virtutech

Simics/SunFire 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/SunFire Overview

2.1 Introduction

Simics/SunFire models the *Sun Enterprise 3500 - 6500* class of servers. A SunFire server can be configured with up to 30 UltraSPARC II processors and 60GB of memory. A variety of SBus and PCI-bus based devices are supported. Both Solaris and Linux are supported as target Operating Systems.

Virtutech does not provide any disk images with Solaris for SunFire, due to licensing issues. However, scripts are included for installing Solaris 8, 9 or 10 on the virtual machine. Installing Solaris is described in chapter 6.1.

2.2 Supported Hardware

SunFire machines have up to 16 board slots depending on the model. Simics models the largest machine, Sun Enterprise 6500, by default, but this is configurable. Each board slot can hold a CPU, SBus or PCI board.

System Backplane	# of Boards	Board slot numbers
Sun Enterprise 3500	5	1, 3, 5, 7, 9
Sun Enterprise 4500	8	0 - 7
Sun Enterprise 5500	8	0 - 7
Sun Enterprise 6500	16	0 - 15

Supported Boards		
CPU	2 UltraSPARC II CPUs	4GB of memory
SBus	3 free SBus slots	On board 'fas' and 'hme' adapters
PCI	2 free PCI slots	On board 'isp' and 'hme' adapters

Supported SBus Devices		
'fas'	SCSI controller	(FAS366U)
'hme'	Ethernet controller	(FEPS, Happy Meal)

Supported PCI Devices

'ce'	Gb Ethernet controller	(Cassini+)
'bge'	Gb Ethernet controller	(BCM5703C)
'bge'	Dual Gb Ethernet controller	(BCM5704C)
'hme'	Ethernet controller	(Cheerio)
'isp'	SCSI controller	(ISP1040)
'pgx64'	24-Bit Frame Buffer	(pgx64)
'qlc'	Fibre-Channel controller	(ISP2200)
	PCI-to-PCI bridge	(i21152)

Other Supported Devices

Sun Type 5 Keyboard	
Sun Type 5 Mouse	
Serial ports	(Z8530)

A good guide to the Sun Enterprise servers and what boards and devices that are supported can be found in the "Sun System Handbook", available online at: `http://sunsolve.sun.com/handbook_pub/`

Chapter 3

Simulated Machines

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

3.1 Peanut

Peanut is a Sun Enterprise 6500 server with a single UltraSPARC II processor running at 168 MHz, and 256 MB of memory. It has one Ethernet adapter, one SCSI disk and one SCSI CD-ROM. The default configuration can be modified as described in section 3.6. The Solaris operating system must be installed on peanut, using the supplied scripts, before it can be used.

3.1.1 Peanut Scripts

`peanut-common.simics`
Starts the Peanut machine with the default configuration.

`peanut-gcache-common.simics`
Default Peanut machine with a *g-cache* cache model connected.

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

`peanut-ooo-common.simics`
Default Peanut machine with a simple out-of-order timing model connected. Simics must be stared in MAI mode (-ma) to run this script.

`peanut-multi.simics`
Example script with two Peanut machines in the same session, connected by an Ethernet link.

peanut-sol<version>-cd-install1.simics
Script for installing Solaris on the simulated machine, phase 1. <version> is one of 8, 9 and 10.

peanut-sol<version>-cd-install2.simics
Script for installing Solaris on the simulated machine, phase 2. <version> is one of 8, 9 and 10.

peanut-sol<version>-cd-install3.simics
Script for installing Solaris on the simulated machine, phase 3. <version> is one of 8, 9 and 10.

3.2 Walnut

Walnut is a Sun Enterprise 6500 server with a single UltraSPARC II processor running at 168 MHz, and 256 MB of memory. It has one Ethernet adapter, one SCSI disk and one SCSI CD-ROM. The default configuration can be modified as described in section 3.6. An operating system must be installed on walnut before it can be used.

3.2.1 Walnut Scripts

walnut-common.simics
Starts the Walnut machine with the default configuration.

walnut-cd-install1.simics
Script for installing an OS on the simulated machine, phase 1.

walnut-cd-install2.simics
Script for installing an OS on the simulated machine, phase 2.

