

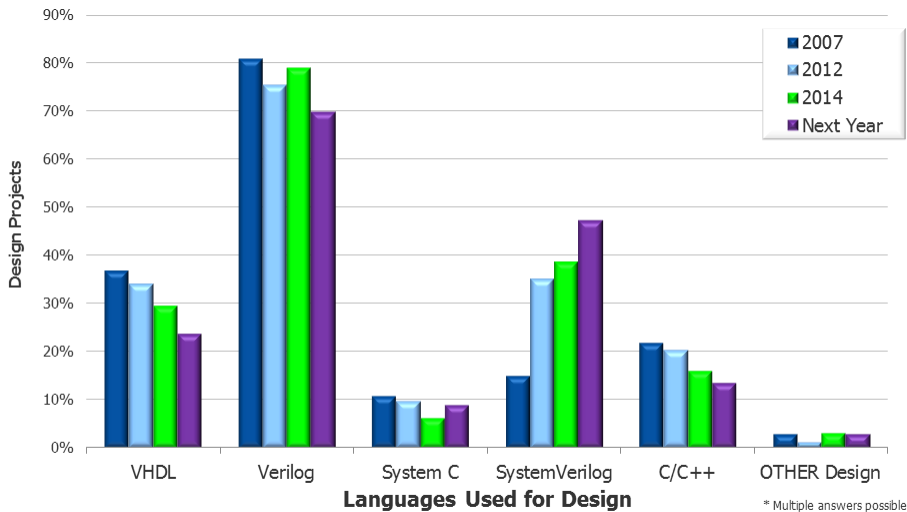
# UVM Framework

Introduction

Stefan Bauer  
FV Application Engineer



## Design Language Adoption Trends (ASIC)

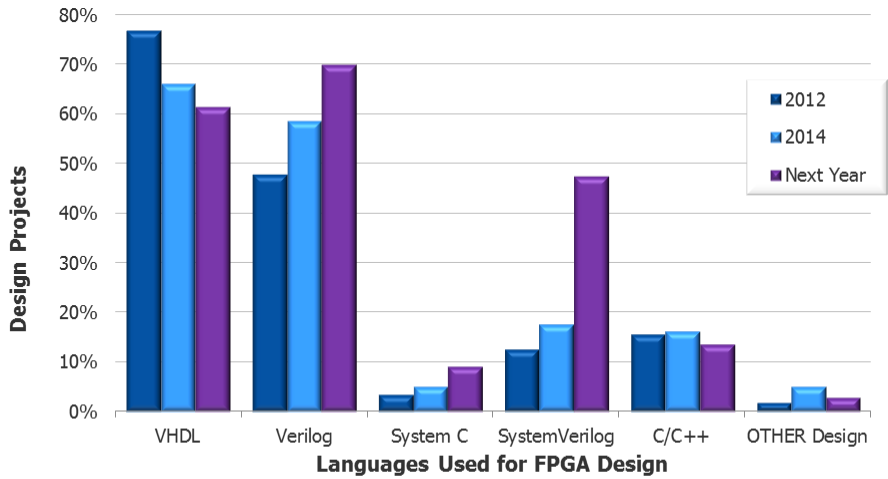


Source: Wilson Research Group and Mentor Graphics, 2014 Functional Verification Study

© Mentor Graphics Corp. Company Confidential  
www.mentor.com



## Design Language Adoption Trends (FPGA)



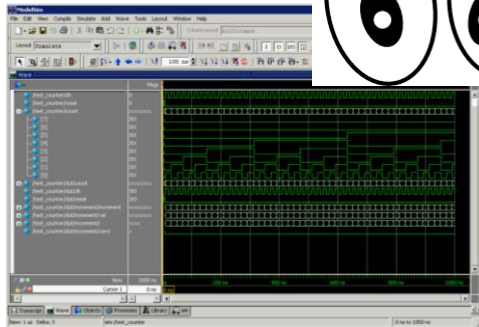
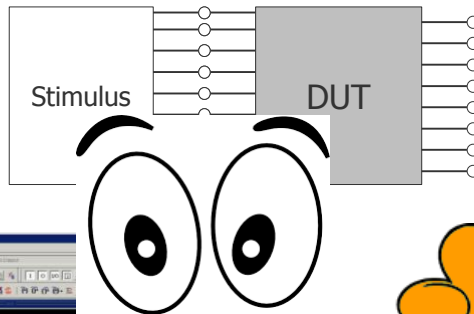
\* Multiple answers possible

