virtutech

Simics x86-440BX Target Guide

Simics Version	3.0
Revision	1406
Date	2008-02-20

VIRTUTECH CONFIDENTIAL

Contents

Chapter 1

About Simics Documentation

1.1 Conventions

Let us take a quick look at the conventions used throughout the Simics documentation. Scripts, screen dumps and code fragments are presented in a `monospace` font. In screen dumps, user input is always presented in bold font, as in:

```
Welcome to the Simics prompt
simics> this is something that you should type
```

Sometimes, artificial line breaks may be introduced to prevent the text from being too wide. When such a break occurs, it is indicated by a small arrow pointing down, showing that the interrupted text continues on the next line:

```
This is an artificial ↵
line break that shouldn't be there.
```

The directory where Simics is installed is referred to as `[simics]`, for example when mentioning the `[simics]/README` file. In the same way, the shortcut `[workspace]` is used to point at the user's workspace directory.

1.2 Simics Guides and Manuals

Simics comes with several guides and manuals, which will be briefly described here. All documentation can be found in `[simics]/doc` as Windows Help files (on Windows), HTML files (on Unix) and PDF files (on both platforms). The new Eclipse-based interface also includes Simics documentation in its own help system.

Simics Installation Guide for Unix and for Windows

These guides describe how to install Simics and provide a short description of an installed Simics package. They also cover the additional steps needed for certain features of Simics to work (connection to real network, building new Simics modules, . . .).

Simics User Guide for Unix and for Windows

These guides focus on getting a new user up to speed with Simics, providing information on Simics features such as debugging, profiling, networks, machine configuration and scripting.

Simics Eclipse User Guide

This is an alternative User Guide describing Simics and its new Eclipse-based graphical user interface.

Simics Target Guides

These guides provide more specific information on the different architectures simulated by Simics and the example machines that are provided. They explain how the machine configurations are built and how they can be changed, as well as how to install new operating systems. They also list potential limitations of the models.

Simics Programming Guide

This guide explains how to extend Simics by creating new devices and new commands. It gives a broad overview of how to work with modules and how to develop new classes and objects that fit in the Simics environment. It is only available when the DML add-on package has been installed.

DML Tutorial

This tutorial will give you a gentle and practical introduction to the Device Modeling Language (DML), guiding you through the creation of a simple device. It is only available when the DML add-on package has been installed.

DML Reference Manual

This manual provides a complete reference of DML used for developing new devices with Simics. It is only available when the DML add-on package has been installed.

Simics Reference Manual

This manual provides complete information on all commands, modules, classes and haps implemented by Simics as well as the functions and data types defined in the Simics API.

Simics Micro-Architectural Interface

This guide describes the cycle-accurate extensions of Simics (Micro-Architecture Interface or MAI) and provides information on how to write your own processor timing models. It is only available when the DML add-on package has been installed.

RELEASENOTES and LIMITATIONS files

These files are located in Simics's main directory (i.e., `[simics]`). They list limitations, changes and improvements on a per-version basis. They are the best source of information on new functionalities and specific bug fixes.

Simics Technical FAQ

This document is available on the Virtutech website at `http://www.simics.net/support`. It answers many questions that come up regularly on the support forums.

Simics Support Forum

The Simics Support Forum is the main support tool for Simics. You can access it at `http://www.simics.net`.

Other Interesting Documents

Simics uses Python as its main script language. A Python tutorial is available at `http://www.python.org/doc/2.4/tut/tut.html`. The complete Python documentation is located at `http://www.python.org/doc/2.4/`.

Chapter 2

Simics/x86 Overview

2.1 Introduction

Simics x86-440bx can model various PC systems with x86 or AMD64 processors based on the 440BX chipset. Legacy systems based on ISA are modelled by Simics/x86-PC-AT. Supported Intel processors range from 486sx to Pentium 4 with EM64T. There is also support for a generic AMD64 processor with some K8-like features, but still using the Intel 440BX chipset.

Several operating systems, including versions of Windows, Linux, and NetBSD have been installed and run successfully on Simics.

Note that the same basic machine configuration is used for all the available target processors. Some operating systems might be confused to see machines that cannot exist in reality, such as a 12 processor 486sx system, or an AMD64 machine with an Intel chipset.

With Simics comes a custom BIOS with full support for multiprocessing according to the MPS and ACPI specifications. The custom BIOS supports up to 15 processors, but note that most Linux versions are limited to 8 processors, and that Windows has different limits depending on which flavour of Windows it is.

Virtutech provides hard-disk dumps for some Linux versions on x86. Windows dumps are not available from Virtutech, but it is possible to create disk dumps by installing Windows on top of Simics. More information on how to install an OS on Simics is available in chapter 6.

2.2 Simulated Hardware

Not all supported hardware models are available in every Simics distribution. Contact your Virtutech sales representative if you are interested in hardware that is not included in your distribution.

Processors

- Intel 80386DX with optional 80387 coprocessor
- Intel 80486SX
- Intel 80486DX2
- Intel Pentium

- Intel Pentium MMX
- Intel Pentium Pro
- Intel Pentium II
- Intel Pentium III
- Intel Pentium 4 with optional support for hyper-threading technology, EM64T, and SSE3.
- AMD Opteron with optional support for SSE3.

PC architecture devices

- Keyboard and mouse (8042)
- Floppy controller and connected drives (82077)
- DMA controller (two 8237)
- Programmable interval timer (8254)
- Programmable interrupt controller (two 8259)
- Real time clock (DS12887)
- Advanced programmable interrupt controller (APIC)

Chipset ASICs

- North bridge 82443BX and south bridge PIIX4 (440BX)
- North bridge 82437FX and south bridge PIIX (430FX)
- South bridge MD1535D (experimental)
- FDC37C665GT (for 80486 systems)
- Simulator specific shadow memory controller
- I/O-APIC

ISA devices

- Standard VGA
- Sound Blaster 16 (experimental)
- Lance ethernet controller (AM79C960)
- IDE controller
- Serial port (NS16450 and NS16550)

PCI devices

- BCM5703C Gigabit Ethernet Controller
- BCM5704C Dual Gigabit Ethernet Controller
- Intel i82543 (Intel PRO) Gigabit Ethernet Controller
- Intel i82546 (Intel PRO) Dual Gigabit Ethernet Controller
- ISP1040 SCSI controller

- ISP2200 Fibre-Channel SCSI Controller
- SymbiosLogic SYM53C810 PCI-SCSI I/O Processor
- SymbiosLogic SYM53C875 PCI-SCSI I/O Processor
- SymbiosLogic SYM53C876 PCI-SCSI I/O Processor
- DEC21041 PCI Ethernet LAN Controller
- DEC21140A PCI Fast Ethernet LAN Controller
- DEC21143 PCI Ethernet LAN Controller
- ATI Rage XL/XC (Mach64) video device
- 3Dfx Voodoo3 graphics card (both PCI and AGP versions)

Chapter 3

Simulated Machines

Simics scripts for starting x86-440bx machines are located in the `[workspace]/targets/x86-440bx/` directory, while the actual configuration scripts can be found in `[simics]/targets/x86-440bx/`.

3.1 Tango

Tango has Fedora Core 5 installed. The base configuration has a single 20 MHz Pentium 4 processor, 256 MB memory, one 19GB IDE disk and one IDE CD-ROM. There is also an AGP based Voodoo3 graphics card and a PCI based DEC21143 network adapter. The default configuration can be modified as described in section 3.5.

Additional information:

- Fedora Core 5. Installed on Simics with developer support with several services disabled.
- Linux kernel 2.6.15, including SMP support.
- SimicsFS support (read-only, write support experimental).
- Login `root`, password "simics".
- X11 configured for Voodoo 3 in 800x600 24-bit.
- Configured to get IP address using DHCP.

3.1.1 Tango Scripts

`tango-common.simics`
Starts the Tango machine with the default configuration.

`tango-gcache-common.simics`
Default Tango machine with a *g-cache* cache model connected.

`tango-ma-common.simics`
Default Tango machine with a simple processor timing model connected. Simics must be stared in MAI mode (-ma) to run this script.

`tango-multi.simics`
Example script with two Tango machines in the same session, connected by an Ethernet link. Note that the graphics console module for Windows host machines can only display one console, meaning that this script cannot be used on windows unless video text consoles are used. Video text consoles are enabled with the `$text_console` variable (see section 3.5).

3.2 Enterprise

Enterprise has Red Hat Linux 7.3 installed. The base configuration has a single 20 MHz Pentium 4 processor, 256 MB memory, one 19GB IDE disk and one IDE CD-ROM. There is also an AGP based Voodoo3 graphics card and an ISA based LANCE network adapter. The default configuration can be modified as described in section 3.5.

