

# Building a better VHDL testing environment

Joren Guillaume

FEA  
Ghent University

Thesis presentation

# Outline

- 1 **Situating**
  - Testing VHDL
  - Proposed solution
- 2 **The framework**
  - Utility library
  - Continuous Integration
  - Python script
  - Using the framework
- 3 **Concluding**
  - Results
  - Future work
- 4 **Demo**

# Outline

## 1 Situating

- Testing VHDL
- Proposed solution

## 2 The framework

- Utility library
- Continuous Integration
- Python script
- Using the framework

## 3 Concluding

- Results
- Future work

## 4 Demo

# Testing VHDL

## VHDL

- VHSIC Hardware Description Language
- Develop hardware
- Tested with test benches
  - ▶ Output tracking

## Problems

- Non-standardized process
- Single point of failure
- Time consuming

# Proposed solution

## Standardized testing framework

- Based on software development techniques
- Cross platform
- Utility library
- Continuous Integration system
- At the core: Python script

# Outline

- 1 Situating
  - Testing VHDL
  - Proposed solution
- 2 The framework
  - Utility library
  - Continuous Integration
  - Python script
  - Using the framework
- 3 Concluding
  - Results
  - Future work
- 4 Demo

# Utility library

## Bitvis utility library

- Compatible with all VHDL versions
- Expands VHDL functions
  - ▶ Easy value checking
  - ▶ Formatted output
- Quick & uniform coding
  - ▶ Reduces time spent coding
  - ▶ Improves readability



# Continuous Integration

## Hudson-CI

- Centralized, automated testing
  - ▶ Revision control
  - ▶ Timed building
- Easy to read results
- Very customizable



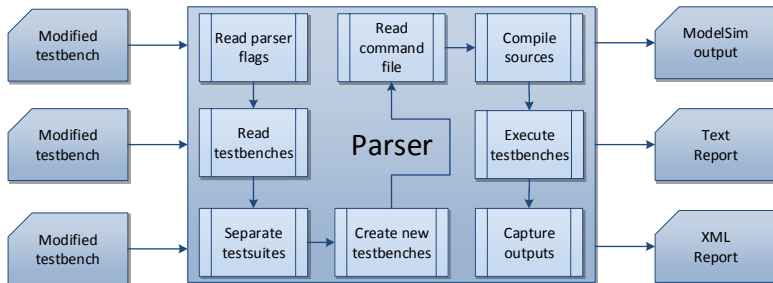


# Python script

## Test bench parser

- Customizable
  - ▶ Command-line arguments
- Separates independent tests
  - ▶ Remove single point of failure
- Multiple useful outputs
  - ▶ Processed text-based report
  - ▶ Processed XML report

# Script workflow



(Steps of the parser logged separately)

# Using the framework

## Preparing the test bench

- ① Import the utility library
- ② Decide on separation method
  - ▶ Line by line → No editing test bench
  - ▶ Start/Stop
  - ▶ Partitioned (recommended)
- ③ Create command file if needed

# Modified test bench example

## D flip-flop

- Old test bench:

```
assert q = '0'  
    report "Wrong output value at startup" severity FAILURE;  
d <= '1';  
WAIT FOR clk_period;  
assert q = '1'  
    report "Wrong output value at first test" severity FAILURE;
```

- Modified test bench:

```
—Test 1  
    check_value(q = '0', FAILURE, "Wrong output value at startup");  
    write(d, '1', "DFF");  
    check_value(q = '1', FAILURE, "Wrong output value at first test");  
    ...  
—End 1
```

# Using the framework

## Running Hudson-CI

- 1 Create new job
- 2 Optional: set for import from revision control source
- 3 Set correct parser flags in shell command
- 4 Build & check results

```
python src\testbench_parser.py -m partitioned -l sim\tb_dff_r.vhd
```

# Outline

- 1 Situating
  - Testing VHDL
  - Proposed solution
- 2 The framework
  - Utility library
  - Continuous Integration
  - Python script
  - Using the framework
- 3 Concluding
  - Results
  - Future work
- 4 Demo

# Results

- Multiple open-source projects converted

S	W	Job ↓	Last Success	Last Failure	Last Duration	Console	
		VHDL-AES	1 min 15 sec ( <a href="#">#30</a> )	3 mo 4 days ( <a href="#">#16</a> )	41 sec		
		VHDL-Bitvis	38 sec ( <a href="#">#35</a> )	1 mo 7 days ( <a href="#">#23</a> )	24 sec		
		VHDL-CRC	3 mo 1 day ( <a href="#">#12</a> )	1 min 3 sec ( <a href="#">#13</a> )	5,5 sec		
		VHDL-SHA	N/A	52 sec ( <a href="#">#5</a> )	6,2 sec		

- Bypassed single point of failure
- Automated testing = faster development

# Future work

- Wider, better tool support
- Lexical analysis
  - ▶ Automated partitioning
  - ▶ Smart test bench generation
- Adapted CI tool
  - ▶ Specific needs of hardware development



Thanks for your attention

Questions?

# Outline

- 1 Situating
  - Testing VHDL
  - Proposed solution
- 2 The framework
  - Utility library
  - Continuous Integration
  - Python script
  - Using the framework
- 3 Concluding
  - Results
  - Future work
- 4 Demo

# Demo