OVP Guide to Using Processor Models

Model specific information for ARM_Cortex-A32MPx1

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

Author	Imperas Software Limited
Version	20200630.0
Filename	OVP_Model_Specific_Information_arm_Cortex-A32MPx1.pdf
Created	2 July 2020
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.

Debug registers are implemented but non-functional (which is sufficient to allow operating systems such as Linux to boot). Debug state is not implemented.

The GICv3 block is implemented without any ITS. Implementation-defined GICR registers

for control of LPIs (GICR_SETLPIR, GICR_CLRLPIR, GICR_INVLPIR, GICR_INVALLR and GICR_SYNCR) are all implemented.

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

AArch32 is implemented at EL3, EL2, EL1 and EL0.

Virtualization extensions are implemented.

1.5.2 Memory System

Large physical address 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 stage 1 secure, non-secure and Hypervisor address translation is implemented. VMSA stage 2 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)

1.5.4 Generic Timer

Generic Timer is present. Use parameter "override_timerScaleFactor" to specify the counter rate as a fraction of the processor MIPS rate (e.g. 10 implies Generic Timer counters increment once every 10 processor instructions).

1.5.5 Generic Interrupt Controller

GIC block is implemented (GICv3, including security extensions). Accesses to GIC registers can be viewed externally by connecting to the 32-bit GICRegisters and GICDRegisters bus ports. Secure register accesses are at offset 0x0 on these busses; for example, a secure access to GICD register GICD_CTLR can be observed by monitoring address 0x00000000 of bus GICDRegisters. Non-secure accesses are at offset 0x80000000 on these busses; for example, a non-secure access to GICD register GICD_CTLR can be observed by monitoring address 0x80000000 of bus GICDRegisters

The internal GIC block can be disabled by raising signal GICCDISABLE, in which case the GIC needs to be modeled using a platform component instead. Input signals vfiq_CPU<N>and virq_CPU<N>can be used by this component to raise virtual FIQ and IRQ interrupts on cores in the cluster if required.

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 0x020: enable debugging of reads and writes of GIC block registers.

Value 0x040: enable debugging of exception routing via the GIC model component.

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 "undefined".

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

Bit 2: EL2 stage 1 TLB

Bit 3: EL0/EL1 non-secure stage 2 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.

1.9 Description

ARM Processor Model

1.10 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.

1.11 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.

Debug registers are implemented but non-functional (which is sufficient to allow operating systems such as Linux to boot). Debug state is not implemented.

The GICv3 block is implemented without any ITS. Implementation-defined GICR registers for control of LPIs (GICR_SETLPIR, GICR_CLRLPIR, GICR_INVLPIR, GICR_INVALLR and GICR_SYNCR) are all implemented.

1.12 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.

1.13 Features

1.13.1 Core Features

AArch32 is implemented at EL3, EL2, EL1 and EL0.

Virtualization extensions are implemented.

1.13.2 Memory System

Large physical address 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 stage 1 secure, non-secure and Hypervisor address translation is implemented. VMSA stage 2 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

1.13.3 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)

1.13.4 Generic Timer

Generic Timer is present. Use parameter "override_timerScaleFactor" to specify the counter rate as a fraction of the processor MIPS rate (e.g. 10 implies Generic Timer counters increment once every 10 processor instructions).

1.13.5 Generic Interrupt Controller

GIC block is implemented (GICv3, including security extensions). Accesses to GIC registers can be viewed externally by connecting to the 32-bit GICRegisters and GICDRegisters bus ports. Secure register accesses are at offset 0x0 on these busses; for example, a secure access to GICD register

GICD_CTLR can be observed by monitoring address 0x00000000 of bus GICDRegisters. Non-secure accesses are at offset 0x80000000 on these busses; for example, a non-secure access to GICD register GICD_CTLR can be observed by monitoring address 0x80000000 of bus GICDRegisters

The internal GIC block can be disabled by raising signal GICCDISABLE, in which case the GIC needs to be modeled using a platform component instead. Input signals vfiq_CPU<N>and virq_CPU<N>can be used by this component to raise virtual FIQ and IRQ interrupts on cores in the cluster if required.

1.14 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 0x020: enable debugging of reads and writes of GIC block registers.

Value 0x040: enable debugging of exception routing via the GIC model component.

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.

1.15 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.

1.15.1 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.

1.15.2 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.

1.15.3 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).

1.15.4 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.

1.15.5 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.

1.15.6 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".

1.15.7 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 "undefined".

1.16 Integration Support

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

1.16.1 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.

1.16.2 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.

1.16.3 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.

1.16.4 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).

1.16.5 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.

1.16.6 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

Bit 2: EL2 stage 1 TLB

Bit 3: EL0/EL1 non-secure stage 2 TLB

1.16.7 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.

1.16.8 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.

1.16.9 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	
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	(described in this document)
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	
GICRegisters	32	32	optional	GIC memory-mapped register block
GICDRegisters	32	32	optional	GICD memory-mapped register block

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
SPI32	input	optional	Shared peripheral interrupt
SPI33	input	optional	Shared peripheral interrupt
SPI34	input	optional	Shared peripheral interrupt
SPI35	input	optional	Shared peripheral interrupt
SPI36	input	optional	Shared peripheral interrupt
SPI37	input	optional	Shared peripheral interrupt
SPI38	input	optional	Shared peripheral interrupt
SPI39	input	optional	Shared peripheral interrupt
SPI40	input	optional	Shared peripheral interrupt
SPI41	input	optional	Shared peripheral interrupt
SPI42	input	optional	Shared peripheral interrupt
SPI43	input	optional	Shared peripheral interrupt
SPI44	input	optional	Shared peripheral interrupt
SPI45	input	optional	Shared peripheral interrupt
SPI46	input	optional	Shared peripheral interrupt
SPI47	input	optional	Shared peripheral interrupt
SPI48	input	optional	Shared peripheral interrupt
SPI49	input	optional	Shared peripheral interrupt
SPI50	input	optional	Shared peripheral interrupt
SPI51	input	optional	Shared peripheral interrupt
SPI52	input	optional	Shared peripheral interrupt
SPI53	input	optional	Shared peripheral interrupt
SPI54	input	optional	Shared peripheral interrupt
SPI55	input	optional	Shared peripheral interrupt
SPI56	input	optional	Shared peripheral interrupt
SPI57	input	optional	Shared peripheral interrupt
SPI58	input	optional	Shared peripheral interrupt
SPI59	input	optional	Shared peripheral interrupt
SPI60	input	optional	Shared peripheral interrupt
SPI61	input	optional	Shared peripheral interrupt
SPI62	input	optional	Shared peripheral interrupt

SPI63	input	optional	Shared peripheral interrupt
SPI64	input	optional	Shared peripheral interrupt
SPI65	input	optional	Shared peripheral interrupt
SPI66	input	optional	Shared peripheral interrupt
SPI67	input	optional	Shared peripheral interrupt
SPI68	input	optional	Shared peripheral interrupt
SPI69	input	optional	Shared peripheral interrupt
SPI70	input	optional	Shared peripheral interrupt
SPI71	input	optional	Shared peripheral interrupt
SPI72	input	optional	Shared peripheral interrupt
SPI73	input	optional	Shared peripheral interrupt
SPI74	input	optional	Shared peripheral interrupt
SPI75	input	optional	Shared peripheral interrupt
SPI76	input	optional	Shared peripheral interrupt
SPI77	input	optional	Shared peripheral interrupt
SPI78	input	optional	Shared peripheral interrupt
SPI79	input	optional	Shared peripheral interrupt
SPI80	input	optional	Shared peripheral interrupt
SPI81	input	optional	Shared peripheral interrupt
SPI82	input	optional	Shared peripheral interrupt
SPI83	input	optional	Shared peripheral interrupt
SPI84	input	optional	Shared peripheral interrupt
SPI85	input	optional	Shared peripheral interrupt
SPI86	input	optional	Shared peripheral interrupt
SPI87	input	optional	Shared peripheral interrupt
SPI88	input	optional	Shared peripheral interrupt
SPI89	input	optional	Shared peripheral interrupt
SPI90	input	optional	Shared peripheral interrupt
SPI91	input	optional	Shared peripheral interrupt
SPI92	input	optional	Shared peripheral interrupt
SPI93	input	optional	Shared peripheral interrupt
SPI94	input	optional	Shared peripheral interrupt
SPI95	input	optional	Shared peripheral interrupt
SPIVector	input	optional	Shared peripheral interrupt vectorized input
periphReset	input	optional	Peripheral reset (active high)
GICCDISABLE	input	optional	GIC CPU interface logic disable (active high, sampled on rising edge of periphReset)
EVENTI	input	optional	Event input signal, active on rising edge
EVENTO	output	optional	Event output signal, active on rising edge
PPI16_CPU0	input	optional	Private peripheral interrupt
PPI17_CPU0	input	optional	Private peripheral interrupt
PPI18_CPU0	input	optional	Private peripheral interrupt
PPI19_CPU0	input	optional	Private peripheral interrupt