Source: Wilson Research Group and Mentor Graphics, 2014 Functional Verification Study

© Mentor Graphics Corp. Company Confidential  
www.mentor.com



## Traditional Verification Flow



© Mentor Graphics Corp. Company Confidential  
www.mentor.com



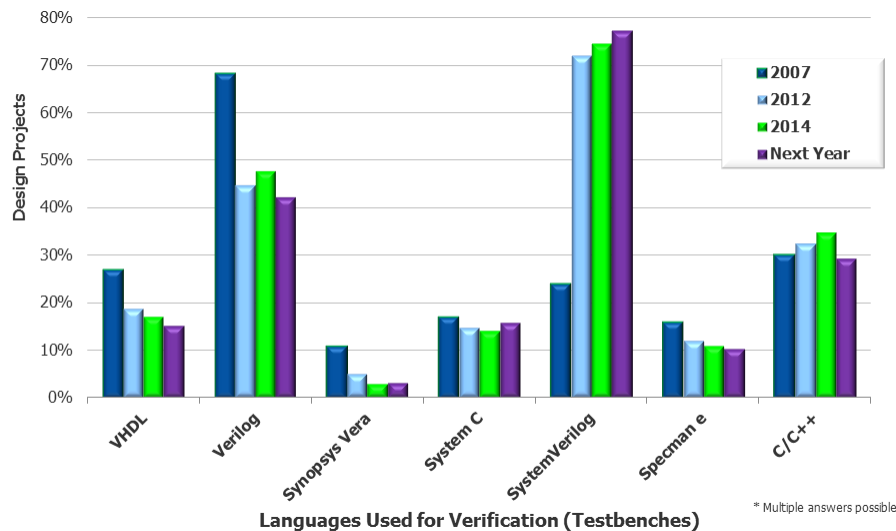
## Problems with the Traditional Approach

- The traditional approach has fundamental problems
  - Nothing is “really” being verified
  - Stimulus is limited to imagination/experience of designer
  - No documented “proof” of what parts of DUT has need tested/exercised
- It also has practical problems
  - Tests are hard or impossible to reuse
  - No standardized structure
    - Hard to maintain
    - Only the person who wrote it can understand it
- In the past, designs were simple enough that these problems were manageable or could just be ignored
- Not so any more...

5

© Mentor Graphics Corp. Company Confidential  
www.mentor.com

## Verification Language Adoption Trends (ASIC)

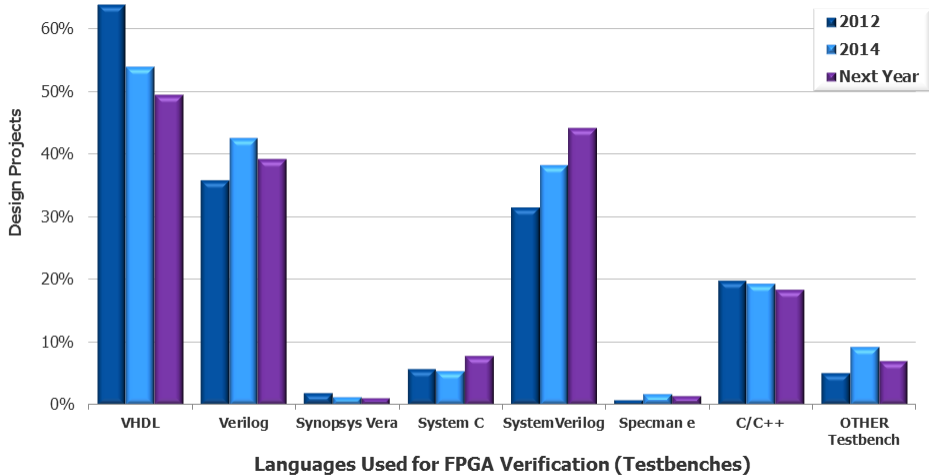


Source: Wilson Research Group and Mentor Graphics, 2014 Functional Verification Study

