DDR Controller Architecture Description and Preliminary Data sheet

DDR Controller Architecture Description and Preliminary Datasheet

2020-05-11

Doc. No DDR-FADDR-D3.CTL.0

Issue 2.0

Contract 400115045/15/NL/LF

Deliverable D3.CTL.0 / D3.CTL.3

CHANGE RECORD

Issue	Date	Section / Page	Description
2.0	2020-05-11	All	Clear changebars and remove implementation notes. Release for ESA IP Core portfolio.

INTRODUCTION

Scope of the Document

This document establishes the DDR Controller Architecture Description and Preliminary Datasheet for the activity “Rad-Hard DDR Physical Interface (PHY and IOs) and Digital Controller for Space-ST65nm” initiated by the European Space Agency under ESTEC contract 400115045/15/NL/LF.

The work has been performed by Cobham Gaisler AB, Göteborg, Sweden.

Structure of document

The remaining sections of this document are from the digital controller IP core architecture description and preliminary data sheet. The format of the digital controller documentation is in FrameMaker format while the surrounding parts of the document have been kept in the same document processing tool as used for the other documents within the project.

The FrameMaker documentation is also available in source format.

1 FTADDR - Autonomous DDR2/DDR3 Controller with EDAC

1.1 Overview

FTADDR is a memory controller for DDR2 and DDR3 type of SDRAM memory devices. On the memory side, it presents a DFI interface for connection to an on-chip physical layer (PHY) that manages the low-level timing and data recovery and then provides the I/O buffers. Towards the system-on-chip, it presents the memory through an AMBA AHB slave interface.

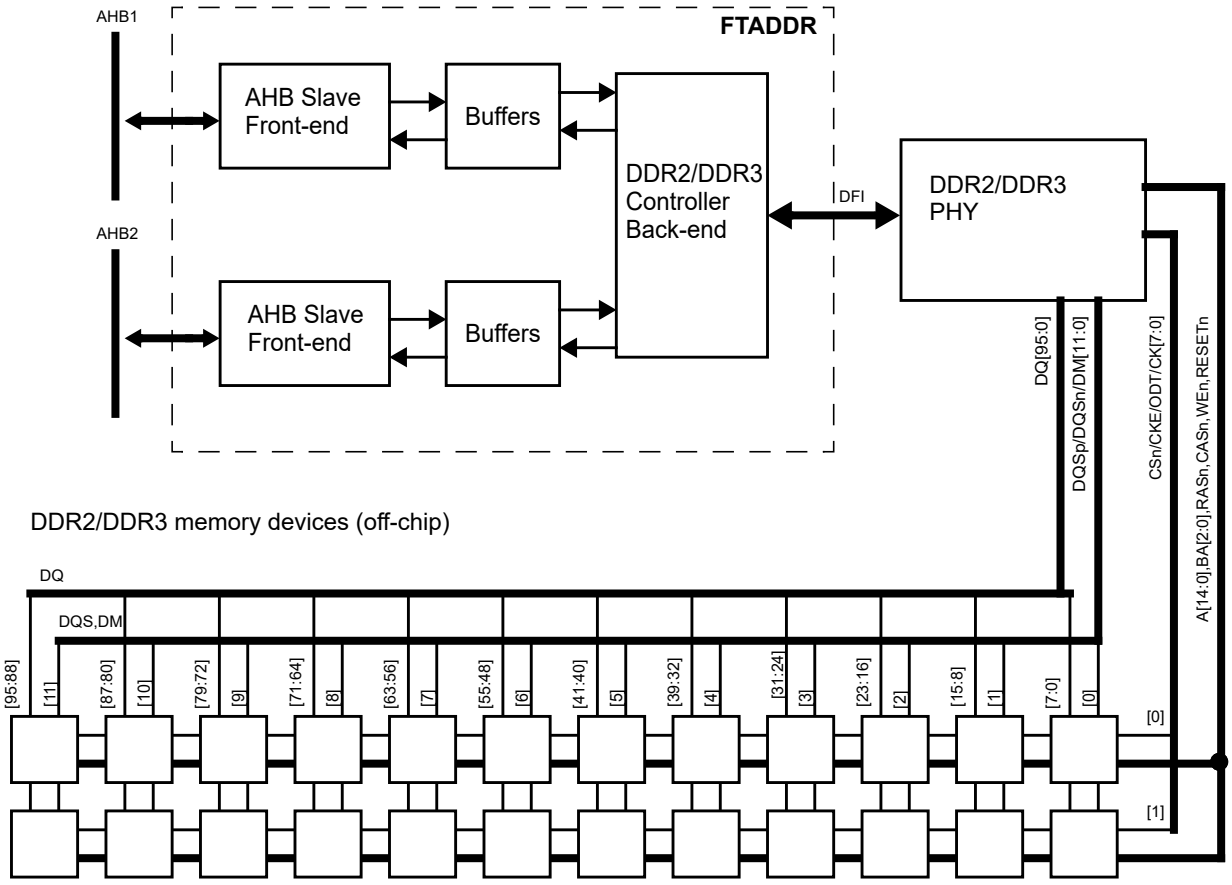


Figure 1. Block diagram of FTADDR Memory controller with two ports and two external banks of 96 bit wide DDR2/DDR3 x8 RAM devices.