Additional information:

- Red Hat Linux 7.3. Installed on Simics as Workstation with no firewall, containing the KDE and Software Developer package groups.
- Linux kernel 2.4.18, including SMP support.
- SimicsFS support (read-only, write support experimental).
- Login `root`, no password.
- X11 configured for Voodoo 3 in 800x600 32-bit.
- Configured with static IP address 10.10.0.15, gateway 10.10.0.1, when DHCP not used.

3.2.1 Enterprise Scripts

`enterprise-common.simics`
Starts the Enterprise machine with the default configuration.

`enterprise-dhcp-common.simics`
Similar to enterprise-common.simics, but gets the host name and IP address from the DHCP server.

`enterprise-gcache-common.simics`
Default Enterprise machine with a *g-cache* cache model connected.

`enterprise-ma-common.simics`
Default Enterprise machine with a simple processor timing model connected. Simics must be stared in MAI mode (-ma) to run this script.

`enterprise-multi.simics`
Example script with two Enterprise machines in the same session, connected by an Ethernet link. Note that the graphics console module for Windows host machines can only display one console, meaning that this script cannot be used on windows unless video text consoles are used. Video text consoles are enabled with the `$text_console` variable (see section 3.5).

3.3 Hippie

Hippie includes a standard Red Hat Linux 6.2 install. In the default configuration hippie has a single 20MHz Pentium 4 processor, 256 MB memory, one 1GB IDE disk and one IDE CD-ROM. There is also a PCI based Voodoo3 graphics card and an ISA based LANCE network adapter. The default configuration can be modified as described in section 3.5.

Additional information:

- Red Hat Linux 6.2, installed on Simics as "KDE Workstation".
- Linux kernel 2.2.14, including SMP support.
- SimicsFS support.
- Login `root`, no password.
- X11 configured for Voodoo 3 in 800x600 32-bit.
- Configured with static IP address 10.10.0.9, gateway 10.10.0.1, when DHCP not used.

3.3.1 Hippie Scripts

`hippie-common.simics`
Starts the Hippie machine with the default configuration.

`hippie-dhcp-common.simics`
Similar to hippie-common.simics, but gets the host name and IP address from the DHCP server.

`hippie-gcache-common.simics`
Default Hippie machine with a *g-cache* cache model connected.

3.4 Dredd

Dredd ships without software, and is intended to be used for installing new OSes on Simics.

In the default configuration dredd has a single 20MHz Pentium 4 processor, 256 MB memory, one 19GB IDE disk and one IDE CD-ROM. There is also an AGP based Voodoo3 graphics card and a PCI based DEC21143 network adapter. The default configuration can be modified as described in section 3.5.

Information about how to install an OS in described in more detail in chapter 6.

3.4.1 Dredd Scripts

`dredd-cd-install1.simics`
Script for installing an OS on Dredd in Simics, phase 1.

`dredd-cd-install2.simics`
Script for installing an OS on Dredd in Simics, phase 2.

dredd-common.simics
Starts the Dredd machine with the default configuration. Can not be run until after an OS has been installed on Simics.

dredd-gcache-common.simics
Default Dredd machine with a *g-cache* cache model connected.

3.5 Parameters for Machine Scripts

The following parameters can be set before running the `tango-common.simics`, `enterprise-common.simics`, `hippie-common.simics`, or `dredd-common.simics` scripts. Other `.simics` scripts may set some of the parameters unconditionally, and do not allow the user to override them. For example, the `enterprise-dhcp-common.simics` script will always set the *$create_network* variable to `yes`.

Parameters can be specified directly on the Simics command line with the run expression (-e) switch:

```
$ ./simics -e '$cpu_class="pentium-ii"' targets/x86-440bx/enterprise-common.s
```

They can also be initialized within Simics before the base script is run:

```
$ ./simics
simics> $cpu_class="pentium-ii"
simics> run-command-file "targets/x86-440bx/enterprise-common.simics"
```

If you are using the wxWindows frontend on Windows, you will need to create a script that both sets the parameter and includes the base script.

3.5.1 tango-common, enterprise-common, hippie-common and dredd-common

$create_network
Set to `yes` if the script should create an Ethernet link and connect the primary Ethernet adapter to it.

$cpu_class
The type of processor to create. Should be one of `pentium`, `pentium-mmx`, `pentium-pro`, `pentium-ii`, `pentium-iii`, `pentium-4`, `pentium-4e`, `pentium-4e-2ht`, `pentium-4e-4ht` and `x86-hammer`.

$disk_size
Size of the primary hard disk. This parameter must match any disk images that are added to the primary disk.

$enter_in_boot_menu
Set to `no` to prevent the script from pressing enter in the boot menu, default is `yes`. This option does not apply to the dredd machine.

$eth_link
The Ethernet link to connect the primary Ethernet adapter to. This parameter should be set when a link already exist and the *$create_network* parameter is `no`.

$freq_mhz
The clock frequency in MHz for all processors.

$host_name
The host name used by the DHCP and DNS servers for this machine. This variable will not change the host name set for the machine on the disk dumps.

$ip_address
The IP address used by the DHCP and DNS servers for this machine. This variable will not change any IP address set for the machine on the disk dumps.

$mac_address
MAC address of the primary Ethernet adapter.

$memory_megs
The total amount of system memory, in MB, in the system.

$num_cpus
The number of processors in the machine.

$rtc_time
Date and time of the real-time clock at boot.

$service_node
The *service node* to use for DHCP and DNS. This parameter should be set when a service noce already exist and the *$create_network* parameter is `no`.

$text_console
Set to `yes` to connect a text console to the VGA display instead of a graphical console that is default.

Chapter 4

Supported Components

The following sections list components that are supported for the x86-440bx architecture. There also exist other components in Simics, such as various PCI devices, that may work for x86-440bx but that have not been tested.

The default machines are constructed from components in the `-system.include` files in `[simics]/targets/x86-440bx/`. See the Configuration and Checkpointing chapter in the Simics User Guide for information on how to define your own machine, or make modifications to an existing machine.

4.1 x86-440BX Components

4.1.1 x86-apic-system

Description

The "x86-apic-system" component represents a multi-processor capable x86 system with up to 255 cpus.

Attributes

acpi

Optional attribute; **read/write** access; type: **b**.

TRUE if the machine uses ACPI. Default is TRUE.

bios

Optional attribute; **read/write** access; type: **String**.

The x86 BIOS file to use.

break_on_reboot

Optional attribute; **read/write** access; type: **b**.

If true, the simulation will stop when machine is rebooted.

memory_megs

Required attribute; **read/write** access; type: **Integer**.

The amount of RAM in megabytes on the processor board.

rtc_time
Required attribute; **read/write** access; type: **String**.
The date and time of the Real-Time clock.

Commands

create-x86-apic-system [*"name"*] *memory_megs "rtc_time"* [*break_on_reboot*] [*"bios"*] [*acpi*]

Creates a non-instantiated component of the class "x86-apic-system". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**x86-apic-system**>**.cmos-base-mem** ***kilobytes***
Sets the amount of base memory (in kB).
This command will update the proper location in the CMOS so that the BIOS will know how much memory is installed in the system. Operating system that use the BIOS to find out the memory size will get confused if this size is set incorrectly (especially if it is set too high). The maximum amount that can be set is 640kB.

<**x86-apic-system**>**.cmos-boot-dev** ***"drive"***
Specifies boot device for the BIOS in the CMOS.
Parameters: A or C, or floppy or HD boot. Default is C.

<**x86-apic-system**>**.cmos-extended-mem** ***megabytes***
Sets the amount of extended memory (in MB).
This command will update the proper location in the CMOS so that the BIOS will know how much memory is installed in the system. Operating system that use the BIOS to find out the memory size will get confused if this size is set incorrectly (especially if it is set too high). The maximum amount that can be set is 63MB.

<**x86-apic-system**>**.cmos-floppy** ***"drive" "type"***
Sets information in the CMOS about floppy drives.
Drive is either A (primary drive) or B (secondary drive), and type is the maximal drive size (in kB or MB); 360, 720, 1.2, 1.44, 2.88. Setting *type* to "none" indicates to the OS/BIOS that no drive is present. Since both arguments are strings, quoting is sometimes necessary.

<**x86-apic-system**>**.cmos-hd** ***"drive" cylinders heads sectors_per_track***
Sets information in the CMOS about the primary hard disk ("C") and the secondary hard disk ("D").

<x86-apic-system>.cmos-info
Print detailed information about the CMOS information from the RTC device.

