

SyscallCapability Reference Manual

Generated by Doxygen 1.5.3

Wed Feb 20 09:59:13 2008

Contents

1	SyscallCapability Hierarchical Index	1
1.1	SyscallCapability Class Hierarchy	1
2	SyscallCapability Class Index	3
2.1	SyscallCapability Class List	3
3	SyscallCapability Class Documentation	5
3.1	Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping > Class Template Reference	5
3.2	memreq< INSTRUCTION, DATASIZE > Class Template Reference	8
3.3	memreq_dataless< INSTRUCTION > Class Template Reference	10
3.4	memreq_types Class Reference	12

Chapter 1

SyscallCapability Hierarchical Index

1.1 SyscallCapability Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping >	5
memreq_types	12
memreq_dataless< INSTRUCTION >	10
memreq< INSTRUCTION, DATASIZE >	8

Chapter 2

SyscallCapability Class Index

2.1 SyscallCapability Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping >	5
memreq< INSTRUCTION, DATASIZE > (Memory request class)	8
memreq_dataless< INSTRUCTION > (Data-less memory request class)	10
memreq_types (Non-templated type class for memreq defining the enums and their pretty printers)	12

Chapter 3

SyscallCapability Class Documentation

3.1 Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping > Class Template Reference

Public Member Functions

- [Bus](#) (const char *name)
Create a new [Bus](#) module.
- void [begin_of_cycle](#) ()
Process called at the beginning of the cycle.
- void [end_of_cycle](#) ()
Process called at the end of the cycle.
- void [on_data](#) ()
Process launched upon receiving a request/data from the CPU & memory, and setting the accept signals.
- void [on_shared](#) ()
Process launched upon receiving a shared flag from the CPU.
- void [on_shared_accept](#) ()
Process setting the shared bit enable signal.
- bool [has_pending_ops](#) ()
Module interface : Returns true if the module has some pending operations.

Public Attributes

- inclock [clock](#)
Clock port.
- inport< [memreq](#)< INSTRUCTION, RequestWidth > > [inCPU](#) [nCPU]

CLM Port receiving requests from CPUs or caches.

- `outport< memreq< INSTRUCTION, RequestWidth > > outCPU [nCPU]`

CLM Port sending answers to CPUs or caches.

- `inport< memreq< INSTRUCTION, RequestWidth > > inMEM`

CLM Port receiving answers from the memory.

- `outport< memreq< INSTRUCTION, RequestWidth > > outMEM`

CLM Port sending requests to the memory.

- `inport< bool, Snooping > inSharedCPU [nCPU]`

Optional CLM Port receiving the shared bits from the CPUs.

- `outport< bool, Snooping > outSharedMEM`

Optional CLM Port sending the shared bit to the memory.

- `ServiceImport< SVGmemreqInterface > svg`

Service port to the SVG service.

- `Queue< BufferLine, BufferSize > buffer`

Bufferized messages to be sent by the bus.

- `int round_robin_index`

Index for the round robin algorithm.

Classes

- struct **BufferLine**

3.1.1 Detailed Description

```
template<class INSTRUCTION, int nCPU, int BufferSize, int RequestWidth, bool Snooping = false> class Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping >
```

Definition at line 63 of file bus.sim.

3.1.2 Member Function Documentation

```
3.1.2.1 template<class INSTRUCTION, int nCPU, int BufferSize, int RequestWidth, bool Snooping = false> void Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping >::begin_of_cycle () [inline]
```

Process called at the beginning of the cycle.

Broadcast the first bufferized value.

Definition at line 144 of file bus.sim.

3.1 Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping > Class Template Reference

References Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping >::buffer, Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping >::outCPU, and Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping >::outMEM.

Referenced by Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping >::Bus().

3.1.2.2 `template<class INSTRUCTION, int nCPU, int BufferSize, int RequestWidth, bool Snooping = false> void Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping >::end_of_cycle () [inline]`

Process called at the end of the cycle.

Discard the first bufferized value if all outputs accepted it.