PPI20_CPU0	input	optional	Private peripheral interrupt
PPI21_CPU0	input	optional	Private peripheral interrupt
PPI22_CPU0	input	optional	Private peripheral interrupt
PPI23_CPU0	input	optional	Private peripheral interrupt
PPI24_CPU0	input	optional	Private peripheral interrupt
PPI25_CPU0	input	optional	Private peripheral interrupt
PPI26_CPU0	input	optional	Private peripheral interrupt
PPI27_CPU0	input	optional	Private peripheral interrupt
PPI28_CPU0	input	optional	Private peripheral interrupt
PPI29_CPU0	input	optional	Private peripheral interrupt
PPI30_CPU0	input	optional	Private peripheral interrupt
PPI31_CPU0	input	optional	Private peripheral interrupt
CNTVIRQ_CPU0	output	optional	Virtual timer event (active high)
CNTPSIRQ_CPU0	output	optional	Secure physical timer event (active high)
CNTPNSIRQ_CPU0	output	optional	Non-secure physical timer event (active high)
CNTPHPIRQ_CPU0	output	optional	Hypervisor physical timer event (active high)
IRQOUT_CPU0	output	optional	IRQ wakeup
FIQOUT_CPU0	output	optional	FIQ wakeup
CLUSTERIDAFF1	input	optional	Configure MPIDR.Aff1
CLUSTERIDAFF2	input	optional	Configure MPIDR.Aff2
VINITHI_CPU0	input	optional	Configure HIVECS mode (SCTLR.V)
CFGEND_CPU0	input	optional	Configure exception endianness (SCTLR.EE)
CFGTE_CPU0	input	optional	Configure exception state at reset (SCTLR.TE)
reset_CPU0	input	optional	Processor reset, active high
fiq_CPU0	input	optional	FIQ interrupt, active high (negation of nFIQ)
irq_CPU0	input	optional	IRQ interrupt, active high (negation of nIRQ)
sei_CPU0	input	optional	System error interrupt, active on rising edge (negation of nSEI)
vfiq_CPU0	input	optional	Virtual FIQ interrupt, active high (negation of nVFIQ)
virq_CPU0	input	optional	Virtual IRQ interrupt, active high (negation of nVIRQ)
vsei_CPU0	input	optional	Virtual system error interrupt, active on rising edge (negation of nVSEI)
AXI_SLVERR_CPU0	input	optional	AXI external abort type (DECERR=0, SLVERR=1)
CP15SDISABLE_CPU0	input	optional	CP15SDISABLE (active high)
PMUIRQ_CPU0	output	optional	Performance monitor event (active high)
SMPEN_CPU0	output	optional	CPUECTLR.SMPEN current value

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
disableGICModel	Boolean	Disable the internal GIC model entirely
enableGICv3	Boolean	Enable/disable GICv3 support
enableGICv2_64kB_Page	Boolean	Enable 64kB page size for GICv2 memory-mapped register groups (Xilinx Zynq Ultrascale support)
supportSTATUSR	Boolean	Enable/disable support for GICv3 GIC[CDV]_STATUSR registers
enableVFPAtReset	Boolean	Enable vector floating point (SIMD and VFP) instructions at reset. (Enables cp10/11 in CPACR and sets FPEXC.EN)
SVEImplementedSizes	Uns32	For processors with ARMv8.2 SVE extension, mask of configurable vector sizes (vector length N is configurable if mask contains 1<<((N/128)-1))
SVEFaultUnknown	Uns64	For processors with ARMv8.2 SVE extension, UNKNOWN value returned for suppressed or inactive FFR elements
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
distinctMTCores	Boolean	For multi-threaded (MT) processors, simulate threads as separate cores (otherwise, simulate MT threads as a single entity)
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_numCPUs	Uns32	Specify the number of cores in a multiprocessor (maximum of 8 for GICv1/GICv2)
override_affinityMask	Uns32	Specify bitmask of implemented affinity bits in format Aff3:Aff2:Aff1:Aff0 (each a byte)
override_MPIDR_MT	Boolean	Specifies that processor is multithreaded
override_MPIDR_Aff0	Uns32	Override Aff0 field in MPIDR/MPIDR_EL1 register
override_MPIDR_Aff1	Uns32	Override Aff1 field in MPIDR/MPIDR_EL1 register (also possible by writing CLUSTERIDAFF1 configuration net)
override_MPIDR_Aff2	Uns32	Override Aff2 field in MPIDR/MPIDR_EL1 register (also possible by writing CLUSTERIDAFF2 configuration net)
override_MPIDR_Aff3	Uns32	Override Aff3 field in MPIDR_EL1 register (also possible by writing CLUSTERIDAFF3 configuration net)
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_timerScaleFactor	Uns32	Specifies the fraction of MIPS rate to use for MPCore timers (generic timers or global/local/watchdogs depending on implementation). Defaults to 20 for generic timers, 2 for others
override_GICD_NSACRPresent	Boolean	Specifies that optional GICD_NSACR distributor registers are present (GICv2 only)
override_GICD_PPISRPresent	Boolean	Specifies that implementation-specific GICD_PPISR distributor register is present (GICv1 ICDPPIS/ICPPISR, GICv1 and GICv2 only)
override_GICD_SPISRPresent	Boolean	Specifies that implementation-specific GICD_SPISR distributor registers are present (GICv1 ICDSPIS/ICSPISR)
override_GICv3_DistributorBase	Uns64	Specify distributor register block base address (GICv3 only)
override_GICv3_E1NWFPresent	Boolean	Specifies that GICR_CTLR.E1NWF is implemented (GICv3 only)
override_GIC_PPIMask	Uns32	Specify bitmask of implemented PPIs in the GIC (e.g. ID16 is 0x0001, ID31 is 0x8000)
override_GICCDISABLE	Boolean	Specify initial value of GICCDISABLE
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_CBAR	Uns32	Override Configuration Base Address Register (Corresponds to value on PERIPHBASE input pins)
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_MMFR4	Uns32	Override ID_MMFR4/ID_MMFR4_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_ISAR6	Uns32	Override ID_ISAR6/ID_ISAR6_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_DBGDEVID	Uns32	Override DBGDEVID register (not functionally significant in the model)
override_DBGDEVID1	Uns32	Override DBGDEVID1 register (not functionally significant in the model)
override_DBGDEVID2	Uns32	Override DBGDEVID2 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_MVFR2	Uns32	Override SIMD/VFP MVFR2/MVFR2_EL1 register
override_FPEXC	Uns32	Override SIMD/VFP FPEXC/FPEXC32_EL2 register
override_GICC_IIDR	Uns32	Override GICC_IIDR register (GICv1 ICCIIDR)
override_GICD_TYPER	Uns32	Override GICD_TYPER register (GICv1 ICDICTR)
override_GICD_TYPER_ITLines	Uns32	Override ITLinesNumber field of GICD_TYPER register (GICv1 ICDICTR)
override_GICD_ICFGRN	Uns32	Override reset value of GICD_ICFGR2...GICD_ICFGRn (GICv1 ICDICFR2...ICDICFRn)
override_GICD_IIDR	Uns32	Override GICD_IIDR register (GICv1 ICDIIDR)
override_GICH_VTR	Uns32	Override GICH_VTR register
override_GICR_IIDR	Uns32	Override GICR_IIDR register (GICv3 and later)
override_GITS_IIDR	Uns32	Override GITS_IIDR register (GICv3 and later)
override_GITS_TYPER	Uns64	Override GITS_TYPER register (GICv3 and later)

override_ICCPMRBits	Uns32	Specify the number of writable bits in GICC_PMR (GICv1 ICCPMR)
override_minICCBPR	Uns32	Specify the minimum possible value for GICC_BPR (GICv1 ICCBPR)
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_RMR	Uns32	Override RMR register alias at highest-implemented exception level
override_RVBAR	Uns64	Override RVBAR register alias at highest-implemented exception level
override_AA64PFR0_EL1	Uns64	Override ID_AA64PFR0_EL1 register
override_AA64PFR1_EL1	Uns64	Override ID_AA64PFR1_EL1 register
override_AA64DFR0_EL1	Uns64	Override ID_AA64DFR0_EL1 register
override_AA64DFR1_EL1	Uns64	Override ID_AA64DFR1_EL1 register
override_AA64AFR0_EL1	Uns64	Override ID_AA64AFR0_EL1 register
override_AA64AFR1_EL1	Uns64	Override ID_AA64AFR1_EL1 register
override_AA64ISAR0_EL1	Uns64	Override ID_AA64ISAR0_EL1 register
override_AA64ISAR1_EL1	Uns64	Override ID_AA64ISAR1_EL1 register
override_AA64MMFR0_EL1	Uns64	Override ID_AA64MMFR0_EL1 register
override_AA64MMFR1_EL1	Uns64	Override ID_AA64MMFR1_EL1 register
override_AA64MMFR2_EL1	Uns64	Override ID_AA64MMFR2_EL1 register
override_DCZID_EL0	Uns32	Override DCZID_EL0 register
override_LORID_EL1	Uns32	Override LORID_EL1 register (ARMv8.1 only)
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_stage1SZMinFault	Boolean	Enable Level 0 Translation faults when stage 1 TCR_ELx.TxSZ <minimum (by default, clamp to minimum)
override_stage1SZMaxFault	Boolean	Enable Level 0 Translation faults when stage 1 TCR_ELx.TxSZ >maximum (by default, clamp to maximum)

override_stage2SZMinFault	Boolean	Enable Level 0 Translation faults when stage 2 VTCR_EL2.T0SZ <minimum (by default, clamp to minimum)
override_stage2SZMaxFault	Boolean	Enable Level 0 Translation faults when stage 2 VTCR_EL2.T0SZ >maximum (by default, clamp to maximum)
override_mask_ACTLR_EL1	Uns64	Override mask of writable bits in AArch64 ACTLR_EL1 register, or AArch32 non-secure ACTLR/ACTLR2 pair, if implemented
override_mask_ACTLR_EL2	Uns64	Override mask of writable bits in AArch64 ACTLR_EL2 register, or AArch32 HACTLR/HACTLR2 pair, if implemented
override_mask_ACTLR_EL3	Uns64	Override mask of writable bits in AArch64 ACTLR_EL3 register, or AArch32 secure ACTLR/ACTLR2 pair, if implemented
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: MPCORE

