OVP Guide to Using Processor Models

Model specific information for ARM_Cortex-A8

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

Author	Imperas Software Limited
Version	20211118.0
Filename	OVP_Model_Specific_Information_arm_Cortex-A8.pdf
Created	31 December 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

ARM 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.

Source of model available under separate Imperas Software License Agreement.

1.3 Limitations

Instruction pipelines are not modeled in any way. All instructions are assumed to complete immediately. This means that instruction barrier instructions (e.g. ISB, CP15ISB) are treated as NOPs, with the exception of any undefined instruction behavior, which is modeled. The model does not implement speculative fetch behavior. The branch cache is not modeled.

Caches and write buffers are not modeled in any way. All loads, fetches and stores complete immediately and in order, and are fully synchronous (as if the memory was of Strongly Ordered or Device-nGnRnE type). Data barrier instructions (e.g. DSB, CP15DSB) are treated as NOPs, with the exception of any undefined instruction behavior, which is modeled. Cache manipulation instructions are implemented as NOPs, with the exception of any undefined instruction behavior, which is modeled.

Real-world timing effects are not modeled: all instructions are assumed to complete in a single cycle.

Performance Monitors are implemented as a register interface only except for the cycle counter, which is implemented assuming one instruction per cycle.

TLBs are architecturally-accurate but not device accurate. This means that all TLB maintenance and address translation operations are fully implemented but the cache is larger than in the real device.

1.4 Verification

Models have been extensively tested by Imperas. ARM Cortex-A models have been successfully used by customers to simulate SMP Linux, Ubuntu Desktop, VxWorks and ThreadX on Xilinx Zynq virtual platforms.

1.5 Features

1.5.1 Core Features

Thumb-2 instructions are supported.

Trivial Jazelle extension is implemented.

1.5.2 Memory System

FCSE extension is implemented.

Security extensions are implemented (also known as TrustZone). Non-secure accesses can be made visible externally by connecting the processor to a 41-bit physical bus, in which case bits 39..0 give the true physical address and bit 40 is the NS bit.

VMSA secure and non-secure address translation is implemented.

TLB behavior is controlled by parameter ASIDCacheSize. If this parameter is 0, then an unlimited number of TLB entries will be maintained concurrently. If this parameter is non-zero, then only TLB entries for up to ASIDCacheSize different ASIDs will be maintained concurrently initially; as new ASIDs are used, TLB entries for less-recently used ASIDs are deleted, which improves model performance in some cases (especially when 16-bit ASIDs are in use). If the model detects that the TLB entry cache is too small (entry ejections are very frequent), it will increase the cache size automatically. In this variant, ASIDCacheSize is 8

1.5.3 Advanced SIMD and Floating-Point Features

SIMD and VFP instructions are implemented.

The model implements trapped exceptions if FPTrap is set to 1 in MVFR0 (for AArch32) or MVFR0_EL1 (for AArch64). When floating point exception traps are taken, cumulative exception flags are not updated (in other words, cumulative flag state is always the same as prior to instruction execution, even for SIMD instructions). When multiple enabled exceptions are raised by a single floating point operation, the exception reported is the one in least-significant bit position in FPSCR (for AArch32) or FPCR (for AArch64). When multiple enabled exceptions are raised by different SIMD element computations, the exception reported is selected from the lowest-index-number SIMD operation. Contact Imperas if requirements for exception reporting differ from these.

Trapped exceptions not are implemented in this variant (FPTrap=0)

1.6 Debug Mask

It is possible to enable model debug features in various categories. This can be done statically using the "override_debugMask" parameter, or dynamically using the "debugflags" command. Enabled debug features are specified using a bitmask value, as follows:

Value 0x004: enable debugging of MMU/MPU mappings.

Value 0x080: enable debugging of all system register accesses.

Value 0x100: enable debugging of all traps of system register accesses.

Value 0x200: enable verbose debugging of other miscellaneous behavior (for example, the reason why a particular instruction is undefined).

Value 0x400: enable debugging of Performance Monitor timers

Value 0x800: enable dynamic validation of TLB entries against in-memory page table contents (finds some classes of error where page table entries are updated without a subsequent flush of affected TLB entries).

All other bits in the debug bitmask are reserved and must not be set to non-zero values.

1.7 AArch32 Unpredictable Behavior

Many AArch32 instruction behaviors are described in the ARM ARM as CONSTRAINED UNPREDICTABLE. This section describes how such situations are handled by this model.

1.7.1 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.

1.7.2 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.

1.7.3 Floating Point VLD[2-4]/VST[2-4] Range Overflow

Instructions that load or store a fixed number of floating point registers (e.g. VST2, VLD2) are CONSTRAINED UNPREDICTABLE if the upper register bound exceeds the number of implemented floating point registers. In this model, these instructions load and store using modulo 32

indexing (consistent with AArch64 instructions with similar behavior).

1.7.4 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.

1.7.5 Use of R13

In architecture variants before ARMv8, use of R13 was described as CONSTRAINED UNPREDICTABLE in many circumstances. From ARMv8, most of these situations are no longer considered unpredictable. This model allows R13 to be used like any other GPR, consistent with the ARMv8 specification.

1.7.6 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 (but are not interworking).

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 "undefined".

1.7.7 Unpredictable Instructions in Some Modes

Some instructions are described as CONSTRAINED UNPREDICTABLE in some modes only (for example, MSR accessing SPSR is CONSTRAINED UNPREDICTABLE in User and System modes). This model allows such use to be configured using the parameter "unpredictableModal", which can have values "undefined" or "nop". See the previous section for more information about the meaning of these values.

In this variant, the default value of "unpredictableModal" is "nop".

1.8 Integration Support

This model implements a number of non-architectural pseudo-registers and other features to facilitate integration.