Internally, the controller is composed of one part clocked by the AMBA clock which manages the user interface, one part clocked by the DFI clock to manage DDR command scheduling and initialization, and FIFOs to communicate between these two parts.

The controller is designed to interface 96-bit wide banks of memory, made out of individual memory devices of width x8 or x4, and then uses a strong error correction code to achieve double-device correction capability. This allows it to deliver correct data despite one full device failure and random SEU-induced errors on the other devices. Up to 8 parallel banks (chip selects) are supported by the controller. Reduced configurations are possible for reduced pin count.

The controller can be used in a multi-ported configuration to support concurrent accesses to different memory banks.

The controller provides configuration registers accessed through a separate address area in the AHB slave. The controller is designed to support processor-less configurations and therefore can operate autonomously with the desired configuration settings supplied as input data at system reset. An interface for direct control of DDR commands and diagnostic reading of data and checkbits is also provided.

1.2 Operation

1.2.1 Memory bank configurations

Single DDR2/DDR3 SDRAM chips are typically 4,8 or 16 data bits wide. By putting multiple identical chips side by side, wider SDRAM memory banks can be built. Since the command signals are common for all chips, the memories behave as one single wide memory chip, called an external bank or rank. This memory controller supports up to eight identical such memory banks.

Three error detection/correction configurations are supported by the controller: EDAC, parity and none. The controller can be set into two width settings, full width and half width. The controller can interface banks of x8 or x4 devices. This combined creates 12 possible configurations, listed below.

Devices of width x16 behave exactly as two x8 devices in terms of protocol and is therefore functionally compatible with the controller, however as one device error may affect up to 16 bits, it is not compatible with the EDAC schemes employed by the controller.

Note that the x4 mode is intended for use with PHYs designed with 8 data bits per DQS pair using only the lowest nibble of each byte lane, if the PHY can manage 4 data bits per DQS and present that as a full DFI data bus then the regular x8 mode can be used.

Table 1. List of memory bank configurations supported by the FTADDR Memory controller

Configuration #	Controller setting			External bank layout		# DQ bits	Device-wide error correction	
	enx4	eccen	hwidth	Device type	# of devices	Data+Check	# Corrected	# Detected
1	0 (x8)	10 (EDAC)	0	x8	12	64+32	2	2
2			1		6	32+16	1	1
3		01 (Parity)	0		9	64+8	0	1
4			1		5	32+8	0	1
5		00 (None)	0		8	64+0	0	0
6			1		4	32+0	0	0
7	1 (x4)	10	0	x4	12	32+16	2	2
8			1		6	16+8	1	1
9		01	0		9	32+4	0	1
10			1		5	16+4	0	1
11		00	0		8	32+0	0	0
12			1		4	16+0	0	0

1.2.2 Configurable timing parameters

To provide optimum access cycles for different DDR2/DDR3 devices (and at different frequencies), seven timing parameters can be programmed through the memory configuration registers. The value of these fields affects the memory timing as described in table 2. Note that if the CAS latency setting is changed after initialization, this change needs also to be programmed into the memory chips by executing the Load Mode Register command.

Table 2. FTADDR programmable DDR2/DDR3 command timings

DDR2/DDR3 SDRAM timing parameter	Register fields used	Minimum timing (clocks) as function of field values	Formula for field value from timing parameters
CAS latency, CL	DDRT, CASLAT	DDR2: CASLAT DDR3: CASLAT+4	DDR2: CASLAT=CL DDR3: CASLAT=CL-4
CAS write latency, CWL (for DDR2, CWL is fixed to (CL-1))	DDRT, CASLAT, WLAT	DDR2: CASLAT-1 DDR3: WLAT+5	DDR2: WLAT=0 (unused) DDR3: WLAT=CWL-5
Activate to read/write command (t_{RCD})	DRCD	DRCD+2	DRCD=ceil(t_{RCD}/t_{CK})-2
Read to precharge (t_{RTP})[2]	DRTP	DDR2: DRTP+0 DDR3: DRTP+2	DDR2: DRTP=min(ceil(t_{RTP}/t_{CK}), 2) DDR3: DRTP=min(ceil(t_{RTP}/t_{CK})-2, 2)
Write recovery time (t_{WR})	DWR, DDRT, CASLAT, WLAT	DDR2: DWR-CASLAT-1 DDR3: DWR-WLAT-7	DWR=ceil(t_{WR}/t_{CK})+CWL+2
Precharge to activate (t_{RP})	DRP	DRP+3	DRP=ceil(t_{RP}/t_{CK})-3
Activate to precharge (t_{RAS})	DRAS	DRAS+3	DRAS=ceil(t_{RAS}/t_{CK})-3
Auto-refresh command period (t_{RFC})	DRFC	DRFC+3	DRFC=ceil(t_{RFC}/t_{CK})-3
Activate to activate command period (t_{RRD})	DRRD	DRRD+2	DRRD=ceil(t_{RRD}/t_{CK})-2
Four activate window (t_{FAW})	DFAW	DFAW+DRAS+4	DFAW=max (0, ceil((t_{FAW}-t_{RAS})/t_{CK})-1)
Write to read command delay (t_{WTR})	DWTR, CASLAT, WLAT	DDR2: DWTR-CASLAT-1 DDR2: DWTR-WLAT-7	DWTR=ceil(t_{WTR}/t_{CK})+CWL+2
DDR3 mode register set command update delay (t_{MOD})[1]	DMOD	DMOD+3	DMOD=t_{MOD} - 3