6

© Mentor Graphics Corp. Company Confidential  
www.mentor.com

## Verification Language Adoption Trends (FPGA)



Source: Wilson Research Group and Mentor Graphics, 2014 Functional Verification Study

7

© Mentor Graphics Corp. Company Confidential  
www.mentor.com



## SystemVerilog

- IEEE 1800-2012
- SystemVerilog provides technology (capability) that doesn't exist in Verilog or VHDL
  - Constraint random number generation (constraint solver)
  - Object oriented programming
  - Unified coverage database
  - Verification IPs available
- Learning SystemVerilog is like learning how to use the tools of a trade like carpentry



8

© Mentor Graphics Corp. Company Confidential  
www.mentor.com



## Problems with SystemVerilog

- Do you know how to build a house (testbench)
- We know in general what needs to be done
  - Generate stimulus, verify results, ...
- But how requires following proven methodologies else we get less than optimal results

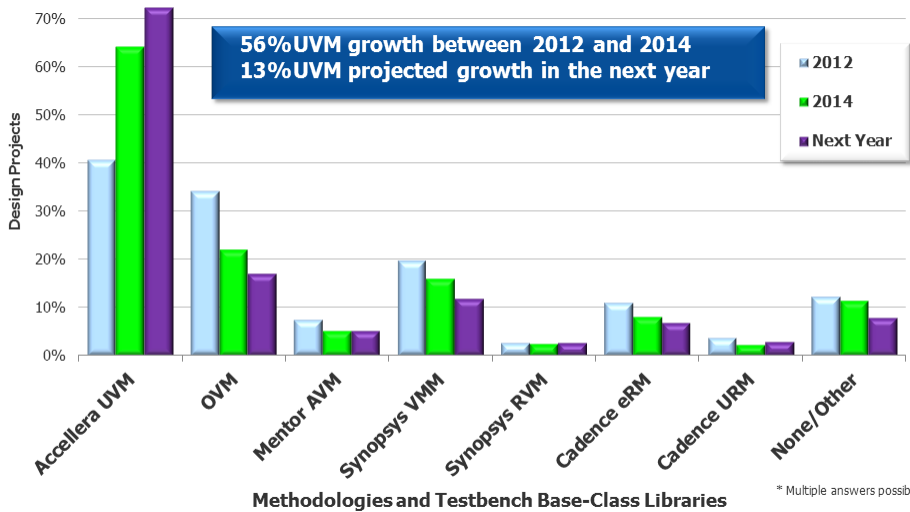


© Mentor Graphics Corp. Company Confidential  
www.mentor.com



9

## Testbench Methodology Adoption Trends (ASIC)

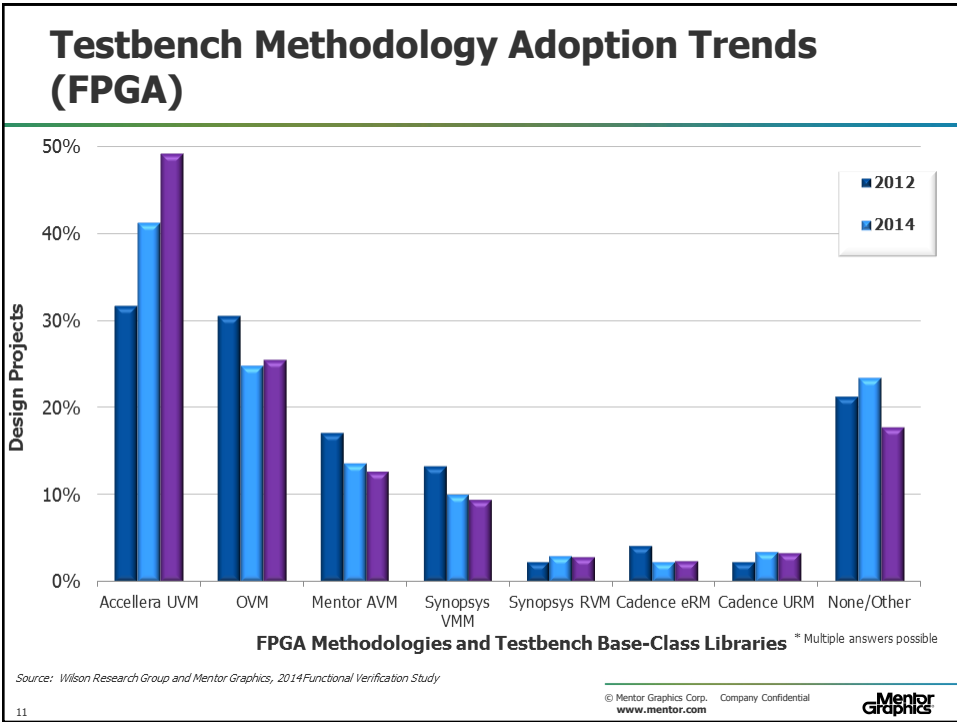


Source: Wilson Research Group and Mentor Graphics, 2014 Functional Verification Study

© Mentor Graphics Corp. Company Confidential  
www.mentor.com





10

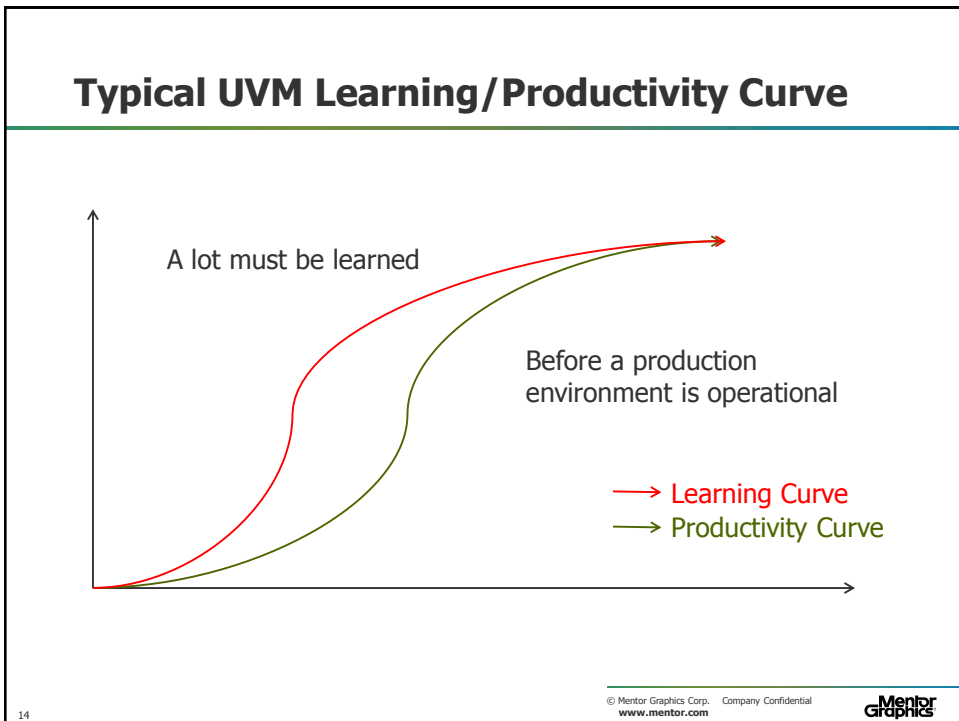
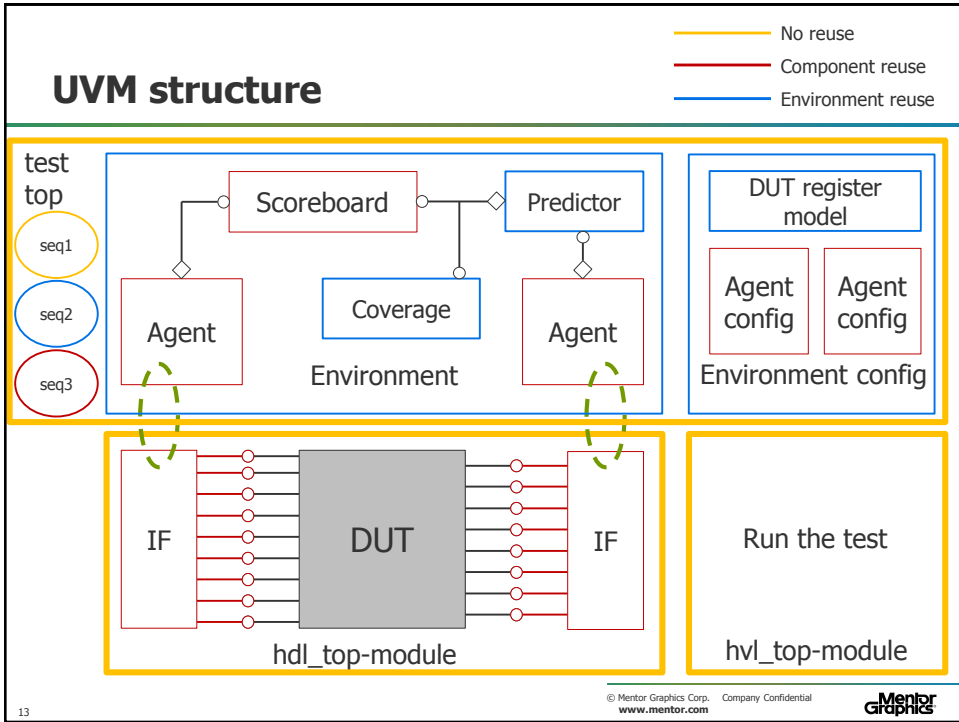


