PCISIM
A Simulator for PCI Bus Based Systems

User's Manual

Robin Sharp
Dept. of Information Technology
Technical University of Denmark.

December 1999

Abstract

This document describes a PCI bus simulator for use in evaluating the feasibility of system designs based on this bus. The simulator allows the user to define a series of devices which can act as bus masters, specifying their maximum data rates and other parameters. The bus arbitration scheme may be rotating or based on fixed priorities assigned to the devices. The simulator can produce printed output describing the progress of the simulation in various levels of detail, and can also produce input files for gnuplot showing the relationship between generated data and transmitted data or the relationship between bus utilisation and time.

1 Introduction

Most modern computer-based systems are built up using a transaction-based I/O bus to transfer data between devices and memory or directly from one device to another. During the design of such a system, it can be important to check that this bus is able to transfer data to or from the attached devices at the rate required for correct operation of the device in the context of the desired system. This manual describes a simulator for evaluating the use of the PCI bus during such a design process. The simulator is based on the PCI Local Bus Specification, version 2.2 [1].

2 What can be simulated?

The PCI bus is controlled by a complex protocol which permits several different forms of transaction, of which the most important are:

- **Data Transfer** transactions for transferring data between peripheral devices or between peripheral devices and main memory.
- **Configuration** transactions, in which data are transferred to configure a device or to determine the properties with which a given device has been configured.
- **Special Cycle** transactions for broadcasting messages to several destinations.
- **Interrupt Acknowledge** transactions.

The simulator only deals with ordinary data transfer transactions. For such transactions, it is able to simulate the operation of both the 32 bit and 64 bit versions of the PCI bus, at clock frequencies of 33 and 66 MHz.

2.1 Data Transfer transactions

A transaction on the PCI bus involves an *address phase*, in which an address is transferred, followed by a sequence of one or more *data phases*, in each of which a data item is transferred. Each transaction is initiated by a bus device acting as *master* and involves another participant known as the *target*. Data transfers may be oriented towards the master or towards the target, with the corresponding transactions respectively being denoted *read* and *write* transactions. The address specified at the start of the transaction is the address within the target of the first data item transferred.

The full PCI bus specification differentiates between *single data phase* transfers, where only a single data item is transferred after the address, and *burst* transfers, where the transaction involves a sequence of several data phases. It also differentiates between transfers where the target is an *I/O device* and transfers where the target is the *memory*. The simulator does not make any of these distinctions.

The PCI bus may be implemented in a *32 bit* or *64 bit* version. In the 32-bit version, an address is represented by 4 bytes, which are transmitted in a single bus clock cycle, and each data item can be from 0 to 4 bytes, which are likewise transmitted in a single bus clock cycle. In the 64-bit version, an address is represented by 8 bytes and each data item by from 0 to 8 bytes, which are transmitted in a single bus clock cycle. According to the PCI bus specification, the number of data bytes to be transferred in a given data phase is specified by *byte enable* bus signals presented immediately prior to the actual transfer of data, and may vary from one data phase to the next within a burst transaction. The simulator makes the simplifying assumption that the first (N-1) data phases of an N data

phase transaction involve transfer of data with the maximum width (4 bytes or 8 bytes) for the bus version concerned, and that only the final transfer may involve fewer bytes, to allow for data transfers whose length is not an integral multiple of 4 or 8 bytes respectively.

The PCI bus specification allows the target to delay the transfer of a data item by inserting *wait states*, each lasting one clock cycle, in any data phase. In more detail, the first data phase of a transaction can be extended to at most 16 clock cycles and each subsequent data phase to at most 8 clock cycles. The simulator allows the user to specify a maximum number, say n_W, of wait states for each device. Note that these wait states are specified for the device as master, although strictly speaking they describe the behaviour of the device's target in the system under consideration. The user can also specify whether the actual number of wait states used in each data phase is selected *deterministically* or *stochastically*. With deterministic selection, exactly n_W wait states are inserted in every data phase. With stochastic selection, the simulator will for each data phase select a random whole number in the interval $[0..n_W]$ as the number of wait states to be inserted. These features make it possible to simulate the behaviour of devices with slow response and of targets such as main memory, which may exhibit access contention.

