

# Using SystemC in the RTL Design Flow

February 22, 2006

Lawrence Case and  
Sergey Sokol

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © Freescale Semiconductor, Inc. 2005.



# Agenda

- Objectives
- Benefits
- Design Overview
- Testbench Overview
- Conclusion
- Question

# Prioritized Objectives

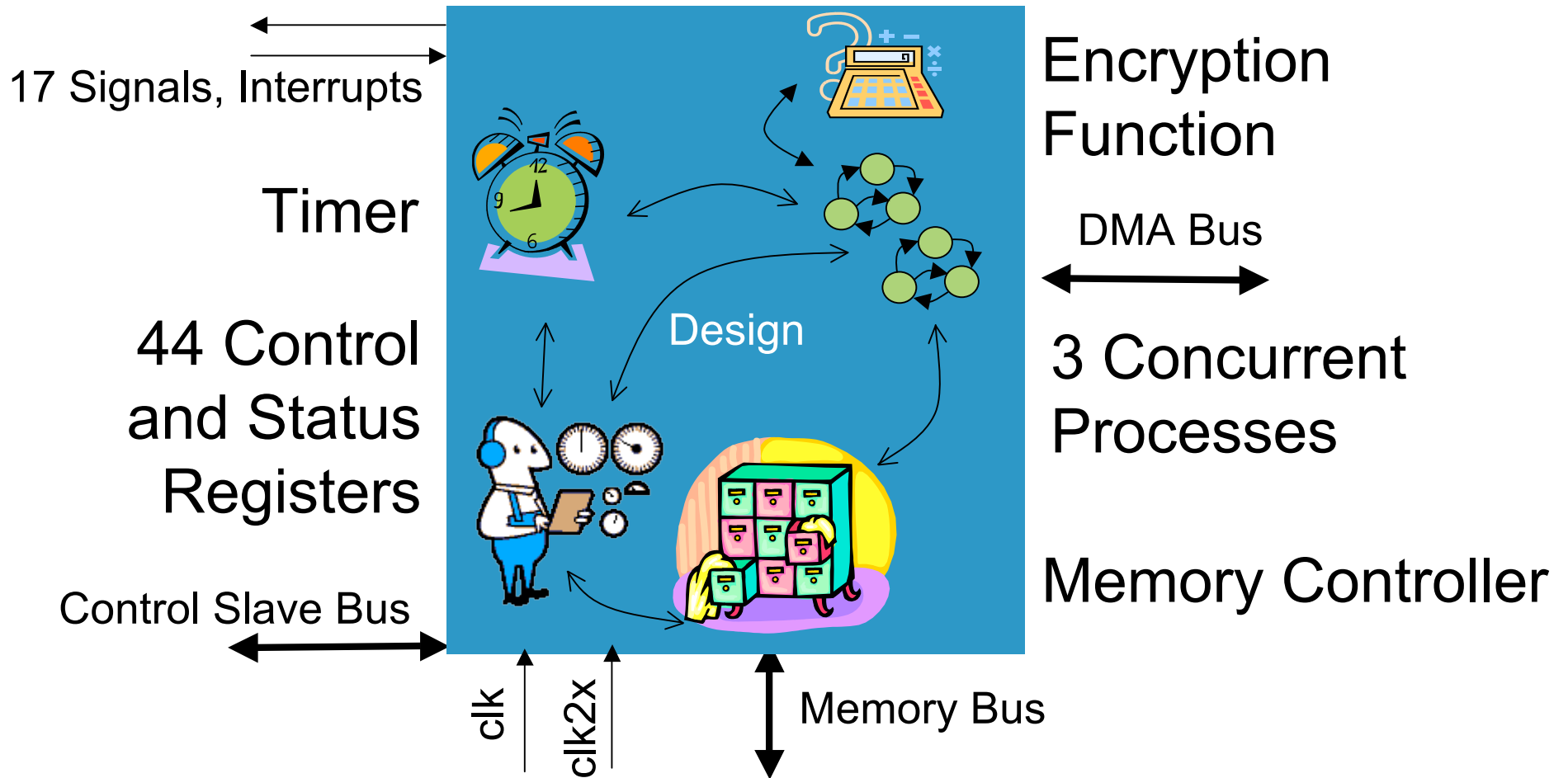
1. Deliver fully verified RTL
2. Deliver model for software driver development
3. Automate stimulus generation.
4. Validate architectural enhancements prior to modifying RTL.

