OVP Guide to Using Processor Models

Model Specific Information for variant Synopsys_ARC_605

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_arc_605.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

ARC 600 processor model (ARCv1 architecture)

1.2 Licensing

Usage of binary model under license governing simulator usage. Source of model available under Imperas Software License Agreement.

1.3 Limitations

Instruction pipelines are not modeled in any way. All instructions are assumed to complete immediately.
Instruction and data caches are not modeled, except for the auxiliary register interface.
External host debug is not modeled, except for the auxiliary register interface.
Real-world timing effects are not modeled. All instructions are assumed to complete in a single cycle.
User extensions are not yet implemented, except for extension core registers.

1.4 Verification

Models have been validated correct in a cooperative project between Imperas and ARC

1.5 Reference

ARC Processor ARC6xx/ARC7xx Reference Documentation

1.6 Debugging

The model has been designed for debug using GNU gdb ARCompact/ARCv2 ISA elf32 version 7.5.1. To ensure correct behavior, enter the following command into gdb before attempting to connect to the processor:
set architecture ARC600
Failure to do this may cause the debugging session to fail because of g-packet size mismatch.

1.7 Features

The model implements the full ARCv1 instruction set.
The model can be configured with either a 16-entry or 32-entry register file using parameter opt-rf16.
The exact set of core instructions present can be configured by a number of parameters: see information for opt-swap, opt-bitscan, opt-extended-arith and opt-multiply in the table below.
Parameter opt-extension-interrupts can be used to enable extension interrupts 16-31.
Timer 0 and Timer 1 can be enabled using parameters opt-timer0 and opt-timer1, respectively.
The sizes of DCCM, ICCM0 and ICCM1 can be specified using parameters opt-dccm-size, opt-iccm0-size and opt-iccm1-size, respectively. Reset base addresses for the ICCMs can be specified using opt-iccm0-base and opt-iccm1-base. Note that the DCCM reset base address is architecturally defined (0x80000000) and not configurable. When CCMs are present, bus ports called DCCM0, ICCM0 and ICCM1 are created so that CCM contents may be viewed or modified externally by connecting to these ports. Parameter opt-internal-ccms specifies whether CCM memory is modeled internally or externally. If modeled externally, the CCMs must be implemented on a bus which is then connected to the CCM bus ports listed above (this parameter is ignored if CCM ports are unconnected; in that case, CCMs are always modeled internally). Parameter opt-reset-internal-ccms indicates that internally-modeled CCMs should be cleared to zero on a processor reset; if False, then internally-modeled CCMs retain their previous state after a reset.
The set of core registers can be specified using parameter opt-extension-core-regs. This is a 64-bit value in which a 1-bit implies the presence of that core extension register. For example, a value of 0xf00000000ULL implies that extension registers r32-r35 should be configured.
The reset value of the exception vector base register can be specified using parameter opt-intvbase-preset.

2.0 Configuration

2.1 Location

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

2.2 GDB Path

The default GDB for this model is found at:
$IMPERAS_HOME/lib/$IMPERAS_ARCH/gdb/arc-elf32-gdb

2.3 Semi-Host Library

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

The ELF code supported by this model is: 0x5d

3.0 Other Variants in this Model

Table 1.

Variant
600
605
700
0x21
0x22
0x31
0x32

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	Description
reset	input	Processor reset
watchdog	output	Watchdog timer
irq4	input	External interrupt
irq5	input	External interrupt
irq6	input	External interrupt
irq8	input	External interrupt
irq9	input	External interrupt
irq10	input	External interrupt
irq11	input	External interrupt
irq12	input	External interrupt
irq13	input	External interrupt
irq14	input	External interrupt
irq15	input	External interrupt

6.0 FIFO Ports

No FIFO Ports in this model.

7.0 Parameters

Table 4.

Name	Type	Description
verbose	Boolean	Enable verbose messages
end-on-halt	Boolean	Specify whether to end simulation when halt bit set in STATUS/STATUS32
dump-bcrs	Boolean	Add BCRs to register trace
format	Enumeration	Select register format gdb=0 metaware=1
compatibility	Enumeration	Select compatibility mode ISA=0 metaware8.2=1
opt-identity	Uns32	Override value of IDENTITY register
opt-intvbase-preset	Uns32	Specify reset vectore base register x 1024 (VECBASE_AC_BUILD.Addr)
opt-rf16	Uns32	Specify 16-entry core register file (RF_BUILD.E)
opt-swap	Uns32	Specify swap instructions version (SWAP_BUILD.Version)
opt-bitscan	Uns32	Specify bitscan instructions version (NORM_BUILD.Version)
opt-extended-arith	Uns32	Specify extended arithmetic version (EA_BUILD.Version)
opt-multiply	Uns32	Specify multiply instructions version (MULTIPLY_BUILD.Version)
opt-extension-interrupts	Uns32	Enable extension interrupts 16-31
opt-timer0	Uns32	Timer 0 present (TIMER_BUILD.T0)
opt-timer1	Uns32	Timer 1 present (TIMER_BUILD.T1)
opt-dccm-size	Uns32	Specify DCCM RAM size (DCCM_BUILD.Size)
opt-iccm0-size	Uns32	Specify ICCM0 RAM size (ICCM_BUILD.ICCM0_SIZE)
opt-iccm1-size	Uns32	Specify ICCM1 RAM size (ICCM_BUILD.ICCM1_SIZE)
opt-iccm0-base	Uns32	Specify ICCM0 RAM base address at reset
opt-iccm1-base	Uns32	Specify ICCM1 RAM base address at reset
opt-internal-ccms	Boolean	Specify that configured CCMs should be modeled internally
opt-reset-internal-ccms	Boolean	Specify that internally-modeled configured CCMs should be zeroed at reset
opt-extension-core-regs	Uns64	Bitmask specifying extension core registers

8.0 Execution Modes

Table 5.

