Verification

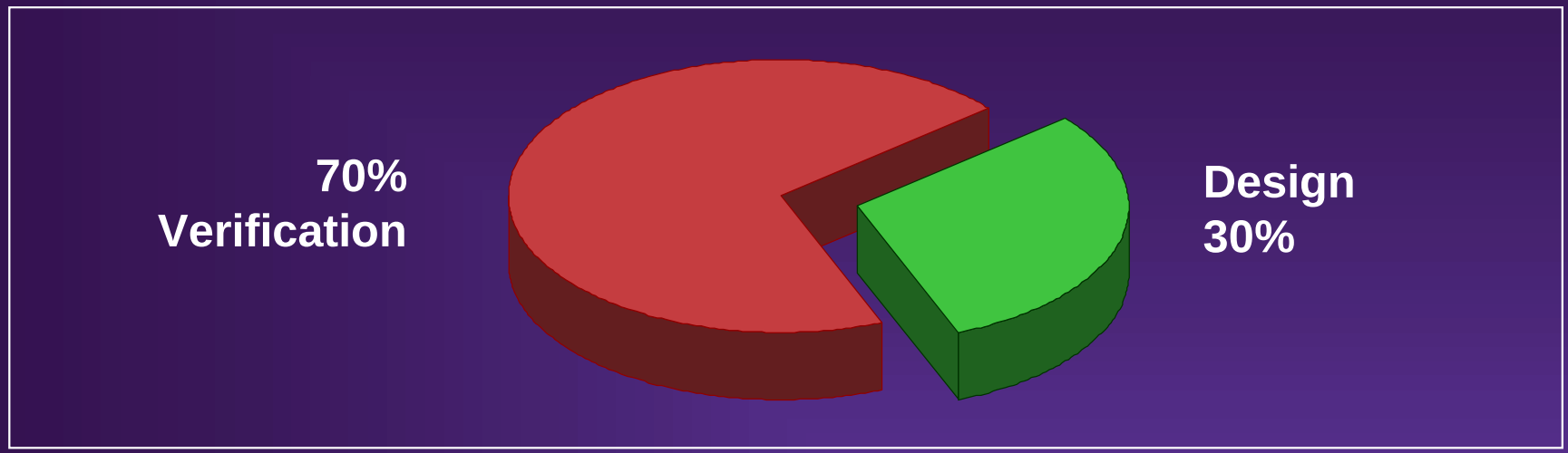
A gap-free Verification Flow from Transactions to RTL using SystemC and SystemVerilog

Holger Keding, Synopsys, Aachen
Thorsten Groetker, Synopsys, Aachen
Alok Kuchlos, Synopsys, Mountain View
Markus Wloka, Synopsys, Aachen

Agenda

- Concept for a Verification Flow, from RTL to Transaction Level
- Concept Example: Simple_Bus (again!)
- Your final takeaway

