

# DRAMModule Reference Manual

Generated by Doxygen 1.5.3

Wed Feb 20 10:43:11 2008



# Contents

<b>1</b>	<b>DRAMModule Hierarchical Index</b>	<b>1</b>
1.1	DRAMModule Class Hierarchy . . . . .	1
<b>2</b>	<b>DRAMModule Class Index</b>	<b>3</b>
2.1	DRAMModule Class List . . . . .	3
<b>3</b>	<b>DRAMModule Class Documentation</b>	<b>5</b>
3.1	DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE > Class Template Reference . . . . .	5
3.2	DRAMBank< INSTRUCTION, nCacheLineSize > Class Template Reference . . . . .	18
3.3	DRAMControlerQueueEntry< INSTRUCTION, nCacheLineSize > Class Template Reference . . . . .	20
3.4	MemoryContainer< VERBOSE > Class Template Reference . . . . .	23
3.5	MemoryPageTableEntry Class Reference . . . . .	25
3.6	memreq< INSTRUCTION, DATASIZE > Class Template Reference . . . . .	26
3.7	memreq_dataless< INSTRUCTION > Class Template Reference . . . . .	28
3.8	memreq_types Class Reference . . . . .	30



# Chapter 1

## DRAMModule Hierarchical Index

### 1.1 DRAMModule Class Hierarchy

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

DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >	5
DRAMBank< INSTRUCTION, nCacheLineSize >	18
DRAMControlerQueueEntry< INSTRUCTION, nCacheLineSize >	20
MemoryContainer< VERBOSE >	23
MemoryPageTableEntry	25
memreq_types	30
memreq_dataless< INSTRUCTION >	28
memreq< INSTRUCTION, DATASIZE >	26



# Chapter 2

## DRAMModule Class Index

### 2.1 DRAMModule Class List

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

<a href="#">DRAM&lt; INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE &gt;</a>	5
<a href="#">DRAMBank&lt; INSTRUCTION, nCacheLineSize &gt; (DRAM Bank)</a>	18
<a href="#">DRAMControlerQueueEntry&lt; INSTRUCTION, nCacheLineSize &gt; (DRAM Controler Queue Entry)</a>	20
<a href="#">MemoryContainer&lt; VERBOSE &gt; (MemoryContainer: memory content)</a>	23
<a href="#">MemoryPageTableEntry</a>	25
<a href="#">memreq&lt; INSTRUCTION, DATASIZE &gt; (Memory request class)</a>	26
<a href="#">memreq_dataless&lt; INSTRUCTION &gt; (Data-less memory request class)</a>	28
<a href="#">memreq_types (Non-templated type class for memreq defining the enums and their pretty printers)</a>	30



## Chapter 3

# DRAMModule Class Documentation

### 3.1 DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE > Class Template Reference

#### Public Member Functions

- [DRAM](#) (const char \*name)
- virtual bool [ClientIndependentSetup](#) ()  
*Service interface : initialization.*
- void [Reset](#) ()  
*Memory interface : reset.*
- bool [FlushMemory](#) (address\_t addr, uint32\_t size)  
*Memory interface : flush.*
- bool [ReadMemory](#) (address\_t addr, void \*buffer, uint32\_t size)  
*Memory interface : read from memory.*
- bool [WriteMemory](#) (address\_t addr, const void \*buffer, uint32\_t size)  
*Memory interface : write to memory.*
- bool [Activate](#) (int bank, int row, [DRAMControlerQueueEntry](#)< INSTRUCTION, nCacheLineSize > \*entry)  
*Activate a bank (open a page).*
- bool [Precharge](#) (int bank, [DRAMControlerQueueEntry](#)< INSTRUCTION, nCacheLineSize > \*entry)  
*Precharge a bank (close a page).*
- bool [PrechargeAll](#) ()
- bool [Read](#) (int bank, int col, [DRAMControlerQueueEntry](#)< INSTRUCTION, nCacheLineSize > \*entry)

*Do a read command.*

- bool [Write](#) (int bank, int col, [DRAMControlerQueueEntry](#)< INSTRUCTION, nCacheLineSize > \*entry)

