The Noxim User Guide

Or Noxim in 24 hours

Noxim - the NoC Simulator

Preface

Noxim is a Network-on-Chip (NoC) simulator developed by the engineers Maurizio Palesi, Davide Patti and Fabrizio Fazzino at the Università degli Studi di Catania (Italy). The simulator is developed in SystemC: a system description language based on C++. The current release (20080119) requires SystemC 2.1.v1. Noxim can be downloaded from sourceforge under GPL license terms.

Noxim has a command line interface for defining several parameters of a NoC. In particular an user can insert the network size, buffer size, packet size distribution, routing algorithm, selection strategy, packet injection rate, traffic time distribution, traffic pattern, hot-spot traffic distribution.

The simulator allows NoC evaluation in terms of throughput, delay and power consumption. This information is delivered to the user both in terms of average and per-communication results. In detail, an user is allowed to collect different evaluation metrics including the total number of received packets/flits, global average throughput, max/min global delay, total energy consumption, per-communications delay/throughput/energy/etc.

In the noxim distribution, the simulator is shipped along with Noxim Explorer (see chapter3). The last tool is very useful during the design space exploration phase. Infact, Noxim Explorer executes many simulations using Noxim in order to explore the design space. This is achieved with the change in configuration parameters for each simulation.

Noxim often is used along with APSRA as showed in the figure 1. The inputs of the simulator are the routing tables created by APSRA and the communication graph. The routing tables are generated for each node of the network. The communication graph describes the application behavioural of a NoC system. The above inputs along with the NoC configuration parameters are sent to simulator.

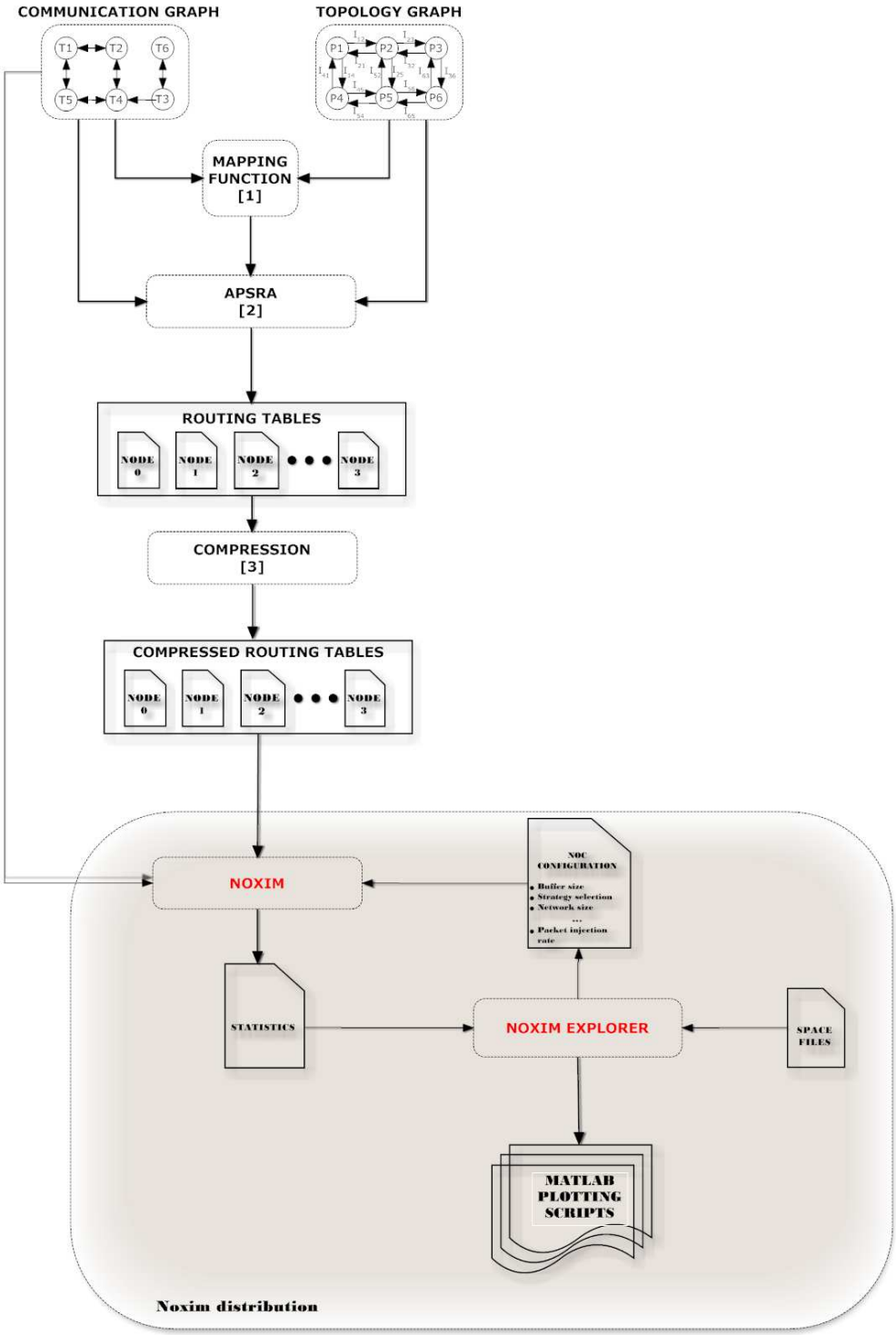

Figure 1: High-level flow design.

The Noxim's outputs will be those one that you can see above. Then, the results are sent to Noxim Explorer. This last will create new configuration parameters or complete the exploration according to the information read from a script (known as exploration script or space file). Information about some of the blocks reported in the figure 1 can be found in:

- G.Ascia, V.Catania, M.Palesi. *A Multi-objective Genetic Approach to Mapping Problem on Network-on-Chip*. Journal of Universal Computer Science, 12(4):370–394, 2006
- M.Palesi, R.Holsmark, S.Kumar, V.Catania. *A Methodology for Design of Application Specific Deadlock-free Routing Algorithms for NoC Systems*. International Conference on Hardware-Software Codesign and System Synthesis, pp. 142-147. Seoul, Korea, October 22-25, 2006
- M.Palesi, S.Kumar, R.Holsmark. *A Method for Router Table Compression for Application Specific Routing in Mesh Topology NoC Architectures*. SAMOS VI Workshop: Embedded Computer Systems: Architectures, Modeling, and Simulation, pp. 373-384. Samos, Greece, July 17-20, 2006
- G.Ascia, V.Catania, M.Palesi, D.Patti. *Neighbors-on-Path: A New Selection Strategy for On-Chip Networks*. Fourth IEEE Workshop on Embedded Systems for Real Time Multimedia, pp. 79-84. Seoul, Korea, October 26-27, 2006
- R.Holsmark, M.Palesi, S.Kumar. *Deadlock free routing algorithms for irregular mesh topology NoC systems with rectangular regions*. Accepted for publication in Journal of Systems Architecture

This introduction is split into three chapters:

Chapter 1 will walk you through the installation process for the Noxim distribution.

