UNIVERSITEIT GENT

FACULTY OF ENGINEERING

Building a better VHDL testing environment

Joren Guillaume

FEA
Ghent University

Thesis presentation

Outline

1. Situating
   - VHDL
   - Testing and problems
   - Software development techniques
2. Proposed solution
   - VHDL testing framework
   - Using the framework
   - Automation
3. Concluding
   - Results
   - Future work
   - Conclusion
4. Demo

Outline

1. Situating
   - VHDL
   - Testing and problems
   - Software development techniques
2. Proposed solution
   - VHDL testing framework
   - Using the framework
   - Automation
3. Concluding
   - Results
   - Future work
   - Conclusion
4. Demo

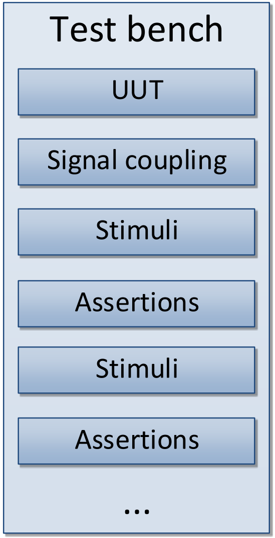
VHDL

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

Testing VHDL

Test benches

- Unit Under Test (UUT)
- Apply stimuli
- Signal/output tracking
  - Assertions
  - Comparison to desired result
  - Wave-check



Testing VHDL

Problems with testing

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

Software development techniques

Unit testing

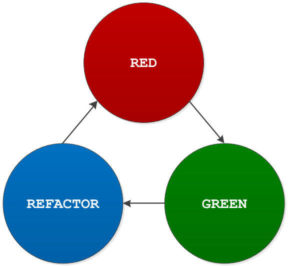
- Unit = smallest behaviour in code
- Test failure ➞ exact location

Test First Development

- Create test before the code
- How will the code behave?

Test Driven Development

- TFD & refactoring
- Proven to significantly reduce errors



Outline

1. Situating
   - VHDL
   - Testing and problems
   - Software development techniques
2. Proposed solution
   - VHDL testing framework
   - Using the framework
   - Automation
3. Concluding
   - Results
   - Future work
   - Conclusion
4. Demo

Proposed solution

Apply unit testing to VHDL?

→ 1 unit test per test bench

→ Extremely large number of TBs

→ Impractical, time wasting

Solution: use a framework to do it for you

VHDL testing framework

1. Arrange test bench into unit tests → Prepare
2. Separate unit tests into files
3. Compile source and newly created files

   } → Process
4. Execute unit tests ("test suite")
5. Capture and process results

   } → Execute

VHDL testing framework

Python script + ModelSim

- Prepare
  - → Developer arranges test suite
- Process
  - → Python script separates unit tests
  - → ModelSim compiles all files
- Execute
  - → ModelSim executes unit tests
  - → Python script reads output

Prepare
Process
Execute

Preparing test benches

- Separate independent tests
  - Line by line
  - Start/Stop
  - Partitioned
- Create commands file

```
...
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;
...
```

Utility library

Use Bitvis utility library for:

- Faster coding
- Improved readability

Modified test bench:

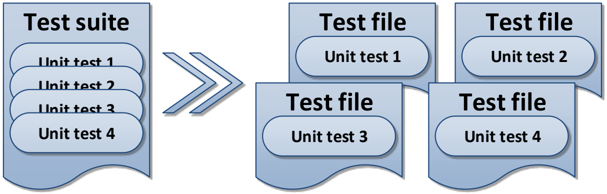
```
 ...
--Test 1
    check_value(q = '0', FAILURE, "Wrong output value at startup");
    write(d, '1', "DFF");
    check_value(q = '1', FAILURE, "Wrong output value at first test");
    ...
--End 1
 ...
```

Prepare → Process → Execute

Processing and compiling

Python script:

- Reads command line arguments
- Reads test suite
- Separates test suite into unit tests