1.8.1 Memory Transaction Query

Two registers are intended for use within memory callback functions to provide additional information about the current memory access. Register transactPL indicates the processor execution level of the current access (0-3). Note that for load/store translate instructions (e.g. LDRT, STRT) the reported execution level will be 0, indicating an EL0 access. Register transactAT indicates the type of memory access: 0 for a normal read or write; and 1 for a physical access resulting from a page table walk.

1.8.2 Page Table Walk Query

A banked set of registers provides information about the most recently completed page table walk. There are up to six banks of registers: bank 0 is for stage 1 walks, bank 1 is for stage 2 walks, and banks 2-5 are for stage 2 walks initiated by stage 1 level 0-3 entry lookups, respectively. Banks 1-5 are present only for processors with virtualization extensions. The currently active bank can be set using register PTWBankSelect. Register PTWBankValid is a bitmask indicating which banks contain valid data: for example, the value 0xb indicates that banks 0, 1 and 3 contain valid data.

Within each bank, there are registers that record addresses and values read during that page table walk. Register PTWBase records the table base address, register PTWInput contains the input address that starts a walk, register PTWOutput contains the result address and register PTWPgSize contains the page size (PTWOutput and PTWPgSize are valid only if the page table walk completes). Registers PTWAddressL0-PTWAddressL3 record the addresses of level 0 to level 3 entries read, respectively. Register PTWAddressValid is a bitmask indicating which address registers contain valid data: bits 0-3 indicate PTWAddressL0-PTWAddressL3, respectively, bit 4 indicates PTWBase, bit 5 indicates PTWInput, bit 6 indicates both PTWOutput and PTWPgSize. For example, the value 0x73 indicates that PTWBase, PTWInput, PTWOutput, PTWPgSize and PTWAddressL0-L1 are valid but PTWAddressL2-L3 are not. Register PTWAddressNS is a bitmask indicating whether an address is in non-secure memory: bits 0-3 indicate PTWAddressL0-PTWAddressL3, respectively, bit 4 indicates PTWBase, bit 6 indicates PTWOutput (PTWInput is a VA and thus has no secure/non-secure info). Registers PTWValueL0-PTWValueL3 contain page table entry values read at level 0 to level 3. Register PTWValueValid is a bitmask indicating which value registers contain valid data: bits 0-3 indicate PTWValueL0-PTWValueL3, respectively.

1.8.3 Artifact Page Table Walks

Registers are also available to enable a simulation environment to initiate an artifact page table walk (for example, to determine the ultimate PA corresponding to a given VA). Register PTWI_EL1S initiates a secure EL1 table walk for a fetch. Register PTWD_EL1S initiates a secure EL1 table walk for a load or store (note that current ARM processors have unified TLBs, so these registers

are synonymous). Registers PTW[ID]_EL1NS initiate walks for non-secure EL1 accesses. Registers PTW[ID]_EL2 initiate EL2 walks. Registers PTW[ID]_S2 initiate stage 2 walks. Registers PTW[ID]_EL3 initiate AArch64 EL3 walks. Finally, registers PTW[ID]_current initiate current-mode walks (useful in a memory callback context). Each walk fills the query registers described above.

1.8.4 MMU and Page Table Walk Events

Two events are available that allow a simulation environment to be notified on MMU and page table walk actions. Event mmuEnable triggers when any MMU is enabled or disabled. Event pageTableWalk triggers on completion of any page table walk (including artifact walks).

1.8.5 Artifact Address Translations

A simulation environment can trigger an artifact address translation operation by writing to the architectural address translation registers (e.g. ATS1CPR). The results of such translations are written to an integration support register artifactPAR, instead of the architectural PAR register. This means that such artifact writes will not perturb architectural state.

1.8.6 TLB Invalidation

A simulation environment can cause TLB state for one or more address translation regimes in the processor to be flushed by writing to the artifact register ResetTLBs. The argument is a bitmask value, in which non-zero bits select the TLBs to be flushed, as follows:

Bit 0: EL0/EL1 stage 1 secure TLB

Bit 1: EL0/EL1 stage 1 non-secure TLB

1.8.7 Halt Reason Introspection

An artifact register HaltReason can be read to determine the reason or reasons that a processor is halted. This register is a bitfield, with the following encoding: bit 0 indicates the processor has executed a wait-for-event (WFE) instruction; bit 1 indicates the processor has executed a wait-for-interrupt (WFI) instruction; and bit 2 indicates the processor is held in reset.

1.8.8 System Register Access Monitor

If parameter "enableSystemMonitorBus" is True, an artifact 32-bit bus "SystemMonitor" is enabled for each PE. Every system register read or write by that PE is then visible as a read or write on this artifact bus, and can therefore be monitored using callbacks installed in the client environment (use opBusReadMonitorAdd/opBusWriteMonitorAdd or icmAddBusReadCallback-/icmAddBusWriteCallback, depending on the client API). The format of the address on the bus is as follows:

bits 31:26 - zero

bit 25 - 1 if AArch64 access, 0 if AArch32 access

bit 24 - 1 if non-secure access, 0 if secure access

bits 23:20 - CRm value

bits 19:16 - CRn value

bits 15:12 - op2 value

bits 11:8 - op1 value

bits 7:4 - op0 value (AArch64) or coprocessor number (AArch32)

bits 3:0 - zero

As an example, to view non-secure writes to writes to CNTFRQ_EL0 in AArch64 state, install a write monitor on address range 0x020e0330:0x020e0333.

1.8.9 System Register Implementation

If parameter "enableSystemBus" is True, an artifact 32-bit bus "System" is enabled for each PE. Slave callbacks installed on this bus can be used to implement modified system register behavior (use opBusSlaveNew or icmMapExternalMemory, depending on the client API). The format of the address on the bus is the same as for the system monitor bus, described above.