### Universal Verification Methodology

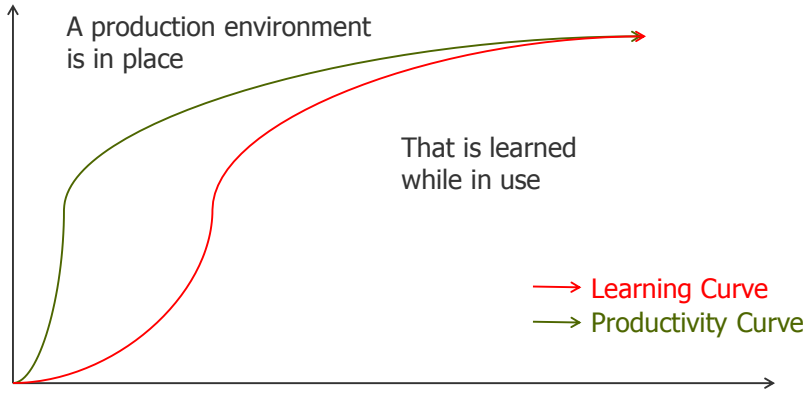
- UVM provides testbench methodology, i.e. how to use the SV tools to create effective testbenches
  - Testbenches that not only allow the application of SV technology but have
    - Reusability
    - Maintainability
    - Scalability

