NIRGAM[1] Manual: A Simulator for **N**oC **I**nterconnect **R**outin**g** and **A**pplication **M**odeling

A collaboration between researchers at

University of Southampton UK and

Malaviya National Institute of Technology India

Lavina Jain (lj06v@ecs.soton.ac.uk)

May 3, 2007

[1]NIRGAM is a joint effort by people from Electronic Systems Design group, School of Electronics and Computer Science, University of Southampton UK and Department of Computer Science and Engineering, Malaviya National Insitute of Technology, Jaipur India. This work is funded by EPSRC (UK) grant EP/C512804/1.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

NIRGAM is a discrete event, cycle accurate simulator targeted at Network on Chip (NoC) research. It provides substantial support to experiment with NoC design in terms of routing algorithms and applications on various topologies. The simulator is written in systemC. This document provides reference documentation for using and extending NIRGAM.

1.1 Network on Chip

This section describes Network on Chip(NoC) terminology and parameters. Users aware of NoC may skip this section.

1.2 NIRGAM Capabilities

NIRGAM is an extensible and modular systemC based simulator. It allows to experiment with various options available at every stage of NoC design: topology, switching technique, virtual channels, buffer parameters, routing mechanism and applications. Besides built-in capabilities, it can be easily extended to include new applications and routing algorithms. The simulator can output performance metrics (latency and throughput) for a given set of choices. This section presents an overview of the capabilities of the simulator. The documentation on running simulation and extending the simulator can be found in subsequent chapters.

Users can configure the following NoC parameters:

1. **Topology**
   One of the following topologies can be selected:

   - 2-dimensional mesh
   - 2-dimensional torus

   The size of topology can be specified in terms of number of rows and number of columns.

2. **Switching mechanism**
   The simulator supports wormhole switching mechanism, in which pakets are divided into flits. A packet consists of 3 types of flits:

   - head flit: This is the first flit.
   - data flit: All flits following the head flit except the last one are data flits.
   - tail flit: This is the last flit.

   Packet switching mechanism can be modeled by creating packets consisting of only one flit. A special flit called hdt flit is provided to model single-flit packets.

3. **Virtual channels**
   Users can configure number of virtual channels per physical channel.

4. **Buffer**
   Number of buffers in an input channel fifo can be specified. Each buffer is of the size of a flit.

5. **Clock Frequency**
   Clock frequency (in GHz) can be specified. The default clock frequency is 1 GHz.

6. **Routing algorithms**
   Users can select one of the following routing algorithms:

   - Source routing - for torus or mesh topology
   - Deterministic XY - for mesh topology
   - Adaptive Odd Even(OE) - for mesh topology

7. **Applications**
   One of the following applications can be attched to each tile in the network:

   - Source (Sender)
   - Sink (Reciever)
   - Synthetic traffic generators:
     - **Constant bit rate (CBR):**
       Configurable parameters for CBR traffic are as follows:
       * Packet size (in bytes)
       * Load percentage (percentage of channel bandwidth to be used)

    * Destination - User can specify a fixed destination or randomly chosen destination
    * Inter-flit interval (in clock cycles)

  - **Bursty:**

    Bursty traffic is represented by alternating on and off periods. During on period, packets are generated in fixed intervals. During off period, no packets are generated. Traffic is characterized by the following exponentially distributed variables:

    * Burst length - Number of packets in a burst (on period)
    * Off time - Interval between two bursts

    Configurable parameters for Bursty traffic are as follows:

    * Packet size (in bytes)
    * Load percentage (percentage of channel bandwidth to be used)
    * Destination - User can specify a fixed destination or randomly chosen destination
    * Inter-flit interval (in clock cycles)
    * Average burst length (in number of packets))
    * Average off time (in number of clock cycles)

  - **Input trace based**

    This application generates the same traffic as in previous simulation. This allows to compare NoC performance for different set of parameter choices for the same traffic.

8. **Performance Analysis**

   NIRGAM allows performance analysis of NoC design on a per channel basis with run time integrated plots. It records the following performance metrics for each simulation:

   - Average latency per packet (in clock cycles) for each channel
   - Average latency per flit (in clock cycles) for each channel
   - Average throughput (in Gbps) for each channel

   It generates gnuplot graphs and matlab script that can be run to generate graphs in matlab.

Chapter 2

Simulator Basics

This chapter describes basic structure of the simulator and model just enough to enable you to play with in-built capabilities. Advanced users interested in extending the simulator should also refer to Chapter 4 for details on simulator internals.