Chapter 9

Execution Modes

Mode	Code
User	16
FIQ	17
IRQ	18
Supervisor	19
Monitor	22
Abort	23
Hypervisor	26
Undefined	27
System	31

Table 9.1: Modes implemented in: CPU

Chapter 10

Exceptions

Exception	Code
Reset	0
Undefined	1
SupervisorCall	2
SecureMonitorCall	3
HypervisorCall	4
PrefetchAbort	5
DataAbort	6
HypervisorTrap	7
IRQ	8
FIQ	9
IllegalState	10

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: MPCORE

This level in the model hierarchy has 2 commands.
This level in the model hierarchy has no register groups.
This level in the model hierarchy has one child:
CPU0

11.2 Level 2: CPU

This level in the model hierarchy has 5 commands.
This level in the model hierarchy has 25 register groups:

Group name	Registers
Core	16
Control	3
User	7
FIQ	8
IRQ	3
Supervisor	3
Monitor	3
Hypervisor	3
Undefined	3
Abort	3
SIMD_VFP	32
SIMD_VFP_SYS	6

Coprocessor_32_bit	275
Coprocessor_32_bit_secure	43
Coprocessor_32_bit_non_secure	43
Coprocessor_64_bit	21
Coprocessor_64_bit_secure	7
Coprocessor_64_bit_non_secure	7
Integration_support	32
MPCore_distributor	178
MPCore_physical_redistributor	43
MPCore_processor_interface	15
MPCore_virtual_interface_control	11
MPCore_virtual_processor_interface	30
MPCore_ITS	14

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: MPCORE

12.1.1 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 12.1: isync command arguments

12.1.2 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 12.2: itrace command arguments

12.2 Level 2: CPU

12.2.1 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 12.3: debugflags command arguments

12.2.2 dumpTLB

report TLB contents

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

Table 12.4: dumpTLB command arguments

12.2.3 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 12.5: isync command arguments

12.2.4 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
-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 12.6: itrace command arguments

12.2.5 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 12.7: validateTLB command arguments

Chapter 13

Registers

13.1 Level 1: MPCORE

No registers.

13.2 Level 2: CPU

13.2.1 Core

Registers at level:2, 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 2, type:CPU group:Core

13.2.2 Control

Registers at level:2, 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 2, type:CPU group:Control

13.2.3 User

Registers at level:2, 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 2, type:CPU group:User

13.2.4 FIQ

Registers at level:2, 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 2, type:CPU group:FIQ

13.2.5 IRQ

Registers at level:2, 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 2, type:CPU group:IRQ

13.2.6 Supervisor

Registers at level:2, 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 2, type:CPU group:Supervisor

13.2.7 Monitor

Registers at level:2, 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 2, type:CPU group:Monitor

13.2.8 Hypervisor

Registers at level:2, type:CPU group:Hypervisor

Name	Bits	Initial-Hex	RW	Description
sp_hyp	32	0	rw	
elr_hyp	32	0	rw	
spsr_hyp	32	0	rw	

Table 13.8: Registers at level 2, type:CPU group:Hypervisor

13.2.9 Undefined

Registers at level:2, 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 13.9: Registers at level 2, type:CPU group:Undefined

13.2.10 Abort

Registers at level:2, 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 13.10: Registers at level 2, type:CPU group:Abort

13.2.11 SIMD_VFP

Registers at level:2, 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 13.11: Registers at level 2, type:CPU group:SIMD_VFP

13.2.12 SIMD_VFP_SYS

Registers at level:2, type:CPU group:SIMD_VFP_SYS

Name	Bits	Initial-Hex	RW	Description
FPSID	32	41033092	r-	floating-point system ID
FPSCR	32	0	rw	floating-point status/control
FPEXC	32	700	rw	floating-point exception
MVFR0	32	10110222	r-	Media/VFP feature 0
MVFR1	32	12111111	r-	Media/VFP feature 1
MVFR2	32	43	r-	Media/VFP feature 2

Table 13.12: Registers at level 2, type:CPU group:SIMD_VFP_SYS

13.2.13 Coprocessor_32_bit

Registers at level:2, type:CPU group:Coprocessor_32_bit

Name	Bits	Initial-Hex	RW	Description
ACTLR	32	0	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
AMAIR0	32	0	rw	Auxilary Memory Attribute Indirection 0
AMAIR1	32	0	rw	Auxilary Memory Attribute Indirection 1
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
ATS1HR	32	-	-w	Address Translate Stage 1 Hyp Mode Read

ATS1HW	32	-	-w	Address Translate Stage 1 Hyp Mode 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
BPIALLIS	32	-	-w	Branch Predictor Invalidate All (IS)
BPIMVA	32	-	-w	Branch Predictor Invalidate by VA
CBAR	32	13080000	r-	Configuration Base Address
CCSIDR	32	701fe00a	r-	Cache Size ID
CDBGDCD	32	-	-w	Data Cache Data Read
CDBGDCT	32	-	-w	Data Cache Tag Read
CDBGDR0	32	0	r-	Data Register 0
CDBGDR1	32	0	r-	Data Register 1
CDBGDR2	32	0	r-	Data Register 2
CDBGDR3	32	0	r-	Data Register 3
CDBGICD	32	-	-w	Instruction Cache Data Read
CDBGICT	32	-	-w	Instruction Cache Tag Read
CDBGTD	32	-	-w	TLB Data Read
CLIDR	32	a200023	r-	Cache Level ID
CNTFRQ	32	4c4b40	rw	Counter Frequency
CNTHCTL	32	3	rw	Timer EL2 Control
CNTHP_CTL	32	0	rw	Counter-Timer Hyp Physical Timer Control
CNTHP_TVAL	32	0	rw	Counter-Timer Hyp Physical Timer TimerValue
CNTKCTL	32	0	rw	Timer EL1 Control
CNTP_CTL	32	0	rw	Counter-Timer Physical Timer Control
CNTP_TVAL	32	0	rw	Counter-Timer Physical Timer TimerValue
CNTV_CTL	32	0	rw	Counter-Timer Virtual Timer Control
CNTV_TVAL	32	0	rw	Counter-Timer Virtual Timer TimerValue
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
CPACR	32	0	rw	Coprocessor Access Control
CSSELR	32	0	rw	Cache Size Selection
CTR	32	84448004	r-	Cache Type
DACR	32	0	rw	Domain Access Control
DBGAUTHSTATUS	32	aa	r-	Debug Authentication Status
DBGBCR0	32	0	rw	Debug Breakpoint Control 0
DBGBCR1	32	0	rw	Debug Breakpoint Control 1
DBGBCR2	32	0	rw	Debug Breakpoint Control 2
DBGBCR3	32	0	rw	Debug Breakpoint Control 3
DBGBCR4	32	0	rw	Debug Breakpoint Control 4
DBGBCR5	32	0	rw	Debug Breakpoint Control 5
DBGBVR0	32	0	rw	Debug Breakpoint Value 0
DBGBVR1	32	0	rw	Debug Breakpoint Value 1
DBGBVR2	32	0	rw	Debug Breakpoint Value 2
DBGBVR3	32	0	rw	Debug Breakpoint Value 3
DBGBVR4	32	0	rw	Debug Breakpoint Value 4
DBGBVR5	32	0	rw	Debug Breakpoint Value 5
DBGBXVR0	32	0	rw	Debug Breakpoint Extended Value 0
DBGBXVR1	32	0	rw	Debug Breakpoint Extended Value 1
DBGBXVR2	32	0	rw	Debug Breakpoint Extended Value 2
DBGBXVR3	32	0	rw	Debug Breakpoint Extended Value 3

