Best Practices in Verification Planning
Benjamin Ehlers

Presented by
Paul Carzola – Senior Architect
Cadence Design Systems

Feb 2013
Discussion Topics:

- Verification planning problems
- Planning challenges to address
- Notion of Executable Verification Plan
- Recommended verification solution
- Example verification planning flow
- Value Analysis
About this Paper

• **Failure to Plan Means Planning to Fail**
• Verification planning, the most crucial step
  • Improves quality
  • Improves efficiency
• Methodology based on actual experiences on an SoC
• Includes strategy, planning, changes and closure
• Begins with culture of Metric Driven Verification
• Follows a well organized process
• Enhanced with the executable verification plan
Verification Planning Problems:

• Verification planning as an afterthought
• Lack of organized or standardized flow
• Absence of verification methodologies
• Lack of process for changing and evolving requirements
• Fear of verification planning costs
• Limitations in Technology
Planning Challenges to Address:

- Large scale distributed planning approach:
  - large team of engineers
  - variety of individual styles, priorities, experience
- Enable and maintain Consistency/Coherency
- Quality of design documentation and design requirements
- Frequent design changes
- Needs to support concept of Executable Verification Plan
Codified and Executable Verification Plan

- Single point of source across entire verification flow
- Defines the scope of verification
- Describes all key design features
- Supports link to the verification metrics
  - testcases, checkers, monitors, coverage, results
- Link to specifications
- Planning for the total verification project
  - complete and coherent verification flow
- Flexible for scripting

What makes a plan “executable”?

- The notion to define and refine your verification strategy
- The notion of collecting all run time results to make the plan dynamic and alive
How Executable Plan Recovers Cost of Up-Front Planning

- Testbench design begins in planning
  - Planned coverage, checks, tests cases provide guidance on testbench code and tests cases that need to be written
  - The final verification results and completions criteria are built into the executable plan
Executable Plan: Linking all the key elements

- Requirements
- Objectives
- Test Scenarios
- Distributed Team and Plans
- Specification
- The verification plan becomes the anchor to connect teams and technologies together
Best Practice Planning Methodology

• Successfully used at Freescale on a large SoC project
• With a widely distributed verification team

1. Create a Template
2. Initial Verification Planning
3. Implementation Planning
4. Closure Planning
5. Reviews/Results/Reporting

All very important aspects to the Planning Methodology
1. Create a Template:

- Enable and help ensure Consistency and Coherency from the beginning
- Provides a prescriptive approach to follow
- May seem trivial, unnecessary...
  - VITAL STEP
- Make sure template supports merging/importing multiple plans
  - distributed verification approach
- Include detailed explanation descriptions
### 2. Initial Verification Planning: Verification Needs

- Design documents
- Tools
- Models behavior or functional
- Design blocks that cannot be modeled
- Protocols used
- Testbench components
- Purchase vs Make components

---

**Plan**

**TEMPLATE**: Project Name – Chapter Name

1. Chapter Name Verification Plan
   1.1. Verification Planning Information
      1.1.1. Verification Needs
      1.1.2. Reuse Assessment
      1.1.3. Verification Assumptions
      1.1.4. Verification Scope Assessment
      1.1.5. Additional Information
      1.1.6. Reviews
   1.2. Features
      1.2.1. feature title 1
         1.2.1.1. testcase name 1
         1.2.1.2. monitor 1
         1.2.1.3. cover assertion 1
      1.2.2. feature title 2
         1.2.2.1. testcase name 2a
         1.2.2.2. testcase name 2b
         1.2.2.3. testcase name 2c
         1.2.2.4. check assertion 2
         1.2.2.5. cover assertion 2
2. Initial Verification Planning: Reuse Assessment

- Determine how much can be reused from previous projects
  - Test benches, checks, tests, VIP, etc…
  - Don’t under scope
- Team reuse and dynamics
  - What teams work well together vs new teams.
- Determine what needs to be developed
- Review existing documentation
- What designers are available for questions
2. Initial Verification Planning: Verification Assumptions

- Assumptions are all too often left unspoken leading to design features being misinterpreted
- Document and review any assumptions
- Review with designer early
- Have face-to-face meetings
- Document all findings
- E.g.
  - Feature X cannot be verified in digital, needs Analog Mixed Signal
  - System level signal needs to be modeled during SoC verification
- Checklist can be useful here
  - Some captured during post mortem time
  - Use action item system
- Make this a committed contract so there are no surprises
• Consider all the available capabilities available to you: directed test, constrained random, formal, checks, coverage properties, code coverage, etc…
• Methodology doc for how to choose.
• Iterative process
• Take inputs from reuse assessment
• Understand the cost to create new test benches
• Determine the skill set of the verification engineers
3. Implementation Planning: Plan Creation

- The essence of successful verification planning
- Verification must be driven by:
  - Requirements Driven – from design requirements
  - Feature Driven – from design specifications
  - Priority Driven – from the priority of Features
- Organize the verification plan according to design feature groups
- Testbench and stimulus can be extracted
3. Implementation Planning: Correlated Results

- Link to tests, checks, cover groups, code coverage, etc...
- Brings the planning to closure into full circle
4. Closure Planning