Chapter 2 will phase Noxim simulator in focusing on the more important options that you can send to simulator.

Chapter 3 will show the basic steps to carry out a design exploration by Noxim Explorer.

Contents

List of Figures

Chapter 1

Installation guide

1.1 Prerequisites

Operating systems Noxim will run on any Posix-compliant operating systems, this includes:

- Linux
- BSD
- Solaris
- AIX
- HP-UX

Environment In order to deploy the Noxim distribution on your desktop the following tools are necessary:

- SystemC 2.1.v1 for deployment. Download from http://www.systemc.org/downloads/stand
- gcc-2.95.3 or greater
- a patch to compile SystemC with gcc 4.x
- a compression tool for uncompressing the Noxim distribution

Disk Space About 41 MB of disk space.

1.2 Installing on Linux/Unix

The following steps take you through installing Noxim on Linux/Unix environments (I installed it on Ubuntu 7.04).

1. Download the latest noxim distribution from http://sourceforge.net/projects/noxim
2. Once the distribution is downloaded, extract the files from the zip file into a directory of your choice
3. Extract the distribution using the *tar* command. For example, if you downloaded 20080119 distribution -

   ```
   tar -xvfz noxim-20080119.tgz
   ```

4. Any noxim distribution downloaded from sourceforge will have the following structure:
   - `./noxim_vcs`
     - `./TOOLS`
       - `./apsra2noxim.cpp`
       - `./Makefile`
       - `./mapping2cg.cpp`
       - `./noxim_explorer.cpp`
       - `./noxim_explorer_new.cpp`
       - `./README_TOOLS`
       - `./spicefilegen.cpp`
     - `./AUTHORS`
     - `./COPYING`
     - `./main.cpp`
     - `./Makefile`
     - `./Makefile.defs`
     - `./README`

- ./TBuffer.cpp
- ./TBuffer.h
- ./TGlobalRoutingTable.cpp
- ./TGlobalRoutingTable.h
- ./TGlobalStats.cpp
- ./TGlobalStats.h
- ./TGlobalTrafficTable.cpp
- ./TGlobalTrafficTable.h
- ./TLocalRoutingTable.cpp
- ./TLocalRoutingTable.h
- ./TNoC.cpp
- ./TNoC.h
- ./TPower.cpp
- ./TPower.h
- ./TProcessingElement.cpp
- ./TProcessingElement.h
- ./TReservationTable.cpp
- ./TReservationTable.h
- ./TRouter.cpp
- ./TRouter.h
- ./TStats.cpp
- ./TStats.h
- ./TTile.h
- ./VERSION

5. Download SystemC version 2.1.v1 from http://www.systemc.org/downloads/standards/. The zip file is labeled systemc-2.1.v1.tgz

6. Change to the ./noxim_cvs directory

7. Inside this directory, extract the files from the zip file using the tar command

```
tar -xvfz systemc-2.1.v1.tgz
```

8. To compile SystemC 2.1.v1 with gcc 4.x, you have to apply a patch. You must download it from http://noxim.sourceforge.net/patch_systemc-2.1.v1-gcc4.tar.gz
9. Extract the files from the zip files labeled patch_systemc-2.1.v1-gcc4.tar.gz
10. Change to your SystemC top level directory (it is the directory that includes the src directory)
11. Move the file patch_systemc-2.1.v1-gcc4 into the current directory
12. Execute the following command:

```
patch -p0 <patch_systemc-2.1.v1-gcc4
```