2.2 Early Termination

The PCI bus specification allows (or in some cases requires) transactions to be terminated before all the data transfers envisaged by the master at the start of the transaction have in fact been carried out. There may be several reasons for this, of which only a few are handled by the simulator:

1. Another master is attempting to gain control of the bus and the bus arbitrator grants the bus to this new master. This case is dealt with by the simulator in accordance with the PCI bus specification for arbitration using rotating or fixed priority. The simulator implements the rules for handling the master's *Latency Timer, LT*, whose initial value is specified by the user. A single idle state lasting one clock cycle is always interpolated on the bus between the current transaction and its successor. The simulator does not support *fast back-to-back access* without interpolated idle states.
2. The target is unable to respond within the limit of 8 or 16 clock cycles set by the rules for insertion of wait states. The target must then issue a *Disconnect* to the master, which must terminate the transaction. The master may subsequently attempt to restart the transaction after a delay of at least two bus clock cycles. In the simulator this type of situation is assumed not to occur.
3. The target does not respond to the master, cannot support a burst mode transaction requested by the master, issues a STOP signal to terminate the current transaction, or determines that the address to be used in the next data phase lies outside the

range of addresses available to it. In the simulator these situations are assumed not to occur.

The simulator does not simulate errors such as data or address parity errors or system errors.

3 Devices and Data Generation

In the simulator, a *device* is a potential bus master which can initiate either read or write data transfer transactions using blocks of data of a given size. Thus a device is a logical device corresponding to what in the PCI specification is denoted a *function*. If a physical device which can both read and write is to be simulated, or if the device uses several different block sizes, then two or more logical devices need to be defined.

Each device is characterised by:

Transaction type: Read for a device which reads data from the target, or Write for a device which writes data to the target.

Priority: An unsigned integer giving the priority of the device during fixed priority bus arbitration. For rotating priority arbitration, the value is ignored.

Buffer size: An unsigned integer, b, giving the size in bytes of the buffer which the device as master will attempt to fill or empty during each transaction. If no competing devices are active during the transaction, this amount of data will be transferred in a single burst. If pre-emption occurs before the end of the transaction, the remainder of the buffer will be transferred in one or more subsequent secondary transactions.

Maximum data rate: The maximum data rate, D_{max} bytes/s, at which the device can operate. The simulator will run a series of simulation trials using increasing fractions of this data rate. For fraction f and buffer size b bytes, the simulator will generate a buffer full of data every $b/(f \cdot D_{max})$ seconds, and will try to transfer its contents by submitting a data transfer transaction. If the contents of the buffer have not been completely transferred by the time the next buffer full is generated, the user is informed that a *data overrun* situation has occured.

Maximum wait states: The maximum number of wait states, n_W (measured in clock cycles), which can potentially be inserted in each data phase by the device's target in the system under consideration.

Wait state generator process: Stochastic if the number of wait states is selected randomly from the interval $[0..n_W]$ for each data phase, or Deterministic if the number of wait states is exactly n_W in each data phase.

Latency Timer: The initial value, LT clock cycles, for the latency timer at the start of each new transaction involving the device.

4 Input

Input to the simulator consists of an initial dialogue for setting general parameters of the simulation and general bus parameters, followed by a sequence of parameter specifications for each of the devices attached to the PCI bus.

4.1 General simulation parameters

These are typed in interactively in response to prompts from the simulator, as follows:

Prompt:	Response
`Enter number of masters:`	A positive integer giving the number of logical devices on the bus.
`Enter no. of clock cycles in simulation:`	A positive integer giving the length of the simulation in clock cycles.
`Enter no. of points on applied load axis:`	A positive integer giving the number, n_r, of simulation trials at increasing applied loads which will be performed. In trial i, each device will generate data at a fraction i/n_r of its maximum data rate.
`Is throughput plot required (y/n)?`	`y` if plot of bus throughput vs. applied load is required; `n` if no plot required.
`Is data rate plot required (y/n)?`	`y` if 3D plot of bus utilisation vs. time at various applied loads is required; `n` if no plot required.
`Are details of all transfers to be output (y/n)?`	`y` if verbose output with details of all transfers is required. These are by default sent to the user's terminal. `n` if no details required.
`Is histogram of transfer times required (y/n)?`	`y` if histogram of transfer times at various applied loads is required; `n` if no histogram required.

The `y/n` responses are not case-sensitive.

If a throughput plot is required, the user will be asked to supply a name for the file to contain data for passing to `gnuplot`.

If a data rate plot is required, the user will be asked to supply a name for the file to contain

data for passing to gnuplot, and also for the size in clock cycles of the time slot to be used for sampling the utilisation of the bus. A point will appear in the gnuplot plot for each time slot. Choosing a smaller value therefore gives a more accurate picture of the way in which the bus utilisation changes with time, but may cause a very large data file to be generated. Conversely, choosing a larger value gives a less accurate picture of how the utilisation develops, but reduces the size of the data file.