<x86-apic-system>.cmos-init
Set initial CMOS values in the RTC device.
This is miscellaneous data that is not set by any of the other cmos-∗ commands. Note that the CMOS values only has to be set if not running from a saved configuration. A saved configuration will have all values stored in the NVRAM area, and the cmos-∗ commands need only be used if some values have to be changed.

<x86-apic-system>.info
Print detailed information about the configuration of the device.

<x86-apic-system>.status
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
chipset	x86-chipset	down
cpu0	x86-apic-processor	down
cpu[1-254]	x86-apic-processor	down
interrupt	sb-interrupt	down
reset	x86-reset-bus	down

4.1.2 pentium-cpu

Description
The "pentium-cpu" component represents an Intel Pentium processor.

Attributes

apic_frequency
Optional attribute; **read/write** access; type: **Integer**.
APIC bus frequency in MHz, default is 10 MHz.

cpi
Optional attribute; **read/write** access; type: **Integer**.
Cycles per instruction.

cpu_frequency
Required attribute; **read/write** access; type: **Integer**.
Processor frequency in MHz.

cpu_threads
Pseudo attribute; **read-only** access; type: **Integer**.
The number of hyper threads in the processor.

Commands

create-pentium-cpu [*"name"*] *cpu_frequency* [*cpi*] [*apic_frequency*]
Creates a non-instantiated component of the class "pentium-cpu". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**pentium-cpu**>**.info**
Print detailed information about the configuration of the device.

<**pentium-cpu**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
system	x86-apic-processor	up
cache-cpu	timing-model	down
direct-pins	x86-pins	down

4.1.3 pentium-mmx-cpu

Description
The "pentium-mmx-cpu" component represents an Intel Pentium MMX processor.

Attributes

apic_frequency
Optional attribute; **read/write** access; type: **Integer**.
APIC bus frequency in MHz, default is 10 MHz.

cpi
Optional attribute; **read/write** access; type: **Integer**.
Cycles per instruction.

cpu_frequency
Required attribute; **read/write** access; type: **Integer**.
Processor frequency in MHz.

cpu_threads
Pseudo attribute; **read-only** access; type: **Integer**.
The number of hyper threads in the processor.

Commands

create-pentium-mmx-cpu [*"name"*] *cpu_frequency* [*cpi*] [*apic_frequency*]
Creates a non-instantiated component of the class "pentium-mmx-cpu". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**pentium-mmx-cpu**>**.info**
Print detailed information about the configuration of the device.

<**pentium-mmx-cpu**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
system	x86-apic-processor	up
cache-cpu	timing-model	down
direct-pins	x86-pins	down

4.1.4 pentium-pro-cpu

Description
The "pentium-pro-cpu" component represents an Intel Pentium Pro (P6) processor.

Attributes

apic_frequency
Optional attribute; **read/write** access; type: **Integer**.
APIC bus frequency in MHz, default is 10 MHz.

cpi
Optional attribute; **read/write** access; type: **Integer**.
Cycles per instruction.

cpu_frequency
Required attribute; **read/write** access; type: **Integer**.
Processor frequency in MHz.

cpu_threads
Pseudo attribute; **read-only** access; type: **Integer**.
The number of hyper threads in the processor.

Commands

create-pentium-pro-cpu [*"name"*] *cpu_frequency* [*cpi*] [*apic_frequency*]
Creates a non-instantiated component of the class "pentium-pro-cpu". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**pentium-pro-cpu**>**.info**
Print detailed information about the configuration of the device.

<**pentium-pro-cpu**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	**Type**	**Direction**
system	x86-apic-processor	up
cache-cpu	timing-model	down
direct-pins	x86-pins	down

4.1.5 pentium-ii-cpu

Description
The "pentium-ii-cpu" component represents an Intel Pentium II processor.

Attributes

apic_frequency
Optional attribute; **read/write** access; type: **Integer**.
APIC bus frequency in MHz, default is 10 MHz.

cpi
Optional attribute; **read/write** access; type: **Integer**.
Cycles per instruction.

cpu_frequency
Required attribute; **read/write** access; type: **Integer**.
Processor frequency in MHz.

cpu_threads
Pseudo attribute; **read-only** access; type: **Integer**.
The number of hyper threads in the processor.

Commands

create-pentium-ii-cpu [*"name"*] *cpu_frequency* [*cpi*] [*apic_frequency*]
Creates a non-instantiated component of the class "pentium-ii-cpu". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**pentium-ii-cpu**>**.info**
Print detailed information about the configuration of the device.

<**pentium-ii-cpu**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	**Type**	**Direction**
system	x86-apic-processor	up
cache-cpu	timing-model	down
direct-pins	x86-pins	down

4.1.6 pentium-iii-cpu

Description
The "pentium-iii-cpu" component represents an Intel Pentium III processor.

Attributes

apic_frequency
Optional attribute; **read/write** access; type: **Integer**.
APIC bus frequency in MHz, default is 10 MHz.

cpi
Optional attribute; **read/write** access; type: **Integer**.
Cycles per instruction.

cpu_frequency
Required attribute; **read/write** access; type: **Integer**.
Processor frequency in MHz.

cpu_threads
Pseudo attribute; **read-only** access; type: **Integer**.
The number of hyper threads in the processor.

Commands

create-pentium-iii-cpu [*"name"*] *cpu_frequency* [*cpi*] [*apic_frequency*]
Creates a non-instantiated component of the class "pentium-iii-cpu". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**pentium-iii-cpu**>**.info**
Print detailed information about the configuration of the device.

<**pentium-iii-cpu**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
system	x86-apic-processor	up
cache-cpu	timing-model	down
direct-pins	x86-pins	down

4.1.7 pentium-4-cpu

Description
The "pentium-4-cpu" component represents an Intel Pentium 4 processor.

Attributes

apic_frequency
Optional attribute; **read/write** access; type: **Integer**.
APIC bus frequency in MHz, default is 10 MHz.

cpi
Optional attribute; **read/write** access; type: **Integer**.
Cycles per instruction.

cpu_frequency
Required attribute; **read/write** access; type: **Integer**.
Processor frequency in MHz.

cpu_threads
Pseudo attribute; **read-only** access; type: **Integer**.
The number of hyper threads in the processor.

Commands

create-pentium-4-cpu [*"name"***] *cpu_frequency* [*cpi*] [*apic_frequency*]**
Creates a non-instantiated component of the class "pentium-4-cpu". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**pentium-4-cpu**>**.info**
Print detailed information about the configuration of the device.

<**pentium-4-cpu**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
system	x86-apic-processor	up
cache-cpu	timing-model	down
direct-pins	x86-pins	down

4.1.8 pentium-4e-cpu

Description
The "pentium-4e-cpu" component represents an Intel 64-bit Pentium 4E processor.

Attributes

apic_frequency
Optional attribute; **read/write** access; type: **Integer**.
APIC bus frequency in MHz, default is 10 MHz.

cpi
Optional attribute; **read/write** access; type: **Integer**.
Cycles per instruction.

cpu_frequency
Required attribute; **read/write** access; type: **Integer**.
Processor frequency in MHz.

cpu_threads
Pseudo attribute; **read-only** access; type: **Integer**.
The number of hyper threads in the processor.

Commands

create-pentium-4e-cpu [*"name"*] *cpu_frequency* [*cpi*] [*apic_frequency*]
Creates a non-instantiated component of the class "pentium-4e-cpu". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**pentium-4e-cpu**>**.info**
Print detailed information about the configuration of the device.

<**pentium-4e-cpu**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
system	x86-apic-processor	up
cache-cpu	timing-model	down
direct-pins	x86-pins	down

4.1.9 pentium-4e-2ht-cpu

Description
The "pentium-4e-2ht-cpu" component represents an Intel 64-bit Pentium 4E processor, with 2 hyper threads.

Attributes

apic_frequency
Optional attribute; **read/write** access; type: **Integer**.
APIC bus frequency in MHz, default is 10 MHz.

cpi
Optional attribute; **read/write** access; type: **Integer**.
Cycles per instruction.

cpu_frequency
Required attribute; **read/write** access; type: **Integer**.
Processor frequency in MHz.

cpu_threads
Pseudo attribute; **read-only** access; type: **Integer**.
The number of hyper threads in the processor.

Commands

create-pentium-4e-2ht-cpu [*"name"*] *cpu_frequency* [*cpi*] [*apic_frequency*]
Creates a non-instantiated component of the class "pentium-4e-2ht-cpu". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**pentium-4e-2ht-cpu**>**.info**
Print detailed information about the configuration of the device.

