


UNIVERSITEIT GENT

FACULTY OF **ENGINEERING**

Building a better VHDL testing environment

Joren Guillaume

FEA
Ghent University

Thesis presentation

Outline

Outline

UNIVERSITEIT GENT

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

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

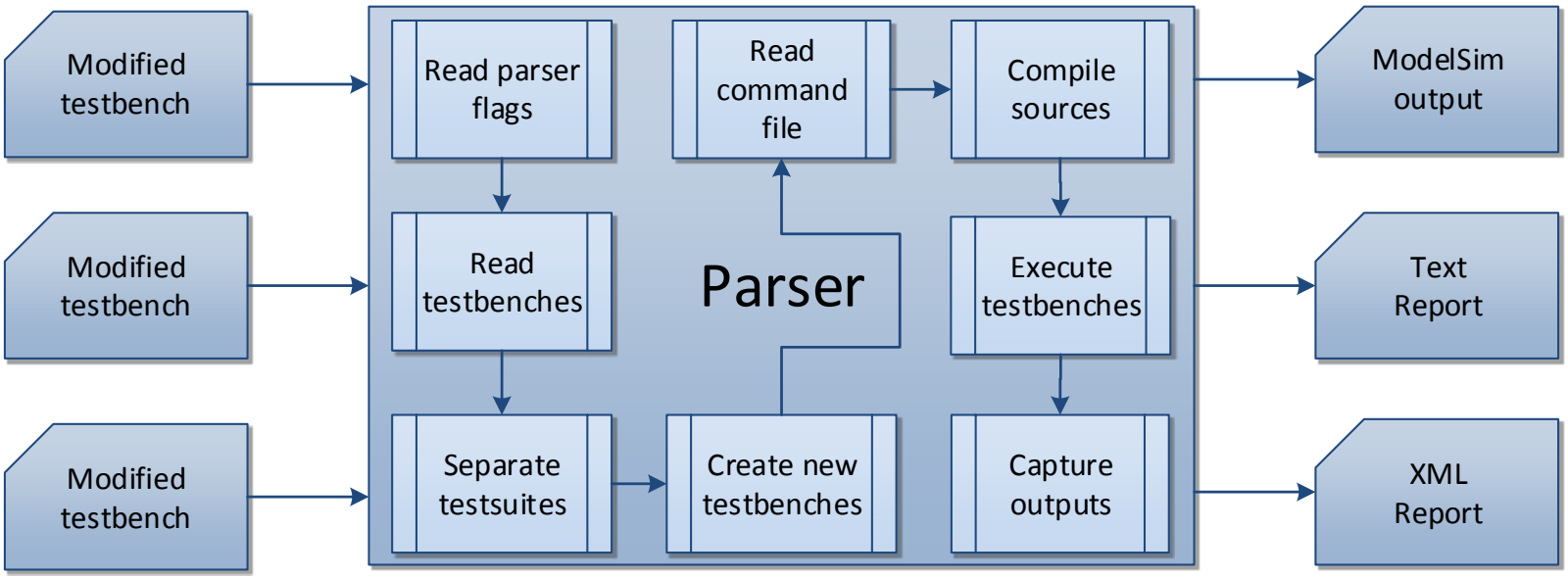
- Centralized, automated testing
  - Revision control
  - Timed building
- Easy to read results
- Very customizable

Python script

Test bench parser

- Separates independent tests
  - Remove single point of failure
- Customizable
  - Command-line arguments
- 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

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

UNIVERSITEIT GENT

Results

- Multiple open-source projects converted

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		

→ 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

Demo