Chapter 2

Configuration

2.1 Location

This model's VLNV is arm.ovpworld.org/processor/arm/1.0.
The model source is usually at:
$IMPERAS_HOME/ImperasLib/source/arm.ovpworld.org/processor/arm/1.0
The model binary is usually at:
$IMPERAS_HOME/lib/$IMPERAS_ARCH/ImperasLib/arm.ovpworld.org/processor/arm/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
ARMv4T	
ARMv4xM	
ARMv4	
ARMv4TxM	
ARMv5xM	
ARMv5	
ARMv5TxM	
ARMv5T	
ARMv5TExP	
ARMv5TE	
ARMv5TEJ	
ARMv6	
ARMv6K	
ARMv6T2	
ARMv6KZ	
ARMv7	
ARM7TDMI	
ARM7EJ-S	
ARM720T	
ARM920T	
ARM922T	
ARM926EJ-S	
ARM940T	
ARM946E	
ARM966E	
ARM968E-S	
ARM1020E	
ARM1022E	
ARM1026EJ-S	
ARM1136J-S	
ARM1156T2-S	

ARM1176JZ-S	
Cortex-R4	
Cortex-R4F	
Cortex-A5UP	
Cortex-A5MPx1	
Cortex-A5MPx2	
Cortex-A5MPx3	
Cortex-A5MPx4	
Cortex-A8	(described in this document)
Cortex-A9UP	
Cortex-A9MPx1	
Cortex-A9MPx2	
Cortex-A9MPx3	
Cortex-A9MPx4	
Cortex-A7UP	
Cortex-A7MPx1	
Cortex-A7MPx2	
Cortex-A7MPx3	
Cortex-A7MPx4	
Cortex-A15UP	
Cortex-A15MPx1	
Cortex-A15MPx2	
Cortex-A15MPx3	
Cortex-A15MPx4	
Cortex-A17MPx1	
Cortex-A17MPx2	
Cortex-A17MPx3	
Cortex-A17MPx4	
AArch32	
AArch64	
Cortex-A32MPx1	
Cortex-A32MPx2	
Cortex-A32MPx3	
Cortex-A32MPx4	
Cortex-A35MPx1	
Cortex-A35MPx2	
Cortex-A35MPx3	
Cortex-A35MPx4	
Cortex-A53MPx1	
Cortex-A53MPx2	
Cortex-A53MPx3	
Cortex-A53MPx4	
Cortex-A55MPx1	
Cortex-A55MPx2	
Cortex-A55MPx3	

Cortex-A55MPx4	
Cortex-A57MPx1	
Cortex-A57MPx2	
Cortex-A57MPx3	
Cortex-A57MPx4	
Cortex-A72MPx1	
Cortex-A72MPx2	
Cortex-A72MPx3	
Cortex-A72MPx4	
Cortex-A73MPx1	
Cortex-A73MPx2	
Cortex-A73MPx3	
Cortex-A73MPx4	
Cortex-A75MPx1	
Cortex-A75MPx2	
Cortex-A75MPx3	
Cortex-A75MPx4	
MultiCluster	

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	41	mandatory	
DATA	32	41	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
EVENTI	input	optional	Event input signal, active on rising edge
EVENTO	output	optional	Event output signal, active on rising edge
VINITHI	input	optional	Configure HIVECS mode (SCTLR.V)
CFGEND	input	optional	Configure exception endianness (SCTLR.EE)
TEINIT	input	optional	Configure exception state at reset (SCTLR.TE)
CFGNMFI	input	optional	Configure non-maskable fast interrupts (SCTLR.NMFI)
reset	input	optional	Processor reset, active high
fiq	input	optional	FIQ interrupt, active high (negation of nFIQ)
irq	input	optional	IRQ interrupt, active high (negation of nIRQ)
sei	input	optional	System error interrupt, active on rising edge (negation of nSEI)
AXI_SLVERR	input	optional	AXI external abort type (DECERR=0, SLVERR=1)
CP15SDISABLE	input	optional	CP15SDISABLE (active high)

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
suppressCPSWarnings	Boolean	Suppress duplicate warnings generated using ARM_CP_CPSI or ARM_CP_CPSD message identifiers
showHiddenRegs	Boolean	Show hidden registers during register tracing
UAL	Boolean	Disassemble using UAL syntax
enableVFPAtReset	Boolean	Enable vector floating point (SIMD and VFP) instructions at reset. (Enables cp10/11 in CPACR and sets FPEXC.EN)
enableSystemBus	Boolean	Add 32-bit artifact System bus port, allowing system registers to be externally implemented
enableSystemMonitorBus	Boolean	Add 32-bit artifact SystemMonitor bus port, allowing system register accesses to be externally monitored
compatibility	Enumeration	Specify compatibility mode (ISA, gdb or nopSVC)
unpredictableR15	Enumeration	Specify behavior for UNPREDICTABLE uses of AArch32 R15 register (undefined, nop, raz_wi, execute or assert)
unpredictableModal	Enumeration	Specify behavior for UNPREDICTABLE instructions in certain AArch32 modes (for example, MRS using SPSR in System mode) (undefined, nop or assert)
maxSIMDUnroll	Uns32	If SIMD operations are supported, specify the maximum number of parallel SIMD operations to unroll (unrolled operations can be faster, but produce more verbose JIT code)
override_debugMask	Uns32	Specifies debug mask, enabling debug output for model components
ASIDCacheSize	Uns32	Specifies the number of different ASIDs for which TLB entries are cached; a value of 0 implies no limit
endian	Endian	Model endian
override_fcsePresent	Boolean	Specifies that FCSE is present (if true)
override_fpexcDexPresent	Boolean	Specifies that the FPEXC.DEX register field is implemented (if true)
override_advSIMDPresent	Boolean	Specifies that Advanced SIMD extensions are present (if true)
override_vfpPresent	Boolean	Specifies that VFP extensions are present (if true)
override_physicalBits	Uns32	Specifies the implemented physical bus bits (defaults to connected physical bus width)
override_SCTLR_V	Boolean	Override SCTLR.V with the passed value (enables high vectors; also configurable using VINITHI pin)