*Do a write command.*

- bool [ReadData](#) (int bank, [DRAMControlerQueueEntry](#)< INSTRUCTION, nCacheLineSize > \*entry)

*Do a read data transfer from the DRAM.*

- bool [WriteData](#) (int bank, [DRAMControlerQueueEntry](#)< INSTRUCTION, nCacheLineSize > \*entry)

*Do a write data transfer from the DRAM.*

- bool [Refresh](#) (int row)

*Performs a refresh of a row on all banks.*

- bool [DRAMDataTransfer](#) ()

*Performs data transfer between memory controler and the DRAM.*

- void [DRAMControl](#) ()

- void [ScheduleDRAMQueue](#) ()

*Perform DRAM Queue pipeline.*

- void [onInData](#) ()

- void [onOutAccept](#) ()

*Process lauched upon receiving an accept for the output port.*

- void [start\\_of\\_cycle](#) ()

*Process launched at the beggining of the cycle.*

- void [end\\_of\\_cycle](#) ()

*Process called at the end of the cycle.*

- void [Read](#) (void \*buffer, address\_t address, int size)

- void [Write](#) (address\_t address, const void \*buffer, int size)

- void [Set](#) (address\_t address, uint8\_t data, int size)

- void [DumpStats](#) (ostream &os, uint64\_t sim\_total\_time, uint64\_t sim\_cycle)

*Dumps some statistics into an output stream.*

- bool [Check](#) ()

*perform a check on the controle entries*

- void [WarmRestart](#) ()

*Perform a warm restart of the memory.*

- void [ResetStats](#) ()

*Reset the statistic counters.*

- void [load\\_checkpoint](#) (istream &is)

*Loads the par corresponding to the memory from a checkpoint file.*

- bool [has\\_pending\\_ops](#) ()  
*Module interface : Returns true if the module has some pending operations.*

## Public Attributes

- MI\_ServiceExport [syscall\\_MemExp](#)  
*Service port from CPU.*
- ServiceImport< SVGmemreqInterface > [svg](#)  
*Service port to SVG builder.*
- inclock [inClock](#)  
*clock port*
- inport< [memreq](#)< INSTRUCTION, nDataPathSize > > [in](#)  
*Input port for memory requests.*
- outport< [memreq](#)< INSTRUCTION, nDataPathSize > > [out](#)  
*Output port for sending data toward CPU.*
- inport< bool, Snooping > [inShared](#)  
*Inport port for the shared bit of some CMP models.*
- uint64\_t [average\\_latency](#)  
*average latency statistic*
- uint64\_t [total\\_precharge](#)  
*total number of precharge statistic*
- uint64\_t [total\\_activate](#)  
*total number of activate statistic*
- uint64\_t [total\\_read](#)  
*total number of read statistic*
- uint64\_t [total\\_write](#)  
*total number of write statistic*
- uint64\_t [total\\_refresh](#)  
*total number of refresh statistic*
- uint64\_t [num\\_read\\_requests](#)  
*total number of read request statistic*
- uint64\_t [cumulative\\_load\\_time](#)  
*total cumulative load time statistic*

- `uint64_t reordered_requests`  
*total number of reordered request statistic*
- `uint64_t min_load_time`  
*minimum load time statistic*
- `uint64_t max_load_time`  
*maximum load time statistic*
- `DRAMBank< INSTRUCTION, nCacheLineSize > banks [nBanks]`  
*Memory banks.*
- `MyMemEmulator * emulator_mem`  
*Validation memory.*

## Friends

- `ostream & operator<<` (`ostream &os, const DRAM &dram`)

### 3.1.1 Detailed Description

`template<class INSTRUCTION, int nBanks, int nRows, int nCols, int TRRD, int TRAS, int TRCD, int CL, int TRP, int TRC, uint64_t TREF, int nDataPathSize, int nCacheLineSize, int nCtrlQueueSize, int nProg = 0, bool Snooping = false, bool VERBOSE = false> class DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >`

A `DRAM` memory model including a basic memory controller (SDRAM or DDR-SDRAM)

Definition at line 96 of file `dram.sim`.