DBGBXVR4	32	0	rw	Debug Breakpoint Extended Value 4
DBGBXVR5	32	0	rw	Debug Breakpoint Extended Value 5
DBGCLAIMCLR	32	0	rw	Debug Claim Tag Clear
DBGCLAIMSET	32	0	rw	Debug Claim Tag Set
DBGDCCINT	32	0	rw	DCC Interrupt Enable
DBGDEVID	32	110f13	r-	Debug Device ID
DBGDEVID1	32	2	r-	Debug Device ID 1
DBGDEVID2	32	0	r-	Debug Device ID 2
DBGDIDR	32	3516d000	r-	Debug ID
DBGDRAR	32	0	r-	Debug ROM Address (32-bit)
DBGDSAR	32	0	r-	Debug Self Address (32-bit)
DBGDSCRext	32	0	rw	Debug Status and Control
DBGDSCRint	32	0	r-	Debug Status and Control, Internal View
DBGDTRRXext	32	0	rw	Debug Data Transfer, Receive, External View
DBGDTRTRXint	32	0	rw	Debug Data Transfer, Transmit/Receive
DBGDTRTXext	32	0	rw	Debug Data Transfer, Transmit, External View
DBGOSDLR	32	0	rw	Debug OS Double Lock
DBGOSECCR	32	0	rw	Debug OS Lock Exception Catch Control
DBGOSLAR	32	-	-w	Debug OS Lock Access
DBGOSLSR	32	a	r-	Debug OS Lock Status
DBGPRCR	32	0	rw	Debug Power Control
DBGVCR	32	0	rw	Debug Vector Catch
DBGWCR0	32	0	rw	Debug Watchpoint Control 0
DBGWCR1	32	0	rw	Debug Watchpoint Control 1
DBGWCR2	32	0	rw	Debug Watchpoint Control 2
DBGWCR3	32	0	rw	Debug Watchpoint Control 3
DBGWFAR	32	0	rw	Debug Watchpoint Fault Address
DBGWVR0	32	0	rw	Debug Watchpoint Value 0
DBGWVR1	32	0	rw	Debug Watchpoint Value 1
DBGWVR2	32	0	rw	Debug Watchpoint Value 2
DBGWVR3	32	0	rw	Debug Watchpoint Value 3
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
DLR	32	0	rw	Debug Link
DSPSR	32	0	rw	Debug Saved Program Status
DTLBIALL	32	-	-w	Invalidate Entire Data TLB
DTLBIASID	32	-	-w	Invalidate Data TLB by ASID
DTLBIMVA	32	-	-w	Invalidate Data TLB by VA
HACR	32	0	rw	Hyp Auxiliary Configuration
HACTLR	32	0	rw	Hyp Auxiliary Control
HADFSR	32	0	rw	Hyp Auxiliary Data Fault Status
HAIFSR	32	0	rw	Hyp Auxiliary Instruction Fault Status
HAMAIR0	32	0	rw	Hyp Auxiliary Memory Attribute Indirection 0
HAMAIR1	32	0	rw	Hyp Auxiliary Memory Attribute Indirection 1
HCPTR	32	33ff	rw	Hyp Coprocessor Trap
HCR	32	0	rw	Hyp Configuration
HCR2	32	0	rw	Hyp Configuration 2
HDCR	32	6	rw	Hyp Debug Configuration
HDFAR	32	0	rw	Hyp Data Fault Address

HIFAR	32	0	rw	Hyp Instruction Fault Address
HMAIR0	32	0	rw	Hyp Memory Attribute Indirection 0
HMAIR1	32	0	rw	Hyp Memory Attribute Indirection 1
HPFAR	32	0	rw	Hyp IPA Fault Address
HSCTLR	32	30c50838	rw	Hyp System Control
HSR	32	0	rw	Hyp Syndrome
HSTR	32	0	rw	Hyp System Trap
HTCR	32	80800000	rw	Hyp Translation Control
HTPIDR	32	0	rw	Hyp Thread and Process ID
HVBAR	32	0	rw	Hyp Vector Base Address
ICC_AP0R0	32	0	rw	Interrupt Controller Active Priorities Group 0, 0
ICC_AP1R0	32	0	rw	Interrupt Controller Active Priorities Group 1, 0
ICC_BPR0	32	2	rw	Interrupt Controller Binary Point 0
ICC_BPR1	32	2	rw	Interrupt Controller Binary Point 1
ICC_CTLR	32	400	rw	Interrupt Controller Control
ICC_DIR	32	-	-w	Interrupt Controller Deactivate Interrupt
ICC_EOIR0	32	-	-w	Interrupt Controller End of Interrupt 0
ICC_EOIR1	32	-	-w	Interrupt Controller End of Interrupt 1
ICC_HPPIR0	32	3ff	r-	Interrupt Controller Highest Priority Pending Interrupt 0
ICC_HPPIR1	32	3ff	r-	Interrupt Controller Highest Priority Pending Interrupt 1
ICC_HSRE	32	0	rw	Interrupt Controller Hyp System Register Enable
ICC_IAR0	32	3ff	r-	Interrupt Controller Interrupt Acknowledge 0
ICC_IAR1	32	3ff	r-	Interrupt Controller Interrupt Acknowledge 1
ICC_IGRPEN0	32	0	rw	Interrupt Controller Interrupt Group 0 Enable
ICC_IGRPEN1	32	0	rw	Interrupt Controller Interrupt Group 1 Enable
ICC_MCTLR	32	400	rw	Interrupt Controller Monitor Control
ICC_MGRPEN1	32	0	rw	Interrupt Controller Monitor Interrupt Group 1 Enable
ICC_MSRE	32	0	rw	Interrupt Controller Monitor System Register Enable
ICC_PMR	32	0	rw	Interrupt Controller Priority Mask
ICC_RPR	32	ff	r-	Interrupt Controller Running Priority
ICC_SRE	32	0	rw	Interrupt Controller System Register Enable
ICH_AP0R0	32	0	rw	Interrupt Controller Hyp Active Priorities Group 0 (Word 0)
ICH_AP1R0	32	0	rw	Interrupt Controller Hyp Active Priorities Group 1 (Word 0)
ICH_EISR	32	0	r-	Interrupt Controller End of Interrupt Status
ICH_ELRSR	32	f	r-	Interrupt Controller Empty List Register Status
ICH_HCR	32	0	rw	Interrupt Controller Hypervisor Control
ICH_LR0	32	0	rw	Interrupt Controller List 0 (Bits 31:0)
ICH_LR1	32	0	rw	Interrupt Controller List 1 (Bits 31:0)
ICH_LR2	32	0	rw	Interrupt Controller List 2 (Bits 31:0)
ICH_LR3	32	0	rw	Interrupt Controller List 3 (Bits 31:0)
ICH_LRC0	32	0	rw	Interrupt Controller List 0 (Bits 63:32)
ICH_LRC1	32	0	rw	Interrupt Controller List 1 (Bits 63:32)
ICH_LRC2	32	0	rw	Interrupt Controller List 2 (Bits 63:32)
ICH_LRC3	32	0	rw	Interrupt Controller List 3 (Bits 63:32)
ICH_MISR	32	0	r-	Interrupt Controller Maintenance Interrupt State
ICH_VMCR	32	4c0000	rw	Interrupt Controller Virtual Machine Control
ICH_VTR	32	90100003	r-	Interrupt Controller VGIC Type
ICIALLU	32	-	-w	Instruction Cache Invalidate All
ICIALLUIS	32	-	-w	Instruction Cache Invalidate All (IS)
ICIMVAU	32	-	-w	Instruction Cache Invalidate by VA
ID_AFR0	32	0	r-	Auxiliary Feature 0
ID_DFR0	32	3010066	r-	Debug Feature 0
ID_ISAR0	32	2101110	r-	Instruction Set Attribute 0
ID_ISAR1	32	13112111	r-	Instruction Set Attribute 1
ID_ISAR2	32	21232042	r-	Instruction Set Attribute 2
ID_ISAR3	32	1112131	r-	Instruction Set Attribute 3

