OVP Guide to Using Processor Models

Model Specific Information for variant xilinx_microblaze_V7_00

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_microblaze_V7_00.pdf
Created	25 August 2015

Copyright Notice

Right to Copy Documentation

Destination Control Statement

Disclaimer

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

Microblaze Processor Model

1.2 Licensing

Apache 2.0 Open Source License

1.3 Features

Instruction Set: This model fully implements the instruction set upto and including V8.2.
Privileged Instructions: Implemented
Virtual-Memory Management: Implemented
Reset, Interrupts, Exceptions and Break: Implemented
Floating Point Unit: Implemented
Stream Link Interface: Implemented

1.4 Limitations

GDB: There is a known issue in the microblaze gdb, a lockup can occur when disassembling from address 0

1.5 ConfigurationFeatures

Barrel Shifter.
Hardware Divider.
Machine Status Set/Clear Instructions.
Hardware Exceptions.
Pattern Compare Instructions.
Floating Point Unit (FPU).
Disable Hardware Multiplier.
Processor Version Register (PVR).
Hardware Multiplier 64-bit Result.
Floating Point Conversion and Square Root Instructions.
Memory Management Unit.
Extended Stream Instructions .

1.6 Verification

Models have been validated correct by running through extensive tests using test suites and technology provided by Xilinx

2.0 Configuration

2.1 Location

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

2.2 GDB Path

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

2.3 Semi-Host Library

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

2.4 Processor Endian-ness

This model can be set to either endian-ness (normally by a pin, or the ELF code).

2.5 QuantumLeap Support

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

2.6 Processor ELF Code

ELF codes supported by this model are: , 0xbd and 0xbaab.

3.0 Other Variants in this Model

Table 1.

Variant
V7_00
V7_10
V7_20
V7_30
V8_00
V8_10
V8_20
ISA

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
Interrupt	input
Reset	input
MB_Reset	input
Ext_BRK	input
Ext_NM_BRK	input

6.0 FIFO Ports

Table 4.

Name	Bits	Type	Description
SFSL0	64	input	FSL Fifo Input port - Controlled by Parameter C_FSL_LINKS
SFSL1	64	input	FSL Fifo Input port - Controlled by Parameter C_FSL_LINKS
SFSL2	64	input	FSL Fifo Input port - Controlled by Parameter C_FSL_LINKS
SFSL3	64	input	FSL Fifo Input port - Controlled by Parameter C_FSL_LINKS
SFSL4	64	input	FSL Fifo Input port - Controlled by Parameter C_FSL_LINKS
SFSL5	64	input	FSL Fifo Input port - Controlled by Parameter C_FSL_LINKS
SFSL6	64	input	FSL Fifo Input port - Controlled by Parameter C_FSL_LINKS
SFSL7	64	input	FSL Fifo Input port - Controlled by Parameter C_FSL_LINKS
SFSL8	64	input	FSL Fifo Input port - Controlled by Parameter C_FSL_LINKS
SFSL9	64	input	FSL Fifo Input port - Controlled by Parameter C_FSL_LINKS
SFSL10	64	input	FSL Fifo Input port - Controlled by Parameter C_FSL_LINKS
SFSL11	64	input	FSL Fifo Input port - Controlled by Parameter C_FSL_LINKS
SFSL12	64	input	FSL Fifo Input port - Controlled by Parameter C_FSL_LINKS
SFSL13	64	input	FSL Fifo Input port - Controlled by Parameter C_FSL_LINKS
SFSL14	64	input	FSL Fifo Input port - Controlled by Parameter C_FSL_LINKS
SFSL15	64	input	FSL Fifo Input port - Controlled by Parameter C_FSL_LINKS
MFSL0	64	output	FSL Fifo Output port - Controlled by Parameter C_FSL_LINKS
MFSL1	64	output	FSL Fifo Output port - Controlled by Parameter C_FSL_LINKS
MFSL2	64	output	FSL Fifo Output port - Controlled by Parameter C_FSL_LINKS
MFSL3	64	output	FSL Fifo Output port - Controlled by Parameter C_FSL_LINKS
MFSL4	64	output	FSL Fifo Output port - Controlled by Parameter C_FSL_LINKS

