OVP Guide to Using Processor Models

Model Specific Information for variant ALTERA_Nios_II_E

Imperas Software Limited
Imperas Buildings, North Weston
Thame, Oxfordshire, OX9 2HA, UK
docs@imperas.com

Author	Imperas Software Limited
Version	0.4
Filename	OVP_Model_Specific_Information_nios_ii_Nios_II_E.pdf
Created	1 May 2014

Copyright Notice

Right to Copy Documentation

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

IMPERAS SOFTWARE LIMITED., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Table of Contents

1.0 Overview

This document provides the details of an OVP Fast Processor Model variant.

OVP Fast Processor Models are written in C and provide a C API for use in C based platforms. The models also provide a native interface for use in SystemC TLM2 platforms.

The models are written using the OVP VMI API that provides a Virtual Machine Interface that defines the behavior of the processor. The VMI API makes a clear line between model and simulator allowing very good optimization and world class high speed performance.

Most models are provided as a binary shared object and also as source. This allows the download and use of the model binary or the use of the source to explore and modify the model.

The models are run through an extensive QA and regression testing process and most model families are validated using technology provided by the processor IP owners.

There is a companion document (OVP Guide to Using Processor Models) which explains the general concepts of OVP Fast Processor Models and their use. It is downloadable from the OVPworld website documentation pages.

1.1 Description

Nios_II Family Processor Model.

1.2 Licensing

Open Source Apache 2.0

1.3 Limitations

No Custom instructions.
No Cache model.
No JTAG.

1.4 Verification

Models have been extensively tested by Imperas, and validated against tests from Altera.

1.5 Features

Barrel Shifter.

2.0 Configuration

2.1 Location

The model source and object file is found in the VLNV tree at:
altera.ovpworld.org/processor/nios_ii/1.0

2.2 GDB Path

The default GDB for this model is found at:
$IMPERAS_HOME/lib/$IMPERAS_ARCH/gdb/nios2-elf-gdb

2.3 Semi-Host Library

The default semi-host library file is found in the VLNV tree at :
altera.ovpworld.org/semihosting/nios_iiNewlib/1.0

2.4 Processor Endian-ness

This is a LITTLE endian model.

2.5 QuantumLeap Support

This processor is qualified to run in a QuantumLeap enabled simulator.

2.6 Processor ELF Code

The ELF code supported by this model is: 0x71

3.0 Other Variants in this Model

Table 1.

Variant
Nios_II_F
Nios_II_S
Nios_II_E

4.0 Bus Ports

Table 2.

Type	Name	Bits
master (initiator)	INSTRUCTION	32
master (initiator)	DATA	32

5.0 Net Ports

Table 3.

Name	Type
reset_n	input
d_irq0	input
d_irq1	input
d_irq2	input
d_irq3	input
d_irq4	input
d_irq5	input
d_irq6	input
d_irq7	input
d_irq8	input
d_irq9	input
d_irq10	input
d_irq11	input
d_irq12	input
d_irq13	input
d_irq14	input
d_irq15	input
d_irq16	input
d_irq17	input
d_irq18	input
d_irq19	input
d_irq20	input
d_irq21	input
d_irq22	input
d_irq23	input

d_irq24	input
d_irq25	input
d_irq26	input
d_irq27	input
d_irq28	input
d_irq29	input
d_irq30	input
d_irq31	input

6.0 FIFO Ports

No FIFO Ports in this model.

7.0 Parameters

Table 4.

Name	Type	Description
verbose	Boolean	Specify verbose output messages
TEST_MODE_EXIT	Boolean	Enable Test mode exit for instruction, cmpltui r0, r0, (0xabc1||0xabc2)
TEST_HALT_EXIT	Boolean	Enable a Halt on 'br 0', branch to self
INST_ADDR_WIDTH	Uns32	Instruction bus Width
DATA_ADDR_WIDTH	Uns32	Data bus Width
HW_MULTIPLY	Boolean	Hardware Multiply
HW_MULX	Boolean	Hardware Extended Multiply
HW_DIVIDE	Boolean	Hardware Divide
RESET_VECTOR	Uns32	Reset Vector Address
EXCEPTION_VECTOR	Uns32	Exception Vector Address
BREAK_VECTOR	Uns32	Break Vector Address
INCLUDE_MMU	Boolean	MMU Available
FAST_TLB_MISS_EXCEPTION_VECTOR	Uns32	Fast TLB Exception Vector Address
INCLUDE_MPU	Boolean	MPU Available
INCLUDE_CPURESETREQUEST	Boolean	CPU Reset Request Signal
INCLUDE_CPURESETTAKEN	Boolean	CPU Reset Taken Signal
CPUID_CONTROL_VALUE	Uns32	CPUID Control Register Value
EXCEPTION_ILLEGAL_INSTRUCTION	Boolean	Generate Illegal Instruction Exception
EXCEPTION_DIVISION_ERROR	Boolean	Generate Division Error Exception