ID_ISAR4	32	11142	r-	Instruction Set Attribute 4
ID_ISAR5	32	10001	r-	Instruction Set Attribute 5
ID_MMFR0	32	10201105	r-	Memory Model Feature 0
ID_MMFR1	32	40000000	r-	Memory Model Feature 1
ID_MMFR2	32	1260000	r-	Memory Model Feature 2
ID_MMFR3	32	2102211	r-	Memory Model Feature 3
ID_PFR0	32	131	r-	Processor Feature 0
ID_PFR1	32	10011011	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
JIDR	32	0	rw	Jazelle ID
JMCR	32	0	rw	Jazelle Main Configuration
JOSCR	32	0	rw	Jazelle OS Control
L2ACTLR	32	0	rw	L2 Auxiliary Control
L2CTLR	32	0	rw	L2 Control
L2ECTLR	32	0	rw	L2 Extended Control
MAIR0	32	98aa4	rw	Memory Attribute Indirection 0
MAIR1	32	44e048e0	rw	Memory Attribute Indirection 1
MIDR	32	410fd011	r-	Main ID
MPIDR	32	80000000	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
PMCCFILTR	32	0	rw	Performance Monitors Cycle Count Filter
PMCCNTR	32	0	rw	Performance Monitors Cycle Count
PMCEID0	32	6fffbfff	r-	Performance Monitors Common Event ID 0
PMCEID1	32	0	r-	Performance Monitors Common Event ID 1
PMCNTENCLR	32	0	rw	Performance Monitors Count Enable Clear
PMCNTENSET	32	0	rw	Performance Monitors Count Enable Set
PMCR	32	41063000	rw	Performance Monitors Control
PMEVCNTR0	32	0	rw	Performance Monitors Event Count 0
PMEVCNTR1	32	0	rw	Performance Monitors Event Count 1
PMEVCNTR2	32	0	rw	Performance Monitors Event Count 2
PMEVCNTR3	32	0	rw	Performance Monitors Event Count 3
PMEVCNTR4	32	0	rw	Performance Monitors Event Count 4
PMEVCNTR5	32	0	rw	Performance Monitors Event Count 5
PMEVTYPER0	32	0	rw	Performance Monitors Event Type 0
PMEVTYPER1	32	0	rw	Performance Monitors Event Type 1
PMEVTYPER2	32	0	rw	Performance Monitors Event Type 2
PMEVTYPER3	32	0	rw	Performance Monitors Event Type 3
PMEVTYPER4	32	0	rw	Performance Monitors Event Type 4
PMEVTYPER5	32	0	rw	Performance Monitors Event Type 5
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
PMOVSSET	32	0	rw	Performance Monitors Overflow Flag Status Set
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
REVIDR	32	0	r-	Revision ID
SCR	32	0	rw	Secure Configuration
SCTLR	32	c50838	rw	System Control
SDCR	32	0	rw	Secure Debug Configuration
SDER	32	0	rw	Secure Debug Enable
TCMTR	32	0	r-	TCM Type
TLBIALL	32	-	-w	Invalidate Entire Unified TLB
TLBIALLH	32	-	-w	Invalidate Entire Hyp Unified TLB
TLBIALLHIS	32	-	-w	Invalidate Entire Hyp TLB (IS)
TLBIALLIS	32	-	-w	Invalidate Entire Unified TLB (IS)
TLBIALLNSNH	32	-	-w	Invalidate Entire Non-Secure Non-Hyp Unified TLB
TLBIALLNSNHIS	32	-	-w	Invalidate Entire Non-Secure Non-Hyp Unified TLB (IS)
TLBIASID	32	-	-w	Invalidate Unified TLB by ASID
TLBIASIDIS	32	-	-w	Invalidate Unified TLB by ASID (IS)
TLBIIPAS2	32	-	-w	Invalidate by IPA, Stage 2
TLBIIPAS2IS	32	-	-w	Invalidate by IPA, Stage 2 (IS)
TLBIIPAS2L	32	-	-w	Invalidate by IPA, Stage 2, Last level
TLBIIPAS2LIS	32	-	-w	Invalidate by IPA, Stage 2, Last level (IS)
TLBIMVA	32	-	-w	Invalidate Unified TLB by VA
TLBIMVAA	32	-	-w	Invalidate Unified TLB by VA, all ASID
TLBIMVAAIS	32	-	-w	Invalidate Unified TLB by VA, all ASID (IS)
TLBIMVAAL	32	-	-w	Invalidate Unified TLB by VA, all ASID, Last level
TLBIMVAALIS	32	-	-w	Invalidate Unified TLB by VA, all ASID, Last level (IS)
TLBIMVAH	32	-	-w	Invalidate Hyp Unified TLB by VA
TLBIMVAHIS	32	-	-w	Invalidate Hyp Unified TLB by VA (IS)
TLBIMVAIS	32	-	-w	Invalidate Unified TLB by VA (IS)
TLBIMVAL	32	-	-w	Invalidate Unified TLB by VA, Last level
TLBIMVALH	32	-	-w	Invalidate Hyp Unified TLB by VA, Last level
TLBIMVALHIS	32	-	-w	Invalidate Hyp Unified TLB by VA, Last level (IS)
TLBIMVALIS	32	-	-w	Invalidate Unified TLB by VA, Last level (IS)
TLBTR	32	0	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
VMPIDR	32	80000000	rw	Virtualization Multiprocessor ID
VPIDR	32	410fd011	rw	Virtualization Processor ID
VTCR	32	80000000	rw	Virtualization Translation Control

Table 13.13: Registers at level 2, type:CPU group:Coprocessor_32_bit

13.2.14 Coprocessor_32_bit_secure

Registers at level:2, type:CPU group:Coprocessor_32_bit_secure

Name	Bits	Initial-Hex	RW	Description
ACTLR_S	32	0	rw	Auxiliary Control
ADFSR_S	32	0	rw	Auxilary Data Fault Status
AIFSR_S	32	0	rw	Auxilary Instruction Fault Status
AMAIR0_S	32	0	rw	Auxilary Memory Attribute Indirection 0
AMAIR1_S	32	0	rw	Auxilary Memory Attribute Indirection 1
CNTP_CTL_S	32	0	rw	Counter-Timer Physical Timer Control

CNTP_TVAL_S	32	0	rw	Counter-Timer Physical Timer TimerValue
CONTEXTIDR_S	32	0	rw	Context ID
CSSELR_S	32	0	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
ICC_AP1R0_S	32	0	rw	Interrupt Controller Active Priorities Group 1, 0
ICC_BPR1_S	32	2	rw	Interrupt Controller Binary Point 1
ICC_CTLR_S	32	400	rw	Interrupt Controller Control
ICC_DIR_S	32	-	-w	Interrupt Controller Deactivate Interrupt
ICC_EOIR0_S	32	-	-w	Interrupt Controller End of Interrupt 0
ICC_EOIR1_S	32	-	-w	Interrupt Controller End of Interrupt 1
ICC_HPPIR0_S	32	3ff	r-	Interrupt Controller Highest Priority Pending Interrupt 0
ICC_HPPIR1_S	32	3ff	r-	Interrupt Controller Highest Priority Pending Interrupt 1
ICC_IAR0_S	32	3ff	r-	Interrupt Controller Interrupt Acknowledge 0
ICC_IAR1_S	32	3ff	r-	Interrupt Controller Interrupt Acknowledge 1
ICC_IGRPEN0_S	32	0	rw	Interrupt Controller Interrupt Group 0 Enable
ICC_IGRPEN1_S	32	0	rw	Interrupt Controller Interrupt Group 1 Enable
ICC_MGRPEN1_S	32	0	rw	Interrupt Controller Monitor Interrupt Group 1 Enable
ICC_PMR_S	32	0	rw	Interrupt Controller Priority Mask
ICC_RPR_S	32	ff	r-	Interrupt Controller Running Priority
ICC_SRE_S	32	0	rw	Interrupt Controller System Register Enable
IFAR_S	32	0	rw	Instruction Fault Address
IFSR_S	32	0	rw	Instruction Fault Status
MAIR0_S	32	98aa4	rw	Memory Attribute Indirection 0
MAIR1_S	32	44e048e0	rw	Memory Attribute Indirection 1
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	c50838	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 13.14: Registers at level 2, type:CPU group:Coprocessor_32_bit_secure

13.2.15 Coprocessor_32_bit_non_secure

Registers at level:2, type:CPU group:Coprocessor_32_bit_non_secure

Name	Bits	Initial-Hex	RW	Description
ACTLR_NS	32	0	rw	Auxiliary Control
ADFSR_NS	32	0	rw	Auxilary Data Fault Status
AIFSR_NS	32	0	rw	Auxilary Instruction Fault Status
AMAIR0_NS	32	0	rw	Auxilary Memory Attribute Indirection 0
AMAIR1_NS	32	0	rw	Auxilary Memory Attribute Indirection 1
CNTP_CTL_NS	32	0	rw	Counter-Timer Physical Timer Control
CNTP_TVAL_NS	32	0	rw	Counter-Timer Physical Timer TimerValue
CONTEXTIDR_NS	32	0	rw	Context ID
CSSELR_NS	32	0	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
ICC_AP1R0_NS	32	0	rw	Interrupt Controller Active Priorities Group 1, 0
ICC_BPR1_NS	32	3	rw	Interrupt Controller Binary Point 1
ICC_CTLR_NS	32	400	rw	Interrupt Controller Control
ICC_DIR_NS	32	-	-w	Interrupt Controller Deactivate Interrupt
ICC_EOIR0_NS	32	-	-w	Interrupt Controller End of Interrupt 0
ICC_EOIR1_NS	32	-	-w	Interrupt Controller End of Interrupt 1
ICC_HPPIR0_NS	32	3ff	r-	Interrupt Controller Highest Priority Pending Interrupt 0
ICC_HPPIR1_NS	32	3ff	r-	Interrupt Controller Highest Priority Pending Interrupt 1
ICC_IAR0_NS	32	3ff	r-	Interrupt Controller Interrupt Acknowledge 0
ICC_IAR1_NS	32	3ff	r-	Interrupt Controller Interrupt Acknowledge 1
ICC_IGRPEN0_NS	32	0	rw	Interrupt Controller Interrupt Group 0 Enable
ICC_IGRPEN1_NS	32	0	rw	Interrupt Controller Interrupt Group 1 Enable
ICC_MGRPEN1_NS	32	0	rw	Interrupt Controller Monitor Interrupt Group 1 Enable
ICC_PMR_NS	32	0	rw	Interrupt Controller Priority Mask
ICC_RPR_NS	32	ff	r-	Interrupt Controller Running Priority
ICC_SRE_NS	32	0	rw	Interrupt Controller System Register Enable
IFAR_NS	32	0	rw	Instruction Fault Address
IFSR_NS	32	0	rw	Instruction Fault Status
MAIR0_NS	32	98aa4	rw	Memory Attribute Indirection 0
MAIR1_NS	32	44e048e0	rw	Memory Attribute Indirection 1
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	c50838	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 13.15: Registers at level 2, type:CPU group:Coprocessor_32_bit_non_secure