### 3.1.2 Constructor & Destructor Documentation

**3.1.2.1** `template<class INSTRUCTION, int nBanks, int nRows, int nCols, int TRRD, int TRAS, int TRCD, int CL, int TRP, int TRC, uint64_t TREF, int nDataPathSize, int nCacheLineSize, int nCtrlQueueSize, int nProg = 0, bool Snooping = false, bool VERBOSE = false> DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::DRAM (const char * name) [inline]`

The constructor

#### Parameters:

*name* the module name

Definition at line 116 of file `dram.sim`.

References `DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::average_latency`, `DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF,`

### 3.1 DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE > Class Template Reference 9

nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::cumulative\_load\_time, DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::emulator\_mem, DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::end\_of\_cycle(), DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::in, DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::inClock, DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::inShared, DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::max\_load\_time, DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::min\_load\_time, DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::num\_read\_requests, DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::onInData(), DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::onOutAccept(), DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::out, DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::reordered\_requests, DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::start\_of\_cycle(), DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::total\_activate, DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::total\_precharge, DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::total\_read, DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::total\_refresh, and DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::total\_write.

### 3.1.3 Member Function Documentation

**3.1.3.1** `template<class INSTRUCTION, int nBanks, int nRows, int nCols, int TRRD, int TRAS, int TRCD, int CL, int TRP, int TRC, uint64_t TREF, int nDataPathSize, int nCacheLineSize, int nCtrlQueueSize, int nProg = 0, bool Snooping = false, bool VERBOSE = false> bool DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::Activate (int bank, int row, DRAMControlerQueueEntry< INSTRUCTION, nCacheLineSize > * entry) [inline]`

Activate a bank (open a page).

#### Parameters:

- bank* the bank number
- row* the row number

*entry* Controller queue entry

**Returns:**

true if activation was performed

Definition at line 303 of file dram.sim.

References DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::banks, and DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::total\_activate.

Referenced by DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::DRAMControl().

**3.1.3.2** `template<class INSTRUCTION, int nBanks, int nRows, int nCols, int TRRD, int TRAS, int TRCD, int CL, int TRP, int TRC, uint64_t TREF, int nDataPathSize, int nCacheLineSize, int nCtrlQueueSize, int nProg = 0, bool Snooping = false, bool VERBOSE = false> bool DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::Precharge (int bank, DRAMControlerQueueEntry< INSTRUCTION, nCacheLineSize > * entry) [inline]`

Precharge a bank (close a page).

**Parameters:**

*bank* the bank number

*entry* Controller queue entry

**Returns:**

true if precharge was performed

Definition at line 339 of file dram.sim.

References DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::banks, and DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::total\_precharge.

Referenced by DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::DRAMControl().

**3.1.3.3** `template<class INSTRUCTION, int nBanks, int nRows, int nCols, int TRRD, int TRAS, int TRCD, int CL, int TRP, int TRC, uint64_t TREF, int nDataPathSize, int nCacheLineSize, int nCtrlQueueSize, int nProg = 0, bool Snooping = false, bool VERBOSE = false> bool DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::PrechargeAll () [inline]`

Precharge all banks (close all pages)

### 3.1 DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE > Class

#### Template Reference

11

#### Returns:

true if precharge was performed

Definition at line 362 of file dram.sim.

References DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::banks, and DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::total\_precharge.

Referenced by DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::end\_of\_cycle().

**3.1.3.4** `template<class INSTRUCTION, int nBanks, int nRows, int nCols, int TRRD, int TRAS, int TRCD, int CL, int TRP, int TRC, uint64_t TREF, int nDataPathSize, int nCacheLineSize, int nCtrlQueueSize, int nProg = 0, bool Snooping = false, bool VERBOSE = false> bool DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::Read (int bank, int col, DRAMControlerQueueEntry< INSTRUCTION, nCacheLineSize > *entry) [inline]`

Do a read command.

#### Parameters:

*bank* the bank number

*col* the column number

*entry* Controler queue entry

#### Returns:

true if read command was performed

Definition at line 394 of file dram.sim.

References DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::banks, and DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::total\_read.

Referenced by DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::DRAMControl(), DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::DRAMDataTransfer(), and DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::ReadMemory().