If a histogram of transfer times is required, the user will be asked to supply a name for the file to contain data for passing to gnuplot, and also for the number of histogram bins which are to correspond to the unit normalised transfer time. The unit normalised transfer time for each device is the time taken to transfer a buffer of data for that device if no pre-emption occurs and no wait states are inserted during the transfer.

4.2 General bus parameters

These are typed in interactively in response to prompts from the simulator, as follows:

Prompt:	Response
`Enter bus parameters:`	
`Frequency (MHz):`	Bus clock frequency, f, in MHz (33 or 66)
`Data size (bytes):`	Size of data objects, w, in bytes (4 or 8)
`Arbitration (f/r):`	`f` if bus arbitration takes place according to fixed (static) priorities associated with the individual devices; `r` if a rotating (round robin) arbitration scheme is to be used.

The `f/r` responses are not case-sensitive.

The values given for the bus clock frequency and data size are checked for validity. If a non-standard value is given, a warning is issued by the simulator, but the simulation can still take place. In this way it is possible to simulate the operation of the bus at frequencies other than 33 or 66 MHz, and using data sizes other than 4 or 8 bytes.

4.3 Device parameters

The parameters for the individual devices attached to the bus are typed in interactively. For each device, the dialogue has the following form:

Prompt:	Response
`Master` i`:`	
`Transaction type (R,W):`	`r` if device as master is to read from its target; `w` if device is to write to its target.
`Priority:`	A non-negative integer giving the priority of the device for use in bus arbitration.
`Buffer size (bytes):`	A positive integer giving the size b in bytes of the buffer which the device as master will attempt to fill or empty during each transaction.
`Max. data rate (bytes/s):`	A positive integer giving the maximum data rate, D_{max} bytes/s, at which the device can operate.
`Max. wait states/transfer:`	A integer in the interval [0..8] giving the maximum number of wait states, n_W, which can potentially be inserted in each data phase by the target when the device under consideration is master.
`Stochastic or determ. (S/D):`	`s` if the number of wait states to be inserted in each data phase is to be selected stochastically from the interval $[0..n_W]$; `d` if the number of wait states is deterministically chosen to be exactly n_W.
`Latency Timer (clock cyc.):`	A non-negative integer giving the value of LT, the latency timer for the device.

The `r/w` and `s/d` responses are not case-sensitive.

The first device to be specified is identified as *Master 1*, the second as *Master 2* and so on. The numbering of the masters and the order in which devices are specified is arbitrary and has no significance for the simulation.

5 Output

Output from the simulator consists of some standard information directed to the standard output stream `stdout` (usually the user's terminal), and a number of optional forms of output, as selected by the user during the initial dialogue.

```
Simulation results:

  Generated data  Transmitted data   Average burst lengths (data cycles)
1      3095840           3095840        12.0   1018.5    256.0   2730.7
2      6259264           6258320        12.0   1018.5    235.5   1068.5
3      9289568           9203340  *     12.0   1018.5    253.7    702.2
4     12384384          11524484  *     12.0   1018.5    245.2    569.9
5     15477152          13723040  *     12.0   1017.4    244.8    409.6
```

Figure 1: Example of standard output from simulation

5.1 Output to `stdout`

The standard output consists, for each trial at increasing fractions of the maximum data rate specified by the user, of:

- The amount of data, d_g bytes, generated by all the devices during the simulation period.
- The amount of data, d_t bytes, actually transferred on the bus during the simulation period. Typically, since there may not have been time to transfer the last data generated before the simulation period ends, d_t will be slightly smaller than d_g. If data overrun occurs, so some generated data cannot be sent off at all, d_t may be significantly smaller than d_g. Data overrun is indicated by an asterisk to the right of the d_t value.
- The average lengths (number of data phases) of the bursts of data transmitted by each of the devices. If a device i has a buffer of size b_i bytes and the bus transmits data objects of size w bytes, then the bursts of data can have a maximum length of b_i/w data phases. If device i is pre-empted by other devices during its transmission, the bursts will be shorter than b_i/w. If the total simulation time is not long enough for *any* data to be generated for a particular device, the average burst length is undefined, and will appear as `NaN`.

An example of this output is shown in Figure 1 for a simulation with four devices, where there are five simulation trials, at 20, 40, 60, 80 and 100% of the maximum data rate for each device respectively. In this example, data overrun occurs for one or more devices at data rates of 60% of the maximum (trial 3) and above. The four devices in this example have priorities 6, 5, 4 and 3, and their maximum possible burst lengths are 12, 1024, 256 and 16384 respectively.

