



MRA == Tape-out

▶ Three instruments:

- Product Potential Study (IP level, temperature, process)
- **Verification** (simulation based, pre-silicon, top-level)
- Validation (measurements, post-silicon)

▶ Verification goal:

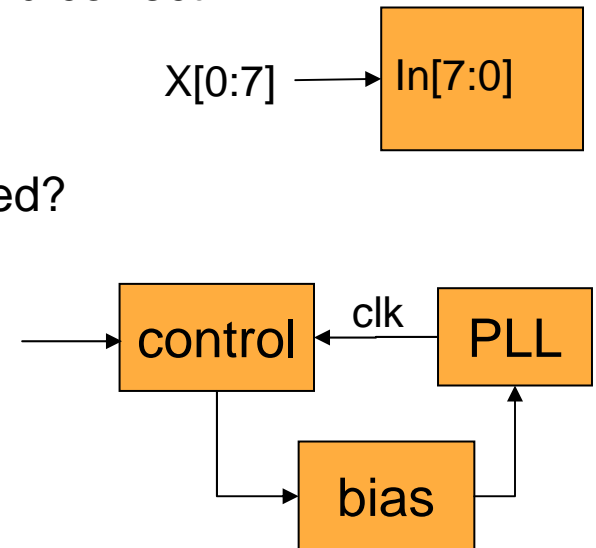
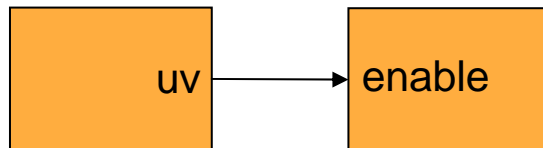
- Design first time functional correct.

Design trends

- ▶ Increasing complexity of our products due to the following trends:
 - Power modes.
 - Green products:
 - E.g. normal, standby and sleep mode.
 - Digital trimming.
 - Adjust analog design after production.
 - Auto calibration.
 - Measure and program required trim values during test.
 - Integration.
 - More IPs on a chip:
 - E.g. LIN transceiver + voltage regulator
- ▶ Top-level verification saves time and money:
 - By finding bugs early in the design phase.
 - By avoiding re-spins.

Top-level Verification Scope

- ▶ Not a top-level product potential study (PPS).
 - Using only typical temperature and process parameters.
- ▶ Assumes IP is functionally correct.
 - No exhaustive verification of IP functions.
- ▶ Interconnect
 - Are all required top-level connection available and correct?
 - E.g. Swapped busses.
- ▶ Interoperability
 - Is the functionality of the connections as expected?
 - E.g. Polarity or Chicken & Egg issues

