How to use the DeSCAM plugin PrintChisel

Author: Lukas Krupp

Contents

1 Install Chisel **1**
1.1 Development Tools . . . 1
1.1.1 Install Java . . . 1
1.1.2 Install sbt (Scala Build Tool) . . . 1
1.1.3 Install Verilator . . . 1
1.2 Chisel Tutorial . . . 1
1.2.1 Getting the repository . . . 2
1.2.2 Using Chisel . . . 2
1.3 Links . . . 2

2 Chisel Projects **2**
2.1 Create a Project . . . 2
2.1.1 Manual . . . 2
2.1.2 Automatic . . . 3
2.2 Using IntelliJ IDEA . . . 3
2.3 Links . . . 3

3 Use PrintChisel **3**
3.1 Workflow . . . 3
3.2 Open Issues . . . 4

1 Install Chisel

1.1 Development Tools

1.1.1 Install Java

```
sudo apt-get install default-jdk
```

1.1.2 Install sbt (Scala Build Tool)

```
echo "deb https://dl.bintray.com/sbt/debian /" | sudo tee -a /etc/apt/sources.list.d/sbt.list
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 642AC823
sudo apt-get update
sudo apt-get install sbt
```

1.1.3 Install Verilator

1. Install prerequisites:

```
sudo apt-get install git make autoconf g++ flex bison
```

2. Clone repository:

```
git clone http://git.veripool.org/git/verilator
```

3. Check out a known good version (Currently recommended: 4.006):

```
git pull
git checkout verilator_4_006
```

4. In the Verilator repository directory, build and install:

```
unset VERILATOR_ROOT # For bash, unsetenv for csh
autoconf # Create ./configure script
./configure
make
sudo make install
```

1.2 Chisel Tutorial

The tutorial teaches how to use the above tools to compile Chisel code and generate hardware.

1.2.1 Getting the repository

```
git clone https://github.com/ucb-bar/chisel-tutorial.git
cd chisel-tutorial
git fetch origin
git checkout release
```

1.2.2 Using Chisel

1. Run sbt:

```
sbt
```

2. Run the first example (In tutorial folder):

```
test:runMain problems.Launcher Mux2
```

1.3 Links

For further details on:

- Installing dev. tools: `https://github.com/freechipsproject/chisel3`
- Setting up tutorial: `https://github.com/ucb-bar/chisel-tutorial`
- Generation of Verilog and simulation files from Chisel code: `https://github.com/ucb-bar/chisel-tutorial/wiki/the-basics`

2 Chisel Projects

2.1 Create a Project

2.1.1 Manual

1. Get the project template:

```
mkdir ~/ChiselProjects
cd ~/ChiselProjects

git clone https://github.com/ucb-bar/chisel-template.git MyChiselProject
cd MyChiselProject
```

2. Clear out old stuff:

```
rm -rf .git
git init
git add .gitignore *
```

3. Rename project in build.sbt file:
Change the line `name := "chisel-module-template"` in file `build.sbt` to `name := "my-chisel-project"`.

2.1.2 Automatic

In the script found in folder `\PrintChisel_Tutorial\scripts\` change the path of the Chisel project directory to the path on your machine and then execute:

```
sudo ./create-chisel-project.sh PROJECT_FOLDER_NAME PACKAGE_NAME
```

2.2 Using IntelliJ IDEA

Many Chisel developers use Jetbrain's IntelliJ IDEA as an IDE to write Chisel code. The following link shows how to install the tool and import your above created projects: `https://github.com/freechipsproject/chisel-template/wiki/IntelliJ-Installation-Guide`

2.3 Links

For further details on:

- Creating Chisel projects: `https://github.com/freechipsproject/chisel-template`
- Generating Verilog from a project: `https://github.com/freechipsproject/chisel3/wiki/Frequently-Asked-Questions#get-me-verilog`

3 Use PrintChisel

3.1 Workflow

1. Invoke plugin by executing DeSCAM with option `-PrintChisel`
2. The tool generates following files (if ESL description is named e.g. `MyDesign.h`):
   - `MyDesign.scala`
   - `MyDesign_functions.scala`
   - `MyDesign.sva`
   - `MyDesign_functions.sva`
   - `types.sva`
   - `ipc.sva`
3. Create a Chisel project by calling `sudo .\create-chisel-project.sh MY_DESIGN mydesign`
4. Copy `MyDesign.scala` and `MyDesign_functions.scala` into folder `\MY_DESIGN\src\main\scala mydesign\`
5. In a terminal from the folder `\MY_DESIGN` invoke `sbt 'runMain mydesign.MyDesign'` to generate Verilog (`MyDesign.v`)

Examples of Chisel projects created following the above workflow can be found in folder `\PrintChisel_Tutorial\examples\`.
Important: Look at 3.2 before working with generated hardware!

3.2 Open Issues

- The automatic refinement (function `autorefine()` in PrintChisel plugin) of the state macros in the SVA file is not yet implemented and has to be performed manually before checking the design with OneSpin.