13.2.16 Coprocessor_64_bit

Registers at level:2, type:CPU group:Coprocessor_64_bit

Name	Bits	Initial-Hex	RW	Description
CNTHP_CVAL	64	0	rw	Counter-Timer Hyp Physical Timer CompareValue
CNTPCT	64	0	r-	Counter-Timer Physical Count
CNTP_CVAL	64	0	rw	Counter-Timer Physical Timer CompareValue
CNTVCT	64	0	r-	Counter-Timer Virtual Count
CNTVOFF	64	0	rw	Virtual Offset
CNTV_CVAL	64	0	rw	Counter-Timer Virtual Timer CompareValue
CPUACTLR	64	0	rw	CPU Auxiliary Control
CPUECTLR	64	0	rw	CPU Extended Control
CPUMERRSR	64	0	rw	CPU Memory Error Syndrome
DBGDRAR64	64	0	r-	Debug ROM Address (64-bit)
DBGDSAR64	64	0	r-	Debug Self Address (64-bit)
HTTBR	64	0	rw	Hyp Translation Table Base
ICC_ASGI1R	64	-	-w	Interrupt Controller Alias SGI Group 1
ICC_SGI0R	64	-	-w	Interrupt Controller SGI Group 0
ICC_SGI1R	64	-	-w	Interrupt Controller SGI Group 1
L2MERRSR	64	0	rw	L2 Memory Error Syndrome

PARLPA	64	0	rw	Physical Address
PMCCNTR64	64	0	rw	Performance Monitors Cycle Count (64-bit)
TTBR0LPA	64	0	rw	Translation Table Base 0
TTBR1LPA	64	0	rw	Translation Table Base 1
VTTBR	64	0	rw	Virtualization Translation Table Base

Table 13.16: Registers at level 2, type:CPU group:Coprocessor_64_bit

13.2.17 Coprocessor_64_bit_secure

Registers at level:2, type:CPU group:Coprocessor_64_bit_secure

Name	Bits	Initial-Hex	RW	Description
CNTP_CVAL_S	64	0	rw	Counter-Timer Physical Timer CompareValue
ICC_ASGI1R_S	64	-	-w	Interrupt Controller Alias SGI Group 1
ICC_SGI0R_S	64	-	-w	Interrupt Controller SGI Group 0
ICC_SGI1R_S	64	-	-w	Interrupt Controller SGI Group 1
PARLPA_S	64	0	rw	Physical Address
TTBR0LPA_S	64	0	rw	Translation Table Base 0
TTBR1LPA_S	64	0	rw	Translation Table Base 1

Table 13.17: Registers at level 2, type:CPU group:Coprocessor_64_bit_secure

13.2.18 Coprocessor_64_bit_non_secure

Registers at level:2, type:CPU group:Coprocessor_64_bit_non_secure

Name	Bits	Initial-Hex	RW	Description
CNTP_CVAL_NS	64	0	rw	Counter-Timer Physical Timer CompareValue
ICC_ASGI1R_NS	64	-	-w	Interrupt Controller Alias SGI Group 1
ICC_SGI0R_NS	64	-	-w	Interrupt Controller SGI Group 0
ICC_SGI1R_NS	64	-	-w	Interrupt Controller SGI Group 1
PARLPA_NS	64	0	rw	Physical Address
TTBR0LPA_NS	64	0	rw	Translation Table Base 0
TTBR1LPA_NS	64	0	rw	Translation Table Base 1

Table 13.18: Registers at level 2, type:CPU group:Coprocessor_64_bit_non_secure

13.2.19 Integration_support

Registers at level:2, 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
PTWBankSelect	8	0	rw	select PTW bank (0 is stage 1, 1 is stage 2, 2-5 are stage 2 walks initiated by stage 1 level 0-3 entry lookups, respectively)
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_EL2	64	-	-w	perform EL2 page table walk for fetch, filling PTW query registers
PTWD_EL2	64	-	-w	perform EL2 page table walk for load/store, filling PTW query registers
PTWI_S2	64	-	-w	perform stage 2 page table walk for fetch, filling PTW query registers
PTWD_S2	64	-	-w	perform stage 2 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 13.19: Registers at level 2, type:CPU group:Integration_support

13.2.20 MPCore_distributor

Registers at level:2, type:CPU group:MPCore_distributor

Name	Bits	Initial-Hex	RW	Description
GICD_CIDR0	32	d	r-	Component ID 0
GICD_CIDR1	32	f0	r-	Component ID 1
GICD_CIDR2	32	5	r-	Component ID 2
GICD_CIDR3	32	b1	r-	Component ID 3
GICD_CLRSPI_NSR	32	0	-w	Clear SPI
GICD_CLRSPI_SR	32	0	-w	Clear SPI, Secure
GICD_CPENDSGIR0	32	0	rw	SGI Clear-Pending 0
GICD_CPENDSGIR1	32	0	rw	SGI Clear-Pending 1

GICD_CPENDSGIR2	32	0	rw	SGI Clear-Pending 2
GICD_CPENDSGIR3	32	0	rw	SGI Clear-Pending 3
GICD_CTLR	32	0	rw	Distributor Control
GICD_ICACTIVER0	32	0	rw	Interrupt Clear-Active 0
GICD_ICACTIVER1	32	0	rw	Interrupt Clear-Active 1
GICD_ICACTIVER2	32	0	rw	Interrupt Clear-Active 2
GICD_ICENABLER0	32	ffff	rw	Interrupt Clear-Enable 0
GICD_ICENABLER1	32	0	rw	Interrupt Clear-Enable 1
GICD_ICENABLER2	32	0	rw	Interrupt Clear-Enable 2
GICD_ICFGR0	32	aaaaaaaa	rw	Interrupt Configuration 0
GICD_ICFGR1	32	55540000	rw	Interrupt Configuration 1
GICD_ICFGR2	32	0	rw	Interrupt Configuration 2
GICD_ICFGR3	32	0	rw	Interrupt Configuration 3
GICD_ICFGR4	32	0	rw	Interrupt Configuration 4
GICD_ICFGR5	32	0	rw	Interrupt Configuration 5
GICD_ICPENDR0	32	0	rw	Interrupt Clear-Pending 0
GICD_ICPENDR1	32	0	rw	Interrupt Clear-Pending 1
GICD_ICPENDR2	32	0	rw	Interrupt Clear-Pending 2
GICD_IGROUPR0	32	0	rw	Interrupt Group 0
GICD_IGROUPR1	32	0	rw	Interrupt Group 1
GICD_IGROUPR2	32	0	rw	Interrupt Group 2
GICD_IGRPMODR0	32	0	rw	Interrupt Group Modifier 0
GICD_IGRPMODR1	32	0	rw	Interrupt Group Modifier 1
GICD_IGRPMODR2	32	0	rw	Interrupt Group Modifier 2
GICD_IIDR	32	102043b	r-	Distributor Implementor ID
GICD_IPRIORITYR0	32	0	rw	Interrupt Priority 0
GICD_IPRIORITYR1	32	0	rw	Interrupt Priority 1
GICD_IPRIORITYR2	32	0	rw	Interrupt Priority 2
GICD_IPRIORITYR3	32	0	rw	Interrupt Priority 3
GICD_IPRIORITYR4	32	0	rw	Interrupt Priority 4
GICD_IPRIORITYR5	32	0	rw	Interrupt Priority 5
GICD_IPRIORITYR6	32	0	rw	Interrupt Priority 6
GICD_IPRIORITYR7	32	0	rw	Interrupt Priority 7
GICD_IPRIORITYR8	32	0	rw	Interrupt Priority 8
GICD_IPRIORITYR9	32	0	rw	Interrupt Priority 9
GICD_IPRIORITYR10	32	0	rw	Interrupt Priority 10
GICD_IPRIORITYR11	32	0	rw	Interrupt Priority 11
GICD_IPRIORITYR12	32	0	rw	Interrupt Priority 12
GICD_IPRIORITYR13	32	0	rw	Interrupt Priority 13
GICD_IPRIORITYR14	32	0	rw	Interrupt Priority 14
GICD_IPRIORITYR15	32	0	rw	Interrupt Priority 15
GICD_IPRIORITYR16	32	0	rw	Interrupt Priority 16
GICD_IPRIORITYR17	32	0	rw	Interrupt Priority 17
GICD_IPRIORITYR18	32	0	rw	Interrupt Priority 18
GICD_IPRIORITYR19	32	0	rw	Interrupt Priority 19
GICD_IPRIORITYR20	32	0	rw	Interrupt Priority 20
GICD_IPRIORITYR21	32	0	rw	Interrupt Priority 21
GICD_IPRIORITYR22	32	0	rw	Interrupt Priority 22
GICD_IPRIORITYR23	32	0	rw	Interrupt Priority 23
GICD_IROUTER32	64	0	rw	Interrupt Routing 32
GICD_IROUTER33	64	0	rw	Interrupt Routing 33
GICD_IROUTER34	64	0	rw	Interrupt Routing 34
GICD_IROUTER35	64	0	rw	Interrupt Routing 35
GICD_IROUTER36	64	0	rw	Interrupt Routing 36
GICD_IROUTER37	64	0	rw	Interrupt Routing 37
GICD_IROUTER38	64	0	rw	Interrupt Routing 38