<**pentium-4e-2ht-cpu**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
system	x86-apic-processor	up
cache-cpu[0-1]	timing-model	down
direct-pins	x86-pins	down

4.1.10 pentium-4e-4ht-cpu

Description
The "pentium-4e-4ht-cpu" component represents an Intel 64-bit Pentium 4E processor, with 4 hyper threads.

Attributes

apic_frequency
Optional attribute; **read/write** access; type: **Integer**.
APIC bus frequency in MHz, default is 10 MHz.

cpi
Optional attribute; **read/write** access; type: **Integer**.
Cycles per instruction.

cpu_frequency
Required attribute; **read/write** access; type: **Integer**.
Processor frequency in MHz.

cpu_threads
Pseudo attribute; **read-only** access; type: **Integer**.
The number of hyper threads in the processor.

Commands

create-pentium-4e-4ht-cpu [*"name"*] *cpu_frequency* [*cpi*] [*apic_frequency*]
Creates a non-instantiated component of the class "pentium-4e-4ht-cpu". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**pentium-4e-4ht-cpu**>**.info**
Print detailed information about the configuration of the device.

<**pentium-4e-4ht-cpu**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
system	x86-apic-processor	up
cache-cpu[0-3]	timing-model	down
direct-pins	x86-pins	down

4.1.11 x86-hammer-cpu

Description
The "x86-hammer-cpu" component represents a generic 64-bit AMD Athlon 64 or Opteron processor without on-chip devices.

Attributes

apic_frequency
Optional attribute; **read/write** access; type: **Integer**.
APIC bus frequency in MHz, default is 10 MHz.

cpi
Optional attribute; **read/write** access; type: **Integer**.
Cycles per instruction.

cpu_frequency
Required attribute; **read/write** access; type: **Integer**.
Processor frequency in MHz.

cpu_threads
Pseudo attribute; **read-only** access; type: **Integer**.
The number of hyper threads in the processor.

Commands

create-x86-hammer-cpu [*"name"*] *cpu_frequency* [*cpi*] [*apic_frequency*]
Creates a non-instantiated component of the class "x86-hammer-cpu". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**x86-hammer-cpu**>**.info**
Print detailed information about the configuration of the device.

<**x86-hammer-cpu**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
system	x86-apic-processor	up
cache-cpu	timing-model	down
direct-pins	x86-pins	down

4.1.12 north-bridge-443bx

Description
The "north-bridge-443bx" component represents an Intel 443BX North Bridge (host-to-PCI bridge) without AGP.

Commands

create-north-bridge-443bx [*"name"*]
Creates a non-instantiated component of the class "north-bridge-443bx". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**north-bridge-443bx**>**.info**
Print detailed information about the configuration of the device.

<**north-bridge-443bx**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
system	x86-chipset	up
pci-slot[0-23]	pci-bus	down

4.1.13 north-bridge-443bx-agp

Description

The "north-bridge-443bx" component represents an Intel 443BX North Bridge (host-to-PCI bridge) with AGP.

Commands

create-north-bridge-443bx-agp [*"name"*]

Creates a non-instantiated component of the class "north-bridge-443bx-agp". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**north-bridge-443bx-agp**>**.info**

Print detailed information about the configuration of the device.

<**north-bridge-443bx-agp**>**.status**

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
system	x86-chipset	up
agp-slot0	agp-bus	down
pci-slot[0-23]	pci-bus	down

4.1.14 south-bridge-piix4

Description

The "south-bridge-piix4" component represents an Intel PIIX4 PCI south bridge with the common legacy PC devices. It includes two IDE controllers, and a real-time clock. There is also a USB controller that is not supported in Simics.

Commands

create-south-bridge-piix4 [*"name"*]
Creates a non-instantiated component of the class "south-bridge-piix4". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**south-bridge-piix4**>**.info**
Print detailed information about the configuration of the device.

<**south-bridge-piix4**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
interrupt	sb-interrupt	up
pci-bus	pci-bus	up
ide0-master	ide-slot	down
ide0-slave	ide-slot	down
ide1-master	ide-slot	down
ide1-slave	ide-slot	down
isa-bus	isa-bus	down
usb[0-1]	usb-port	down

4.1.15 south-bridge-md1535d

Description
The "south-bridge-md1535d" component represents an ALi MD1535D PCI south bridge with the common legacy PC devices. It includes two IDE controllers, a floppy controller with two drives, three serial ports, and a real-time clock. There is also a USB controller that is not supported in Simics.

Commands

create-south-bridge-md1535d [*"name"*]
Creates a non-instantiated component of the class "south-bridge-md1535d". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**south-bridge-md1535d**>**.info**
Print detailed information about the configuration of the device.

<**south-bridge-md1535d**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
interrupt	sb-interrupt	up
pci-bus-ide	pci-bus	up
pci-bus-isa	pci-bus	up
pci-bus-pm	pci-bus	up
com[1-3]	serial	down
ide0-master	ide-slot	down
ide0-slave	ide-slot	down
ide1-master	ide-slot	down
ide1-slave	ide-slot	down
isa-bus	isa-bus	down

4.2 PCI Device Components

4.2.1 pci-bcm5703c

Description

The "pci-bcm5703c" component represents a Broadcom 5703C PCI based gigabit Ethernet adapter.

Attributes

bios

Optional attribute; **read/write** access; type: **String**.

The x86 BIOS file to use.

mac_address

Required attribute; **read/write** access; type: **String**.

The MAC address of the Ethernet adapter.

Commands

create-pci-bcm5703c [*"name"*] *"mac_address"* [*"bios"*]

Creates a non-instantiated component of the class "pci-bcm5703c". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<pci-bcm5703c>.info

Print detailed information about the configuration of the device.

<pci-bcm5703c>.status

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
ethernet	ethernet-link	down

4.2.2 pci-bcm5704c

Description

The "pci-bcm5704c" component represents a Broadcom 5704C PCI based dual-port gigabit Ethernet adapter.

Attributes

bios

Optional attribute; **read/write** access; type: **String**.
The x86 BIOS file to use.

mac_address0

Required attribute; **read/write** access; type: **String**.
The MAC address of the first Ethernet adapter.

mac_address1

Required attribute; **read/write** access; type: **String**.
The MAC address of the second Ethernet adapter.

Commands

create-pci-bcm5704c [*"name"*] *"mac_address0" "mac_address1"* [*"bios"*]

Creates a non-instantiated component of the class "pci-bcm5704c". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<pci-bcm5704c>.info

Print detailed information about the configuration of the device.

<pci-bcm5704c>.status

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
ethernet[0-1]	ethernet-link	down

4.2.3 pci-i82543gc

Description

The "pci-i82543gc" component represents an Intel 82543 (Intel PRO) PCI based gigabit Ethernet adapter.

Attributes

bios

Optional attribute; **read/write** access; type: **String**.

The x86 BIOS file to use.

mac_address

Required attribute; **read/write** access; type: **String**.

The MAC address of the Ethernet adapter.

Commands

create-pci-i82543gc [*"name"*] *"mac_address"* [*"bios"*]

Creates a non-instantiated component of the class "pci-i82543gc". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<pci-i82543gc>.info

Print detailed information about the configuration of the device.

<pci-i82543gc>.status

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
ethernet	ethernet-link	down

4.2.4 pci-i82546bg

Description

The "pci-i82546bg" component represents an Intel 82546 (Intel PRO) PCI based dual-port gigabit Ethernet adapter.

Attributes

bios

Optional attribute; **read/write** access; type: **String**.

The x86 BIOS file to use.

mac_address

Required attribute; **read/write** access; type: **String**.

The MAC address of the first Ethernet adapter. The last bit is toggled to get the address for the second interface.

Commands

create-pci-i82546bg [*"name"*] *"mac_address"* [*"bios"*]

Creates a non-instantiated component of the class "pci-i82546bg". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<pci-i82546bg>.info

Print detailed information about the configuration of the device.

<pci-i82546bg>.status

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
ethernet[0-1]	ethernet-link	down

4.2.5 pci-isp1040

Description

The "pci-isp1040" component represents an ISP1040 PCI based SCSI controller.

Attributes

bios

Optional attribute; **read/write** access; type: **String**.

The x86 SCSI BIOS file to use.

scsi_id

Optional attribute; **read/write** access; type: **Integer**.

The ID on the SCSI bus.

Commands

create-pci-isp1040 [*"name"*] [*scsi_id*] [*"bios"*]
Creates a non-instantiated component of the class "pci-isp1040". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<pci-isp1040>.info
Print detailed information about the configuration of the device.

<pci-isp1040>.status
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
scsi-bus	scsi-bus	down

4.2.6 pci-isp2200

Description
The "pci-isp2200" component represents an ISP2200 PCI based Fibre-Channel SCSI controller.