13. Create a temporary directory, e.g:

```
mkdir objdir
```

14. Change to the temporary directory, e.g:

```
cd objdir
```

15. Configure the package for your system:

```
../configure
```

16. Compile the package:

```
make
```

17. Install the package:

```
make install
```

18. Check the installation:

```
make check
```

19. Change to your Noxim top level directory

20. Touch the file Makefile.defs. Below are reported only the changed lines:

```
SYSTEMC=./systemc-2.1.v1
```

21. Include the file Makefile.deps. Below there are the contents of this file: # DO NOT DELETE

    TNoC.o: TNoC.h TTile.h TRouter.h NoximDefs.h TBuffer.h TStats.h TPower.h

    TNoC.o: TGlobalRoutingTable.h TLocalRoutingTable.h TReservationTable.h

    TNoC.o: TProcessingElement.h TGlobalTrafficTable.h /usr/include/stdio.h

    TNoC.o: /usr/include/features.h /usr/include/sys/cdefs.h

    TNoC.o: /usr/include/bits/wordsize.h /usr/include/gnu/stubs.h

    TNoC.o: /usr/include/gnu/stubs-32.h /usr/include/bits/types.h

    TNoC.o: /usr/include/bits/typesizes.h /usr/include/libio.h

    TNoC.o: /usr/include/_G_config.h /usr/include/wchar.h

    TNoC.o: /usr/include/bits/wchar.h /usr/include/gconv.h

    TNoC.o: /usr/include/bits/stdio_lim.h /usr/include/bits/sys_errlist.h

    TNoC.o: /usr/include/stdlib.h /usr/include/sys/types.h /usr/include/time.h

    TNoC.o: /usr/include/endian.h /usr/include/bits/endian.h

TNoC.o: /usr/include/sys/select.h /usr/include/bits/select.h

TNoC.o: /usr/include/bits/sigset.h /usr/include/bits/time.h

TNoC.o: /usr/include/sys/sysmacros.h /usr/include/bits/pthreadtypes.h

TNoC.o: /usr/include/alloca.h

TRouter.o: TRouter.h NoximDefs.h TBuffer.h TStats.h TPower.h

TRouter.o: TGlobalRoutingTable.h TLocalRoutingTable.h TReservationTable.h

TProcessingElement.o: TProcessingElement.h NoximDefs.h TGlobalTrafficTable.h

TProcessingElement.o: /usr/include/stdio.h /usr/include/features.h

TProcessingElement.o: /usr/include/sys/cdefs.h /usr/include/bits/wordsize.h

TProcessingElement.o: /usr/include/gnu/stubs.h /usr/include/gnu/stubs-32.h

TProcessingElement.o: /usr/include/bits/types.h /usr/include/bits/typesizes.h

TProcessingElement.o: /usr/include/libio.h /usr/include/_G_config.h

TProcessingElement.o: /usr/include/wchar.h /usr/include/bits/wchar.h

TProcessingElement.o: /usr/include/gconv.h /usr/include/bits/stdio_lim.h

TProcessingElement.o: /usr/include/bits/sys_errlist.h /usr/include/stdlib.h

TProcessingElement.o: /usr/include/sys/types.h /usr/include/time.h

TProcessingElement.o: /usr/include/endian.h /usr/include/bits/endian.h

TProcessingElement.o: /usr/include/sys/select.h /usr/include/bits/select.h

TProcessingElement.o: /usr/include/bits/sigset.h /usr/include/bits/time.h

TProcessingElement.o: /usr/include/sys/sysmacros.h

TProcessingElement.o: /usr/include/bits/pthreadtypes.h /usr/include/alloca.h

TBuffer.o: TBuffer.h NoximDefs.h

TStats.o: TStats.h NoximDefs.h TPower.h

TGlobalStats.o: TGlobalStats.h TNoC.h TTile.h TRouter.h NoximDefs.h TBuffer.h

TGlobalStats.o: TStats.h TPower.h TGlobalRoutingTable.h TLocalRoutingTable.h

TGlobalStats.o: TReservationTable.h TProcessingElement.h

TGlobalStats.o: TGlobalTrafficTable.h /usr/include/stdio.h

TGlobalStats.o: /usr/include/features.h /usr/include/sys/cdefs.h

TGlobalStats.o: /usr/include/bits/wordsize.h /usr/include/gnu/stubs.h

TGlobalStats.o: /usr/include/gnu/stubs-32.h /usr/include/bits/types.h

TGlobalStats.o: /usr/include/bits/typesizes.h /usr/include/libio.h

TGlobalStats.o: /usr/include/_G_config.h /usr/include/wchar.h

TGlobalStats.o: /usr/include/bits/wchar.h /usr/include/gconv.h

TGlobalStats.o: /usr/include/bits/stdio_lim.h /usr/include/bits/sys_errlist.h

TGlobalStats.o: /usr/include/stdlib.h /usr/include/sys/types.h

TGlobalStats.o: /usr/include/time.h /usr/include/endian.h

TGlobalStats.o: /usr/include/bits/endian.h /usr/include/sys/select.h

TGlobalStats.o: /usr/include/bits/select.h /usr/include/bits/sigset.h

TGlobalStats.o: /usr/include/bits/time.h /usr/include/sys/sysmacros.h

TGlobalStats.o: /usr/include/bits/pthreadtypes.h /usr/include/alloca.h

TGlobalRoutingTable.o: /usr/include/stdio.h /usr/include/features.h

TGlobalRoutingTable.o: /usr/include/sys/cdefs.h /usr/include/bits/wordsize.h

TGlobalRoutingTable.o: /usr/include/gnu/stubs.h /usr/include/gnu/stubs-32.h

TGlobalRoutingTable.o: /usr/include/bits/types.h

TGlobalRoutingTable.o: /usr/include/bits/typesizes.h /usr/include/libio.h

TGlobalRoutingTable.o: /usr/include/_G_config.h /usr/include/wchar.h

TGlobalRoutingTable.o: /usr/include/bits/wchar.h /usr/include/gconv.h

TGlobalRoutingTable.o: /usr/include/bits/stdio_lim.h

TGlobalRoutingTable.o: /usr/include/bits/sys_errlist.h /usr/include/stdlib.h

TGlobalRoutingTable.o: /usr/include/sys/types.h /usr/include/time.h

TGlobalRoutingTable.o: /usr/include/endian.h /usr/include/bits/endian.h

TGlobalRoutingTable.o: /usr/include/sys/select.h /usr/include/bits/select.h

TGlobalRoutingTable.o: /usr/include/bits/sigset.h /usr/include/bits/time.h

TGlobalRoutingTable.o: /usr/include/sys/sysmacros.h

TGlobalRoutingTable.o: /usr/include/bits/pthreadtypes.h /usr/include/alloca.h

TGlobalRoutingTable.o: /usr/include/string.h NoximDefs.h

TGlobalRoutingTable.o: TGlobalRoutingTable.h

TLocalRoutingTable.o: TLocalRoutingTable.h TGlobalRoutingTable.h NoximDefs.h

TGlobalTrafficTable.o: TGlobalTrafficTable.h /usr/include/stdio.h

TGlobalTrafficTable.o: /usr/include/features.h /usr/include/sys/cdefs.h

TGlobalTrafficTable.o: /usr/include/bits/wordsize.h /usr/include/gnu/stubs.h

TGlobalTrafficTable.o: /usr/include/gnu/stubs-32.h /usr/include/bits/types.h

TGlobalTrafficTable.o: /usr/include/bits/typesizes.h /usr/include/libio.h

TGlobalTrafficTable.o: /usr/include/_G_config.h /usr/include/wchar.h

TGlobalTrafficTable.o: /usr/include/bits/wchar.h /usr/include/gconv.h

TGlobalTrafficTable.o: /usr/include/bits/stdio_lim.h

TGlobalTrafficTable.o: /usr/include/bits/sys_errlist.h /usr/include/stdlib.h

TGlobalTrafficTable.o: /usr/include/sys/types.h /usr/include/time.h

TGlobalTrafficTable.o: /usr/include/endian.h /usr/include/bits/endian.h

TGlobalTrafficTable.o: /usr/include/sys/select.h /usr/include/bits/select.h

TGlobalTrafficTable.o: /usr/include/bits/sigset.h /usr/include/bits/time.h

TGlobalTrafficTable.o: /usr/include/sys/sysmacros.h

TGlobalTrafficTable.o: /usr/include/bits/pthreadtypes.h /usr/include/alloca.h

TGlobalTrafficTable.o: NoximDefs.h

TReservationTable.o: NoximDefs.h TReservationTable.h

TPower.o: NoximDefs.h TPower.h

main.o: NoximDefs.h TNoC.h TTile.h TRouter.h TBuffer.h TStats.h TPower.h

main.o: TGlobalRoutingTable.h TLocalRoutingTable.h TReservationTable.h

main.o: TProcessingElement.h TGlobalTrafficTable.h /usr/include/stdio.h

main.o: /usr/include/features.h /usr/include/sys/cdefs.h

main.o: /usr/include/bits/wordsize.h /usr/include/gnu/stubs.h

main.o: /usr/include/gnu/stubs-32.h /usr/include/bits/types.h

main.o: /usr/include/bits/typesizes.h /usr/include/libio.h

main.o: /usr/include/_G_config.h /usr/include/wchar.h

main.o: /usr/include/bits/wchar.h /usr/include/gconv.h

main.o: /usr/include/bits/stdio_lim.h /usr/include/bits/sys_errlist.h

main.o: /usr/include/stdlib.h /usr/include/sys/types.h /usr/include/time.h

main.o: /usr/include/endian.h /usr/include/bits/endian.h

main.o: /usr/include/sys/select.h /usr/include/bits/select.h

main.o: /usr/include/bits/sigset.h /usr/include/bits/time.h

main.o: /usr/include/sys/sysmacros.h /usr/include/bits/pthreadtypes.h

main.o: /usr/include/alloca.h TGlobalStats.h

22. Execute the command (don't worry about the warnings...):

```
make depend
```

23. Compile Noxim package:

```
make
```

24. Change to the TOOLS directory

25. Compile the Noxim Tools:

```
make
```

26. Enjoy yourself!

1.3 Installing on Windows using coLinux

In this section we will see about how to install Noxim on a Windows system. There are many ways to do that. For example one way is to use a virtual machines or linux-like enviroments (Cygwin is an example). But this two ways are not so good because the first demands too much hardware resources (RAM and processor frequency) and the second solution doesn't act as a fully linux operating system (you can't do many things). Luckily there is a third way: Cooperative Linux. Cooperative Linux is the first working free and open source method for optimally running Linux on Microsoft Windows natively. More generally, Cooperative Linux (short-named coLinux) is a port of the Linux kernel that allows it to run cooperatively alongside another operating system on a single machine. For instance, it allows one to freely run Linux on Windows 2000/XP, without using a commercial PC virtualization software such as VMware, in a way which is much more optimal than using any general purpose PC virtualization software. For more information see http://www.colinux.org/. First we will see the steps necessary to set up the coLinux software, than we will see the steps to install the Noxim software on a linux distribution running up on coLinux. I have installed coLinux on Windows XP but it is also possible to install it on Windows Vista following similar steps to those ones showed below:

1. Download the most recent version of coLinux from http://www.colinux.org/. I downloaded the version 0.7.3 when I wrote wrote this manual

2. Run the executable installer

3. Push the button "Next" in the first window

4. Accept the License Agreement
5. Push the button "Next" in the Choose Components window
6. Select the installation location. For example, I have choosen `C:\coLinux`
7. In Get WinPCAP window push button "Next"
8. Select the root file system image. I selected Ubunutu 6.06.1 requiring 1 Gb of free space
9. Select a mirror (random is okay)
10. Push the button "Install"
11. Don't worry about the warning to install TAB-Win32AdapterV8. Then push "continue anyway" button
12. When installation is done push "Next" button
13. Push "Finish" button
14. Change to your coLinux directory (the installation location)
15. Extract here the filesystem named Ubuntu-6.06.1.ext3.1gb.bz2
16. Download the filesystem swap file from http://gniarf.nerim.net/colinux/fs/. I choosed a swap of 128 Mb, that is the archive file named fs_128Mb.bz2
17. Move the swap image into coLinux directory
18. Extract here the swap image
19. Create a configuration file (the file extension is .conf). For example, I have created a file named coLinux.conf which contains the following lines:

    # The default kernel kernel=vmlinux

    # File contains the root file system. # Download and extract preconfigured file from SF "Images for 2.6". cobd0="c:/coLinux/Ubuntu-6.06.1.ext3.1gb"

    # Swap device, should be an empty file with 128..512MB. cobd1="c:/coLinux/fs_128Mb"

Tell kernel the name of root device (mostly /dev/cobd0, # /dev/cobd/0 on Gentoo) # This parameter will be forward to Linux kernel. root=/dev/cobd0

Additional kernel parameters (ro = rootfs mount read only) #ro

Initrd installs modules into the root file system. # Need only on first boot. initrd=initrd.gz

Maximal memory for linux guest mem=300

Slirp for internet connection (outgoing) # Inside running coLinux configure eth0 with this static settings: ipaddress 10.0.2.15 broadcast 10.0.2.255 netmask 255.255.255.0 gateway 10.0.2.2 nameserver 10.0.2.3 eth0=slirp,02:00:00:00:0

20. Create a shortcut for colinux-daemon.exe
21. Rename it in coLinux
22. Right-click on it e go into Properties
23. In the target field write a command like this:

```
C:\coLinux\colinux-daemon.exe @coLinux.conf
```

24. Move the shortcut in your desired location
25. Double-click on the shortcut to run coLinux
26. When the system will ask you to insert a username digit root
27. When the system will ask you to insert a password digit root
28. Run the following command to install the right keyboard

```
dpkg-reconfigure console-data
```

Note: to write the '-' symbol you must digit '+'

29. A window will be displayed. Select "OK"
30. Select keymap from arch list

31. Select the keyboard layout. In my case I selected qwerty
32. Select your national keyboard. I selected Swedish keyboard :)
33. Select standard
34. In order to access to files and directories of a Windows host system from a coLinux guest system, you have to change the file /etc/fstab in this way:

    ```
    0 /d/daten  cofs user,noauto 0 0
    1 /media/cdrom  cofs user,noauto 0 0
    2 /floppy cofs user,noauto 0 0
    ```

    For example my pen drive is mounted on E partition. To mount it on my ubuntu system, I need to run the following command:

    ```
    mount -t cofs cofs1 /mnt/KINGSTON
    ```

    Change the fstab and configuration file according your needs
35. Run `apt-get install make`
36. Run `apt-get install gawk`
37. Run `apt-get install gcc`
38. Run `apt-get install g++`
39. Run `apt-get install makedepend`
40. Run `apt-get install update`
41. Run `apt-get install upgrade`
42. Now you can follow the normal steps to install the Noxim distribution as showed in section1.2

Chapter 2

Getting Started with Noxim

2.1 Synopsis

noxim [options]

options Command-line options.

2.2 Description

The noxim tool launches a NoC simulation. It does this by invoking the sc_main function. Infact, every C/C++ program has a main() function. When the SystemC library is used the main() function is not used. Instead the function sc_main() is the entry point of the application. This is because the main() function is defined into the library so that when the program starts initialization of the simulation kernel and the structures this requires to be performed, before execution of the application code. The main() defined into the library calls the sc_main() function after it has finished with the initialization process.

The sc_main() function is defined as follows:

int sc_main(int argc, char *argv) which is the same as a main() function in common C++ programs. The values of the arguments are forwarded from the main() defined in the library.

In the sc_main() function the structural elements of the system are created and connected throughout the system hierarchy. This is facilitated by the C++

class object construction behavior. When a module comes into existence all submodules this contains are also created. (When a C++ object is created all objects that this contains are also created). After all modules have been created and connections have been established the simulation can start by calling the function sc_start() where the simulation kernel takes control and executes the processes of each module depending on the events occurring at every simulated cycle. In the first cycle all the Processes are executed at least once unless otherwise stated with a call to the function dont_initialize() when the Process is registered. For more details about getting started SystemC I suggest you to read http://staff.science.uva.nl/ jesshope/Downloads/Assignment.1.doc. Note that in the noxim sc_main function before invoking sc_start() and run a new simulation all modules are reset.