GICD_IROUTER39	64	0	rw	Interrupt Routing 39
GICD_IROUTER40	64	0	rw	Interrupt Routing 40
GICD_IROUTER41	64	0	rw	Interrupt Routing 41
GICD_IROUTER42	64	0	rw	Interrupt Routing 42
GICD_IROUTER43	64	0	rw	Interrupt Routing 43
GICD_IROUTER44	64	0	rw	Interrupt Routing 44
GICD_IROUTER45	64	0	rw	Interrupt Routing 45
GICD_IROUTER46	64	0	rw	Interrupt Routing 46
GICD_IROUTER47	64	0	rw	Interrupt Routing 47
GICD_IROUTER48	64	0	rw	Interrupt Routing 48
GICD_IROUTER49	64	0	rw	Interrupt Routing 49
GICD_IROUTER50	64	0	rw	Interrupt Routing 50
GICD_IROUTER51	64	0	rw	Interrupt Routing 51
GICD_IROUTER52	64	0	rw	Interrupt Routing 52
GICD_IROUTER53	64	0	rw	Interrupt Routing 53
GICD_IROUTER54	64	0	rw	Interrupt Routing 54
GICD_IROUTER55	64	0	rw	Interrupt Routing 55
GICD_IROUTER56	64	0	rw	Interrupt Routing 56
GICD_IROUTER57	64	0	rw	Interrupt Routing 57
GICD_IROUTER58	64	0	rw	Interrupt Routing 58
GICD_IROUTER59	64	0	rw	Interrupt Routing 59
GICD_IROUTER60	64	0	rw	Interrupt Routing 60
GICD_IROUTER61	64	0	rw	Interrupt Routing 61
GICD_IROUTER62	64	0	rw	Interrupt Routing 62
GICD_IROUTER63	64	0	rw	Interrupt Routing 63
GICD_IROUTER64	64	0	rw	Interrupt Routing 64
GICD_IROUTER65	64	0	rw	Interrupt Routing 65
GICD_IROUTER66	64	0	rw	Interrupt Routing 66
GICD_IROUTER67	64	0	rw	Interrupt Routing 67
GICD_IROUTER68	64	0	rw	Interrupt Routing 68
GICD_IROUTER69	64	0	rw	Interrupt Routing 69
GICD_IROUTER70	64	0	rw	Interrupt Routing 70
GICD_IROUTER71	64	0	rw	Interrupt Routing 71
GICD_IROUTER72	64	0	rw	Interrupt Routing 72
GICD_IROUTER73	64	0	rw	Interrupt Routing 73
GICD_IROUTER74	64	0	rw	Interrupt Routing 74
GICD_IROUTER75	64	0	rw	Interrupt Routing 75
GICD_IROUTER76	64	0	rw	Interrupt Routing 76
GICD_IROUTER77	64	0	rw	Interrupt Routing 77
GICD_IROUTER78	64	0	rw	Interrupt Routing 78
GICD_IROUTER79	64	0	rw	Interrupt Routing 79
GICD_IROUTER80	64	0	rw	Interrupt Routing 80
GICD_IROUTER81	64	0	rw	Interrupt Routing 81
GICD_IROUTER82	64	0	rw	Interrupt Routing 82
GICD_IROUTER83	64	0	rw	Interrupt Routing 83
GICD_IROUTER84	64	0	rw	Interrupt Routing 84
GICD_IROUTER85	64	0	rw	Interrupt Routing 85
GICD_IROUTER86	64	0	rw	Interrupt Routing 86
GICD_IROUTER87	64	0	rw	Interrupt Routing 87
GICD_IROUTER88	64	0	rw	Interrupt Routing 88
GICD_IROUTER89	64	0	rw	Interrupt Routing 89
GICD_IROUTER90	64	0	rw	Interrupt Routing 90
GICD_IROUTER91	64	0	rw	Interrupt Routing 91
GICD_IROUTER92	64	0	rw	Interrupt Routing 92
GICD_IROUTER93	64	0	rw	Interrupt Routing 93
GICD_IROUTER94	64	0	rw	Interrupt Routing 94

GICD_IROUTER95	64	0	rw	Interrupt Routing 95
GICD_ISACTIVER0	32	0	rw	Interrupt Set-Active 0
GICD_ISACTIVER1	32	0	rw	Interrupt Set-Active 1
GICD_ISACTIVER2	32	0	rw	Interrupt Set-Active 2
GICD_ISENABLER0	32	ffff	rw	Interrupt Set-Enable 0
GICD_ISENABLER1	32	0	rw	Interrupt Set-Enable 1
GICD_ISENABLER2	32	0	rw	Interrupt Set-Enable 2
GICD_ISPENDR0	32	0	rw	Interrupt Set-Pending 0
GICD_ISPENDR1	32	0	rw	Interrupt Set-Pending 1
GICD_ISPENDR2	32	0	rw	Interrupt Set-Pending 2
GICD_ITARGETSR0	32	0	rw	Interrupt Processor Targets 0
GICD_ITARGETSR1	32	0	rw	Interrupt Processor Targets 1
GICD_ITARGETSR2	32	0	rw	Interrupt Processor Targets 2
GICD_ITARGETSR3	32	0	rw	Interrupt Processor Targets 3
GICD_ITARGETSR4	32	0	rw	Interrupt Processor Targets 4
GICD_ITARGETSR5	32	0	rw	Interrupt Processor Targets 5
GICD_ITARGETSR6	32	0	rw	Interrupt Processor Targets 6
GICD_ITARGETSR7	32	0	rw	Interrupt Processor Targets 7
GICD_ITARGETSR8	32	0	rw	Interrupt Processor Targets 8
GICD_ITARGETSR9	32	0	rw	Interrupt Processor Targets 9
GICD_ITARGETSR10	32	0	rw	Interrupt Processor Targets 10
GICD_ITARGETSR11	32	0	rw	Interrupt Processor Targets 11
GICD_ITARGETSR12	32	0	rw	Interrupt Processor Targets 12
GICD_ITARGETSR13	32	0	rw	Interrupt Processor Targets 13
GICD_ITARGETSR14	32	0	rw	Interrupt Processor Targets 14
GICD_ITARGETSR15	32	0	rw	Interrupt Processor Targets 15
GICD_ITARGETSR16	32	0	rw	Interrupt Processor Targets 16
GICD_ITARGETSR17	32	0	rw	Interrupt Processor Targets 17
GICD_ITARGETSR18	32	0	rw	Interrupt Processor Targets 18
GICD_ITARGETSR19	32	0	rw	Interrupt Processor Targets 19
GICD_ITARGETSR20	32	0	rw	Interrupt Processor Targets 20
GICD_ITARGETSR21	32	0	rw	Interrupt Processor Targets 21
GICD_ITARGETSR22	32	0	rw	Interrupt Processor Targets 22
GICD_ITARGETSR23	32	0	rw	Interrupt Processor Targets 23
GICD_NSACR0	32	0	rw	Interrupt Non-secure Access Control 0
GICD_NSACR1	32	0	rw	Interrupt Non-secure Access Control 1
GICD_NSACR2	32	0	rw	Interrupt Non-secure Access Control 2
GICD_NSACR3	32	0	rw	Interrupt Non-secure Access Control 3
GICD_NSACR4	32	0	rw	Interrupt Non-secure Access Control 4
GICD_NSACR5	32	0	rw	Interrupt Non-secure Access Control 5
GICD_PIDR0	32	92	r-	Peripheral ID 0
GICD_PIDR1	32	b4	r-	Peripheral ID 1
GICD_PIDR2	32	3b	r-	Peripheral ID 2
GICD_PIDR3	32	0	r-	Peripheral ID 3
GICD_PIDR4	32	44	r-	Peripheral ID 4
GICD_PIDR5	32	0	r-	Peripheral ID 5
GICD_PIDR6	32	0	r-	Peripheral ID 6
GICD_PIDR7	32	0	r-	Peripheral ID 7
GICD_SETSPI_NSR	32	0	-w	Set SPI
GICD_SETSPI_SR	32	0	-w	Set SPI, Secure
GICD_SGIR	32	0	-w	Software-Generated Interrupt
GICD_SPENDSGIR0	32	0	rw	SGI Set-Pending 0
GICD_SPENDSGIR1	32	0	rw	SGI Set-Pending 1
GICD_SPENDSGIR2	32	0	rw	SGI Set-Pending 2
GICD_SPENDSGIR3	32	0	rw	SGI Set-Pending 3
GICD_SPISR0	32	0	r-	SPI Status 0

GICD_SPISR1	32	0	r-	SPI Status 1
GICD_TYPER	32	7b0402	r-	Interrupt Controller Type

