OVP Guide to Using Processor Models

Model specific information for ARM_ARMv6-M

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

Author	Imperas Software Limited
Version	20210408.0
Filename	OVP_Model_Specific_Information_armm_ARMv6-M.pdf
Created	5 May 2021
Status	OVP Standard Release

Copyright Notice

Right to Copy Documentation

Destination Control Statement

Disclaimer

Model Release Status

This model is released as part of OVP releases and is included in OVPworld packages. Please visit OVPworld.org.

Contents

Chapter 1

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

ARMM Processor Model

1.2 Licensing

Usage of binary model under license governing simulator usage.

Note that for models of ARM CPUs the license includes the following terms:

Licensee is granted a non-exclusive, worldwide, non-transferable, revocable licence to:

If no source is being provided to the Licensee: use and copy only (no modifications rights are granted) the model for the sole purpose of designing, developing, analyzing, debugging, testing, verifying, validating and optimizing software which: (a) (i) is for ARM based systems; and (ii) does not incorporate the ARM Models or any part thereof; and (b) such ARM Models may not be used

to emulate an ARM based system to run application software in a production or live environment.

If source code is being provided to the Licensee: use, copy and modify the model for the sole purpose of designing, developing, analyzing, debugging, testing, verifying, validating and optimizing software which: (a) (i) is for ARM based systems; and (ii) does not incorporate the ARM Models or any part thereof; and (b) such ARM Models may not be used to emulate an ARM based system to run application software in a production or live environment.

In the case of any Licensee who is either or both an academic or educational institution the purposes shall be limited to internal use.

Except to the extent that such activity is permitted by applicable law, Licensee shall not reverse engineer, decompile, or disassemble this model. If this model was provided to Licensee in Europe, Licensee shall not reverse engineer, decompile or disassemble the Model for the purposes of error correction.

The License agreement does not entitle Licensee to manufacture in silicon any product based on this model.

The License agreement does not entitle Licensee to use this model for evaluating the validity of any ARM patent.

The License agreement does not entitle Licensee to use the model to emulate an ARM based system to run application software in a production or live environment.

Source of model available under separate Imperas Software License Agreement.

1.3 Limitations

Performance Monitors are not implemented.

Debug Extension and related blocks are not implemented.

1.4 Verification

Models have been extensively tested by Imperas. ARM Cortex-M models have been successfully used by customers to simulate the Micrium uC/OS-II kernel and FreeRTOS.

1.5 Features

The model is configured with 16 interrupts and 2 priority bits (use override_numInterrupts parameter to change the number of interrupts; the number of priority bits is fixed in this profile).

MPU is not present. Use parameter override_MPU_TYPE to enable it if required.

SysTick timer is present. Use parameter SysTickPresent to disable it if required.

Unprivileged/Privileged Extension is not present. Use parameter unprivilegedExtension to enable it if required.

VTOR register is not present. Use parameter VTORPresent to enable it if required.

1.6 Unpredictable Behavior

Many instruction behaviors are described in the ARM ARM as CONSTRAINED UNPREDICTABLE. This section describes how such situations are handled by this model.

1.6.1 Equal Target Registers

Some instructions allow the specification of two target registers (for example, double-width SMULL, or some VMOV variants), and such instructions are CONSTRAINED UNPREDICTABLE if the same target register is specified in both positions. In this model, such instructions are treated as UNDEFINED.

1.6.2 Floating Point Load/Store Multiple Lists

Instructions that load or store a list of floating point registers (e.g. VSTM, VLDM, VPUSH, VPOP) are CONSTRAINED UNPREDICTABLE if either the uppermost register in the specified range is greater than 32 or (for 64-bit registers) if more than 16 registers are specified. In this model, such instructions are treated as UNDEFINED.

1.6.3 If-Then (IT) Block Constraints

Where the behavior of an instruction in an if-then (IT) block is described as CONSTRAINED UNPREDICTABLE, this model treats that instruction as UNDEFINED.

1.6.4 Use of R13

Use of R13 is described as CONSTRAINED UNPREDICTABLE in many circumstances. This model allows R13 to be used like any other GPR.

1.6.5 Use of R15

Use of R15 is described as CONSTRAINED UNPREDICTABLE in many circumstances. This model allows such use to be configured using the parameter “unpredictableR15” as follows:

Value “undefined”: any reference to R15 in such a situation is treated as UNDEFINED;

Value “nop”: any reference to R15 in such a situation causes the instruction to be treated as a NOP;

Value “raz_wi”: any reference to R15 in such a situation causes the instruction to be treated as a RAZ/WI (that is, R15 is read as zero and write-ignored);

Value “execute”: any reference to R15 in such a situation is executed using the current value of R15 on read, and writes to R15 are allowed.