If verbose output is requested by the user, so that details of all transfers are sent to standard output, the table of results is supplemented with information about:

1. The time instants at which new buffers full of data are generated for the individual

```
Simulation results:

    Generated data  Transmitted data   Average burst lengths (data cycles)
***Time=     1058: New buffer, device 3.  Gen.data=1024
***Time=     1058: Select master, device 3
***Time=     2132: New buffer, device 2.  Gen.data=5120
***Time=     2132: Select master, device 2
***Time=     2410: New buffer, device 3.  Gen.data=6144
***Time=     3762: Data overrun, device 3
***Time=     4182: Select master, device 3
***Time=     5114: New buffer, device 3.  Gen.data=8192
***Time=     5114: Select master, device 3
***Time=     6466: New buffer, device 3.  Gen.data=9216
***Time=     6466: Select master, device 3
***Time=     7539: New buffer, device 2.  Gen.data=13312
***Time=     7539: Select master, device 2
***Time=     7818: New buffer, device 3.  Gen.data=14336
***Time=     9170: Data overrun, device 3
***Time=     9567: Select master, device 3
***Time=    10522: New buffer, device 3.  Gen.data=16384
***Time=    10522: Select master, device 3
     :
     :
***Data overrun messages suppressed.
5     15477152          13723040  *     12.0   1017.4    244.8    409.6
```

Figure 2: Example of verbose output from simulation

devices. At each such instant, the total number of bytes of data generated so far during the simulation is also given, in the form `Gen.data=`*nnnn*.

2. The time instants at which a new master is selected on the PCI bus.
3. The time instants at which data overruns occur. If more than 10 data overruns take place in a given simulation trial, further messages about data overruns during the given trial are suppressed, but at the end of the trial a message `***Data overrun messages suppressed` is produced.

If the same set of devices is used as in Figure 1, and the data generation rate is chosen to be 100% of the maximum, then the start and end of the output if verbose output is selected might, for example, look as shown in Figure 2. Very large amounts of data may be produced if verbose output is requested, so this facility should be used with care.

```
# Throughput plot
# ===============
# No. of masters:                   4
# Bus frequency (MHz):              33.0
# Size of data objects (bytes):     4
# Arbitration scheme:               Fixed
#
# Devices attached as masters:
#    R/W  Prio   Block size     Data rate   Max. WS  S/D   LT
#  1  R     6         48          4800.0       0     D     48
#  2  R     5       4096      25000000.0       2     S    128
#  3  W     4       1024      25000000.0       0     D    128
#  4  W     3      65536       1150000.0       2     S     64
#
# Simulation time (clock cycles):   10000000
# Generated data  Transmitted data
     3095840           3095840
     6259264           6258320
     9289568           9203340
    12384384          11524484
    15477152          13723040
```

Figure 3: Example of gnuplot input file containing throughput data

5.2 Gnuplot input for throughput plot

If the user requests that a file containing data for a throughput plot be produced, then an output file containing data suitable for reading into gnuplot will be generated. This file is headed by details of the simulation and of the devices attached to the bus, in the form of gnuplot comments. The actual data to be plotted consist of *(Generated data, Transmitted data)* pairs, (d_g, d_t), suitable for plotting via the gnuplot `plot` command. A complete example of the output in the file is shown in Figure 3.

If a throughput data file, say `tplot.dat`, is generated, gnuplot is automatically activated at the end of the simulation. A suitable sequence of commands for setting up gnuplot to display the throughput plot is given in Appendix A. If these commands are stored in a file `throughp.gnp`, the plot can be set up by using the gnuplot `load` command, and displayed by using the `plot` command. For example:

```
gnuplot> load "throughp.gnp"
gnuplot> plot "tplot.dat" notitle with linespoints
```

This will by default display the plot in a separate window. For the example given in Figure 3, the appearance of the plot is shown in Figure 4.

