ERC32 VHDL models user's manual

Version 1.0
October 1996

Jiri Gaisler
European Space Research and Technology Centre (ESA/ESTEC)

European Space Agency

jgais@ws.estec.esa.nl

ERC32 VHDL models user's manual

1 Introduction

This document describes the ERC32 VHDL models. Discussions are provided for the following topics:

- contents and directory structure of the models
- installation and compilation
- usage of the models
- discrepancies between the models and the real devices
- technical support

The ERC32 VHDL models consists of fully functional, timing accurate VHDL models of the TSC691E integer unit, TSC692E floating-point unit and TSC693E memory controller. These models are intended to be used in board level simulation and verification of ERC32-based systems. Due to the complexity of the models, they are to typically too slow to be used for software validation; it is better to use simulators like SIS or erc32sim for this task.

The models are distributed under GNU library general public license (GLGPL). Briefly, this means that anyone may use the models in any way he might see fit, but modification and re-distribution must be performed according to the GLGPL. The complete license agreement is provided in the distribution in file COPYING.LIB. The users are especially asked to take notice of paragraph 15 and 16:

2 Installation and directory structure

2.1 Obtaining ERC32 VHDL models

The ERC32 models are only available via anonymous ftp. The primary home of the models is the ESTEC ftp-server at URL: ftp://ftp.estec.esa.nl/pub/ws/wsd/erc32/erc32/vhdl. The models are distributed in an compressed tarfile; `erc32vhdl-1.0.tar.gz`.

2.2 Installation

After obtaining the compressed tarfile, uncompress and untar it in a suitable location. Following commands can be used on most UNIX systems:

```
gunzip -c erc32vhdl-1.0.tar.gz | tar xf -
```

This will create the `erc32vhdl` directory in the current directory. The `erc32vhdl` directory contains the following files:

COPYING.LIB	fpurt_lib.vhd	sparc_lib.vhd
MECNom.tim	iurt_lib.vhd	stdsim.dft
board.vhd	meclibrary.vhd	synopsys_vss.setup
compile.modeltech	memory.vhd	vsystem.ini
compile.synopsys	mms.vhd	work
factlib.vhd	prom.dat	

The models have been verified on two simulators, modeltech VSIM v4.6g and synopsys VSS v3.4b. The files `compile.modeltech` and `compile.synopsys` contains a script to compile the models for the two simulators. For other simulators, modify these scripts according to your needs. The models define and make use of a number of libraries; the files vsystem.ini and synopsys_vss.setup provides the library mapping for modeltech and synopsys simulators. The models are divided into files according to which library they are called from:

mms.vhd	MMS
sparc_lib.vhd	SPARC_LIB
iurt_lib.vhd	IURT_LIB
fpurt_lib.vhd	FPURT_LIB
meclibrary.vhd	MECLIBRARY
memory.vhd	MEMORY
board.vhd	WORK

3 Usage

3.1 System simulation

For detailed information on how the models work, please refer to the data sheets of the individual devices. An example on how to use the models in system can be found in `board.vhd`. This file contains the model of a simple ERC32-based computer, consisting of address and data buffers, RAM and boot-prom. The boot-prom is loaded with the contents of the file `prom.dat` at reset time. To simulate this system, first compile all models using one of the compile scripts. The configuration to simulate is `BOARD_CFG`, for the modeltech simulator the start command would be:

```
vsim board_cfg &
```

The provided prom.dat contains a program that will run a number of IU and FPU instructions and then print “TEST OK” five times on the console if no errors were detected. The simulator needs to be run for 1,511,000 ns to complete the program (this may take up to 20 minutes on some simulators). The file `stdsim.dft` and `MECNom.tim` needs to be in the simulation directory (see below). Note that the modelled system in `board.vhd` runs at 14 MHz and requires one waitstate during write to operate correctly.

3.2 Timing

The models include timing as defined in the device data sheets. The file stdsim.dft can be used to specify different process, temperature, voltage and load conditions. Below is a listing of a sample stdsim.dft:

```
CHECK_ON 0
ENVIRONMENT_BOARD 2
SIM_BOARD 3
LOAD_BOARD 50
T_BOARD_SPECIFIC 37
V_BOARD_SPECIFIC 5000
PROCES_BOARD_SPECIFIC 0

-------------------------------------------
-- CHECK_ON : switch on/off timing checkers
--  0 : off (FALSE)
--  1 : on  (TRUE)
-------------------------------------------
-- ENVIRONMENT_BOARD : environment conditions
--  0 : COMMERCIAL
--  1 : INDUSTRIAL
--  2 : MILITARY
-------------------------------------------
-- SIM_BOARD : simulation conditions
--  0 : SPECIFIC
--  1 : MINIMUM
--  2 : TYPICAL
--  3 : MAXIMUM
-------------------------------------------
-- LOAD_BOARD : default load value in pF
```

```
-------------------------------------------
-- T_BOARD_SPECIFIC : specific default temperature in Celsius
-------------------------------------------
-- V_BOARD_SPECIFIC : specific default voltage in mV
-------------------------------------------
-- PROCES_BOARD_SPECIFIC : specific default process
--  0 : TYPICAL
--  1 : BEST
--  2 : WORST
-------------------------------------------
-- if SIM_BOARD = SPECIFIC, the specific values are used
-- They need not to be declared if not in specific mode
-------------------------------------------
```

The file `MECNom.tim` contains the worst-case timing for the MEC according to the data sheet. These timings are then scaled according to the conditions defined in `stdsim.dft`. Changes to `stdsim.dft` and `MECNom.tim` file can be done without recompiling the models; the files are read on each invocation of the simulator.

3.3 Model discrepancies

Some behaviour of the real devices have not been modelled completely, the paragraphs below indicate the known differences between the models and the devices.

3.3.1 Integer Unit

Co-processor interface. The IU takes a co-processor disabled trap whenever a co-processor instruction is encountered, regardless of the CP_N input. The model is not sensitive to the co-processor interface signals CCC, CEXC_N and CXACK, but will hold the pipeline when CHOLD_N or CCCV are asserted. The CINS1/2 are never asserted.

Hardware interlock. Hardware interlock is not generated for CALL and JMPL. Interlock is also not generated for LDD with a dependency on `rd` but not `rd+1`.

3.3.2 Floating-point Unit

Instruction timing. In the real device, FPU instructions execute in a varying number of cycles depending on the operands. In the model, the FPU instructions execute in the number of cycles as defined in the *typical* case in the FPU data sheet.

3.3.3 Memory Controller

CLK2 to SYSCLK delay. The delay between CLK2 and SYSCLK is not modelled. Typically, CLK2 is not used outside the MEC and the CLK2/SYSCLK delay can be ignored. The timing requirements for signals related to CLK2 has been transformed into requirements related to SYSCLK, taking into account the specified CLK2/SYSCLK delay in the MEC data sheet.

4 Technical support

The models are provided “as is” with no technical support except bug fixes. The IU, FPU and MEC data sheets should be used to understand how the models behave. ESTEC in general and Jiri Gaisler in particular cannot provide VHDL or application engineering support. Use the provided board design (board.vhd) as a template on how to incorporate the models in your system.

The models have been used extensively during the development of the ERC32 devices and are believed to be very accurate with few or no remaining bugs. Nevertheless, if you think you have found a bug, document it thoroughly and report it to jgais@ws.estec.esa.nl. If you also have fixed the bug, submit the modified code as well so that other users can benefit from it. After all, the idea of free software is that you share your efforts with others...