Introducing hdlmake Version 2.0

Javier D. Garcia-Lasheras

Independent Consultant
Pamplona (SPAIN)

javier@garcialasheras.com

8th White Rabbit Workshop
CERN, October 2014

Outline

Outline

What is HDLMake?

What is HDLMake?

Purpose

HDLMake is a Python application targeted to make the life easier for the HDL developer.

What is HDLMake?

Purpose

HDLMake is a Python application targeted to make the life easier for the HDL developer.

Modular design

It allows the agile management and reuse of HDL cores:

What is HDLMake?

Purpose

HDLMake is a Python application targeted to make the life easier for the HDL developer.

Modular design

It allows the agile management and reuse of HDL cores:

- Uses a hierachy of Python Manifest.py files.

What is HDLMake?

Purpose

HDLMake is a Python application targeted to make the life easier for the HDL developer.

Modular design

It allows the agile management and reuse of HDL cores:

- Uses a hierachy of Python Manifest.py files.
- Supports both local and remote modules (SVN/GIT).

What is HDLMake?

Purpose

HDLMake is a Python application targeted to make the life easier for the HDL developer.

Modular design

It allows the agile management and reuse of HDL cores:

- Uses a hierachy of Python Manifest.py files.
- Supports both local and remote modules (SVN/GIT).

Command line helper

It provides a set of functionalitites targeted to scripting:

What is HDLMake?

Purpose

HDLMake is a Python application targeted to make the life easier for the HDL developer.

Modular design

It allows the agile management and reuse of HDL cores:

- Uses a hierachy of Python Manifest.py files.
- Supports both local and remote modules (SVN/GIT).

Command line helper

It provides a set of functionalitites targeted to scripting:

- Project creation/update for selected FPGA synthesis tools.

What is HDLMake?

Purpose

HDLMake is a Python application targeted to make the life easier for the HDL developer.

Modular design

It allows the agile management and reuse of HDL cores:

- Uses a hierachy of Python Manifest.py files.
- Supports both local and remote modules (SVN/GIT).

Command line helper

It provides a set of functionalitites targeted to scripting:

- Project creation/update for selected FPGA synthesis tools.
- Makefile generation for both Synthesis and Simulation.

What is HDLMake?

Purpose

HDLMake is a Python application targeted to make the life easier for the HDL developer.

Modular design

It allows the agile management and reuse of HDL cores:

- Uses a hierachy of Python Manifest.py files.
- Supports both local and remote modules (SVN/GIT).

Command line helper

It provides a set of functionalitites targeted to scripting:

- Project creation/update for selected FPGA synthesis tools.
- Makefile generation for both Synthesis and Simulation.
- Many others...

HDLMake evolution

HDLMake evolution

Version 1.0 (a.k.a. ISYP)

HDLMake evolution

Version 1.0 (a.k.a. ISYP)

- Targeted to Xilinx ISE and Modelsim.

HDLMake evolution

Version 1.0 (a.k.a. ISYP)

- Targeted to Xilinx ISE and Modelsim.
- Very stable and well documented.

HDLMake evolution

Version 1.0 (a.k.a. ISYP)

- Targeted to Xilinx ISE and Modelsim.
- Very stable and well documented.
- Still used by most of the White Rabbit projects at OHR.

HDLMake evolution

Version 1.0 (a.k.a. ISYP)

- Targeted to Xilinx ISE and Modelsim.
- Very stable and well documented.
- Still used by most of the White Rabbit projects at OHR.

The Master branch

HDLMake evolution

Version 1.0 (a.k.a. ISYP)

- Targeted to Xilinx ISE and Modelsim.
- Very stable and well documented.
- Still used by most of the White Rabbit projects at OHR.

The Master branch

- Added Quartus project generation and ISim support.

HDLMake evolution

Version 1.0 (a.k.a. ISYP)

- Targeted to Xilinx ISE and Modelsim.
- Very stable and well documented.
- Still used by most of the White Rabbit projects at OHR.

The Master branch

- Added Quartus project generation and ISim support.
- Very unstable/buggy and not documented.