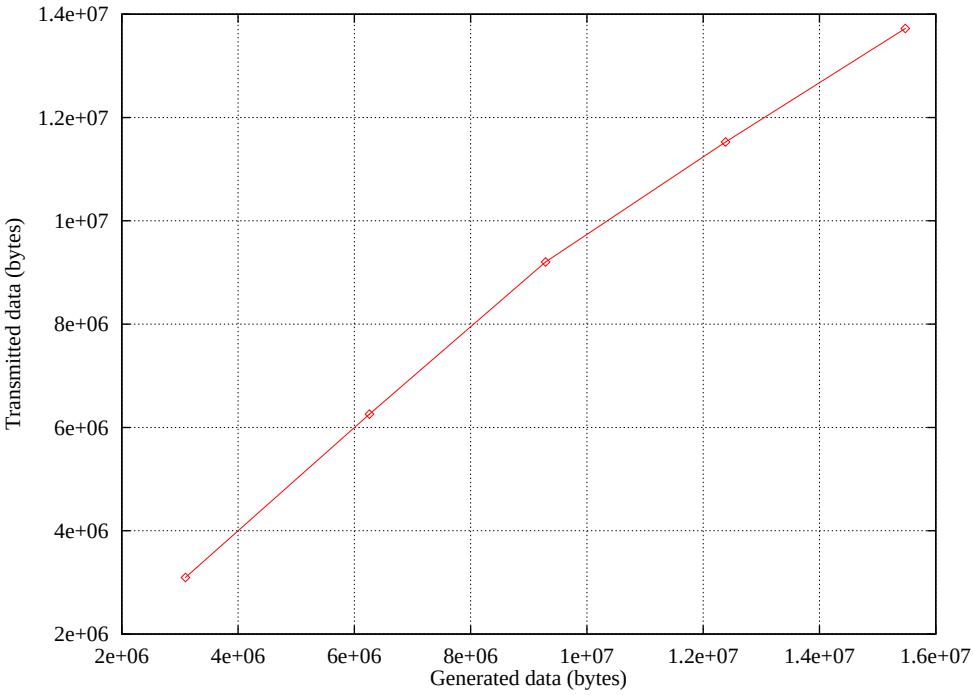

Figure 4: Example of throughput plot

For details of how to save the plot in machine-readable formats such as PostScript, LaTeX or TIFF, you should consult the original gnuplot documentation, which is available both in hardcopy form and as on line help (via the gnuplot command `help`). To leave gnuplot, use the gnuplot command `quit`.

5.3 Gnuplot input for data rate plot

If the user requests that a file containing data for a data rate plot be produced, then an output file containing data suitable for reading into gnuplot will be generated. As in the case of the throughput data file, this file is headed by details of the simulation and of the devices attached to the bus, in the form of gnuplot comments. The actual data to be plotted consist of *(Time, Load, Bus utilisation)* triplets, (t, l, u), suitable for plotting as a 3D plot via the gnuplot `splot` command, such that the time, t, will be plotted along the x-axis, load l along the y-axis and bus utilisation u along the z-axis. In each triplet, t is given in bus cycles, l is the fraction of the maximum data rate generated by each device (in the interval $[0; 1]$), and u is the fraction of the bus cycles during which data are actually transferred. If none of the devices insert wait states, the maximum possible value for u is 1. If wait states are inserted, fewer bus cycles are available for actual data transfer, and the maximum achievable value for u is less than 1.

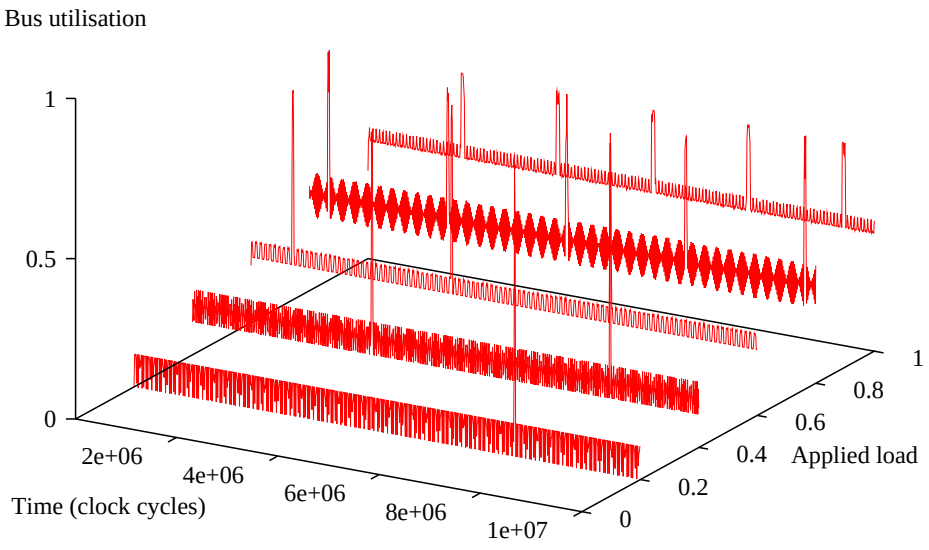

Figure 5: Example of 3D data rate plot

An example of the beginning of such a file is shown in Figure 6. Note that the generated file may be very large, especially if a small value of the time slot between consecutive utilisation sampling points is selected.