[1] tMOD also exists with a different definition on DDR2, this does not require the DMOD field to be set

[2] Note that tRTP is defined differently for DDR2 (with BL8) and DDR3, leading to the 2 cycle difference shown here.

If the fields in table 2 are programmed such that the memory specifications are fulfilled, and periodic refresh is correctly setup, then the remaining SDRAM command timing parameters will also be met by design. Table 3 derives example register value for some of the standard speed bins.

Table 3. FTADDR timing setup examples based on JEDEC speed bins

Speed bin	DDR2-400B	DDR2-800C	DDR3-800D	DDR3-1600G	
Density / Page size	1 Gb / 2 K	1 Gb / 2 K	2 Gb / 2 K	2 Gb / 2 K	
Cycle time	5.0	2.5	2.5	1.25	ns
CL	3	4	5	8	nCK
CWL	2	3	5	8	nCK
tRCD	15	10	12.5	10	ns
tRTP	7.5	7.5	10	7.5	ns
tWR	15	15	15	15	ns
tRP	15	12.5	12.5	10	ns
tRAS	40	45	37.5	35	ns
tRFC	127.5	127.5	160	160	ns
tRRD	7.5	7.5	10	6	ns
tFAW	50	45	50	40	ns
tWTR	7.5	7.5	10	7.5	ns
tMOD,DDR3	-	-	12	12	nCK
DDRT	0	0	1	1	
CASLAT	3	4	1	4	
WLAT	0	0	0	3	
DRCD	1	2	3	6	
DRTP	2	3	2	4	
DWR	7	11	13	22	
DRP	0	2	2	5	
DRAS	5	15	12	25	
DRFC	23	48	48	99	
DRRD	0	1	2	3	
DFAW	1	0	4	3	
DWTR	6	8	11	16	
DMOD	0	0	9	9	
Memory configuration register 1	0x00c0a008	0x0101e081	0x804180a2	0x8133215b	
Memory configuration register 2	0x0c7210b8	0x10b32180	0x16d23380	0x21646318	
Memory configuration register 3	0x00000000	0x00000000	0x00000009	0x00000009	

1.2.3 Registered SDRAM

Registered memory modules (RDIMM:s) have one cycle extra latency on the control signals due to the external register. They can be supported with this core by setting the REG control register bit. When set this shifts the internal latency used by the controller one cycle relative to the latency programmed to the memory, which compensates for the delay.

This should not be confused with Fully-Buffered DDR2/DDR3 memory, which uses a different protocol and is not supported by this controller.

1.2.4 On-die termination management

At the memory device side, for each chip select the controller can be set up to program different values of on-die termination (Rtt, and for DDR3 also Rtt_wr) and output inpedance setting into the mode register.

To control the on-die termination in the memory devices (via the ODT control signal), the configuration registers of the controller contain an ODT enable matrix deciding which of the ODT signals are enabled when a specific chip select is being read and written. There is also a default ODT configuration set when neither reading nor writing. By default ODT is always off.

The timing of the ODT signal enabling relative to the read and write command is programmable. The earliest supported switching of the ODT signals are one cycle after the command. For low-latency DDR2 writes this may not be fast enough to switch on the ODT in time, in which case the default ODT configuration should be set to the write configuration instead so that it is set already when the command is given.

A similar configuration is available to enable the controller-side termination on reads, if such termination exist in the PHY.

For DDR3 memory, the Rtt setting and ODT enable state desired during write leveling can be selected separately from the setting used during functional mode. Note that the JEDEC standard allows only a subset of the Rtt settings during write leveling.

1.2.5 Setup

The controller is designed to be configurable using two possible means:

- Setting up all configuration register's reset values using external signals, then using the controller directly. The external signals can be either tied to fixed values at higher level in the design, tied to top-level inputs or driven by other logic at the top level of the design
- Configuring parameters over the on-chip bus from processor or other SoC resource

The type of memory (DDR2 or DDR3) and the external bank configuration (see section 1.2.1) must be set up before starting up the controller and must remain at the same setting, while other settings can be modified at any time. For correct operation, the timing and size parameters must be set to match the device used, see sections 1.2.2 and 1.2.8.

