OVP Guide to Using Processor Models

Model specific information for ARM_MultiCluster

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_MultiCluster.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

This model implements an ARM system containing clusters of MPCore processors communicating using a common GICv2 or GICv3 block.

By default, the system contains Cortex-A53MPx4 and Cortex-A57MPx4 clusters, but this can be changed using parameter "override_clusterVariants". This parameter is a comma-separated list of cluster components (e.g. "Cortex-A53MPx4,Cortex-A57MPx4"). Note that if a GICv2 is selected, the total number of PEs must not exceed 8.

1.2 Licensing

This document describes the interface to the MultiCluster only. Refer to documentation of individual clusters for information regarding implemented features, licensing and limitations.

1.3 Limitations

1.4 Features

By default, the model implements a GICv2. Parameter enableGICv3 can be used to select a GICv3 instead.

1.5 Description

ARM Processor Model

1.6 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.

1.7 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.

1.8 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.

1.9 Features

1.9.1 Core Features

AArch64 is implemented at EL3, EL2, EL1 and EL0.

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

1.9.2 Memory System

Security extensions are implemented (also known as TrustZone). To make non-secure accesses visible externally, override ID_AA64MMFR0_EL1.PARange to specify the required physical bus size (32, 36, 40, 42, 44, 48 or 52 bits) and connect the processor to a bus one bit wider (33, 37, 41, 43, 45, 49 or 53 bits, respectively). The extra most-significant bit is the NS bit, indicating a

non-secure access. If non-secure accesses are not required to be made visible externally, connect the processor to a bus of exactly the size implied by ID_AA64MMFR0_EL1.PARange.

VMSA EL1, EL2 and EL3 stage 1 address translation is implemented. VMSA stage 2 address translation is implemented.

LPA (large physical address extension) is implemented as standard in ARMv8.

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 0

1.9.3 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)

1.9.4 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).

1.9.5 Generic Interrupt Controller

GIC block is implemented (GICv2, including security extensions). Accesses to GIC registers can be viewed externally by connecting to the 32-bit GICRegisters bus port. Secure register accesses are at offset 0x0 on this bus; for example, a secure access to GIC register GICD_CTLR can be observed by monitoring address 0x00001000. Non-secure accesses are at offset 0x80000000 on this bus; for example, a non-secure access to GIC register GICD_CTLR can be observed by monitoring address 0x80001000

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.

1.10 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.

1.11 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.

1.11.1 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.

1.11.2 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.

1.11.3 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).

1.11.4 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.

1.11.5 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.

1.11.6 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".

1.11.7 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".

1.12 Integration Support

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

1.12.1 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.

1.12.2 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.

1.12.3 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.

1.12.4 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).

1.12.5 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.

1.12.6 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

Bit 4: EL3 stage 1 TLB

1.12.7 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.

1.12.8 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.

1.12.9 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.

1.13 Description

ARM Processor Model

1.14 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.

1.15 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.

1.16 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.

1.17 Features

1.17.1 Core Features

AArch64 is implemented at EL3, EL2, EL1 and EL0.

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

1.17.2 Memory System

Security extensions are implemented (also known as TrustZone). To make non-secure accesses visible externally, override ID_AA64MMFR0_EL1.PARange to specify the required physical bus size (32, 36, 40, 42, 44, 48 or 52 bits) and connect the processor to a bus one bit wider (33, 37, 41, 43, 45, 49 or 53 bits, respectively). The extra most-significant bit is the NS bit, indicating a non-secure access. If non-secure accesses are not required to be made visible externally, connect the processor to a bus of exactly the size implied by ID_AA64MMFR0_EL1.PARange.

VMSA EL1, EL2 and EL3 stage 1 address translation is implemented. VMSA stage 2 address translation is implemented.

LPA (large physical address extension) is implemented as standard in ARMv8.

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 0

1.17.3 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)

1.17.4 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).

1.17.5 Generic Interrupt Controller

GIC block is implemented (GICv2, including security extensions). Accesses to GIC registers can be viewed externally by connecting to the 32-bit GICRegisters bus port. Secure register accesses are at offset 0x0 on this bus; for example, a secure access to GIC register GICD_CTLR can be observed by monitoring address 0x00001000. Non-secure accesses are at offset 0x80000000 on this bus; for example, a non-secure access to GIC register GICD_CTLR can be observed by monitoring address 0x80001000

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.