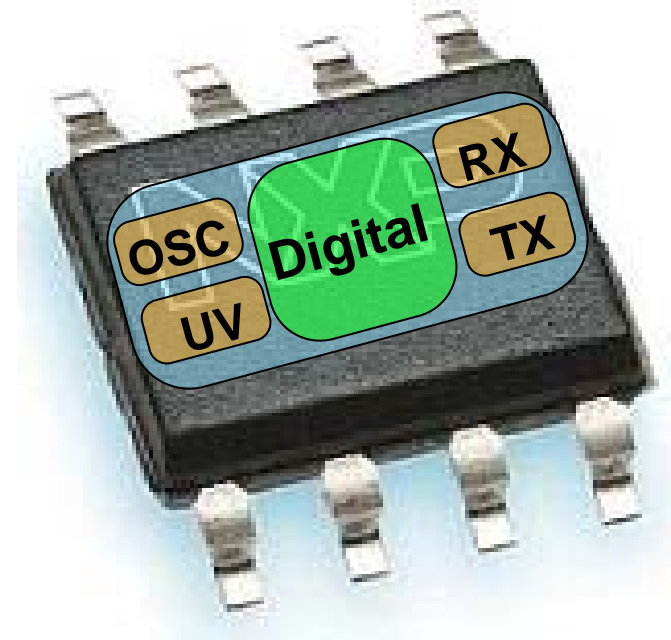


Agenda

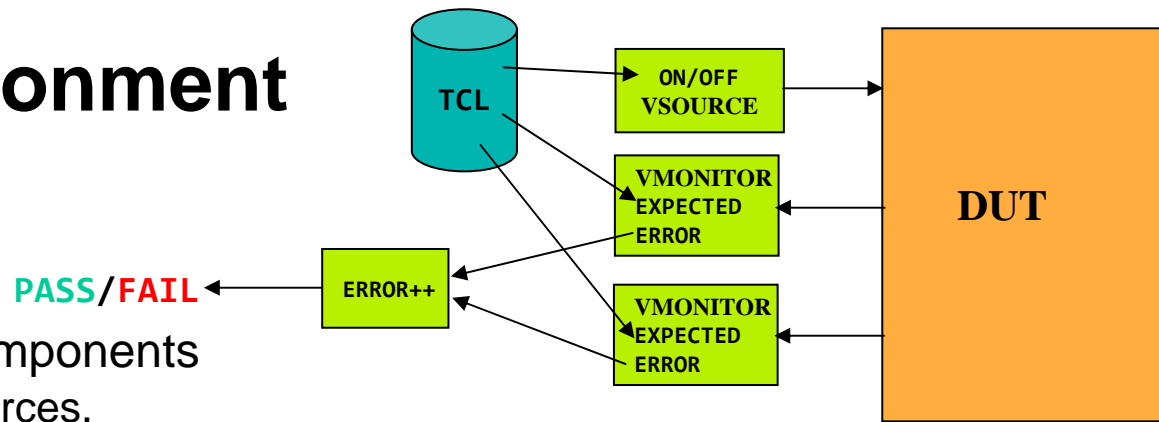
- ▶ Our products (DUT)
- ▶ Second time right strategy
- ▶ Design Trends and Verification Scope
- ▶ **Architecture**
- ▶ Used methods
 - Self Checking (vmonitor)
 - Checkers
 - Modelling
 - Coverage
- ▶ Summary

Design Architecture

- ▶ Analog IP is mainly connected via digital core.
 - Very few connection between analog blocks directly.
- ▶ Almost all chip pins have a “digital” behavior:
 - Transition between two valid steady states.
- ▶ Enables ‘digital’ verification techniques



Test-bench environment

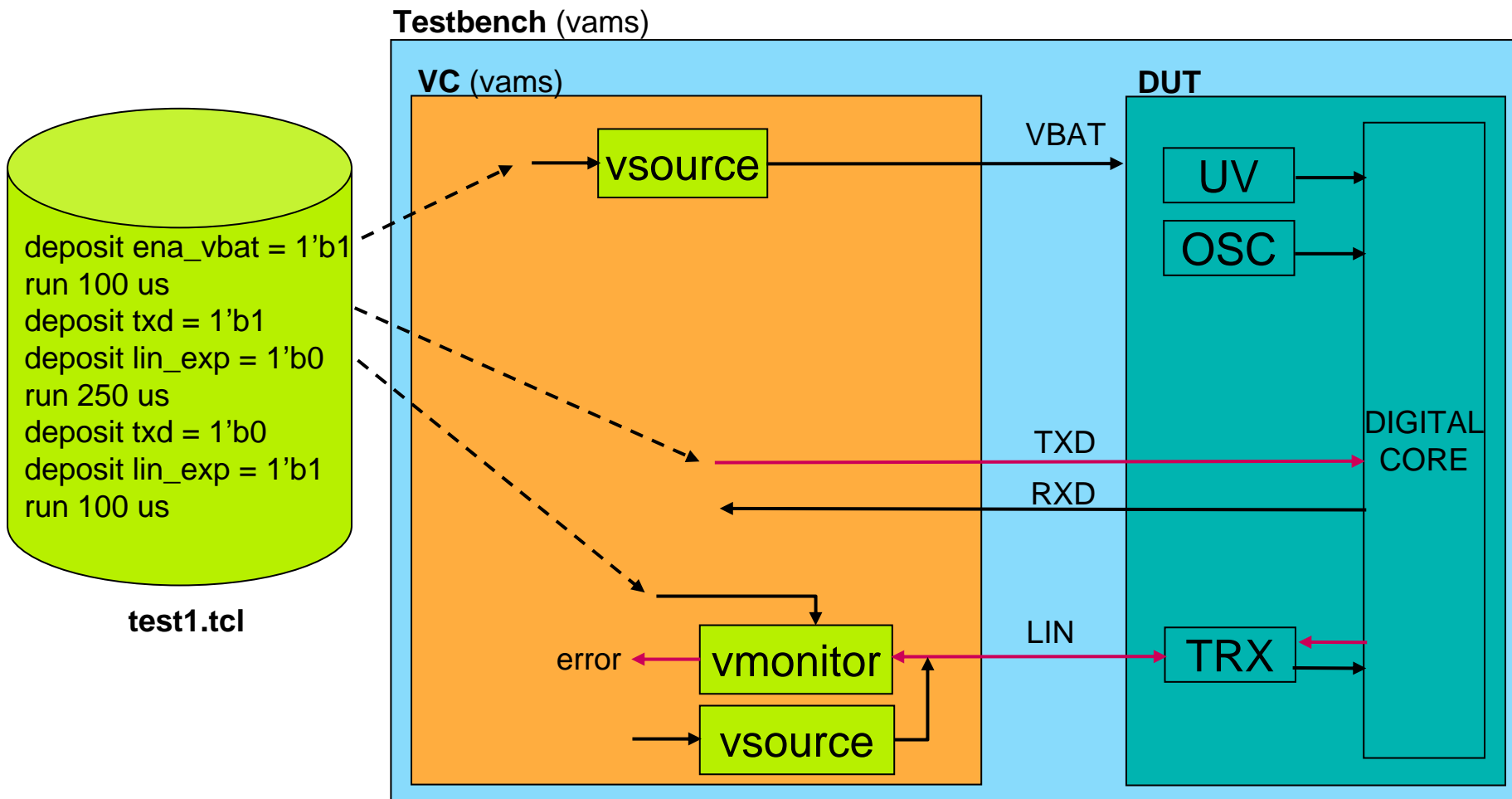


- ▶ TCL input file:
 - Controls test-bench components
 - Enabling voltage sources.
 - Generating expected values for checkers.
- ▶ Central error signal to indicate passing/failing test-cases.
- ▶ Scripts:
 - Run a single test-case on the DUT with a selectable configuration
 - Regression to simulate all test-cases efficiently.
- ▶ Simulations:
 - Spectre (analog)
 - Nc-sim (digital)