HDLMake evolution

Version 1.0 (a.k.a. ISYP)

- Targeted to Xilinx ISE and Modelsim.
- Very stable and well documented.
- Still used by most of the White Rabbit projects at OHR.

The Master branch

- Added Quartus project generation and ISim support.
- Very unstable/buggy and not documented.
- New command line syntax and Manifest.py variables.

HDLMake evolution

Version 1.0 (a.k.a. ISYP)

- Targeted to Xilinx ISE and Modelsim.
- Very stable and well documented.
- Still used by most of the White Rabbit projects at OHR.

The Master branch

- Added Quartus project generation and ISim support.
- Very unstable/buggy and not documented.
- New command line syntax and Manifest.py variables.

Version 2.0 (a.k.a. 2014)

HDLMake evolution

Version 1.0 (a.k.a. ISYP)

- Targeted to Xilinx ISE and Modelsim.
- Very stable and well documented.
- Still used by most of the White Rabbit projects at OHR.

The Master branch

- Added Quartus project generation and ISim support.
- Very unstable/buggy and not documented.
- New command line syntax and Manifest.py variables.

Version 2.0 (a.k.a. 2014)

- This is where we are now!

Outline

Documentation: goals

Documentation: goals

Goals for the HDLMake v2.0 documentation

A good quality documentation must be a fundamental piece for this release:

Documentation: goals

Goals for the HDLMake v2.0 documentation

A good quality documentation must be a fundamental piece for this release:

- Usefull and handy.

Documentation: goals

Goals for the HDLMake v2.0 documentation

A good quality documentation must be a fundamental piece for this release:

- Usefull and handy.
- Targeted to users and developers.

Documentation: goals

Goals for the HDLMake v2.0 documentation

A good quality documentation must be a fundamental piece for this release:

- Usefull and handy.
- Targeted to users and developers.
- Synchronized with the code!!

Documentation: Sphinx

Documentation: Sphinx

http://sphinx-doc.org/

Documentation: Sphinx

http://sphinx-doc.org/

- Sphinx was originally created for the new Python documentation.

Documentation: Sphinx

http://sphinx-doc.org/

- Sphinx was originally created for the new Python documentation.
- C/C++ is already supported and it is planned to add special support for other languages as well.

Documentation: Sphinx

http://sphinx-doc.org/

- Sphinx was originally created for the new Python documentation.
- C/C++ is already supported and it is planned to add special support for other languages as well.
- Multiple output formats supported: HTML, LaTeX, PDF, ePub, Texinfo, man pages...

Documentation: Sphinx

http://sphinx-doc.org/

- Sphinx was originally created for the new Python documentation.
- C/C++ is already supported and it is planned to add special support for other languages as well.
- Multiple output formats supported: HTML, LaTeX, PDF, ePub, Texinfo, man pages...
- Sphinx uses reStructuredText as its markup language.

Documentation: Sphinx

http://sphinx-doc.org/

- Sphinx was originally created for the new Python documentation.
- C/C++ is already supported and it is planned to add special support for other languages as well.
- Multiple output formats supported: HTML, LaTeX, PDF, ePub, Texinfo, man pages...
- Sphinx uses reStructuredText as its markup language.
- Extensive cross-references, code handling, API autodocumentation...

Documentation: Read the Docs

Documentation: Read the Docs

http://www.readthedocs.org http://www.rtfd.org

Documentation: Read the Docs

http://www.readthedocs.org http://www.rtfd.org

- Read the Docs hosts documentation for the open source community.

Documentation: Read the Docs

http://www.readthedocs.org http://www.rtfd.org

- Read the Docs hosts documentation for the open source community.
- It supports Sphinx docs written with reStructuredText.

Documentation: Read the Docs

http://www.readthedocs.org http://www.rtfd.org

- Read the Docs hosts documentation for the open source community.
- It supports Sphinx docs written with reStructuredText.
- It can pull from your Subversion, Bazaar, Git, and Mercurial repositories.