# Benefits From Experience

- Constant signal monitoring.
- Accurate Driver Developer's model.
- Golden and Guidance references both help.
- “Golden-izing” model leads to randomizing vectors.
- Strengths of each representation supports the other.
- 3 votes/views (Spec, Model, RTL)
- Model view provides “higher” perspective.

# Design Section

# Challenging Design Features for Model




# Model Requirements

- Bit Accurate Control and Status Registers.
- Correct Sequence of Commands and Status
- Correct Sequence of I/O Signals and Interrupts.
- Data Accurate Functions.
- Expected Responses Generated during Simulation
- Model Integrates into SW and Verification Environments

# Model Techniques – Interrupt-able Thread

```
function thread
while (1) {
    wait (function event);
    try (
        setup ();
        ...
        execute ();
        do wait (delay, event);
        finish ();
    )
    catch failure (
        set failure condition ();
        failure notify;
    )
}
```



```
do wait (delay, event) {
    wait (delay_evt | event);
    if (failure is asserted)
        throw failure event;
}
```



# Model Techniques – Reset Thread

```
reset thread
while (1) {
    wait (reset asserted);
    clear registers ();
    wait (reset deasserted);
    notify startup threads
    ...
}
```

# Model Techniques – Timer Status

```
case Timer Write
```

```
...
```

```
previous cycles = clock ->get cycle();
```

```
}
```

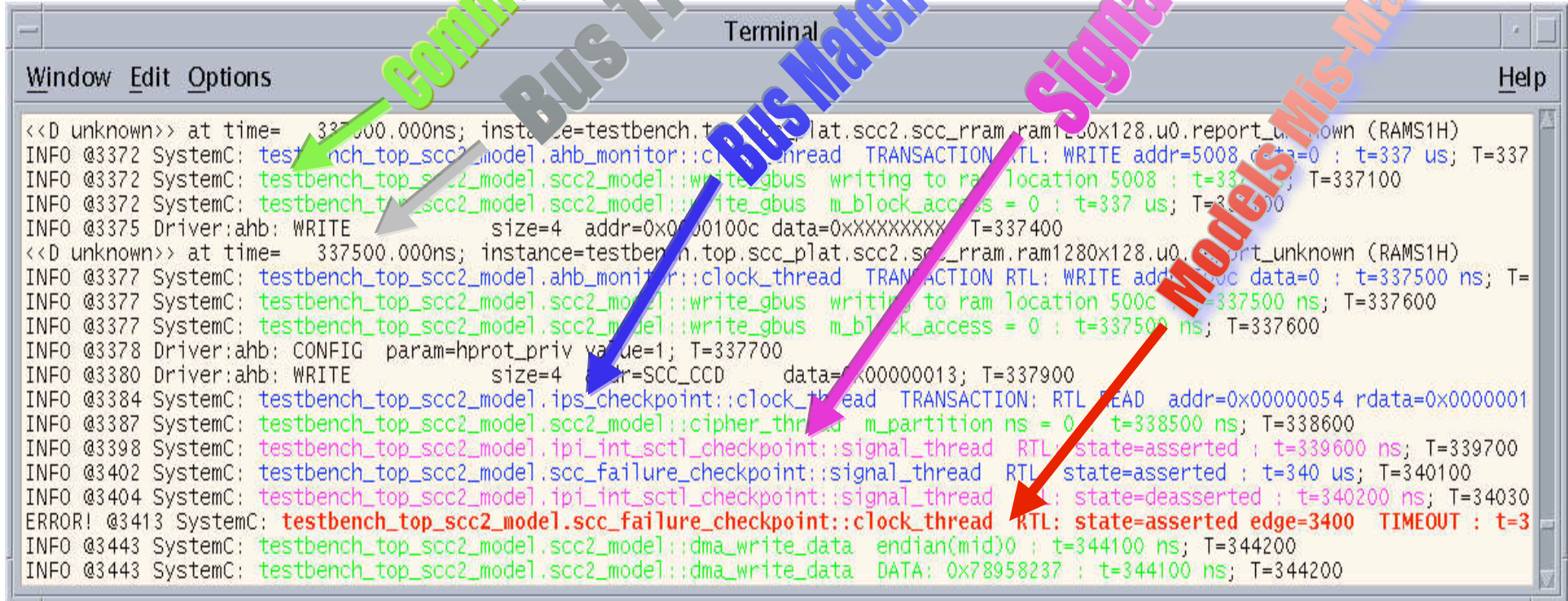
```
case Timer Read
```

```
...
```

```
time = clock -> get cycle() - previous cycles;
```

