中核控制系统工程有限公司
CHINA NUCLEAR CONTROL SYSTEM ENGINEERING CO., LTD.

CNCS

Test technology of FPGA-based safety I&C system of nuclear power plant

Xu Zhan

2015.10.13

智慧科技 • 精控未来

Outline

- FPGA test background
- FPGA test process
- FPGA verification methodology
- FPGA test benches
- FPGA test application

FPGA test background

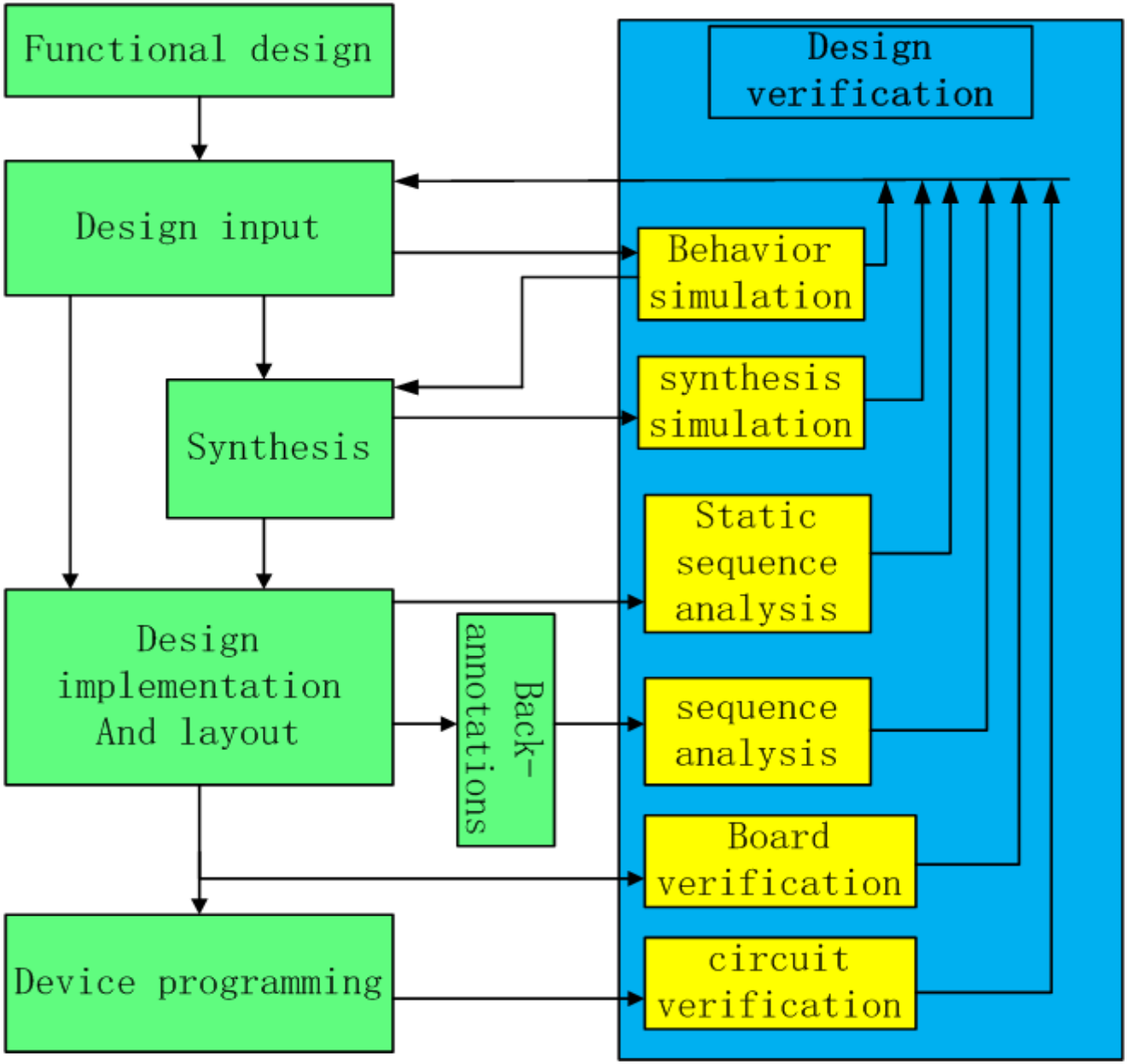
- Application on FPGA-based safety I &C system of NPP

- For FPGA-based I&C systems, the testing activity needed satisfy the verification objectives of IEC62566 or NUREG CR7006 for safety application.

- With the complexity of FPGA design increasing, the FPGA testing confront significant challenges :

- Verification consumes more than 70% of resources
- Time to market affected
- Bugs remain undetected
- Conventional simulation inadequate
- Better approaches needed

FPGA test process



Test approach

Directed test：

-Testbenches without randomness, targeting a specific item in the verification plan.

-If the design is complex enough, it is impossible to cover all features with directed testbenches.