override_SCTLR_IE	Boolean	Override SCTLR.IE with the passed value (configures instruction endianness; also configurable using CFGIE pin)
override_SCTLR_EE	Boolean	Override SCTLR.EE with the passed value (configures exception data endianness; also configurable using CFGEE pin)
override_SCTLR_TE	Boolean	Override SCTLR.TE with the passed value (configures Thumb state for exception handling; also configurable using TEINIT pin)
override_SCTLR_NMFI	Boolean	Override SCTLR.NMFI with the passed value (configures NMFI state for exception handling; also configurable using CFGNMFI pin)
override_SCTLR_CP15BEN_Present	Boolean	Enable ARMv7 SCTLR.CP15BEN bit (CP15 barrier enable)
override_MIDR	Uns32	Override MIDR/MIDR_EL1 register
override_CTR	Uns32	Override CTR/CTR_EL0 register
override_TLBTR	Uns32	Override TLBTR register
override_CLIDR	Uns32	Override CLIDR/CLIDR_EL1 register
override_AIDR	Uns32	Override AIDR/AIDR_EL1 register
override_PFR0	Uns32	Override ID_PFR0/ID_PFR0_EL1 register
override_PFR1	Uns32	Override ID_PFR1/ID_PFR1_EL1 register
override_DFR0	Uns32	Override ID_DFR0/ID_DFR0_EL1 register
override_AFR0	Uns32	Override ID_AFR0/ID_AFR0_EL1 register
override_MMFR0	Uns32	Override ID_MMFR0/ID_MMFR0_EL1 register
override_MMFR1	Uns32	Override ID_MMFR1/ID_MMFR1_EL1 register
override_MMFR2	Uns32	Override ID_MMFR2/ID_MMFR2_EL1 register
override_MMFR3	Uns32	Override ID_MMFR3/ID_MMFR3_EL1 register
override_ISAR0	Uns32	Override ID_ISAR0/ID_ISAR0_EL1 register
override_ISAR1	Uns32	Override ID_ISAR1/ID_ISAR1_EL1 register
override_ISAR2	Uns32	Override ID_ISAR2/ID_ISAR2_EL1 register
override_ISAR3	Uns32	Override ID_ISAR3/ID_ISAR3_EL1 register
override_ISAR4	Uns32	Override ID_ISAR4/ID_ISAR4_EL1 register
override_ISAR5	Uns32	Override ID_ISAR5/ID_ISAR5_EL1 register
override_PMCR	Uns32	Override PMCR/PMCR_EL0 register (not functionally significant in the model)
override_PMCEID0	Uns64	Override PMCEID0/PMCEID0_EL0 register (not functionally significant in the model)
override_PMCEID1	Uns64	Override PMCEID1/PMCEID1_EL0 register (not functionally significant in the model)
override_DBGDIDR	Uns32	Override DBGDIDR register (not functionally significant in the model)
override_FPSID	Uns32	Override SIMD/VFP FPSID register
override_MVFR0	Uns32	Override SIMD/VFP MVFR0/MVFR0_EL1 register
override_MVFR1	Uns32	Override SIMD/VFP MVFR1/MVFR1_EL1 register
override_FPEXC	Uns32	Override SIMD/VFP FPEXC/FPEXC32_EL2 register
override_ERG	Uns32	Specifies exclusive reservation granule
override_CCSIDR_1I	Uns32	Override CCSIDR/CCSIDR_EL1 (level 1 instruction)
override_CCSIDR_1D	Uns32	Override CCSIDR/CCSIDR_EL1 (level 1 data)
override_CCSIDR_2I	Uns32	Override CCSIDR/CCSIDR_EL1 (level 2 instruction)
override_CCSIDR_2D	Uns32	Override CCSIDR/CCSIDR_EL1 (level 2 data)
override_CCSIDR_3I	Uns32	Override CCSIDR/CCSIDR_EL1 (level 3 instruction)
override_CCSIDR_3D	Uns32	Override CCSIDR/CCSIDR_EL1 (level 3 data)
override_CCSIDR_4I	Uns32	Override CCSIDR/CCSIDR_EL1 (level 4 instruction)
override_CCSIDR_4D	Uns32	Override CCSIDR/CCSIDR_EL1 (level 4 data)
override_CCSIDR_5I	Uns32	Override CCSIDR/CCSIDR_EL1 (level 5 instruction)
override_CCSIDR_5D	Uns32	Override CCSIDR/CCSIDR_EL1 (level 5 data)
override_CCSIDR_6I	Uns32	Override CCSIDR/CCSIDR_EL1 (level 6 instruction)

override_CCSIDR_6D	Uns32	Override CCSIDR/CCSIDR_EL1 (level 6 data)
override_CCSIDR_7I	Uns32	Override CCSIDR/CCSIDR_EL1 (level 7 instruction)
override_CCSIDR_7D	Uns32	Override CCSIDR/CCSIDR_EL1 (level 7 data)
override_STRoffsetPC12	Boolean	Specifies that STR/STR of PC should do so with 12:byte offset from the current instruction (if true), otherwise an 8:byte offset is used
override_fcseRequiresMMU	Boolean	Specifies that FCSE is active only when MMU is enabled (if true)
override_ignoreBadCp15	Boolean	Specifies whether invalid coprocessor 15 access should be ignored (if true) or cause Invalid Instruction exceptions (if false)
override_SGIDisable	Boolean	Override whether GIC SGIs may be disabled (if true) or are permanently enabled (if false)
override_condUndefined	Boolean	Force undefined instructions to take Undefined Instruction exception even if they are conditional
override_deviceStrongAligned	Boolean	Force accesses to Device and Strongly Ordered regions to be aligned
override_Control_V	Boolean	Override SCTLR.V with the passed value (deprecated, use override_SCTLR_V)
override_MainId	Uns32	Override MIDR register (deprecated, use override_MIDR)
override_CacheType	Uns32	Override CTR register (deprecated, use override_CTR)
override_TLBType	Uns32	Override TLBTR register (deprecated, use override_TLBTR)
override_InstructionAttributes0	Uns32	Override ID_ISAR0 register (deprecated, use override_ISAR0)
override_InstructionAttributes1	Uns32	Override ID_ISAR1 register (deprecated, use override_ISAR1)
override_InstructionAttributes2	Uns32	Override ID_ISAR2 register (deprecated, use override_ISAR2)
override_InstructionAttributes3	Uns32	Override ID_ISAR3 register (deprecated, use override_ISAR3)
override_InstructionAttributes4	Uns32	Override ID_ISAR4 register (deprecated, use override_ISAR4)
override_InstructionAttributes5	Uns32	Override ID_ISAR5 register (deprecated, use override_ISAR5)

Table 8.1: Parameters that can be set in: CPU

8.1 Parameter values

These are the current parameter values.