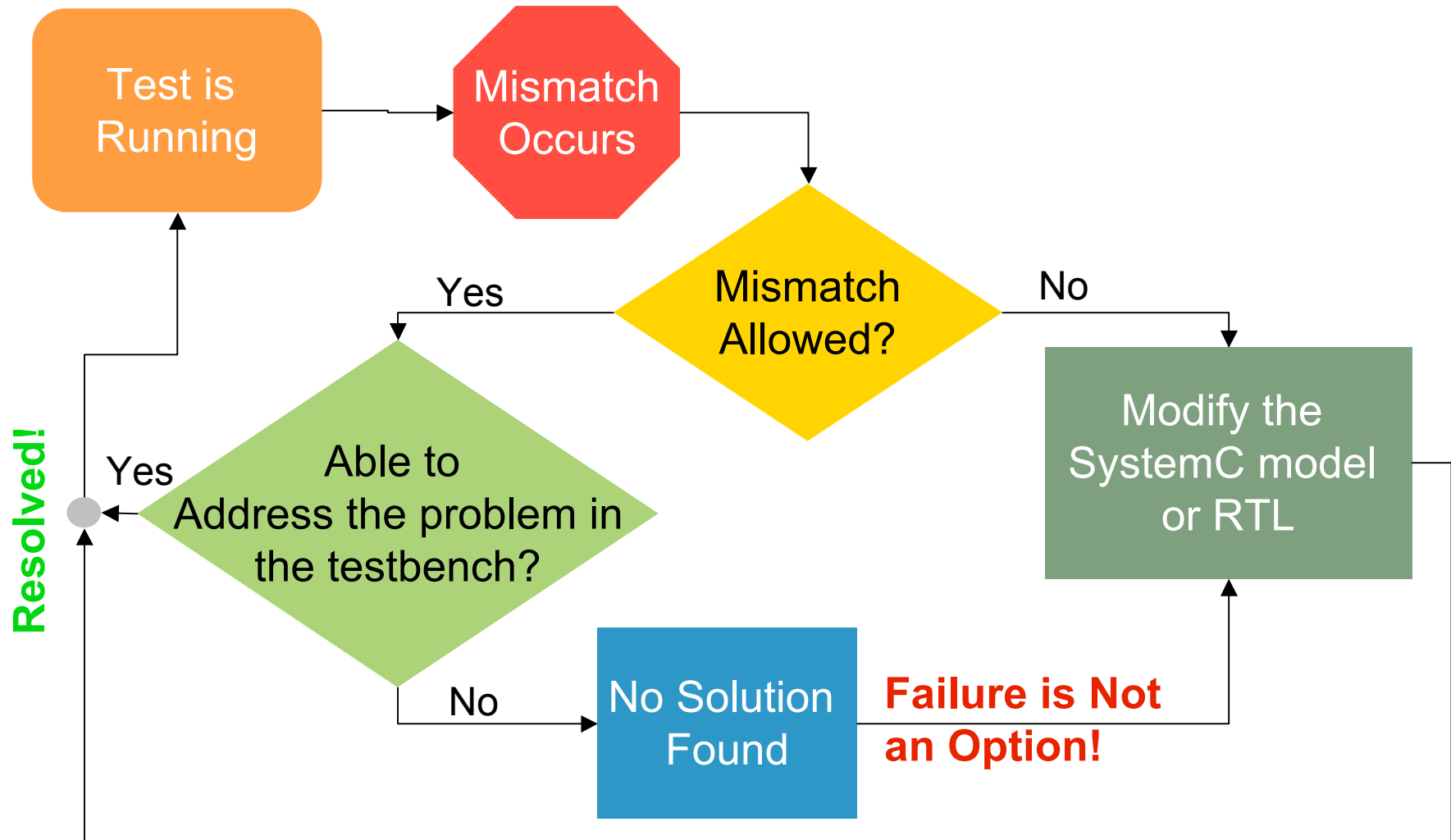
```
}
```

# Snapshot of Output Log



```
<<D unknown>> at time= 337000.000ns; instance=testbench_top.scc2_model.plat.scc2.scc_rram.ram1280x128.u0.report_unknown (RAMS1H)
INFO @3372 SystemC: testbench_top.scc2_model.ahb_monitor::clock_thread TRANSACTION RTL: WRITE addr=5008 rdata=0 ; t=337 us; T=337
INFO @3372 SystemC: testbench_top.scc2_model.scc2_model::write_gbus writing to ram location 5008 ; t=337000 ns; T=337100
INFO @3372 SystemC: testbench_top.scc2_model.scc2_model::write_gbus m_block_access = 0 ; t=337 us; T=337100
INFO @3375 Driver:ahb: WRITE size=4 addr=0x0000100c data=0XXXXXXXXX ; T=337400
<<D unknown>> at time= 337500.000ns; instance=testbench_top.scc2_model.plat.scc2.scc_rram.ram1280x128.u0.report_unknown (RAMS1H)
INFO @3377 SystemC: testbench_top.scc2_model.ahb_monitor::clock_thread TRANSACTION RTL: WRITE addr=500c rdata=0 ; t=337500 ns; T=
INFO @3377 SystemC: testbench_top.scc2_model.scc2_model::write_gbus writing to ram location 500c ; t=337500 ns; T=337600
INFO @3377 SystemC: testbench_top.scc2_model.scc2_model::write_gbus m_block_access = 0 ; t=337500 ns; T=337600