**3.1.3.5** `template<class INSTRUCTION, int nBanks, int nRows, int nCols, int TRRD, int TRAS, int TRCD, int CL, int TRP, int TRC, uint64_t TREF, int nDataPathSize, int nCacheLineSize, int nCtrlQueueSize, int nProg = 0, bool Snooping = false, bool VERBOSE = false> bool DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::Write (int bank, int col, DRAMControlerQueueEntry< INSTRUCTION, nCacheLineSize > * entry) [inline]`

Do a write command.

**Parameters:**

*bank* the bank number  
*col* the column number  
*entry* Controler queue entry

**Returns:**

true if write command was performed

Definition at line 425 of file dram.sim.

References DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::banks, and DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::total\_write.

Referenced by DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::DRAMControl(), DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::DRAMDataTransfer(), and DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::WriteMemory().

**3.1.3.6** `template<class INSTRUCTION, int nBanks, int nRows, int nCols, int TRRD, int TRAS, int TRCD, int CL, int TRP, int TRC, uint64_t TREF, int nDataPathSize, int nCacheLineSize, int nCtrlQueueSize, int nProg = 0, bool Snooping = false, bool VERBOSE = false> bool DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::ReadData (int bank, DRAMControlerQueueEntry< INSTRUCTION, nCacheLineSize > * entry) [inline]`

Do a read data transfer from the [DRAM](#).

**Parameters:**

*bank* the bank number  
*entry* Controler queue entry

**Returns:**

true if performed

**3.1 DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE > Class**  
**Template Reference**

13

Definition at line 454 of file dram.sim.

References DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::banks.

Referenced by DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::DRAMDataTransfer().

**3.1.3.7** `template<class INSTRUCTION, int nBanks, int nRows, int nCols, int TRRD, int TRAS, int TRCD, int CL, int TRP, int TRC, uint64_t TREF, int nDataPathSize, int nCacheLineSize, int nCtrlQueueSize, int nProg = 0, bool Snooping = false, bool VERBOSE = false> bool DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::WriteData (int bank, DRAMControlerQueueEntry< INSTRUCTION, nCacheLineSize > * entry) [inline]`

Do a write data transfer from the [DRAM](#).

**Parameters:**

*bank* the bank number

*entry* Controler queue entry

**Returns:**

true if performed

Definition at line 495 of file dram.sim.

References DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::banks.

Referenced by DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::DRAMDataTransfer().

**3.1.3.8** `template<class INSTRUCTION, int nBanks, int nRows, int nCols, int TRRD, int TRAS, int TRCD, int CL, int TRP, int TRC, uint64_t TREF, int nDataPathSize, int nCacheLineSize, int nCtrlQueueSize, int nProg = 0, bool Snooping = false, bool VERBOSE = false> bool DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::Refresh (int row) [inline]`

Performs a refresh of a row on all banks.

**Parameters:**

*row* the row number

**Returns:**

true if refresh was performed

Definition at line 528 of file dram.sim.

References DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::banks, and DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::total\_refresh.

Referenced by DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::end\_of\_cycle().

**3.1.3.9** `template<class INSTRUCTION, int nBanks, int nRows, int nCols, int TRRD, int TRAS, int TRCD, int CL, int TRP, int TRC, uint64_t TREF, int nDataPathSize, int nCacheLineSize, int nCtrlQueueSize, int nProg = 0, bool Snooping = false, bool VERBOSE = false> bool DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::DRAMDataTransfer () [inline]`

Performs data transfer between memory controller and the [DRAM](#).

**Returns:**

true if performed

Definition at line 566 of file dram.sim.

References DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::banks, DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::Read(), DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::ReadData(), DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::Write(), and DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::WriteData().

Referenced by DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::end\_of\_cycle().

**3.1.3.10** `template<class INSTRUCTION, int nBanks, int nRows, int nCols, int TRRD, int TRAS, int TRCD, int CL, int TRP, int TRC, uint64_t TREF, int nDataPathSize, int nCacheLineSize, int nCtrlQueueSize, int nProg = 0, bool Snooping = false, bool VERBOSE = false> void DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::DRAMControl () [inline]`

Controls the [DRAM](#) banks

Definition at line 651 of file dram.sim.

References DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::Activate(), DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::banks, DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::Precharge(), DRAM< INSTRUCTION, nBanks,

**3.1 DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE > Class**

**Template Reference**

**15**

~~nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::Read(), and DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::Write().~~

Referenced by DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::end\_of\_cycle().

**3.1.3.11** `template<class INSTRUCTION, int nBanks, int nRows, int nCols, int TRRD, int TRAS, int TRCD, int CL, int TRP, int TRC, uint64_t TREF, int nDataPathSize, int nCacheLineSize, int nCtrlQueueSize, int nProg = 0, bool Snooping = false, bool VERBOSE = false> void DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::onInData () [inline]`

A SystemC process for managing the valid, accept and enable hand-shaking

Definition at line 751 of file dram.sim.

References memreq\_types::cmd\_FLUSH, memreq\_types::cmd\_WRITE, memreq\_dataless< INSTRUCTION >::command, DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::in, DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::inShared, DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::out, memreq\_dataless< INSTRUCTION >::req\_sender, DRAMControlerQueueEntry< INSTRUCTION, nCacheLineSize >::size, and DRAMControlerQueueEntry< INSTRUCTION, nCacheLineSize >::write.

Referenced by DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::DRAM().

**3.1.3.12** `template<class INSTRUCTION, int nBanks, int nRows, int nCols, int TRRD, int TRAS, int TRCD, int CL, int TRP, int TRC, uint64_t TREF, int nDataPathSize, int nCacheLineSize, int nCtrlQueueSize, int nProg = 0, bool Snooping = false, bool VERBOSE = false> void DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::Read (void * buffer, address_t address, int size) [inline]`

Read data from the memory

#### Parameters:

*buffer* a buffer where to put the data

*address* the starting address

*size* the size in bytes

Definition at line 1174 of file dram.sim.

**3.1.3.13** `template<class INSTRUCTION, int nBanks, int nRows, int nCols, int TRRD, int TRAS, int TRCD, int CL, int TRP, int TRC, uint64_t TREF, int nDataPathSize, int nCacheLineSize, int nCtrlQueueSize, int nProg = 0, bool Snooping = false, bool VERBOSE = false> void DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::Write (address_t address, const void * buffer, int size) [inline]`

Write data into the memory

**Parameters:**

*address* the starting address  
*buffer* a buffer to copy into memory  
*size* the size in bytes

Definition at line 1183 of file dram.sim.

**3.1.3.14** `template<class INSTRUCTION, int nBanks, int nRows, int nCols, int TRRD, int TRAS, int TRCD, int CL, int TRP, int TRC, uint64_t TREF, int nDataPathSize, int nCacheLineSize, int nCtrlQueueSize, int nProg = 0, bool Snooping = false, bool VERBOSE = false> void DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE >::Set (address_t address, uint8_t data, int size) [inline]`

Initialize several byte into the memory

**Parameters:**

*address* the starting address  
*data* the value to copie into the memory  
*size* the size in bytes

Definition at line 1207 of file dram.sim.

## 3.1.4 Friends And Related Function Documentation

**3.1.4.1** `template<class INSTRUCTION, int nBanks, int nRows, int nCols, int TRRD, int TRAS, int TRCD, int CL, int TRP, int TRC, uint64_t TREF, int nDataPathSize, int nCacheLineSize, int nCtrlQueueSize, int nProg = 0, bool Snooping = false, bool VERBOSE = false> ostream& operator<< (ostream & os, const DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE > & dram) [friend]`

Dumps the [DRAM](#) and the [DRAM](#) controler state into an output stream

**Parameters:**

*os* an output stream  
*dram* a [DRAM](#) module

### 3.1 DRAM< INSTRUCTION, nBanks, nRows, nCols, TRRD, TRAS, TRCD, CL, TRP, TRC, TREF, nDataPathSize, nCacheLineSize, nCtrlQueueSize, nProg, Snooping, VERBOSE > Class

#### Template Reference

17

#### Returns:

the output stream

Definition at line 1216 of file dram.sim.