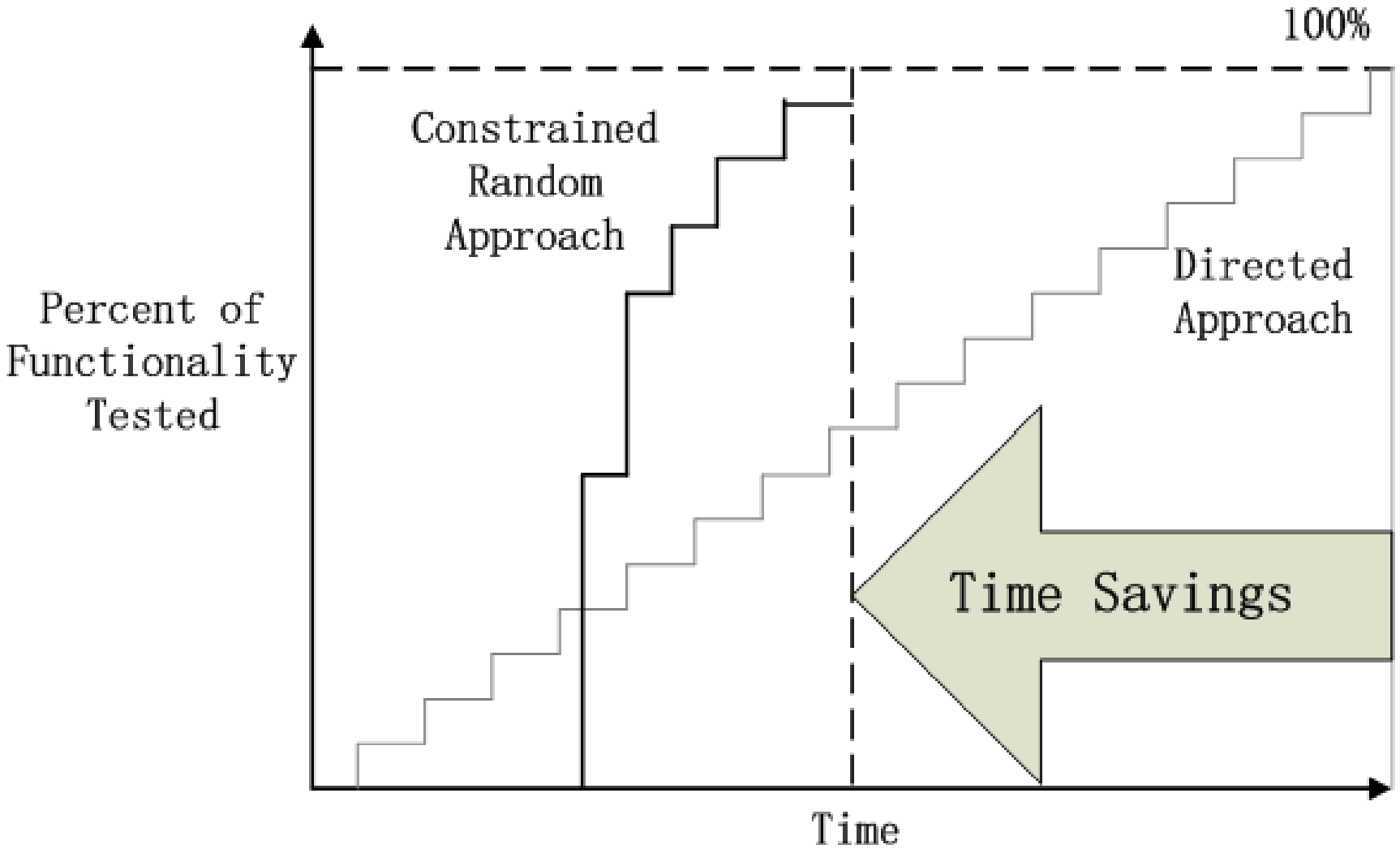
Random verification:

1) Generate random tests using random constrained stimuli generation.

2) Check for bugs and correct them if there are

3) Check for the coverage values. If not satisfying, add constraints and repeat from 1.

Efficient test approach



Verification methodology

What is verification methodology?

The methodology for constructing a software verification platform and providing the reusable verification platform.

Why verification methodology can be used?

1) Not to re-develop new solutions
2) Not to spend time writing new code
3) To increase testing quality, more stable and reliable