If a data rate file, say `rplot.dat`, is generated, `gnuplot` is automatically activated at the end of the simulation. A suitable sequence of commands for setting up `gnuplot` to display the data rate plot is given in Appendix B. If these commands are stored in a file `drate.gnp`, the plot can be set up by using the `gnuplot load` command, and displayed by using the `splot` command. For example:

```
gnuplot> load  "drate.gnp"
gnuplot> splot "rplot.dat" notitle
```

As in the case of the throughput plot, this will by default display the plot in a separate window. For the example given in Figure 6, the appearance of the plot is shown in Figure 5.

```
# Data rate plot
# ==============
# Sample slot width (clock cycles): 10000
# No. of masters:                   4
# Bus frequency (MHz):              33.0
# Size of data objects (bytes):     4
# Arbitration scheme:               Fixed
#
# Devices attached as masters:
#    R/W  Prio   Block size     Data rate   Max. WS  S/D   LT
#  1  R     6         48          4800.0       0     D     48
#  2  R     5       4096      25000000.0       2     S    128
#  3  W     4       4096      25000000.0       2     S    128
#  4  W     3      65536       1150000.0       0     D     64
#
# Simulation time (clock cycles):   1000000
#    Time          Load     Bus utilisation
     10000      0.200000     0.000000
     20000      0.200000     0.102400
     30000      0.200000     0.102400
     40000      0.200000     0.102400
     50000      0.200000     0.089300
     60000      0.200000     0.013100
     70000      0.200000     0.102400
     80000      0.200000     0.102400
     90000      0.200000     0.000000
    100000      0.200000     0.102400
    110000      0.200000     0.102400
    120000      0.200000     0.060900
    130000      0.200000     0.076700
    140000      0.200000     0.067200
    150000      0.200000     0.102400
    160000      0.200000     0.102400
    170000      0.200000     0.000000
    180000      0.200000     0.102400
    190000      0.200000     0.102400
    200000      0.200000     0.004900
    210000      0.200000     0.097500
       :
       :
```

Figure 6: Part of a gnuplot input file for a data rate plot

5.4 Gnuplot input for histogram

If the user requests that a file containing data for a histogram be produced, then an output file containing data suitable for reading into gnuplot will be generated. As in the case of the throughput data file, this file is headed by details of the simulation and of the devices attached to the bus, in the form of gnuplot comments. The actual data to be plotted consist of *(Transfer time, Load, Samples)* triplets, (t, l, n), suitable for plotting as a 3D plot via the gnuplot `splot` command, such that the transfer time, t, will be plotted along the x-axis, load l along the y-axis and the number of samples n along the z-axis. In each triplet, t is given as a normalised transfer time (whose unit is the minimum transfer time if no pre-emption or wait states occur), l is the fraction of the maximum data rate generated by each device (in the interval $[0; 1]$), and n is the number of samples observed for this transfer time and load.

An example of the beginning of such a file is shown in Figure 7. In this example, since there are several devices, the file contains several *Samples* columns, one for each device. When displaying the file, the user must select the desired samples column, as illustrated in the gnuplot commands in the next paragraph.

If a histogram file, say `histo.dat`, is generated, gnuplot is automatically activated at the end of the simulation. A suitable sequence of commands for setting up gnuplot to display the data rate plot is given in Appendix C. If these commands are stored in a file `histo.gnp`, the plot can be set up by using the gnuplot `load` command, and displayed by using the `splot` command. For example:

```
gnuplot> load  "histo.gnp"
gnuplot> splot "histo.dat" using 1:2:4 notitle
```

In this example, the `using 1:2:4` parameter to the `splot` command specifies that time values are selected from column 1, load from column 2 and samples from column 4 of the file. Transfer times for several devices can be plotted on the same histogram by using several file/using specifications in the `splot` command. For example:

```
gnuplot> load  "histo.gnp"
gnuplot> splot "histo.dat" using 1:2:3, "histo.dat" using 1:2:6
```

gives the histograms for devices 1 (column 3) and 4 (column 6). For further details, consult the gnuplot documentation (use the gnuplot command `help plot datafile using`.)