Table 13.20: Registers at level 2, type:CPU group:MPCore_distributor

13.2.21 MPCore_physical_redistributor

Registers at level:2, type:CPU group:MPCore_physical_redistributor

Name	Bits	Initial-Hex	RW	Description
GICR_CIDR0	32	d	r-	Redistributor Component ID 0
GICR_CIDR1	32	f0	r-	Redistributor Component ID 1
GICR_CIDR2	32	5	r-	Redistributor Component ID 2
GICR_CIDR3	32	b1	r-	Redistributor Component ID 3
GICR_CLRLPIR	64	0	-w	Clear LPI Pending
GICR_CTLR	32	0	rw	Redistributor Control
GICR_ICACTIVER0	32	0	rw	Interrupt Clear-Active 0
GICR_ICENABLER0	32	ffff	rw	Interrupt Clear-Enable 0
GICR_ICFGR0	32	aaaaaaaa	rw	Interrupt Configuration 0
GICR_ICFGR1	32	55540000	rw	Interrupt Configuration 1
GICR_ICPENDR0	32	0	rw	Interrupt Clear-Pending 0
GICR_IGROUPR0	32	0	rw	Interrupt Group 0
GICR_IGRPMODR0	32	0	rw	Interrupt Group Modifier 0
GICR_IIDR	32	43b	r-	Redistributor Implementer Identification
GICR_INVALLR	64	0	-w	Redistributor Invalidate All
GICR_INVLPIR	64	0	-w	Redistributor Invalidate LPI
GICR_IPRIORITYR0	32	0	rw	Interrupt Priority 0
GICR_IPRIORITYR1	32	0	rw	Interrupt Priority 1
GICR_IPRIORITYR2	32	0	rw	Interrupt Priority 2
GICR_IPRIORITYR3	32	0	rw	Interrupt Priority 3
GICR_IPRIORITYR4	32	0	rw	Interrupt Priority 4
GICR_IPRIORITYR5	32	0	rw	Interrupt Priority 5
GICR_IPRIORITYR6	32	0	rw	Interrupt Priority 6
GICR_IPRIORITYR7	32	0	rw	Interrupt Priority 7
GICR_ISACTIVER0	32	0	rw	Interrupt Set-Active 0
GICR_ISENABLER0	32	ffff	rw	Interrupt Set-Enable 0
GICR_ISPENDR0	32	0	rw	Interrupt Set-Pending 0
GICR_NSACR	32	0	rw	Interrupt Non-secure Access Control
GICR_PENDBASER	64	0	rw	Redistributor LPI Pending Table Base Address
GICR_PIDR0	32	93	r-	Redistributor Peripheral ID 0
GICR_PIDR1	32	b4	r-	Redistributor Peripheral ID 1
GICR_PIDR2	32	3b	r-	Redistributor Peripheral ID 2
GICR_PIDR3	32	0	r-	Redistributor Peripheral ID 3
GICR_PIDR4	32	44	r-	Redistributor Peripheral ID 4
GICR_PIDR5	32	0	r-	Redistributor Peripheral ID 5
GICR_PIDR6	32	0	r-	Redistributor Peripheral ID 6
GICR_PIDR7	32	0	r-	Redistributor Peripheral ID 7
GICR_PROPBASER	64	0	rw	Redistributor Properties Base Address
GICR_SETLPIR	64	0	-w	Set LPI Pending
GICR_STATUSR	32	0	rw	Redistributor Status
GICR_SYNCR	32	0	r-	Redistributor Synchronize
GICR_TYPER	64	19	r-	Redistributor Type
GICR_WAKER	32	6	rw	Redistributor Wake

Table 13.21: Registers at level 2, type:CPU group:MPCore_physical_redistributor

13.2.22 MPCore_processor_interface

Registers at level:2, type:CPU group:MPCore_processor_interface

Name	Bits	Initial-Hex	RW	Description
GICC_ABPR	32	3	rw	Aliased Binary Point
GICC_AEOIR	32	0	-w	Aliased End of Interrupt
GICC_AHPPIR	32	3ff	r-	Aliased Highest Priority Pending Interrupt
GICC_AIAR	32	3ff	r-	Aliased Interrupt Acknowledge
GICC_APR0	32	0	rw	Active Priorities 0
GICC_BPR	32	2	rw	Binary Point
GICC_CTLR	32	0	rw	CPU Interface Control
GICC_DIR	32	0	-w	Deactivate Interrupt
GICC_EOIR	32	0	-w	End of Interrupt
GICC_HPPIR	32	3ff	r-	Highest Priority Pending Interrupt
GICC_IAR	32	3ff	r-	Interrupt Acknowledge
GICC_IIDR	32	14143b	r-	CPU Interface ID
GICC_NSAPR0	32	0	rw	Non-secure Active Priorities 0
GICC_PMR	32	0	rw	Interrupt Priority Mask
GICC_RPR	32	ff	r-	Running Priority

Table 13.22: Registers at level 2, type:CPU group:MPCore_processor_interface

13.2.23 MPCore_virtual_interface_control

Registers at level:2, type:CPU group:MPCore_virtual_interface_control

Name	Bits	Initial-Hex	RW	Description
GICH_APR0	32	0	rw	Active Priorities 0
GICH_EISR0	32	0	r-	End of Interrupt Status 0
GICH_ELRSR0	32	f	r-	Empty List Register Status 0
GICH_HCR	32	0	rw	Hypervisor Control
GICH_LR0	32	0	rw	List 0
GICH_LR1	32	0	rw	List 1
GICH_LR2	32	0	rw	List 2
GICH_LR3	32	0	rw	List 3
GICH_MISR	32	0	r-	Maintenance Interrupt Status
GICH_VMCR	32	4c0000	rw	Virtual Machine Control
GICH_VTR	32	90000003	r-	VGIC Type

Table 13.23: Registers at level 2, type:CPU group:MPCore_virtual_interface_control

13.2.24 MPCore_virtual_processor_interface

Registers at level:2, type:CPU group:MPCore_virtual_processor_interface

Name	Bits	Initial-Hex	RW	Description
GICV_ABPR	32	3	rw	VM Aliased Binary Point
GICV_AEOIR	32	0	-w	VM Aliased End of Interrupt
GICV_AHPPIR	32	3ff	r-	VM Aliased Highest Priority Pending Interrupt
GICV_AIAR	32	3ff	r-	VM Aliased Interrupt Acknowledge
GICV_APR0	32	0	rw	VM Active Priorities 0
GICV_BPR	32	2	rw	VM Binary Point
GICV_CTLR	32	0	rw	Virtual Machine Control
GICV_DIR	32	0	-w	VM Deactivate Interrupt
GICV_EOIR	32	0	-w	VM End of Interrupt

GICV_HPPIR	32	3ff	r-	VM Highest Priority Pending Interrupt
GICV_IAR	32	3ff	r-	VM Interrupt Acknowledge
GICV_IIDR	32	14143b	r-	VM CPU Interface ID
GICV_PMR	32	0	rw	VM Priority Mask
GICV_RPR	32	ff	r-	VM Running Priority
ICV_AP0R0_EL1	32	0	rw	VM Group 0 Active Priorities 0
ICV_AP1R0_EL1	32	0	rw	VM Group 1 Active Priorities 0
ICV_BPR0_EL1	32	2	rw	VM Group 0 Binary Point
ICV_BPR1_EL1	32	3	rw	VM Group 1 Binary Point
ICV_CTLR_EL1	32	400	rw	Virtual Machine Control
ICV_DIR_EL1	32	0	-w	VM Deactivate Interrupt
ICV_EOIR0_EL1	32	0	-w	VM Group 0 End of Interrupt
ICV_EOIR1_EL1	32	0	-w	VM Group 1 End of Interrupt
ICV_HPPIR0_EL1	32	3ff	r-	VM Group 0 Highest Priority Pending Interrupt
ICV_HPPIR1_EL1	32	3ff	r-	VM Group 1 Highest Priority Pending Interrupt
ICV_IAR0_EL1	32	3ff	r-	VM Group 0 Interrupt Acknowledge
ICV_IAR1_EL1	32	3ff	r-	VM Group 1 Interrupt Acknowledge
ICV_IGRPEN0_EL1	32	0	rw	VM Group 0 Interrupt Enable
ICV_IGRPEN1_EL1	32	0	rw	VM Group 1 Interrupt Enable
ICV_PMR_EL1	32	0	rw	VM Priority Mask
ICV_RPR_EL1	32	ff	r-	VM Running Priority

Table 13.24: Registers at level 2, type:CPU group:MPCore_virtual_processor_interface

13.2.25 MPCore_ITS

Registers at level:2, type:CPU group:MPCore_ITS

Name	Bits	Initial-Hex	RW	Description
GITS_BASER0	64	0	rw	ITS Translation Table Descriptor 0
GITS_BASER1	64	0	rw	ITS Translation Table Descriptor 1
GITS_BASER2	64	0	rw	ITS Translation Table Descriptor 2
GITS_BASER3	64	0	rw	ITS Translation Table Descriptor 3
GITS_BASER4	64	0	rw	ITS Translation Table Descriptor 4
GITS_BASER5	64	0	rw	ITS Translation Table Descriptor 5
GITS_BASER6	64	0	rw	ITS Translation Table Descriptor 6
GITS_BASER7	64	0	rw	ITS Translation Table Descriptor 7
GITS_CBASER	64	0	rw	ITS Command Queue Descriptor
GITS_CREADR	64	0	rw	ITS Read
GITS_CTLR	32	80000000	rw	ITS Control
GITS_CWRITER	64	0	rw	ITS Write
GITS_IIDR	32	43b	r-	ITS Identification
GITS_TYPER	64	9ef79	r-	ITS Type

Table 13.25: Registers at level 2, type:CPU group:MPCore_ITS