

Building a better VHDL testing environment

Joren Guillaume

FEA
Ghent University

Preliminary Presentation

Outline

1 Introduction

- VHDL
- Testing VHDL

2 Proposed solution

- VHDL testing Framework
- Test Driven Development
- Utility library
- Script-based processing
- Continuous Integration

3 Concluding

- Problems
- Future work
- Demo



Outline

1 Introduction

- VHDL
- Testing VHDL

2 Proposed solution

- VHDL testing Framework
- Test Driven Development
- Utility library
- Script-based processing
- Continuous Integration

3 Concluding

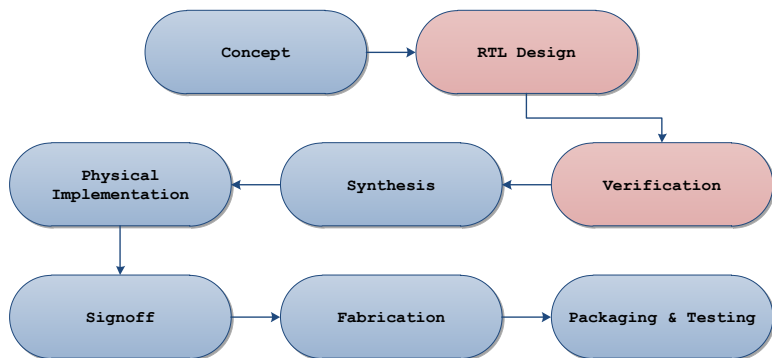
- Problems
- Future work
- Demo



VHDL

- VHSIC Hardware Description Language
- Used for describing digital and mixed-signal systems
- Developed by U.S. Department of Defense
 - ▶ Document → Simulate → Synthesize

VHDL - design flow



Outline

1 Introduction

- VHDL
- Testing VHDL

2 Proposed solution

- VHDL testing Framework
- Test Driven Development
- Utility library
- Script-based processing
- Continuous Integration

3 Concluding

- Problems
- Future work
- Demo



Testing VHDL

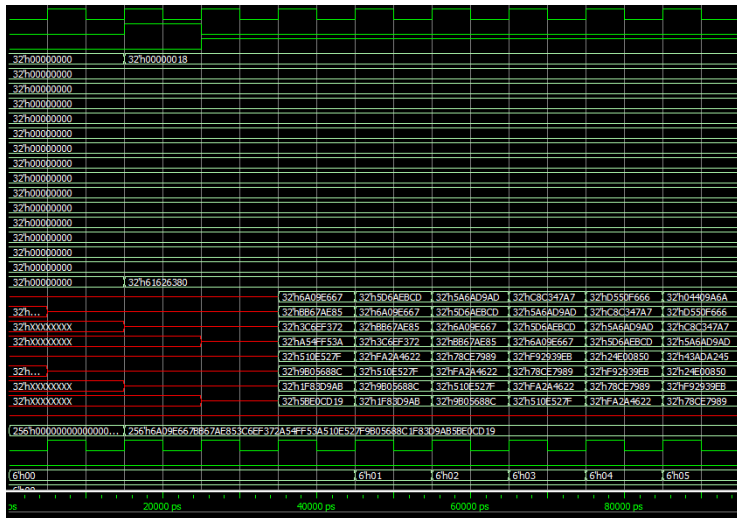
Testbenches

- Unit Under Test (UUT)
- Signal drivers, stimuli & processes
- Assertions and output tracking
 - ▶ Comparison to "golden reference"
 - ▶ Manual check
 - ▶ Wave-check

Problems

- Non-standardized process
- Single point of failure

Modelsim - waves



Outline

1 Introduction

- VHDL
- Testing VHDL

2 Proposed solution

- VHDL testing Framework
- Test Driven Development
- Utility library
- Script-based processing
- Continuous Integration

3 Concluding

- Problems
- Future work
- Demo



VHDL testing framework

Standardized testing framework

- Based on Test Driven Development (TDD)
- Cross platform
- Utility library
 - ▶ Standardized testbenches
 - ▶ Swift coding
- Script-based processing → Standardized processing & output
- Continuous Integration (CI) system

Outline

1 Introduction

- VHDL
- Testing VHDL

2 Proposed solution

- VHDL testing Framework
- **Test Driven Development**
- Utility library
- Script-based processing
- Continuous Integration

3 Concluding

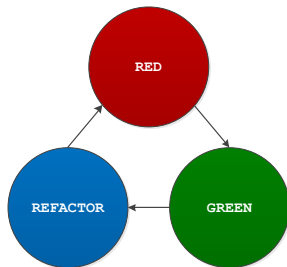
- Problems
- Future work
- Demo



Test Driven Development

Test Driven Development

- Software development technique
- Proven to significantly reduce errors
- All behaviour is tested
- Unit testing & short development cycle
- Red - Green - Refactor



Outline

1 Introduction

- VHDL
- Testing VHDL

2 Proposed solution

- VHDL testing Framework
- Test Driven Development
- **Utility library**
- Script-based processing
- Continuous Integration

3 Concluding

- Problems
- Future work
- Demo



Specialized library

Bitvis utility library

- Expands VHDL functions
 - ▶ Easy value checking
 - ▶ Clock & pulse generators
 - ▶ String handling & random generation
 - ▶ Easy output logging
- Quick & uniform coding
- Compatible with all VHDL versions



bitvis

Outline

1 Introduction

- VHDL
- Testing VHDL

2 Proposed solution

- VHDL testing Framework
- Test Driven Development
- Utility library
- **Script-based processing**
- Continuous Integration