INFO @3378 Driver:ahb: CONFIG param=hprot_priv value=1; T=337700
INFO @3380 Driver:ahb: WRITE size=4 addr=SCC_CCD data=0x00000013; T=337900
INFO @3384 SystemC: testbench_top.scc2_model.ips_checkpoint::clock_thread TRANSACTION: RTL READ addr=0x00000054 rdata=0x0000001
INFO @3387 SystemC: testbench_top.scc2_model.scc2_model::cipher_thread m_partition ns = 0 ; t=338500 ns; T=338600
INFO @3398 SystemC: testbench_top.scc2_model.ipi_int_sctl_checkpoint::signal_thread RTL state=asserted ; t=339600 ns; T=339700
INFO @3402 SystemC: testbench_top.scc2_model.scc_failure_checkpoint::signal_thread RTL state=asserted ; t=340 us; T=340100
INFO @3404 SystemC: testbench_top.scc2_model.ipi_int_sctl_checkpoint::signal_thread : state=deasserted ; t=340200 ns; T=34030
ERROR! @3413 SystemC: testbench_top.scc2_model.scc_failure_checkpoint::clock_thread RTL: state=asserted edge=3400 TIMEOUT : t=3
INFO @3443 SystemC: testbench_top.scc2_model.scc2_model::dma_write_data endian(mid)0 ; t=344100 ns; T=344200
INFO @3443 SystemC: testbench_top.scc2_model.scc2_model::dma_write_data DATA: 0x78958237 ; t=344100 ns; T=344200
```