1.2.6 Memory device initialization and management

Assuming the controller is setup correctly, the controller will manage initialization, refresh and other housekeeping functions. For details on initialization and low-level DDR handling, see section 1.3

1.2.7 Data access

Data accesses to the AHB slave ports are translated to DDR memory accesses by the controller, towards the rest of the system-on-chip the DDR memory will appear as a flat memory space. On the DDR side, all accesses are bursts of length 8, while on the AHB side any length is supported and will be mapped to the necessary number of bursts.

In the multi-ported configuration, each slave port maps to a separate fraction of the available memory. This is done by mapping different ports to different internal banks. Having a fixed mapping all the way from front-end ports minimizes the timing interference between the ports, since accesses on different ports can never map to the same memory row. Accesses can always be done overlapping and therefore the worst case interference becomes only the sum of the maximum data burst length of the other ports rather than one full row open-close cycle for each other port.

For example, with DDR3-800 memory, burst length 8, four ports, the worst case interference becomes 3 ports*(4 cycles burst + 2 cycles turnaround) = 18 DDR cycles = 45 ns, while with a any-to-any mapping the worst case interference would be (all writing to different rows in same internal bank) 3 ports

* 25 cycles = 75 cycles = 187.5 ns. The difference gets worse with increasing DDR memory clock rate.

1.2.8 Logical address to memory address mapping

The controller uses a linear address mapping scheme between on-chip bus addresses and external memory addresses. From the least significant bits upward the bits are interpreted as:

- Byte lane on DDR bus. Depending on external bank configuration (see section 1.2.1) the data width is 16,32 or 64 bits making this field take up the lowest 2-4 bits of the address.
- Column address. Width of this field depends on the number of columns in the external memory devices, between 9-12 bits
- Row address. Width of this field depends on the number of rows in the external memory, between 12-16 bits
- Internal bank. In multi-port configurations the least significant bits of the bank number is implied by which port is accessed and is therefore not included in the address translation. Number of banks can be either 4 or 8 for DDR2, and is always 8 for DDR3. This leads to between 0-3 address bits.
- External bank/chip select. Up to 8 are supported, leading to 3 address bits for this field.

To achieve a correct address translation without holes or aliases, the AHB address decode register fields need to be set up to match the external memory devices. The algorithm to do this is as follows:

1. Set COLBASE to match the width of the data bus (not including ECC bits).

2. Set ROWBASE so that (9+ROWBASE-COLBASE) matches the number of column address bits on the DDR memories (bits actually used for addressing the column, not counting that A(10) and A(12) are used for auto-precharge and burst-chop indication).

3. Set BANKBASE so that (10+BANKBASE-ROWBASE) matches the number of row address bits on the DDR memories.

4. Set CSBASE so that (2^(CSBASE-BANKBASE)+NPORTS) matches the number of internal banks (4 or 8) or the DDR memories.

5. If the resulting CSBASE value from step 4 became 11 or higher then subtract (CSBASE-10) from both CSBASE and BANKBASE

1.3 Back-end functional description

1.3.1 Controller back-end states

From a usage perspective, the DDR side of the controller follows the simplified state diagram shown in figure 2.