MFSL5	64	output	FSL Fifo Output port - Controlled by Parameter C_FSL_LINKS
MFSL6	64	output	FSL Fifo Output port - Controlled by Parameter C_FSL_LINKS
MFSL7	64	output	FSL Fifo Output port - Controlled by Parameter C_FSL_LINKS
MFSL8	64	output	FSL Fifo Output port - Controlled by Parameter C_FSL_LINKS
MFSL9	64	output	FSL Fifo Output port - Controlled by Parameter C_FSL_LINKS
MFSL10	64	output	FSL Fifo Output port - Controlled by Parameter C_FSL_LINKS
MFSL11	64	output	FSL Fifo Output port - Controlled by Parameter C_FSL_LINKS
MFSL12	64	output	FSL Fifo Output port - Controlled by Parameter C_FSL_LINKS
MFSL13	64	output	FSL Fifo Output port - Controlled by Parameter C_FSL_LINKS
MFSL14	64	output	FSL Fifo Output port - Controlled by Parameter C_FSL_LINKS
MFSL15	64	output	FSL Fifo Output port - Controlled by Parameter C_FSL_LINKS

7.0 Parameters

Table 5.

Name	Type	Description
verbose	Boolean	Specify verbose output messages
C_FAMILY	Uns32	Target Family
C_AREA_OPTIMIZED	Uns32	Select implementation to optimize area with lower instruction throughput
C_INTERCONNECT	Uns32	Select interconnect 1 = PLBv46, 2 = AXI4
C_ENDIANNESS	Uns32	Select endianness 0 = Big endian, 1 = Little endian
C_FAULT_TOLERANT	Uns32	Implement fault tolerance
C_ECC_USE_CE_EXCEPTION	Uns32	Generate Bus Error Exceptions for correctable errors
C_PVR	Uns32	Processor version register mode selection
C_PVR_USER1	Uns32	Processor version register USER1 constant
C_PVR_USER2	Uns32	Processor version register USER2 constant
C_RESET_MSR	Enumeration	Reset value for MSR register 0x00=0 0x20=1 0x80=2 0xa0=3
C_D_PLB	Uns32	Data side PLB interface
C_D_AXI	Uns32	Data side AXI interface

C_D_LMB	Uns32	Data side LMB interface
C_I_PLB	Uns32	Instruction side PLB interface
C_I_AXI	Uns32	Instruction side AXI interface
C_I_LMB	Uns32	Instruction side LMB interface
C_USE_BARREL	Uns32	Include barrel shifter
C_USE_DIV	Uns32	Include hardware divider
C_USE_HW_MUL	Uns32	Include hardware integer multiplier
C_USE_FPU	Uns32	Include hardware floating integer point unit
C_USE_MSR_INSTR	Uns32	Enable use of instructions: integer MSRSET and MSRCLR
C_USE_PCMP_INSTR	Uns32	Enable use of instructions: integer CLZ, PCMPBF, PCMPEQ, and PCMPNE
C_UNALIGNED_EXCEPTIONS	Uns32	Enable exception handling for unaligned data accesses
C_ILL_OPCODE_EXCEPTION	Uns32	Enable exception handling for illegal op-code
C_IPLB_BUS_EXCEPTION	Uns32	Enable exception handling for IPLB bus error
C_DPLB_BUS_EXCEPTION	Uns32	Enable exception handling for DPLB bus error
C_M_AXI_I_BUS_EXCEPTION	Uns32	Enable exception handling for M_AXI_I bus error
C_M_AXI_D_BUS_EXCEPTION	Uns32	Enable exception handling for M_AXI_D bus error
C_DIV_ZERO_EXCEPTION	Uns32	Enable exception handling for division by zero or division overflow
C_FPU_EXCEPTION	Uns32	Enable exception handling for hardware floating point unit exceptions
C_OPCODE_0x0_ILLEGAL	Uns32	Detect opcode 0x0 as an illegal instruction
C_FSL_EXCEPTION	Uns32	Enable exception handling for Stream Links
C_USE_STACK_PROTECTION	Uns32	Generate exception for stack overflow or stack underflow
C_DEBUG_ENABLED	Uns32	MDM Debug interface
C_NUMBER_OF_PC_BRK	Uns32	Number of hardware breakpoints
C_NUMBER_OF_RD_ADDR_BRK	Uns32	Number of read address watchpoints
C_NUMBER_OF_WR_ADDR_BRK	Uns32	Number of write address watchpoints
C_INTERRUPT_IS_EDGE	Uns32	Level/Edge Interrupt
C_EDGE_IS_POSITIVE	Uns32	Negative/Positive Edge integer Interrupt
C_FSL_LINKS	Uns32	Number of stream interfaces (FSL or AXI)
C_USE_EXTENDED_FSL_INSTR	Uns32	Enable use of extended integer stream instructions
C_ICACHE_BASEADDR	Uns32	Instruction cache base address