# Mismatch Handling



# Observations

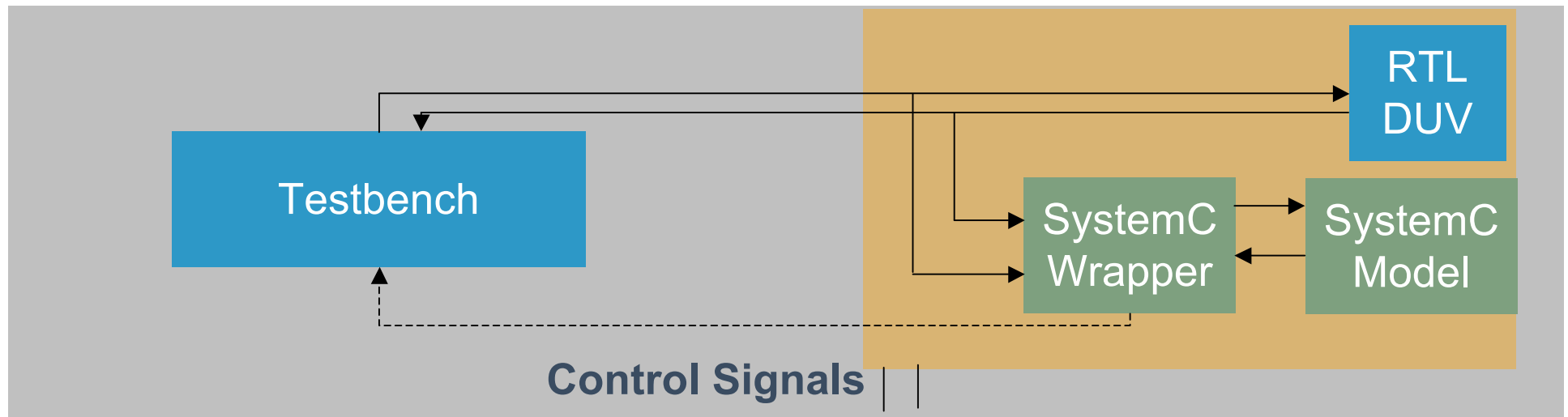
- 2 model method takes extra communication, TB overhead, and of course, code.
- Golden reference not always model, even when model component designed first.
- At this level, model tended to be buggier than RTL
- Lines: RTL ~10000, Model ~5000 (ex. Encryption)
- Yet, at least one bug found that would not have been found in traditional flow.
- Tens of thousands of randomized vectors passed.