1.18 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.

1.19 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.

1.19.1 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.

1.19.2 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.

1.19.3 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).

1.19.4 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.

1.19.5 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.

1.19.6 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".

1.19.7 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".

1.20 Integration Support

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

1.20.1 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.

1.20.2 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.

1.20.3 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.

1.20.4 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).

1.20.5 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.

1.20.6 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

Bit 4: EL3 stage 1 TLB

1.20.7 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.

1.20.8 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.

1.20.9 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 Asymmetric Multicore Processor

This processor contains more than one core of differing architectures

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	
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	(described in this document)

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	53	mandatory	
DATA	32	53	optional	
GICRegisters	32	32	optional	GIC 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)
CFGSDISABLE	input	optional	Secure configuration lockdown (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_C0_0	input	optional	Private peripheral interrupt
PPI17_C0_0	input	optional	Private peripheral interrupt

PPI18_C0_0	input	optional	Private peripheral interrupt
PPI19_C0_0	input	optional	Private peripheral interrupt
PPI20_C0_0	input	optional	Private peripheral interrupt
PPI21_C0_0	input	optional	Private peripheral interrupt
PPI22_C0_0	input	optional	Private peripheral interrupt
PPI23_C0_0	input	optional	Private peripheral interrupt
PPI24_C0_0	input	optional	Private peripheral interrupt
PPI25_C0_0	input	optional	Private peripheral interrupt
PPI26_C0_0	input	optional	Private peripheral interrupt
PPI27_C0_0	input	optional	Private peripheral interrupt
PPI28_C0_0	input	optional	Private peripheral interrupt
PPI29_C0_0	input	optional	Private peripheral interrupt
PPI30_C0_0	input	optional	Private peripheral interrupt
PPI31_C0_0	input	optional	Private peripheral interrupt
CNTVIRQ_C0_0	output	optional	Virtual timer event (active high)
CNTPSIRQ_C0_0	output	optional	Secure physical timer event (active high)
CNTPNSIRQ_C0_0	output	optional	Non-secure physical timer event (active high)
CNTPHPIRQ_C0_0	output	optional	Hypervisor physical timer event (active high)
IRQOUT_C0_0	output	optional	IRQ wakeup
FIQOUT_C0_0	output	optional	FIQ wakeup
CLUSTERIDAFF1_C0	input	optional	Configure MPIDR.Aff1
CLUSTERIDAFF2_C0	input	optional	Configure MPIDR.Aff2
CLUSTERIDAFF3_C0	input	optional	Configure MPIDR.Aff3
RVBARADDRx_C0_0	input	optional	Configure AArch64 Reset Vector Base Address at reset
AA64nAA32_C0_0	input	optional	Register width state at reset
VINITHI_C0_0	input	optional	Configure HIVECS mode (SCTLR.V)
CFGEND_C0_0	input	optional	Configure exception endianness (SCTLR.EE)
CFGTE_C0_0	input	optional	Configure exception state at reset (SCTLR.TE)
reset_C0_0	input	optional	Processor reset, active high
fiq_C0_0	input	optional	FIQ interrupt, active high (negation of nFIQ)
irq_C0_0	input	optional	IRQ interrupt, active high (negation of nIRQ)
sei_C0_0	input	optional	System error interrupt, active on rising edge (negation of nSEI)
vfiq_C0_0	input	optional	Virtual FIQ interrupt, active high (negation of nVFIQ)
virq_C0_0	input	optional	Virtual IRQ interrupt, active high (negation of nVIRQ)

