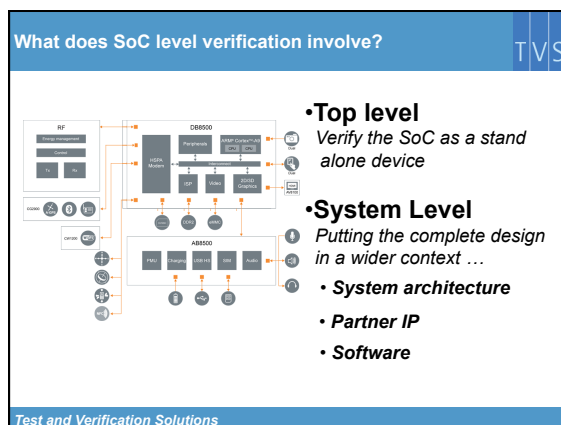
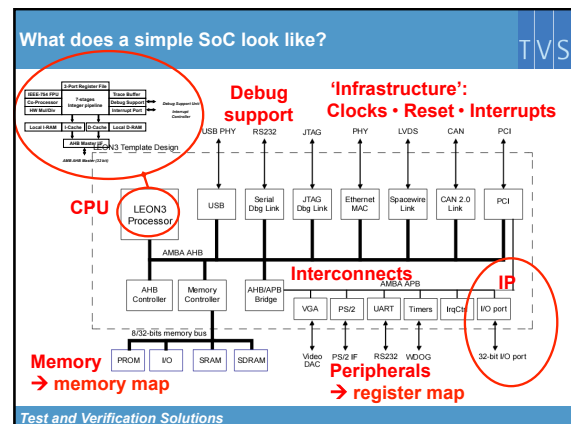


TVS

SoC Verification

Mike Benjamin
Associate at TVS

Test and Verification Solutions



- TVS
- ## Why write SoC level tests?
- **Some top level functionality not visible at unit level**
 - Imported IP
 - Signal connectivity
 - Power management
 - Coherence?
 - **Register / address mapping**
 - **Performance verification**
 - **Power on / reset**
 - **Benchmarking**
 - **Allows verification to focus on actual use model**
 - Testing restricted to real use model
 - Configurability / parameterized blocks instantiated!
 - **Generate typical/worst case waveforms for power analysis!**
 - **Missing system level functionality & compliance testing**
 - Partner IP
 - Software
 - System architecture
- Test and Verification Solutions

- TVS
- ## Why bother doing unit level testing?
- **Controllability at top level v unit level?**
→ **REDUCED**
 - Harder to hit corner case and longer run times
 - **Visibility at top level v unit level?**
→ **REDUCED**
 - Harder to debug fails
 - **Overhead on testing at top level v unit level?**
→ **INCREASED**
 - Need working top level integration before testing
 - Need to propagate block level fixes/changes to top level before they can be tested
 - Need to understand the complete SoC to test and debug a single block
- Test and Verification Solutions

- TVS
- ## Barriers to top level testing
- **Barriers to top level verification?**
 - B1: Complexity of building the complete top level design
 - B2: Late availability of key blocks / functionality
 - B3: Difficulty of anyone understanding the complete design
 - B4: Size of full top level design
 - B5: Limited controllability of the design from outside
 - B6: Limited visibility inside design
 - **Solutions?**
 - S1: Require changes to be co-ordinated between dependent blocks
 - S2: Regression testing before changes are committed
 - S3: A schedule defining milestones for delivering features
 - S4: Ensure major interfaces are stable and well defined
 - S5: Black box some components
 - S6: Replace components with abstract models or BFM's (eg. CPU, memories)
-
- Test and Verification Solutions

Reuse from unit level?

TVS

- VIP**
 - BFMs
 - Monitors and scoreboards
 - Protocol checkers
- Assertions**
- Functional coverage points**
- Tests**
 - Integration tests
 - Connectivity, address mapping
 - Stress tests
 - Cross cutting concerns such as interrupts or power management
 - Shared resources or 'convergence points' (eg: memory synchronisation)
 - Right level of abstraction
 - Transactions and/or bus accesses
 - Relative address map

Need to plan for reuse!