Value “assert”: any reference to R15 in such a situation causes the simulation to halt with an assertion message (allowing any such unpredictable uses to be easily identified).

In this variant, the default value of “unpredictableR15” is “execute”.

Chapter 2

Configuration

2.1 Location

This model's VLNV is arm.ovpworld.org/processor/armm/1.0.
The model source is usually at:
$IMPERAS_HOME/ImperasLib/source/arm.ovpworld.org/processor/armm/1.0
The model binary is usually at:
$IMPERAS_HOME/lib/$IMPERAS_ARCH/ImperasLib/arm.ovpworld.org/processor/armm/1.0

2.2 GDB Path

The default GDB for this model is: $IMPERAS_HOME/lib/$IMPERAS_ARCH/gdb/arm-none-eabi-gdb.

2.3 Semi-Host Library

The default semi-host library file is arm.ovpworld.org/semihosting/armNewlib/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: 0x28.

Chapter 3

All Variants in this model

This model has these variants

Variant	Description
ARMv6-M	(described in this document)
ARMv7-M	
Cortex-M0	
Cortex-M0plus	
Cortex-M1	
Cortex-M3	
Cortex-M4	
Cortex-M4F	
Cortex-M7	
Cortex-M7F	
Cortex-M23	
Cortex-M33	
Cortex-M33F	

Table 3.1: All Variants in this model

Chapter 4

Bus Master Ports

This model has these bus master ports.

Name	min	max	Connect?	Description
INSTRUCTION	32	32	mandatory	
DATA	32	32	optional	

Table 4.1: Bus Master Ports

Chapter 5

Bus Slave Ports

This model has no bus slave ports.

Chapter 6

Net Ports

This model has these net ports.

Name	Type	Connect?	Description
sysResetReq	output	optional	
intISS	output	optional	
eventOut	output	optional	
lockup	output	optional	
int	input	optional	
reset	input	optional	
nmi	input	optional	
eventIn	input	optional	
int0	input	optional	
int1	input	optional	
int2	input	optional	
int3	input	optional	
int4	input	optional	
int5	input	optional	
int6	input	optional	
int7	input	optional	
int8	input	optional	
int9	input	optional	
int10	input	optional	
int11	input	optional	
int12	input	optional	
int13	input	optional	
int14	input	optional	
int15	input	optional	

Table 6.1: Net Ports

Chapter 7

FIFO Ports

This model has no FIFO ports.

Chapter 8

Formal Parameters

Name	Type	Description
variant	Enumeration	Selects variant (either a generic ISA or a specific model)
verbose	Boolean	Specify verbosity of output
showHiddenRegs	Boolean	Show hidden registers during register tracing
UAL	Boolean	Disassemble using UAL syntax
compatibility	Enumeration	Specify compatibility mode (ISA, gdb or nopBKPT)
unpredictableR15	Enumeration	Specify behavior for UNPREDICTABLE uses of R15 (undefined, nop, raz_wi, execute or assert)
override_debugMask	Uns32	Specifies debug mask, enabling debug output for model components
endian	Endian	Model endian
instructionEndian	Endian	The architecture specifies that instruction fetch is always little endian; this attribute allows the defined instruction endianness to be overridden if required
resetAtTime0	Boolean	Reset the model at time=0 (default=1)
unprivilegedExtension	Boolean	Specify presence of Unprivileged/Privileged Extension
VTORPresent	Boolean	Specify presence of VTOR register
SysTickPresent	Uns32	Specify number of SysTick timers present
override_CPUID	Uns32	Override system CPUID register
override_MPU_TYPE	Uns32	Override system MPU_TYPE register
override_VTOR	Uns32	Override VTOR register reset value
override_deviceStrongAligned	Boolean	Force accesses to Device and Strongly Ordered regions to be aligned
override_STRoffsetPC12	Uns32	Specifies that STR/STR of PC should do so with 12:byte offset from the current instruction (if 1), otherwise an 8:byte offset is used
override_ERG	Uns32	Specifies exclusive reservation granule
override_numInterrupts	Uns32	Specifies number of external interrupt lines

Table 8.1: Parameters

Chapter 9

Execution Modes

Mode	Code
Thread	0
Handler	1

Table 9.1: Modes implemented in this processor

Chapter 10

Exceptions

Exception	Code
None	0
Reset	1
NMI	2
HardFault	3
SVCall	11
PendSV	14
SysTick	15
ExternalInt000	16
ExternalInt001	17
ExternalInt002	18
ExternalInt003	19
ExternalInt004	20
ExternalInt005	21
ExternalInt006	22
ExternalInt007	23
ExternalInt008	24
ExternalInt009	25
ExternalInt00a	26
ExternalInt00b	27
ExternalInt00c	28
ExternalInt00d	29
ExternalInt00e	30
ExternalInt00f	31

Table 10.1: Exceptions implemented by this processor

Chapter 11

Hierarchy of the model

A CPU core may be configured 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.