vsei_C0_0	input	optional	Virtual system error interrupt, active on rising edge (negation of nVSEI)
AXI_SLVERR_C0_0	input	optional	AXI external abort type (DECERR=0, SLVERR=1)
CP15SDISABLE_C0_0	input	optional	CP15SDISABLE (active high)
PMUIRQ_C0_0	output	optional	Performance monitor event (active high)
SMPEN_C0_0	output	optional	CPUECTLR.SMPEN current value
PPI16_C0_1	input	optional	Private peripheral interrupt
PPI17_C0_1	input	optional	Private peripheral interrupt
PPI18_C0_1	input	optional	Private peripheral interrupt
PPI19_C0_1	input	optional	Private peripheral interrupt
PPI20_C0_1	input	optional	Private peripheral interrupt
PPI21_C0_1	input	optional	Private peripheral interrupt
PPI22_C0_1	input	optional	Private peripheral interrupt
PPI23_C0_1	input	optional	Private peripheral interrupt
PPI24_C0_1	input	optional	Private peripheral interrupt
PPI25_C0_1	input	optional	Private peripheral interrupt
PPI26_C0_1	input	optional	Private peripheral interrupt
PPI27_C0_1	input	optional	Private peripheral interrupt
PPI28_C0_1	input	optional	Private peripheral interrupt
PPI29_C0_1	input	optional	Private peripheral interrupt
PPI30_C0_1	input	optional	Private peripheral interrupt
PPI31_C0_1	input	optional	Private peripheral interrupt
CNTVIRQ_C0_1	output	optional	Virtual timer event (active high)
CNTPSIRQ_C0_1	output	optional	Secure physical timer event (active high)
CNTPNSIRQ_C0_1	output	optional	Non-secure physical timer event (active high)
CNTPHPIRQ_C0_1	output	optional	Hypervisor physical timer event (active high)
IRQOUT_C0_1	output	optional	IRQ wakeup
FIQOUT_C0_1	output	optional	FIQ wakeup
RVBARADDRx_C0_1	input	optional	Configure AArch64 Reset Vector Base Address at reset
AA64nAA32_C0_1	input	optional	Register width state at reset
VINITHI_C0_1	input	optional	Configure HIVECS mode (SCTLR.V)
CFGEND_C0_1	input	optional	Configure exception endianness (SCTLR.EE)
CFGTE_C0_1	input	optional	Configure exception state at reset (SCTLR.TE)
reset_C0_1	input	optional	Processor reset, active high
fiq_C0_1	input	optional	FIQ interrupt, active high (negation of nFIQ)
irq_C0_1	input	optional	IRQ interrupt, active high (negation of nIRQ)

sei_C0_1	input	optional	System error interrupt, active on rising edge (negation of nSEI)
vfiq_C0_1	input	optional	Virtual FIQ interrupt, active high (negation of nVFIQ)
virq_C0_1	input	optional	Virtual IRQ interrupt, active high (negation of nVIRQ)
vsei_C0_1	input	optional	Virtual system error interrupt, active on rising edge (negation of nVSEI)
AXI_SLVERR_C0_1	input	optional	AXI external abort type (DECERR=0, SLVERR=1)
CP15SDISABLE_C0_1	input	optional	CP15SDISABLE (active high)
PMUIRQ_C0_1	output	optional	Performance monitor event (active high)
SMPEN_C0_1	output	optional	CPUECTLR.SMPEN current value
PPI16_C0_2	input	optional	Private peripheral interrupt
PPI17_C0_2	input	optional	Private peripheral interrupt
PPI18_C0_2	input	optional	Private peripheral interrupt
PPI19_C0_2	input	optional	Private peripheral interrupt
PPI20_C0_2	input	optional	Private peripheral interrupt
PPI21_C0_2	input	optional	Private peripheral interrupt
PPI22_C0_2	input	optional	Private peripheral interrupt
PPI23_C0_2	input	optional	Private peripheral interrupt
PPI24_C0_2	input	optional	Private peripheral interrupt
PPI25_C0_2	input	optional	Private peripheral interrupt
PPI26_C0_2	input	optional	Private peripheral interrupt
PPI27_C0_2	input	optional	Private peripheral interrupt
PPI28_C0_2	input	optional	Private peripheral interrupt
PPI29_C0_2	input	optional	Private peripheral interrupt
PPI30_C0_2	input	optional	Private peripheral interrupt
PPI31_C0_2	input	optional	Private peripheral interrupt
CNTVIRQ_C0_2	output	optional	Virtual timer event (active high)
CNTPSIRQ_C0_2	output	optional	Secure physical timer event (active high)
CNTPNSIRQ_C0_2	output	optional	Non-secure physical timer event (active high)
CNTPHPIRQ_C0_2	output	optional	Hypervisor physical timer event (active high)
IRQOUT_C0_2	output	optional	IRQ wakeup
FIQOUT_C0_2	output	optional	FIQ wakeup
RVBARADDRx_C0_2	input	optional	Configure AArch64 Reset Vector Base Address at reset
AA64nAA32_C0_2	input	optional	Register width state at reset
VINITHI_C0_2	input	optional	Configure HIVECS mode (SCTLR.V)
CFGEND_C0_2	input	optional	Configure exception endianness (SCTLR.EE)
CFGTE_C0_2	input	optional	Configure exception state at reset (SCTLR.TE)