Bufferize new incoming values with higher priority to memory, then round robin between CPUs.

Definition at line 186 of file bus.sim.

References Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping >::buffer, Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping >::inCPU, Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping >::inMEM, and Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping >::svg.

Referenced by Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping >::Bus().

3.1.2.3 `template<class INSTRUCTION, int nCPU, int BufferSize, int RequestWidth, bool Snooping = false> void Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping >::on_shared () [inline]`

Process launched upon receiving a shared flag from the CPU.

Shared is set to true on every output, if it set to true to one of the input, and none of the input is nothing.

Definition at line 357 of file bus.sim.

References Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping >::buffer, Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping >::inSharedCPU, Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping >::outSharedMEM, and Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping >::svg.

Referenced by Bus< INSTRUCTION, nCPU, BufferSize, RequestWidth, Snooping >::Bus().

The documentation for this class was generated from the following file:

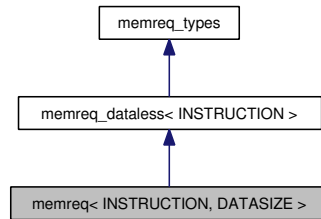
- bus.sim

3.2 memreq< INSTRUCTION, DATASIZE > Class Template Reference

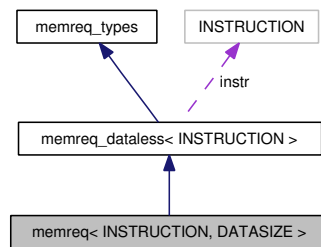
Memory request class.

```
#include <memreq.h>
```

Inheritance diagram for memreq< INSTRUCTION, DATASIZE >:



Collaboration diagram for memreq< INSTRUCTION, DATASIZE >:



Public Member Functions

- [memreq\(\)](#)
Creates a new memory request.
- `const char * Read\(\) const`
Read from the message data.
- `void Write(const char *buf, int num_bytes)`
Write to the message data.
- `uint32_t Read32\(\) const`
Returns the data of the [memreq](#) as a 32bit value.
- `void Write32(uint32_t val)`
Set the data of the [memreq](#) as a 32bit value.

Public Attributes

- `ByteArray< DATASIZE > data`

The data in the message.

Friends

- ostream & [operator<<](#) (ostream &os, const [memreq](#) &req)

Pretty printer for the [memreq](#) class.

3.2.1 Detailed Description

template<class INSTRUCTION, int DATASIZE> class memreq< INSTRUCTION, DATASIZE >

Memory request class.

Definition at line 229 of file memreq.h.

The documentation for this class was generated from the following file:

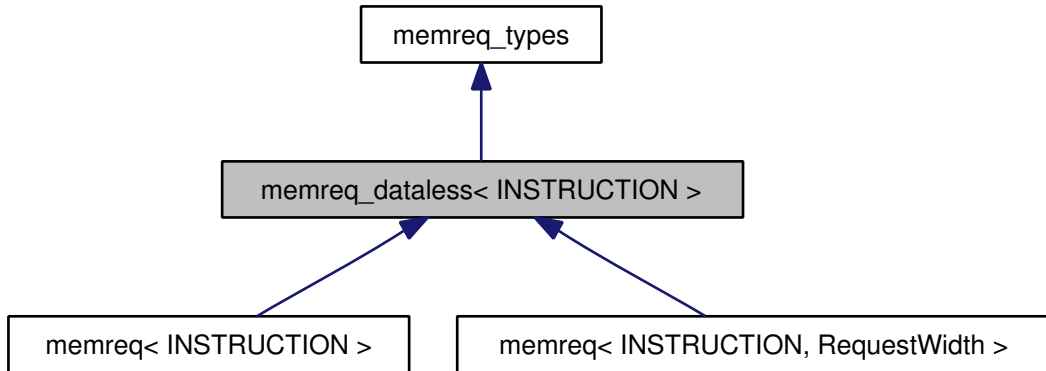
- /home/girbal/svn/unisim.org/public/components/CycleLevel/packages/memreq.h

3.3 memreq_dataless< INSTRUCTION > Class Template Reference

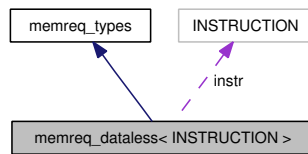
Data-less memory request class.

```
#include <memreq.h>
```

Inheritance diagram for memreq_dataless< INSTRUCTION >:



Collaboration diagram for memreq_dataless< INSTRUCTION >:



Public Member Functions

- `memreq_dataless ()`
Creates a new memory request.
- virtual const char * `Read () const`
Returns the data of the `memreq`.