Functional Bugs Break Schedules…

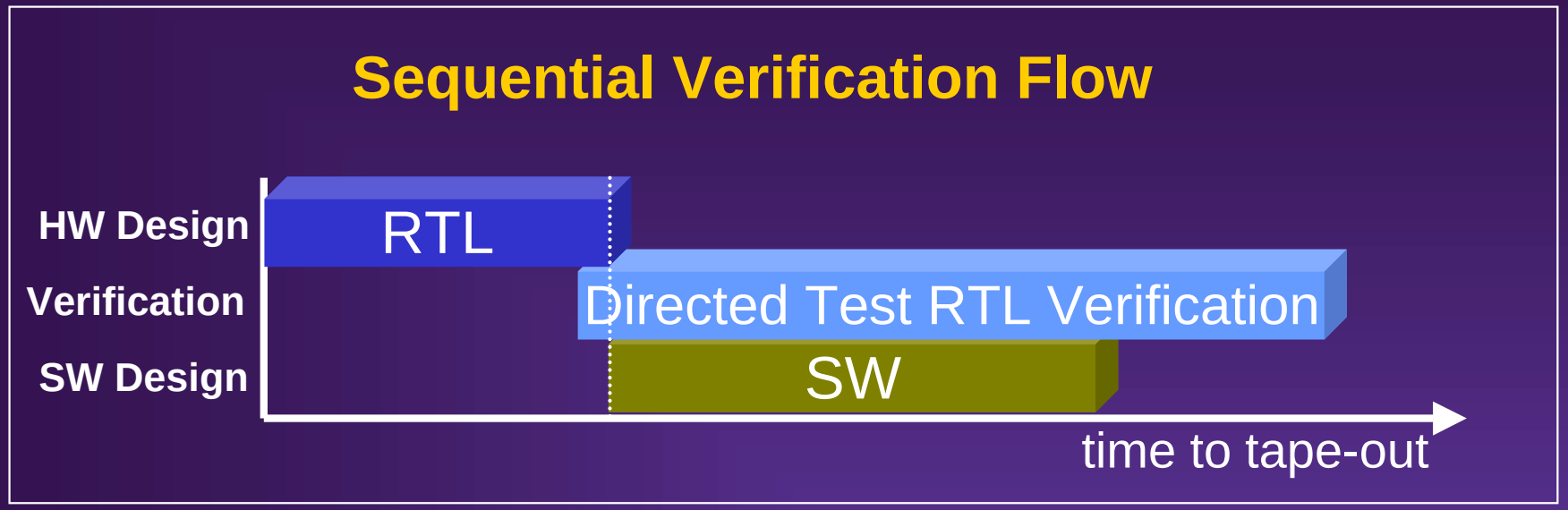


- Over 60% of design re-spins are caused by functional errors!

  (Ref. 2001 Collett International Research)

- Re-spin NREs at 0.13 µ costs >$500K

Sequential Verification delays schedule



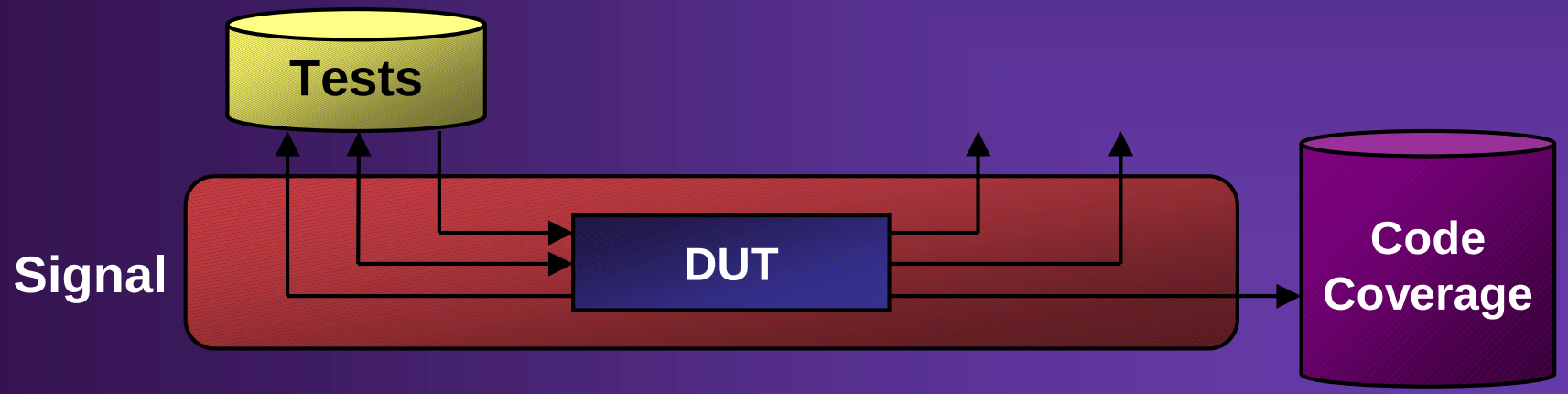
Delay happens because

- directed test verification are time consuming and inefficient
- verification starts at the end of the RTL design phase
- SW development & verification starts after RTL completion