2.1 NoC Model

NIRGAM models Network-on-Chip (NoC) as a 2-dimensional interconnect of tiles. Each tile consist of an ipcore connected to a router/switch by a bidirectional core channel. Hereafter, we will use router and switch interchangeably. A tile is connected to neighbor tiles by bidirectional channels. Each tile is identified by a unique integer id called tileID. Also, each tile can be identified by a pair x-cordinate and y-cordinate. Figure 2.1 and Figure 2.2 illustrate 3 x 4 mesh and torus topologies respectively, where each circle represents a tile in the network.

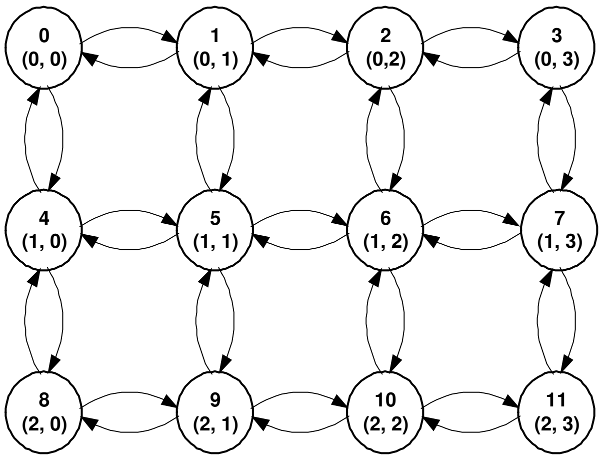

FIGURE 2.1: Mesh topology

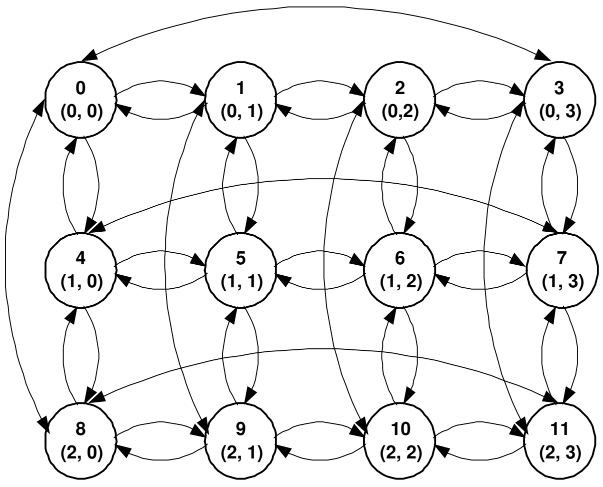

FIGURE 2.2: Torus topology

2.2 Routing

XY and OE routing algorithms can be used for 2-dimensional mesh topologies. The input required for these algorithms is the tileID of the destination tile.
Source routing algorithm can be used for mesh as well as torus topologies. The input required for source routing algorithm is a route code that encodes entire path from source tile to destination tile. At each router there could be five possible directions in which a packet can be routed: North(N), South(S), East(E), West(W) and Core(C). Each direction is represented by a code as shown in Table 2.1.

Direction	**Decimal code**	**Binary code**
N	0	000
S	1	001
E	2	010
W	3	011
C	4	100

TABLE 2.1: Source routing: route code

At any router, rightmost 3 bits of the route code determine the output direction. Source routing is implemented by reading rightmost 3 bits of the route code and then right shifting the code by 3 bits. We explain the method to generate route code for a particular path with the help of an example. Consider topology shown in Figure 2.2. Let us calculate route code for sending packet from tile 9 to tile 1 along path: 9 $\rightarrow$ 8 $\rightarrow$ 0 $\rightarrow$ 1. This path can be represented by direction sequence: W $\rightarrow$ S $\rightarrow$ E $\rightarrow$ C. Route code can be obtained by writing the direction codes for these directions in reverse order, while maitaining the order of 3 bits in each direction code.
Thus route code for W(011) $\rightarrow$ S(001) $\rightarrow$ E(010) $\rightarrow$ C(100) = 100 010 001 011 = 2187. Alternatively route code in decimal can be obtained by repeatedly multiplying by 8 and adding direction codes in reverse order.

Thus route code for W(3) $\rightarrow$ S(1) $\rightarrow$ E(2) $\rightarrow$ C(4) $= (((((4\times8)+2)\times8)+1)\times8)+3 = 2187$.

2.3 Code Structure

Figure 2.3 shows the directory structure of the simulator code.