Name	Value
(Others)	
variant	Cortex-A8
verbose	T
suppressCPSWarnings	F
showHiddenRegs	F
UAL	T
enableVFPAtReset	F
enableSystemBus	F
enableSystemMonitorBus	F
compatibility	ISA

unpredictableR15	undefined
unpredictableModal	nop
maxSIMDUnroll	2
override_debugMask	0
ASIDCacheSize	8
endian	none
override_fcsePresent	F
override_fpexcDexPresent	F
override_advSIMDPresent	F
override_vfpPresent	F
override_physicalBits	32
override_SCTLR_V	F
override_SCTLR_IE	F
override_SCTLR_EE	F
override_SCTLR_TE	F
override_SCTLR_NMFI	F
override_SCTLR_CP15BEN_Present	F
override_MIDR	0x413fc082
override_CTR	0x82048004
override_TLBTR	0x202001
override_CLIDR	0xa000003
override_AIDR	0
override_PFR0	0x1131
override_PFR1	17
override_DFR0	0
override_AFR0	0
override_MMFR0	0x1100003
override_MMFR1	0x20000000
override_MMFR2	0x1202000
override_MMFR3	0x211
override_ISAR0	0x101111
override_ISAR1	0x13112111
override_ISAR2	0x21232031
override_ISAR3	0x11112131
override_ISAR4	0x11142
override_ISAR5	0
override_PMCR	0x41002000
override_PMCEID0	0
override_PMCEID1	0
override_DBGDIDR	0
override_FPSID	0x410330c3
override_MVFR0	0x11110222
override_MVFR1	0x11111
override_FPEXC	0
override_ERG	3

override_CCSIDR_1I	0x701fe019
override_CCSIDR_1D	0x201fe019
override_CCSIDR_2I	0
override_CCSIDR_2D	0
override_CCSIDR_3I	0
override_CCSIDR_3D	0
override_CCSIDR_4I	0
override_CCSIDR_4D	0
override_CCSIDR_5I	0
override_CCSIDR_5D	0
override_CCSIDR_6I	0
override_CCSIDR_6D	0
override_CCSIDR_7I	0
override_CCSIDR_7D	0
override_STRoffsetPC12	T
override_fcseRequiresMMU	F
override_ignoreBadCp15	F
override_SGIDisable	F
override_condUndefined	F
override_deviceStrongAligned	F
override_Control_V	F
override_MainId	0x413fc082
override_CacheType	0x82048004
override_TLBType	0x202001
override_InstructionAttributes0	0x101111
override_InstructionAttributes1	0x13112111
override_InstructionAttributes2	0x21232031
override_InstructionAttributes3	0x11112131
override_InstructionAttributes4	0x11142
override_InstructionAttributes5	0

Table 8.2: Parameter values

Chapter 9

Execution Modes

Mode	Code
User	16
FIQ	17
IRQ	18
Supervisor	19
Monitor	22
Abort	23
Undefined	27
System	31

Table 9.1: Modes implemented in: CPU

Chapter 10

Exceptions

Exception	Code
Reset	0
Undefined	1
SupervisorCall	2
SecureMonitorCall	3
PrefetchAbort	5
DataAbort	6
IRQ	8
FIQ	9

Table 10.1: Exceptions implemented in: CPU

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: CPU

This level in the model hierarchy has 5 commands.
This level in the model hierarchy has 15 register groups:

Group name	Registers
Core	16
Control	3
User	7
FIQ	8
IRQ	3
Supervisor	3
Monitor	3
Undefined	3
Abort	3
SIMD_VFP	32
SIMD_VFP_SYS	5
Coprocessor_32_bit	106
Coprocessor_32_bit_secure	21
Coprocessor_32_bit_non_secure	21
Integration_support	27

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: CPU

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 0x000003e4 can be modified)

Table 12.1: debugflags command arguments

12.1.2 dumpTLB

report TLB contents

Argument	Type	Description
-all	Boolean	show the contents of all TLBs (if False, show just the current TLB)

Table 12.2: dumpTLB command arguments

12.1.3 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 12.3: isync command arguments

12.1.4 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
-memory	String	show memory accesses by this instruction. Argument can be any combination of X (execute), L (load or store access) and S (system)
-off	Boolean	disable instruction tracing
-on	Boolean	enable instruction tracing
-processorname	Boolean	Include processor name in all trace lines
-registerchange	Boolean	show registers changed by this instruction
-registers	Boolean	show registers after each trace

Table 12.4: itrace command arguments

12.1.5 validateTLB

check TLB contents against page tables in memory and report incoherent entries

Argument	Type	Description
-all	Boolean	check all TLBs (if False, validate just the current TLB)
-verbose	Boolean	show all TLB entries (if False, show only incoherent entries)

Table 12.5: validateTLB command arguments

Chapter 13

Registers

13.1 Level 1: CPU

13.1.1 Core

Registers at level:1, type:CPU 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, type:CPU group:Core

13.1.2 Control

Registers at level:1, type:CPU group:Control

Name	Bits	Initial-Hex	RW	Description
fps	32	0	rw	archaic FPSCR view (for gdb)
cpsr	32	1d3	rw	
spsr	32	0	rw	