3.3 Cashew

Cashew is a Sun Enterprise 6500 server with a single UltraSPARC II processor running at 168 MHz, and 256 MB of memory. It has one Ethernet adapter, one SCSI disk and one SCSI CD-ROM. The default configuration can be modified as described in section 3.6.

Cashew is configured for an existing Aurora Linux 2.0 disk dump, that can be downloaded from the Virtutech web site.

Additional information:

- Aurora 2.0 Linux (Fedora Cora 3), installed directly on Simics.
- Linux kernel 2.6.13
- SimicsFS support.
- Login `root`, password “simics”.
- Configured to get IP address using DHCP.

3.3.1 Cashew Scripts

`cashew-common.simics`
Starts the Cashew machine with the default configuration.

`cashew-fb-common.simics`
Similar to cashew-common.simics, but also adds a frame-buffer device (graphic card) and a graphical console.

`cashew-gcache-common.simics`
Default Cashew machine with a *g-cache* cache model connected.

`cashew-multi.simics`
Example script with two Cashew machines in the same session, connected by an Ethernet link.

3.4 Bagle

Bagle is a Sun Enterprise 6500 server with a single UltraSPARC II processor running at 168 MHz, and 256 MB of memory. It has one Ethernet adapter, one SCSI disk and one SCSI CD-ROM. The default configuration can be modified as described in section 3.6.

Bagle is configured for an existing SuSE Linux 7.3 disk dump, that can be downloaded from the Virtutech web site.

Additional information:

- SuSE 7.3 Linux, installed directly on Simics.
- Linux kernel 2.4.14
- SimicsFS support (read-only, write support experimental).
- Login `root`, no password.
- Configured with static IP address 10.10.0.6, gw 10.10.0.1, when DHCP not used.

3.4.1 Bagle Scripts

`bagle-common.simics`
Starts the Bagle machine with the default configuration.

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

`bagle-fb-common.simics`
Similar to bagle-common.simics, but also adds a frame-buffer device (graphic card) and a graphical console.

`bagle-gcache-common.simics`
Default Bagle machine with a *g-cache* cache model connected.

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

bagle-ooo-common.simics
Default Bagle machine with a simple out-of-order timing model connected. Simics must be stared in MAI mode (-ma) to run this script.

bagle-multi.simics
Example script with two Bagle machines in the same session, connected by an Ethernet link.

3.5 Donut

Donut is a Sun Enterprise 6500 server with a single UltraSPARC II processor running at 168 MHz, and 256 MB of memory. It has one Ethernet adapter, one SCSI disk and one SCSI CD-ROM. The default configuration can be modified as described in section 3.6.

The Donut machine is configured for existing Solaris 8, 9 or 10 disk dumps. The disk dumps are only available for commercial customer with a special license agreement with Sun. Some common GNU utilities are installed on the disk images, such as `bash`, `gcc`, `gmake` and `emacs`. The *SimicsFS* file-system is also included.

Additional information:

- Solaris 8 (7/01) and Solaris 9 (5/02) installed as "Developer System" directly on Simics.
- SimicsFS support.
- Login `root`, no password.
- Configured with static IP address 10.10.0.5, gw 10.10.0.1, when DHCP not used.

3.5.1 Donut Scripts

donut-common.simics
Starts the Donut machine with the default configuration.

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

donut-fb-common.simics
Similar to donut-common.simics, but also adds a frame-buffer device (graphic card) and a graphical console.

donut-gcache-common.simics
Default Donut machine with a *g-cache* cache model connected.

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

`donut-ooo-common.simics`
Default Donut machine with a simple out-of-order timing model connected. Simics must be stared in MAI mode (-ma) to run this script.

`donut-multi.simics`
Example script with two Donut machines in the same session, connected by an Ethernet link.

3.6 Parameters for Machine Scripts

The following parameters can be set before running the `peanut-common.simics`, `walnut-common.simics`, `bagle-common.simics` or `donut-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 `bagle-dhcp-common.simics` script will always set the *$create_network* variable to `yes`.

3.6.1 peanut-common, walnut-common, bagle-common and donut-common

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

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

$do_boot
Set to `no` to stop at OBP prompt, without booting the OS.

$do_login
Set to `no` to prevent the script from logging in as root automatically when the operating system has reached the login prompt.

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

$hostid
The *hostid* for the simulated machine.

$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.

$num_cpus
The number of processors in the machine.

