virtutech

Simics/Fiesta Target Guide

Simics Version 3.0

Revision 1376
Date 2007-01-24

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/Fiesta Overview

2.1 Introduction

Simics/Fiesta models the Sun Blade 1500 workstation, also called Taco, with an UltraSPARC-IIIi processor. The other workstation based on the Fiesta architecture, Sun Blade 2500 (Enchilada), is not supported in Simics. Taco supports a single processor and up to 8GB of memory. The Fiesta system is build around the Tomatillo host-to-PCI bridge, and the ALi MD1535D+ south bridge with several on-board devices, such as IDE harddisks and an AT-API CD-ROM.

Virtutech does not provide any disk images with Solaris for Fiesta, 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

A Taco workstation has several on-board devices on the mother board as well as five empty PCI slots. Not all devices are currently supported in Simics, a full list can be found in chapter 7, Limitations.

On-board Devices

Tomatillo Host-to-PCI bridge	
MD1535D+ South Bridge	
Two IDE controllers	
Two serial ports	(NS16550)

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)

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/`

Note:
The Simics/Fiesta model does not currently support a keyboard or mouse, and thus the pgx64 card cannot be used to create an interactive X session. It can be used as a passive display by starting an X server in no mouse and no keyboard mode.

Chapter 3

Simulated Machines

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

3.1 Chili

Chili is a Sun Blade 1500 workstation with a single UltraSPARC IIIi processor running at 96 MHz, and 256 MB of memory. It has one Ethernet adapter, one IDE disk and one IDE CD-ROM. The default configuration can be modified as described in section 3.3. An operating system must be installed on chili before it can be used.

3.1.1 Chili Scripts

`chili-common.simics`
Starts the Chili machine with the default configuration.

`chili-sol<version>-cd-install.simics`
Script for installing Solaris on the simulated machine. <version> is one of 8, 9 and 10.

3.2 Salsa

Salsa is a Sun Blade 1500 workstation with a single UltraSPARC IIIi processor running at 96 MHz, and 256 MB of memory. It has one Ethernet adapter, one IDE disk and one IDE CD-ROM. The default configuration can be modified as described in section 3.3.

The Salsa 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 (7/03) and Solaris 9 (4/04) installed as "Developer System" directly on Simics.

- SimicsFS support.
- Login `root`, no password.
- Configured with static IP address 10.10.0.18, gw 10.10.0.1, when DHCP not used.

3.2.1 Salsa Scripts

`salsa-common.simics`
Starts the Salsa machine with the default configuration.

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

3.3 Parameters for Machine Scripts

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

3.3.1 chili-common and salsa-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.

$os
The operating system to boot, one of `solaris10`, `solaris9`, and `solaris8`. Requires that a matching disk dump exists.

$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 Fiesta architecture. There also exist other components in Simics, such as various PCI devices, that may work for Fiesta but that have not been tested.

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

4.1.1 taco-system

Description

The "taco-system" component represents a Sun Blade 1500 single processor workstation mother-board with all on-board devices except the south-bridge and PCI devices.

PCI slot mappings:

Simics slot	Tomatillo	Bus	Device id	Bus address	Note
0	31	A	2	1e,600000	
1	31	A	3	1e,600000	
2	31	A	4	1e,600000	
3	31	A	5	1e,600000	
4	31	B	2	1f,700000	bge (BCM5703C)
5	31	B	3	1f,700000	

Attributes

cpu_frequency

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

Processor frequency in MHz.

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.

memory_megs
Pseudo attribute; **read-only** access; type: **Integer**.
The amount of RAM in mega-bytes in the machine.

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

Commands

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

<**taco-system**>**.get-nvram-hostid**
Reads the Sun hostid from the NVRAM.

<**taco-system**>**.get-nvram-mac**
Reads the default MAC address from the NVRAM.

<**taco-system**>**.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.

<**taco-system**>**.info**
Print detailed information about the configuration of the device.

<**taco-system**>**.set-nvram-hostid *hostid***
Writes the Sun hostid into the NVRAM.

<**taco-system**>**.set-nvram-mac *"mac"***
Writes the default MAC address into the NVRAM.

<**taco-system**>**.set-prom-defaults**
Restores all OBP variables to their default values.

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

<**taco-system**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	**Type**	**Direction**
ddr-slot[0-3]	mem-bus	down
i2c	i2c-bus	down
pci-slot-ide	pci-bus	down
pci-slot-isa	pci-bus	down
pci-slot-pm	pci-bus	down
pci-slot[0-5]	pci-bus	down

4.1.2 south-bridge-sun-md1535d

Description
The "south-bridge-sun-md1535d" component represents the ALi MD1535D south-bridge modified for use in a Fiesta (Taco/Enchilada) system.

Commands

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

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

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

Connectors

Name	Type	Direction
i2c	i2c-bus	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

4.2 PCI Device Components

4.2.1 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

4.2.2 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

4.2.3 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

4.2.4 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

4.2.5 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

4.2.6 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

4.2.7 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

4.2.8 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

4.2.9 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

4.2.10 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 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

4.3.2 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

4.3.3 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

4.3.4 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.

4.3.5 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

4.3.6 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

4.3.7 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

4.3.8 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

4.3.9 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

4.3.10 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

4.3.11 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

4.3.12 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

4.3.13 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

4.3.14 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

4.4 Timing Components

4.4.1 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

4.4.2 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

4.4.3 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

4.5 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

Miscellaneous Notes

5.1 Notes on Solaris for Fiesta

- 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
```

5.2 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.

5.3 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 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 chili machine: `chili-sol<version>-cd-install.simics`, where <version> is `8`, `9` or `10`. The scripts will answer all questions automatically to create a standard workstation install.

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 `chili-sol10-cd-install.simics` for Solaris 10, `chili-sol9-cd-install.simics` for Solaris 9, or `chili-sol8-cd-install.simics` for Solaris 8.

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

   ```
   $cdrom_path[$idx] = ("sol-10-u2-ga-sparc-v" + $idx + ".iso")
   ```

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

3. Start the first installation script, for example:

   ```
   $ ./simics targets/fiesta/chili-sol10-cd-install.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, the installation is ready. The newly created disk image has the following file name: `chili-sol<version>-install.disk`.

5. To boot a machine with the newly installed Solaris OS, run the `chili-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"
```

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

Chapter 7

Limitations

7.1 Limitations of the Simulated Model

- The following UltraSPARC registers are not implemented:
  - The ECC error registers.
  - Cache diagnostic registers.
  - Performance control registers, and counters.
- The following on-board devices are not modeled in Simics.
  - Missing from the MD1535D+ South Bridge:
    - USB ports.
    - AC97 Sound Card.
    - Modem.
    - Parallel port.
  - Smart card reader.
- Most legacy devices in the MD1535D+ South Bridge have hard-coded addresses in Simics that can not be reconfigured by target software.
- Several registers in the MD1535D+ south-bridge are not yet implemented.
- Changing the processor frequency is not yet supported.

7.2 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