11.1 Level 1

This level in the model hierarchy has 3 commands.
This level in the model hierarchy has 4 register groups:

Group name	Registers
Core	16
Control	5
System	21
Integration_support	2

Table 11.1: Register groups

This level in the model hierarchy has no children.

Chapter 12

Model Commands

A Processor model can implement one or more **Model Commands** available to be invoked from the simulator command line, from the OP API or from the Imperas Multiprocessor Debugger.

12.1 Level 1

12.1.1 debugflags

show or modify the processor debug flags

Argument	Type	Description
-get	Boolean	print current processor flags value
-mask	Boolean	print valid debug flag bits
-set	Int32	new processor flags (only flags 0x0000008c can be modified)

Table 12.1: debugflags command arguments

12.1.2 isync

specify instruction address range for synchronous execution

Argument	Type	Description
-addresshi	Uns64	end address of synchronous execution range
-addresslo	Uns64	start address of synchronous execution range

Table 12.2: isync command arguments

12.1.3 itrace

enable or disable instruction tracing

Argument	Type	Description
-after	Uns64	apply after this many instructions
-enable	Boolean	enable instruction tracing
-instructioncount	Boolean	include the instruction number in each trace
-off	Boolean	disable instruction tracing

-on	Boolean	enable instruction tracing
-registerchange	Boolean	show registers changed by this instruction
-registers	Boolean	show registers after each trace

Table 12.3: itrace command arguments

Chapter 13

Registers

13.1 Level 1

13.1.1 Core

Registers at level:1, group:Core

Name	Bits	Initial-Hex	RW	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	frame pointer
r12	32	0	rw	
sp	32	0	rw	stack pointer
lr	32	0	rw	
pc	32	0	rw	program counter

Table 13.1: Registers at level 1, group:Core

13.1.2 Control

Registers at level:1, group:Control

Name	Bits	Initial-Hex	RW	Description
cpsr	32	0	rw	xPSR register. Includes APSR, IPSR and EPSR
control	32	0	rw	
primask	32	0	rw	
sp_process	32	0	rw	stack pointer
sp_main	32	0	rw	stack pointer

Table 13.2: Registers at level 1, group:Control

13.1.3 System

Registers at level:1, group:System

Name	Bits	Initial-Hex	RW	Description
ACTLR	32	0	rw	0xe000e008: Auxiliary Control
SYST_CSR	32	0	rw	0xe000e010: SysTick Control and Status
SYST_RVR	32	0	rw	0xe000e014: SysTick Reload Value
SYST_CVR	32	0	rw	0xe000e018: SysTick Current Value
SYST_CALIB	32	0	rw	0xe000e01c: SysTick Calibration Value
NVIC_ISER0	32	0	rw	0xe000e100: Interrupt Set Enable 0
NVIC_ICER0	32	0	rw	0xe000e180: Interrupt Clear Enable 0
NVIC_ISPR0	32	0	rw	0xe000e200: Interrupt Set Pending 0
NVIC_ICPR0	32	0	rw	0xe000e280: Interrupt Clear Pending 0
NVIC_IPR0	32	0	rw	0xe000e400: Interrupt Priority 0
NVIC_IPR1	32	0	rw	0xe000e404: Interrupt Priority 1
NVIC_IPR2	32	0	rw	0xe000e408: Interrupt Priority 2
NVIC_IPR3	32	0	rw	0xe000e40c: Interrupt Priority 3
CPUID	32	c0000	r-	0xe000ed00: CPUID Base
ICSR	32	1000	rw	0xe000ed04: Interrupt Control and State
AIRCR	32	fa050000	rw	0xe000ed0c: Application Interrupt and Reset Control
SCR	32	0	rw	0xe000ed10: System Control
CCR	32	208	rw	0xe000ed14: Configuration and Control
SHPR2	32	0	rw	0xe000ed1c: System Handler Priority 2
SHPR3	32	0	rw	0xe000ed20: System Handler Priority 3
SHCSR	32	0	rw	0xe000ed24: System Handler Control and State

Table 13.3: Registers at level 1, group:System

13.1.4 Integration_support

Registers at level:1, group:Integration_support

Name	Bits	Initial-Hex	RW	Description
executionPri	32	7fffffff	r-	current execution priority level
stackDomain	64	959b60	r-	stack domain for current execution level

Table 13.4: Registers at level 1, group:Integration_support