$os
The operating system to boot, one of `solaris10`, `solaris9`, and `solaris8`. Requires that a matching disk dump exists. This variable does not exist for the bagle 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`.

Chapter 4

Supported Components

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

The default machines are constructed from components in the `-system.include` files in `[simics]/targets/sunfire/`. 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 SunFire Components

4.1.1 sunfire-3500-backplane

Description

The "sunfire-3500-backplane" component represents the chassis, backplane and clock-board of a Sun Enterprise 3500 server, with slots for up to five boards.

Attributes

cpu_frequency

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

Processor frequency in MHz. Supported frequencies are 168, 200, 248, 296, 336, 360, 400, 464, 496, 592, 720, 800, 992.

hostid

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

The hostid of the machine.

mac_address

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

The main MAC address is the machine.

master_cpu

Optional attribute; **read/write** access; type: **s**|**o**.

Internal attribute.

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

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

Commands

create-sunfire-3500-backplane [*"name"*] *cpu_frequency hostid "mac_address" "rtc_time"*

Creates a non-instantiated component of the class "sunfire-3500-backplane". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**sunfire-3500-backplane**>**.get-nvram-hostid**
Reads the Sun hostid from the NVRAM.

<**sunfire-3500-backplane**>**.get-nvram-mac**
Reads the default MAC address from the NVRAM.

<**sunfire-3500-backplane**>**.get-prom-env [*"variable"*]**
Prints an OBP variable with its value, or all variables if no argument is specified. Only variables with string, integer, boolean and enumeration types are supported.

<**sunfire-3500-backplane**>**.info**
Print detailed information about the configuration of the device.

<**sunfire-3500-backplane**>**.key-switch [*"position"*]**
Changes the position of the key-switch on the clock-board. Possible arguments are "diag", "insecure" and "secure". If no argument is specified, the current position is reported.

<**sunfire-3500-backplane**>**.set-nvram-hostid *hostid***
Writes the Sun hostid into the NVRAM.

<**sunfire-3500-backplane**>**.set-nvram-mac "*mac*"**
Writes the default MAC address into the NVRAM.

<**sunfire-3500-backplane**>**.set-prom-defaults**
Restores all OBP variables to their default values.

<**sunfire-3500-backplane**>**.set-prom-env** *"variable"* **(***int***|***"string"***)**
Sets the value OBP variable in the NVRAM. Only variables with string, integer, boolean and enumeration types are supported.

<**sunfire-3500-backplane**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	**Type**	**Direction**
central-cpu	sunfire-central-cpu	down
keyboard	serial	down
mouse	serial	down
remote_console	serial	down
slot[0-15]	sunfire-backplane	down
ttya	serial	down
ttyb	serial	down

4.1.2 sunfire-4500-backplane

Description
The "sunfire-4500-backplane" component represents the chassis, backplane and clock-board of a Sun Enterprise 4500 server, with slots for up to eight boards.

Attributes

cpu_frequency
Required attribute; **read/write** access; type: **Integer**.
Processor frequency in MHz. Supported frequencies are 168, 200, 248, 296, 336, 360, 400, 464, 496, 592, 720, 800, 992.

hostid
Required attribute; **read/write** access; type: **Integer**.
The hostid of the machine.

mac_address
Required attribute; **read/write** access; type: **String**.
The main MAC address is the machine.

master_cpu
Optional attribute; **read/write** access; type: **s**|**o**.
Internal attribute.

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

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

Commands

create-sunfire-4500-backplane [*"name"*] *cpu_frequency hostid "mac_address" "rtc_time"*

Creates a non-instantiated component of the class "sunfire-4500-backplane". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**sunfire-4500-backplane**>**.get-nvram-hostid**
Reads the Sun hostid from the NVRAM.

<**sunfire-4500-backplane**>**.get-nvram-mac**
Reads the default MAC address from the NVRAM.

<**sunfire-4500-backplane**>**.get-prom-env [*"variable"*]**
Prints an OBP variable with its value, or all variables if no argument is specified. Only variables with string, integer, boolean and enumeration types are supported.

<**sunfire-4500-backplane**>**.info**
Print detailed information about the configuration of the device.

<**sunfire-4500-backplane**>**.key-switch [*"position"*]**
Changes the position of the key-switch on the clock-board. Possible arguments are "diag", "insecure" and "secure". If no argument is specified, the current position is reported.