Focus on

- Random constrained stimuli generation
- Assertions
- Functional coverage

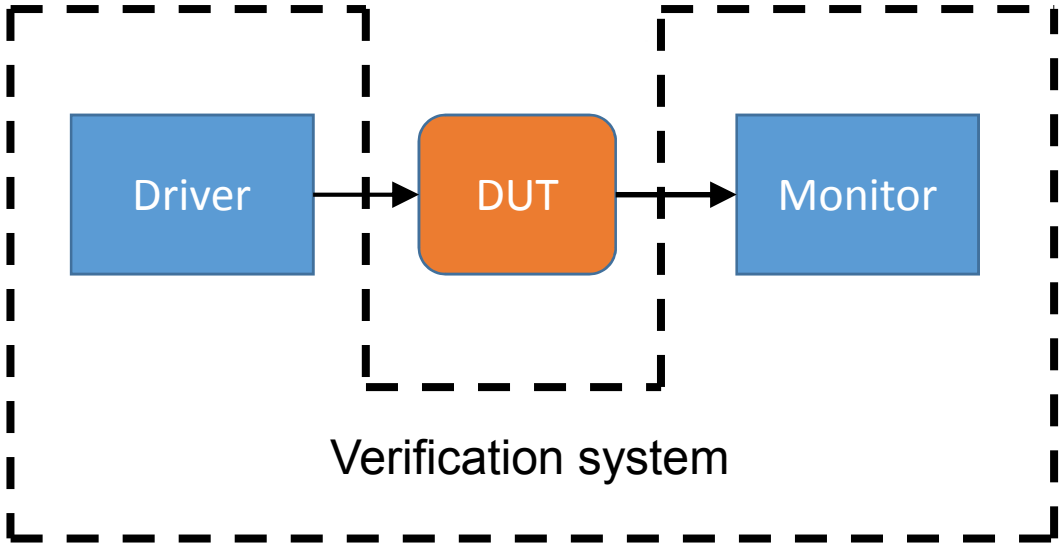
Conventional verification system

DUT: Design Under Test

- Time based simulation
- Cycle based simulation
- Simple design

Defect：

- Not scalability
- Not reusability



Layered verification system

From bottom to top

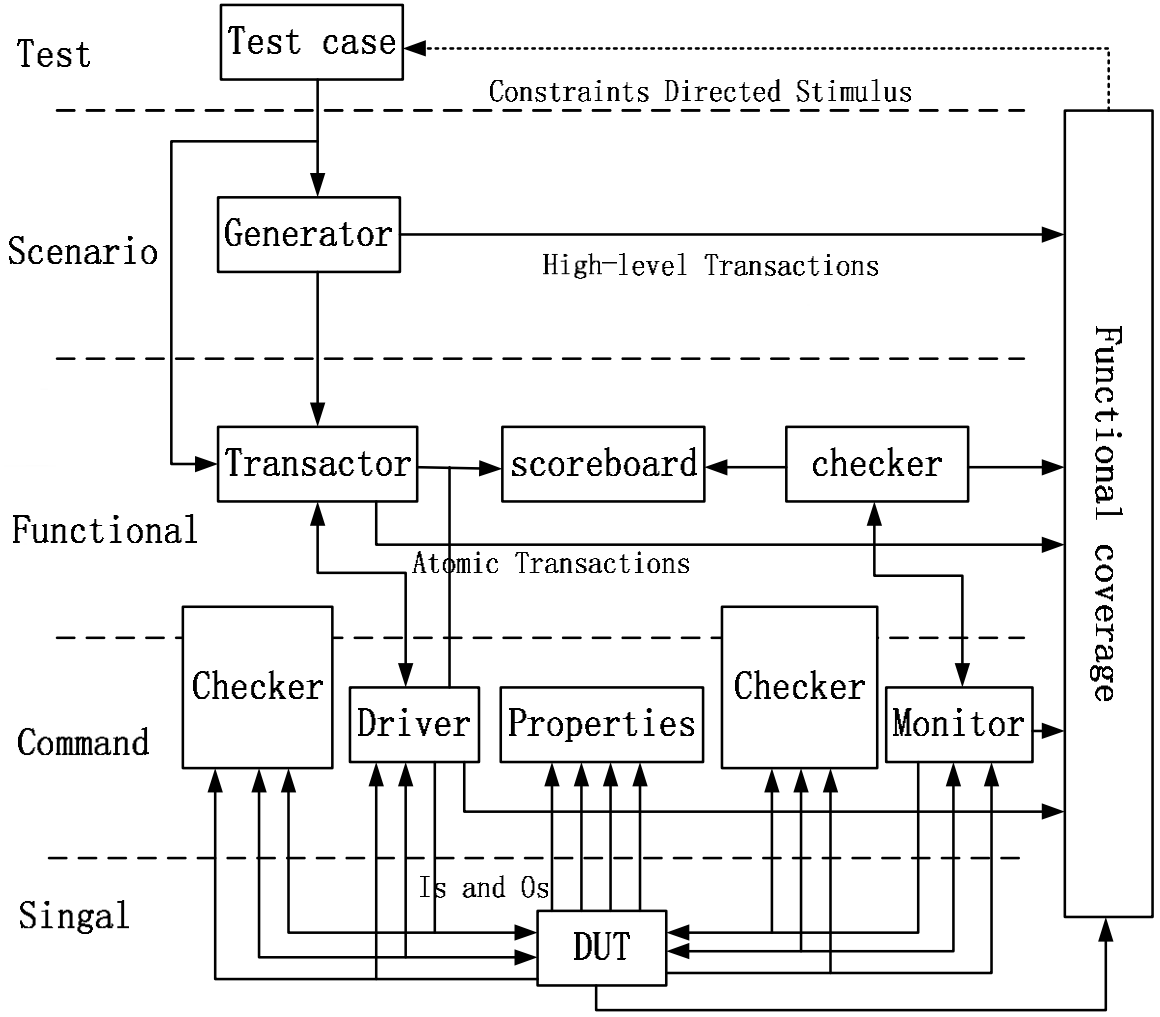
1) Signal layer

2) Command layer

3) Functional layer

4) Scenario layer

5) Test layer



Random constrained stimuli generation

-Define random variables and constrants

-Ask the random solver to find a random set of variables that satisfies the constraints

-Constraints can be added, disabled to create different tests.

-It would be hard to mark a routine to randomly generate one of the legal combinations using only direct randomization of variables

Assertions

- Tools for automatic checking of properties

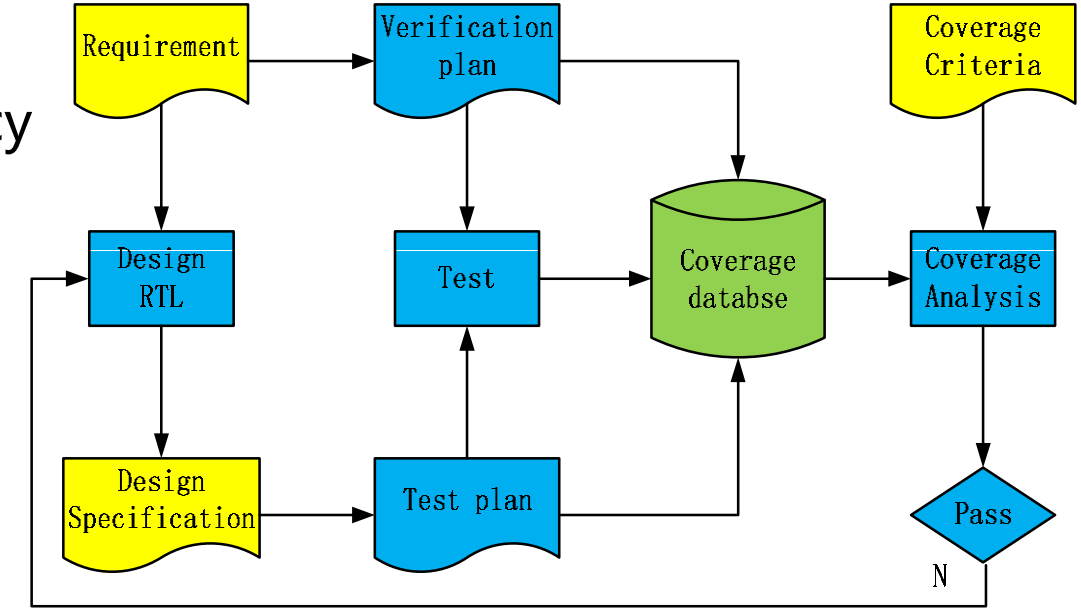
- prove how the design behavior can meet the requirements