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

- dram.sim

## 3.2 DRAMBank< INSTRUCTION, nCacheLineSize > Class Template Reference

DRAM Bank.

```
#include <dram_components.h>
```

### Public Member Functions

- [DRAMBank \(\)](#)  
*Creates a new DRAM bank.*
- void [WarmRestart \(\)](#)  
*Perform a warm restart.*
- void [ResetStats \(\)](#)  
*Reset all the statistics.*

### Public Attributes

- SDRAMState [state](#)  
*Associated state.*
- int [col](#)  
*Associated column.*
- int [row](#)  
*Associated row.*
- int [burst\\_counter](#)  
*Burst counter.*
- uint64\_t [lastACTV](#)  
*Event timestamp to compute latencies.*
- uint64\_t [lastPRE](#)  
*Event timestamp to compute latencies.*
- uint64\_t [lastPREALL](#)  
*Event timestamp to compute latencies.*
- uint64\_t [lastREFR](#)  
*Event timestamp to compute latencies.*
- uint64\_t [lastSLFR](#)  
*Event timestamp to compute latencies.*
- uint64\_t [lastSLFRE](#)

*Event timestamp to compute latencies.*

- `uint64_t lastMRS`  
*Event timestamp to compute latencies.*
- `uint64_t lastREAD`  
*Event timestamp to compute latencies.*
- `uint64_t lastWRT`  
*Event timestamp to compute latencies.*
- `uint64_t lastREADP`  
*Event timestamp to compute latencies.*
- `uint64_t lastWRTP`  
*Event timestamp to compute latencies.*
- `DRAMControlerQueueEntry< INSTRUCTION, nCacheLineSize > * entry`  
*Controller queue entry.*
- `long long int total_activate`  
*Statistics: total number of activate.*
- `long long int total_precharge`  
*Statistics: total number of precharge.*
- `long long int total_read`  
*Statistics: total number of read.*
- `long long int total_write`  
*Statistics: total number of write.*

## Friends

- `ostream & operator<<< (ostream &os, const DRAMBank &bk)`  
*Class pretty printer.*

### 3.2.1 Detailed Description

`template<class INSTRUCTION, int nCacheLineSize> class DRAMBank< INSTRUCTION, nCacheLineSize >`

[DRAM](#) Bank.

Definition at line 170 of file `dram_components.h`.

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

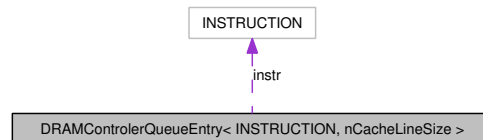
- `dram_components.h`

### 3.3 DRAMControlerQueueEntry< INSTRUCTION, nCacheLineSize > Class Template Reference

DRAM Controler Queue Entry.

```
#include <dram_components.h>
```

Collaboration diagram for DRAMControlerQueueEntry< INSTRUCTION, nCacheLineSize >:



#### Public Member Functions

- [DRAMControlerQueueEntry](#) & [operator=](#) (const [DRAMControlerQueueEntry](#) &entry)  
*Copy constructor.*

#### Public Attributes

- bool [launched](#)  
*True once the request start to be processed.*
- bool [dram\\_data\\_transfer\\_finished](#)  
*True once the data transfer is over.*
- bool [write](#)  
*Wether the entry correspond to a write.*
- address\_t [address](#)  
*Target address of the entry.*
- address\_t [base\\_address](#)  
*Base address of the entry.*
- int [write\\_offset](#)  
*Entry write offset.*
- int [read\\_offset](#)  
*Entry read offset.*
- int [row](#)  
*Corresponding row.*
- int [col](#)  
*Corresponding column.*

### 3.3 DRAMControlerQueueEntry< INSTRUCTION, nCacheLineSize > Class Template Reference 21

- int [bank](#)  
*Corresponding bank.*
- INSTRUCTION [instr](#)  
*Instruction that issued this entry.*
- int [size](#)  
*Size of the request (in bytes).*
- int [read](#)  
*???*
- char [data](#) [nCacheLineSize]  
*Corresponding data.*
- int [tag](#)  
*Associated tag.*
- uint64\_t [memreq\\_id](#)  
*Corresponding memreq\_id.*
- module \* [req\\_sender](#)  
*Module that issued the corresponding request.*
- bool [cachable](#)  
*Whether the request was cachable.*
- memreq\_types::command\_t [command](#)  
*Request command type (read, write, ...).*
- unsigned long long int [time](#)  
*???*

#### Friends

- ostream & [operator<<](#) (ostream &os, const [DRAMControlerQueueEntry](#) &entry)  
*Class pretty printer.*

#### 3.3.1 Detailed Description

`template<class INSTRUCTION, int nCacheLineSize> class DRAMControlerQueueEntry< INSTRUCTION, nCacheLineSize >`

[DRAM](#) Controler Queue Entry.