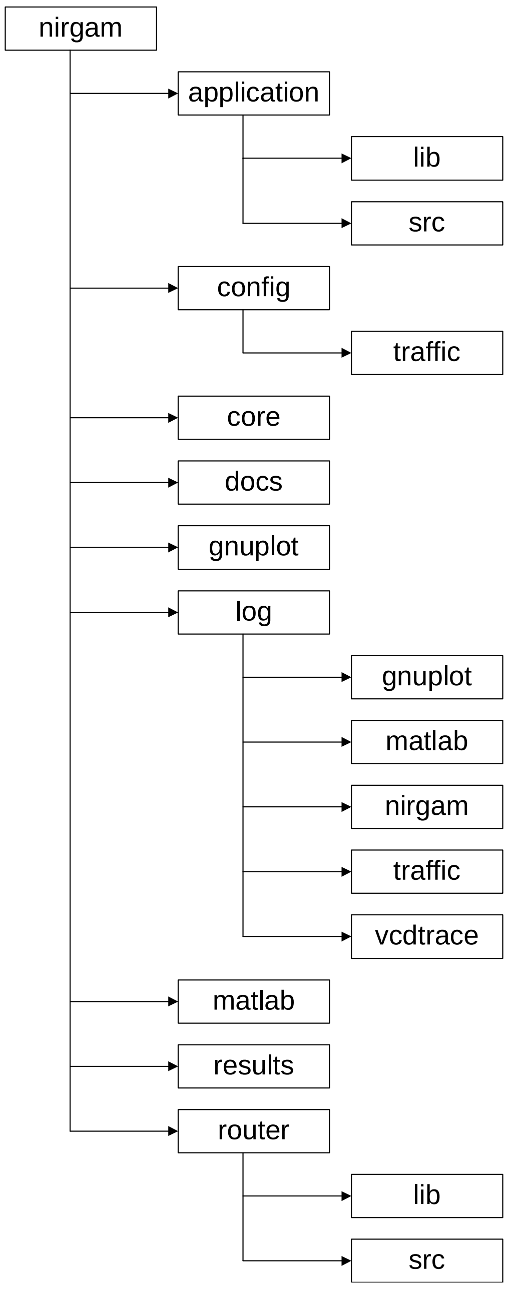

FIGURE 2.3: NIRGAM directory structure

The top level directory 'nirgam' contains code subdirectories, makefile and the executable. A brief description of the contents of each directory follows.

1. application
   This folder contains the source code of applications and application libraries that can be plugged in to an ipcore.

2. config
   This folder contains all the configuration files for the simulator as well as traffic. Users can experiment with different design choices by changing parameters in these files.

3. core
   This folder is the core engine that implements NoC. It contains code for implementation of NoC topology and architecture.

4. docs
   This folder contains the documentation.

5. gnuplot
   This folder contains gnuplot script for generating graphs.

6. log
   This folder contains log files. The subfolders are as follows:

   - gnuplot - contains input data files for gnuplot script.
   - matlab - contains input data files for matlab script.
   - nirgam - contains event log file for the complete simulation.
   - traffic - contains log of traffic generated by each tile.
   - vcdtrace - contains any vcdtrace generated using systemC.

7. matlab
   This folder contains matlab script to generate graphs.

8. results
   This folder contains result statistics and graphs for the simulations.

9. router
   This folder contains the source code of routing algorithms and respective libraries that can be attached to the tiles in the network.

Chapter 3

Running Simulations

This chapter provides a detailed documentation on configuring and running simulations.

3.1 Installation

1. Download and install systemC from www.systemc.org.
2. Download the simulator archive nirgam.tgz and extract to a location of your choice. You should see a folder ‘nirgam’ having the directory structure as Figure 2.3.
3. cd into the nirgam directory.
   Hereafter we will refer to the path to nirgam directory as $NIRGAM.
4. Edit $NIRGAM/Makefile.defs to set variable SYSTEMC equal to your systemC installation path.
5. run make.
   This should create an executable ‘nirgam’ in $NIRGAM.

3.2 Configuration

The configuration files to specify simulation parameters can be found in $NIRGAM/config.

- NoC design parameters can be set in nirgam.config.
- Application mapping (attach which application to which tile) can be specified in application.config.
- Traffic configuration parameters for synthetic traffic generators can be specified in $NIRGAM/config/traffic.

Next we describe each of these configuration files and their parameters in detail.