- Assertions are instantiated similarly to other design blocks and are active for the entire simulation

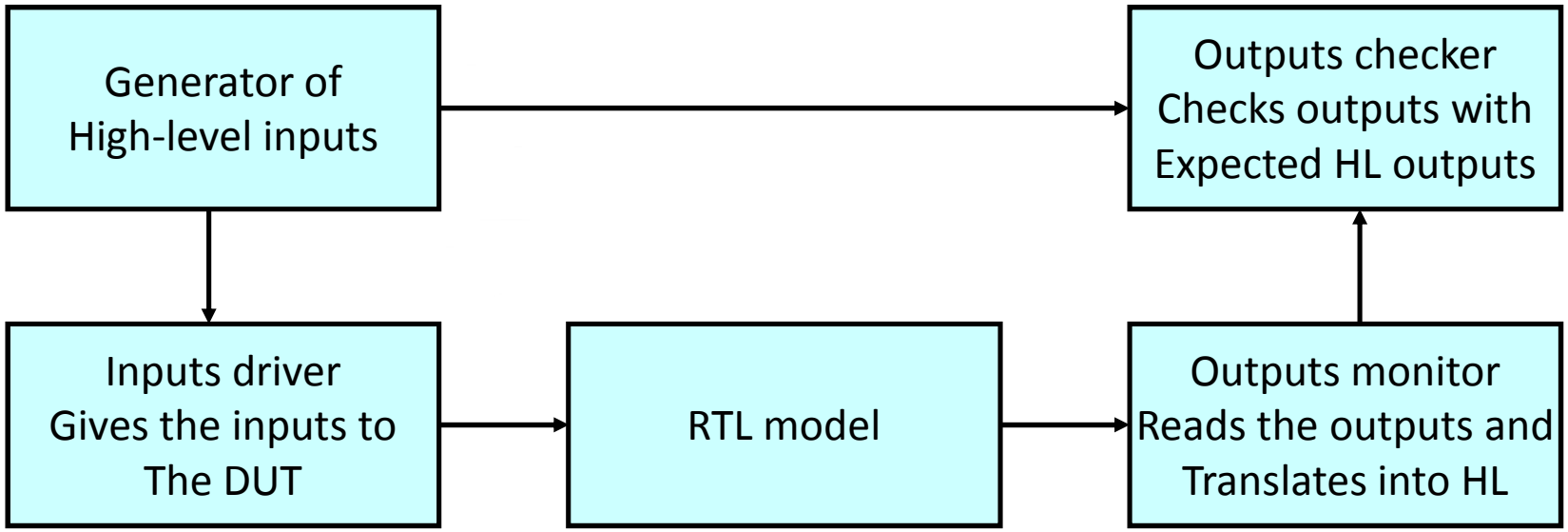
- The simulator keeps track of what assertions have triggered, and so you can gather functional coverage data on them

Test coverage

- RTL code coverage
  - –Statement, decision, condition, path, toggle, triggering, FSM
  - –100% of statement, decision and condition, path
- Quality metrics
  - – Bug detection frequency
  - – Length of simulation
  - – Simulation Minimize
- Functional coverage
- Requirement coverage
- Regression test
  - – Quantitative stopping criterion
  - – Test more but simulate less



Test benches structure



Test benches

It is important to conceptually divide testbenches into blocks, depending on the function:

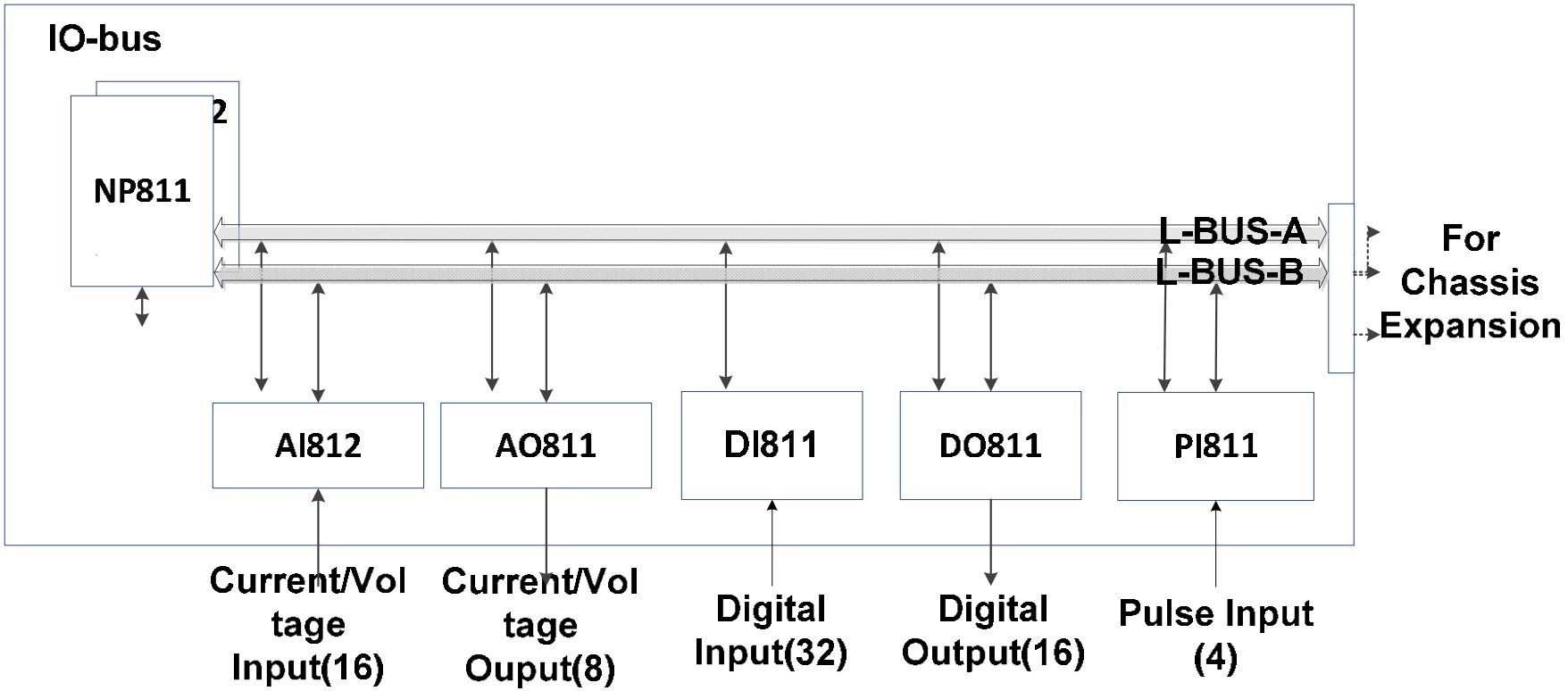
- Generator of high-level input data
- Driver: read the high-level input data and drive the DUT input ports.
- Output monitor: read data from the DUT's output ports.
- Output checking: check correctness of the output

This allows easier readability and reuse. If the DUT input protocol change, only the driver must change.
This requirement is most important for big systems, with a lot of reuse and efficiency.