Table 13.2: Registers at level 1, type:CPU group:Control

13.1.3 User

Registers at level:1, type:CPU group:User

Name	Bits	Initial-Hex	RW	Description
r8_usr	32	0	rw	
r9_usr	32	0	rw	
r10_usr	32	0	rw	
r11_usr	32	0	rw	
r12_usr	32	0	rw	
sp_usr	32	0	rw	
lr_usr	32	0	rw	

Table 13.3: Registers at level 1, type:CPU group:User

13.1.4 FIQ

Registers at level:1, type:CPU group:FIQ

Name	Bits	Initial-Hex	RW	Description
r8_fiq	32	0	rw	
r9_fiq	32	0	rw	
r10_fiq	32	0	rw	
r11_fiq	32	0	rw	
r12_fiq	32	0	rw	
sp_fiq	32	0	rw	
lr_fiq	32	0	rw	
spsr_fiq	32	0	rw	

Table 13.4: Registers at level 1, type:CPU group:FIQ

13.1.5 IRQ

Registers at level:1, type:CPU group:IRQ

Name	Bits	Initial-Hex	RW	Description
sp_irq	32	0	rw	
lr_irq	32	0	rw	
spsr_irq	32	0	rw	

Table 13.5: Registers at level 1, type:CPU group:IRQ

13.1.6 Supervisor

Registers at level:1, type:CPU group:Supervisor

Name	Bits	Initial-Hex	RW	Description
sp_svc	32	0	rw	
lr_svc	32	0	rw	
spsr_svc	32	0	rw	

Table 13.6: Registers at level 1, type:CPU group:Supervisor

13.1.7 Monitor

Registers at level:1, type:CPU group:Monitor

Name	Bits	Initial-Hex	RW	Description

sp_mon	32	0	rw	
lr_mon	32	0	rw	
spsr_mon	32	0	rw	

Table 13.7: Registers at level 1, type:CPU group:Monitor

13.1.8 Undefined

Registers at level:1, type:CPU group:Undefined

Name	Bits	Initial-Hex	RW	Description
sp_undef	32	0	rw	
lr_undef	32	0	rw	
spsr_undef	32	0	rw	

Table 13.8: Registers at level 1, type:CPU group:Undefined

13.1.9 Abort

Registers at level:1, type:CPU group:Abort

Name	Bits	Initial-Hex	RW	Description
sp_abt	32	0	rw	
lr_abt	32	0	rw	
spsr_abt	32	0	rw	

Table 13.9: Registers at level 1, type:CPU group:Abort

13.1.10 SIMD_VFP

Registers at level:1, type:CPU group:SIMD_VFP

Name	Bits	Initial-Hex	RW	Description
d0	64	0	rw	
d1	64	0	rw	
d2	64	0	rw	
d3	64	0	rw	
d4	64	0	rw	
d5	64	0	rw	
d6	64	0	rw	
d7	64	0	rw	
d8	64	0	rw	
d9	64	0	rw	
d10	64	0	rw	
d11	64	0	rw	
d12	64	0	rw	
d13	64	0	rw	
d14	64	0	rw	
d15	64	0	rw	
d16	64	0	rw	
d17	64	0	rw	
d18	64	0	rw	
d19	64	0	rw	
d20	64	0	rw	
d21	64	0	rw	

d22	64	0	rw	
d23	64	0	rw	
d24	64	0	rw	
d25	64	0	rw	
d26	64	0	rw	
d27	64	0	rw	
d28	64	0	rw	
d29	64	0	rw	
d30	64	0	rw	
d31	64	0	rw	

Table 13.10: Registers at level 1, type:CPU group:SIMD_VFP

13.1.11 SIMD_VFP_SYS

Registers at level:1, type:CPU group:SIMD_VFP_SYS

Name	Bits	Initial-Hex	RW	Description
FPSID	32	410330c3	r-	floating-point system ID
FPSCR	32	0	rw	floating-point status/control
FPEXC	32	0	rw	floating-point exception
MVFR0	32	11110222	r-	Media/VFP feature 0
MVFR1	32	11111	r-	Media/VFP feature 1

Table 13.11: Registers at level 1, type:CPU group:SIMD_VFP_SYS

13.1.12 Coprocessor_32_bit

Registers at level:1, type:CPU group:Coprocessor_32_bit

Name	Bits	Initial-Hex	RW	Description
ACTLR	32	2	rw	Auxiliary Control
ADFSR	32	0	rw	Auxilary Data Fault Status
AIDR	32	0	r-	Auxiliary ID
AIFSR	32	0	rw	Auxilary Instruction Fault Status
ATS1CPR	32	-	-w	Address Translate Stage 1 Current State EL1 Read
ATS1CPW	32	-	-w	Address Translate Stage 1 Current State EL1 Write
ATS1CUR	32	-	-w	Address Translate Stage 1 Current State Unprivileged Read
ATS1CUW	32	-	-w	Address Translate Stage 1 Current State Unprivileged Write
ATS12NSOPR	32	-	-w	Address Translate Stages 1 and 2 Non-Secure Only EL1 Read
ATS12NSOPW	32	-	-w	Address Translate Stages 1 and 2 Non-Secure Only EL1 Write
ATS12NSOUR	32	-	-w	Address Translate Stages 1 and 2 Non-Secure Only Unprivileged Read
ATS12NSOUW	32	-	-w	Address Translate Stages 1 and 2 Non-Secure Only Unprivileged Write
BPIALL	32	-	-w	Branch Predictor Invalidate All
BPIMVA	32	-	-w	Branch Predictor Invalidate by VA
CCSIDR	32	201fe019	r-	Cache Size ID
CLIDR	32	a000003	r-	Cache Level ID
CONTEXTIDR	32	0	rw	Context ID
CP15DMB	32	-	-w	CP15 Data Memory Barrier
CP15DSB	32	-	-w	CP15 Data Synchronization Barrier
CP15ISB	32	-	-w	CP15 Instruction Synchronization Barrier
CP15NOP	32	-	-w	CP15 NOP
CPACR	32	0	rw	Coprocessor Access Control