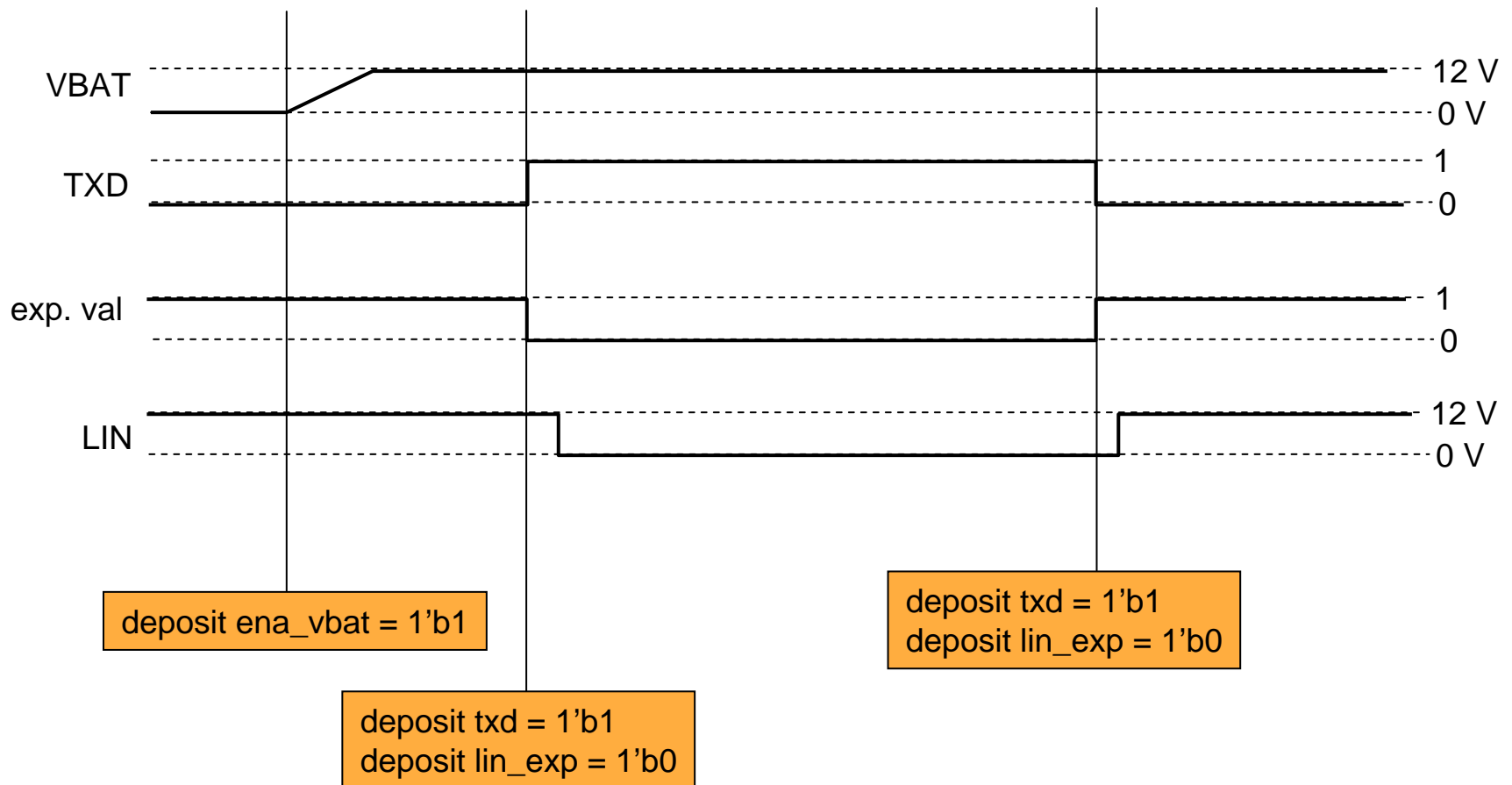
Test-bench architecture

- ▶ Verilog-AMS:
 - TB and verification components:
 - Manually (ASCII editor).
 - DUT and sub-components
 - Generated from the schematic by virtuoso (verilog-AMS netlist).
- ▶ Verification components for driving and checking signals
 - Verification library.
 - Digitally controlled:
 - Vsource (on/off, levels, rise time, fall time)
 - Vmonitor (enable, expected value, levels)