© Mentor Graphics Corp. Company Confidential www.mentor.com



## Preferred UVM Learning/Productivity Curve

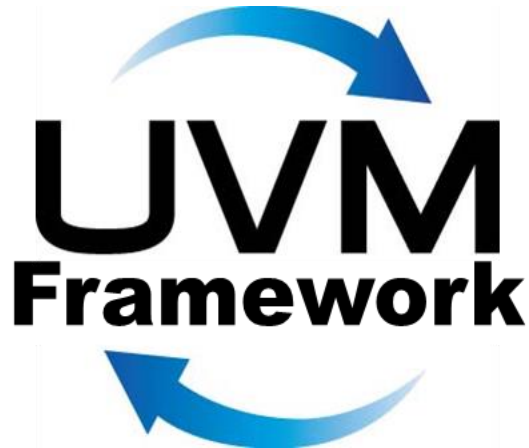


15

© Mentor Graphics Corp. Company Confidential  
www.mentor.com



## Mentor Graphics' solution



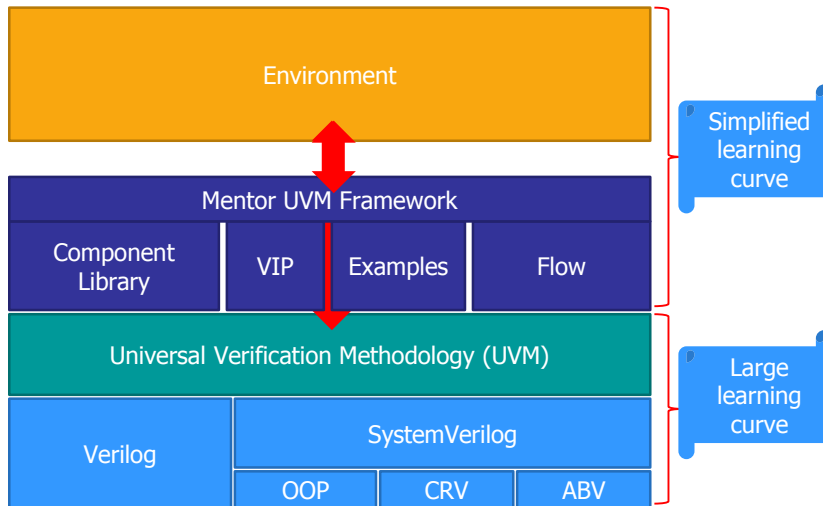
16

© Mentor Graphics Corp. Company Confidential  
www.mentor.com





## Learning Simplification



17

© Mentor Graphics Corp. Company Confidential  
www.mentor.com

## The UVM Framework

- **Class library**
  - Component base classes
  - Utility and convenience classes
  - Packages
- **Scripts**
  - Auto generation of components and testbench
  - Makefiles with common tool flow operations
- **Examples**
  - Block and chip level benches
  - Reuse
  - Technology integrations
- **Documentation**
  - Natural Docs
  - Diagrams

18

© Mentor Graphics Corp. Company Confidential  
www.mentor.com

## Key Characteristics of the UVM Framework

- Accelerate environment development
  - Better environments in less time
- Reusable
  - Across projects, sites, simulation levels
- Scalable
  - Across design size, complexity, and target technology
- Emulatable
  - Same environment and stimulus

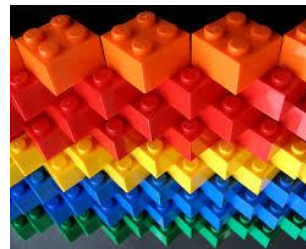


19

© Mentor Graphics Corp. Company Confidential  
www.mentor.comMentor  
graphics

## Technology integrated into UVM Framework

- Verification IP
  - Environments with AMBA and PCIe QVIP
- Graph based testing – Accelerated coverage closure
  - inFact testbench import to create protocol sequences
- Verification Management
  - Coverage ranking, merging, reporting
- Verification Run Manager
  - Automated regression management
- SystemVerilog Assistant
  - Templates for code generation
  - Visualization of UVM environments
- Emulation
  - TB structure is emulation compliant