Public Attributes

- `INSTRUCTION instr`
The incoming instruction.
- `uint32_t address`
Address of the memory access.
- `int size`
Size of the retrieved data in bytes.

- [command_t command](#)
Type of the memory request (READ, WRITE, ...).
- [uint64_t uid](#)
Unique ID fields used by multi-port caches.
- [uint64_t memreq_id](#)
Unique ID. Is kept the same for the corresponding answer.
- [sender_type_t sender_type](#)
Type of the sender (CPU, CACHE, MEMORY, ...).
- [message_type_t message_type](#)
Type of the message (Request, Answer).
- `module * sender`
Module that has sent this message.
- `module * req_sender`
Module that has sent the request this message is about.
- `bool cachable`
Whether the request address is cachable.

Friends

- `ostream & operator<< (ostream &os, const memreq_dataless &req)`
Pretty printer for the [memreq](#) class.

3.3.1 Detailed Description

`template<class INSTRUCTION> class memreq_dataless< INSTRUCTION >`

Data-less memory request class.

Definition at line 161 of file memreq.h.

The documentation for this class was generated from the following file:

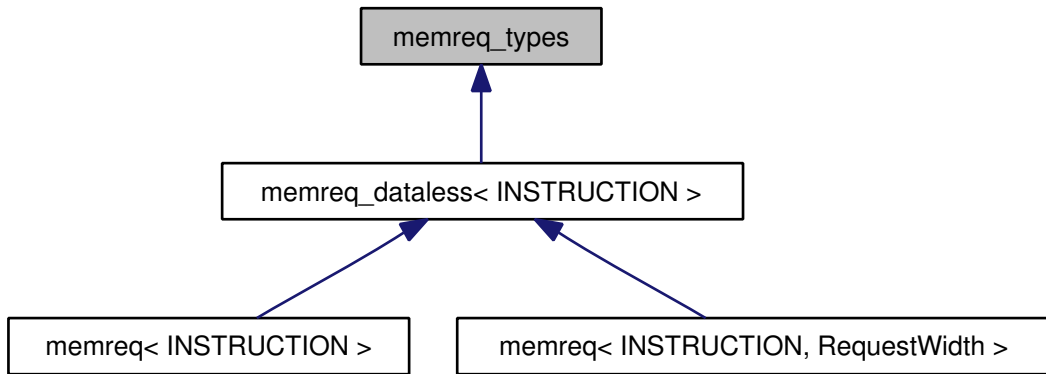
- `/home/girbal/svn/unisim.org/public/components/CycleLevel/packages/memreq.h`

3.4 memreq_types Class Reference

Non-templated type class for `memreq` defining the enums and their pretty printers.

```
#include <memreq.h>
```

Inheritance diagram for `memreq_types`:



Public Types

- enum `command_t` {
`cmd_UNKNOWN`, `cmd_READ`, `cmd_READX`, `cmd_WRITE`,
`cmd_PREFETCH`, `cmd_EVICT`, `cmd_FLUSH`, `cmd_BLOCK_INVALIDATE` }
Command of the memory request (READ, WRITE, ...).
- enum `sender_type_t` { `sender_UNKNOWN`, `sender_CPU`, `sender_CACHE`, `sender_MEM` }
Type of the module which has sent the request (CPU, CACHE, MEMORY).
- enum `message_type_t` { `type_UNKNOWN`, `type_REQUEST`, `type_ANSWER` }
Type of the message (REQUEST, ANSWER).