Signal-Level Verification in RTL

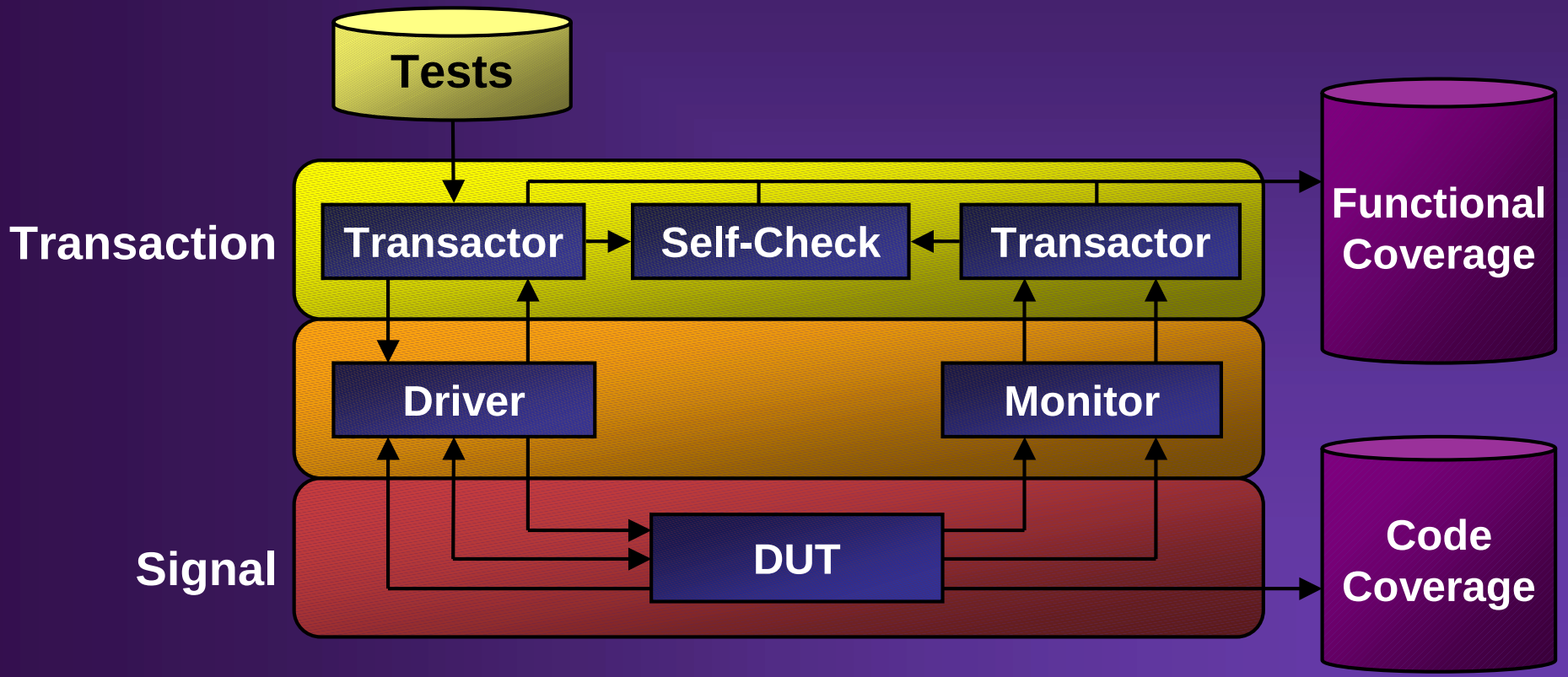
Verification of DUT at RTL implementation is mandatory part of flow

- Wide range of verification methods are available:
  - Constrained random stimulus generation
  - Assertions
  - Coverage
- Signal-level stimulus and Testbench