C_ICACHE_HIGHADDR	Uns32	Instruction cache high address
C_USE_ICACHE	Uns32	Instruction cache
C_ALLOW_ICACHE_WR	Uns32	Instruction cache write enable
C_ICACHE_LINE_LEN	Enumeration	Instruction cache line length 4=0 8=1
C_ICACHE_ALWAYS_USED	Uns32	Instruction cache CacheLink used for all memory accesses
C_ICACHE_INTERFACE	Enumeration	Instruction cache CacheLink interface protocol IXCL=0 IXCL2=1
C_ICACHE_FORCE_TAG_LUTRAM	Uns32	Instruction cache tag always implemented with distributed RAM
C_ICACHE_STREAMS	Uns32	Instruction cache streams
C_ICACHE_VICTIMS	Enumeration	Instruction cache victims 0=0 2=1 4=2 8=3
C_ICACHE_DATA_WIDTH	Uns32	Instruction cache data width, 0 = 32 bits, 1 = Full cache line, 2 = 512 bits
C_ADDR_TAG_BITS	Uns32	Instruction cache address tags
C_CACHE_BYTE_SIZE	Enumeration	Instruction cache size 64=0 128=1 256=2 512=3 1024=4 2048=5 4096=6 8192=7 16384=8 32768=9 65536=10
C_ICACHE_USE_FSL	Uns32	Cache over CacheLink instead of peripheral bus for instructions
C_DCACHE_BASEADDR	Uns32	Data cache base address
C_DCACHE_HIGHADDR	Uns32	Data cache high address
C_USE_DCACHE	Uns32	Data cache
C_ALLOW_DCACHE_WR	Uns32	Data cache write enable
C_DCACHE_LINE_LEN	Enumeration	Data cache line length 4=0 8=1
C_DCACHE_ALWAYS_USED	Uns32	Data cache CacheLink used for all memory accesses
C_DCACHE_INTERFACE	Enumeration	Data cache CacheLink interface protocol DXCL=0 DXCL2=1
C_DCACHE_FORCE_TAG_LUTRAM	Uns32	Data cache tag always implemented with distributed RAM
C_DCACHE_USE_WRITEBACK	Uns32	Data cache write-back storage policy used
C_DCACHE_VICTIMS	Enumeration	Data cache victims 0=0 2=1 4=2 8=3
C_DCACHE_DATA_WIDTH	Uns32	Data cache data width, 0 = 32 bits, 1 = Full cache line, 2 = 512 bits
C_DCACHE_ADDR_TAG	Uns32	Data cache address tags
C_DCACHE_BYTE_SIZE	Enumeration	Data cache size 64=0 128=1 256=2 512=3 1024=4 2048=5 4096=6 8192=7 16384=8 32768=9 65536=10
C_DCACHE_USE_FSL	Uns32	Cache over CacheLink instead of peripheral bus for data
C_USE_MMU	Uns32	0 = None, 1 = Usermode, 2 = Protection, 3 = Virtual

