Simple Guide for NVDLA Hardware Integrator

Chen Tinghuan

April 17, 2019

1 Environment Setup

note: I use linux9 to run this code, please don't use linux2-4.

The big picture about NVDLA Hardware integrator is shown in Fig.??:

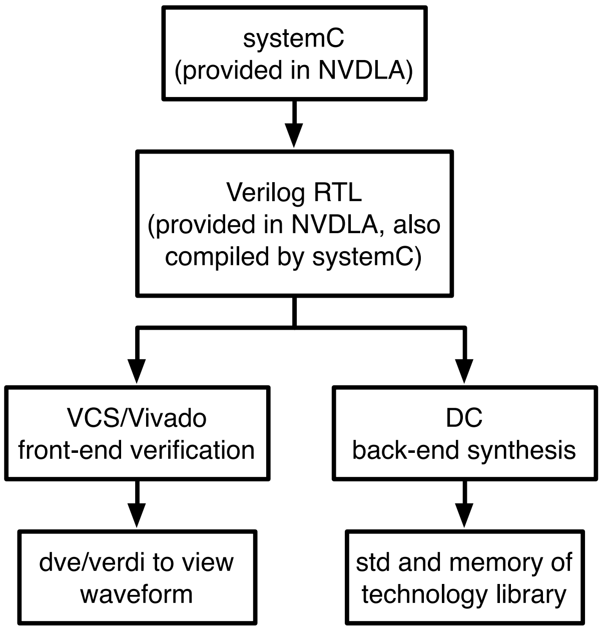
systemC
(provided in NVDLA)
Verilog RTL
(provided in NVDLA, also compiled by systemC)
VCS/Vivado
front-end verification
DC
back-end synthesis
dve/verdi to view waveform
std and memory of technology library

Figure 1: Hardware process for NVDLA

- Java - jdk1.7 (installed in "/usr/bin/java")
- Perl - perl-5.10 (installed in "/usr/bin/perl")
  XML::Simple
  Capture::Tiny
- CPP - gcc-4.9.3 (installed in "/usr/bin/g++"")
- CPP (installed in "/usr/bin/cpp")
- Python - python2.6 (installed in "/usr/bin/python")
- SystemC - systemc-2.3.0 (need to be installed, by "./configure –prefix" to install in local)

- Verilator - Verilator 3.912 (for Verilator builds, need to be installed, by "./configure –prefix" to install in local)
- clang - clang 3.4 (for Verilator builds, installed in "/usr/bin/clang" in linux2-4)

Additional tools:

- cpan (installed, however you have to reconfigurate installation in local dir and collect all add URL:https://metacpan.org)
- IO::Tee (run cpan IO::Tee to install)
- VCS (installed in "/opt2/synopsys/vcs-F-2011.12/bin")
- Verdi (installed in "/opt2/synopsys/verdi/Verdi_M-2017.03-SP2-2/bin")

2 Tree Build

Revise the Makefile in hw root:

- DEFAULT_CPP := /usr/bin/cpp
- DEFAULT_GCC := /research/byu1/thchen/tools/gcc4.9.3/bin/g++
- DEFAULT_PERL := /usr/bin/perl
- DEFAULT_JAVA := /usr/bin/java
- DEFAULT_SYSTEMC := /uac/gds/thchen/systemc/
- DEFAULT_VERILATOR := /research/byu1/thchen/tools/verilator-3.9.12/bin/verilator
- DEFAULT_CLANG := /research/byu1/thchen/tools/build/bin/clang

Until now, you can run in hw root:

- make
- ./tools/bin/tmake -build vmod
- ./tools/bin/tmake -build verif_sim
- ./tools/bin/tmake -build cmod_top

3 Front-end Verification

Next, we will compile Verilog RTL code and generate waveform.

If you want to use VCS to compile verilog RTL, you will revise the Makefile in hw-root/verif/sim/

- export VCS_HOMW := /opt2/synopsys/vcs-F-2011.12
- export VERDI_HOME := /opt2/synopsys/verdi/Verdi_M-2017.03-SP2-2
- export NOVAS_HOME := /opt2/synopsys/verdi/Verdi_M-2017.03-SP2-2

- export LM_LICENSE_FILE := /opt2/synopsys/key
- export VCS_CC := /usr/bin/g++

Until now, you can run in hw-root/verif/sim

- make build
- make run TESTDIR=../traces/traceplayer/sanity0
- make build DUMP=1 DUMPER=VERDI
- make vericom DUMP=1 DUMPER=VERDI
- make run DUMP=1 DUMPER=VERDI TESTDIR=../traces/traceplayer/sanity0
- make verdi DUMP=1 DUMPER=VERDI TESTDIR=../traces/traceplayer/sanity0

If you want to use vivado to compile verilog RTL, you will revise the Makefile in hw-root/verif/sim_vivado

- export XILINX_HOME := /research/byu1/thchen/tools/vivado0/Vivado/2017.1
- PERL := /bin/perl
- AWK := /bin/awk
- TEE := /bin/tee

Until now, you can run in hw-root/verif/sim_vivado

- make build
- make run TESTDIR=../traces/traceplayer/sanity0
- make run GUI=1 TESTDIR=../traces/traceplayer/sanity0
- make build DUMP=1
- make run DUMP=1 TESTDIR=../traces/traceplayer/sanity0
- make regress
- make regress MINIREGRESS=1

4 Back-end Synthesis

To be up-dated soon...

Acknowledge

I many thank Yuzhe Ma@CSE, CUHK and Zhenyu Zhu@IC school, SEU/Nanrui Micro give help for building this environment.

5 Reference

1. NVDLA, http://nvdla.org