<**sunfire-4500-backplane**>**.set-nvram-hostid *hostid***
Writes the Sun hostid into the NVRAM.

<**sunfire-4500-backplane**>**.set-nvram-mac "*mac*"**
Writes the default MAC address into the NVRAM.

<**sunfire-4500-backplane**>**.set-prom-defaults**
Restores all OBP variables to their default values.

<**sunfire-4500-backplane**>**.set-prom-env** *"variable"* **(***int*|*"string"***)**
Sets the value OBP variable in the NVRAM. Only variables with string, integer, boolean and enumeration types are supported.

<**sunfire-4500-backplane**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	**Type**	**Direction**
central-cpu	sunfire-central-cpu	down
keyboard	serial	down
mouse	serial	down
remote_console	serial	down
slot[0-15]	sunfire-backplane	down
ttya	serial	down
ttyb	serial	down

4.1.3 sunfire-6500-backplane

Description
The "sunfire-6500-backplane" component represents the chassis, backplane and clock-board of a Sun Enterprise 6500 server, with slots for up to sixteen boards.

Attributes

cpu_frequency
Required attribute; **read/write** access; type: **Integer**.
Processor frequency in MHz. Supported frequencies are 168, 200, 248, 296, 336, 360, 400, 464, 496, 592, 720, 800, 992.

hostid
Required attribute; **read/write** access; type: **Integer**.
The hostid of the machine.

mac_address
Required attribute; **read/write** access; type: **String**.
The main MAC address is the machine.

master_cpu
Optional attribute; **read/write** access; type: **s**|**o**.
Internal attribute.

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

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

Commands

create-sunfire-6500-backplane [*"name"*] *cpu_frequency hostid "mac_address" "rtc_time"*

Creates a non-instantiated component of the class "sunfire-6500-backplane". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**sunfire-6500-backplane**>**.get-nvram-hostid**
Reads the Sun hostid from the NVRAM.

<**sunfire-6500-backplane**>**.get-nvram-mac**
Reads the default MAC address from the NVRAM.

<**sunfire-6500-backplane**>**.get-prom-env [*"variable"*]**
Prints an OBP variable with its value, or all variables if no argument is specified. Only variables with string, integer, boolean and enumeration types are supported.

<**sunfire-6500-backplane**>**.info**
Print detailed information about the configuration of the device.

<**sunfire-6500-backplane**>**.key-switch [*"position"*]**
Changes the position of the key-switch on the clock-board. Possible arguments are "diag", "insecure" and "secure". If no argument is specified, the current position is reported.

<**sunfire-6500-backplane**>**.set-nvram-hostid *hostid***
Writes the Sun hostid into the NVRAM.

<**sunfire-6500-backplane**>**.set-nvram-mac "*mac*"**
Writes the default MAC address into the NVRAM.

<**sunfire-6500-backplane**>**.set-prom-defaults**
Restores all OBP variables to their default values.

<**sunfire-6500-backplane**>**.set-prom-env** ***"variable"*** **(*int*|*"string"*)**
Sets the value OBP variable in the NVRAM. Only variables with string, integer, boolean and enumeration types are supported.

<**sunfire-6500-backplane**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	**Type**	**Direction**
central-cpu	sunfire-central-cpu	down
keyboard	serial	down
mouse	serial	down
remote_console	serial	down
slot[0-15]	sunfire-backplane	down
ttya	serial	down
ttyb	serial	down

4.1.4 sunfire-cpu-board

Description
The "sunfire-cpu-board" component represents a processor board with up to two UltraSPARC II processors and 4GB of memory, for use in Sun Enterprise 3500-6500 servers.

Attributes

memory_megs
Required attribute; **read/write** access; type: **Integer**.
The amount of RAM in megabytes on the processor board.

num_cpus
Required attribute; **read/write** access; type: **Integer**.
Number of processors on the board (1 or 2).

Commands

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

<sunfire-cpu-board>.info
Print detailed information about the configuration of the device.

<sunfire-cpu-board>.status
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
backplane	sunfire-backplane	up
central-cpu	sunfire-central-cpu	up
cache-cpu[0-1]	timing-model	down

4.1.5 sunfire-sbus-board

Description

The "sunfire-sbus-board" component represents an I/O board with slots for up to three SBus cards, for use in Sun Enterprise 3500-6500 servers. The board has one HME ethernet controller and one FAS SCSI controller on-board.

