

Close Your Coverage Loop with Graph-Based Scenario Models

Jörg Große

Applications Engineer

Breker Verification Systems, Inc.



BREKER
THE SOC VERIFICATION COMPANY

Verification Futures 2012

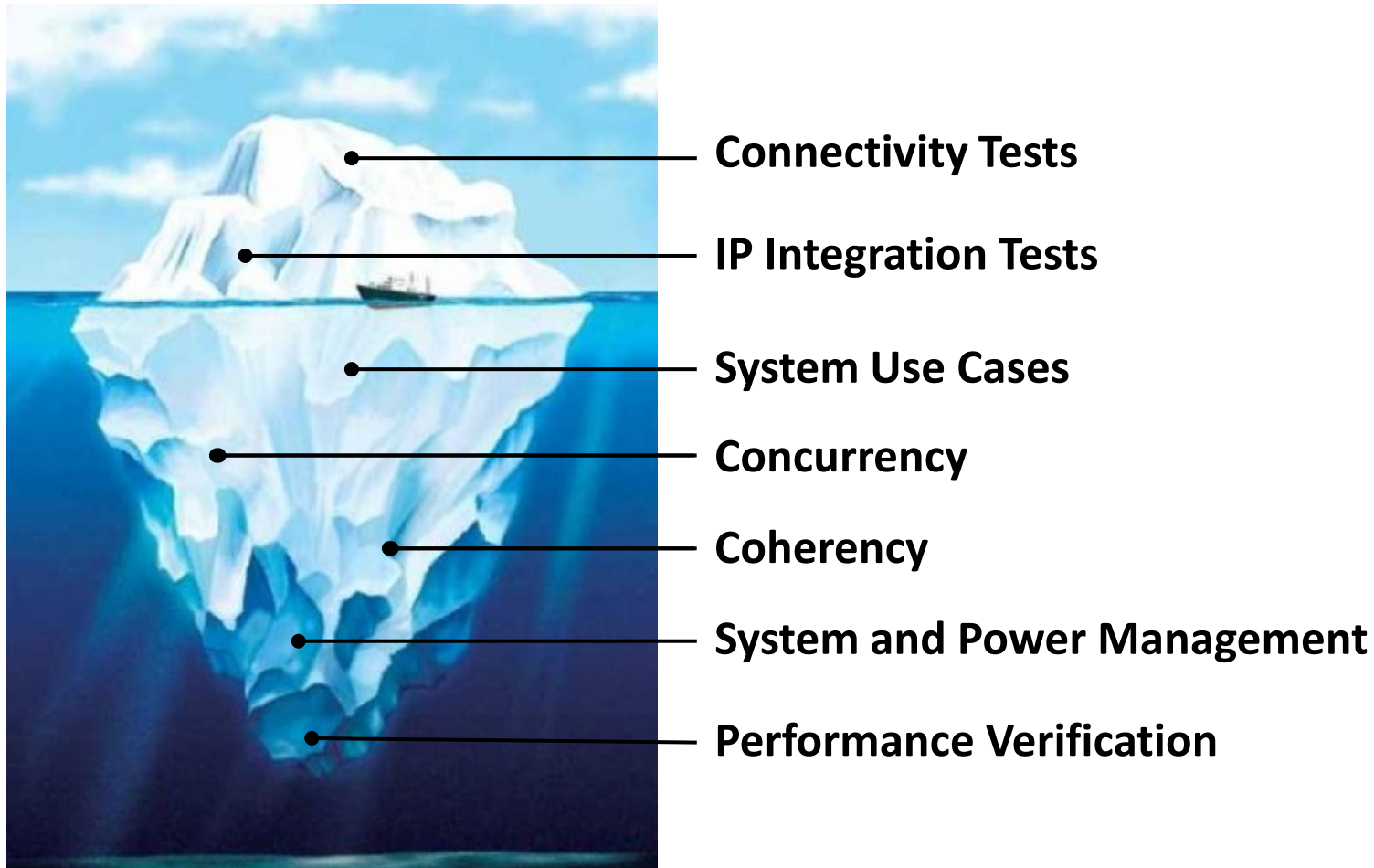
SoC Verification Has a Problem

- The Universal Verification Methodology (UVM) is breaking down for full-SoC verification
 - Full-chip simulation too slow to run long random tests
 - No link between testbench and embedded processors
- Hard to hand-write C tests for embedded processors
 - Can't track multi-threaded, multi-processor tests in parallel
 - Hand written C-test are difficult to maintain and extend
- Weak quality metrics
 - Only interface toggle coverage
 - Hand written C tests == functional coverage