Test-bench architecture



Resulting Waveforms

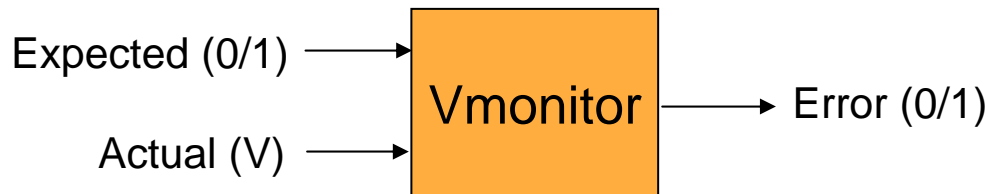


Agenda

- ▶ Our products (DUT)
- ▶ Second time right strategy
- ▶ Design Trends and Verification Scope
- ▶ Architecture
- ▶ **Used methods**
 - Self Checking (vmonitor)
 - Checkers
 - Modelling
 - Coverage
- ▶ Summary

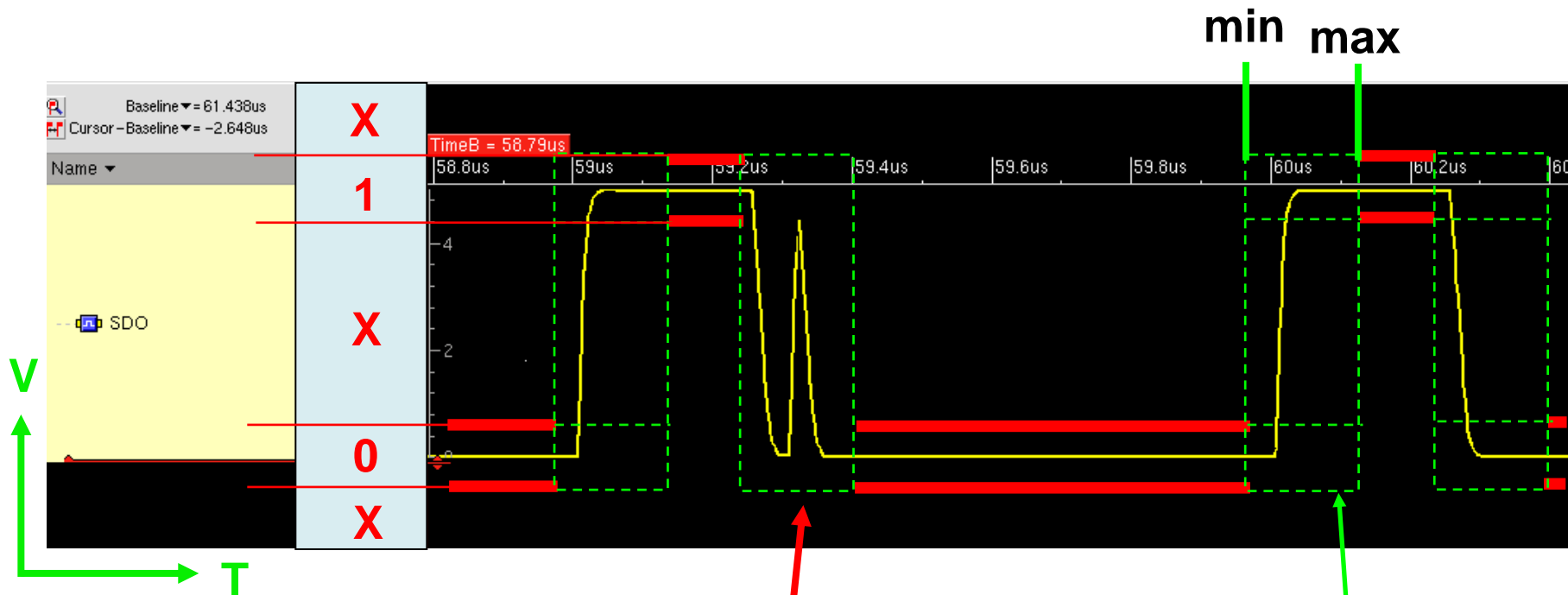
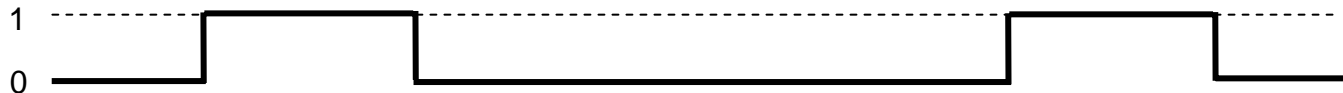
Self Checking

- ▶ Required for:
 - handling a large set of test-cases or comparing tape-outs:
 - Pass/Fail detection without eye-balling waveforms.
 - Continuous checking
 - No sampling
- ▶ Vmonitor:
 - Comparing expected value with actual value.
 - Can be applied for analog signal with “digital” behavior.
 - Two steady states.