The UPA in the table is relative the first UPA on the board. (The UPA is an identifier for each CPU and I/O controller in the system. There are two UPAs per board, resulting in a total of 32.)

SBus slot mappings:

Simics slot	UPA	SBus Slot	Bus address	Note
0	1	0	1,0 at 0,0	
1	0	1	0,0 at 1,0	
2	0	2	0,0 at 2,0	
3	1	3	1,0 at 3,0	On-board fas, hme

Attributes

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

scsi_id
Optional attribute; **read/write** access; type: **Integer**.
The ID on the SCSI bus.

Commands

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

<sunfire-sbus-board>.info
Print detailed information about the configuration of the device.

<sunfire-sbus-board>.status
Print detailed information about the current status of the device.

Connectors

Name	**Type**	**Direction**
backplane	sunfire-backplane	up
ethernet	ethernet-link	down
scsi-bus	scsi-bus	down
slot[0-2]	sun-sbus	down

4.1.6 sunfire-pci-board

Description

The "sunfire-pci-board" component represents an I/O board with slots for up to two PCI cards, for use in Sun Enterprise 3500-6500 servers. The board has one HME ethernet controller and one ISP SCSI controller on-board.

The UPA in the table is relative the first UPA on the board. (The UPA is an identifier for each CPU and I/O controller in the system. There are two UPAs per board, resulting in a total of 32.)

PCI slot mappings:

Simics slot	UPA	PCI Bus	PCI Slot	Bus address	Note
0	0	A	2	0,2000	
1	0	B	1	0,4000	On-board hme
2	1	A	2	1,2000	
3	1	B	3	1,4000	On board isp

Attributes

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

scsi_id
Optional attribute; **read/write** access; type: **Integer**.
The ID on the SCSI bus.

Commands

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

<**sunfire-pci-board**>**.info**
Print detailed information about the configuration of the device.

<**sunfire-pci-board**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
backplane	sunfire-backplane	up
ethernet	ethernet-link	down
pci-slot0	pci-bus	down
pci-slot2	pci-bus	down
scsi-bus	scsi-bus	down

4.2 SBus and PCI Device Components

4.2.1 sun-sbus-fas-hme

Description
The "sun-sbus-fas-hme" component represents an SBus card with one HME ethernet controller and one FAS SCSI controller for use in Sun systems.

Attributes

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

scsi_id
Optional attribute; **read/write** access; type: **Integer**.
The ID on the SCSI bus.

Commands

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

<**sun-sbus-fas-hme**>**.info**
Print detailed information about the configuration of the device.

<**sun-sbus-fas-hme**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	**Type**	**Direction**
sbus	sun-sbus	up
ethernet	ethernet-link	down
scsi-bus	scsi-bus	down

4.2.2 sun-pci-ce

Description
The "sun-pci-ce" component represents a PCI card with a Cassini gigabit Ethernet adapter, for use in Sun systems.

Attributes

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

Commands

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

<**sun-pci-ce**>**.info**
Print detailed information about the configuration of the device.

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

Connectors

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

4.2.3 sun-pci-hme

Description
The "sun-pci-hme" component represents a PCI card with a HME Ethernet adapter, for use in Sun systems.

Attributes

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

Commands

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

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

<sun-pci-hme>.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 sun-pci-hme-isp

Description

The "sun-pci-hme-isp" component represents a PCI card with one HME Ethernet adapter and one ISP SCSI controller for use in Sun systems.

Attributes

mac_address

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

The MAC address of the Ethernet adapter.

scsi_id

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

The ID on the SCSI bus.

Commands

create-sun-pci-hme-isp [*"name"*] *"mac_address"* [*scsi_id*]

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

<**sun-pci-hme-isp**>**.info**

Print detailed information about the configuration of the device.

<**sun-pci-hme-isp**>**.status**

Print detailed information about the current status of the device.

Connectors

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

4.2.5 sun-pci-pgx64

Description

The "sun-pci-pgx64" component represents a PCI card with a PGX64 (Rage XL) graphics adapter, for use in Sun systems.