# Testbench Section

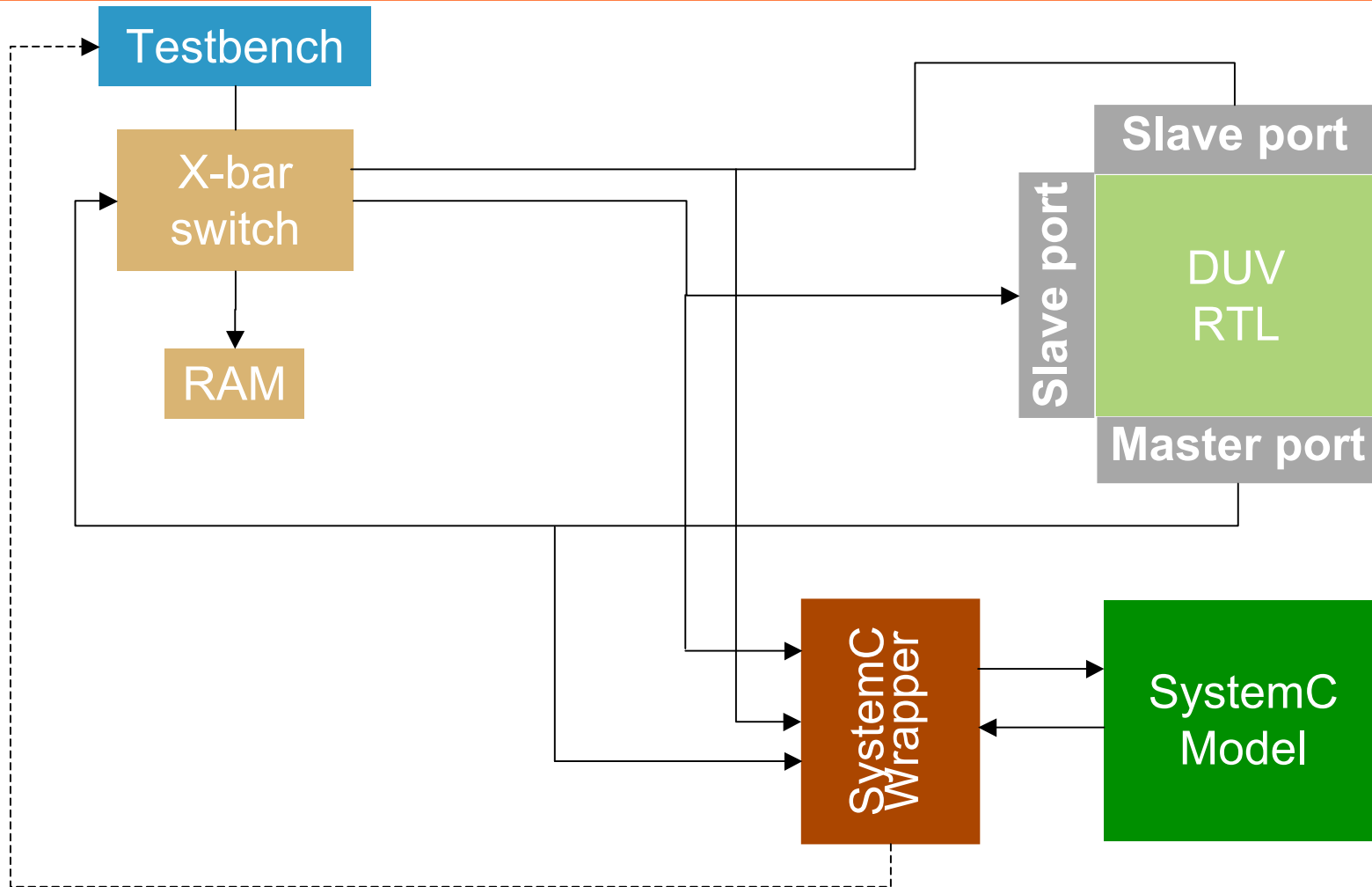
# Testbench Section

# Dynamic Response Checking

- RTL – developed by designer
- SystemC model ideally developed independently
- Dynamic response checking on interfaces



