Simics/AlphaPC-164LX Target Guide

Simics Version	3.0
Revision	1406
Date	2008-02-19

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/Alpha Overview

2.1 Introduction

Simics/Alpha models machines based on the Alpha 21164 (EV5) processor and the DEC 21174 (also known as "Pyxis") chipset. Only Linux is supported as target operating system.

2.2 Supported Hardware

The simulated machines are similar to an AlphaPC 164LX, which is an OEM design that combins the Alpha 21164 processor with the 21174 chipset and a number of legacy PC components (behind a PCI-to-ISA south bridge).

Supported Devices	
Ethernet controller	(DEC21041)
Ethernet controller	(DEC21140A)
Ethernet controller	(DEC21143)
SCSI controller	(SYM53C810)
SCSI controller	(SYM53C875)
Graphics Adapter	(Voodoo3)

Chapter 3

Simulated Machines

Simics scripts for starting AlphaPC-164LX machines are located in the `[workspace]/targets/alphapc-164/` directory, while the actual configuration scripts can be found in `[simics]/targets/alphapc-164/`.

3.1 Torus

Torus is an AlphaPC 164LX workstation with a single Alpha 21164 processor running at 5 MHz and 48 MB of memory. It has one SCSI disk and one SCSI CD-ROM, but no network device. The default configuration can be modified as described in section 3.2.

Torus is configured for an existing Red Hat Linux 6.0 disk dump, that can be downloaded from the Virtutech web site.

Additional information:

- Red Hat 6.0 Linux.
- Linux kernel 2.2.5
- SimicsFS support (read-only, write support experimental).
- Login `root`, password `virtualpha`.

3.1.1 Torus Scripts

`torus-common.simics`
Starts the Torus machine with the default configuration.

3.2 Parameters for Machine Scripts

The following parameters can be set before running the `torus-common.simics` script. Other `.simics` scripts may set some of the parameters unconditionally, and do not allow the user to override them.

3.2.1 torus-common

$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 MILO 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.

$freq_mhz
The clock frequency in MHz for the processor.

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

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

$text_console
Set to "yes" in order to use a text console with the VGA device (default), "no" creates a graphical console.

Chapter 4

Supported Components

The following sections list components that are supported for the AlphaPC-164LX architecture. There also exist other components in Simics, such as various PCI devices, that may work for AlphaPC-164LX but that have not been tested.

The default machines are constructed from components in the `-system.include` files in `[simics]/targets/alphapc-164/`. 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 Alpha Components

4.1.1 alpha-lx164-system

Description

The "alpha-lx164-system" component represents a single-processor Alpha LX164 system with a Pyxis PCI north bridge.

Attributes

bios

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

The boot BIOS file to use.

cpu_frequency

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

Processor frequency in MHz.

memory_megs

Required attribute; **read/write** 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-alpha-lx164-system [*"name"*] *cpu_frequency memory_megs "rtc_time"* [*"bios"*]

Creates a non-instantiated component of the class "alpha-lx164-system". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<alpha-lx164-system>.info
Print detailed information about the configuration of the device.

<alpha-lx164-system>.status
Print detailed information about the current status of the device.

Connectors

Name	**Type**	**Direction**
interrupt	alpha-interrupt	down
pci-slot[5-9]	pci-bus	down

4.1.2 south-bridge-saturn

Description
The "south-bridge-saturn" component represents a Saturn PCI based south bridge for use in Alpha LX164 systems. It includes the common legacy PC devices, two IDE controllers, a real-time clock and a floppy controller with two drives attached.

Commands

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

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

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

Connectors

Name	Type	Direction
interrupt	alpha-interrupt	up
pci-bus	pci-bus	up
ide0-master	ide-slot	down
ide0-slave	ide-slot	down
ide1-master	ide-slot	down
ide1-slave	ide-slot	down
isa-bus	isa-bus	down

4.2 PCI Device Components

4.2.1 pci-sym53c810

Description

The “pci-sym53C810” component represents a SYM53C810PCI based SCSI controller.

Attributes

bios

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

The x86 SCSI BIOS file to use.

Commands

create-pci-sym53c810 [*“name”*] [*“bios”*]

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

<pci-sym53c810>.info

Print detailed information about the configuration of the device.

<pci-sym53c810>.status

Print detailed information about the current status of the device.

Connectors

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

4.2.2 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

4.2.3 pci-dec21041

Description

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

Attributes

bios

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

The x86 BIOS file to use.

mac_address

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

The MAC address of the Ethernet adapter.

Commands

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

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

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

Connectors

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

4.2.4 pci-dec21140a

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

Attributes

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

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

Commands

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

<pci-dec21140a>.info
Print detailed information about the configuration of the device.

<pci-dec21140a>.status
Print detailed information about the current status of the device.

Connectors

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

4.2.5 pci-dec21143

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

Attributes

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

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

Commands

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

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

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

Connectors

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

4.2.6 pci-voodoo3

Description

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

Commands

create-pci-voodoo3 [*“name”*]

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

<pci-voodoo3>.info

Print detailed information about the configuration of the device.

<pci-voodoo3>.status

Print detailed information about the current status of the device.

Connectors

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

4.3 PC Legacy Components

4.3.1 ps2-keyboard-mouse

Description

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

Commands

create-ps2-keyboard-mouse [*“name”*]

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

<ps2-keyboard-mouse>.info

Print detailed information about the configuration of the device.

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

Connectors

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

4.3.2 pc-dual-serial-ports

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

Commands

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

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

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

Connectors

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

4.4 Standard Components

4.4.1 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

4.4.2 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.

4.4.3 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

4.4.4 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

4.4.5 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

4.4.6 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

4.4.7 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

4.4.8 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

4.4.9 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

4.4.10 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

4.5 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

Limitations

5.1 Limitations of the Simulated Model

- VAX floating-point formats are unsupported. This includes:
  - F_floating
  - G_floating
  - D_floating

  Note that VAX floating-point operate and memory instructions are an optional subset group in the Alpha architecture.

- The Alpha architecture provides a choice of three different computational models within the IEEE floating-point subset. Simics does not make this distinction, and floating-point operations within Simics can best be described as *IEEE-Compliant Arithmetic Without Inexact Exception*.

- Some floating-point instructions may return approximative results, and may ignore trapping and rounding mode settings.

- NT related features in the processor models are unsupported. For the Alpha 21164 this includes (but not limited to):
  - NT mode of the IFAULT_VA_FORM register
  - NT mode of the IVPTBR register
  - NT mode of the VA_FORM register
  - NT mode for superpage mapping as controlled by the IDU register

Index