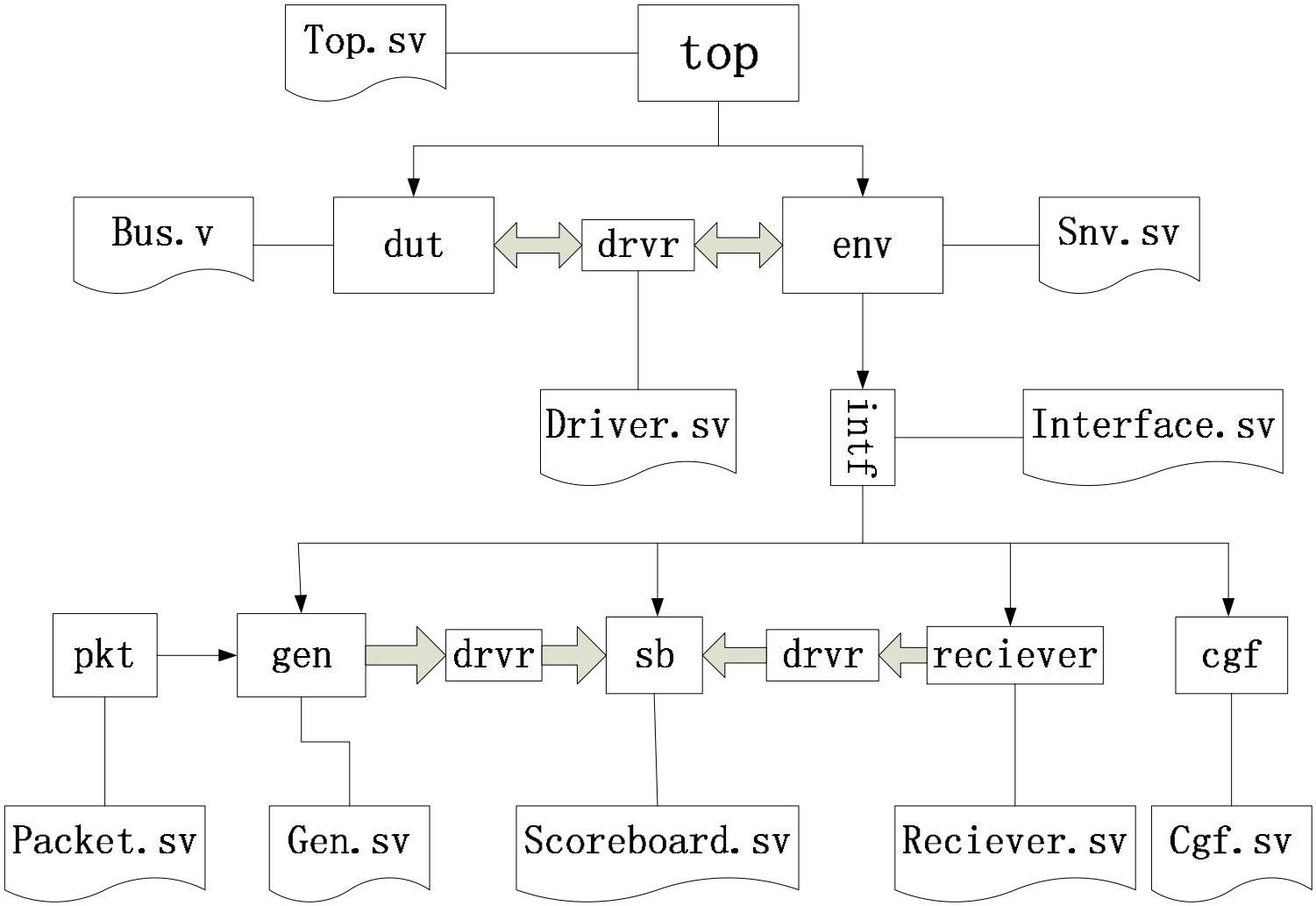
FPGA Test application

Simulation verification tool: Medolsim10.0b

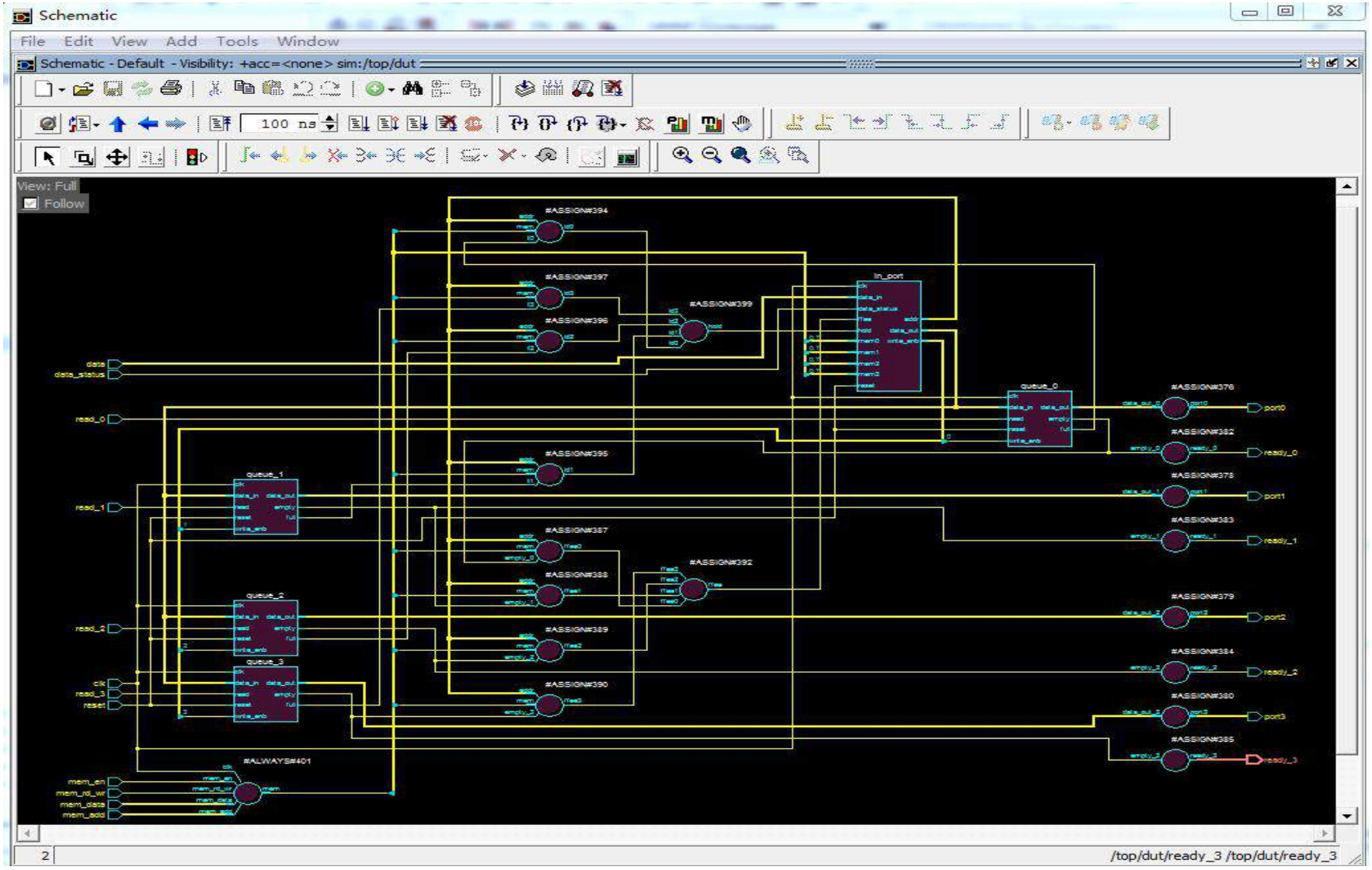
DUT: IO-bus



Verification platform

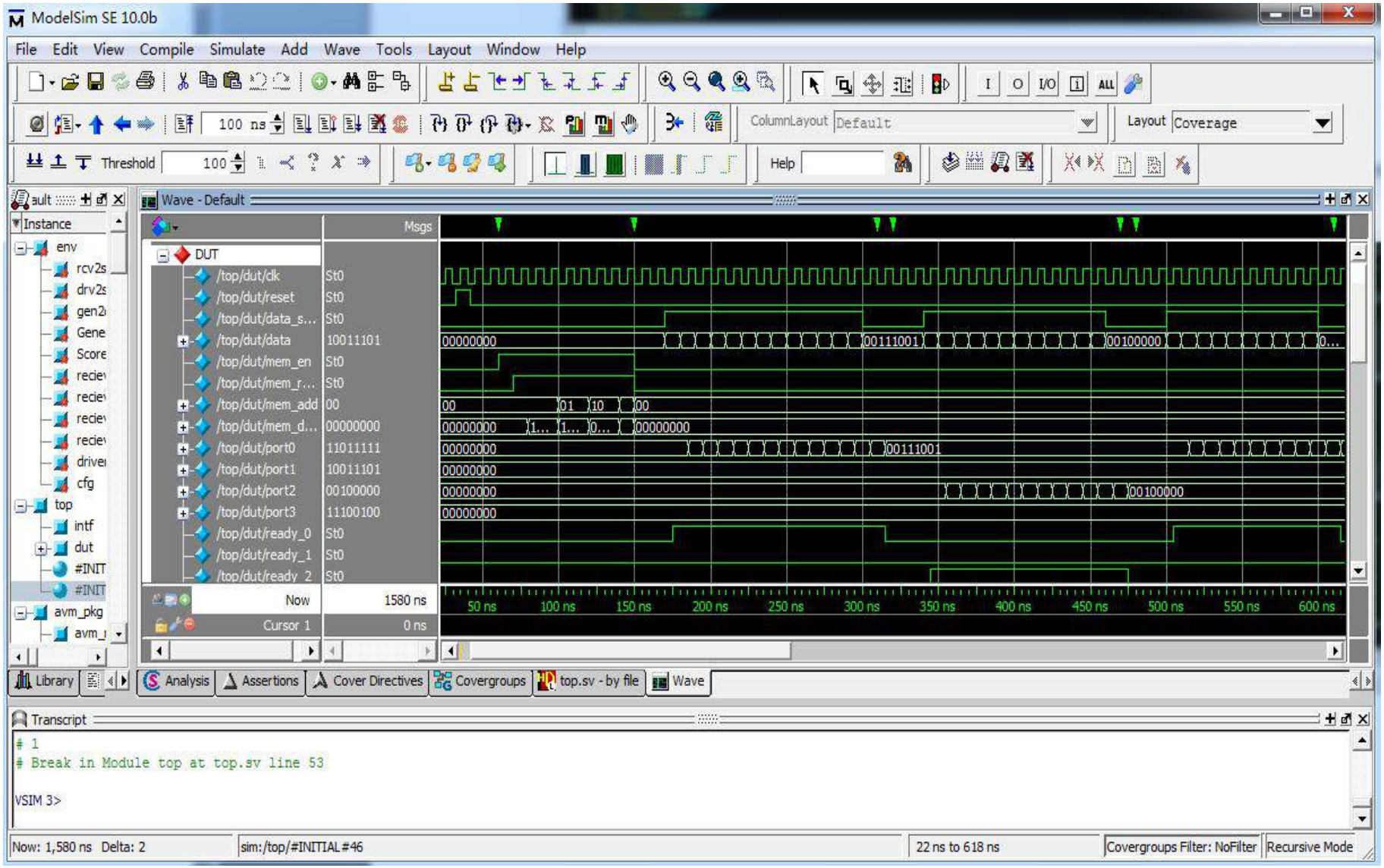


Schematic diagram



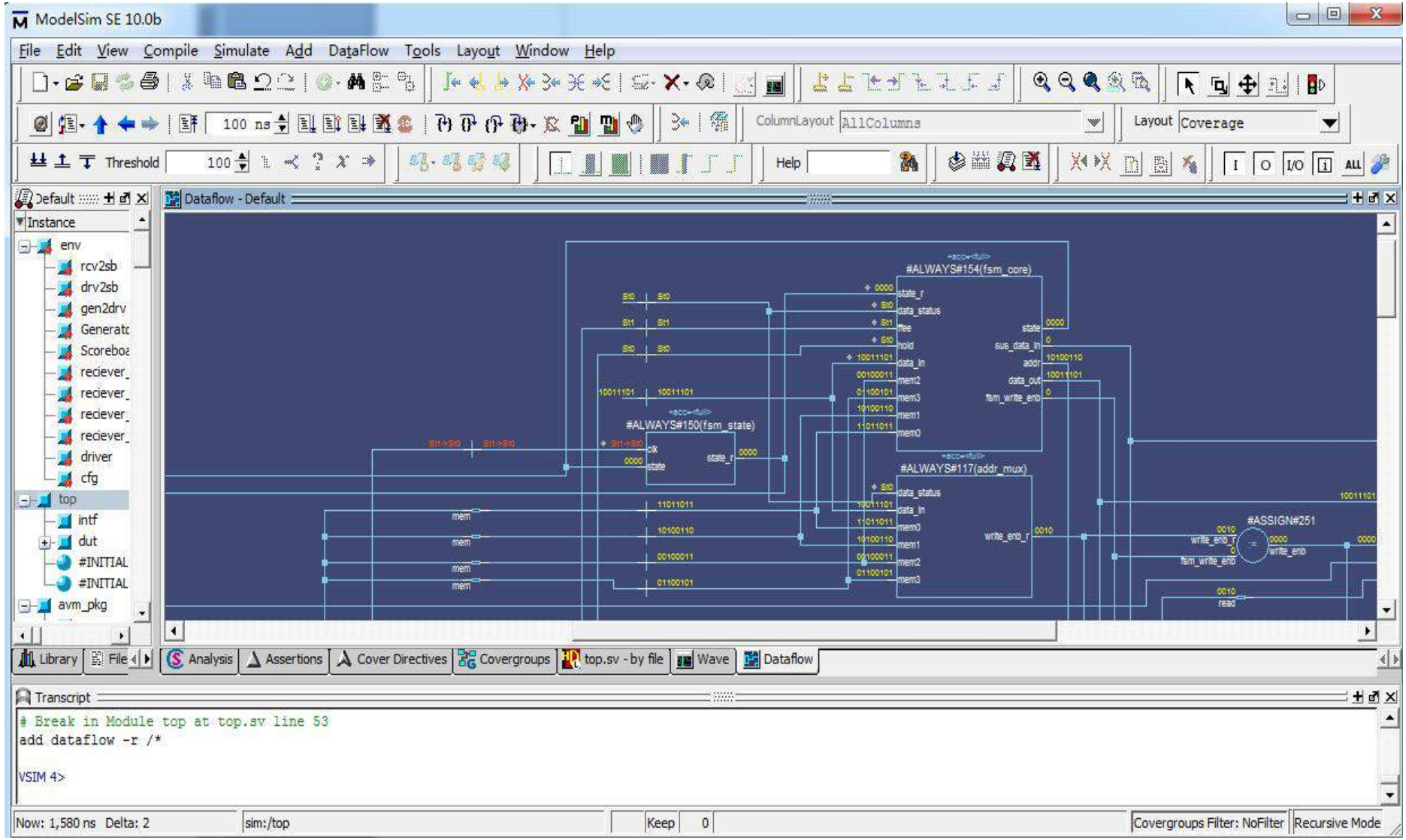