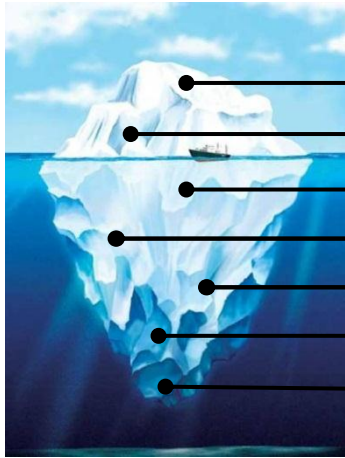
Today's Reality

- Many SoC teams just “stitch and ship”
 - Run checks for connectivity of blocks within the full chip
 - Run a few sanity tests with an incomplete UVM testbench
 - Do basic IP integration testing with hand-written C tests
 - Insufficient coverage metrics at the full-SoC level
- Some SoC teams rely on emulation and prototyping
 - Bugs found post-simulation are harder to diagnose and fix
 - Bugs found late in the project tend to delay the schedule
 - Production software is well-behaved and unlikely to hit corner cases

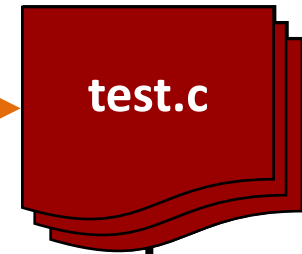
The SoC Verification “Iceberg”



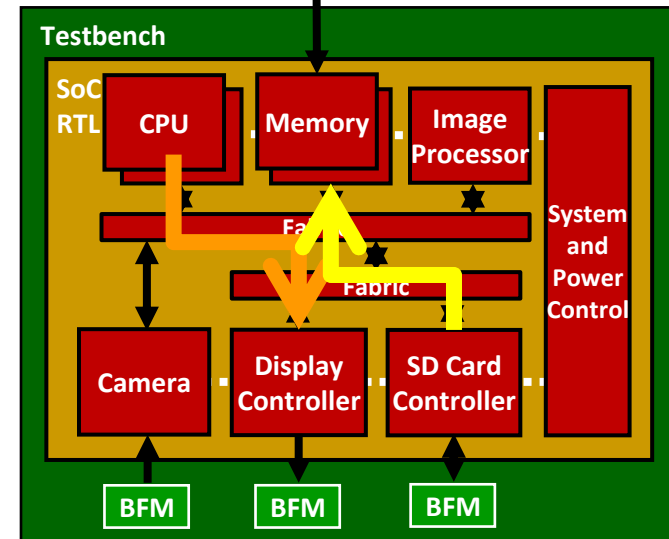
Manually Developed SoC Tests



- Connectivity Tests
- IP Integration Tests
- System Use Cases
- Concurrency
- Coherency
- System & Power Management
- Performance Verification