Vmonitor

Expected value



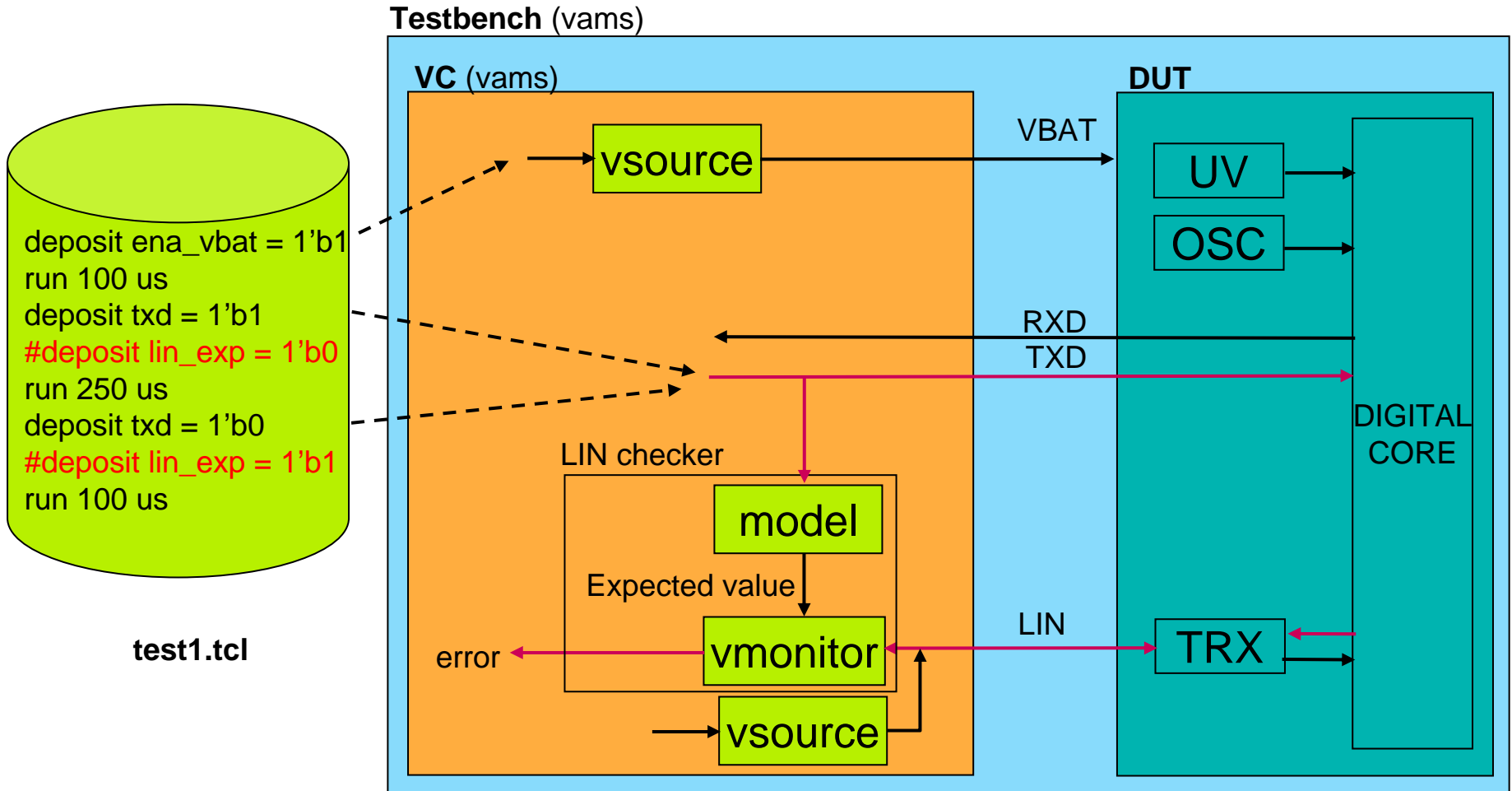
Monitor detects failure.

Transition box

Checkers

- ▶ Splitting stimuli and expected value generation.
 - Enables re-usability.
 - Better maintainable:
 - No need to change expected value when changing stimuli.
- ▶ Models are used to generate the expected values.
 - By using the input stimuli.
 - Enables random stimuli generation.
- ▶ Checker
 - Combination of a model(s) and vmonitor(s).
- ▶ Checkers can be reused.
 - Within the project:
 - Checkers enabled during all use cases
 - Across projects:
 - Checkers are often IP specific and can be reused when IP is reused.