reset_C0_2	input	optional	Processor reset, active high
fiq_C0_2	input	optional	FIQ interrupt, active high (negation of nFIQ)
irq_C0_2	input	optional	IRQ interrupt, active high (negation of nIRQ)
sei_C0_2	input	optional	System error interrupt, active on rising edge (negation of nSEI)
vfiq_C0_2	input	optional	Virtual FIQ interrupt, active high (negation of nVFIQ)
virq_C0_2	input	optional	Virtual IRQ interrupt, active high (negation of nVIRQ)
vsei_C0_2	input	optional	Virtual system error interrupt, active on rising edge (negation of nVSEI)
AXI_SLVERR_C0_2	input	optional	AXI external abort type (DECERR=0, SLVERR=1)
CP15SDISABLE_C0_2	input	optional	CP15SDISABLE (active high)
PMUIRQ_C0_2	output	optional	Performance monitor event (active high)
SMPEN_C0_2	output	optional	CPUECTLR.SMPEN current value
PPI16_C0_3	input	optional	Private peripheral interrupt
PPI17_C0_3	input	optional	Private peripheral interrupt
PPI18_C0_3	input	optional	Private peripheral interrupt
PPI19_C0_3	input	optional	Private peripheral interrupt
PPI20_C0_3	input	optional	Private peripheral interrupt
PPI21_C0_3	input	optional	Private peripheral interrupt
PPI22_C0_3	input	optional	Private peripheral interrupt
PPI23_C0_3	input	optional	Private peripheral interrupt
PPI24_C0_3	input	optional	Private peripheral interrupt
PPI25_C0_3	input	optional	Private peripheral interrupt
PPI26_C0_3	input	optional	Private peripheral interrupt
PPI27_C0_3	input	optional	Private peripheral interrupt
PPI28_C0_3	input	optional	Private peripheral interrupt
PPI29_C0_3	input	optional	Private peripheral interrupt
PPI30_C0_3	input	optional	Private peripheral interrupt
PPI31_C0_3	input	optional	Private peripheral interrupt
CNTVIRQ_C0_3	output	optional	Virtual timer event (active high)
CNTPSIRQ_C0_3	output	optional	Secure physical timer event (active high)
CNTPNSIRQ_C0_3	output	optional	Non-secure physical timer event (active high)
CNTPHPIRQ_C0_3	output	optional	Hypervisor physical timer event (active high)
IRQOUT_C0_3	output	optional	IRQ wakeup
FIQOUT_C0_3	output	optional	FIQ wakeup
RVBARADDRx_C0_3	input	optional	Configure AArch64 Reset Vector Base Address at reset
AA64nAA32_C0_3	input	optional	Register width state at reset