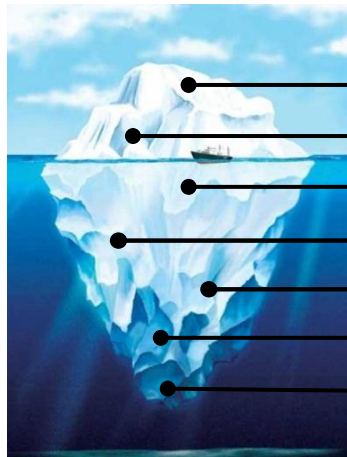
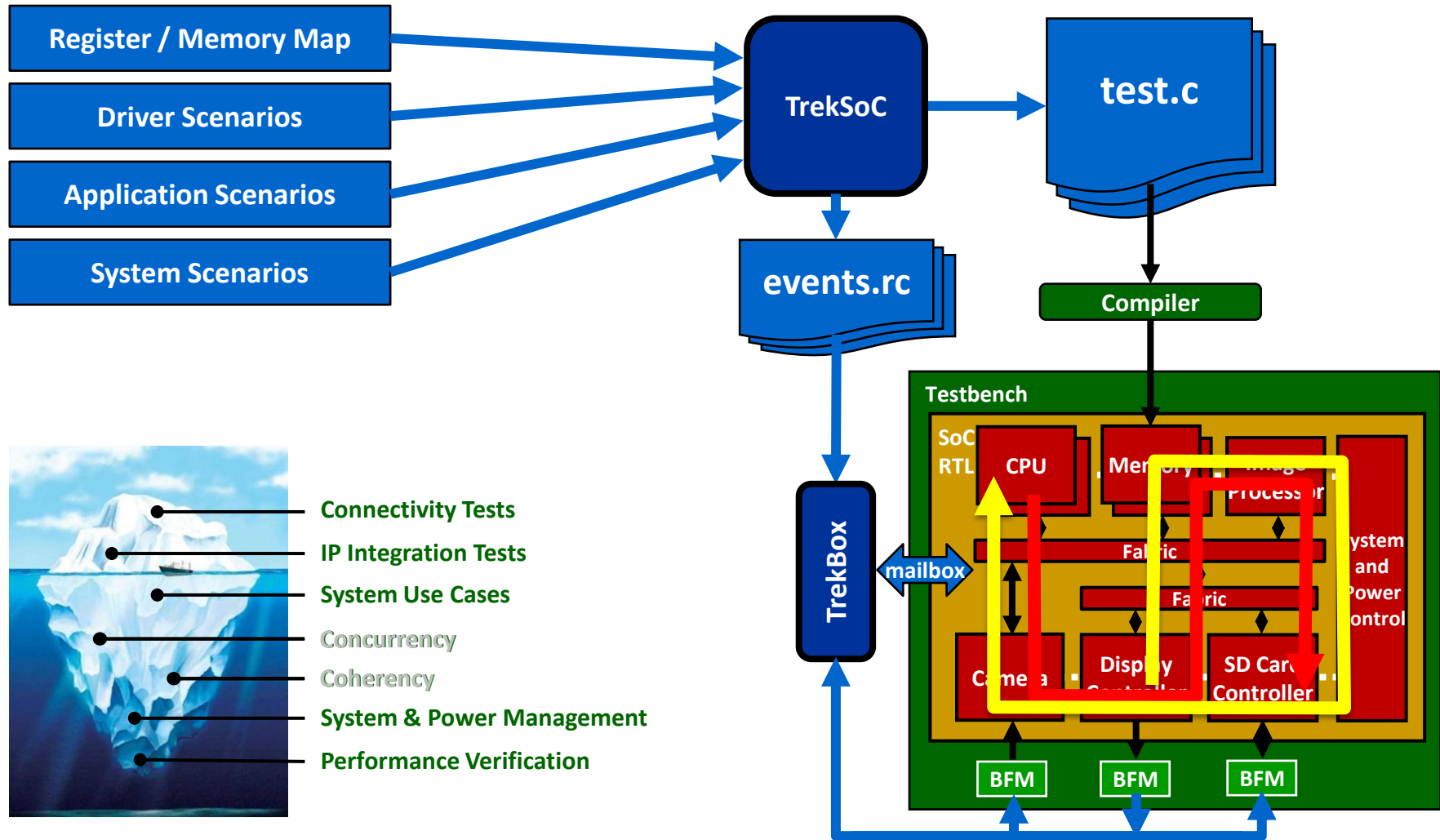