Name	Code	Description
Kernel	0	Kernel mode

9.0 Exceptions

Table 6.

Name	Code
Reset	0
IllegalInstruction	2
MisalignedDataAccess	28

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 3 register groups:

Table 7.

Group name	Registers
core.arcompact	34
aux-minimal	20
BCR	28

This level in the model hierarchy has no children.

11.0 Model Commands

11.1 Level 1:

Table 8.

Name	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 core.arcompact

Table 9.

Name	Bits	Initial value (Hex)		Description
R0	32	0	rw	
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	
GP	32	0	rw	
FP	32	0	rw	frame pointer
SP	32	4000	rw	stack pointer
ILINK1	32	0	rw	
ILINK2	32	0	rw	
BLINK	32	0	rw	
LP_COUNT	32	0	r-	
PCL	32	0	r-	

12.1.2 aux-minimal

Table 10.

Name	Bits	Initial value (Hex)		Description
STATUS	32	0	r-	0x000: Status (Obsolete)
SEMAPHORE	32	0	rw	0x001: Semaphore
LP_START	32	0	rw	0x002: Loop Start
LP_END	32	0	rw	0x003: Loop End
IDENTITY	32	123	r-	0x004: Identity
DEBUG	32	0	rw	0x005: Debug
PC	32	0	rw	0x006: Program Counter
STATUS32	32	0	r-	0x00a: 32-bit Status
STATUS32_L1	32	0	rw	0x00b: L1 Interrupt Status
STATUS32_L2	32	0	rw	0x00c: P0 Interrupt Status
COUNT0	32	0	rw	0x021: Timer 0 Count Value
CONTROL0	32	0	rw	0x022: Timer 0 Control
LIMIT0	32	ffffff	rw	0x023: Timer 0 Limit
INT_VECTOR_BASE	32	0	rw	0x025: Interrupt Vector Base
AUX_IRQ_LV12	32	0	rw	0x043: L1/L2 Interrupt Level
COUNT1	32	0	rw	0x100: Timer 1 Count Value
CONTROL1	32	0	rw	0x101: Timer 1 Control
LIMIT1	32	ffffff	rw	0x102: Timer 1 Limit
AUX_IRQ_LEV	32	c0	rw	0x200: Interrupt Level Programming
AUX_IRQ_HINT	32	0	rw	0x201: Software Interrupt Trigger

12.1.3 BCR

Table 11.

Name	Bits	Initial value (Hex)		Description

BCR_VER	32	2	r-	0x060: Configuration Register Version
BCR_DCCM_BASE	32	0	r-	0x061: DCCM Base Address
BCR_CRC	32	0	r-	0x062: CRC Configuration
BCR_VBFDW	32	0	r-	0x064: VBFDW Configuration
BCR_EA_BUILD	32	0	r-	0x065: EA Configuration
BCR_DATASPACE	32	0	r-	0x066: DataSpace Configuration
BCR_MEMSUBSYS	32	1	r-	0x067: Memory Subsystem Configuration
BCR_VECBASE_AC_BUILD	32	0	r-	0x068: Interrupt Vector Base Address Configuration
BCR_PBASEADDR	32	0	r-	0x069: PBASE Configuration
BCR_MPU_BUILD	32	0	r-	0x06d: MPU Configuration
BCR_RF_BUILD	32	1	r-	0x06e: Core Register Set Configuration
BCR_VECBASE_BUILD	32	0	r-	0x071: VECBASE Configuration
BCR_DCACHE_BUILD	32	0	r-	0x072: Data Cache Configuration
BCR_MADI	32	0	r-	0x073: MADI Configuration
BCR_LDSTRAM	32	701	r-	0x074: LDSTRAM Configuration
BCR_TIMER_BUILD	32	303	r-	0x075: Timer Configuration
BCR_AP_BUILD	32	0	r-	0x076: Actionpoints Configuration
BCR_ICACHE_BUILD	32	0	r-	0x077: Instruction Cache Configuration
BCR_ICCM_BUILD	32	1	r-	0x078: ICCM RAM Configuration
BCR_DSPRAM	32	0	r-	0x079: SPRAM Configuration
BCR_MAC_BUILD	32	0	r-	0x07a: MAC Configuration
BCR_MULTIPLY_BUILD	32	0	r-	0x07b: Multiply Configuration
BCR_SWAP_BUILD	32	0	r-	0x07c: Swap Configuration
BCR_NORM_BUILD	32	0	r-	0x07d: Normalize Configuration
BCR_MINMAX_BUILD	32	0	r-	0x07e: Min/Max Configuration
BCR_BARREL_BUILD	32	2	r-	0x07f: Barrel Shifter Configuration
BCR_PMU	32	0	r-	0x0f7: PMU Configuration
BCR_IFETCHQUEUE	32	0	r-	0x0fe: Instruction Fetch Queue Configuration

#