1. **nirgam.config**

   This is the main configuration file that contains parameters specific to NoC design and simulation. Table 3.1 describes the parameters in nirgam.config.

Parameter name	**Valid values**	**Description**
TOPOLOGY	MESH	Defines 2-dimensional mesh topology.
	TORUS	Defines 2-dimensional torus topology.
NUM_ROWS	Any natural number	Defines number of rows in the selected topology.
NUM_COLS	Any natural number	Defines number of columns in the selected topology.
RT_ALGO	XY	XY routing algorithm.
	OE	Odd Even routing algorithm.
	SOURCE	Source routing algorithm.
NUM_BUFS	Any natural number <= MAX_NUM_BUFS	Number of buffers in input channel fifo. MAX_NUM_BUFS is defined in $NIRGAM/config/constants.h and its default value is 16.
FLITSIZE	Any natural number	Size of flit in bytes.
HEAD_PAYLOAD	Any natural number <= FLITSIZE	Payload size (in bytes) in head/hdt flit.
DATA_PAYLOAD	Any natural number <= FLITSIZE	Payload size (in bytes) in data/tail flit.
DIRNAME	An alphanumeric string	Directory name in which results will be stored after simulation. Results are stored in $NIRGAM/results/DIRNAME.
LOG	0-4	Defines log level for the event log generated in $NIRGAM/log/nirgam/event.log.
	0	No log.
	1	Logs send and recieve events at ipcore.
	2	Logs major events in each module.
	3	Detailed log including request and ready signals.
	4	Detailed buffer and credit information at each clock cycle.
SIM_NUM	Any natural number	Defines clock cycles for which simulation runs.
WARMUP	Any natural number: 1 < WARMUP < SIM_NUM	Required only for synthetic traffic generators. Defines warmup period: number of clock cycles before traffic generation begins.
TG_NUM	Any natural number: WARMUP < TG_NUM < SIM_NUM	Required only for synthetic traffic generators. Defines clock cycle until which traffic is generated.
CLK_FREQ	Floating point value	Defines clock frequency in GHz.

TABLE 3.1: nirgam.config

2. **application.config**
This configuration file lets you specify application mapping. It accepts input as pairs of tileID and name of application library to be attached to the tile. The applications that can be attached are stored in $NIRGAM/application/lib. Tiles not specified in this file have no application attached to them and behave as only routers.
Table 3.2 lists the available applications.

Application library	**Description**
App_send.so	This is a demo application to display the working of simulator. It sends a packet consisting of 3 flits to tile 5 (destination tile can be changed in App_send.cpp). Each flit contains a string to be concatenated by the recieving tile.
App_concat.so	This is a demo application that recieves flits sent by App_send application and concatenates the recieved strings. It should be attached to tile specified as destination in App_send application (tileID 5). On recieving tail flit, it sends the concatenated string to tile 6.
App_recv.so	This is a demo application that recieves flit containing concatenated string sent by application App_concat.
Bursty.so	Bursty traffic generator.
CBR.so	Constant bit rate (CBR) traffic generator.
Trace_traffic.so	Generates same traffic as in previous simulation.
Sink.so	Sink/Reciever application, recieves flits destined to the tile to which it is attached.

TABLE 3.2: application.config

If one of the synthetic traffic generators (Bursty.so or CBR.so) is attached to any tile, then traffic parameters for that tile must be specified in $NIRGAM/config/traffic/tile-n, where n is the tileID.

Table 3.3 describes traffic parameters for CBR traffic generator.

Parameter name	**Description**
PKT_SIZE	Packet size in bytes.
LOAD	Percentage load. This determines percentage of maximum bandwidth being used.
DESTINATION	Destination tileID. Destination could be fixed or randomly chosen. To specify fixed destination, entry in configuration file looks like: DESTINATION FIXED n, where n is tileID. To select random destination, entry in configuration file looks like: DESTINATION RANDOM.
FLIT_INTERVAL	Interval between succesive flits in clock cycles.

TABLE 3.3: CBR traffic configuration

Bursty traffic generator requires all the parameters that are required for CBR, specified in Table 3.3. In addition, it requires parameters listed in Table 3.4 for exponentailly distributed variables. Refer to point 7 in Section 1.2 to read about modeling of synthetic traffic generators in NIRGAM.

Parameter name	**Description**
AVG_BURST_LEN	Average burst length (number of packets generated during on time).
AVG_OFFTIME	Average offtime: interval (in clock cycles) between two bursts.