CNCS

Simulation wave

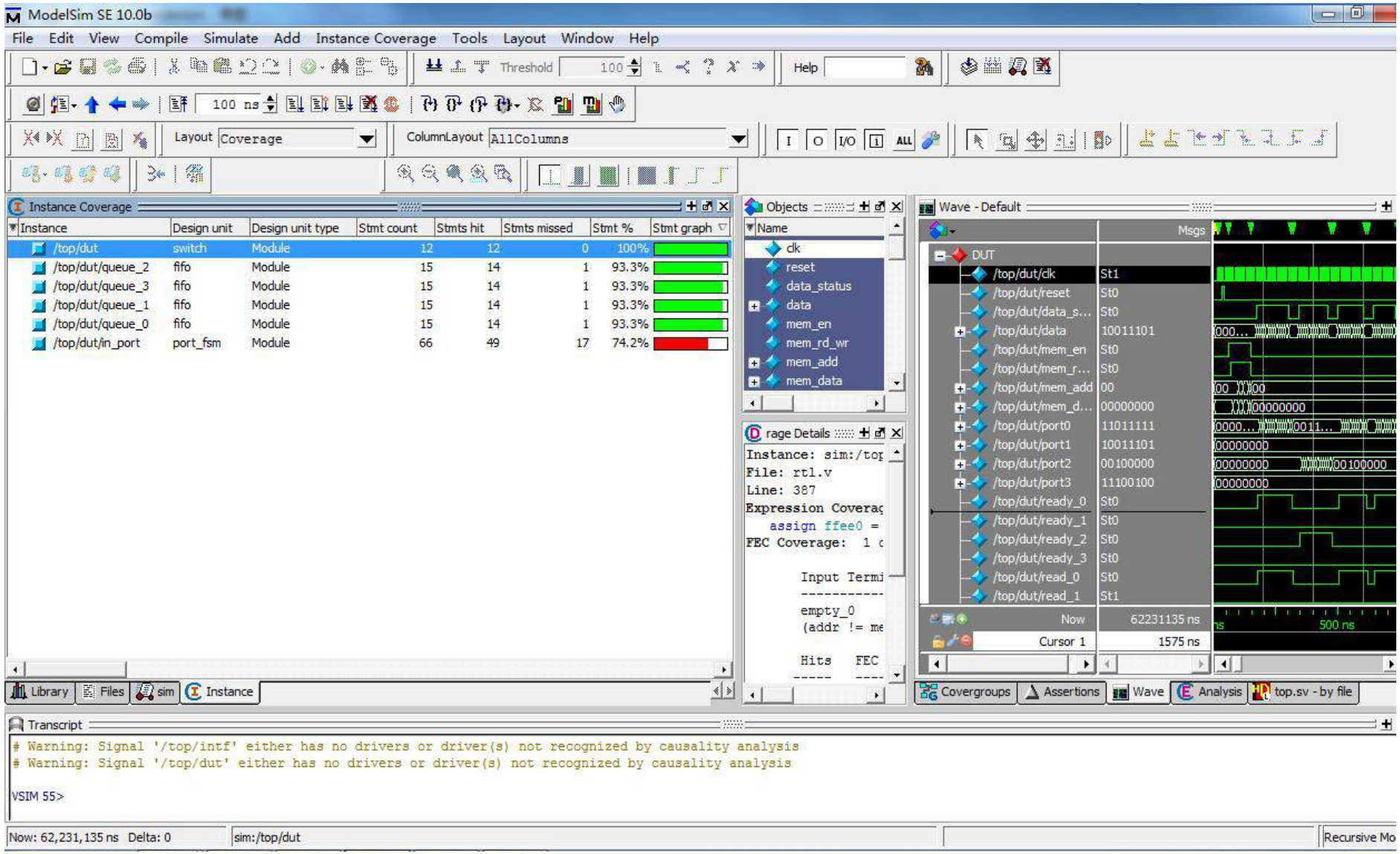


CNCS

Data flow


ModelSim SE 10.0b
File Edit View Compile Simulate Add DataFlow Tools Layout Window Help
Dataflow - Default
#ALWAYS#154(fsm_core)
#ALWAYS#150(fsm_state)
#ALWAYS#117(addr_mux)
#ASSIGN#251
Library File
Analysis Assertions Cover Directives Covergroups top.sv - by file Wave Dataflow
Transcript
Break in Module top at top.sv line 53
add dataflow -r /*
VSIM 4>
Now: 1,580 ns Delta: 2
sim:/top
Keep 0
Covergroups Filter: NoFilter
Recursive Mode

Coverage



CNCS

Thank you for your attention!

Question & Comments?