Compiler



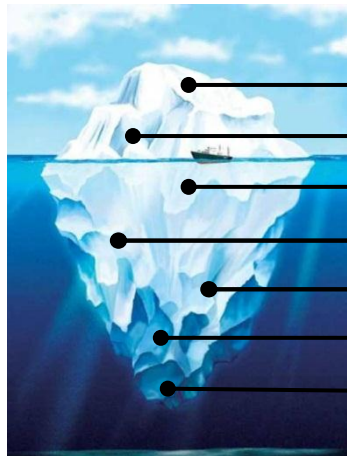
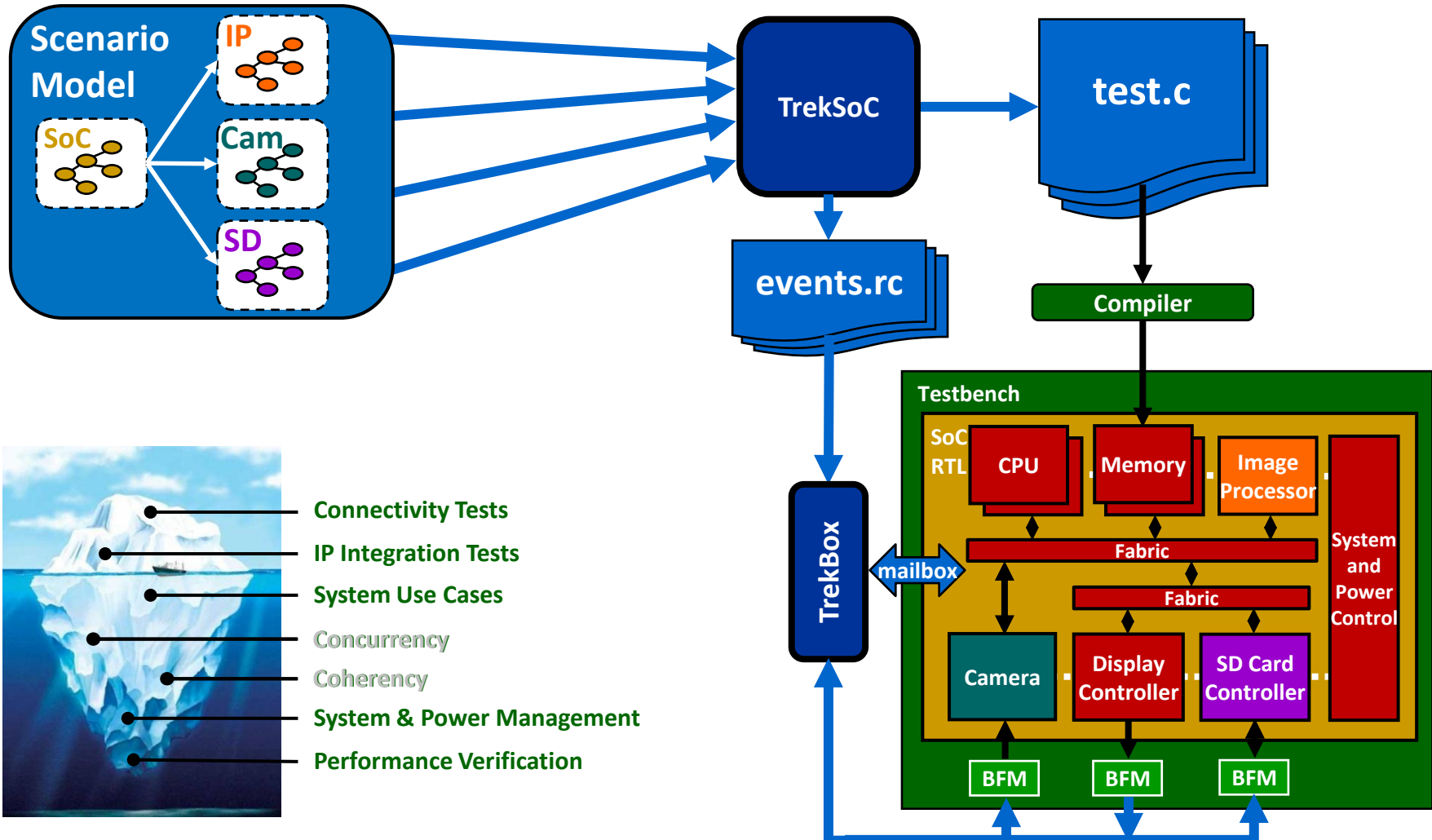
- Manual development and maintenance
- Difficult to manage:
 - Multiple threads
 - Multiple processors
 - Multiple memories
 - Interaction with I/O ports

