FPGA Verification using System Verilog and UVM
Verification vs Design Resource

Trend of Engineering headcount
- Mean peak number of engineers
- Compounded Annual Growth Rate

Figure 4. Mean Peak Number of Design and Verification Engineers working on a project
Verification Language Adoption Trend

Decrease Low Level Language
- Verilog
- VHDL

Increase High Level Language
- System Verilog
Unique main Actor: UVM
Universal Verification Methodology

Old Methodology story
System Verilog

High Level Language, new constructs, task and function which allow to development complex testbench with less and more concise code.

3 Main Verification Components Advantages

1) Stimulus Generation:
   New constructs Randomization more efficient
   New High level approach: OOP Approach

2) Functional Coverage
   New constructs for collect Functional Coverage

3) Response checking
   New constructs make Assertion more efficient
Universal Verification Methodology

This Methodology takes out the advantages of System Verilog by standardizing the test bench structure through a base classes library with defaults useful methods.

1) Reduces test bench development time
2) Makes porting and reuse in different project very simple

UVM Advantages:

• **High modularity and reusability**
  - by using a base classes library
  - hierarchy testbench structure of modular components with their own function.
  - UVM phasing

• **Efficient stimuli generation**
  - by using transaction level modelling
  - by separating tests from testbench

• **Chance to use DPI & VIP**
SKYTECHNOLOGY RECENT VERIFICATION PROJECTS

• UVM Test Bench
  
  Design: FPGA Stratix V Video acquisition and processing system
  
  Work: Development of entire UVM test bench
    - build of the environment
    - development of several Video protocol Agent
    - use of QVIP Menthor Graphics

• System Verilog Test Bench
  
  Design: FPGA Zynq Ultrascale Interrogator submodule system
  
  Work: Development of Testbench for Interfaces A429, MIL1553
    - Development of driver and generator classes for Random stimulus and error injection
    - Development of monitor and scoreboard classes for self check
System Verilog Test Bench

Wrapper

LPXU(DUT)

Driver/Monitor RTL

Driver/Monitor RTL

Driver

Monitor

Generator

Scoreboard

Arinc429

MIL-1553
Define the verification strategy
Develop a complete verification specification, test plan, and cover plan
Put in place a fully-scripted verification environment
Develop a high-level, self-checking, simulation test bench with functional coverage
Create a comprehensive test case suite to achieve full functional and code coverage
Reach coverage goal faster using constrained random verification
Set up and run gate-level simulations
We can take into account special requirement e.g. DO254, ISO26262