Documentation: Read the Docs

http://www.readthedocs.org http://www.rtfd.org

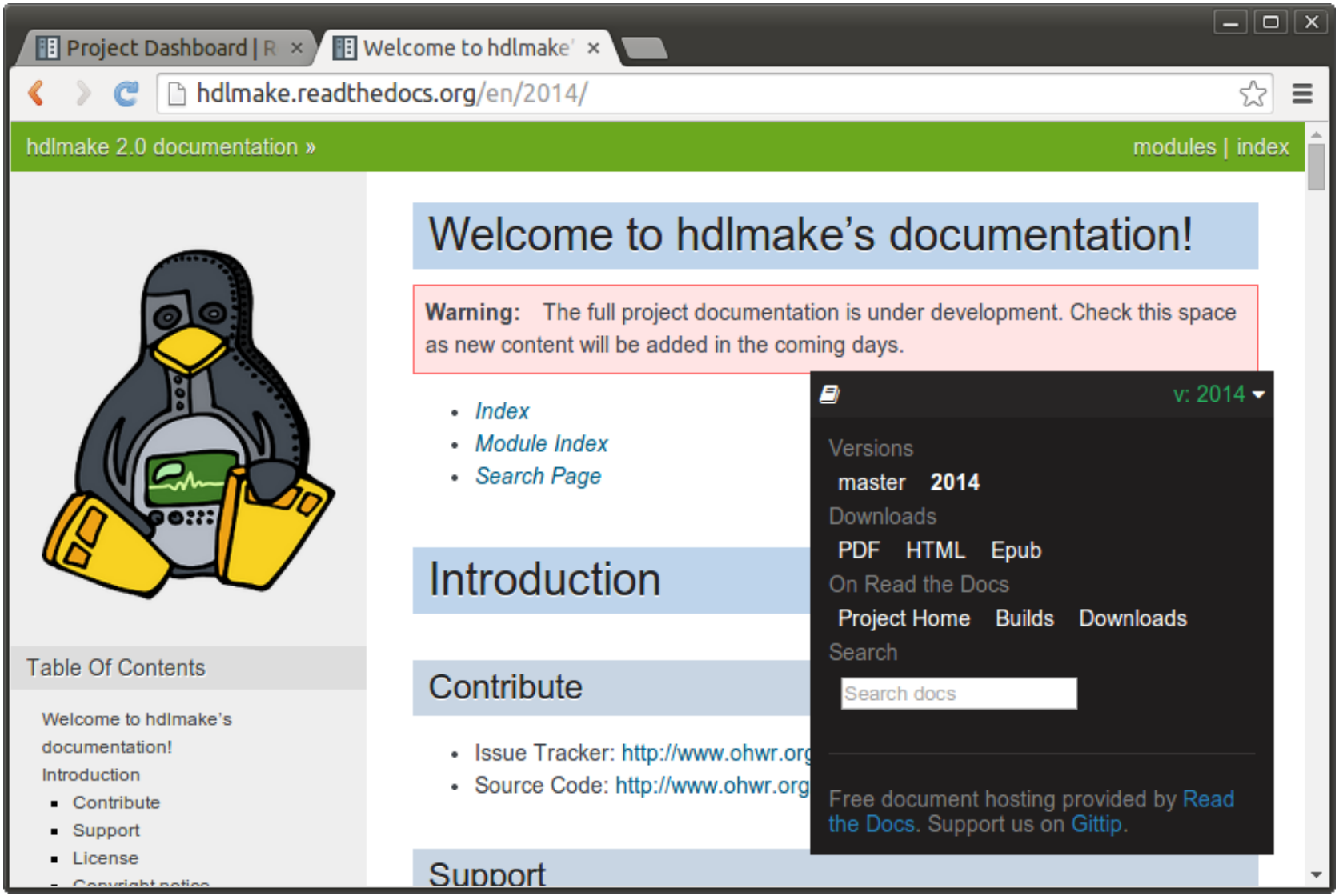
- Read the Docs hosts documentation for the open source community.
- It supports Sphinx docs written with reStructuredText.
- It can pull from your Subversion, Bazaar, Git, and Mercurial repositories.
- It automatically generates browsable HTML, pdf, e-pub...