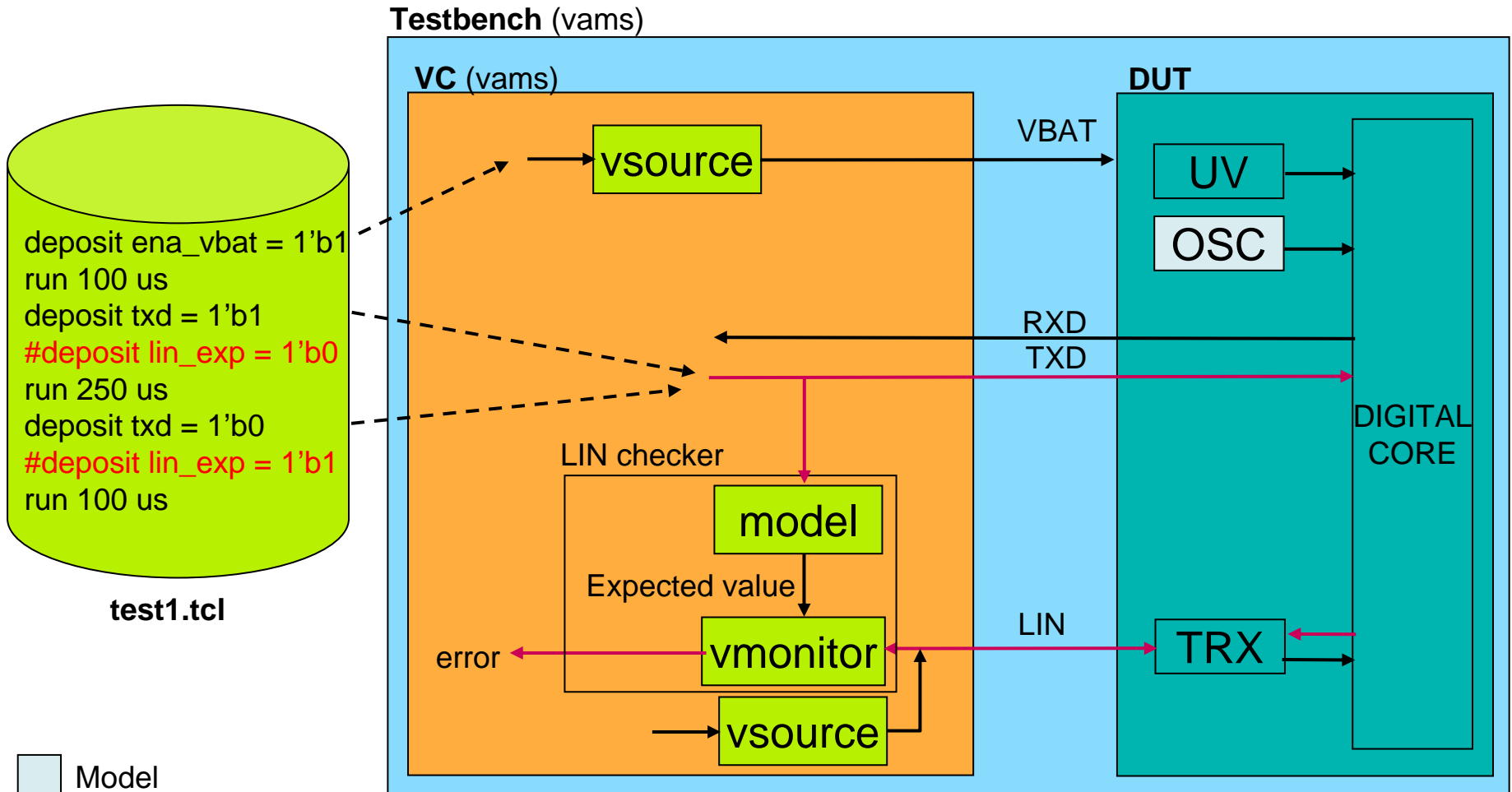
Checkers



Modeling

- ▶ Use verilog-ams models to speed-up simulation.
- ▶ Possible for IP:
 - that is already verified in other test-cases (using the analog design).
 - that is not used in the current test-case.
- ▶ Typical example of IP that are candidate to be replaced:
 - Oscillators
- ▶ Selection between schematic or model is stored in a configuration file.