# RTL Platform with SystemC Model



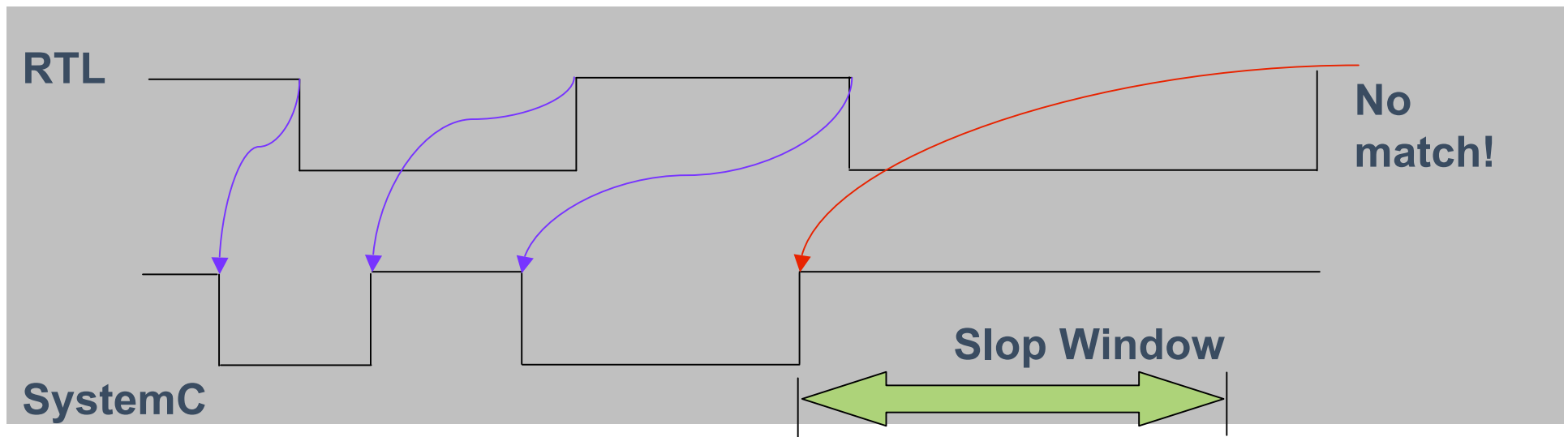


# Test Bench and the DUV

- Testbench provides stimulus to RTL
- SystemC wrapper of design
  - Snoops stimulus and presents it to Model
  - Compares responses with “Slop” window – **Coming Up!!**
  - Forwards “Matching” information to Test Bench
- Test Bench Logs output and discrepancies

# “Slop” Window

- Why “Slop” Handling?
- Testbench queues response events from RTL and model
- Testbench compares within “Slop” window



# “Slop” Window

RTL  
Queue

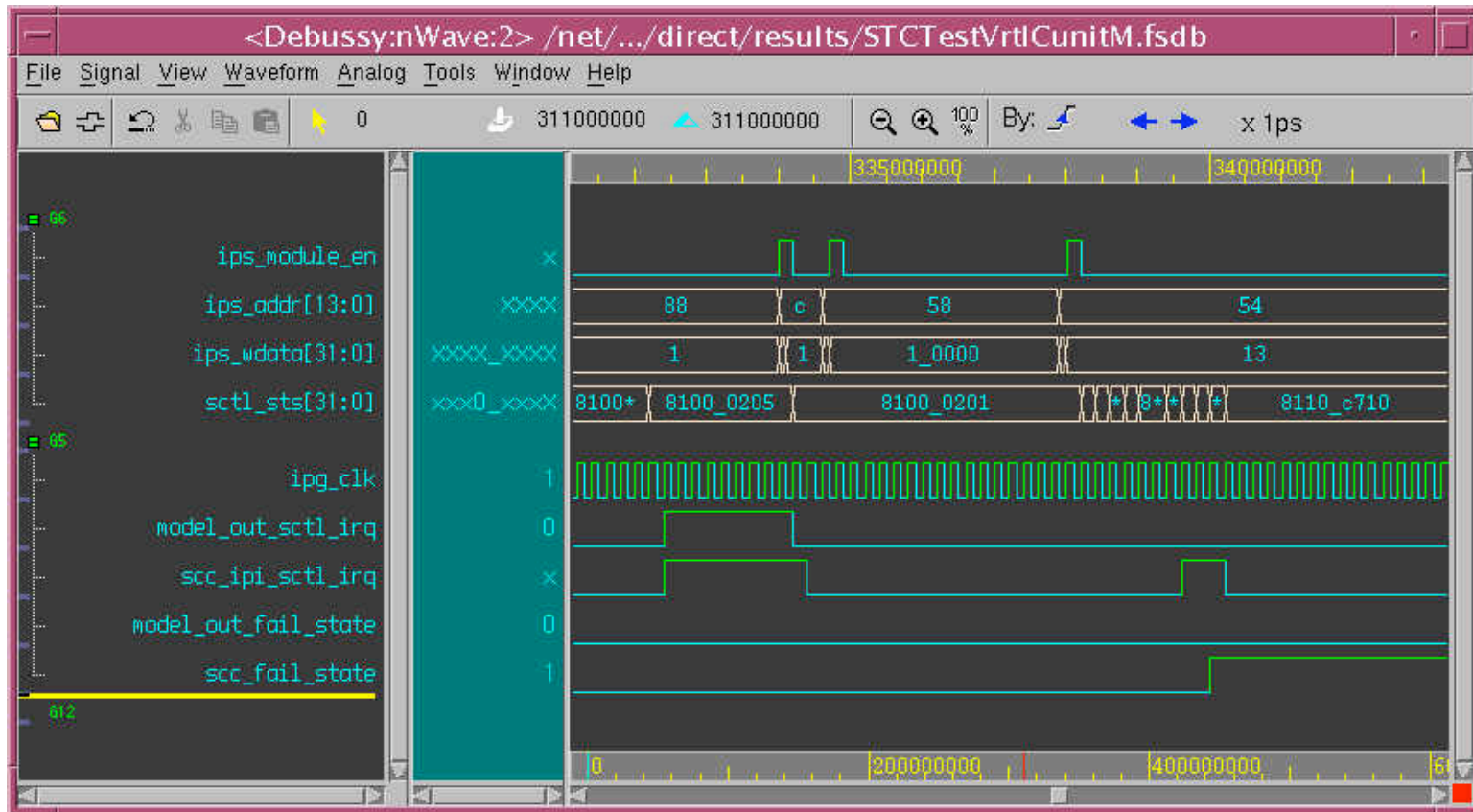
Event R3, Time R3  
Event R2, Time R2  
Event R1, Time R1