Documentation: Read the Docs

http://www.readthedocs.org http://www.rtfd.org

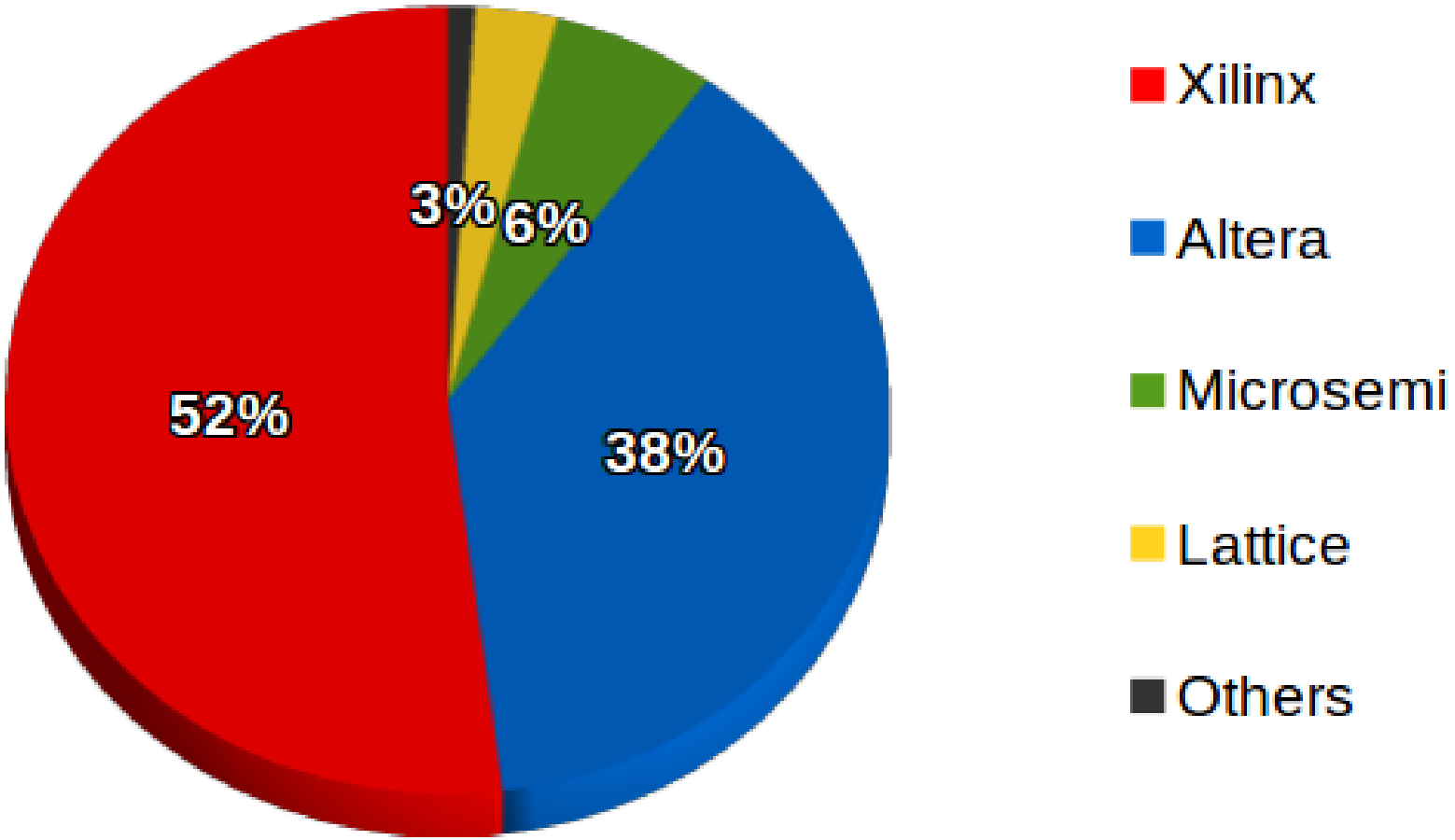
- Read the Docs hosts documentation for the open source community.
- It supports Sphinx docs written with reStructuredText.
- It can pull from your Subversion, Bazaar, Git, and Mercurial repositories.
- It automatically generates browsable HTML, pdf, e-pub...
- It is written in Python, hosted in Github and released under MIT license.