C_MMU_DTLB_SIZE	Uns32	Data shadow Translation Look-Aside Buffer size 1,2,4,8
C_MMU_ITLB_SIZE	Uns32	Instruction shadow Translation Look-Aside Buffer size 1,2,4,8
C_MMU_TLB_ACCESS	Uns32	Access to memory management special registers: 0 = Minimal, 1 = Read, 2 = Write, 3 = Full
C_MMU_ZONES	Uns32	Number of memory protection zones
C_MMU_PRIVILEGED_INSTR	Uns32	Privileged instructions 0 = Full protection, 1 = Allow stream instrs
C_USE_INTERRUPT	Uns32	Enable interrupt handling
C_USE_EXT_BRK	Uns32	Enable external break handling
C_USE_EXT_NM_BRK	Uns32	Enable external non-maskable break handling
C_USE_BRANCH_TARGET_CACHE	Uns32	Enable Branch Target Cache
C_BRANCH_TARGET_CACHE_SIZE	Uns32	Branch Target Cache size: 0 = Default, 1 = 8 entries, 2 = 16 entries, 3 = 32 entries, 4 = 64 entries, 5 = 512 entries, 6 = 1024 entries, 7 = 2048 entries
C_STREAM_INTERCONNECT	Uns32	Select AXI4-Stream integer interconnect

8.0 Execution Modes

Table 6.

Name	Code	Description
REAL	0	Real mode
VIRTUAL_PRIV	1	Virtual privileged mode
VIRTUAL_USER	2	Virtual user mode

9.0 Exceptions

Table 7.

Name	Code
STREAM_EXCEPTION	0
UNALIGNED_DATA_ACCESS	1
ILLEGAL_OPCODE_EXCEPTION	2
INSTRUCTION_BUS_ERROR_EXCEPTION	3
DATA_BUS_ERROR_EXCEPTION	4
DIVIDE_EXCEPTION	5
FLOATING_POINT_UNIT_EXCEPTION	6
PRIVILEGED_INSTRUCTION_EXCEPTION	7
STACK_PROTECTION_VIOLATION_EXCEPTION	8
DATA_STORAGE_EXCEPTION	9

INSTRUCTION_STORAGE_EXCEPTION	10
DATA_TLB_MISS_EXCEPTION	11
INSTRUCTION_TLB_MISS_EXCEPTION	12
RESET	13
INTERRUPT	14

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 3 commands.

This level in the model hierarchy has 2 register groups:

Table 8.

Group name	Registers
User	32
System	25

This level in the model hierarchy has no children.

11.0 Model Commands

11.1 Level 1:

Table 9.

Name	Arguments
dumpTLB	- no arguments
isync	specify instruction address range for synchronous execution
itrace	enable or disable instruction tracing

12.0 Registers

12.1 Level 1:

12.1.1 User

Table 10.

Name	Bits	Initial value (Hex)		Description
R0	32	0	r-	
R1	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	
R24	32	0	rw	
R25	32	0	rw	
R26	32	0	rw	
R27	32	0	rw	
R28	32	0	rw	
R29	32	0	rw	
R30	32	0	rw	
R31	32	0	rw	

12.1.2 System

Table 11.

Name	Bits	Initial value (Hex)		Description
SPR_PC	32	0	rw	program counter
SPR_MSR	32	0	rw	
SPR_EAR	32	0	rw	
SPR_ESR	32	0	rw	
SPR_FSR	32	0	rw	
SPR_BTR	32	0	rw	
SPR_PVR0	32	10001500	rw	
SPR_PVR1	32	0	rw	
SPR_PVR2	32	54831000	rw	
SPR_PVR3	32	2000000	rw	
SPR_PVR4	32	45000000	rw	
SPR_PVR5	32	47000000	rw	
SPR_PVR6	32	0	rw	
SPR_PVR7	32	3fffffff	rw	
SPR_PVR8	32	0	rw	
SPR_PVR9	32	3fffffff	rw	
SPR_PVR10	32	0	rw	
SPR_PVR11	32	ae00000	rw	
SPR_EDR	32	0	rw	
SPR_PID	32	0	rw	
SPR_ZPR	32	0	rw	
SPR_TLBX	32	0	rw	
SPR_TLBSX	32	0	rw	
SPR_TLBLO	32	0	rw	
SPR_TLBHI	32	0	rw	

#