CREEP: Chalmers RTL Energy Evaluation for Pipelines

Daniel Moreau

May 29, 2016

Contents

1 Introduction

Welcome to Chalmers RTL based Energy Evaluation for Pipelines abbreviated CREEP. This user manual will provide an overview of the framework and guide the user on how to install and use the framework.

2 Installation

The framework is provided as a tar.gz file and should be unpacked to a location chosen by the user. The following command will extract the framework to the current directory:

```
# tar -zxvf CREEP.tar.gz
```

The default framework folder structure immediately after unpacking is showed in figure 1. The contents of each folder is elaborated on below:

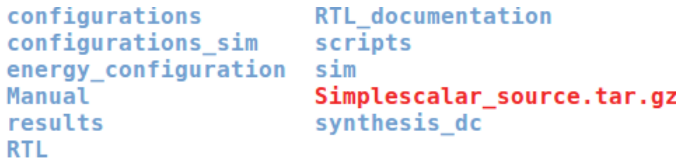

Figure 1: Structure of the CREEP package after unpacking

- **configurations** - pipeline configurations shipped with the framework.
- **configurations_sim** - statistics generated from Simplescalar.
- **energy_configurations** - energy extracted from the pipeline RTL.
- **results** - final energy estimation output of the framework.
- **RTL** - pipeline RTL.
- **RTL_documentation** - documentation of the RTL code.
- **scripts** - automating scripts.
- **sim** - RTL simulation output.
- **synthesis_dc** - synthesis output.

In addition to the aforementioned folders the Simplescalar simulator is also shipped with the framework in an uncompiled and compressed format (*Simplescalar_source.tar.gz*)

2.1 Simulator

The Simplescalar simulator and cross-compiler need to be extracted to the top CREEP directory, the following command achieves this:

```
# tar -zxvf Simplescalar_source.tar.gz
```

A shell script *build.sh* available inside the extracted Simplescalar directory automates the installation and is executed by issuing:

```
# sh build.sh
```

Note: The simulator has been verified to build on Linux Red hat 6.5 (Santiago).

Warning: The installation of the simulator will take several minutes.

If the script executes without error, the simulator and cross-compiler are now ready to be used in the framework workflow. However, the path to the simulator binaries must be appended to the system path variable (contained in the simplesim-3.0 sub directory) before running the framework. If this is not done, the system will not find the sim-outorder simulator binary when attempting to execute the Mibench benchmarks. The following command achieves this:

```
# PATH=$PATH:/chalmers/users/tester1/Desktop/CREEP/Simplescalar/simplesim-3.0
```

2.1.1 Benchmarks

The Mibench benchmark suite is included in the framework and integrates well with the framework workflow. Essentially any benchmark suite that compiles for the Simplescalar simulator (using the provided cross-compiler) can be used. However, should another suite be used, the CREEP.pl script (see section 3) needs to be changed to accommodate this.

Some minor adjustments to the Makefile located in the Mibench folder are necessary. Modify the first line in the file:

```
# CREEP_PATH=/
```

to the path where CREEP is located on the current machine, e.g.,

```
# CREEP_PATH=/chalmers/users/tester1/Desktop/CREEP
```

The benchmark suite is now ready to be built, issue the following command:

```
# make compile_gcc
```

2.2 RTL

The RTL flow of the framework is optional and reserved for advanced users that have access to EDA tools that support:

- Compiling RTL.
- Directed RTL simulation and verification.
- Synthesis.
- Directed netlist simulation and verification.
- Power extraction.

The framework is shipped with scripts that automate the entire RTL flow. The scripts are built on the following EDA tools:

- Cadence Incisive Enterprise Simulator - Compiling, simulating and verifying RTL.
- Synopsys Design Compiler - Synthesis.
- Cadence Incisive Enterprise Simulator - simulating and verifying netlist.
- Synopsys Prime Time - Power extraction.

Should the user have access to these tools, the RTL flow is almost complete out of the box. To complete the RTL flow the user must change the paths to the cell libraries in the synthesis script in the *scripts* folder. Furthermore, the flow relies on EEMBC benchmarks which due to license issues could not be shipped with the framework. The following EEMBC benchmarks were used:

- Autocor
- viterbi

- rgbcmy
- fft
- conven

Optimally the same benchmarks should be used as this reduces the need for modification of the workflow. However, essentially all embedded benchmarks can be used. Regardless of benchmark, it needs to be compiled with a fixed starting address for it to integrate into the RTL simulation and verification. If a different set of benchmarks is used the RTL and netlist verification flow needs to be modified and the CREEP.pl script needs to be changed to reflect this.

3 Use

The functionality of the framework is made easily accessible through the CREEP.pl script in the scripts folder (see figure 1). The script is run with the following syntax:

```
# ./CREEP.pl -[flag] -[config]
```

Flags:

-sim: launches the simplescalar simulator with a configuration defined in the provided configuration file. Produces a simulation statistics file in the *configuration_sim* folder and loads a previously generated power report (with the same name as the provided configuration) in the *configuration_energy* folder. By combining the generated statistics and the loaded power values a final energy estimation file generated and stored in the *results* folder.

-rtl: Used by advanced users that have access to appropriate EDA tools. Launches the RTL with a configuration defined in the provided configuration file. Produces a power report in the *configuration_energy* folder with the same name as the provided configuration file.

-f: A combination of aforementioned flags. Launches the entire framework workflow starting with the optional rtl workflow and then the simulator workflow. Produces all the aforementioned files.

Config:

The provided configuration files can be found in the *configuration* folder in the top CREEP directory (see figure 1). The configuration file is parsed and

applied to the RTL and Simplescalar configuration file (config subdirectory in Mibench directory, see figure 1).