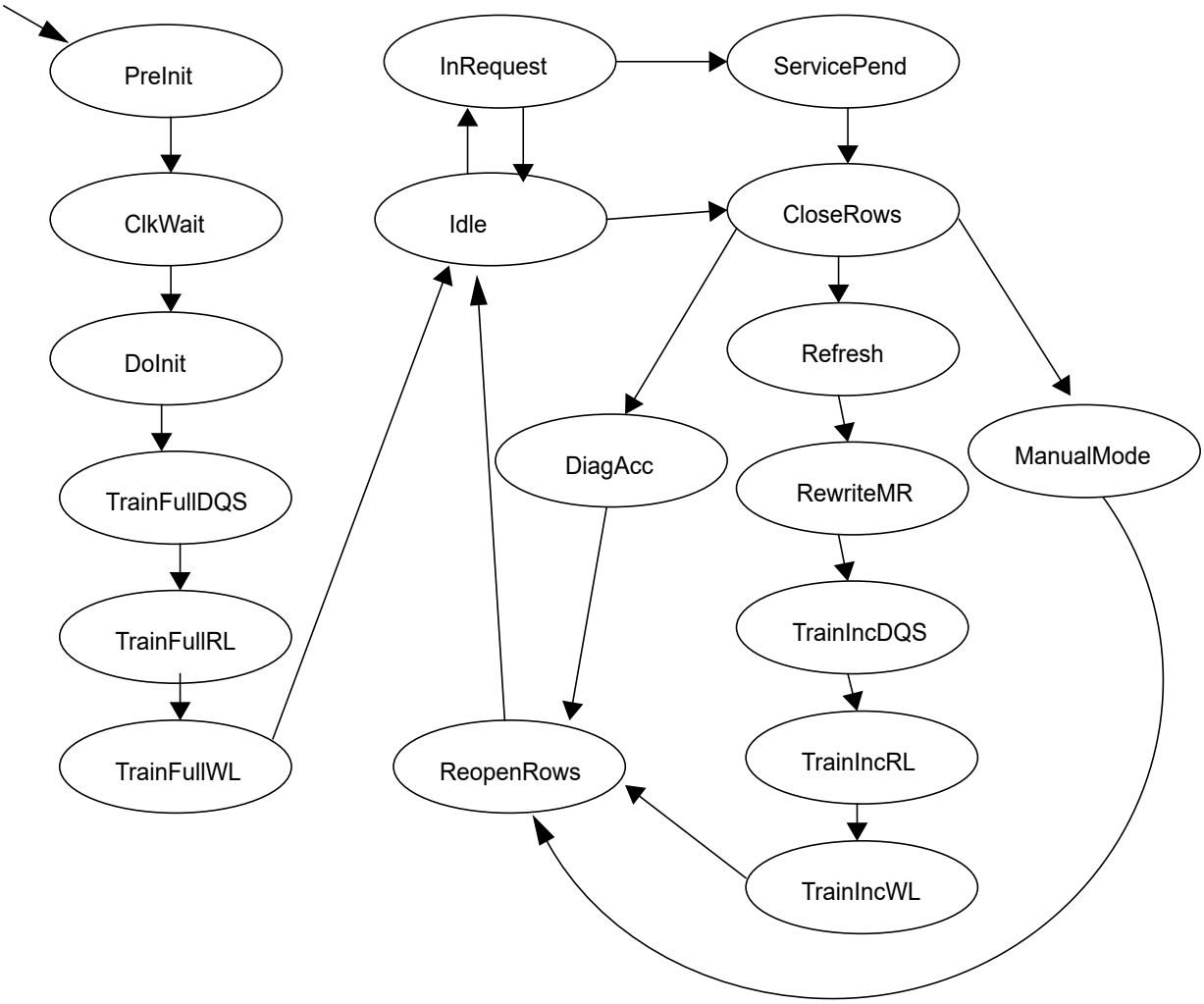


Figure 2. Simplified state diagram of DDR2/DDR3 controller back-end

PreInit - the controller waits for the configuration settings to be set and the pwron bit to be set in the controller. Note that if pwron is set high by the signal interface then this state will be left immediately after reset.

ClkWait - wait until the minimum time has passed from power-on reset before starting the initialization sequence as required by the JEDEC standard. This can be skipped for testing/simulation purposes by writing into the training time counter register while in this state.

DoInit - Initialization sequence according to the selected memory standard.

TrainFullDQS - Training of DQS gate timing

TrainFullRL - Training of read data leveling (DDR3 only)

TrainFullWL - Training of write data leveling (DDR3 only)

Idle - Waiting for incoming request or service interval timer

InRequest - Serving one or more read/write requests. Can start additional requests to other banks in parallel as decided by internal scheduler and timers.

ServicePend - Serving one or more read/write requests, the service interval timer has expired. New requests are not accepted.

CloseRows - At start of service interval, if any rows are then send a close all rows command to all chip selects. Wait for any tRP delay timers to expire.

Refresh - Send auto-refresh command to all memory devices, if configured to do so at this service interval.

TrainIncDQS - Performing periodic incremental DQS gate delay adjustment, if configured to do so at this service interval.

TrainIncRL - Performing periodic incremental read data leveling (DDR3 only), if configured to do so at this service interval.

TrainIncWL - Performing periodic incremental write data leveling (DDR3 only), if configured to do so at this service interval.

ReopenRows - Re-open any of the open rows that were closed at the beginning of the service interval. If the controller can see that the next command coming in on a port will result in closing the row (a row close command, or a row open command to a different row), then the row will not be re-opened.

DiagAcc - Perform diagnostic access requested by user.

ManualMode - Performing manual or diagnostic command requested by user (AHB frontend)

1.3.2 Data transfers

The controller contains four bank tracking units to keep track of ongoing data transfers and currently opened rows in the external memory. Each tracking unit has a fixed mapping to a specific set of internal banks in the memory devices, based on the two least significant bits of the internal bank number. At most one opened row at a time can be tracked by each bank tracking unit.

A command scheduler module looks at the state of the bank tracking units, and schedules commands to the DDR command bus as they are ready and can be issued without causing data bus collisions with other ongoing commands. In case more than one bank tracker has a command ready the same cycle, the scheduler will use a round-robin arbiter as a tie-breaker.