Commands

create-sun-pci-pgx64 [*"name"*]
Creates a non-instantiated component of the class "sun-pci-pgx64". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<sun-pci-pgx64>.info
Print detailed information about the configuration of the device.

<sun-pci-pgx64>.status
Print detailed information about the current status of the device.

Connectors

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

4.2.6 sun-pci-qlc

Description
The "sun-pci-qlc" component represents a PCI card with a QLC Fibre-Channel SCSI controller for use in Sun systems.

Attributes

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

Commands

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

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

<sun-pci-qlc>.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 sun-pci-qlc-qlc

Description

The "sun-pci-qlc-qlc" component represents a PCI card with two QLC Fibre-Channel SCSI controller for use in Sun systems.

Attributes

loop_id0

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

The FC loop ID of the first QLC controller.

loop_id1

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

The FC loop ID of the second QLC controller.

Commands

create-sun-pci-qlc-qlc [*"name"*] *loop_id0 loop_id1*

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

<**sun-pci-qlc-qlc**>**.info**

Print detailed information about the configuration of the device.

<**sun-pci-qlc-qlc**>**.status**

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
fc-loop[0-1]	simple-fc-loop	down

4.2.8 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

4.2.9 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

4.2.10 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

4.2.11 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

4.3 Standard Components

4.3.1 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

4.3.2 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.

4.3.3 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

4.3.4 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

4.3.5 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

4.3.6 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

4.3.7 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"*] [*u*

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

4.3.8 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

4.3.9 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

4.3.10 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

4.4 Other Device Components

4.4.1 sun-type5-keyboard

Description

The "sun-type5-keyboard" component represents a Sun Type 5 serial keyboard for use in Sun systems.

Commands

create-sun-type5-keyboard [*"name"*]

Creates a non-instantiated component of the class "sun-type5-keyboard". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

new-sun-type5-keyboard [*"name"*]
Creates an instantiated component of the class "sun-type5-keyboard". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<sun-type5-keyboard>.info
Print detailed information about the configuration of the device.

<sun-type5-keyboard>.status
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
device	serial	up
console	keyboard	down

4.4.2 sun-type5-mouse

Description
The "sun-type5-mouse" component represents a Sun Type 5 serial mouse for use in Sun systems.

Commands

create-sun-type5-mouse [*"name"*]
Creates a non-instantiated component of the class "sun-type5-mouse". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

new-sun-type5-mouse [*"name"*]
Creates an instantiated component of the class "sun-type5-mouse". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<sun-type5-mouse>.info
Print detailed information about the configuration of the device.

<sun-type5-mouse>.status
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
device	serial	up
console	mouse	down

4.5 Timing Components

4.5.1 sample-gcache

Description

A pre-configured combined L1 instruction and data cache

Commands

create-sample-gcache [*"name"*]

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

new-sample-gcache [*"name"*]

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

<**sample-gcache**>**.info**

Print detailed information about the configuration of the device.

<**sample-gcache**>**.status**

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
cpu-space	timing-model	up

4.5.2 sample-ma-model

Description

A sample SPARC MAI model with a simple cache

Commands

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

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

<**sample-ma-model**>**.info**
Print detailed information about the configuration of the device.

<**sample-ma-model**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	**Type**	**Direction**
cpu-space	timing-model	up

4.5.3 sample-ooo-model

Description

A sample SPARC MAI model based on **ooo_micro_arch** and a simple cache.

Commands

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

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

<sample-ooo-model>.info
Print detailed information about the configuration of the device.

<sample-ooo-model>.status
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
cpu-space	timing-model	up

4.6 Base Components

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

4.6.1 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.

4.6.2 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 new Disk to a SunFire Machine

1. **Add a SCSI Disk Component**

   First create a file in `[workspace]/targets/sunfire/` called `peanut-disk.simics`. In this file add the following contents:

   ```
   script-branch {
       wait-for-variable machine_defined
       local $disk = (create-std-scsi-disk size = 4256972800)
       connect-components $scsi_bus scsi-slot-2 $disk scsi-bus
   }

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

   This will run a *script branch* that first waits for the machine to be defined by the machine configuration script (included from `peanut-common.simics`). Once the *$machine_defined* variable has triggered, a SCSI disk component, representing a 4GB disk, will be created that on the following line is connected to the SCSI bus component on connector `scsi-slot-2`.

2. **Prepare the Boot**

   Start Simics, but do not start the simulation. Before booting, the disk needs an empty partition table for Solaris to recognize the disk. The partition table must contain a geometry that matches the size of the disk. Also add a large partition that covers the full disk.

   ```
   simics> sd1.create-sun-vtoc-header 5470 19 80
   simics> sd1.create-sun-vtoc-partition number = 0 start-block = 0 num-bloc
   ```

Note: Configuring partitions can also be done using the Solaris **format** command once the simulated machine has booted.

The partition table should now look something like:

```
Partition Table:
Number    Tag             Flag                  Start        End        Size
  0       2 (root)        0 (RW)                    0    8311359    8311360
  2       5 (backup)      1 (unmountable)           0    8311359    8311360
```

Before booting, tell Solaris that new hardware has been added, by adding the `-r` argument to the OBP boot command variable:

```
simics> system_cmp0.set-prom-env boot-command "boot disk1 -rv"
```

3. **Configure Solaris**

   Start the simulation and wait for the simulated machine to reach the prompt. A file system has to be created on the new disk, this is done using the Solaris **newfs** command. At the same time, also add a mount point, and an entry in the file-system table. This way Solaris will automatically mount the disk on the next boot.

```
# newfs /dev/dsk/c0t2d0s0
newfs: construct a new file system /dev/rdsk/c0t2d0s0: (y/n)? y

<output from newfs here>

# mkdir /disk
# cat >> /etc/vfstab
/dev/dsk/c0t2d0s0    /dev/rdsk/c0t2d0s0   /disk ufs  2    yes    -
<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 Solaris to shut down, stop the simulation and save the all modifications using the **save-persistent-state** command. But first remove `-r` from the boot command, or it will be for the next boot as well, making the boot slower.

```
simics> system_cmp0.set-prom-env boot-command "boot disk1 -v"
simics> save-persistent-state new-disk1.state
```

Now exit Simics, and restart the `peanut-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

6.1 Installing Solaris on Simics

Solaris can be installed directly on the simulated machine in Simics. Solaris can be obtained from Sun's web-site at `http://www.sun.com/software/solaris/binaries/get.html` in the form of ISO images.

To simplify the installation process, some scripts are supplied with the Simics distribution for the peanut machine: `peanut-sol<version>-cd-install1.simics`, `peanut-sol<version>-cd-install2.simics` and `peanut-sol<version>-cd-install3.simics`, where <version> is `8`, `9` or `10`. The scripts will answer all questions automatically to create a standard workstation install.

It is also possible to install Solaris 2.6 and Solaris 7 on Simics, but for that, no automatic scripts are supplied.

6.1.1 Installation, step by step

This section describes how to install Solaris using the command-line version Simics.

1. Select the install script to use, depeding on Solaris version to install, either `peanut-sol10-cd-install1.simics` for Solaris 10, `peanut-sol9-cd-install1.simics` for Solaris 9, or `peanut-sol8-cd-install1.simics` for Solaris 8.

2. Set the path to the CD image in the simics script. The line

   ```
   $cdrom_path = "sol-10-u2-ga-sparc-v1.iso"
   ```

   should be changed to reflect the location and name of the CD image for stage one of the installlation. It can either be an ISO image file, or a CD-ROM device file (Linux and Solaris host only).

3. Start the first installation script, for example:

   ```
   $ ./simics targets/sunfire/peanut-sol10-cd-install1.simics
   ```

and wait for it to complete. This may take several hours, depending on the performance of the host machine.

4. When the script stops, installation from the first CD is finished, and Solaris has tried to reboot the system. Since Simics does not support system reboot for this architecture, exit Simics at this point.

   If the installation is performed from a real CD, it is now time to change disc in the drive. Also make sure that the path to the CD is correct in the second install script.

5. Now run the second script in the same way as the first, this script may also take a few hours to complete.

6. When the second script has stopped, run the third and last one. This script only takes a few minutes to finish.

7. When the third script has stopped the installation is ready. The newly created disk image has the following file name: `peanut-sol<version>-install.disk`. There are also a number of *persistent state* files.