VINITHI_C0_3	input	optional	Configure HIVECS mode (SCTLR.V)
CFGEND_C0_3	input	optional	Configure exception endianness (SCTLR.EE)
CFGTE_C0_3	input	optional	Configure exception state at reset (SCTLR.TE)
reset_C0_3	input	optional	Processor reset, active high
fiq_C0_3	input	optional	FIQ interrupt, active high (negation of nFIQ)
irq_C0_3	input	optional	IRQ interrupt, active high (negation of nIRQ)
sei_C0_3	input	optional	System error interrupt, active on rising edge (negation of nSEI)
vfiq_C0_3	input	optional	Virtual FIQ interrupt, active high (negation of nVFIQ)
virq_C0_3	input	optional	Virtual IRQ interrupt, active high (negation of nVIRQ)
vsei_C0_3	input	optional	Virtual system error interrupt, active on rising edge (negation of nVSEI)
AXI_SLVERR_C0_3	input	optional	AXI external abort type (DECERR=0, SLVERR=1)
CP15SDISABLE_C0_3	input	optional	CP15SDISABLE (active high)
PMUIRQ_C0_3	output	optional	Performance monitor event (active high)
SMPEN_C0_3	output	optional	CPUECTLR.SMPEN current value
PPI16_C1_0	input	optional	Private peripheral interrupt
PPI17_C1_0	input	optional	Private peripheral interrupt
PPI18_C1_0	input	optional	Private peripheral interrupt
PPI19_C1_0	input	optional	Private peripheral interrupt
PPI20_C1_0	input	optional	Private peripheral interrupt
PPI21_C1_0	input	optional	Private peripheral interrupt
PPI22_C1_0	input	optional	Private peripheral interrupt
PPI23_C1_0	input	optional	Private peripheral interrupt
PPI24_C1_0	input	optional	Private peripheral interrupt
PPI25_C1_0	input	optional	Private peripheral interrupt
PPI26_C1_0	input	optional	Private peripheral interrupt
PPI27_C1_0	input	optional	Private peripheral interrupt
PPI28_C1_0	input	optional	Private peripheral interrupt
PPI29_C1_0	input	optional	Private peripheral interrupt
PPI30_C1_0	input	optional	Private peripheral interrupt
PPI31_C1_0	input	optional	Private peripheral interrupt
CNTVIRQ_C1_0	output	optional	Virtual timer event (active high)
CNTPSIRQ_C1_0	output	optional	Secure physical timer event (active high)
CNTPNSIRQ_C1_0	output	optional	Non-secure physical timer event (active high)
CNTPHPIRQ_C1_0	output	optional	Hypervisor physical timer event (active high)

IRQOUT_C1_0	output	optional	IRQ wakeup
FIQOUT_C1_0	output	optional	FIQ wakeup
CLUSTERIDAFF1_C1	input	optional	Configure MPIDR.Aff1
CLUSTERIDAFF2_C1	input	optional	Configure MPIDR.Aff2
CLUSTERIDAFF3_C1	input	optional	Configure MPIDR.Aff3
RVBARADDRx_C1_0	input	optional	Configure AArch64 Reset Vector Base Address at reset
AA64nAA32_C1_0	input	optional	Register width state at reset
VINITHI_C1_0	input	optional	Configure HIVECS mode (SCTLR.V)
CFGEND_C1_0	input	optional	Configure exception endianness (SCTLR.EE)
CFGTE_C1_0	input	optional	Configure exception state at reset (SCTLR.TE)
reset_C1_0	input	optional	Processor reset, active high
fiq_C1_0	input	optional	FIQ interrupt, active high (negation of nFIQ)
irq_C1_0	input	optional	IRQ interrupt, active high (negation of nIRQ)
sei_C1_0	input	optional	System error interrupt, active on rising edge (negation of nSEI)
vfiq_C1_0	input	optional	Virtual FIQ interrupt, active high (negation of nVFIQ)
virq_C1_0	input	optional	Virtual IRQ interrupt, active high (negation of nVIRQ)
vsei_C1_0	input	optional	Virtual system error interrupt, active on rising edge (negation of nVSEI)
AXI_SLVERR_C1_0	input	optional	AXI external abort type (DECERR=0, SLVERR=1)
CP15SDISABLE_C1_0	input	optional	CP15SDISABLE (active high)
PMUIRQ_C1_0	output	optional	Performance monitor event (active high)
SMPEN_C1_0	output	optional	CPUECTLR.SMPEN current value
PPI16_C1_1	input	optional	Private peripheral interrupt
PPI17_C1_1	input	optional	Private peripheral interrupt
PPI18_C1_1	input	optional	Private peripheral interrupt
PPI19_C1_1	input	optional	Private peripheral interrupt
PPI20_C1_1	input	optional	Private peripheral interrupt
PPI21_C1_1	input	optional	Private peripheral interrupt
PPI22_C1_1	input	optional	Private peripheral interrupt
PPI23_C1_1	input	optional	Private peripheral interrupt
PPI24_C1_1	input	optional	Private peripheral interrupt
PPI25_C1_1	input	optional	Private peripheral interrupt
PPI26_C1_1	input	optional	Private peripheral interrupt
PPI27_C1_1	input	optional	Private peripheral interrupt
PPI28_C1_1	input	optional	Private peripheral interrupt
PPI29_C1_1	input	optional	Private peripheral interrupt