Attributes

bios
Optional attribute; **read/write** access; type: **String**.
The x86 SCSI BIOS file to use.

loop_id
Required attribute; **read/write** access; type: **Integer**.
The FC loop ID of the ISP2200 Fibre-Channel controller.

Commands

create-pci-isp2200 [*"name"*] *loop_id* [*"bios"*]
Creates a non-instantiated component of the class "pci-isp2200". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<pci-isp2200>.info
Print detailed information about the configuration of the device.

<pci-isp2200>.status
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
fc-loop	simple-fc-loop	down

4.2.7 pci-sym53c810

Description
The "pci-sym53C810" component represents a SYM53C810PCI based SCSI controller.

Attributes

bios
Optional attribute; **read/write** access; type: **String**.
The x86 SCSI BIOS file to use.

Commands

create-pci-sym53c810 [*"name"*] [*"bios"*]
Creates a non-instantiated component of the class "pci-sym53c810". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<pci-sym53c810>.info
Print detailed information about the configuration of the device.

<pci-sym53c810>.status
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
scsi-bus	scsi-bus	down

4.2.8 pci-sym53c875

Description

The "pci-sym53C875" component represents a SYM53C875PCI based SCSI controller.

Attributes

bios

Optional attribute; **read/write** access; type: **String**.

The x86 SCSI BIOS file to use.

Commands

create-pci-sym53c875 [*"name"*] [*"bios"*]

Creates a non-instantiated component of the class "pci-sym53c875". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**pci-sym53c875**>**.info**

Print detailed information about the configuration of the device.

<**pci-sym53c875**>**.status**

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
scsi-bus	scsi-bus	down

4.2.9 pci-sym53c876

Description

The "pci-sym53C876" component represents a SYM53C876PCI based dual-port SCSI controller.

Commands

create-pci-sym53c876 [*"name"*]

Creates a non-instantiated component of the class "pci-sym53c876". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<pci-sym53c876>.info
Print detailed information about the configuration of the device.

<pci-sym53c876>.status
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
scsi-bus[0-1]	scsi-bus	down

4.2.10 pci-dec21041

Description
The "pci-dec21041" component represents a DEC21041 PCI based fast Ethernet adapter.

Attributes

bios
Optional attribute; **read/write** access; type: **String**.
The x86 BIOS file to use.

mac_address
Required attribute; **read/write** access; type: **String**.
The MAC address of the Ethernet adapter.

Commands

create-pci-dec21041 [*"name"*] *"mac_address"* [*"bios"*]
Creates a non-instantiated component of the class "pci-dec21041". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<pci-dec21041>.info
Print detailed information about the configuration of the device.

<pci-dec21041>.status
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
ethernet	ethernet-link	down

4.2.11 pci-dec21140a

Description

The "pci-dec21140a" component represents a DEC21140A PCI based fast Ethernet adapter.

Attributes

bios

Optional attribute; **read/write** access; type: **String**.

The x86 BIOS file to use.

mac_address

Required attribute; **read/write** access; type: **String**.

The MAC address of the Ethernet adapter.

Commands

create-pci-dec21140a [*"name"*] *"mac_address"* [*"bios"*]

Creates a non-instantiated component of the class "pci-dec21140a". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**pci-dec21140a**>**.info**

Print detailed information about the configuration of the device.

<**pci-dec21140a**>**.status**

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
ethernet	ethernet-link	down

4.2.12 pci-dec21143

Description

The "pci-dec21143" component represents a DEC21143 PCI based fast Ethernet adapter.

Attributes

bios

Optional attribute; **read/write** access; type: **String**.

The x86 BIOS file to use.

mac_address
Required attribute; **read/write** access; type: **String**.
The MAC address of the Ethernet adapter.

Commands

create-pci-dec21143 [*"name"*] *"mac_address"* [*"bios"*]
Creates a non-instantiated component of the class "pci-dec21143". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<pci-dec21143>.info
Print detailed information about the configuration of the device.

<pci-dec21143>.status
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
ethernet	ethernet-link	down

4.2.13 pci-pd6729

Description
The "pci-pd6729" component represents a Cirrus Logic PD6729 PCI-to-PCMCIA (PC-Card) Controller with two slots.

Commands

create-pci-pd6729 [*"name"*]
Creates a non-instantiated component of the class "pci-pd6729". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<pci-pd6729>.info
Print detailed information about the configuration of the device.

<pci-pd6729>.status
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
pcmcia[0-1]	pcmcia-slot	down

4.2.14 pci-ragexl

Description

The "pci-ragexl" component represents a Rage XL PCI based VGA compatible graphics adapter.

Commands

create-pci-ragexl [*"name"*]

Creates a non-instantiated component of the class "pci-ragexl". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<pci-ragexl>.info

Print detailed information about the configuration of the device.

<pci-ragexl>.status

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
console	graphics-console	down

4.2.15 pci-voodoo3

Description

The "pci-voodoo3" component represents a 3dfx Voodoo3 PCI based VGA compatible graphics adapter.

Commands

create-pci-voodoo3 [*"name"*]

Creates a non-instantiated component of the class "pci-voodoo3". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<pci-voodoo3>.info
Print detailed information about the configuration of the device.

<pci-voodoo3>.status
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
console	graphics-console	down

4.2.16 agp-voodoo3

Description
The "pci-voodoo3" component represents a 3dfx Voodoo3 AGP based VGA compatible graphics adapter.

Commands

create-agp-voodoo3 [*"name"*]
Creates a non-instantiated component of the class "agp-voodoo3". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<agp-voodoo3>.info
Print detailed information about the configuration of the device.

<agp-voodoo3>.status
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
agp-bus	agp-bus	up
console	graphics-console	down

4.3 PC Legacy and ISA Components

4.3.1 std-super-io

Description

The "std-super-io" component represents a generic Super I/O device with legacy PC devices such as two serial ports, one PS/2 keyboard and mouse controller, one floppy device and a parallel port

Attributes

add_par_port

Optional attribute; **read/write** access; type: **b**.

Set to TRUE to add a parallel port to the Super I/O device. Default is FALSE since the current implementation is a dummy device.

Commands

create-std-super-io [*"name"*] [*add_par_port*]

Creates a non-instantiated component of the class "std-super-io". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**std-super-io**>**.info**

Print detailed information about the configuration of the device.

<**std-super-io**>**.status**

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
isa-bus	isa-bus	up
reset	x86-reset-bus	up
com[1-2]	serial	down
kbd-console	keyboard	down
mse-console	mouse	down

4.3.2 ps2-keyboard-mouse

Description

The "ps2-keyboard-mouse" component represents the PS/2 8042 keyboard controller with a connected 105 key keyboard and three button mouse.

Commands

create-ps2-keyboard-mouse [*"name"*]
Creates a non-instantiated component of the class "ps2-keyboard-mouse". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**ps2-keyboard-mouse**>**.info**
Print detailed information about the configuration of the device.

<**ps2-keyboard-mouse**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	**Type**	**Direction**
isa-bus	isa-bus	up
reset	x86-reset-bus	up
kbd-console	keyboard	down
mse-console	mouse	down

4.3.3 pc-dual-serial-ports

Description
The "pc-dual-serial-ports" component represents two PC compatible serial ports.

Commands

create-pc-dual-serial-ports [*"name"*]
Creates a non-instantiated component of the class "pc-dual-serial-ports". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**pc-dual-serial-ports**>**.info**
Print detailed information about the configuration of the device.

<**pc-dual-serial-ports**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
isa-bus	isa-bus	up
com[1-2]	serial	down

4.3.4 pc-quad-serial-ports

Description

The "pc-quad-serial-ports" component represents four PC compatible serial ports.

Commands

create-pc-quad-serial-ports [*"name"*]

Creates a non-instantiated component of the class "pc-quad-serial-ports". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**pc-quad-serial-ports**>**.info**

Print detailed information about the configuration of the device.

<**pc-quad-serial-ports**>**.status**

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
isa-bus	isa-bus	up
com[1-4]	serial	down

4.3.5 pc-floppy-controller

Description

The "pc-floppy-controller" component represents a legacy pc floppy controller with two attached drives.

Commands

create-pc-floppy-controller [*"name"*]

Creates a non-instantiated component of the class "pc-floppy-controller". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**pc-floppy-controller**>**.info**
Print detailed information about the configuration of the device.

<**pc-floppy-controller**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
isa-bus	isa-bus	up

4.3.6 isa-vga

Description
The "isa-vga" component represents an ISA bus based VGA compatible graphics adapter.

Attributes

bios
Optional attribute; **read/write** access; type: **String**.
The VGA BIOS file to use.