8. To boot a machine with the newly installed Solaris OS, run the `peanut-common.simics` and make sure that the variable `$os` is set to `"solaris10"`, `"solaris9"` or `"solaris8"` depending on the operating system version installed). Add a line like the following first in that simics script:

   ```
   $os = "solaris10"
   ```

9. An optional last step is to compress the disk image with the `craff` utility to save some disk space.

6.2 Installing Linux or another OS on Simics

The files `walnut-cd-install1.simics` and `walnut-cd-install2.simics`, that are supplied with the walnut simulated machine provide a way to install an operating system on Simics from a bootable CD-ROM. Before starting, please read the instructions in `walnut-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/sunfire/walnut-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/sunfire/walnut-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 `walnut-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 3.6.

Chapter 7

Miscellaneous Notes

7.1 Notes on Solaris for SunFire

- For information about system administration of Solaris, see the `http://docs.sun.com` web site.
- Remember to boot with the `-r` flag after changing a machine configuration. Example:

```
simics> system_cmp0.set-prom-env boot-command "boot disk1 -rv"
```

- Booting from a disk in a different location that it was setup for is not recommended. It is possible, but requires some knowledge of Solaris administration.
- When using multiple Ethernet adapters in a Serengeti system, all will be assigned the same system-wide MAC address by Solaris. To avoid this, the OBP variable `local-mac-address?` can be set to `true`. Setting this variable from the Simics command-line is done using the following command:

```
simics> system_cmp0.set-prom-env local-mac-address? true
```

7.2 Multiple Network Devices

By default, only the first network device is connected to a simulated network when running with `$create_network` set to `yes`. To run with multiple network devices, the <**device**>**.connect** command should be used to connect each additional device to the Ethernet link. If several network devices have the same MAC address (default unless the `local-mac-address?` OBP variable is set) they must be connected to different simulated links.

7.3 Notes on Linux for SunFire

- There is a bug in the 2.4.0 - 2.4.18 versions of the Linux linux kernel that disables serial interrupts some time into the boot. This bug is not present in the 2.2 kernels, or in 2.4.19 and newer.

  The workaround, already applied to the bagle machine, is to set the `workaround_linux_bug` attribute to 1 in the `fhc0` object.

- A bug in all Linux kernels prior to version 2.4.19 requires the machine to have devices on both SYSIO controllers on an SBus board. The workaround is to add some device to slot 1 or 2 that is empty by default in Simics. This workaround is already present in the bagle machine configuration.

  On real machines there is an on-board SOC controller on one of the SYSIOs, and 'fas' + 'hme' on the other, hiding this bug. But Simics does not model the SOC device.

- SuSE Linux 7.3 incorrectly expects PC keyboard codes although the kernel correctly has identified a Sun Type 5 keyboard. The following workaround, that is already applied to the bagle scripts, will allow keyboard input in the graphical console. (Note: not needed for the text based xterm console). If X is started, the `send_pc_codes` attribute should be set to 0 again since the XFree86 keyboard handler uses Sun Type 5 keycodes.

  ```
  @conf.kbd.send_pc_codes = 1
  ```

- Linux 2.4.0 - 2.4.18 does not map interrupts correctly for devices below a PCI-to-PCI bridge. As a result, most PCI boards listed earlier in this document may not work when running Linux on the simulated machine. This problem is fixed starting with version 2.4.19 of the Linux kernel.

- Some versions of SILO (The Linux loader for SPARC) allocates large MMU pages for the kernel. There is a bug in many Linux kernels causing it to panic with "Trap 5", when mapped on a large MMU page.

  Detailed information: The kernel tries to find the TLB entry of the current PC by masking the PC with an 8K page mask (0x1fff). But SILO has mapped the kernel in an 4M page. Since the PC isn't in the first 8K of this page when this happens, the kernel will never find any matching TLB entry and it then panics by running a "ta 5" instruction.

  This bug is known to exist in early 2.6 kernels, at least up to 2.6.8. There exists a workaround, but it has to be adapted for each kernel version.

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

- The following UltraSPARC registers are not implemented:
  - The ECC error registers.
  - Cache diagnostic registers.
  - Performance control registers, and counters.
- Simics does not run the real POST, and as a result system management such as dynamic reconfiguration is not supported.
- System reset is not supported.
- Multi-machine configurations with graphical consoles do not work on Windows hosts since it uses SDL that only supports one window.

8.2 Other Limitations

- The Solaris version of SimicsFS does not support truncating files.

Index

T

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