20

© Mentor Graphics Corp. Company Confidential  
www.mentor.comMentor  
graphics

## UVM Framework's proven track record

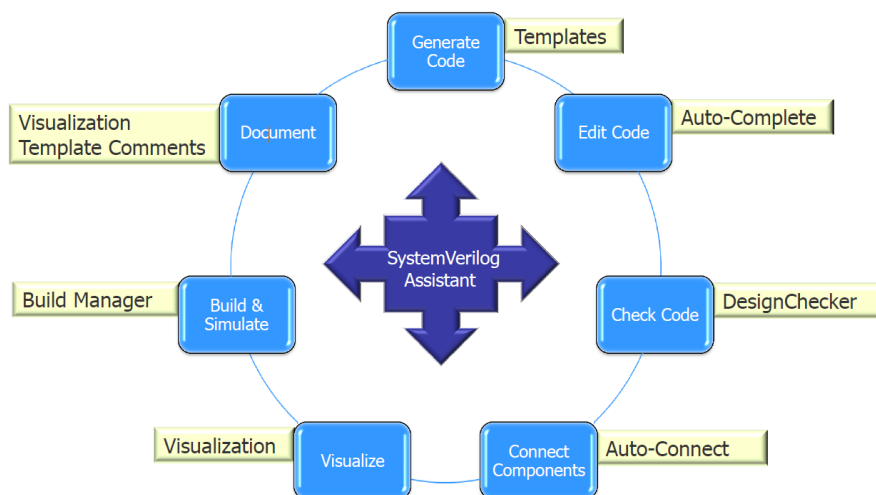
- Deployed on production designs at thirty companies
  - Across projects within companies
  - Across sites within companies
  - One DARPA SOC project spanned multiple companies
  - Used in many industries
  - Used on FPGA and ASIC
  - Used for Verilog/SV and VHDL designs
  - Used by UVM experts and novices



21

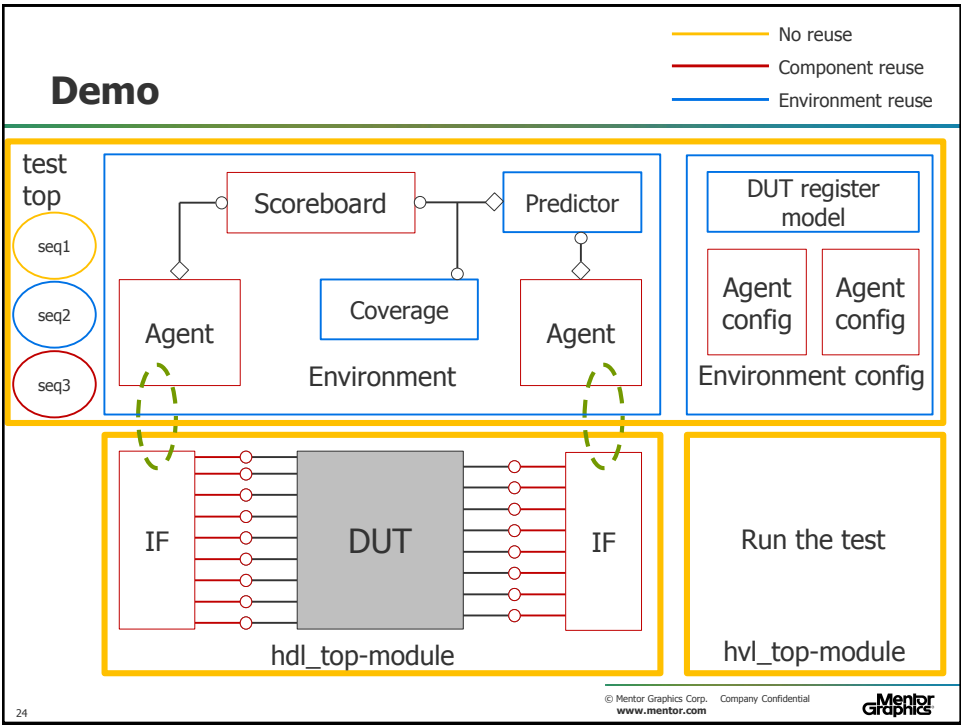
© Mentor Graphics Corp. Company Confidential  
www.mentor.com