SystemVerilog Transaction-Level Testbench

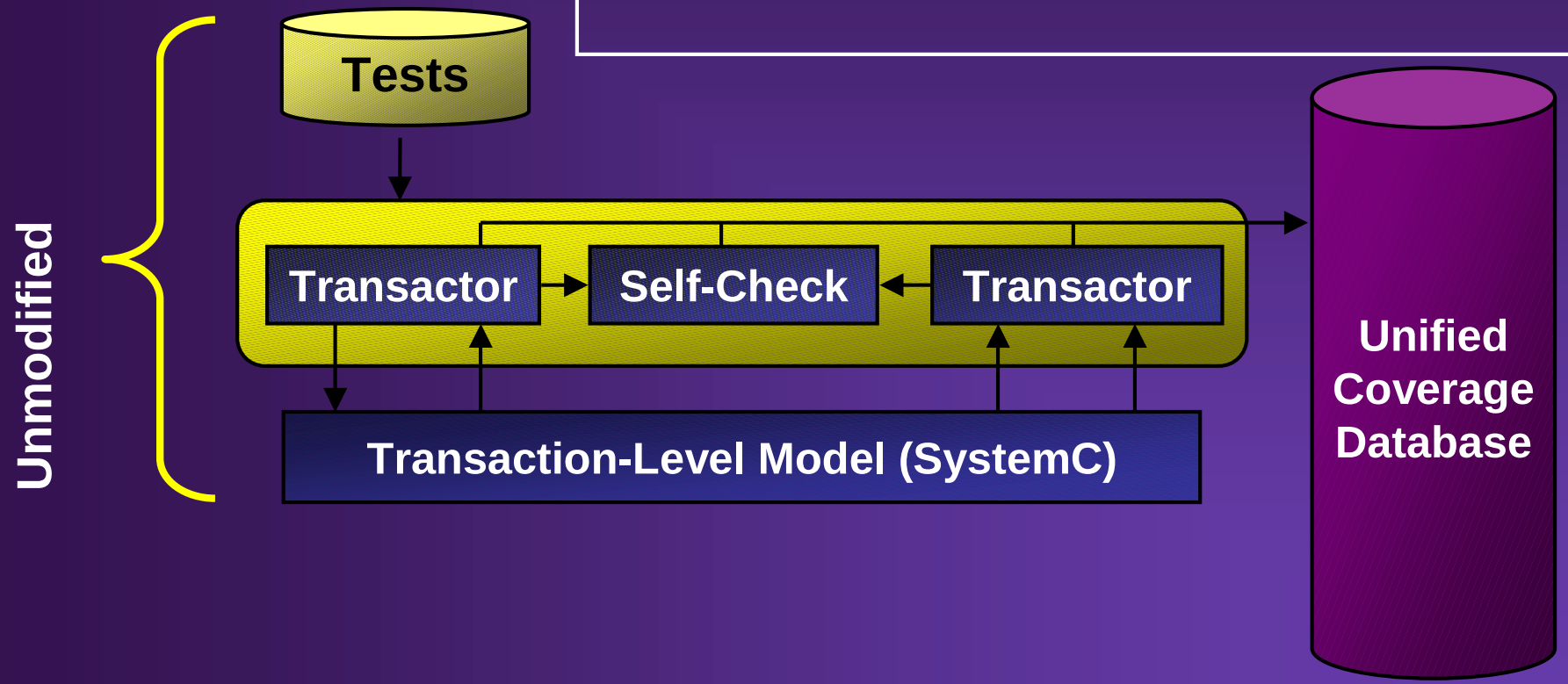
Increase Testbench abstraction level
- Reduced coding effort
- Self-checking
- Functional coverage



Increase DUT Abstraction Level: System TLM

Develop, refine testbench against fast TL model
- Improved simulation efficiency
- Create reusable testbench