Event S3, Time S3  
Event S2, Time S2  
Event S1, Time S1

SystemC  
Queue

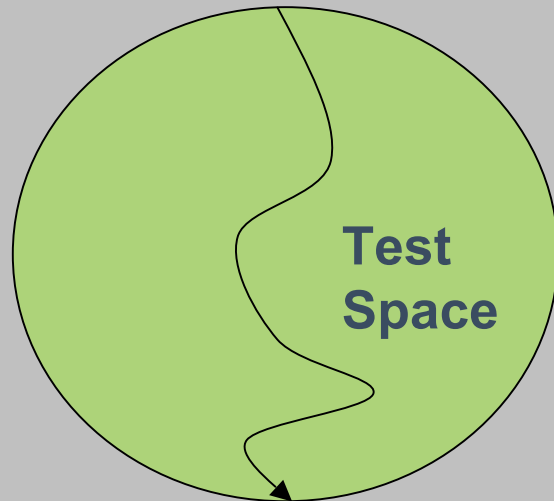
Compare  
within “Slop”  
Window  
 $|R1 - S1| < \text{Slop}$

# Real World Example 1

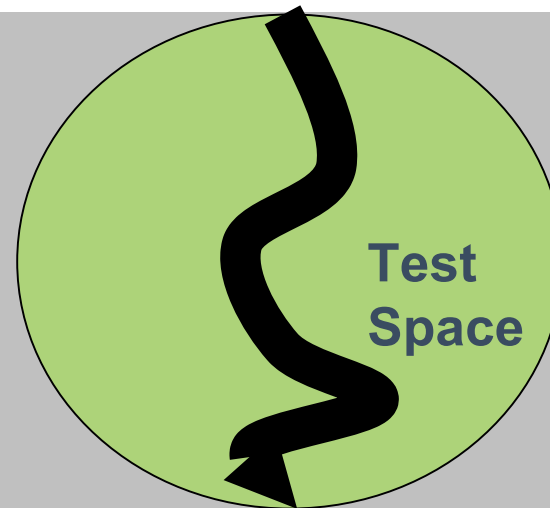


# Getting Good Coverage

- Develop a set of “Directors” that...
  - Emulate a software driver
  - Randomize over all possible parameters
- Corner cases not covered by Directors



**Directed  
Test**



**Randomized  
Test**

# Resolved in Testbench

- Ignoring discrepancies during Reset.
- Matching events within time frame.
- Re-synchronization after missed events.
- Slop Window handling *some* drift issues.

*Need more mismatch handling and order-enforcement techniques.*

- Testbench is staying the course on current project despite challenges in schedule.
- Plan to continue testbench on next project with new ideas incorporated.