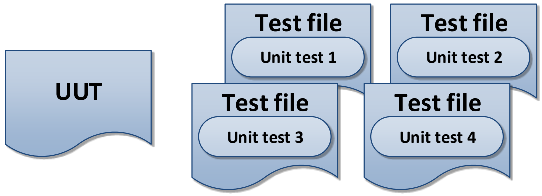


Prepare
Process
Execute

Processing and compiling

ModelSim:

- Compiles source code (UUT)
- Compiles test suite
  - → One entity, many architectures



Prepare
Process
Execute

Execution and results

ModelSim:

- Executes each unit test

Python script:

- Captures ModelSim output
- Processes results
  - Text report
  - JUnit XML report

Prepare
Process
Execute

Automation

Hudson-CI

- Gets latest version from RC
  - Timed retrieval
  - Detect changes
- Automated script execution
- Result progress (XML)

Outline

1 Situating
- VHDL
- Testing and problems
- Software development techniques

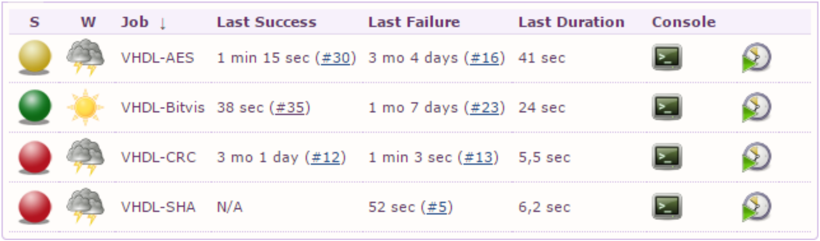
2 Proposed solution
- VHDL testing framework
- Using the framework
- Automation

3 Concluding
- Results
- Future work
- Conclusion

4 Demo

Results

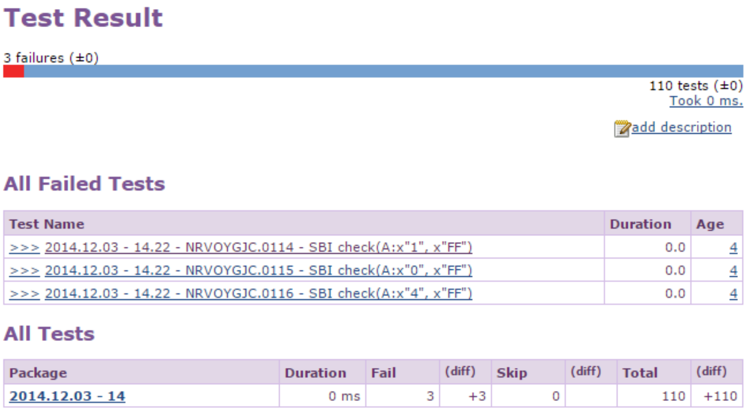
Multiple open-source projects tested



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

Results

Precise debugging information



Test Result

3 failures (±0)

110 tests (±0)
Took 0 ms.

add description

All Failed Tests

Test Name	Duration	Age
>>> 2014.12.03 - 14.22 - NRVOYGJC.0114 - SBI check(A:x"1", x"FF")	0.0	4
>>> 2014.12.03 - 14.22 - NRVOYGJC.0115 - SBI check(A:x"0", x"FF")	0.0	4
>>> 2014.12.03 - 14.22 - NRVOYGJC.0116 - SBI check(A:x"4", x"FF")	0.0	4

All Tests

Package	Duration	Fail	(diff)	Skip	(diff)	Total	(diff)
2014.12.03 - 14	0 ms	3	+3	0		110	+110

Future work

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

Conclusion

- Software methods are applicable if:
  - Tailored to development needs
  - Integrated with existing methods
- The framework provided:
  - Easier to read code
  - Precise debugging information
  - Eliminated single point of failure

End

Thanks for your attention!

Questions?

Outline

1 Situating
- VHDL
- Testing and problems
- Software development techniques

2 Proposed solution
- VHDL testing framework
- Using the framework
- Automation

3 Concluding
- Results
- Future work
- Conclusion

4 Demo

Demo

Demo