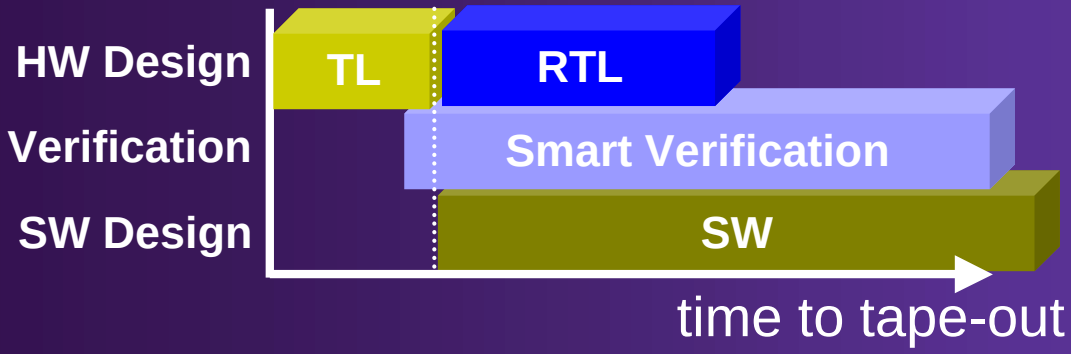
Reuse models from architectural exploration

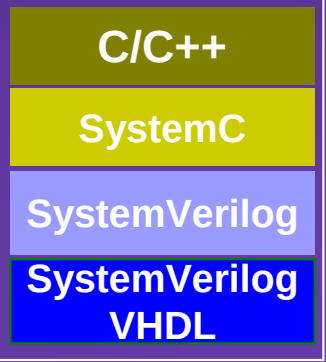


SystemVerilog TL testbench verifying SystemC TL model

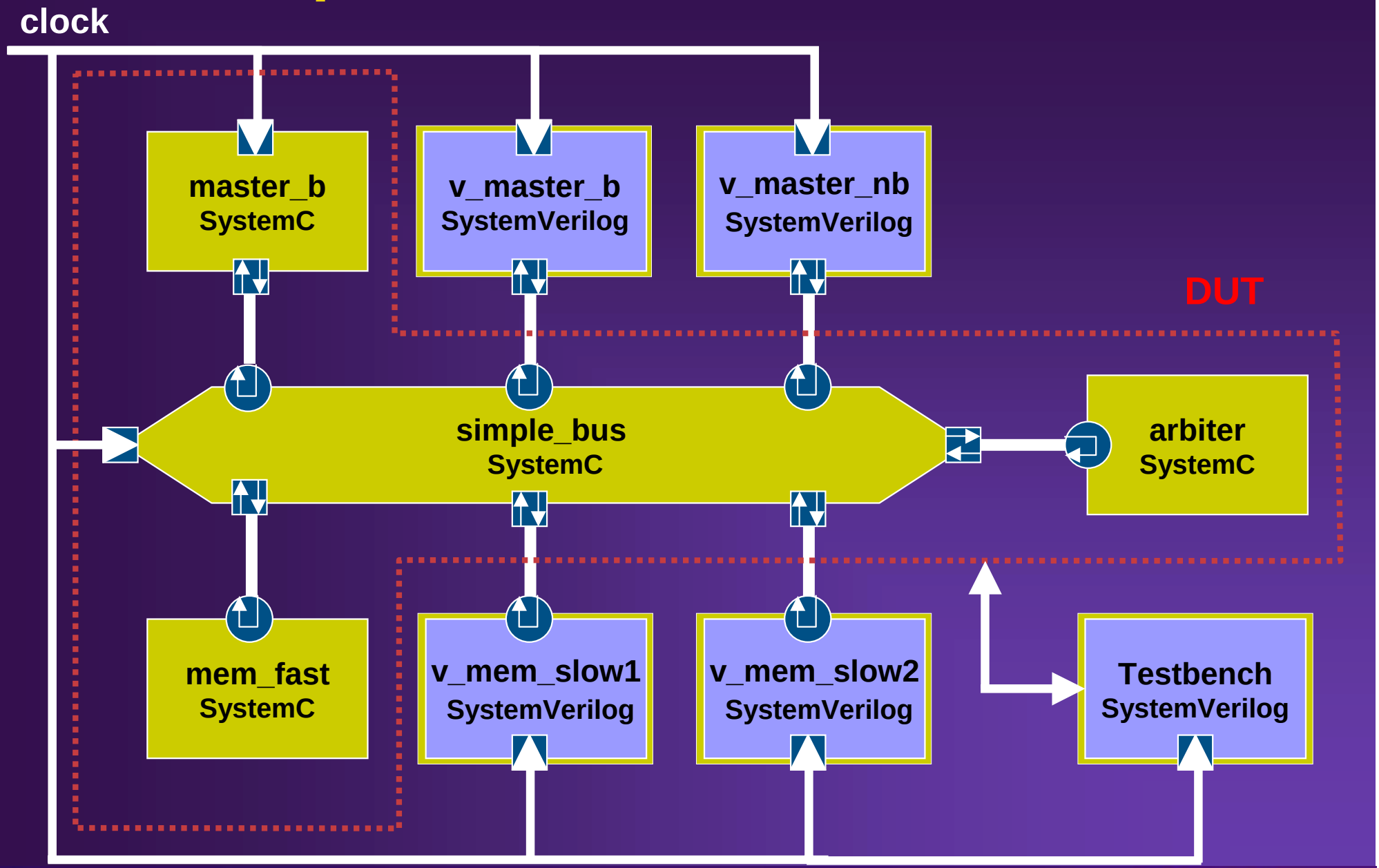
- smart verification using assertions, functional coverage, and powerful constraint random generators
- start verification already at transaction level
- start SW development & verification also at TL
- reuse testbench from TL at RTL