Static Public Attributes

- static `uint64_t` `memreq_id_max`
Maximum memreq_id used so far.

Friends

- `ostream` & `operator<<` (`ostream` &os, const `command_t` &c)
Pretty printer for the command_t enum.
- `ostream` & `operator<<` (`ostream` &os, const `sender_type_t` &t)
Pretty printer for the sender_type_t enum.

- ostream & operator<< (ostream &os, const message_type_t &m)
Pretty printer for the sender_type_t enum.

3.4.1 Detailed Description

Non-templated type class for memreq defining the enums and their pretty printers.
Definition at line 54 of file memreq.h.

3.4.2 Member Enumeration Documentation

3.4.2.1 enum memreq_types::command_t

Command of the memory request (READ, WRITE, ...).

Enumerator:

- cmd_UNKNOWN* Unset request type.
- cmd_READ* Read request / answer.
- cmd_READX* Exclusive read request / answer.
- cmd_WRITE* Write request.
- cmd_PREFETCH* Prefetch request / answer.
- cmd_EVICT* Evict request.
- cmd_FLUSH* Cache flush line request.
- cmd_BLOCK_INVALIDATE* Block invalidate request.

Definition at line 56 of file memreq.h.

3.4.2.2 enum memreq_types::sender_type_t

Type of the module which has sent the request (CPU, CACHE, MEMORY).

Enumerator:

- sender_UNKNOWN* Unset sender type.
- sender_CPU* Request / answer issued by a CPU.
- sender_CACHE* Request / answer issued by a CACHE.
- sender_MEM* Answer issued by the Memory.

Definition at line 66 of file memreq.h.

3.4.2.3 enum memreq_types::message_type_t

Type of the message (REQUEST, ANSWER).

Enumerator:

- type_UNKNOWN* Unset message type.

type_REQUEST Request message.

type_ANSWER Answer message.

Definition at line 72 of file memreq.h.

The documentation for this class was generated from the following file:

- </home/girbal/svn/unisim.org/public/components/CycleLevel/packages/memreq.h>

Index

- begin_of_cycle
 - Bus, 6
- Bus, 5
 - begin_of_cycle, 6
 - end_of_cycle, 7
 - on_shared, 7
- cmd_BLOCK_INVALIDATE
 - memreq_types, 13
- cmd_EVICT
 - memreq_types, 13
- cmd_FLUSH
 - memreq_types, 13
- cmd_PREFETCH
 - memreq_types, 13
- cmd_READ
 - memreq_types, 13
- cmd_READX
 - memreq_types, 13
- cmd_UNKNOWN
 - memreq_types, 13
- cmd_WRITE
 - memreq_types, 13
- command_t
 - memreq_types, 13
- end_of_cycle
 - Bus, 7
- memreq, 8
- memreq_dataless, 10
- memreq_types
 - cmd_BLOCK_INVALIDATE, 13
 - cmd_EVICT, 13
 - cmd_FLUSH, 13
 - cmd_PREFETCH, 13
 - cmd_READ, 13
 - cmd_READX, 13
 - cmd_UNKNOWN, 13
 - cmd_WRITE, 13
 - sender_CACHE, 13
 - sender_CPU, 13
 - sender_MEM, 13
 - sender_UNKNOWN, 13
 - type_ANSWER, 14
 - type_REQUEST, 13
 - type_UNKNOWN, 13
- memreq_types, 12
 - command_t, 13
 - message_type_t, 13
 - sender_type_t, 13
- message_type_t
 - memreq_types, 13
- on_shared
 - Bus, 7
- sender_CACHE
 - memreq_types, 13
- sender_CPU
 - memreq_types, 13
- sender_MEM
 - memreq_types, 13
- sender_type_t
 - memreq_types, 13
- sender_UNKNOWN
 - memreq_types, 13
- type_ANSWER
 - memreq_types, 14
- type_REQUEST
 - memreq_types, 13
- type_UNKNOWN
 - memreq_types, 13