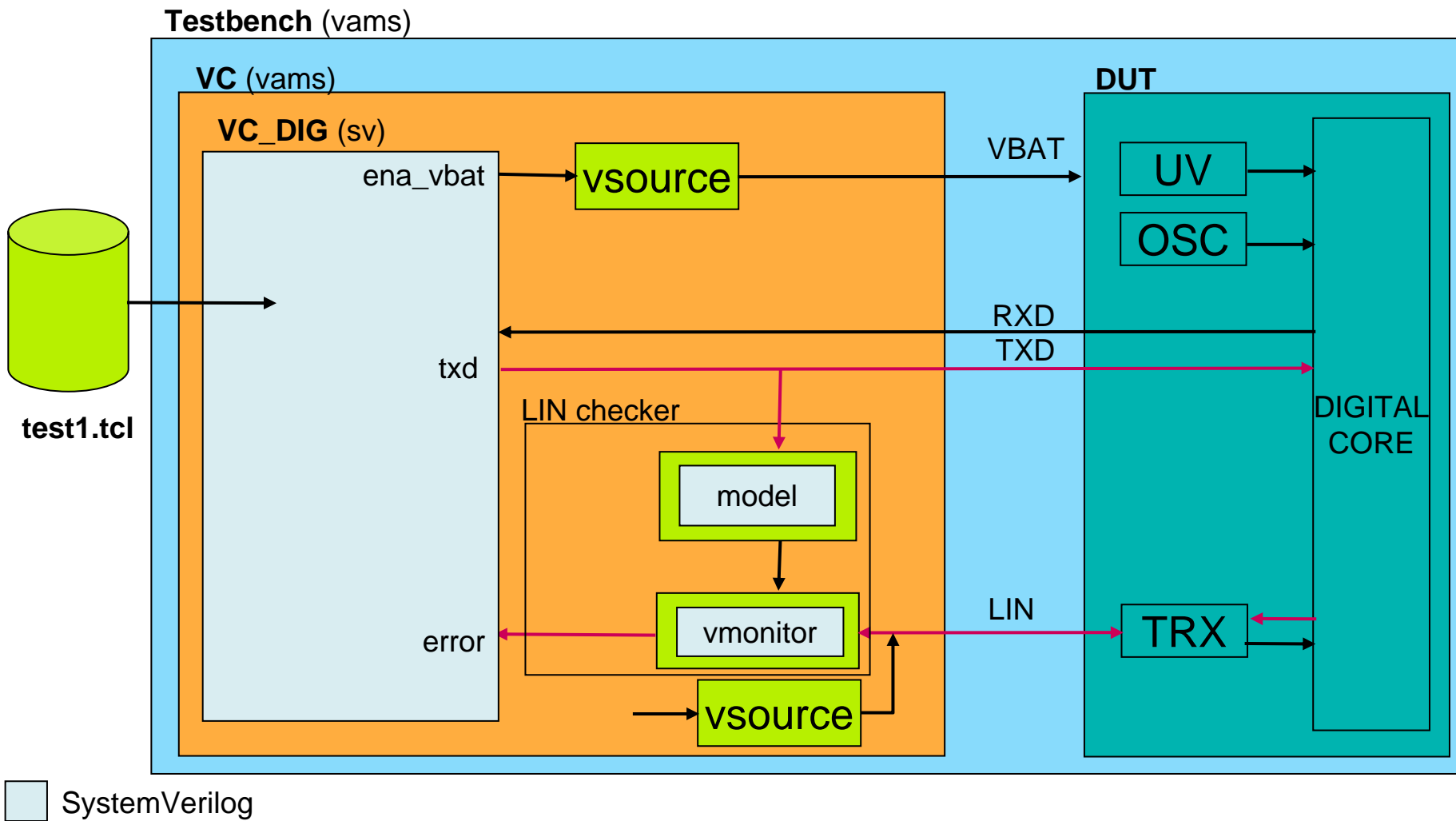
Modeling



Coverage

- ▶ Used to give a better insight in the verification quality.
- ▶ Implemented using SystemVerilog
 - Merge of Verilog/Verilog-AMS/SystemVerilog planned.
- ▶ Usage:
 - Verification Component (vc_dig)
 - Cover the executed test-cases and the results (pass/fail).
 - Checker:
 - Cover the state in which the design has been during simulation.
 - Vmonitor:
 - How often a value has been checked.
 - The expected value that to be checked.
 - The delay between expected value an actual value.

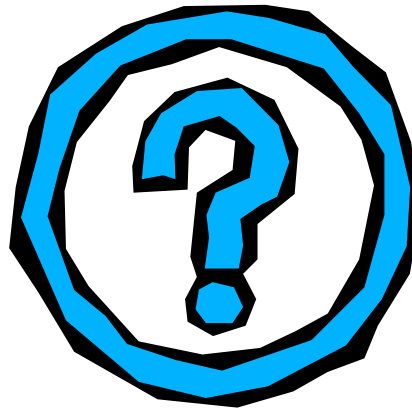
Coverage



Summary

- ▶ Initial projects:
 - Interconnect and Interoperability.
 - Directed Test Cases.
 - Self Checking.
- ▶ Enhanced:
 - Split Stimuli and checkers:
 - Enables constraint random stimuli.
 - Start using checkers.
 - Expected value retrieved from model.
 - Modeling.
 - Coverage.