2.3 Options

In this section it is provided a detailed description concerning the options that a noxim user can set. You can execute the command to access to the list of options:

```
./noxim -help
```

Then, in screen video you should view something lik this:
SystemC 2.1.v1 — Sep 18 2007 16:17:16
Copyright (c) 1996-2005 by all Contributors
ALL RIGHTS RESERVED
Noxim - the NoC Simulator
(C) University of Catania
Usage: ./noxim [options]
where [options] is one or more of the following ones:

- help Show this help and exit
- verbose N Verbosity level (1=low, 2=medium, 3=high, default off)
- trace FILENAME Trace signals to a VCD file named 'FILENAME.vcd' (default off)

- dimx N Set the mesh X dimension to the specified integer value (default 4)
- dimy N Set the mesh Y dimension to the specified integer value (default 4)
- buffer N Set the buffer depth of each channel of the router to the specified integer value [flits] (default 4)
- size Nmin Nmax Set the minimum and maximum packet size to the specified integer values [flits] (default min=2, max=10)
- routing TYPE Set the routing algorithm to TYPE where TYPE is one of the following (default 0): xy XY routing algorithm

  westfirst West-First routing algorithm

  northlast North-Last routing algorithm

  negativefirst Negative-First routing algorithm

  oddeven Odd-Even routing algorithm

  dyad T DyAD routing algorithm with threshold T

  fullyadaptive Fully-Adaptive routing algorithm

  table FILENAME Routing Table Based routing algorithm with table in the specified file
- sel TYPE Set the selection strategy to TYPE where TYPE is one of the following (default 0): random Random selection strategy

  bufferlevel Buffer-Level Based selection strategy

  nop Neighbors-on-Path selection strategy
- pir R Set the packet injection rate to the specified real value [0..1] (default 0.01)
- dist TYPE Set the time distribution of traffic to TYPE where TYPE is one of the following: poisson Memory-less Poisson distribution (default)

  burst R Burst distribution with given real burstness

pareto on off r Self-similar Pareto distribution with given real parameters (alfa-on alfa-off r)

custom R Custom distribution with given real probability of retransmission

- traffic TYPE Set the spatial distribution of traffic to TYPE where TYPE is one of the following (default 0'):

  random Random traffic distribution

  transpose1 Transpose matrix 1 traffic distribution

  transpose2 Transpose matrix 2 traffic distribution

  table FILENAME Traffic Table Based traffic distribution with table in the specified file

- hs ID P Add node ID to hotspot nodes, with percentage P (0..1) (Only for 'random' traffic)

- warmup N Start to collect statistics after N cycles (default 1000)

- seed N Set the seed of the random generator (default time())

- detailed Show detailed statistics

- volume N Stop the simulation when either the maximum number of cycles has been reached or N flits have been delivered

- sim N Run for the specified simulation time [cycles] (default 10000)

Now we take a closer look at each option.

The *help* option allow you to know the possible options accepted by Noxim (the same list that you can see above).

When you insert the *verbose* option you can monitor the verbosity level of the output generated by Noxim. There are four levels. By default verbosity output is off. In this case you can know only the main statistics produced by Noxim (total received packets, total received flits, global average delay, global average throughput, throughput, max delay, total energy).

When the verbosity level is set to low, in addition to the ouput generated when verbosity is off, the configuration parameters are reported and you can see the work done by each element of the NoC system, that is processing elements and routers.

About processing element the information delivered will be in the following format:

simulation time: ProcessingElement[id] ACTIVITY [flit type, sequence number, source node->destination node]

Where ACTIVITY can be one of the following values:

- SENDING: when the processing element send a flit to a given destination node
- RECEIVING: when the processing element receives a flit from a given source node

Where flit type can be one of the following values:

- H: head flit
- B: body flit
- T: tail flit

Instead for a router the information delivered is so structured:

simulation time: Router[id] Input[id] ACTIVITY [flit type, sequence number, source node->destination node]

Where ACTIVITY can be one of the following values:

- SENDING: when the router send a flit to a given destination node
- Received flit: when the router receives a flit from a given source node
- (number of flits), reserved Output[id]: when the router reserves an output port for the given number of flits
- forward to Output[id]: when the router forwards a flit to another router NoC

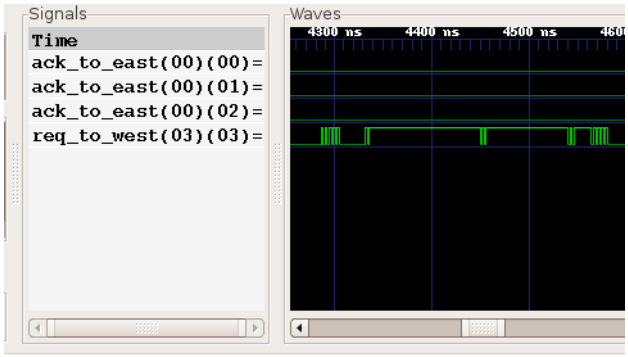

Figure 2.1: Example of vcd file created by Noxim.

If the verbosity level is set to medium you see nothing difference with the previous case (so at least it appears).

When the verbosity level is set to high, in addition to the ouput generated when verbosity is low (medium), you can see a detailed information about flit for each activity performed by each NoC element. This information is delivered in the following format:

Source Tile[id] Destination Tile[id] Flit type Sequence number Payload printing (not implemented yet) Unix timestamp at packet generation Total number of hops from source to destination.

You use the *trace* option to trace all the SystemC signals used in the NoC simulator (clock, reset, req_to_east, etc.) to a VCD file named 'FILENAME.vcd'. The default value is off. The figure 2.1 shows the waveforms housed in the vcd file created by Noxim using trace option. I used gtkwave to view the waveforms.

The options *dimx* and *dimy* are used to insert topology information.

The option *buffer* is used to define the buffer size. of each channel of the router. This size is expressed in flits.

The option *size* is clearly explained in the menu viewed when you insert help option.

The option *routing* enable you to specify a number of routing algorithms as above listed. The explanation of each algorithm is beyond the purposes of this guide so to get a complete figure of this topics users will have to cross-refer to specific literature about routing algorithms.

You choose the selection function by setting *sel* option. The default is random. That is the output port where to send a flit is choosen in a pseudo-random fashion among the set of admissible output ports. You'd opt for a buffer-level strategy. Here the favourites output ports are those wich are connected to the more free channels. A free channel is one with the greater number of free slots in the destination FIFO buffer. About Neighbors-on-Path selection stratey please refer to [1].

With the *pir* option you can set the packet injection rate to the specified real value.

If you wish to manage the time distribution of traffic you use *dist* option. In particular you can decide if the traffic generation is patterned as poisson, burst, pseudo-pareto or custom distribution.

For random traffic you can define some nodes as hot spot nodes. This is accomplished with the *hs* option. Along with the node identificator you must specify the hot spot percentage.

With *warmup* option you can set the start time after which the simulator begins to collect statistics.

In order to understand the *seed*,*volume*,*sim* option please read the information provided by help option.

The *detailed* option provide per-communications statistics. In particular, for each destination node n_d are collected the aggregated average delay and throughput. Then the statistics for each communication having n_d as a destination node are reported using the table showed in figure 2.2. Figure 2.3 shows the final information returned by Noxim. The matrix labeled *matrix_delay* is a *NxN* matrix where *N* is the number of nodes in one dimension. So the element at row *i*, column *j* represents the max delay for the node situated at position *i* in X dimension and *j* in Y dimension.