PPI30_C1_1	input	optional	Private peripheral interrupt
PPI31_C1_1	input	optional	Private peripheral interrupt
CNTVIRQ_C1_1	output	optional	Virtual timer event (active high)
CNTPSIRQ_C1_1	output	optional	Secure physical timer event (active high)
CNTPNSIRQ_C1_1	output	optional	Non-secure physical timer event (active high)
CNTPHPIRQ_C1_1	output	optional	Hypervisor physical timer event (active high)
IRQOUT_C1_1	output	optional	IRQ wakeup
FIQOUT_C1_1	output	optional	FIQ wakeup
RVBARADDRx_C1_1	input	optional	Configure AArch64 Reset Vector Base Address at reset
AA64nAA32_C1_1	input	optional	Register width state at reset
VINITHI_C1_1	input	optional	Configure HIVECS mode (SCTLR.V)
CFGEND_C1_1	input	optional	Configure exception endianness (SCTLR.EE)
CFGTE_C1_1	input	optional	Configure exception state at reset (SCTLR.TE)
reset_C1_1	input	optional	Processor reset, active high
fiq_C1_1	input	optional	FIQ interrupt, active high (negation of nFIQ)
irq_C1_1	input	optional	IRQ interrupt, active high (negation of nIRQ)
sei_C1_1	input	optional	System error interrupt, active on rising edge (negation of nSEI)
vfiq_C1_1	input	optional	Virtual FIQ interrupt, active high (negation of nVFIQ)
virq_C1_1	input	optional	Virtual IRQ interrupt, active high (negation of nVIRQ)
vsei_C1_1	input	optional	Virtual system error interrupt, active on rising edge (negation of nVSEI)
AXI_SLVERR_C1_1	input	optional	AXI external abort type (DECERR=0, SLVERR=1)
CP15SDISABLE_C1_1	input	optional	CP15SDISABLE (active high)
PMUIRQ_C1_1	output	optional	Performance monitor event (active high)
SMPEN_C1_1	output	optional	CPUECTLR.SMPEN current value
PPI16_C1_2	input	optional	Private peripheral interrupt
PPI17_C1_2	input	optional	Private peripheral interrupt
PPI18_C1_2	input	optional	Private peripheral interrupt
PPI19_C1_2	input	optional	Private peripheral interrupt
PPI20_C1_2	input	optional	Private peripheral interrupt
PPI21_C1_2	input	optional	Private peripheral interrupt
PPI22_C1_2	input	optional	Private peripheral interrupt
PPI23_C1_2	input	optional	Private peripheral interrupt
PPI24_C1_2	input	optional	Private peripheral interrupt
PPI25_C1_2	input	optional	Private peripheral interrupt

PPI26_C1_2	input	optional	Private peripheral interrupt
PPI27_C1_2	input	optional	Private peripheral interrupt
PPI28_C1_2	input	optional	Private peripheral interrupt
PPI29_C1_2	input	optional	Private peripheral interrupt
PPI30_C1_2	input	optional	Private peripheral interrupt
PPI31_C1_2	input	optional	Private peripheral interrupt
CNTVIRQ_C1_2	output	optional	Virtual timer event (active high)
CNTPSIRQ_C1_2	output	optional	Secure physical timer event (active high)
CNTPNSIRQ_C1_2	output	optional	Non-secure physical timer event (active high)
CNTPHPIRQ_C1_2	output	optional	Hypervisor physical timer event (active high)
IRQOUT_C1_2	output	optional	IRQ wakeup
FIQOUT_C1_2	output	optional	FIQ wakeup
RVBARADDRx_C1_2	input	optional	Configure AArch64 Reset Vector Base Address at reset
AA64nAA32_C1_2	input	optional	Register width state at reset
VINITHI_C1_2	input	optional	Configure HIVECS mode (SCTLR.V)
CFGEND_C1_2	input	optional	Configure exception endianness (SCTLR.EE)
CFGTE_C1_2	input	optional	Configure exception state at reset (SCTLR.TE)
reset_C1_2	input	optional	Processor reset, active high
fiq_C1_2	input	optional	FIQ interrupt, active high (negation of nFIQ)
irq_C1_2	input	optional	IRQ interrupt, active high (negation of nIRQ)
sei_C1_2	input	optional	System error interrupt, active on rising edge (negation of nSEI)
vfiq_C1_2	input	optional	Virtual FIQ interrupt, active high (negation of nVFIQ)
virq_C1_2	input	optional	Virtual IRQ interrupt, active high (negation of nVIRQ)
vsei_C1_2	input	optional	Virtual system error interrupt, active on rising edge (negation of nVSEI)
AXI_SLVERR_C1_2	input	optional	AXI external abort type (DECERR=0, SLVERR=1)
CP15SDISABLE_C1_2	input	optional	CP15SDISABLE (active high)
PMUIRQ_C1_2	output	optional	Performance monitor event (active high)
SMPEN_C1_2	output	optional	CPUECTLR.SMPEN current value
PPI16_C1_3	input	optional	Private peripheral interrupt
PPI17_C1_3	input	optional	Private peripheral interrupt
PPI18_C1_3	input	optional	Private peripheral interrupt
PPI19_C1_3	input	optional	Private peripheral interrupt
PPI20_C1_3	input	optional	Private peripheral interrupt
PPI21_C1_3	input	optional	Private peripheral interrupt