TrekSoC Automatic Self-Verifying C Test Cases



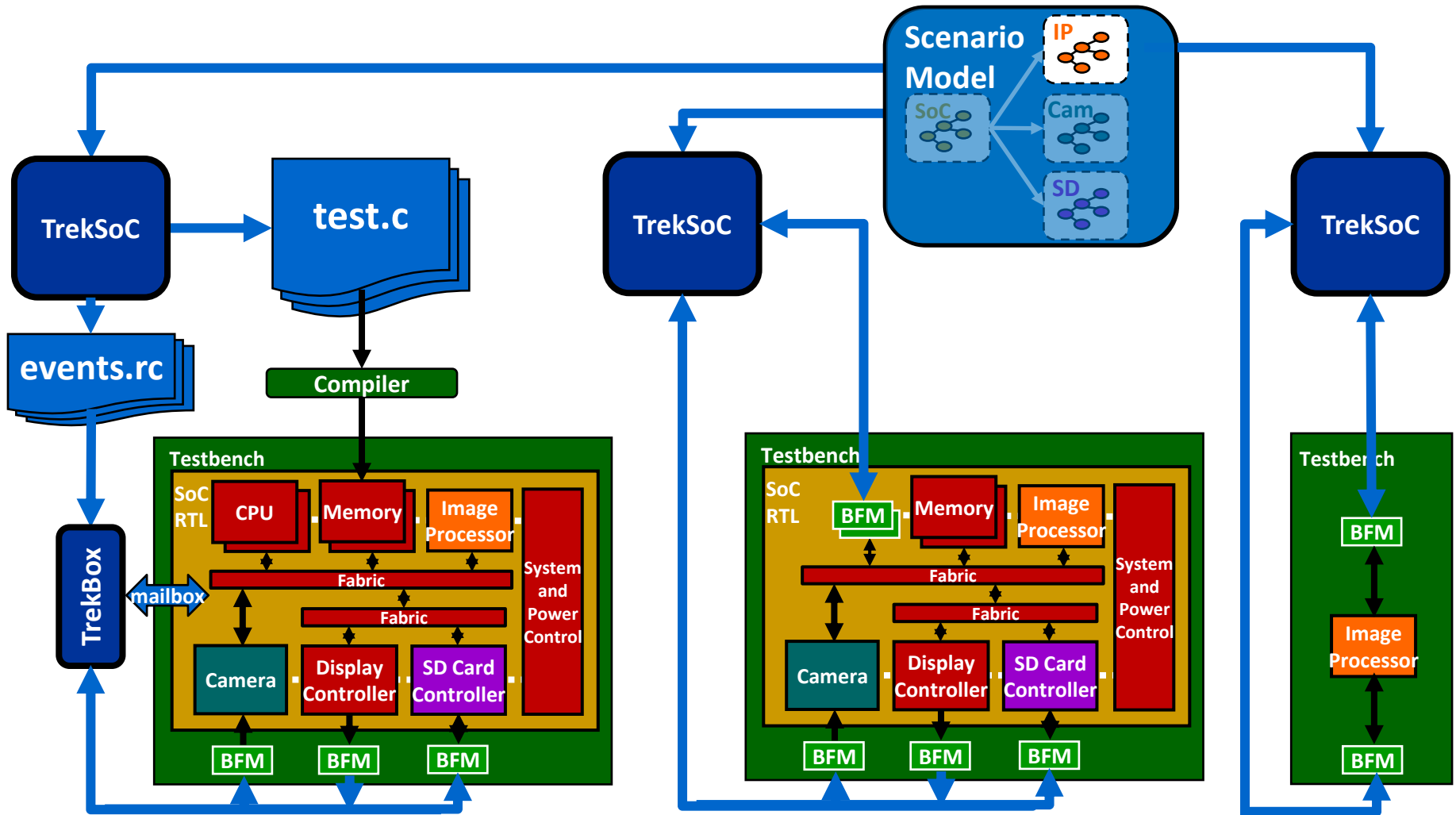
- Connectivity Tests
- IP Integration Tests
- System Use Cases
- Concurrency
- Coherency
- System & Power Management
- Performance Verification