The front-end ports are mapped to the bank tracking units in a fixed 1:4, 1:2, or 1:1 mapping depending on how many front-end ports are implemented in the controller, so each front-end port will map to a separate slice of the external memory.

1.3.3 Service intervals

There are several functions in the controller that need to be done periodically, and that require the memory devices to be in idle state so can not be done in parallel with regular data transfers. To handle this efficiently, these are handled in common service intervals. The number of cycles per interval is can be set through configuration registers.

You do not need to do every function at every service interval. For each of the functions (refresh, incremental training, mode register reprogramming), you can configure the frequency as an integer multiple of service intervals. For example, you can set refresh to be done every service interval and incremental training every third interval. Service intervals where nothing is to be done are optimized away so there is no blocking of transfers in that case.

Within each service iteration, all currently opened rows are closed, the pending operations are performed, and the rows are then re-opened to resume operation. To avoid unnecessary re-opening of rows just in order to close them shortly after, the controller peeks at the following command in the command FIFOs, and if that is a row-close or row-open command, the re-opening is skipped for that bank machine. This optimization can be disabled with the FREOP configuration register bit.

It is possible to postpone service iterations using the SHOLD configuration bit. The pending bits and the time counting will still proceed normally. Since there is only a single pending bit for each refresh (or other action), suspending the service iteration for longer than the refresh interval can result in losing a pending refresh command leading to lower refresh rate than expected. Therefore when using the SHOLD function, the user either has to ensure the time is shorter than the refresh interval or the user needs to manually trigger additional refreshes to compensate for the lost refresh commands.

1.3.4 Refresh

The controller contains a refresh function that during a service interval issues an AUTO-REFRESH command to all SDRAM banks. Depending on SDRAM type, the required average period is typically 7.8 us.

1.3.5 ZQ calibration

For DDR3 memories, the controller can be programmed to issue ZQ calibration short (ZQCS) commands periodically at service intervals in order to prevent the input/output drivers on the memory devices from drifting out of specification. ZQ calibration short (ZQCS) and ZQ calibration long (ZQCL) commands can also be manually commanded through the command register. One ZQCL command is always issued during initialization as this is required by the JEDEC standard.

1.3.6 Periodic mode register reprogramming

The controller can be set up to reprogram the memory device's mode registers periodically during service intervals to ensure they are set to correct values.

1.3.7 Delay training

Depending on the PHY design and of the speed of the DDR interface, training of delays may be required. The controller has been designed to support such training.

If the PHY uses MC evaluation mode (DFI mode 01), the training is implemented in a custom sub-block of the controller with an algorithm designed for the specific PHY. For a new PHY that requires this a new phyimpl constant must be defined to the controller which instantiates the new sub-block.

In other training modes (PHY evaluation, PHY independent, and None), the calibration is implemented inside the PHY and the controller only provides some supporting services. This is supported in the generic DFI implementation of the controller without any custom logic.

Incremental training can be done periodically within service intervals to track changes in delays. At each incremental training, a fine adjustment of DQS delays and write leveling is performed. The controller can also be setup to re-do the full training with some interval.

1.3.8 User specified commands

The user can force a refresh or mode register load by writing to command register bits. The controller will schedule the requested command as soon as possible, letting any pending data transfers complete, and ensure the memory timing parameters are respected.

1.3.9 Diagnostic access

The controller has a diagnostic interface where you can command a read or write from a specific location via the register interface. The diagnostic interface can be used safely while the core is running normally.

The procedure for performing a diagnostic access is:

1. Write to the Diagnostic access control register 1, set the DIAGCS,DIAGBANK,DIAGROW fields with the desired location

2. If it’s a write, set up the diagnostic checkbit and data registers with the desired data

3. Write to the Diagnostic access control register 2, DIAGCOL field with the desired column, DODG=1, DGWR=1/0 depending on if it’s a read or write

4. Poll the diagnostic access status register until the DDONE field is 1.

5. If it was a read, the diagnostic checkbit and data registers hold the read data

The diagnostic access is handled similar to service iterations, current accesses complete, all rows are closed, the diagnostic access is performed, and rows are then re-opened.

The unit of the diagnostic access is always a 64-bit word with the corresponding checkbits. When reading with EDAC enabled, the EDAC decoder error outputs can be seen in the diagnostic status register after the access, however the data that is stored in the diagnostic data/checkbit registers is always uncorrected.

In half-width mode, each diagnostic access will read or write two columns to get a complete EDAC codeword. The column that is addressed should always be even in this mode. Otherwise the diagnostic register interface behaves exactly the same towards the user.

The diagnostic interface ignores the x4 mode setting, and will handle it like the x8 mode. The user has to ignore the top nibbles of the data and checkbits, and mask the corresponding correctable error bits, when using diagnostic accesses in x4 mode.