Smart Verification Flow





Example: Verification of a TL DUT



SystemVerilog Coverage Driven Verification for SystemC TL Example

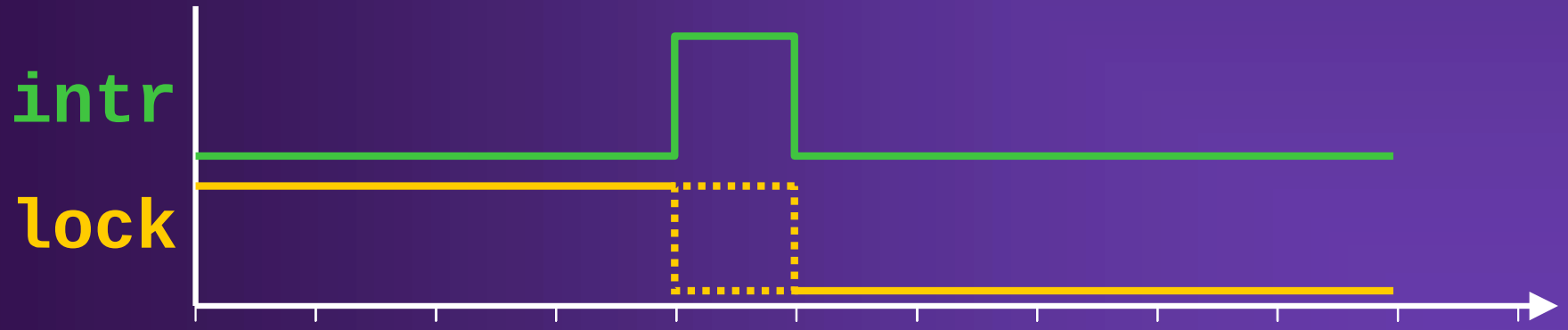
- Constraint Random Data Generation
  - read and write delays
  - access address
  - burst size
  - bus locking (on/off)
  - payload data to be written
- Check for states + events to be covered
  - Transactions
  - States
  - Transitions
  - Memory accesses

Temporal Assertions in SystemVerilog Checking SystemC Behavior

- Check for *bad transfer interrupt*

```
assert bad_transfer_interrupt: forbid
  (posedge intr) #[0..1] (negedge lock)
```

intr: interrupt an uncompleted transfer
lock: bus lock set for current transfer



Summary

- SystemVerilog enhances verification flows for designs modeled with TL SystemC
- Verification flow covers multiple levels of abstraction, from TLM to RTL (Verilog, VHDL)
- TL modeling capabilities of SystemVerilog and SystemC shorten the verification cycle

Our final comment

If you use this

your colleagues should use this