- Determine the signoff and completion criteria
- Don’t wait until the end
- Verification Completion Criteria must be based on metrics
  - Coverage metrics:
  - Executable verification plan includes direct coverage correlation

<table>
<thead>
<tr>
<th>Coverage Metric</th>
<th>70% Goal</th>
<th>60% Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master Mode</td>
<td>100%</td>
<td>60%</td>
</tr>
<tr>
<td>Slave Mode</td>
<td>100%</td>
<td>60%</td>
</tr>
<tr>
<td>Bus Arbitration</td>
<td>100%</td>
<td>60%</td>
</tr>
<tr>
<td>Transfer</td>
<td>100%</td>
<td>60%</td>
</tr>
<tr>
<td>Collision</td>
<td>100%</td>
<td>60%</td>
</tr>
<tr>
<td>Error</td>
<td>100%</td>
<td>60%</td>
</tr>
</tbody>
</table>
5. Plan for **Reviews/Results/Reporting**

- Plan for key milestone reviews and get commitment up front
  - If not, they don’t get done
- Need a consistent review flow and formats
- Executable verification plan
  - includes direct results correlation
- Report merge and roll-up

<table>
<thead>
<tr>
<th>Status</th>
<th>Test Name</th>
<th>Plan Name</th>
<th>Test Group</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>failed</td>
<td>master_transmit</td>
<td>Serial Interface</td>
<td>master_mode</td>
<td>comparison error! Expected data = 124, Actual Data = 127</td>
</tr>
<tr>
<td>failed</td>
<td>slave_transmit_1</td>
<td>Serial Interface</td>
<td>slave_mode</td>
<td>comparison error! Expected data = 148, Actual Data = 147</td>
</tr>
<tr>
<td>failed</td>
<td>master_mode_stall_detect</td>
<td>Serial Interface</td>
<td>stall_detection</td>
<td>FATAL ERROR! State machine in unknown state</td>
</tr>
<tr>
<td>failed</td>
<td>slave_transmit_2</td>
<td>Serial Interface</td>
<td>slave_mode</td>
<td>(None)</td>
</tr>
<tr>
<td>failed</td>
<td>slave_transmit_3</td>
<td>Serial Interface</td>
<td>slave_mode</td>
<td>(None)</td>
</tr>
<tr>
<td>failed</td>
<td>slave_transmit_4</td>
<td>Serial Interface</td>
<td>slave_mode</td>
<td>(None)</td>
</tr>
<tr>
<td>passed</td>
<td>slave_receive_1</td>
<td>Serial Interface</td>
<td>slave_mode</td>
<td>(None)</td>
</tr>
<tr>
<td>passed</td>
<td>slave_receive_2</td>
<td>Serial Interface</td>
<td>slave_mode</td>
<td>(None)</td>
</tr>
<tr>
<td>failed</td>
<td>master_receive</td>
<td>Serial Interface</td>
<td>master_mode</td>
<td>(None)</td>
</tr>
<tr>
<td>failed</td>
<td>slave_receive_3</td>
<td>Serial Interface</td>
<td>slave_mode</td>
<td>(None)</td>
</tr>
<tr>
<td>passed</td>
<td>slave_receive_4</td>
<td>Serial Interface</td>
<td>slave_mode</td>
<td>(None)</td>
</tr>
<tr>
<td>failed</td>
<td>master_transmit</td>
<td>Serial Interface</td>
<td>master_mode</td>
<td>Halt_mode: Unexpected transition of signal</td>
</tr>
<tr>
<td>passed</td>
<td>bridge_transmit_1</td>
<td>Core</td>
<td>bridge_gasket</td>
<td>(None)</td>
</tr>
<tr>
<td>passed</td>
<td>bridge_transmit_2</td>
<td>Core</td>
<td>bridge_gasket</td>
<td>(None)</td>
</tr>
<tr>
<td>passed</td>
<td>bus_error_detect</td>
<td>Core</td>
<td>bus_seo</td>
<td>(None)</td>
</tr>
</tbody>
</table>
Formalized Planning: Value

**Baseline Project**

- Duration - Months

**New Project with Advanced Planning**

- 10-20% Additional Planning Time Required
- 

**Overall Schedule Savings**

- 20% (1) Reporting Automation
- 10% (2) Closure Automation

**Importance of finding bugs early**

- Cost of Fixing Bugs:
  - x
  - 2x
  - 4x
  - 6x
  - 8x

**Verification Closure**

- Importance of finding bugs early

**Table: Cost of Fixing Bugs**

<table>
<thead>
<tr>
<th>Importance of finding bugs early</th>
<th>Cost of Fixing Bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1x</td>
</tr>
<tr>
<td>2x</td>
<td>2x</td>
</tr>
<tr>
<td>4x</td>
<td>4x</td>
</tr>
<tr>
<td>6x</td>
<td>6x</td>
</tr>
<tr>
<td>8x</td>
<td>8x</td>
</tr>
</tbody>
</table>
Summary

- Verification Planning is not simply a task that is done once and forgotten
- It is a living, breathing and executable methodology saving time and resources
- There is a cost but with a tremendous set of benefits
- Analysis and reporting is automated
- Saved 10%-40% in Freescale projects
Q/A

• Questions?