1.3.10 Manual mode

A raw interface allows to manually perform arbitrary DDR commands such as mode register write, auto refresh, activate, RAS, read/write, and precharge. Note that these manual commands are not managed by the bank tracker and it’s the responsibility of the user to not violate the SDRAM protocol. While manual mode is activated, normal accesses will be blocked. Also, no refresh or other service commands will be performed during that time

To activate the manual mode, the user writes 1 to the MANE configuration register field, and then polls the Backend status register until the back-end has entered the manual mode. To return back to normal mode, write 0 to the MANE register field.

In manual mode, the address lines are continuously driven with the value of the DIAGROW field, the bank address lines are driven with the value of the DIAGBANK field. The chip select lines are normally driven high but can be asserted for one cycle by writing into the MANUAL_CSN field.

It is possible to command a read or write command and transfer data with proper DQS strobe generation. This is done by asserting the MRDC or MWDC fields at the same time as giving the CAS command. The data transfer is handled in a similar way as for a diagnostic access, and will use the diagnostic data and checkbit registers (however the diagnostic status register is not modified).

1.3.11 PHY-specific support

The controller has a phyimpl configuration option specifying if a specific kind of PHY is used. The back-end adapts the default control signal timing (via the timing masks) and may also adjust to other ‘quirks’ in the specific PHY. Some PHY implementations also provide dedicated configuration or status registers that can be accessed via the PHY indirect address and data registers, those are listed in table 12.

Currently the PHYs in table 4 are implemented.

Table 4. FTADDR back-end PHY implementations supported

Value	Name	PHY-specific registers	Description
0	Generic	No	Generic DFI PHY interface
1	ISD65	Yes	ISD S.A. DDR PHY for C65SPACE.
2	Altera UniPhy/AFI interface	No	For Altera FPGAs. Can interface a full-rate PHY generated by the Quartus UniPhy MegaWizard

1.3.12 Memory reboot support

The controller has features to support resetting and possibly also power cycling of individual byte lanes of memory, in order to recover from permanent SEFI errors. While rebooting one byte lane, the other memories are held in self refresh mode to preserve their contents, and the rebooted byte lane's contents can be recovered using the EDAC.

Using the reboot feature requires special support both on the DDR PHY, system design and on the board level. Instead of a separate RESET and CKE signal per rank, there needs to be a separate RESET and CKE signal per byte slice but shared over all ranks. These are output from the controller on the xdfi_bl_resetn and xdfi_bl_cke signals with the same timings as the regular DFI dfi_reset_n and dfi_cke signals. In addition, there needs to be support for either driving both clk_p and clk_n low, or tristating them, this is activated through the additional xdfi_clk_zero signal. An additional output signal xdfi_bl_pdn signals when the memory should be power cycled, this needs to propagate out to the board level.

If the reboot support is not required, the added signals can be ignored and the standard DFI signals used instead. Using the reboot function will then not have any useful effect, it will just put the memories into self-refresh mode, and then after some time leave self-refresh mode and go back into normal operation.

1.3.13 Internal assertion error flags

There are internal assertion error flags in the back-end status register, that are set when an internally inconsistent state is detected. These are intended mainly for use as assertions in development (for checking after tests) and for debugging malfunctioning systems. The conditions for these flags should never occur under correct operation, and if one of the conditions have occurred then a full system reset is required to get back to a known good state. A brief description of each flag is provided in table 5 below.

Table 5. Internal assertion error flags for back-end

Bit	Description summary
5	Back-end asserted write into status FIFO when FIFO signaled full
4	Back-end asserted write into prefetch data FIFO when FIFO signaled full.
3	Back-end asserted write into read data FIFO when FIFO signaled full
2	Write command received from front-end but expected amount of write data not in write data FIFO.
1	Read data valid received from PHY when no read data has been requested.
0	Currently unused (always zero)

1.4 Front-end functional description

1.4.1 AHB slave ports

Each port implements an AMBA 2.0 AHB compliant slave interface. The AHB interface can have either 32, 64 or 128 bits width on the data bus, selected through the ahbbits VHDL generic.

The GRLIB version of the controller wraps the AHB interface into the GRLIB AMBA record format and also includes the additional sideband signals used for the GRLIB plug-n-play extension to AMBA.

The memory mapped area supports all types of accesses, however some result in suboptimal performance due to read-modify-write cycles or emulating bursts using single cycle accesses or shorter bursts internally. For optimum performance, the following rules should be followed:

- Accesses should be made using the full width of the AHB data bus as HSIZE.
- Wrapping bursts should not be used.

1.4.2 Response patterns

The slave port will respond with wait states using the HREADY signal whenever needed. Retry and split responses are never used.

ERROR responses are used when accessing outside the available memory range, and also in case of EDAC uncorrectable error. The error responses can be disabled using configuration register bits.

1.4.3 Memory read and write

When reading from the memory, the slave port will give wait states until the data is available and then deliver the data streamed out of the internal FIFO as soon as it is ready. The master should be prepared to handle wait states issued at any time within the burst.