Definition at line 55 of file dram\_components.h.

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

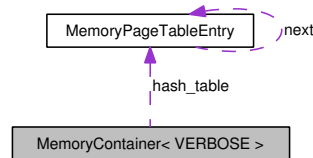
- dram\_components.h

## 3.4 MemoryContainer< VERBOSE > Class Template Reference

[MemoryContainer](#): memory content.

```
#include <MemoryContainer.h>
```

Collaboration diagram for MemoryContainer< VERBOSE >:



### Public Member Functions

- [MemoryContainer](#) ()  
*Creates a new [MemoryContainer](#).*
- [~MemoryContainer](#) ()  
*Class destructor.*
- void [load\\_checkpoint](#) (istream &is)  
*Loads the Memory content from a checkpoint file.*
- void [Read](#) (void \*buffer, address\_t address, int size)  
*Memory interface : read from memory.*
- void [Write](#) (address\_t address, const void \*buffer, int size)  
*Memory interface : write to memory.*
- void [Set](#) (address\_t address, uint8\_t data, int size)  
*Set a data in the corresponding memory page.*
- void [Scan](#) ()  
*Scan or consistency the whole memmory hash.*

### Friends

- ostream & [operator<<](#) (ostream &os, [MemoryContainer](#) &mc)  
*pretty printer*

### 3.4.1 Detailed Description

```
template<bool VERBOSE> class MemoryContainer< VERBOSE >
```

[MemoryContainer](#): memory content.

Definition at line 114 of file MemoryContainer.h.

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

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

## 3.5 MemoryPageTableEntry Class Reference

Collaboration diagram for MemoryPageTableEntry:



### Public Member Functions

- [MemoryPageTableEntry \(\)](#)  
*Empty constructor, required to build lists.*
- [MemoryPageTableEntry \(address\\_t \\_address, MemoryPageTableEntry \\*\\_next\)](#)  
*Class constructor.*
- [~MemoryPageTableEntry \(\)](#)  
*Class destructor.*
- void [dump](#) (ostream &os, address\_t addr, const list< address\_t > &last\_modified)  
*Dumps some memory content.*

### Public Attributes

- address\_t [address](#)  
*Corresponding address.*
- class [MemoryPageTableEntry](#) \* [next](#)  
*Pointer to next memory page for this hash key.*
- uint8\_t [storage](#) [MEMORY\_CONTAINER\_PAGE\_SIZE]  
*Memory data.*

### 3.5.1 Detailed Description

Definition at line 55 of file MemoryContainer.h.

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

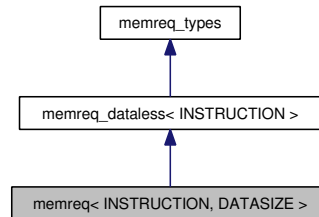
- /home/girbal/svn/unisim.org/public/components/CycleLevel/packages/system/MemoryContainer.h

### 3.6 memreq< INSTRUCTION, DATASIZE > Class Template Reference

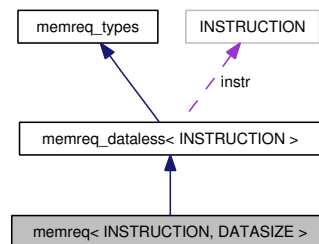
Memory request class.

```
#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.*

### 3.6.1 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:

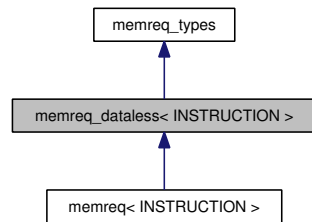
- /home/girbal/svn/unisim.org/public/components/CycleLevel/packages/memreq.h

### 3.7 memreq\_dataless< INSTRUCTION > Class Template Reference

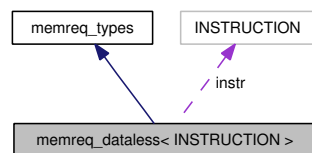
Data-less memory request class.