Commands

create-isa-vga [*"name"*] [*"bios"*]
Creates a non-instantiated component of the class "isa-vga". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**isa-vga**>**.info**
Print detailed information about the configuration of the device.

<**isa-vga**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
isa-bus	isa-bus	up
console	graphics-console	down

4.3.7 isa-lance

Description

The "isa-lance" component represents an ISA bus based Ethernet adapter.

Attributes

base_port

Optional attribute; **read/write** access; type: **Integer**.

The starting port number in I/O space. The default port is 0x300, and the mapping is 0x17 bytes large.

irq_level

Optional attribute; **read/write** access; type: **Integer**.

The interrupt level for the Lance device, default is 7.

mac_address

Required attribute; **read/write** access; type: **String**.

The MAC address of the Ethernet adapter.

Commands

create-isa-lance [*"name"*] *"mac_address"* [*irq_level*] [*base_port*]

Creates a non-instantiated component of the class "isa-lance". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<isa-lance>.info

Print detailed information about the configuration of the device.

<isa-lance>.status

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
isa-bus	isa-bus	up
ethernet	ethernet-link	down

4.4 Standard Components

4.4.1 ddr-memory-module

Description

The "ddr-memory-module" component represents a DDR memory module.

Attributes

banks

Optional attribute; **read/write** access; type: **Integer**.
Number of banks.

cas_latency

Optional attribute; **read/write** access; type: **Integer**.
CAS-latency; each set bit corresponds to a latency the memory can handle

columns

Optional attribute; **read/write** access; type: **Integer**.
Number of columns.

ecc_width

Optional attribute; **read/write** access; type: **Integer**.
The error correction width.

memory_megs

Pseudo attribute; **read-only** access; type: **Integer**.
Total about of memory in MB.

module_data_width

Optional attribute; **read/write** access; type: **Integer**.
The module SDRAM width.

module_type

Optional attribute; **read/write** access; type: **String**.
Type of memory.

primary_width

Optional attribute; **read/write** access; type: **Integer**.
Primary SDRAM width.

rank_density

Optional attribute; **read/write** access; type: **Integer**.
The rank density.

ranks

Optional attribute; **read/write** access; type: **Integer**.
Number of ranks (logical banks).

rows
Optional attribute; **read/write** access; type: **Integer**.
Number of rows.

speed
Optional attribute; **read/write** access; type: **String**.
PC standard speed. Supported values are PC2700 and none.

Commands

create-ddr-memory-module [*"name"*] [*rows*] [*columns*] [*ranks*] [*module_data_width*] [*primary_w

Creates a non-instantiated component of the class "ddr-memory-module". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

**new-ddr-memory-module [*"name"*] [*rows*] [*columns*] [*ranks*] [*module_data_width*] [*primary_wid*

Creates an instantiated component of the class "ddr-memory-module". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**ddr-memory-module**>**.info**
Print detailed information about the configuration of the device.

<**ddr-memory-module**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	**Type**	**Direction**
mem-bus	mem-bus	up

4.4.2 std-ethernet-link

Description
The "std-ethernet-link" component represents a standard Ethernet link.

Attributes

frame_echo
Optional attribute; **read/write** access; type: **Integer**.
Set this attribute to echo frames back to the sender. Default is not to echo frames.

link_name

Optional attribute; **read/write** access; type: **String**.

The name to use for the **ethernet-link** object. An error will be raised at instantiation time if the link cannot be given this name.

Commands

create-std-ethernet-link [*"name"*] [*"link_name"*] [*frame_echo*]

Creates a non-instantiated component of the class "std-ethernet-link". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

new-std-ethernet-link [*"name"*] [*"link_name"*] [*frame_echo*]

Creates an instantiated component of the class "std-ethernet-link". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**std-ethernet-link**>**.info**

Print detailed information about the configuration of the device.

<**std-ethernet-link**>**.status**

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
device	ethernet-link	any

4.4.3 std-serial-link

Description

The "std-serial-link" component represents a standard Serial link.

Commands

create-std-serial-link [*"name"*]

Creates a non-instantiated component of the class "std-serial-link". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

new-std-serial-link [*"name"*]
Creates an instantiated component of the class "std-serial-link". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<std-serial-link>.info
Print detailed information about the configuration of the device.

<std-serial-link>.status
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
serial[0-1]	serial	any

4.4.4 std-service-node

Description
The "std-service-node" component represents a network service node that can be connected to Ethernet links to provide services such as DNS, DHCP/BOOTP, RARP and TFTP. A service node component does not have any connectors by default. Instead, connectors have to be added using the **<std-service-node>.add-connector** command.

Attributes

dynamic_connectors
Optional attribute; **read/write** access; type: **[[iss]*]**.
List of user added connectors

next_connector_id
Optional attribute; **read/write** access; type: **Integer**.
Next service-node device ID.

Commands

create-std-service-node [*"name"*]
Creates a non-instantiated component of the class "std-service-node". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

new-std-service-node [*"name"*]
Creates an instantiated component of the class "std-service-node". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**std-service-node**>**.add-connector** *"ip"* [*"netmask"*]
Adds a connector to the service-node with specified IP address and netmask. A connector must be created for the service-node before an Ethernet link can be connected to it. The *ip* argument is the IP address that the service node will use on the link. The *netmask* argument is optional, and defaults to `255.255.255.0`. The name of the new connector is returned.

<**std-service-node**>**.info**
Print detailed information about the configuration of the device.

<**std-service-node**>**.status**
Print detailed information about the current status of the device.

4.4.5 std-ide-disk

Description
The "std-ide-disk" component represents an IDE disk.

Attributes

file
Optional attribute; **read/write** access; type: **String**.
File with disk contents for the full disk Either a raw file or a CRAFF file.

size
Required attribute; **read/write** access; type: **Integer**.
The size of the IDE disk in bytes.

Commands

create-std-ide-disk [*"name"*] ***size*** [*"file"*]
Creates a non-instantiated component of the class "std-ide-disk". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**std-ide-disk**>**.info**
Print detailed information about the configuration of the device.

<**std-ide-disk**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	**Type**	**Direction**
ide-slot	ide-slot	up

4.4.6 std-ide-cdrom

Description

The "std-ide-cdrom" component represents an IDE ATAPI CD-ROM.

Commands

create-std-ide-cdrom [*"name"*]

Creates a non-instantiated component of the class "std-ide-cdrom". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**std-ide-cdrom**>**.info**

Print detailed information about the configuration of the device.

<**std-ide-cdrom**>**.status**

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
ide-slot	ide-slot	up

4.4.7 std-scsi-bus

Description

The "std-scsi-bus" component represents a 16 slot SCSI bus.

Commands

create-std-scsi-bus [*"name"*]

Creates a non-instantiated component of the class "std-scsi-bus". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**std-scsi-bus**>**.info**

Print detailed information about the configuration of the device.

<**std-scsi-bus**>**.status**

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
scsi-bus	scsi-bus	any

4.4.8 std-scsi-disk

Description

The "std-scsi-disk" component represents a SCSI-2 disk.

Attributes

file

Optional attribute; **read/write** access; type: **String**.

File with disk contents for the full disk Either a raw file or a CRAFF file.

scsi_id

Required attribute; **read/write** access; type: **Integer**.

The ID on the SCSI bus.

size

Required attribute; **read/write** access; type: **Integer**.

The size of the SCSI disk in bytes.

Commands

create-std-scsi-disk [*"name"*] *scsi_id size* [*"file"*]

Creates a non-instantiated component of the class "std-scsi-disk". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<std-scsi-disk>.info

Print detailed information about the configuration of the device.

<std-scsi-disk>.status

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
scsi-bus	scsi-bus	up

4.4.9 std-scsi-cdrom

Description

The "std-scsi-cdrom" component represents a SCSI-2 CD-ROM.

Attributes

scsi_id

Required attribute; **read/write** access; type: **Integer**.

The ID on the SCSI bus.

Commands

create-std-scsi-cdrom [*"name"*] *scsi_id*

Creates a non-instantiated component of the class "std-scsi-cdrom". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<std-scsi-cdrom>.info

Print detailed information about the configuration of the device.

<std-scsi-cdrom>.status

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
scsi-bus	scsi-bus	up

4.4.10 simple-fc-disk

Description

The "simple-fc-disk" component represents a SCSI-2 disk for use with Fibre-Channel SCSI controllers using the simplified FC protocol in Simics.

Attributes

file

Optional attribute; **read/write** access; type: **String**.

File with disk contents for the full disk Either a raw file or a CRAFF file.

loop_id

Required attribute; **read/write** access; type: **Integer**.