Q & A



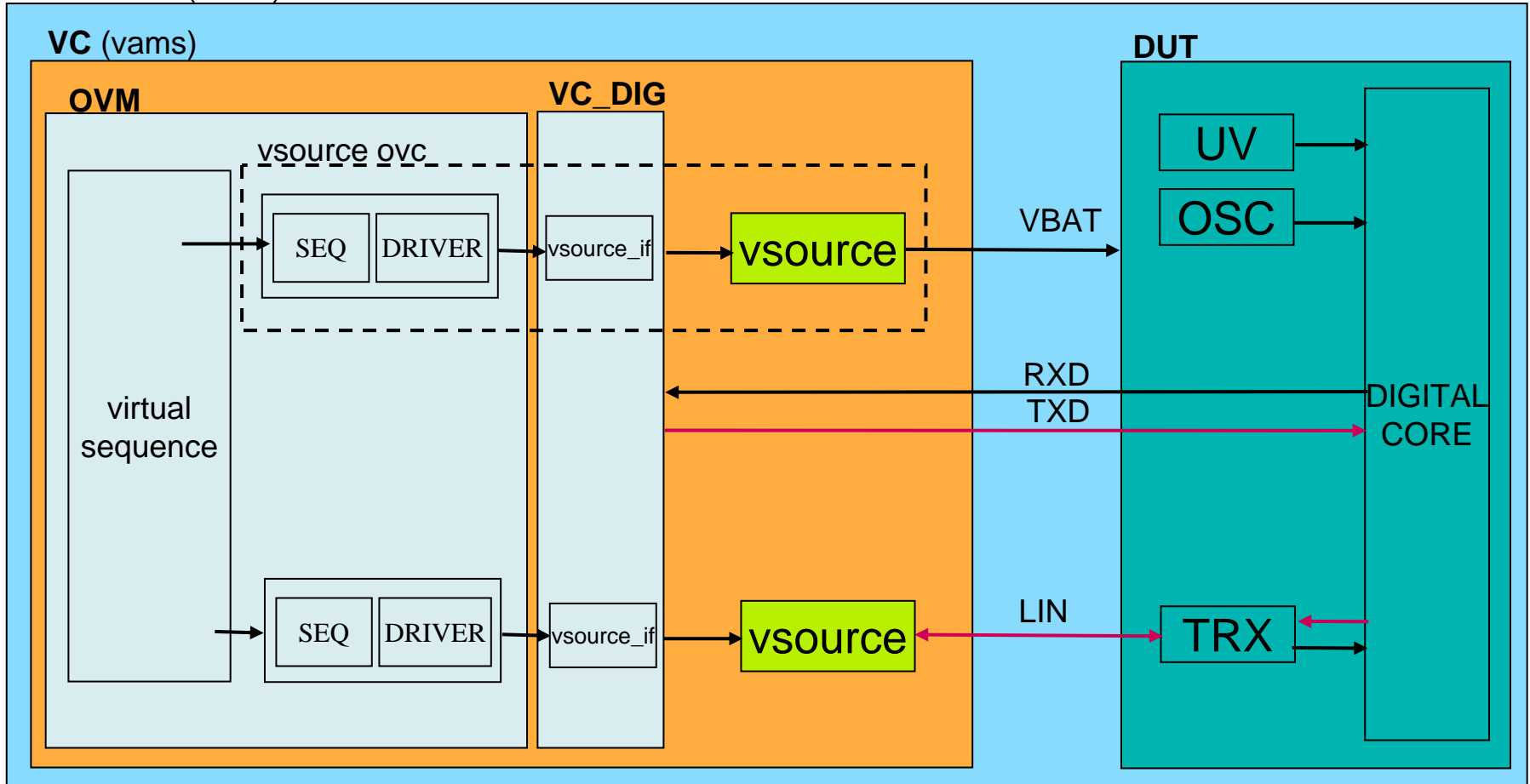
Open Verification Methodology

- ▶ Natural next step
 - Split between stimuli and checks
 - Enables constraint random stimuli → OVM Sequences
 - Usage of checkers/scoreboards → OVM Scoreboards
 - TLM makes connections between modules easier
 - Analysis ports
 - Generic report mechanism

- ▶ Standard methodology
 - Guidelines for creating OVC, scoreboards and generating stimuli.

OVM example

Testbench (vams)



Verification Components

▶ VSOURCE:

- Sequence item (transfer):
 - real vv; // Output voltage value.
 - real tr; // Rise/fall time.
 - real td; // Delay.
 - real r; // Value of the internal series resistor.
 - logic din; // Switch-on of voltage source.
 - logic connect; // Disconnect from analog network.
 - logic mode; // Use vv or input voltage source for output voltage.
- D2A done using an existing component (vsource).

▶ SPI:

- Sequence item (transfer):
 - bit [4:0] len; // Number of bits.
 - bit [31:0] data;
- D2A/A2D done via connectrules.

▶ FLEXRAY:

- Sequence Item (transfer):
 - abln_flr_state_t state; // lowpower, idle, data0, data1, disabled.
 - int duration;
- D2A/A2D done using an existing component (flr_transceiver).

Virtual Sequences

- ▶ Able to create a use-case from a central point.
- ▶ Use Case 1:
 - Ramp-up VBAT from 0V to 12 V.
 - Program device via SPI.
 - Transfer data on one flexray port.
- ▶ Use Case 2:
 - Ramp-up VBAT from 0V to 12 V.
 - Program device via SPI.
 - Transfer data on two flexray ports at almost the same time.
 - Use case for the arbiter.
 - Concurrent stimuli generation.