As in the case of the throughput and data rate plots, this will by default display the plot in a separate window. For the example given in Figure 7, and using an `splot` command

```
# Transfer time histogram
# ======================
# No. of masters:                   1
# Bus frequency (MHz):              33.0
# Size of data objects (bytes):     4
# Arbitration scheme:               Fixed
#
# Devices attached as masters:
#    R/W  Prio   Block size     Data rate   Max. WS  S/D   LT
#  1  R     6         48          4800.0       0     D     48
#  2  R     5       4096      25000000.0       2     S    128
#  3  W     4       4096      25000000.0       2     S    128
#  4  W     3      65536       1150000.0       0     D     64
#
# Simulation time (clock cycles):   10000000
#Transfer time    Load     Samples
  0.100000      0.200000        0        0        0        0
  0.200000      0.200000        0        0        0        0
  0.300000      0.200000        0        0        0        0
  0.400000      0.200000        0        0        0        0
  0.500000      0.200000        0        0        0        0
  0.600000      0.200000        0        0        0        0
  0.700000      0.200000        0        0        0        0
  0.800000      0.200000        0        0        0        0
  0.900000      0.200000        0        0        0        0
  1.000000      0.200000        6        0        0        0
  1.100000      0.200000        0        0        0        0
  1.200000      0.200000        0        0        0        1
  1.300000      0.200000        0        0        0        0
  1.400000      0.200000        0        0        0        0
  1.500000      0.200000        0        0        0        0
  1.600000      0.200000        0        0        0        0
  1.700000      0.200000        0        0        0        0
  1.800000      0.200000        0        0        0        0
  1.900000      0.200000        0        4        3        0
  2.000000      0.200000        0      362      363        0
  2.100000      0.200000        0        4        4        0
  2.200000      0.200000        0        0        0        0
  2.300000      0.200000        0        0        0        0
  2.400000      0.200000        0        0        0        0
      :
      :
```

Figure 7: Part of a gnuplot input file for a histogram

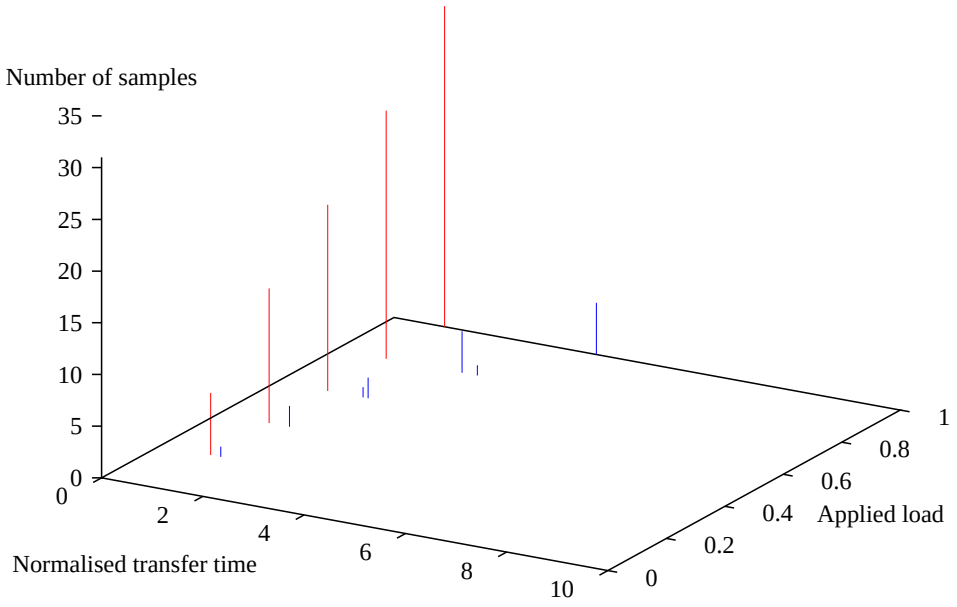

Figure 8: Example of transfer time histogram

to select the histograms for devices 1 and 4 as shown above, the appearance of the plot is shown in Figure 8. The red histogram is for device 1 (column 3), and the blue histogram for device 4 (column 6). These particular histograms demonstrate, for example, how the transfer time for the highest priority device (device 1) remains constant at its minimum value (normalised transfer time = 1.0) as the load on the system increases, whereas the normalised transfer time for the lowest priority device (device 4) is systematically degraded to around 4.0 at the maximum load.