The loop ID for the FC disk.

node_name
Required attribute; **read/write** access; type: **Integer**.
The node name for the FC disk.

port_name
Required attribute; **read/write** access; type: **Integer**.
The port name for the FC disk.

size
Required attribute; **read/write** access; type: **Integer**.
The size of the FC disk in bytes.

Commands

create-simple-fc-disk [*"name"*] *size* [*"file"*] *loop_id node_name port_name*
Creates a non-instantiated component of the class "simple-fc-disk". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<simple-fc-disk>.info
Print detailed information about the configuration of the device.

<simple-fc-disk>.status
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
fc-loop	simple-fc-loop	up

4.4.11 std-text-console

Description
The "std-text-console" component represents a serial text console.

Attributes

bg_color
Optional attribute; **read/write** access; type: **String**.
The background color.

fg_color
Optional attribute; **read/write** access; type: **String**.
The foreground color.

height
Optional attribute; **read/write** access; type: **Integer**.
The height of the console window.

title
Optional attribute; **read/write** access; type: **String**.
The Window title.

width
Optional attribute; **read/write** access; type: **Integer**.
The width of the console window.

win32_font
Optional attribute; **read/write** access; type: **String**.
Font to use in the console on Windows host.

x11_font
Optional attribute; **read/write** access; type: **String**.
Font to use in the console when using X11 (Linux/Solaris host).

Commands

**create-std-text-console [*"name"*] [*"title"*] [*"bg_color"*] [*"fg_color"*] [*"x11_font"*] [*"win32_font"*] [*w*

Creates a non-instantiated component of the class "std-text-console". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

**new-std-text-console [*"name"*] [*"title"*] [*"bg_color"*] [*"fg_color"*] [*"x11_font"*] [*"win32_font"*] [*wi*

Creates an instantiated component of the class "std-text-console". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**std-text-console**>**.info**
Print detailed information about the configuration of the device.

<**std-text-console**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	**Type**	**Direction**
serial	serial	up

4.4.12 std-server-console

Description

The "std-server-console" component represents a serial console accessible from the host using telnet.

Attributes

telnet_port

Required attribute; **read/write** access; type: **Integer**.

TCP/IP port to connect the telnet service of the console to.

Commands

create-std-server-console [*"name"*] *telnet_port*

Creates a non-instantiated component of the class "std-server-console". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

new-std-server-console [*"name"*] *telnet_port*

Creates an instantiated component of the class "std-server-console". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**std-server-console**>**.info**

Print detailed information about the configuration of the device.

<**std-server-console**>**.status**

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
serial	serial	up

4.4.13 std-graphics-console

Description

The "std-graphics-console" component represents a graphical console for displaying output from a simulated graphics adapters and getting input for mouse and keyboard devices.

Attributes

window
Optional attribute; **read/write** access; type: **b**.
Try to open window if TRUE (default). FALSE disabled the window.

Commands

create-std-graphics-console [*"name"*] [*window*]
Creates a non-instantiated component of the class "std-graphics-console". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

new-std-graphics-console [*"name"*] [*window*]
Creates an instantiated component of the class "std-graphics-console". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**std-graphics-console**>**.info**
Print detailed information about the configuration of the device.

<**std-graphics-console**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	**Type**	**Direction**
device	graphics-console	up
keyboard	keyboard	up
mouse	mouse	up

4.4.14 std-text-graphics-console

Description
The "std-text-graphics-console" component represents a text console for use with VGA instead of a graphics console.

Commands

create-std-text-graphics-console [*"name"*]
Creates a non-instantiated component of the class "std-text-graphics-console". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

new-std-text-graphics-console [*"name"*]
Creates an instantiated component of the class "std-text-graphics-console". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<std-text-graphics-console>.info
Print detailed information about the configuration of the device.

<std-text-graphics-console>.status
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
device	graphics-console	up
keyboard	keyboard	up

4.5 Base Components

The base components are abstract classes that contain generic component attributes and commands available for all components.

4.5.1 component

Description
Base component class, should not be instantiated.

Attributes

connections
Optional attribute; **read/write** access; type: **[[sos]*]**.
List of connections for the component. The format is a list of lists, each containing the name of the connector, the connected component, and the name of the connector on the other component.

connectors
Pseudo class attribute; **read-only** access; type: **D**.
Dictionary of dictionaries with connectors defined by this component class, indexed by name. Each connector contains the name of the connector "type", a "direction" ("up", "down" or "any"), a flag indicating if the connector can be "empty", another flag that is set if the connector is "hotplug" capable, and finally a flag that is TRUE if muliple connections to this connector is allowed.

instantiated
Optional attribute; **read/write** access; type: **b**.
Set to TRUE if the component has been instantiated.

object_list
Optional attribute; **read/write** access; type: **D**.
Dictionary with objects that the component consists of.

object_prefix
Optional attribute; **read/write** access; type: **String**.
Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component
Optional attribute; **read/write** access; type: **Object**.
The top level component. Attribute is not valid until the component has been instantiated.

top_level
Optional attribute; **read/write** access; type: **b**.
Set to TRUE for top-level components, i.e. the root of a hierarchy.

4.5.2 top-component

Description
Base top-level component class, should not be instantiated.

Attributes

components
Optional attribute; **read/write** access; type: **[o∗]**.
List of components below the the top-level component. This attribute is not valid until the object has been instantiated.

cpu_list
Optional attribute; **read/write** access; type: **[o∗]**.
List of all processors below the the top-level component. This attribute is not valid until the object has been instantiated.

Chapter 5

Examples

5.1 Adding a SCSI Disk to an x86-440bx Machine

The following section described how to add a SCSI controller and a single empty SCSI disk to the Enterprise machine. Adding a disk to other simulated machines is done in a similar way.

1. **Add SCSI Controller and Disk Components**

   First create a file in `[workspace]/targets/x86-440bx/` called `enterprise-disk.simics`. In this file add the following contents:

```
script-branch {
    wait-for-variable machine_defined
    local $sym = (create-pci-sym53c810)
    local $scsi_bus = (create-std-scsi-bus)
    local $scsi_disk = (create-std-scsi-disk scsi_id = 0 size = 425697280

    $nb.connect pci-slot2 $sym
    $scsi_bus.connect $sym
    $scsi_bus.connect $scsi_disk
}

run-command-file enterprise-common.simics
```

   This will run a *script branch* that first waits for the machine to be defined by the machine configuration script (included from `enterprise-common.simics`). Once the *$machine_defined* variable has triggered, a SCSI controller, SCSI bus and a SCSI disk component representing a 4GB disk, will be created. On the last lines the controller is connected to the north-bridge (the `$nb` variable) in PCI slot 2, and to the SCSI bus. The disk is also connected to the SCSI bus.

The SCSI id must be specified for attached SCSI devices, and we choose to use id 0 for the disk. We do not have to specify an id for the sym53c810 controller since it always uses id 7.

2. **Prepare the Boot**

   Start Simics, but do not start the simulation. Before booting, it is a good idea to enable *real-time mode* in Simics by issuing the command **enable-real-time-mode**. The reason for this is that the Linux hardware configurator program (kudzu) will detect the SCSI controller and ask if it should be configured. Kudzu then waits for 30 seconds for input, but this wait will run much faster than in reality since Simics optimizes the idle loop.

   When kudzu has detected the SCSI controller, choose the option of configuring it. After this point the real-time mode can be disabled again in Simics. If the kudzu times out before it receives any user input, it can be run from the Linux command line manually once the machine has booted. (**/sbin/kudzu**).

3. **Configure Linux**

   Once Linux has booted, and detected the SCSI controller, run **fdisk** to create a partition on the new disk. At the menu, select `n` to create a new partition, then `p` for primary partition and `1` for the first partition.

```
# fdisk /dev/sda
ncr53c8xx: setting PCI_COMMAND_MASTER...(fix-up)
ncr53c8xx: changing PCI_LATENCY_TIMER from 0 to 32.
  Vendor: Vtech     Model: Turbo_Disk(tm)    Rev: 0001
  Type:   Direct-Access                      ANSI SCSI revision: 02
Attached scsi disk sda at scsi0, channel 0, id 0, lun 0
SCSI device sda: 8314400 512-byte hdwr sectors (4257 MB)
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF diskla
bel
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous
content won't be recoverable.


Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
```

   Next use the default options for cylinder, by simply pressing return on each line. Finally select the `w` command to write the partition table to the new disk.

```
First cylinder (1-1023, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-1023, default 1023):
Using default value 1023

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
SCSI device sda: 8314400 512-byte hdwr sectors (4257 MB)
SCSI device sda: 8314400 512-byte hdwr sectors (4257 MB)
Syncing disks.
```

A file system now has to be created on the new partition using the **mkfs** command. At the same time, also add a mount point, and an entry in the file-system table. This way Linux will automatically mount the disk on the next boot.

```
# mkfs /dev/sda1

<output from mkfs here>

# mkdir /disk
# cat >> /etc/fstab
/dev/sda1       /disk       ext2    defaults  1 2
<control-D>
# mount /disk
```

The disk can now be accessed as `/disk/` in the file-system.