```
#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.*

### 3.7.1 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:

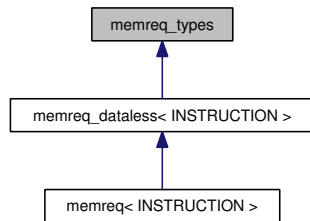
- `/home/girbal/svn/unisim.org/public/components/CycleLevel/packages/memreq.h`

## 3.8 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 `message_type_t` enum.*

### 3.8.1 Detailed Description

Non-templated type class for [memreq](#) defining the enums and their pretty printers.

Definition at line 54 of file memreq.h.

### 3.8.2 Member Enumeration Documentation

#### 3.8.2.1 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.

#### 3.8.2.2 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.

#### 3.8.2.3 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

- Activate
  - DRAM, 9
- cmd\_BLOCK\_INVALIDATE
  - memreq\_types, 31
- cmd\_EVICT
  - memreq\_types, 31
- cmd\_FLUSH
  - memreq\_types, 31
- cmd\_PREFETCH
  - memreq\_types, 31
- cmd\_READ
  - memreq\_types, 31
- cmd\_READX
  - memreq\_types, 31
- cmd\_UNKNOWN
  - memreq\_types, 31
- cmd\_WRITE
  - memreq\_types, 31
- command\_t
  - memreq\_types, 31
- DRAM, 5
  - Activate, 9
  - DRAM, 8
  - DRAMControl, 14
  - DRAMDataTransfer, 14
  - onInData, 15
  - operator<<, 16
  - Precharge, 10
  - PrechargeAll, 10
  - Read, 11, 15
  - ReadData, 12
  - Refresh, 13
  - Set, 16
  - Write, 11, 15
  - WriteData, 13
- DRAMBank, 18
- DRAMControl
  - DRAM, 14
- DRAMControlQueueEntry, 20
- DRAMDataTransfer
  - DRAM, 14
- MemoryContainer, 23
- MemoryPageTableEntry, 25
- memreq, 26
- memreq\_dataless, 28
- memreq\_types
  - cmd\_BLOCK\_INVALIDATE, 31
  - cmd\_EVICT, 31
  - cmd\_FLUSH, 31
  - cmd\_PREFETCH, 31
  - cmd\_READ, 31
  - cmd\_READX, 31
  - cmd\_UNKNOWN, 31
  - cmd\_WRITE, 31
  - sender\_CACHE, 31
  - sender\_CPU, 31
  - sender\_MEM, 31
  - sender\_UNKNOWN, 31
  - type\_ANSWER, 31
  - type\_REQUEST, 31
  - type\_UNKNOWN, 31
- memreq\_types, 30
  - command\_t, 31
  - message\_type\_t, 31
  - sender\_type\_t, 31
- message\_type\_t
  - memreq\_types, 31
- onInData
  - DRAM, 15
- operator<<
  - DRAM, 16
- Precharge
  - DRAM, 10
- PrechargeAll
  - DRAM, 10
- Read
  - DRAM, 11, 15
- ReadData
  - DRAM, 12
- Refresh
  - DRAM, 13
- sender\_CACHE
  - memreq\_types, 31
- sender\_CPU

---

- memreq\_types, [31](#)
- sender\_MEM
  - memreq\_types, [31](#)
- sender\_type\_t
  - memreq\_types, [31](#)
- sender\_UNKNOWN
  - memreq\_types, [31](#)
- Set
  - DRAM, [16](#)
  
- type\_ANSWER
  - memreq\_types, [31](#)
- type\_REQUEST
  - memreq\_types, [31](#)
- type\_UNKNOWN
  - memreq\_types, [31](#)
  
- Write
  - DRAM, [11](#), [15](#)
- WriteData
  - DRAM, [13](#)