CSSELR	32	1	rw	Cache Size Selection
CTR	32	82048004	r-	Cache Type
DACR	32	0	rw	Domain Access Control
DBGDIDR	32	0	r-	Debug ID
DCCIMVAC	32	-	-w	Data Cache Line Clean and Invalidate by VA to PoC
DCCISW	32	-	-w	Data Cache Line Clean and Invalidate by Set/Way
DCCMVAC	32	-	-w	Data Cache Line Clean by VA to PoC
DCCMVAU	32	-	-w	Data Cache Line Clean by VA to PoU
DCCSW	32	-	-w	Data Cache Line Clean by Set/Way
DCIMVAC	32	-	-w	Data Cache Line Invalidate by VA to PoC
DCISW	32	-	-w	Data Cache Line Invalidate by Set/Way
DFAR	32	0	rw	Data Fault Address
DFSR	32	0	rw	Data Fault Status
DTLBIALL	32	-	-w	Invalidate Entire Data TLB
DTLBIASID	32	-	-w	Invalidate Data TLB by ASID
DTLBIMVA	32	-	-w	Invalidate Data TLB by VA
DTLBLR	32	0	rw	TLB Lockdown
DTLBPL	32	-	-w	Data TLB Preload
FCSEIDR	32	0	rw	FCSE Process ID
ICIALLU	32	-	-w	Instruction Cache Invalidate All
ICIMVAU	32	-	-w	Instruction Cache Invalidate by VA
ID_AFR0	32	0	r-	Auxiliary Feature 0
ID_DFR0	32	0	r-	Debug Feature 0
ID_ISAR0	32	101111	r-	Instruction Set Attribute 0
ID_ISAR1	32	13112111	r-	Instruction Set Attribute 1
ID_ISAR2	32	21232031	r-	Instruction Set Attribute 2
ID_ISAR3	32	11112131	r-	Instruction Set Attribute 3
ID_ISAR4	32	11142	r-	Instruction Set Attribute 4
ID_ISAR5	32	0	r-	Instruction Set Attribute 5
ID_MMFR0	32	1100003	r-	Memory Model Feature 0
ID_MMFR1	32	20000000	r-	Memory Model Feature 1
ID_MMFR2	32	1202000	r-	Memory Model Feature 2
ID_MMFR3	32	211	r-	Memory Model Feature 3
ID_PFR0	32	1131	r-	Processor Feature 0
ID_PFR1	32	11	r-	Processor Feature 1
IFAR	32	0	rw	Instruction Fault Address
IFSR	32	0	rw	Instruction Fault Status
ISR	32	0	r-	Interrupt Status
ITLBIALL	32	-	-w	Invalidate Entire Instruction TLB
ITLBIASID	32	-	-w	Invalidate Instruction TLB by ASID
ITLBIMVA	32	-	-w	Invalidate Instruction TLB by VA
ITLBLR	32	0	rw	Instruction TLB Lockdown
ITLBPL	32	-	-w	Instruction TLB Preload
JIDR	32	0	rw	Jazelle ID
JMCR	32	0	rw	Jazelle Main Configuration
JOSCR	32	0	rw	Jazelle OS Control
L2CACTLR	32	42	rw	L2 Cache Auxiliary Control
L2CLR	32	0	rw	L2 Cache Lockdown
MIDR	32	413fc082	r-	Main ID
MPIDR	32	0	r-	Multiprocessor Affinity
MVBAR	32	0	rw	Monitor Vector Base Address
NMRR	32	44e048e0	rw	Normal Memory Remap
NSACR	32	0	rw	Non-Secure Access Control
PAR	32	0	rw	Physical Address
PMCCNTR	32	0	rw	Performance Monitors Cycle Count
PMCNTENCLR	32	0	rw	Performance Monitors Count Enable Clear

PMCNTENSET	32	0	rw	Performance Monitors Count Enable Set
PMCR	32	41002000	rw	Performance Monitors Control
PMINTENCLR	32	0	rw	Performance Monitors Interrupt Enable Clear
PMINTENSET	32	0	rw	Performance Monitors Interrupt Enable Set
PMOVSR	32	0	rw	Performance Monitors Overflow Flag Status
PMSELR	32	0	rw	Performance Monitors Event Counter Selection
PMSWINC	32	-	-w	Performance Monitors Software Increment
PMUSERENR	32	0	rw	Performance Monitors User Enable
PMXEVCNTR	32	0	rw	Performance Monitors Selected Event Count
PMXEVTYPER	32	0	rw	Performance Monitors Selected Event Type
PRRR	32	98aa4	rw	Primary Region Remap
SCR	32	0	rw	Secure Configuration
SCTLR	32	c50078	rw	System Control
SDER	32	0	rw	Secure Debug Enable
TCMTR	32	0	r-	TCM Type
TEECR	32	0	rw	T32EE Configuration
TEEHBR	32	0	rw	T32EE Handler Base
TLBIALL	32	-	-w	Invalidate Entire Unified TLB
TLBIASID	32	-	-w	Invalidate Unified TLB by ASID
TLBIMVA	32	-	-w	Invalidate Unified TLB by VA
TLBTR	32	202001	r-	TLB Type
TPIDRPRW	32	0	rw	PL0 Read/Write Software Thread ID
TPIDRURO	32	0	rw	PL0 Read-Only Software Thread ID
TPIDRURW	32	0	rw	PL1 Software Thread ID
TTBCR	32	0	rw	Translation Table Base Control
TTBR0	32	0	rw	Translation Table Base 0
TTBR1	32	0	rw	Translation Table Base 1
VBAR	32	0	rw	Vector Base Address