Test and Verification Solutions

What do our top level tests contain?

TVS

- Tests are typically C programs running on an SoC CPU
- Loaded into SoC memory
- Component tests
- Register / address map
- Result checking
- Trace and error reporting
- Halt mechanism
- Interrupt handling

```

main()
{
    report_start(1, 0x0000200, 0);
    test_main(0x0000200, 8);
    report_end(1);
}

int gpio_test(int addr)
{
    gpio = (int *) addr;
    int mask;
    int width;

    report_device(0x0101a000);
    gpio[3] = 0; gpio[2] = 0; gpio[1] = 0;
    gpio[2] = 0xFFFFFFFF;

    /* determine port width and mask */
    mask = 0; width = 0;

    while( (gpio[2] > width) & 1) && (width <= 32) {
        mask = mask | (1 << width);
        width++;
    }

    gpio[2] = mask;
    if (gpio[0] & mask) != 0) fail(1);
    gpio[1] = 0x9ABCDEF;
    if (gpio[0] & mask != (0x9ABCDEF & mask)) fail(2);
    gpio[2] = 0;
    return width;
}

```

Test and Verification Solutions

How to check the test results

TVS

- Fail causes test to hang
- Dump results to memory and compare to reference results from model
 - mpeg decoder video stream
 - reference simulator
- Explicit checks in the test
 - Observe and count interrupts
 - Check data values
- Trace comparison
 - Compare simulation state to a reference model cycle by cycle during the simulation
- Use of monitors, scoreboards or assertions

Need error propagated to end of test

Sensitive to accuracy of reference model (especially timing)

Test and Verification Solutions

Coverage

TVS

Why add coverage?

- Conformance testing:
 - Need complete coverage of cases
- Targeting specific scenarios:
 - Hitting required corner cases
- Soak testing
 - Ensure testing is effective eg: not becoming repetitive

Test and Verification Solutions

Coverage Metrics

TVS

- Code Coverage**
 - Toggle coverage on interconnects and top level signals
 - Have all the blocks been wired up correctly?
- Functional Coverage**
 - Behaviour that can only be fully tested at top level
 - For example: power management
- Test case coverage**
 - Quality of architectural, conformance or soak testing
 - Measure by parsing tests or looking at simulation trace

Test and Verification Solutions

Adding Coverage by Parsing the Tests

TVS

```

graph TD
    Tests --> ITT[Instruction / transaction Trace]
    Tests --> ET[Execution Trace]
    ITT --> CM[Coverage Model: Parse & Decode]
    CM --> CBC[Coverage Base Classes: ISA view of resources]
    CBC --> CG[Coverage Grids]
    ET --> ISS[Instruction Set Simulator: ISS]
    ISS --> CD[Coverage Database]
    CD --> CG
    
```

Test and Verification Solutions

TVS

-

TVS

- ## Test and Verification Solutions

TVS

- (Verification is not just the responsibility of verification Engineers)*

TVS





TVS



- TVS

- ## Test and Verification Solutions

How to go faster!

Compute Farm, Emulators, FPGA and test chips

Test and Verification Solutions

The 'tradeoffs' for different platforms

	Compute farm	Emulator	FPGA	Test chip
Speed	10Hz - 100Hz ...per machine	1MHz	2MHz - 50MHz	GHz
Observability	Total	Trace window + host debug	Probes + host debug	Host debug
Behavioural testbench?	Yes	Co-emulation (speed penalty)	Co-emulation (speed penalty)	No
Test in 'real world' systems	No	Host debug + ICE with speed bridges	Mostly	Yes
Are fails easily reproducible in simulation?	Yes	Yes	No	No
Bring-up time	Minutes	Weeks → hours	Weeks → Days	Months


Complex timing dependencies

Depends on process maturity

Test and Verification Solutions

Summary

- What is SoC level verification? (Top v System)
- Looked at structure of a simple SoC
- Why do both 'SoC level' & 'unit level' verification?
- A methodology for SoC level verification
- System level verification



If time permits

- [RIS \(Random Instruction Stream\) Test Generators](#)
- [Looked at IP-XACT](#)

Test and Verification Solutions