6 Installing and running the simulator

The simulator can be run in a standard Red Hat Linux environment in which gnuplot has been installed. The program is a standard ANSI C program, which can be compiled by using the command:

```
g++ -o pcisim pcimodel.c
```

The simulator can then be running by using the command `pcisim`. No command line parameters are required.

Before running the simulator, you should determine where `gnuplot` is installed in your system. If this is `/usr/bin/gnuplot`, no further action is required. Otherwise, an environment variable `GNUPLOT` should be set to the path of the `gnuplot` program. The way to do this depends on the shell which you are using.

References

[1] PCI Special Interest Group. *PCI Local Bus Specification, revision 2.2*, February 1999.

A A Gnuplot parameter file for throughput plots

```
set terminal x11
set output
set noclip points
set clip one
set noclip two
set border
set boxwidth
set dummy x,y
set format x "%g"
set format y "%g"
set format z "%g"
set grid
set key
set nolabel
set noarrow
set nologscale
set offsets 0, 0, 0, 0
set nopolar
set angles radians
set noparametric
set view 60, 30, 1, 1
set samples 100, 100
set isosamples 10, 10
set surface
set nocontour
set clabel
set nohidden3d
set cntrparam order 4
set cntrparam linear
set cntrparam levels auto 5
set cntrparam points 5
set size 1,1
set data style points
set function style lines
set xzeroaxis
set yzeroaxis
set tics in
set ticslevel 0.5
set xtics
set ytics
```

```
set ztics
set title "" 0,0
set notime
set rrange [-0 : 10]
set trange [-5 : 5]
set urange [-5 : 5]
set vrange [-5 : 5]
set xlabel "Generated data (bytes)" 0,0
set ylabel "Transmitted data (bytes)" 0,0
set zlabel "" 0,0
set zrange [-10 : 10]
set autoscale r
set autoscale t
set autoscale xy
set autoscale z
set zero 1e-08
```

B A Gnuplot parameter file for data rate plots

```
set terminal x11
set output
set noclip points
set clip one
set noclip two
set border
set boxwidth
set dummy x,y
set format x "%g"
set format y "%g"
set format z "%g"
set key 4,6.5,0
set nolabel
set noarrow
set nologscale
set offsets 0, 0, 0, 0
set nopolar
set angles radians
set parametric
set view 60, 30, 1, 1
set samples 100, 100
```

```
set isosamples 10, 10
set surface
set nocontour
set clabel
set nohidden3d
set cntrparam order 4
set cntrparam linear
set cntrparam levels auto 5
set cntrparam points 5
set size 1,1
set data style lines
set function style lines
set tics out
set ticslevel 0.0
set xtics 0, 2000000
set ytics 0.0, 0.2
set ztics
set grid
set xzeroaxis
set yzeroaxis
set title "" 0,0
set notime
set rrange [-0 : 10]
set trange [0 : 5]
set urange [0 : 5]
set vrange [0 : 5]
set xlabel "Time (clock cycles)" -10,0
set xrange [0 : 1000000]
set ylabel "Applied load" 0,0
set zlabel "Bus utilisation" 0,0
set autoscale r
set autoscale t
set autoscale xy
set yrange [0.0 : 1.0]
set autoscale z
set zrange [0.0 : 1.0]
set zero 1e-08
```

C A Gnuplot parameter file for histogram plots

```
set terminal x11
set output
set noclip points
set clip one
set noclip two
set border
set boxwidth
set dummy x,y
set format x "%g"
set format y "%g"
set format z "%g"
set key 4,6.5,0
set nolabel
set noarrow
set nologscale
set offsets 0, 0, 0, 0
set nopolar
set angles radians
set parametric
set view 60, 30, 1, 1
set samples 100, 100
set isosamples 10, 10
set surface
set nocontour
set clabel
set nohidden3d
set cntrparam order 4
set cntrparam linear
set cntrparam levels auto 5
set cntrparam points 5
set size 1,1
set data style impulses
set function style impulses
set tics out
set ticslevel 0.0
set xtics 0, 2
set ytics 0.0, 0.2
set ztics
set grid
set xzeroaxis
```

```
set yzeroaxis
set title "" 0,0
set notime
set rrange [-0 : 10]
set trange [0 : 5]
set urange [0 : 5]
set vrange [0 : 5]
set xlabel "Normalised transfer time" -10,0
set ylabel "Applied load" 0,0
set zlabel "Number of samples" 0,0
set autoscale r
set autoscale t
set autoscale xy
set yrange [0.0 : 1.0]
set xrange [0.0 : 10.0]
set autoscale z
set zero 1e-08
```