Table 13.12: Registers at level 1, type:CPU group:Coprocessor_32_bit

13.1.13 Coprocessor_32_bit_secure

Registers at level:1, type:CPU group:Coprocessor_32_bit_secure

Name	Bits	Initial-Hex	RW	Description
ADFSR_S	32	0	rw	Auxilary Data Fault Status
AIFSR_S	32	0	rw	Auxilary Instruction Fault Status
CONTEXTIDR_S	32	0	rw	Context ID
CSSELR_S	32	1	rw	Cache Size Selection
DACR_S	32	0	rw	Domain Access Control
DFAR_S	32	0	rw	Data Fault Address
DFSR_S	32	0	rw	Data Fault Status
FCSEIDR_S	32	0	rw	FCSE Process ID
IFAR_S	32	0	rw	Instruction Fault Address
IFSR_S	32	0	rw	Instruction Fault Status
NMRR_S	32	44e048e0	rw	Normal Memory Remap
PAR_S	32	0	rw	Physical Address
PRRR_S	32	98aa4	rw	Primary Region Remap
SCTLR_S	32	c50078	rw	System Control
TPIDRPRW_S	32	0	rw	PL0 Read/Write Software Thread ID
TPIDRURO_S	32	0	rw	PL0 Read-Only Software Thread ID
TPIDRURW_S	32	0	rw	PL1 Software Thread ID
TTBCR_S	32	0	rw	Translation Table Base Control
TTBR0_S	32	0	rw	Translation Table Base 0
TTBR1_S	32	0	rw	Translation Table Base 1

VBAR_S	32	0	rw	Vector Base Address

Table 13.13: Registers at level 1, type:CPU group:Coprocessor_32_bit_secure

13.1.14 Coprocessor_32_bit_non_secure

Registers at level:1, type:CPU group:Coprocessor_32_bit_non_secure

Name	Bits	Initial-Hex	RW	Description
ADFSR_NS	32	0	rw	Auxilary Data Fault Status
AIFSR_NS	32	0	rw	Auxilary Instruction Fault Status
CONTEXTIDR_NS	32	0	rw	Context ID
CSSELR_NS	32	1	rw	Cache Size Selection
DACR_NS	32	0	rw	Domain Access Control
DFAR_NS	32	0	rw	Data Fault Address
DFSR_NS	32	0	rw	Data Fault Status
FCSEIDR_NS	32	0	rw	FCSE Process ID
IFAR_NS	32	0	rw	Instruction Fault Address
IFSR_NS	32	0	rw	Instruction Fault Status
NMRR_NS	32	44e048e0	rw	Normal Memory Remap
PAR_NS	32	0	rw	Physical Address
PRRR_NS	32	98aa4	rw	Primary Region Remap
SCTLR_NS	32	c50078	rw	System Control
TPIDRPRW_NS	32	0	rw	PL0 Read/Write Software Thread ID
TPIDRURO_NS	32	0	rw	PL0 Read-Only Software Thread ID
TPIDRURW_NS	32	0	rw	PL1 Software Thread ID
TTBCR_NS	32	0	rw	Translation Table Base Control
TTBR0_NS	32	0	rw	Translation Table Base 0
TTBR1_NS	32	0	rw	Translation Table Base 1
VBAR_NS	32	0	rw	Vector Base Address

Table 13.14: Registers at level 1, type:CPU group:Coprocessor_32_bit_non_secure

13.1.15 Integration_support

Registers at level:1, type:CPU group:Integration_support

Name	Bits	Initial-Hex	RW	Description
transactPL	32	1	r-	privilege level of current memory transaction
transactAT	32	0	r-	current memory transaction type: PA=1, VA=0
artifactPAR	64	0	r-	result of address translation for artifact write to ATS1CPR etc
PTWBankValid	8	0	r-	bitmask of valid banks (0x01 is stage 1, 0x02 is stage 2, 0x04-0x20 are stage 2 walks initiated by stage 1 level 0-3 entry lookups, respectively)
PTWAddressValid	8	0	r-	bitmask of valid bits for each of PTWAddressL0...PTWAddressL3, PTWBase, PTWInput and PTWOutput in current bank
PTWAddressNS	8	0	r-	bitmask of Non-Secure bits for each of PTWAddressL0...PTWAddressL3, PTWBase and PTWOutput in current bank (PTWInput bit is always 0)
PTWValueValid	8	0	r-	bitmask of valid bits for each of PTWValueL0...PTWValueL3 in current bank
PTWAddressL0	64	0	r-	current bank PTW address, level 0
PTWAddressL1	64	0	r-	current bank PTW address, level 1
PTWAddressL2	64	0	r-	current bank PTW address, level 2

PTWAddressL3	64	0	r-	current bank PTW address, level 3
PTWValueL0	64	0	r-	current bank PTW value, level 0
PTWValueL1	64	0	r-	current bank PTW value, level 1
PTWValueL2	64	0	r-	current bank PTW value, level 2
PTWValueL3	64	0	r-	current bank PTW value, level 3
PTWBase	64	0	r-	current bank PTW table base address
PTWInput	64	0	r-	current bank PTW input address
PTWOutput	64	0	r-	current bank PTW output address
PTWPgSize	64	0	r-	current bank PTW page size (Valid only when PTWOutput is valid)
PTWI_EL1S	64	-	-w	perform EL1(S) stage 1 page table walk for fetch, filling PTW query registers
PTWD_EL1S	64	-	-w	perform EL1(S) stage 1 page table walk for load/store, filling PTW query registers
PTWI_EL1NS	64	-	-w	perform EL1(NS) stage 1 page table walk for fetch, filling PTW query registers
PTWD_EL1NS	64	-	-w	perform EL1(NS) stage 1 page table walk for load/store, filling PTW query registers
PTWI_current	64	-	-w	perform current mode page table walk for fetch, filling PTW query registers
PTWD_current	64	-	-w	perform current mode page table walk for load/store, filling PTW query registers
ResetTLBs	8	-	-w	reset all implemented TLBs to initial state
HaltReason	8	0	r-	bit field indicating halt reason

Table 13.15: Registers at level 1, type:CPU group:Integration_support