Documentation: screenshot



Outline

FPGA Market Share by Vendor



Supported Synthesis Tools

Supported Synthesis Tools

Xilinx

- ISE
- PlanAhead

Supported Synthesis Tools

Xilinx

- ISE
- PlanAhead ...supports the Zynq family. Check: http://www.ohwr.org/projects/evo

Supported Synthesis Tools

Xilinx

- ISE
- PlanAhead ...supports the Zynq family. Check: http://www.ohwr.org/projects/evo

Altera

- Quartus II

Supported Synthesis Tools

Xilinx

- ISE
- PlanAhead ...supports the Zynq family. Check: http://www.ohwr.org/projects/evo

Altera

- Quartus II

Microsemi

- Libero IDE / SoC

Supported Synthesis Tools

Xilinx

- ISE
- PlanAhead ...supports the Zynq family. Check: http://www.ohwr.org/projects/evo

Altera

- Quartus II

Microsemi

- Libero IDE / SoC

Lattice

- Diamond IDE

Supported Simulation Tools

Supported Simulation Tools

Proprietary (but can be free for entry level)

Supported Simulation Tools

Proprietary (but can be free for entry level)

- Xilinx Isim

Supported Simulation Tools

Proprietary (but can be free for entry level)

- Xilinx Isim
- Mentor Graphics Modelsim (Altera and Microsemi edition)

Supported Simulation Tools

Proprietary (but can be free for entry level)

- Xilinx Isim
- Mentor Graphics Modelsim (Altera and Microsemi edition)
- Aldec Active-HDL (Lattice edition)

Supported Simulation Tools

Proprietary (but can be free for entry level)

- Xilinx Isim
- Mentor Graphics Modelsim (Altera and Microsemi edition)
- Aldec Active-HDL (Lattice edition)

FOSS

Supported Simulation Tools

Proprietary (but can be free for entry level)

- Xilinx Isim
- Mentor Graphics Modelsim (Altera and Microsemi edition)
- Aldec Active-HDL (Lattice edition)

FOSS

- Icarus Verilog (Verilog support only)

Supported Simulation Tools

Proprietary (but can be free for entry level)

- Xilinx Isim
- Mentor Graphics Modelsim (Altera and Microsemi edition)
- Aldec Active-HDL (Lattice edition)

FOSS

- Icarus Verilog (Verilog support only)
- GHDL (VHDL support only)

Supported Operating Systems

Supported Operating Systems

Linux

Supported Operating Systems

Linux

- Tested on Ubuntu 12.04 LTS Precise / 14.04 LTS Trusty

Supported Operating Systems

Linux

- Tested on Ubuntu 12.04 LTS Precise / 14.04 LTS Trusty
- Tested on CentOS 6 / 7 (Binary compatible with RHEL)

Supported Operating Systems

Linux

- Tested on Ubuntu 12.04 LTS Precise / 14.04 LTS Trusty
- Tested on CentOS 6 / 7 (Binary compatible with RHEL)
- Should work on any Linux distro including Python2.7!

Supported Operating Systems

Linux

- Tested on Ubuntu 12.04 LTS Precise / 14.04 LTS Trusty
- Tested on CentOS 6 / 7 (Binary compatible with RHEL)
- Should work on any Linux distro including Python2.7!

Windows

Supported Operating Systems

Linux

- Tested on Ubuntu 12.04 LTS Precise / 14.04 LTS Trusty
- Tested on CentOS 6 / 7 (Binary compatible with RHEL)
- Should work on any Linux distro including Python2.7!

Windows

- Most of the FPGA tools are only supported in Windows!

Supported Operating Systems

Linux

- Tested on Ubuntu 12.04 LTS Precise / 14.04 LTS Trusty
- Tested on CentOS 6 / 7 (Binary compatible with RHEL)
- Should work on any Linux distro including Python2.7!