TrekSoC Automatic Self-Verifying C Test Cases

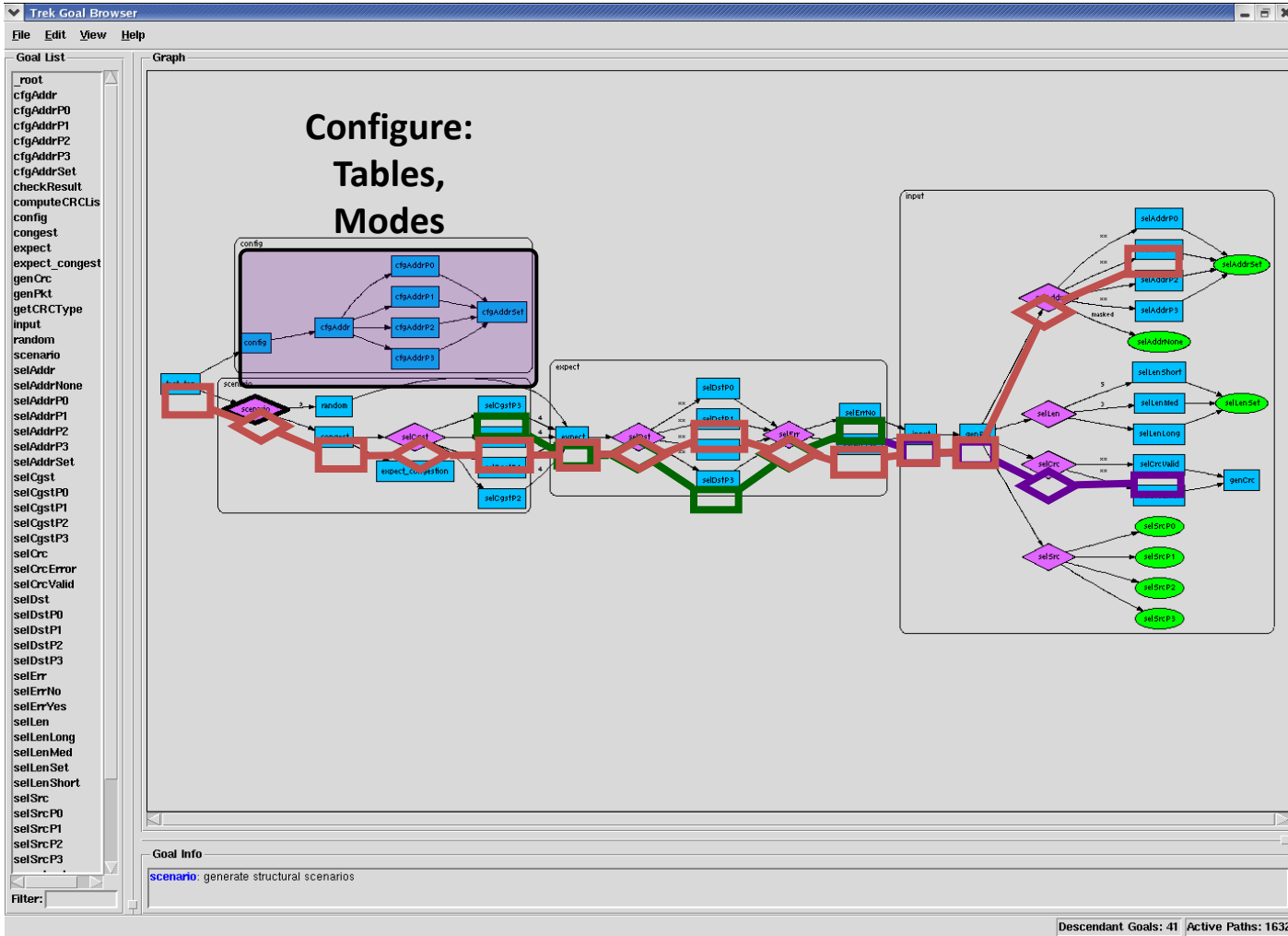


- Connectivity Tests
- IP Integration Tests
- System Use Cases
- Concurrency
- Coherency
- System & Power Management
- Performance Verification