Figure 2.4 summarize it. The second matrix labeled *routed_flits* houses the total number of routed flits. The format of matrix routed_flits is the same of previous matrix.

```
% Aggregated average delay (cycles): 8.4625
% Aggregated average throughput (flits/cycle): 0.063386
%  src  dst delay avg delay max     throughput       energy    received    received
%              cycles    cycles    flits/cycle        Joule     packets       flits
     9   15   10.6667        16     0.00498063    7.407e-09           3          27
     8   15      17.5        25     0.00310835    9.876e-09           4          28
     7   15   7.66667        11     0.00354233    4.938e-09           3          20
     6   15     10.75        19     0.00291788    7.407e-09           4          28
     5   15     13.75        25     0.00252092    9.876e-09           4          25
    11   15   15.5556        38     0.00594813    2.469e-09           9          50
     0   15      18.8        25     0.00314999   1.4814e-08           5          26
     3   15   10.5556        24     0.00634076    7.407e-09           9          62
    14   15   4.45455         9     0.00733226    2.469e-09          11          73
    12   15         8         8     0.00169707    7.407e-09           4          16
    10   15         6         6     0.00333637    4.938e-09           4          22
     4   15      13.2        17     0.00245198   1.2345e-08           5          24
     1   15     12.75        18     0.00569647   1.2345e-08           8          53
    13   15   7.85714        19     0.00466695    4.938e-09           7          33
     2   15        12        16     0.00237369    9.876e-09           3          21
% Aggregated average delay (cycles): 11.0361
% Aggregated average throughput (flits/cycle): 0.0600638
];
```

Figure 2.2: Output generated using detailed option.

```
max_delay = [
       41    38    29    38
       47    24    31    47
       50    41    37    35
       36    26    31    31
];

routed_flits = [
          302       902       889       341
          942      1713      1746       980
         1082      1791      1657      1014
          406       993      1074       389
];
```

Figure 2.3: The final information returned when you run Noxim with detailed option.

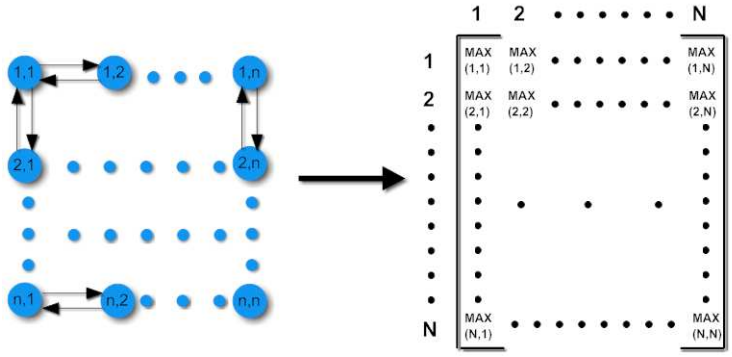

Figure 2.4: Format of matrix *max_delay* generated when you run Noxim with detailed option.

2.4 Learn Noxim Through an Example

If you want to simulate a 8x8 NoC, you must execute this command:

```
./noxim -dimx 8 -dimy 8
```

To make a more accurate simulation you must specify the simulation time (in terms of cycles) and the warm-up session. This is accomplished using the sim and warmup options. Then to carry out a 8x8 NoC with a simulation time of 40000 cycles and warm-up session of 5000 cycles you must execute the following command:

```
./noxim -sim 40000 -warmup 5000 -dimx 8 -dimy 8
```

Put hard warm-up session must be about 20-25% simulation time!

If you want change the FIFO buffer size you can set the buffer option. For example, if you wish a buffer size of two flits you must run this command:

```
./noxim -sim 40000 -warmup 5000 -dimx 8 -dimy 8 -buffer 2
```

To set up a good simulation you must provide the routing and traffic information. That objective is achieved using routing and traffic option. By all means, you want to insert the routing table and the communication graph generated by APSRA algorithm. If this is your case, then follow this steps (I'm going to suppose that you are skilled with APSRA):

1. Redirect the APSRA output to a file, e.g a file named Foo:

   ```
   ./apsra_general [ options ] > Foo
   ```

2. Create the routing and traffic file by means of the command:

   ```
   ./apsra2noxim Foo
   ```

   You can find the executable apsra2noxim.exe into the TOOLS directory (for Noxim directory structure see chapter 1). The output of the previous command will be the two files labeled respectively Foo.rt and Foo.cg.

3. Send the routing and traffic information to the simulator to carry out a NoC simulation:

   ./noxim -sim 40000 -warmup 5000 -dimx 8 -dimy 8 -buffer 2 -routing table Foo.rt -traffic table Foo.cg

Also it's important to specify a selection policy case you want a behaviour different from the default one (pseudo-random fashion). Use the sel option to accomplish it, e.g if you wish to use a buffer-level policy then you must execute:

./noxim -sim 40000 -warmup 5000 -dimx 8 -dimy 8 -buffer 2 -routing table Foo.rt -traffic table Foo.cg -sel bufferlevel

Finally, you must specify the packet injection rate by means of pir option.

Well, now you can use and try yourself the Noxim capabilities. However, this is useful only for a starting training phase. If you want to carry on thorough simulation you cannot realize it without the help provided a tool called Noxim Explorer (see chapter 3).

Chapter 3

Getting Started with Noxim Explorer

Noxim Explorer is a powerful tool that allow you to explore the design space. For example, you can simulate a given NoC at different pir values. But first you have to create a space file to accomplish it.

3.1 Writing the space file

To create a configuration file you can follow two ways. Initially you use the first way to create a configuration file using the executable labeled spacefilegen. Then you always use the second way consisting in to change the template according your needs.

In the first way, the configuration file (e.g Bar.cfg) is generated by noxim spacefilegen. Below the steps to create the space file are showed:

- First change to TOOLS directory
- Run spacefilegen.exe
- Insert the mesh topology. You can set the X and Y dimension. It is possible to insert multiple networks. All networks will be simulated taking the same space file but with different information topology. As consequence Noxim

Explorer will create different Matlab plotting scripts each for a different network (see section 3.4).

- Insert the traffic information. If you use the routing tables generated by AP-SRA you must choose the table item from traffic menu. When you choose that, you must insert the traffic filename (for details how to generate this file view the chapter 2). You can specify different traffic patterns. Noxim Explorer will carry out different simulations each for a traffic pattern.

- Insert the routing information. Please, note that if you choose the eight item (insert a routing information file), you cannot specify a filename. The reason is quite simple, there is a bug! But dont'worry about it, after you will fix it. You can specify different routing algorithms. Noxim Explorer will carry out different simulations each for a routing algorithm.

- Choose the selection strategy. You can specify different selection strategies. Noxim Explorer will carry out different simulations each for a selection strategy.

- Specify the packet size. For each new entry you must specify the min and max packet size separated by a backspace. You can specify different packet sizes. Noxim Explorer will carry out different simulations each for a packet size.

- Insert the buffer size. You can specify different buffer sizes. Noxim Explorer will carry out different simulations each for a buffer size.