Windows

- Most of the FPGA tools are only supported in Windows!
- HDLMake has been modified to allow full compatibility with Cygwin environments.

Supported Operating Systems

Linux

- Tested on Ubuntu 12.04 LTS Precise / 14.04 LTS Trusty
- Tested on CentOS 6 / 7 (Binary compatible with RHEL)
- Should work on any Linux distro including Python2.7!

Windows

- Most of the FPGA tools are only supported in Windows!
- HDLMake has been modified to allow full compatibility with Cygwin environments.
- HDLMake over Cygwin has been tested on Windows 7 / 8 / 8.1

Outline

The counter test

The counter test

Purpose

The counter test

Purpose

- it serves as a template set for HDLMake newcomers.

The counter test

Purpose

- it serves as a template set for HDLMake newcomers.
- it checks that every supported tool is working properly.

The counter test

Purpose

- it serves as a template set for HDLMake newcomers.
- it checks that every supported tool is working properly.
- it is delivered into the hdlmake source tree.

The counter test

Purpose

- it serves as a template set for HDLMake newcomers.
- it checks that every supported tool is working properly.
- it is delivered into the hdlmake source tree.

The counter test root folder

```
user@host:~\$ tree -d -L 1 counter/
counter/
|-- modules
|-- sim
|-- syn
|-- testbench
`-- top
```

A very simple HDLMake module

A very simple HDLMake module

Overview of the counter modules

```
user@host:~\$ tree counter/modules/
counter/modules/
‘-- counter
    |-- verilog
    | |-- counter.v
    | ‘-- Manifest.py
    ‘-- vhdl
        |-- counter.vhd
        ‘-- Manifest.py
```

A very simple HDLMake module

Overview of the counter modules

```
user@host:~\$ tree counter/modules/
counter/modules/
‘-- counter
    |-- verilog
    | |-- counter.v
    | ‘-- Manifest.py
    ‘-- vhdl
        |-- counter.vhd
        ‘-- Manifest.py
```

The simplest Manifest.py for VHDL

```
files = ["counter.vhd",]
```

A very simple testbench

A very simple testbench

The counter testbench folder

```
user@host:~\$ tree counter/testbench/
counter/testbench/
‘-- counter_tb
    |-- verilog
    | |-- counter_tb.v
    | ‘-- Manifest.py
    ‘-- vhdl
        |-- counter_tb.vhd
        ‘-- Manifest.py
```

A very simple testbench

The counter testbench folder

```
user@host:~\$ tree counter/testbench/
counter/testbench/
‘-- counter_tb
    |-- verilog
    | |-- counter_tb.v
    | ‘-- Manifest.py
    ‘-- vhdl
        |-- counter_tb.vhd
        ‘-- Manifest.py
```

The VHDL Manifest.py for counter testbench

```
files = ["counter_tb.vhd",]
modules = {"local":["../../../modules/counter/vhdl"],}
```

Covered simulation tools

Covered simulation tools

The simulation folder

```
user@host:~\$ tree -d -L 1 counter/sim
counter/sim
|-- aldec
|-- ghdl
|-- isim
|-- iverilog
`-- modelsim
```

Inside a simulation folder

Inside a simulation folder

e.g. the modelsim folder

```
user@host:~\$ tree counter/sim/modelsim/
counter/sim/modelsim/
|-- verilog
|  `-- Manifest.py
|-- vhdl
|  `-- Manifest.py
`-- vsim.do
```

Running a simulation

Running a simulation

e.g. Modelsim simulation top Manifest.py for VHDL

```
action = "simulation"
sim_tool = "modelsim"
top_module = "counter_tb"
sim_post_cmd = "vsim -do ../vsim.do -i counter_tb"
modules = {
    "local" : ["../../../testbench/counter_tb/vhdl"],
}
```

Running a simulation

e.g. Modelsim simulation top Manifest.py for VHDL

```
action = "simulation"
sim_tool = "modelsim"
top_module = "counter_tb"
sim_post_cmd = "vsim -do ../vsim.do -i counter_tb"
modules = {
    "local" : ["../../../testbench/counter_tb/vhdl"],
}
```