TABLE 3.4: Bursty traffic configuration

3.3 Simulation

run ./nirgam from $NIRGAM directory.
Results, logs and graphs should be created as per configuration.

3.4 Performance analysis

NIRGAM measures performance of NoC on a per-channel basis. Figure 3.1 shows representation of performance metrics in matlab generated graphs. R0 - R15 shows the placement of tiles/routers. Red bar between R0 and R1 represents metric for east channel from R0 to R1. Blue bar between R0 and R1 represents metric for west channel from R1 to R0. Green bar between R0 and R4 represents metric for south channel from R0 to R4. Orange bar between R0 and R4 represents metric for north channel from R4 to R0.

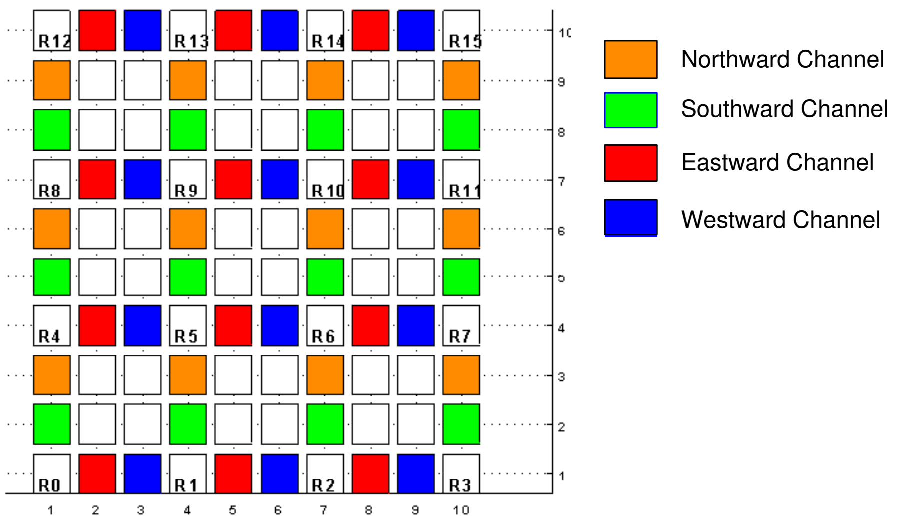

FIGURE 3.1: Matlab graph

3.5 Cleaning

Commands to clean files generated as a result of compiling and running simulations are as follows:

1. make clean - deletes all object files and libraries.
2. make cleanlogs - deletes all logfiles generated.
3. make cleanresults - deletes all results generated.
4. make ultraclean - does all of the above, deletes object files, libraries, logfiles and results.

3.6 Examples

We describe simulator configuration with the help of three example applications:

- String concatenation
- CBR with random destinations
- CBR with fixed destinations

3.6.1 String concatenation

Application description:

- Tile 0 sends a packet consisting of 3 flits to tile 5. Each flit consists of a string in its payload.
- Tile 5 concatenates the strings as it recieves the flits from tile 0. On recieving tail flit from tile 0, it sends a flit consisting of concatenated string in it's payload to tile 6.
- Tile 6 recieves the concatenated string.

Listing 3.1 lists nirgam.config. It simulates a 2-dimentional mesh topology of size 3 x 4. XY routing algorithm is used and results are stored in str_concat.

```
TOPOLOGY MESH
NUM_ROWS 3
NUM_COLS 4
RT_ALGO XY
DIRNAME str_concat
LOG 1
WARMUP 5
SIM_NUM 100
```

LISTING 3.1: String concatenation: nirgam.config

Listing 3.2 lists application.config.

```
0 App_send.so
5 App_concat.so
6 App_recv.so
```

LISTING 3.2: String concatenation: application.config

Sample configuration files for string concatenation application can also be found in $NIRGAM/config/examples/string_concatenation/.

3.6.2 CBR with random destinations

Application description:
Each tile behaves as a source as well as sink, implying that it is capable of generating as well as recieving flits. Each tile generates CBR traffic to randomly chosen destination.