- Insert the min packet injection rate. I suggest you to insert random values, after it will be exposed a way to select the pir values in a correct manner

- Insert the number of simulations for each pir value in order to reduce the confidence interval. I suggest you to insert at least 20 repetitions for each simulation point

- Specify the noxim simulator path. If you are into the TOOLS directory then the path to noxim is `../noxim`

- Insert the simulation time
- Insert warm-up session time
- Insert some aggregation parameters. In that manner, Noxim Explorer will not create new output files when the specified parameters change. A common choice is to enter 'pir' so that for each configuration of all other parameters a file is created containing all the simulations at the different pir values. You often carry out a simulation with the only change in packet injection rate. If that is your case, simply enter 'pir'.
- Choose a filename to save the spacefile, e.g Foo

Open this file, you should view something like this:

```
% automatically generated by noxim spacefilegen

[routing]
table ./Foo/Foo.rt
[/routing]

[sel]
bufferlevel
[/sel]

[traffic]
table ./Foo/Foo.cg
[/traffic]

[pir]
\% min max step
0.005 0.0075 0.00025
[/pir]
```

```
[topology]
8 x 8
[/topology]

[buffer]
\% buffer fifo size in flits
3
[/buffer]

[size]
\% packet size in flits
5 10
[/size]

[default]
\% parameters not changed during space exploration
-sim 40000 -warmup 5000
[/default]

[aggregation]
\% parameters that change value inside each generated file
pir
[/aggregation]

[explorer]
repetitions 20
simulator ../noxim
[/explorer]

% end of file
```

As you can see, this is a structured file. There are different sections in it. Let's

take a closer look into the inside of this file:

- routing section: contains the information about routing in the format: <type> <filename>. You must specify a filename only if the routing type is set to `table`
- sel section: the selection policy.
- traffic section: contains the information about traffic in the format: <type> <filename>. You must specify a filename only if the traffic type is set to `table`
- pir section: contains the information about pir values
- topology section: in this section we collect the topology information
- buffer section: the buffer size
- size section: maximum and minimum packet size
- default section: simulation and warm-up time
- aggregation section: the aggregation parameters. The name of a parameter must be equal to the name of a section wherein you can insert multiple values. In particular, the allowed values are: routing, sel, traffic, pir, topology, buffer and size.
- explorer: the number of repetitions and the path to noxim

If you set multiple vaules for one or more parameters then Noxim Explorer run a number of simulations equal to the number of multiple values. For example, if you set two buffer sizes and two packet sizes, Noxim Explorer will execute four simulations. Below you will find the space file of the talked-about example:

```
% automatically generated by noxim spacefilegen

[routing]
```

```
table ./Foo/Foo.rt
[/routing]

[sel]
bufferlevel
[/sel]

[traffic]
table ./Foo/Foo.cg
[/traffic]

[pir]
\% min max step
0.005 0.0075 0.00025
[/pir]

[topology]
8 x 8
[/topology]

[buffer]
\% buffer fifo size in flits
3
4
[/buffer]

[size]
\% packet size in flits
5 10
7 11
[/size]

[default]
```

```
\% parameters not changed during space exploration
-sim 40000 -warmup 5000
[/default]

[aggregation]
\% parameters that change value inside each generated file
pir
[/aggregation]

[explorer]
repetitions 20
simulator ../noxim
[/explorer]

% end of file
```

In the first simulation the first buffer and first packet size are used while in the second simulation the first buffer size and second packet size are used and so on. At the end, Noxim Explorer will create four Matlab plotting scripts. Taking the above example, if you wish to create a single Matlab file you must use three aggregation parameters (pir,buffer and size):

```
% automatically generated by noxim spacefilegen

[routing]
table ./Foo/Foo.rt
[/routing]

[sel]
bufferlevel
[/sel]
```

```
[traffic]
table ./Foo/Foo.cg
[/traffic]

[pir]
\% min max step
0.005 0.0075 0.00025
[/pir]

[topology]
8 x 8
[/topology]

[buffer]
\% buffer fifo size in flits
3
4
[/buffer]

[size]
\% packet size in flits
5 10
7 11
[/size]

[default]
\% parameters not changed during space exploration
-sim 40000 -warmup 5000
[/default]

[aggregation]
\% parameters that change value inside each generated file
pir
```

```
buffer
size
[/aggregation]

[explorer]
repetitions 20
simulator ../noxim
[/explorer]

% end of file
```

3.2 Run noxim explorer

After you have created a space file, it is time to start simulations. It is achieved running noxim explorer and entering the space files as parameters, e.g execute this command for the a space file labeled Foo (I assume that Foo is in the same directory as noxim_explorer executable):

```
./noxim_explorer ./Foo
```

The synopsis of the command is:

```
./noxim_explorer <cfg file> [<cfg file>]
```

As you can see, you can insert more space file simultaneously.

3.3 Select pir values in a correct way

Below is showed the basic rules to set the packet injection rate:

1. Change to TOOLS directory
2. Run Noxim Explorer, e.g ./noxim_explorer Foo.

3. Stop the execution after the beginning of the first simulation.
4. Copy the command line.
5. Change to Noxim top level directory.
6. Paste the command line.
7. Perform different simulations with changes only in pir values.
8. The packet injection rate so that the simulation returns a global average delay ranging from 20 cycles to 30 cycles is the minimum pir.
9. The packet injection rate so that the simulation returns a global average delay about 200 cycles is the maximum pir.
10. Set the pir step in order to collect at least ten simulation points.

3.4 Matlab plotting scripts

At the end of simulations, noxim explorer will create a matlab file. If you use as aggregation parameters others variables than 'pir', than the explorer will create different matlab files. One for each value of aggregation parameter. In order to view the simulation waveform you must run Matlab and execute the Matlab function labeled with the same name of the file, e.g if it is supposed that the plot script is labeled Bar.m, inside Matlab you must enter the following command:

```
Bar('b-+')
```

Thanks to this command, you set the waveform color to blue, continuous line and mark the simulation point with '+' symbol. The figure 3.1 show the output of the previous commands. You can specify different patterns about the appearance of waveform. Refer to Matlab help (in Matlab environment press F1 hot key to invoke this help).

Let us to study the structure of a Matlab file. Below is showed the contents of a M-FILE created by Noxim Explorer:

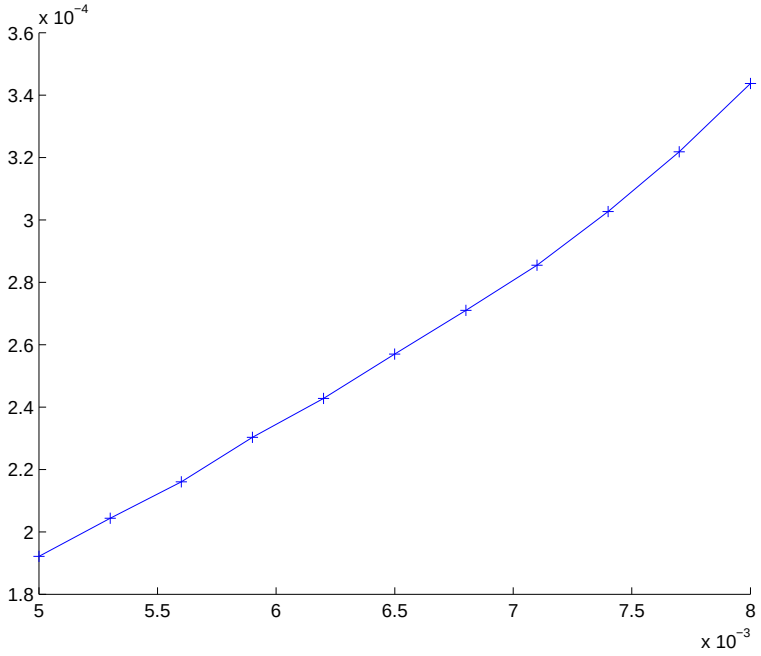

Figure 3.1: Example of output from matlab plotting script created by Noxim Explorer.

```
1:  % fname: routing_table___FE_BufferLevel_FE_BufferLevel_rt__sel_bufferlevel__siz
2:  % ../noxim -routing table ./FE_BufferLevel/FE_BufferLevel.rt -sel bufferlevel -size 3
3:  function [max_pir, max_throughput, min_delay] = routing_table___FE_BufferLevel_
4:  data = [
5:  % buffer pir avg_delay throughput max_delay total_energy rpackets rflits
6:                          3 0.003 18.1953 0.0165526 92 0.000113242 5217 28967
7:                          5 0.003 18.4697 0.0167211 92 0.000115238 5316 29262
8:                          3 0.0033 18.9956 0.0179394 100 0.000123343 5693 31394
9:                          5 0.0033 19.0728 0.0184571 103 0.000127775 5840 32300
10:                         3 0.0036 19.6067 0.0198114 148 0.000137131 6282 34670
11:                         5 0.0036 19.551 0.01996 138 0.000140534 6347 34930
12:                         3 0.0039 20.2779 0.0220463 164 0.000153854 6973 38581
13:                         5 0.0039 19.907 0.0215663 139 0.000149293 6857 37741
14:                         3 0.0042 20.2807 0.0230069 168 0.000160899 7328 40262
15:                         5 0.0042 20.9006 0.0236811 288 0.000166285 7538 41442
16:                         3 0.0045 21.2879 0.0247869 192 0.000175309 7902 43377
17:                         5 0.0045 20.5315 0.0243286 141 0.000169865 7753 42575
18:                         3 0.0048 21.7358 0.0256606 248 0.000183184 8216 44906
19:                         5 0.0048 22.0027 0.0265154 257 0.000184574 8458 46402
20:                         3 0.0051 23.1103 0.0279497 299 0.000199941 8910 48912
21:                         5 0.0051 22.7227 0.0279371 285 0.000195468 8875 48890
22:                         3 0.0054 24.9849 0.029856 457 0.000214055 9540 52248
23:                         5 0.0054 24.2277 0.0295829 470 0.000208746 9431 51770
24:                         3 0.0057 28.1506 0.031308 385 0.000229286 9966 54789
25:                         5 0.0057 29.371 0.031492 939 0.000227437 9985 55111
26:                         3 0.006 25.3521 0.0326709 305 0.000232782 10407 57174
27:                         5 0.006 25.99 0.0324606 324 0.00023148 10361 56806
28: ];
29: rows = size(data, 1);
30: cols = size(data, 2);
31: data_delay = [ ];
32: for i = 1:rows/1,
```

```
33:  ifirst = (i - 1) * 1 + 1;
34:  ilast = ifirst + 1 - 1;
35:  tmp = data(ifirst:ilast, cols-6+1);
36:  avg = mean(tmp);
37:  [h sig ci] = ttest(tmp, 0.1);
38:  ci = (ci(2)-ci(1))/2;
39:  data_delay = [data_delay; data(ifirst, 1:cols-6), avg ci];
40:  end
41:  figure(1);
42:  hold on;
43:  plot(data_delay(:,1), data_delay(:,2), symbol);
44:  data_throughput = [ ];
45:  for i = 1:rows/1,
46:  ifirst = (i - 1) * 1 + 1;
47:  ilast = ifirst + 1 - 1;
48:  tmp = data(ifirst:ilast, cols-6+2);
49:  avg = mean(tmp);
50:  [h sig ci] = ttest(tmp, 0.1);
51:  ci = (ci(2)-ci(1))/2;
52:  data_throughput = [data_throughput; data(ifirst, 1:cols-6), avg ci];
53:  end
54:  figure(2);
55:  hold on;
56:  plot(data_throughput(:,1), data_throughput(:,2), symbol);
57:  data_maxdelay = [ ];
58:  for i = 1:rows/1,
59:  ifirst = (i - 1) * 1 + 1;
60:  ilast = ifirst + 1 - 1;
61:  tmp = data(ifirst:ilast, cols-6+3);
62:  avg = mean(tmp);
63:  [h sig ci] = ttest(tmp, 0.1);
64:  ci = (ci(2)-ci(1))/2;
65:  data_maxdelay = [data_maxdelay; data(ifirst, 1:cols-6), avg ci];
```

```
66:  end
67:  figure(3);
68:  hold on;
69:  plot(data_maxdelay(:,1), data_maxdelay(:,2), symbol);
70:  data_totalenergy = [ ];
71:  for i = 1:rows/1,
72:  ifirst = (i - 1) * 1 + 1;
73:  ilast = ifirst + 1 - 1;
74:  tmp = data(ifirst:ilast, cols-6+4);
75:  avg = mean(tmp);
76:  [h sig ci] = ttest(tmp, 0.1);
77:  ci = (ci(2)-ci(1))/2;
78:  data_totalenergy = [data_totalenergy; data(ifirst, 1:cols-6), avg ci];
79:  end
80:  figure(4);
81:  hold on;
82:  plot(data_totalenergy(:,1), data_totalenergy(:,2), symbol);
83:  %——— Saturation Analysis ————
84:  slope=[ ];
85:  for i=2:size(data_throughput,1),
86:       slope(i-1) = (data_throughput(i,2)-data_throughput(i-1,2))/(data_throughput(i,1)-
87:  end
88:  for i=2:size(slope,2),
89:       if slope(i) <(0.95*mean(slope(1:i)))
90:            max_pir = data_throughput(i, 1);
91:            max_throughput = data_throughput(i, 2);
92:            min_delay = data_delay(i, 2);
93:            break;
94:       end
95:  end
```

The first two columns of matrix *data* refer to aggregation parameters buffer and pir. The third, fourth, fifth, sixth columns contains average delay, throughput, max delay, exploited total energy values, respectively. The next to last column contains the number of received packets. In the last column is reported the number of received flits. The rest of M-File contains Matlab plotting instructions. To understand MATLAB Programming consult this page http://en.wikibooks.org/wiki/Matlab.

Bibliography

[1] G.Ascia, V.Catania, M.Palesi, D.Patti. *Neighbors-on-Path: A New Selection Strategy for On-Chip Networks*. Fourth IEEE Workshop on Embedded Systems for Real Time Multimedia, pp. 79-84. Seoul, Korea, October 26-27, 2006.