Common simulation command sequence!

```
user@host:~\$ cd counter/sim/modelsim/vhdl
user@host:~\$ hdlmake
user@host:~\$ make sim_post_cmd
```

Real boards under test

Real boards under test

Sample FPGA boards for testing

```
user@host:~\$ tree -d -L 1 counter/top
counter/top
|-- brevia2_dk
|-- cyclone3_sk
|-- proasic3_sk
`-- spec_v4
```

A constrained top design

A constrained top design

e.g. the SPEC top board folder

```
user@host:~\$ tree counter/top/spec_v4/
counter/top/spec_v4/
|-- spec_top.ucf
|-- verilog
| |-- Manifest.py
| `-- spec_top.v
`-- vhdl
    |-- Manifest.py
    `-- spec_top.vhd
```

A constrained top design

e.g. the SPEC top board folder

```
user@host:~\$ tree counter/top/spec_v4/
counter/top/spec_v4/
|-- spec_top.ucf
|-- verilog
| |-- Manifest.py
| `-- spec_top.v
`-- vhdl
    |-- Manifest.py
    `-- spec_top.vhd
```

The VHDL Manifest.py for a top board

```
files = ["spec_top.vhd", "../spec_top.ucf"]
modules = {"local":["../../../modules/counter/vhdl"],}
```

Covered synthesis tools

Covered synthesis tools

The synthesis folder

```
user@host:~\$ tree -d -L 1 counter/syn
counter/syn
|-- brevia2_dk_diamond
|-- cyclone3_sk_quartus
|-- proasic3_sk_libero
|-- spec_v4_ise
`-- spec_v4_planahead
```

Inside a synthesis folder

Inside a synthesis folder

e.g. SPEC v4 ISE folder

```
user@host:~\$ tree -d -L 1 counter/syn/spec_v4_ise
counter/syn/spec_v4_ise/
|-- verilog
| `-- Manifest.py
`-- vhdl
    `-- Manifest.py
```

Running a synthesis

Running a synthesis

e.g. SPEC v4 synthesis Manifest.py with ISE

```
target = "xilinx"
action = "synthesis"
syn_device = "xc6slx45t"
syn_grade = "-3"
syn_package = "fgg484"
syn_top = "spec_top"
syn_project = "demo.xise"
syn_tool = "ise"
modules = {"local":["../../../top/spec_v4/vhdl"],}
```

Running a synthesis

e.g. SPEC v4 synthesis Manifest.py with ISE

```
target = "xilinx"
action = "synthesis"
syn_device = "xc6slx45t"
syn_grade = "-3"
syn_package = "fgg484"
syn_top = "spec_top"
syn_project = "demo.xise"
syn_tool = "ise"
modules = {"local":["../../../top/spec_v4/vhdl"],}
```

Synthesis command sequence!

```
user@host:~\$ cd counter/syn/spec_v4_ise/vhdl
user@host:~\$ hdlmake
user@host:~\$ make
```

Outline

Conclusions

Only one global idea

We have now a stable enough base from which we can start mantaining HDLMake in the long term:

Conclusions

Only one global idea

We have now a stable enough base from which we can start mantaining HDLMake in the long term:

- Agile documentation mechanism.

Conclusions

Only one global idea

We have now a stable enough base from which we can start mantaining HDLMake in the long term:

- Agile documentation mechanism.
- Modular multitool support.

Conclusions

Only one global idea

We have now a stable enough base from which we can start mantaining HDLMake in the long term:

- Agile documentation mechanism.
- Modular multitool support.
- A first step towards Quality-Assurance by testing.

Conclusions

Only one global idea

We have now a stable enough base from which we can start mantaining HDLMake in the long term:

- Agile documentation mechanism.
- Modular multitool support.
- A first step towards Quality-Assurance by testing.

Keep in touch

If you are interested in knowing more about HDLMake, just take a look to its OHR project site:

▸ http://www.ohwr.org/projects/hdl-make

THANK YOU!!