*Department of Computing Science*
*Illinois Institute of Technology*

# HORNET: A Highly Configurable, Cycle-Level Multicore Simulator

**Marko Zivkovic and Shangping Ren**

Department of Computing Science
Illinois Institute of Technology
10 W. 31st Street, Chicago, IL 60616

Telephone (312) 567-3993
Email {mzivkovi, ren}@hawk.iit.edu

*Technical Report IIT-CS-115-08-2012*

## Acknowledgements

The research is support in part by NSF CAREER 0746643 and NSF CNS 1018731.

# 1 Installation Requirements

This chapter explains the installation of HORNET Simulation Software in de- tail. It covers the installation of all necessary packages, installation procedures and error handling. Moreover, we describe how the versions of these packages can cause installation problems. Please read this document entirely for straightforward installation.

## 1.1 Before Installation

Hardware requirements are not published, but from our experience, most of the currently used hardware is sufficient. For clarification, we are listing the hard- ware and software which we used in our testing.

- Linux Distribution (32 or 64)
- UBUNTU 12.04 and Debian 6.0.5 (used in our research)
- Intel Core Duo 2.2 GHz
- 3 Gb RAM DDR2
- HardDrive 160 Gb

## 1.2 Required Packages

This is a full list of the packages required for the HORNET's installation on fresh Operational System. Note that HORNET for some of these packages supports different versions. These packages will be labeled as "NOT UNIQUE." Software Package Required:

- Autoconf 2.64 is required to build GCC scripts.
- Automake 1.11 automatically generates 'Make
  le.in's from file called 'Make
  le.am'. NOT UNIQUE
- Autotools 20100122.1 used by the Automake and Libtool packages.
- Boost C++ library 1.49. NOT UNIQUE
- Gcc and g++ 4.4.5 compilers (MUST USE THIS VERSION)
- M4 1.4.14 for macro processors. NOT UNIQUE
- Libc6 2.13 additional library.
- QMTest 2.4.1 general purpose testing solution. NOT UNIQUE
- Python 2.7.
- Binutils 2.20.1 used to assemble, link and manipulate.
- Sourcery G++ Lite 4.4-303 is used for MIPS Core Simulation Mode as a cross-compiler. NOT UNIQUE

After installation of required packages, download HORNET installation
le from http://csg.csail.mit.edu/hornet/and unpack it.

## 1.3 Installation Problems and Solutions

We are presenting all of the problems which we encountered and how to dealing with them.

### 1.3.1 Gcc Compiler Problem

The biggest problem was caused by gcc and g++ compiler. Namely, the versions after gcc 4.4.5 are too strict for the HORNET's code. Dealing with these exceptions did not end successfully. The downgrade of gcc compiler requires a lot of work and we generally think that there is a better solution to this problem. First, one should try to install gcc 4.4.5 as the additional compiler. Now, during the execution, the command should be directed (setup the PATH) to the desired compiler version. Moreover, UBUNTU 12.4 comes with the latest gcc 4.7 and the problems are guaranteed. We simply decided to go with the Debian 6.0.5 ( Squeeze) distribution which by default has installed gcc and g++ 4.4.5.

### 1.3.2 Exception Handling 32 bits vs 64 bits

Even though HORNET is designed for 32 and 64 bits platforms, the installation process on 32 bits OS will cause additional compiling exceptions comparing to the 64 bits. Dealing with these exceptions is tricky and will most likely cause the additional problems so we what you to be familiar with these errors and how to fix them. The computer with 64 bits address space does not have this type of a problems. The difference between 32 and 64 bits machines is visible in step 3 of the installation steps (please refer to the "Installation Basics") The compiler on 32 bits machines generates 2 exceptions.

The first exception: "ld variable expects long int, instead of 64bit type." Commenting variable "ld" will not help because this variable is crucial for building the executable file. The only way is to change the type of the ld, but ld is not global variable which means that needs to be corrected in the whole file. The second exception is caused by a string: char lc=NULL; We can fix this problem but placing /0 instead of NULL. So, now we have: char lc=/0; After these corrections, the installation should be successful. Note that these 2 exceptions are not generated on 64 bits computers.

# 2 HORNET Installation

This chapter will explain the basic and additional Installation Steps depending on the HORNET mode.

## 2.1 Installation Basics

Ideally, the installation process should be
nished with these steps.

- In terminal, go to the HORNET installation folder and run:./bootstrap
- After bootstrap is done, type: ./configure prefix=home/(destination) - the destination for the generated files
- Commands "make" and "make install" will generate two files darsim and darimg. Finally, HORNET is ready to use.

## 2.2 Network-Only Mode Installation

This mode gives you the ability to test "syntectic patterns or application traces." The installation commands used in " Installation Basics" section describes how to finish this type of the installation. The generated files DARSIM and DARIMG will be used to run the simula- tions. If you successfully finished this step, please read section "Working with HORNET" where we will describe how to use configuration file and provided examples.