## UVM Framework Overview



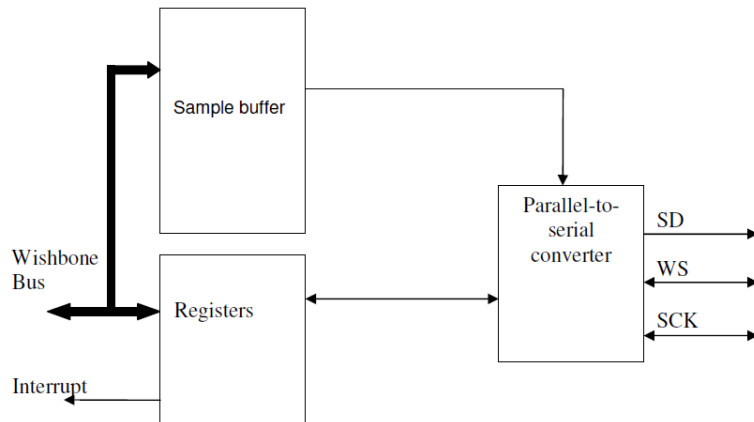
22

© Mentor Graphics Corp. Company Confidential  
www.mentor.com



## Demo

### ■ DUT: I<sup>2</sup>S Transmitter in slave mode



25

© Mentor Graphics Corp. Company Confidential  
www.mentor.com

## Testbench Infrastructure & Plumbing

```

UVMF
├── bench
│   ├── docs
│   ├── parameters
│   ├── registers
│   ├── rtl
│   ├── sequences
│   ├── sim
│   ├── testbench
│   └── tests
│       ├── complete_project_bench.svt
│       ├── environment
│       │   ├── complete_environment_package.svt
│       │   ├── environment_configuration.svt
│       │   ├── environment_environment.svt
│       │   └── environment_makefile_target.svt
│       └── interface
│           ├── complete_interface_package.svt
│           ├── interface_configuration.svt
│           ├── interface_driver.svt
│           ├── interface_driver_bfm.svt
│           ├── interface_driver_bfm_api.svt
│           ├── interface_filelist.svt
│           ├── interface_filelist_hdl.svt
│           ├── interface_filelist_hvl.svt
│           ├── interface_if.svt
│           ├── interface_makefile_target.svt
│           ├── interface_monitor.svt
│           ├── interface_monitor_bfm.svt
│           ├── interface_monitor_bfm_api.svt
│           ├── interface_pkg_hdl.svt
│           ├── interface_sequence_lib.svt
│           ├── interface_transaction.svt
│           ├── interface_transaction_coverage.svt
│           ├── interface_typedefs.svt
│           └── interface_typedefs_hdl.svt
  
```

- Generate interface (agents):
  - Transactions & transaction viewing setup
  - Sequences, UVM monitor & monitor BFM
  - Interface
  - Drivers
  - Configuration
- Generate the environment
  - Environment package
  - Environment with each agent instantiated
  - Configuration for each agent
- Generate the project testbench
  - Top level module
  - Parameter package
  - Top level sequence package
  - Top level test package

26

© Mentor Graphics Corp. Company Confidential  
www.mentor.com

## Templates

- Flexible capability that allows us to capture the UVMF process
- Attempt to automate as much as possible

**Template Parameters**  
Specify the variable values used by complete\_interface\_package.svt (changes will apply to file name and location using naming rules).

Template variables:  
Those marked \* are required and must have their final values entered.

Variable	Description	Value
*	Name of the new interface to be created. This value is also used in the default file and folder names. _pkg will be appended to create the interface package name.	specify %(fName) value
*	Transaction class variable types and names separated by commas.	bit,transVar1,integer,transVar2
*	Configuration class variable types and names separated by commas.	bit,cfgVar1,int,cfgVar2
*	Make this interface Veloce, emulation, ready?	no

File name and location

File name: verification\_ip/interface\_packages/%(fName)\_pkg/%(fName)\_pkg.sv

Location: F:\uvmf\_lab Browse...

File path: F:\uvmf\_lab\verification\_ip\interface\_packages/%(fName)\_pkg/%(fName)\_pkg.sv

Virtual folder: my\_project Browse...

Create sub virtual folder: verification\_ip/interface\_packages/%(fName)\_pkg

< Back Next > Finish Cancel

27

© Mentor Graphics Corp. Company Confidential  
www.mentor.com

## Summary

- Benefit from industry best practices
- Better environment in less time
- constraint random number generation with constraint solver
- Unified database
- Run testcases in parallel
- Verification IPs available
- Reusable across projects, sites, simulation levels

28

© Mentor Graphics Corp. Company Confidential  
www.mentor.com

## Free useful link

[www.verificationacademy.com](http://www.verificationacademy.com)