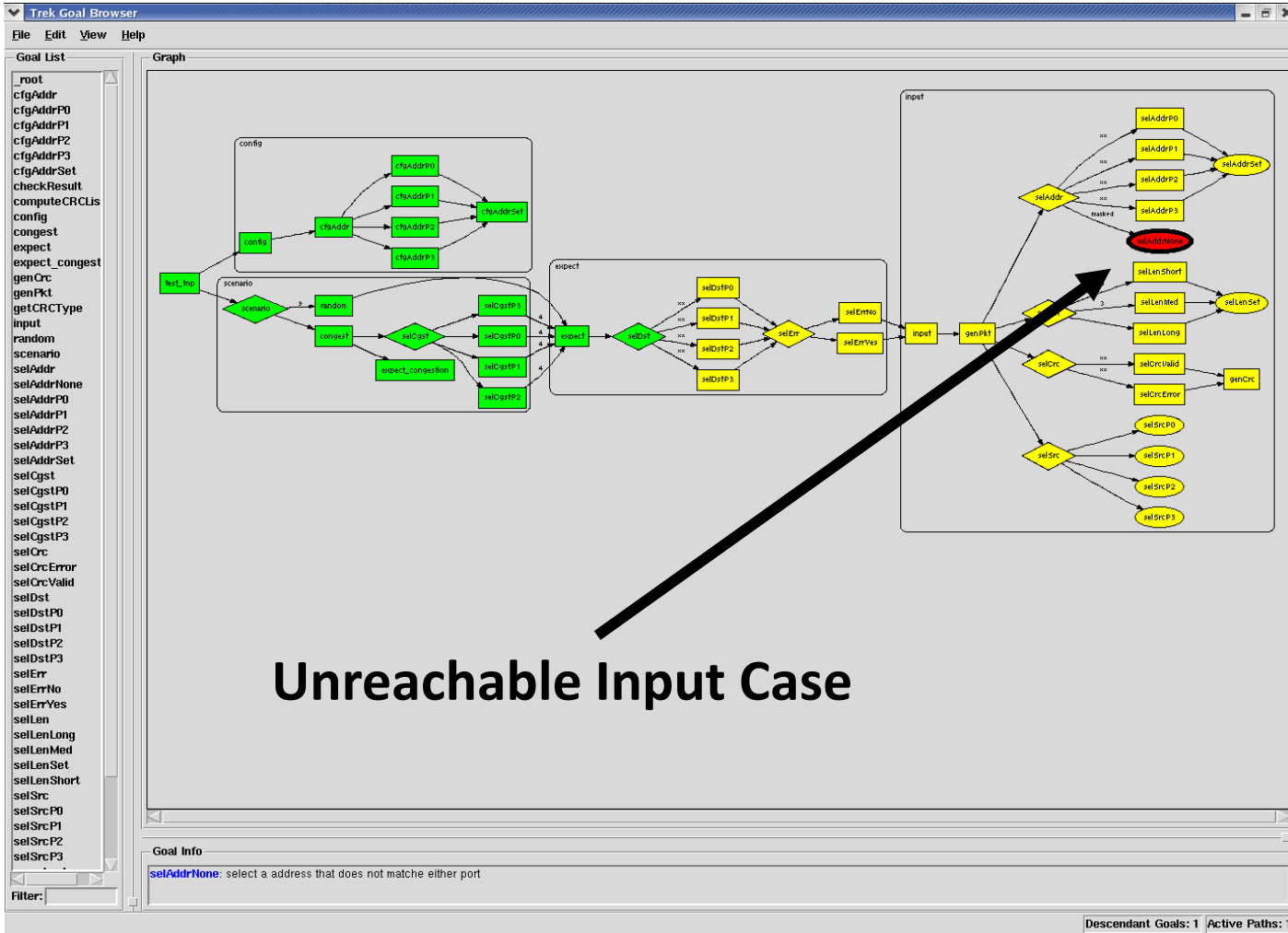
Multiple Design Level Support



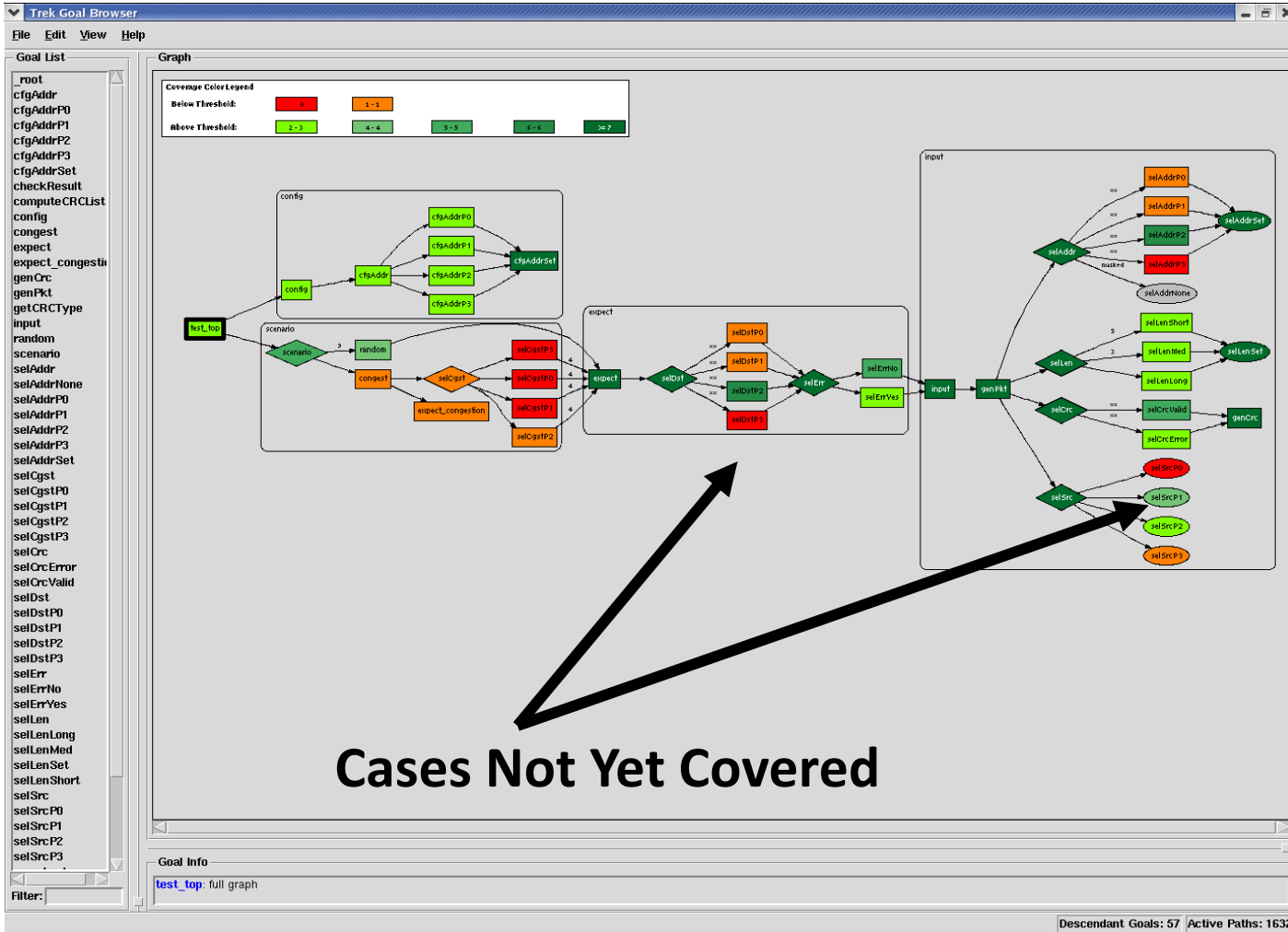
Scenario Models: Visualize Your Verification Space