EXCEPTION_MISALIGNED_MEMORY_ACCESS	Boolean	Generate Misaligned Memory Access Exception
EXCEPTION_EXTRA_INFORMATION	Boolean	Generate Extra Exception information
INTERRUPT_CONTROLLER_INTERFACE	Enumeration	Interrupt Controller Interface Internal=0 External=1
NUMBER_SHADOW_REGISTER_SETS	Uns32	Number of Shadow Register Sets
HARDCOPY_COMPATIBILITY	Boolean	Hardcopy Compatibility
MMU_PID_BITS	Uns32	MMU Process ID (PID) Bits
MMU_OPTIMIZE_NUMBER_OF_TLB_ENTRIES	Boolean	MMU Optimize Number of TLB Entries based on device family
MMU_TLB_ENTRIES	Enumeration	MMU TLB Entries 128=0 256=1 512=2 1024=3
MMU_TLB_SET_ASSOCIATIVITY	Enumeration	MMU TLB Entries 8=0 16=1
MMU_MICRO_DTLB_ENTRIES	Uns32	MMU Micro data TLB Entries
MMU_MICRO_ITLB_ENTRIES	Uns32	MMU Micro instruction TLB Entries
MPU_USE_LIMIT_FOR_REGION_RANGE	Boolean	Controls Memory
MPU_NUMBER_DATA_REGIONS	Uns32	Number of Data Regions to Allocate
MPU_MINIMUM_DATA_REGION_SIZE	Uns32	Minimum Data Region Size 64Bytes to 1MBytes (power of 2)
MPU_NUMBER_INSTRUCTION_REGIONS	Uns32	Number of Instruction Regions to Allocate
MPU_MINIMUM_INSTRUCTION_REGION_SIZE	Uns32	Minimum Instruction Region Size 64Bytes to 1MBytes (power of 2)
CUSTOM_FP_DIVISION	Boolean	Enable Custom Hardware for FP Division Instruction
CUSTOM_BITSWAP	Boolean	Enable Custom Hardware for Bit Swap Instruction
CUSTOM_ENDIAN_CONVERT	Boolean	Enable Custom Hardware for Endian Conversion Instruction
CUSTOM_INTERRUPT_VECTOR	Boolean	Enable Custom Interrupt Vector Instruction

8.0 Execution Modes

Table 5.

Name	**Code**	**Description**

VM_MODE_KERNEL	0	Supervisor Mode
VM_MODE_USER	1	User Mode
VM_MODE_KERNEL_MPU	2	Supervisor Mode MPU
VM_MODE_USER_MPU	3	User Mode MPU

9.0 Exceptions

Table 6.

Name	**Code**
NONE	0
RESET	1
HARDWARE_BREAK	2
PROCESSOR_ONLY_RESET_REQUEST	4
INTERNAL_INTERRUPT	8
EXTERNAL_NONMASKABLE_INTERRUPT	16
EXTERNAL_MASKABLE_INTERRUPT	32
SUPERVISOR_ONLY_INSTRUCTION_ADDRESS	64
FAST_TLB_MISS_INSTRUCTION	128
DOUBLE_TLB_MISS_INSTRUCTION	256
TLB_PERMISSION_VIOLATION_EXECUTE	512
MPU_REGION_VIOLATION_INSTRUCTION	1024
SUPERVISOR_ONLY_INSTRUCTION	2048
TRAP_INSTRUCTION	4096
ILLEGAL_INSTRUCTION	8192
UNIMPLEMENTED_INSTRUCTION	16384
BREAK_INSTRUCTION	32768
SUPERVISOR_ONLY_DATA_ADDRESS	65536
MISALIGNED_DATA_ADDRESS	131072
MISALIGNED_DESTINATION_ADDRESS	262144
DIVISION_ERROR	524288
FAST_TLB_MISS_DATA	1048576
DOUBLE_TLB_MISS_DATA	2097152
TLB_PERMISSION_VIOLATION_READ	4194304
TLB_PERMISSION_VIOLATION_WRITE	8388608
MPU_REGION_VIOLATION_DATA	16777216

10.0 Hierarchy of the model

A CPU core may allow the user to configure it to instance many processors of a Symmetrical Multi Processor (SMP). A CPU core may also have sub elements within a processor, for example hardware threading blocks.

OVP processor models can be written to include SMP blocks and to have many levels of hierarchy.

Some OVP CPU models may have a fixed hierarchy, and some may be configured by settings in a configuration register. Please see the register definitions of this model.

This model documentation shows the settings and hierarchy of the default settings for this model variant.

10.1 Level 1:

This level in the model hierarchy has 2 commands.

This level in the model hierarchy has 2 register groups:

Table 7.

Group name	**Registers**
User	32
System	17

This level in the model hierarchy has no children.

11.0 Model Commands

11.1 Level 1:

Table 8.

Name	Arguments
dumpTLB	- no arguments
itrace	enable or disable instruction tracing

12.0 Registers

12.1 Level 1:

12.1.1 User

Table 9.

Name	Bits	Initial value (Hex)		Description
zero	32	0	r-	
at	32	0	rw	
r2	32	0	rw	
r3	32	0	rw	
r4	32	0	rw	
r5	32	0	rw	
r6	32	0	rw	
r7	32	0	rw	
r8	32	0	rw	
r9	32	0	rw	
r10	32	0	rw	
r11	32	0	rw	
r12	32	0	rw	
r13	32	0	rw	
r14	32	0	rw	
r15	32	0	rw	
r16	32	0	rw	
r17	32	0	rw	
r18	32	0	rw	
r19	32	0	rw	
r20	32	0	rw	
r21	32	0	rw	
r22	32	0	rw	
r23	32	0	rw	

et	32	0	rw	
bt	32	0	rw	
gp	32	0	rw	
sp	32	0	rw	
fp	32	0	rw	
ea	32	0	rw	
ba	32	0	rw	
ra	32	0	rw	

12.1.2 System

Table 10.

Name	**Bits**	**Initial value (Hex)**		**Description**
PC	32	0	rw	program counter
status	32	0	rw	
estatus	32	0	rw	
bstatus	32	0	rw	
ienable	32	0	rw	
ipending	32	0	rw	
cpuid	32	0	rw	
ctl6	32	0	rw	
except	32	0	rw	
pteaddr	32	0	rw	
tlbacc	32	0	rw	
tlbmisc	32	0	rw	
eccinj	32	0	rw	
badaddr	32	0	rw	
config	32	0	rw	
mpubase	32	0	rw	
mpuacc	32	0	rw	

#