4. **Save the Changes**

   To save the changes to the new disk, shut down the simulated machine and save the modifications. Issue:

```
# init 0
```

   then wait for Linux to shut down, stop the simulation and save the all modifications using the **save-persistent-state** command.

```
simics> save-persistent-state new-disk1.state
```

   Now exit Simics, and restart the `enterprise-disk.simics` script. Before running, load the disk modifications saved earlier:

```
simics> load-persistent-state new-disk1.state
```

Now boot the machine again. The new disk will be mounted as `/disk/`.

Chapter 6

Installing an OS on Simics

The files `dredd-cd-install1.simics` and `dredd-cd-install2.simics`, that are supplied with the dredd simulated machine provide a way to install an operating system on Simics from a bootable CD-ROM. Before starting, please read the instructions in `dredd-cd-install1.simics`.

Note: If you plan to mount your finished disk using loopback, you should make sure that each simulated file system is small enough to fit uncompressed on your local machine.

Note: The instructions given here for installing from CD-ROM also apply to installing from DVD. The ISO format is the same for data CD-ROMs and DVDs and you can install from an ISO image of any size using the **new-file-cdrom** command described below.

---

1. Create an ISO image file of the bootable CD-ROM you want to install from. Linux example:

   ```
   $ dd if=/dev/cdrom of=myos.iso
   ```

   On Windows, a third-party tool is necessary to create the ISO image. For more information on how to deal with CD-ROMs and other disk images, see the chapter titled *Managing Disks, Floppies, and CD-ROMs* in the *Simics User Guide*. On Linux and Solaris hosts it is not necessary to create the ISO file, Simics can also access the actual CD directly by specifying the device file.

2. Run Simics with the first script:

   ```
   $ ./simics targets/x86-440bx/dredd-cd-install1.simics
   ```

3. Insert the ISO image with the OS into the simulated CD-ROM drive with the following two commands, and then start the simulation.

```
simics> new-file-cdrom myos.iso
cdrom 'myos' created
simics> cd0.insert myos
simics> c
```

To install from a CD that is inserted in the host CD-ROM unit instead of from an ISO file, replace the first line in the example above with the following example line. The argument `/dev/cdrom` is the path to the device and may be different depending on your host system. Note that this is **not** the path to the files on the CD, such as `/mnt/cdrom`. Accessing a CD on the host is not supported on Windows hosts.

```
simics> new-file-cdrom "/dev/cdrom" myos
```

4. When the installation is finished, shutdown the simulated machine (a lot of OSes do this automatically when the install is complete). The way to shutdown the machine differs between operating systems, but there is usually some help available on the screen.

5. You may get a error message from Simics saying that reboot/suspend/shutdown is an experimental feature, but that can be ignored. At this point save the persistent state of the machine:

   ```
   simics> save-persistent-state install-phase1.state
   ```

   The persistent state contains all information on the disk, as well as the contents of NVRAMs and other devices that survive reboot. Finally exit Simics.

6. Start Simics using the second script, which will boot from the installation disk and not from the CD-ROM. If the CD-ROM is needed, it has to be inserted into the machine in the same way as in the first phase of the installation.

   If the installer program in the simulated machine asks for the path to the CD-ROM, give the path in the simulated machine. Since there is only one CD-ROM in the default machine setup, this should be quite easy to identify.

   ```
   $ ./simics targets/x86-440bx/dredd-cd-install2.simics
   ```

   Then, before starting the simulation, load the persistent state that was saved at the end of the first phase.

   ```
   simics> load-persistent-state install-phase1.state
   ```

7. Since there are probably changes done to the disk in the second phase as well, reboot the simulated machine, and when it has completed the shut down, save the persistent state to a new file.

8. If there are several reboots during the installation process, save a new state file each time. Only the most recent one has to be loaded when Simics is restarted, but all state files must be kept since they depend on each other.

9. When the install is complete, you will have one or more persistent state files. To create a single one, use the *checkpoint-merge* utility to join them. After verifying that the combined persistent state works, the previous ones can be removed.

10. To run the machine with the newly installed operating system, start the `dredd-common.simics` and load the most recent persistent state file, or the combined state file that was created using the checkpoint-merge utility.

11. If you want to change the number of processors in the machine or some other machine parameters, you can do so now (provided that you have installed support for multiple processors or other hardware during your install). See chapter 3.5.

Chapter 7

Miscellaneous Notes

7.1 Timing problems

If you have problems typing in the password when logging in because the simulated machine times out too quickly you can issue the Simics command:

```
simics> enable-real-time-mode
```

This will cause Simics not to run faster then the the wall clock. Normally this is not a problem and it will only happens when the x86 processor is sitting idle on a HLT instruction.

For NetBSD, you will also need to lower the check interval to avoid repeated keystrokes:

```
simics> enable-real-time-mode check_interval=100
```

To run as fast as possible again you simply run the command:

```
simics> disable-real-time-mode
```

See the Reference Manual for further information.

7.2 Console input

Certain key combinations are normally caught by the host operating system or the window manager and may have dangerous side effects. An example is the ctrl + alt + del combination used to bring up the task manager on Windows and to reboot Linux systems. Such key combinations can be safely inserted using Simics commands instead.

For example to send ctrl + alt + del:

```
simics> con0.input -e "C-A-Del"
```

Regular strings can also be issued:

```
simics> con0.input "mkdir foo\n"
```

There are also some functions written in Python available in the `home/scripts/expect.py` file. They can be used to wait for certain output to appear on the simulated screen. Similar functionality is also available through the console if the `use_video_text_console` option has been activated.

7.3 BIOS Timer Error

The firmware will most likely pause, complaining about a `Timer Error`. At that point F1 must be pressed to continue the boot process. To send F1 to the system issue:

```
simics> con0.input -e "F1"
```

The halt on system error can also be disabled through the firmware configuration menues.

7.4 Changing the Processor Clock Frequency

The clock frequency of a simulated processor can be set arbitrarily in Simics. This will not affect the actual speed of simulation, but it will affect the number of instructions that need to be executed for a certain amount of simulated time to pass. If your execution only depends on executing a certain number of instructions, increasing the clock frequency will take the same amount of host time (but a shorter amount of target time). However, if there are time based delays of some kind in the simulation, these will take longer to execute.

At a simulated 1 MHz, one million target instructions will correspond to a simulated second (assuming the simple default timing of one cycle per instruction). At 100 MHz, on the other hand, it will take 100 million target instructions to complete a simulated second. So with a higher clock frequency, less simulated target time is going to pass for a certain period of host execution time.

If Simics is used to emulate an interactive system (especially one with a graphical user interface) it is a good idea to set the clock frequency quite low. Keyboard and mouse inputs events are handled by periodic interrupts in most operating systems, using a higher clock frequency will result in longer delays between invocations of periodic interrupts. Thus, the simulated system will feel slower in its user response, and update the mouse cursor position etc. less frequently. If this is a problem, the best technique for running experiments at a high clock frequency is to first complete the configuration of the machine using a low clock frequency. Save all configuration changes to a disk diff (like when installing operating systems). Then change the configuration to use a higher a clock frequency and reboot the target machine.

Note that for a lightly-loaded machine (for example, working at an interactive prompt on a serial console to an embedded Linux system), Simics will often execute quickly enough at the real target clock frequency that there is no need to artifically lower it.

Chapter 8

Limitations

8.1 Limitations of the Simulated Model

- Not all 3DNow! instructions are implemented (not an issue for Intel-based targets).
- Legacy (x87) floating point is not always bit-by-bit compatible with the simulated target processor.
- Floating point approximation operations are always computed with full precision.
- Multi-machine configurations with graphical consoles do not work on Windows hosts since it uses SDL that only supports one window.
- For 32-bit targets, the maximum supported RAM size is 3840 MB. There is no such limitation in the 64-bit targets.

8.2 Other Limitations

- The SimicsFS file system is only read-only in Linux 2.4.x.

Index

D

I

N

O

P

S

T

X

Virtutech, Inc.

1740 Technology Dr., suite 460
San Jose, CA 95110
USA

Phone +1 408-392-9150
Fax +1 408-608-0430

http://www.virtutech.com