Pre-Simulation Reachability



Coverage



Closed-Loop Coverage

Trek Goal Browser

File Edit View Help

Goal List

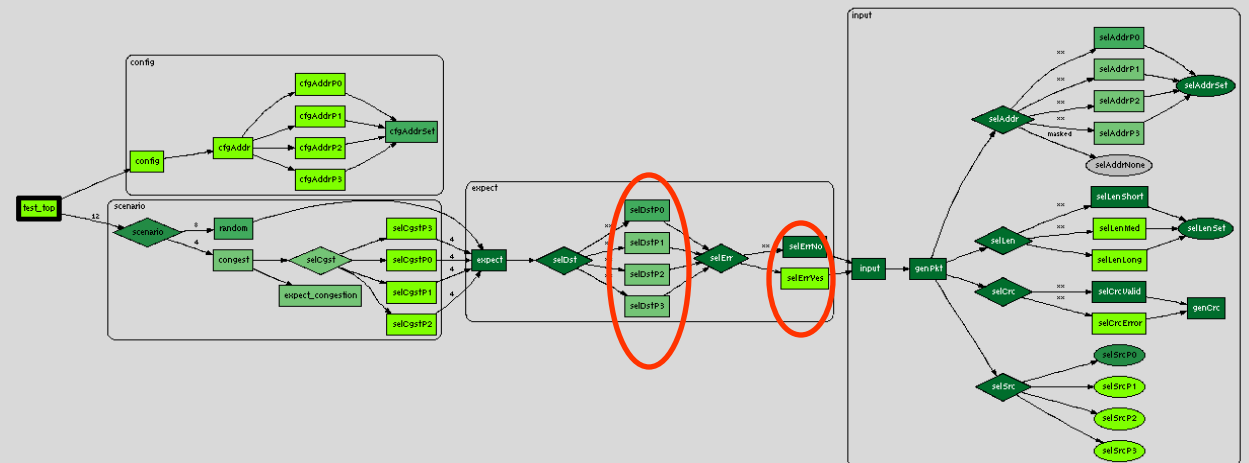
- _root
- cfgAddr
- cfgAddrP0
- cfgAddrP1
- cfgAddrP2
- cfgAddrP3
- checkRes
- computeC
- config
- congest
- expect
- expect_cc
- genCrc
- genPkt
- getCRCTy
- input
- random
- scenario
- selAddr
- selAddrNo
- selAddrP0
- selAddrP1
- selAddrP2
- selAddrP3
- selAddrSe
- selCgst
- selCgstP0
- selCgstP1
- selCgstP2
- selCgstP3
- selCrc
- selCrcErr
- selCrcVali
- selDst
- selDstP0
- selDstP1
- selDstP2
- selDstP3
- selErr
- selErrNo
- selErrYes
- selLen
- selLenLon
- selLenMed
- selLenSet
- selLenShc
- selSrc
- selSrcP0
- selSrcP1
- selSrcP2
- selSrcP3

Graph

Coverage Color Legend

Below Threshold: ■ 0 ■ 1-0

Above Threshold: ■ 1-2 ■ 2-3 ■ 4-3 ■ 5-4 ■ 6-5



Automatically Close Coverage Targets with minimal cycles
 Example: "cross the two sets and walk all 8 paths"

Goal Info

test_top: full graph

Filter:

Descendant Goals: 57 Active Paths: 1632

Summary

- Stitch and ship verification is not sufficient for an SoC
- TrekSoC automatically generates C test cases
 - Run on the embedded processors and link to testbench
 - Quickly satisfy your functional coverage requirements
 - But still generating deep corner case scenarios
- Scenario models are easy and natural to create
 - Graphs efficiently capture verification knowledge
 - Graphs help internal communication and allow teams to define the functional coverage requirements



Thanks for Listening!