## 2.3 MIPS Multicore Simulations

By using this mode, we can run application in different setups (Hardware, NoC, memory, etc). Since we have to compile the application for the MIPS type of architecture, we need a cross-compiler (listed in required packages). Before we move on the MIPS Multicore Simulations installation process, be sure that this type of compiler is installed. Otherwise, the HORNET installation will by unsuccessful.

## 2.4 MIPS Multicore Simulations

By using this mode, we can run application in different setups (Hardware, NoC, memory, etc). Since we have to compile the application for the MIPS type of architecture, we need a cross-compiler (listed in required packages). Before we move on the MIPS Multicore Simulations installation process, be sure that this type of compiler is installed. Otherwise, the HORNET installation will by unsuccessful.

## 2.5 Sourcery G++ Lite Installation

There are many MIPS cross-compilers on the market, but we will use Sourcery G++ Lite 4.4-303 ( free version) which is proven to work with HORNET. To download the software, please go to:
https://sourcery.mentor.com/GNUToolchain/subscription3130?lite=Please follow the instructions.

First, in order to start installation, we need to change the default shell of the OS. Most of the Linux distributions comes with /bin/dash type of shell and Sourcery G++ Lite DOES NOT support it. So, lets use ZSH shell. First me have to be sure that we have zsh shell installed in our system. Please type:

whereis zsh

Check for /bin/zsh it means that zsh is installed. If zsh in not installed, use this command:

apt-get install zsh

After we are sure that zsh is installed in our system, we can change it using the command:

chsh -s /bin/zsh

after this command, please reset the root terminal To be sure that you are using zsh shell, use command:

echo $SHELL

Now we are ready to start the graphical installation by using: /bin/sh ./path/to/package.bin path to the Sourcery G++ Installation file Please follow the wizard and finish the installation. Now, we are ready to continue with the installation of the HORNET for MIPS Multicore Simulator Mode. NOTE: Strongly suggest to check the ./configure help options After using step 1 in "Installation Basics" please this command:

./configure enable-mips-rts

If the compilation is successful use "make" and "make install" to generate DARSIM,DARIMG and DAR-RTS-CONFIG. If compilation process is interrupted with the exception "MIPS Linker not found." that means that Sourcery G++ Lite is not the the right PATH with compiling process. You may use this command to specify where the files are located:

./configure -enable-mips-rts -with-mips-ld=EXEC -with-mips-as=EXEC -with-mips-cc=EXEC -with-mips-ar=EX -with-mips-ranlib=EXEC

OR use this command to setup the right PATH to the Sourcery Installation Files.

PATH=HOME=CodeSourcery=SourceryG++Lite=bin :PATH export PATH

MIPS Multicore Simulator Installation should be successful after this step. Note that now, the generated files should be DARIMG, DARSIM and DAR-RTS-CONFIG.

## 2.6 Running Simulation

HORNET Installation package contains the example application(blackscholes). Go to :
/hornet-1.0/examples/blacksholes

and check for the installation files. To run the simulation enter use "make" and "make install" The simulation will start and on the end, the statistics will be provided. It is important to know that running simulation in MIPS Multycore Mode requires additional configuration of the system image file. Please, check the HORNET Configuration section.

# 3 HORNET Configuration

## 3.1 Working with HORNET

This chapter explains how to use configuration scripts to set up your system. Moreover, it provides the details how to use files generated by HORNET. We will list the code which is relevant for the system configuration and explain how that code affects the system. So far, we generated files ( DARSIM and

DARIMG) should be located in specified folder. ./configure prefix=PATH. NOTE: For convenience and better understanding, we attached statically-linked DARSIM and DARIMG which can be used to run the HORNET. In addition to the generated files, we will provide configuration file in different stages(compiled, uncompiled) and specify their use.

Configuration file, hornet-1.0/scripts/dor-o1turn.py can be opened and edited in any PYTHON editor platform. All the changes and configurations are done in this step. HORNET package provides more types of the already configured files. NOTE: Please open doro1turn.py and check the comments for better understanding of the code and the system configured. After execution of the doro1turn.py in python, we will have multiple configuration files. For convenience, we provided only one(image.cfg) which can be used for the demonstration purposes. When you open the file, you can see the detail summary of the system configuration. Since we have the configuration file which describes our testing system, we can now use DARIMG command to get the image file. Please type: DARIMG image.cfg

After this step, please check your folder for "output.img" file. The file is also attached for convenience and can be executed by DARSIM command to get the statistics. The same configuration file is used in every HORNET's mode, but Full Multi- core Simulation with MIPS core requires additional settings. Since, this mode presents the whole system, we need to setup Memory Hierarchy and parameters

such as latency, bandwidth, cache size, table size, hit and miss latency, etc. For demonstration, please open file "test2.py" and check the memory hierarchy comments. This file contains additional information necessary to run the simulation.