Simics/ARM-SA1110 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/ARM-SA1110 Overview

2.1 Introduction

Simics/ARM-SA1110 models an ARMv5 processor, memory and some of the devices of the Intel StrongARM processor. It is loosely based on the Intel Assabet test board.

2.2 Supported Hardware

The Simics/ARM-SA1110 consists of a 32-bit StrongARM RISC Processor with a memory controller. None of the SA1110 devices are fully supported.

Chapter 3

Simulated Machines

Simics scripts for starting ARM-SA1110 machines are located in the `[workspace]/targets/arm-sa1110/` directory, while the actual configuration scripts can be found in `[simics]/targets/arm-sa1110/`.

3.1 sa1110-linux

The sa1110-linux is a simple Intel StrongARM system, with a single ARMv5 processor running at 30 MHz, and 32 MB of memory. It has a single serial port with a text console connected to it.

The sa1110-linux machine runs a Linux 2.4.12 kernel with a RAM disk, that both are loaded directly into the main memory, since there is no storage device modelled. There is no SimicsFS support in the supplied Linux kernel. As no bootloader is run, a faked bootloader information structure is also set up.

The following files are also required by sa1110-linux, but are not included in the Simics distribution. They must be downloaded separately from `http://www.simics.net/download` and placed in the `[simics]/import/arm` directory.

File	**Description**
`vmlinux`	Linux kernel
`initrd`	RAM disk

3.1.1 sa1110-linux Scripts

`sa1110-linux-common.simics`
Starts the sa1110-linux machine with the default configuration.

3.2 Parameters for Machine Scripts

The following parameters can be set before running the `sa1110-linux-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 sa1110-linux-common

$freq_mhz
The clock frequency in MHz for the processor.

$initrd
The file containing the initrd ram disk for Linux. If set, this file is loaded directly into memory at address 0xc0400000.

$vmlinux
The file containing the Linux kernel. If set, this file is loaded directly into memory, and the program counter is set to point to the entry point of the file.

Chapter 4

Supported Devices

The following devices are partially supported: gpio (General purpouse IO), ic (interrupt controler) ost (OS timer), ppc (peripheral pin controller), rtc (real time clock), sp1 (serial port 1 - UART).

Chapter 5

Supported Components

The following sections list components that are supported for the ARM-SA1110 architecture. There also exist other components in Simics, such as various PCI devices, that may work for ARM-SA1110 but that have not been tested.

The default machines are constructed from components in the `-system.include` files in `[simics]/targets/arm-sa1110/`. 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.

5.1 ARM SA1110 Components

5.1.1 arm-sa1110-system

Description

The "arm-sa1110-system" component represents a simple ARM SA1110 system with a single processor and a serial device.

Attributes

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

memory_megs
Optional attribute; **read/write** access; type: **Integer**.
Size of RAM (mapped from 0xc0000000) in MB.

Commands

create-arm-sa1110-system [*"name"*] *cpu_frequency* [*memory_megs*]
Creates a non-instantiated component of the class "arm-sa1110-system". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

new-arm-sa1110-system [*"name"*] *cpu_frequency* [*memory_megs*]
Creates an instantiated component of the class "arm-sa1110-system". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<arm-sa1110-system>.info
Print detailed information about the configuration of the device.

<arm-sa1110-system>.status
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
console	serial	down

5.2 Standard Components

5.2.1 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

5.2.2 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

Chapter 6

Installing Linux or another OS on Simics/ARM

If you want to install another operating system on Simics/ARM, it must be a statically linked binary in either ELF format or some format that can be loaded straight into memory without modification.

If you have an ELF binary that you want to install, you use the **load-binary** command. It loads an ELF file into memory and performs the necessary relocation. The `arm-sa1110.simics` script uses the **load-binary** command to load the Linux kernel directly into the memory of the simulated machine.

If you instead have a binary in some format that can be loaded straight into memory without modification, you can use the **load-file** command. The `arm-sa1110.simics` script uses the **load-file** command to load the RAM disk and faked bootloader information directly into the memory of the simulated machine.

If you are trying to install another Linux kernel, you must use the plain `vmlinux` file, not the `Image` file used to install Linux on a system that boots from a disk.

Note: Standard Linux configurations for StrongARM machines will likely not work. Support for devices not present in Simics/ARM will have to be removed for the kernel to work. It is actually unlikely that another operating system than Linux will work on the arm-sa1110, especially if it uses or expects any particular devices to be present.

Chapter 7

Miscellaneous Notes

7.1 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 Simics Features

Simics/ARM features most of the generic Simics features. There are, however, a number of features not yet supported by Simics/ARM:

User decoder
It is currently not possible for the user to define or override instructions in Simics/ARM.

User MMU
It is currently not possible for the user to override the MMU in Simics/ARM.

Memory hierachy
It is currently not possible to connect a simulated memory hierachy to Simics/ARM.

Tracing
It is currently not possible to do tracing of memory accesses with Simics/ARM.

Networking
It is currently not possible to connect Simics/ARM to a simulated network.

Symtable
The **symtable** module has not been updated to support the ARM ABI and is therefore only partially supported.

Haps
Many haps are not implemented. This means that the **break-hap** command does not work on these haps and that modules cannot depend on these unimplemented haps.

8.2 ARM ISA

Simics/ARM has a fairly complete implementation of the core ARMv5 instruction set. The Thumb and enhanced DSP extensions are not implemented.

Only the instructions and functions of the system control coprocessor that are needed to boot Linux are implemented.

No floating-point coprocessor is implemented.

8.3 ARM Miscellaneous

Simics/ARM currently only supports little-endian mode.

Index

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