PPI22_C1_3	input	optional	Private peripheral interrupt
PPI23_C1_3	input	optional	Private peripheral interrupt
PPI24_C1_3	input	optional	Private peripheral interrupt
PPI25_C1_3	input	optional	Private peripheral interrupt
PPI26_C1_3	input	optional	Private peripheral interrupt
PPI27_C1_3	input	optional	Private peripheral interrupt
PPI28_C1_3	input	optional	Private peripheral interrupt
PPI29_C1_3	input	optional	Private peripheral interrupt
PPI30_C1_3	input	optional	Private peripheral interrupt
PPI31_C1_3	input	optional	Private peripheral interrupt
CNTVIRQ_C1_3	output	optional	Virtual timer event (active high)
CNTPSIRQ_C1_3	output	optional	Secure physical timer event (active high)
CNTPNSIRQ_C1_3	output	optional	Non-secure physical timer event (active high)
CNTPHPIRQ_C1_3	output	optional	Hypervisor physical timer event (active high)
IRQOUT_C1_3	output	optional	IRQ wakeup
FIQOUT_C1_3	output	optional	FIQ wakeup
RVBARADDRx_C1_3	input	optional	Configure AArch64 Reset Vector Base Address at reset
AA64nAA32_C1_3	input	optional	Register width state at reset
VINITHI_C1_3	input	optional	Configure HIVECS mode (SCTLR.V)
CFGEND_C1_3	input	optional	Configure exception endianness (SCTLR.EE)
CFGTE_C1_3	input	optional	Configure exception state at reset (SCTLR.TE)
reset_C1_3	input	optional	Processor reset, active high
fiq_C1_3	input	optional	FIQ interrupt, active high (negation of nFIQ)
irq_C1_3	input	optional	IRQ interrupt, active high (negation of nIRQ)
sei_C1_3	input	optional	System error interrupt, active on rising edge (negation of nSEI)
vfiq_C1_3	input	optional	Virtual FIQ interrupt, active high (negation of nVFIQ)
virq_C1_3	input	optional	Virtual IRQ interrupt, active high (negation of nVIRQ)
vsei_C1_3	input	optional	Virtual system error interrupt, active on rising edge (negation of nVSEI)
AXI_SLVERR_C1_3	input	optional	AXI external abort type (DECERR=0, SLVERR=1)
CP15SDISABLE_C1_3	input	optional	CP15SDISABLE (active high)
PMUIRQ_C1_3	output	optional	Performance monitor event (active high)
SMPEN_C1_3	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)
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
distinctMTCores	Boolean	For multi-threaded (MT) processors, simulate threads as separate cores (otherwise, simulate MT threads as a single entity)
override_clusterVariants	String	Specifies a comma-separated list of cluster variant names in this multicluster
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_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)

Table 8.1: Parameters that can be set in: CLUSTER_GROUP

Chapter 9

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.

9.1 Level 1: CLUSTER_GROUP

9.1.1 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 9.1: isync command arguments

9.1.2 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 9.2: itrace command arguments

Chapter 10

Registers

10.1 Level 1: CLUSTER_GROUP

No registers.