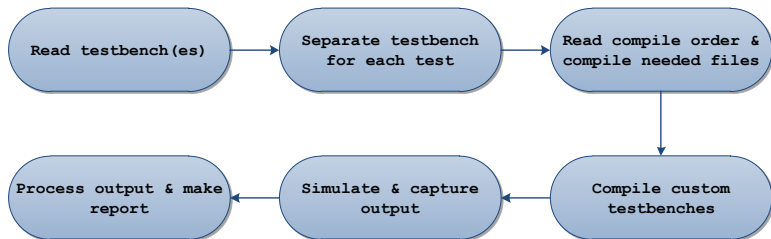
3 Concluding

- Problems
- Future work
- Demo



Script-based processing

Specialized python script



Script-based processing

Features

- Standalone functions
- Customizable process
- Text & JUnit format reports
- Automated file cleanup

Outline

1 Introduction

- VHDL
- Testing VHDL

2 Proposed solution

- VHDL testing Framework
- Test Driven Development
- Utility library
- Script-based processing
- **Continuous Integration**

3 Concluding

- Problems
- Future work
- Demo



Continuous Integration

Hudson-CI

- Centralized, automated testing
- Revision control integration (e.g. Git)
- Statistics on success
- Standardized test reports (JUnit)
- Very customizable

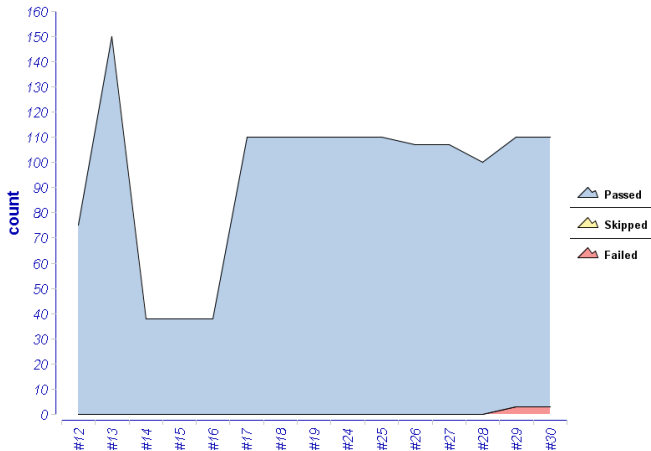


Hudson interface

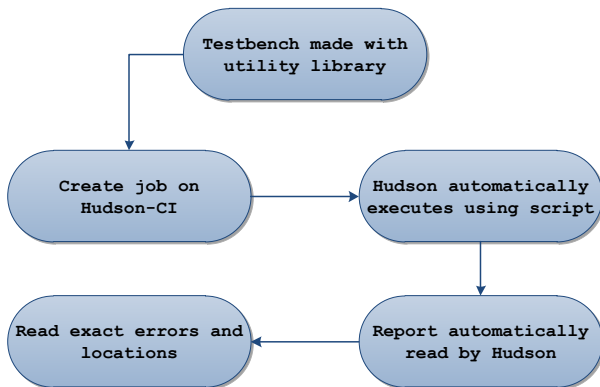
The screenshot shows the Hudson web interface. The top header has the Hudson logo, a search bar, and a link to 'ENABLE AUTO REFRESH'. The left sidebar contains navigation links: 'New Job', 'Manage Hudson', 'People', 'Build History', 'New View', 'Build Queue' (with 'No builds in the queue.'), and 'Build Executor Status' (showing 'Master 0/2' and 'Idle'). The main content area is titled 'Jobs Status' and features a table of build jobs. The table has columns for status (S/W), job name, last success/failure times, duration, and console links. There are also links for 'add description' and a tab selector for 'All' jobs.

S	W	Job ↓	Last Success	Last Failure	Last Duration	Console
		VHDL-AES	7 hr 38 min (#29)	1 mo 27 days (#16)	32 sec	
		VHDL-Bitvis	2 hr 27 min (#30)	7 hr 21 min (#23)	21 sec	
		VHDL-CRC	1 mo 24 days (#12)	1 mo 25 days (#11)	5,5 sec	
		VHDL-SHA	N/A	6 hr 58 min (#4)	5,4 sec	

Hudson statistics



Framework design flow



Outline

1 Introduction

- VHDL
- Testing VHDL

2 Proposed solution

- VHDL testing Framework
- Test Driven Development
- Utility library
- Script-based processing
- Continuous Integration

3 Concluding

- Problems
- Future work
- Demo



Problems

- VHDL has no reflection
 - ▶ Circumvent with higher level language
- Code duplication increases compile & simulation time
 - ▶ Implement regression testing
 - ▶ Smart code filtering
- Simulation is not synthesis
 - ▶ Wait statements, wrong sensitivity list ...

Outline

1 Introduction

- VHDL
- Testing VHDL

2 Proposed solution

- VHDL testing Framework
- Test Driven Development
- Utility library
- Script-based processing
- Continuous Integration

3 Concluding

- Problems
- **Future work**
- Demo



Future work

- Improving base script
 - ▶ Better integration utility library
 - ▶ More options
- Smart code analysis
- Regression testing
- Proper documentation & examples

Outline

1 Introduction

- VHDL
- Testing VHDL

2 Proposed solution

- VHDL testing Framework
- Test Driven Development
- Utility library
- Script-based processing
- Continuous Integration

3 Concluding

- Problems
- Future work
- Demo



Demo