Listing 3.3 lists nirgam.config. It simulates a 2-dimentional mesh topology of size 3 x 4. XY routing algorithm is used and results are stored in cbr_random. Traffic generation begins after 5 clock cycles, and continues until 300 clock cycles. Simulation stops after 1000 clock cycles.

```
TOPOLOGY MESH
NUM_ROWS 3
NUM_COLS 4
RT_ALGO XY
DIRNAME cbr_random
LOG 1
WARMUP 5
SIM_NUM 1000
TG_NUM 300
```

LISTING 3.3: CBR random: nirgam.config

Listing 3.4 lists application.config.

```
0 CBR.so
1 CBR.so
2 CBR.so
3 CBR.so
4 CBR.so
5 CBR.so
6 CBR.so
7 CBR.so
8 CBR.so
9 CBR.so
10 CBR.so
11 CBR.so
```

Listing 3.4: CBR random: application.config

Traffic configuration file for each tile must be provided in $NIRGAM/config/traffic/. Listing 3.5 lists an example traffic configuration file tile-n, where n is the tileID. Tile n generates 8-byte packets to a random destination. Interval between flits is 2 clock cycles. Traffic load is 50%, implying that packet injection rate is 50% of bandwidth. Interval between packets is a function of load percentage.

```
PKT_SIZE 8
LOAD 50
DESTINATION RANDOM
FLIT_INTERVAL 2
```

Listing 3.5: CBR random: traffic/tile-n

Sample configuration files for this application are provided in $NIRGAM/config/examples/cbr_random/.

3.6.3 CBR with fixed destinations

Application description:

- Tile 0 sends CBR traffic to tile 6.
- Tile 5 sends CBR traffic to tile 2.
- Tile 10 sends CBR traffic to tile 0.
- Tile 11 sends CBR traffic to tile 13.
- Tile 14 sends CBR traffic to tile 5.

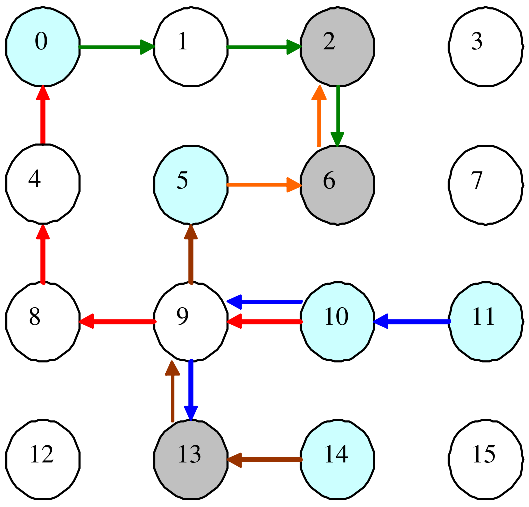

Figure 3.2: CBR traffic

Figure 3.2 illustrates traffic generated by this application.

Listing 3.6 lists nirgam.config. It simulates a 2-dimensional mesh topology of size 4 x 4. XY routing algorithm is used and results are stored in cbr_fixed. Traffic generation begins after 5 clock cycles, and continues until 300 clock cycles. Simulation stops after 1000 clock cycles.

```
TOPOLOGY MESH
NUM_ROWS 4
NUM_COLS 4
RT_ALGO XY
DIRNAME cbr_fixed
LOG 1
WARMUP 5
SIM_NUM 1000
TG_NUM 300
```

Listing 3.6: CBR fixed: nirgam.config

Listing 3.7 lists application.config.

```
0 CBR.so
5 CBR.so
10 CBR.so
11 CBR.so
14 CBR.so
2 Sink.so
6 Sink.so
13 Sink.so
```

Listing 3.7: CBR fixed: application.config

Traffic configuration file for tiles 0, 5, 10, 11 and 14 must be provided in $NIRGAM/config/traffic/. Listing 3.8, 3.9, 3.10, 3.11 and 3.12 list traffic configuration files.

```
PKT_SIZE 8
LOAD 100
DESTINATION FIXED 6
FLIT_INTERVAL 2
```

Listing 3.8: CBR fixed: traffic/tile-0

```
PKT_SIZE 8
LOAD 100
DESTINATION FIXED 2
FLIT_INTERVAL 2
```

Listing 3.9: CBR fixed: traffic/tile-5

```
PKT_SIZE 8
LOAD 100
DESTINATION FIXED 0
FLIT_INTERVAL 2
```

Listing 3.10: CBR fixed: traffic/tile-10

```
PKT_SIZE 8
LOAD 100
DESTINATION FIXED 13
FLIT_INTERVAL 2
```

Listing 3.11: CBR fixed: traffic/tile-11

```
PKT_SIZE 8
LOAD 100
DESTINATION FIXED 5
FLIT_INTERVAL 2
```

Listing 3.12: CBR fixed: traffic/tile-14

Sample configuration files for this application are provided in $NIRGAM/config/examples/cbr_fixed/.

Chapter 4

Simulator Internals