When writing to memory, the controller will normally accept the write (single or burst) without wait-states, however if the command FIFO is full then it will issue wait states.

When EDAC is enabled, writes smaller than the codeword size will result in read-modify-write cycles. These are queued internally and will not cause wait states unless the internal FIFOs are full. If the read part of the RMW cycle detects an uncorrectable error, the memory location will be left unmodified and the error will be logged in the core’s register and an interrupt will be issued.

1.4.4 Write buffering

Normally any number of writes up to the capacity of the internal buffer can be accepted, however through a configuration register, a maximum limit can be set on the number of queued writes in the buffer, where further accesses will cause wait states. At a minimum, the limit can be set to allow only one buffered write.

The amount of buffering should not be more than needed to mask the latency through the controller and to cover AHB bus timing jitter (unrelated accesses causing gaps in the write stream). Increasing the buffering beyond this will only result in degrading worst-case read latency without improving throughput.

1.4.5 Read pre-fetching

To better support masters streaming long packets of data without requiring very long burst lengths, the AHB slave port supports a read pre-fetching feature. When a read burst is made, the port will queue up one or more additional number of bursts of the same length to the back-end and store these in a prefetch buffer when the data arrives. When the same master comes back and requests the following data, it will then serve this from the prefetch buffer and instead request an additional read burst after the currently prefetched point.

The controller has separate tracking registers for each master in order to detect when prefetching should be activated. The prefetching is activated when the same master makes three read bursts that are back-to-back in terms of address. A prefetch master mask configuration register allows to limit which masters can activate the pre-fetching logic so that random access masters such as caches can be prevented from activating it.

Only one prefetching stream can be ongoing at the same time on each AHB port. The current prefetch is cancelled when the currently prefetching master makes an access which does not match the continued stream, or when a prefetch time-out counter expires without the master coming in and requesting data from the next address in the stream. If a write to the same row as is being prefetched is performed on the same AHB port by any master then the prefetch data is discarded in order to prevent serving out-of-date data.

The prefetching continues only to the end of the row and does not cross across row boundaries. It will need to be re-activated on the following row in the same way as described above.

The number of additional read bursts to prefetch is controlled through a configuration register.

1.4.6 Configuration area

Each slave port provides two separate memory areas, one memory mapped area for regular access and one register area for configuration register access.

In the stand-alone version, the ahb_hsel_reg signal selects if memory or registers are accessed. This signal is only considered when the ahb_hsel signal is also asserted on the same cycle. In the GRLIB version, the two AHB areas to two different BARs of the slave similar to other memory controller IP in the library.

The configuration register area must always only be accessed using 32-bit wide accesses, however both single accesses and bursts are permitted.

Any port can access all of the configuration registers, and the register interface is internally arbitrated between the ports.

1.4.7 Multi-ported configuration

In multi-ported configuration, multiple AHB slave ports will be presented. The slave ports are logically separated so they can be either tied to the same bus, or to different AHB buses.

1.5 Error detection and correction features

1.5.1 Overview

The memory controller can be configured to support bit-error tolerant operation by choosing a suitable external memory configuration (see section 1.2.1). In this mode, the DDR data bus is widened and the extra bits are used to store 32 checkbits corresponding to each 64 bit data word.

When writing, the controller generates the checkbits and stores them along with the data on the added part of the data bus. When reading, the controller will transparently correct any correctable bit errors and provide the corrected data on the AHB bus. An extra corrected error output signal is asserted when a correctable read error occurs, at the same cycle as the corrected data is delivered. This can be connected to an interrupt input or to a memory scrubber.

A scrubber function is built into the core that periodically reads from incrementing addresses and writes the data back in case a correctable error is detected. The rate of the scrubbing is programmable and it can also be disabled. The core can also be configured to automatically write back to memory any correctable error that is detected on read.

When performing a write smaller than the codeword size of 64 bits, the controller will automatically perform a read-correct-modify-write cycle to correctly update the checkbits.

1.5.2 Correctable error handling

The intent is for error correction to be transparent to the application, and the controller delivers corrected data with the same timing as uncorrected data. There is no automatic write-back on correctable errors, the controller instead provides a periodic scrubbing function that should be used to prevent error build-up.

A separate ahb_ce output signal is asserted in parallel with the data, this can be used for handling externally to the controller. An internal register holds the address of the correctable error, and an optional interrupt can also be used if desired by the application. All of these features are optional and the controller will work properly even if they are not used.

1.5.3 Uncorrectable error handling

Uncorrectable errors occur when reading a data and checkbit combination that does not match any possible data with any possible combination of errors. To trigger an uncorrectable error, three bytes or more of the data bus must be incorrect (in the default configuration, see table 1 for other configurations). Note that the design intent of the controller with features such as strong EDAC, scrubbing, and SEFI handling is to make the risk of uncorrectable errors occurring as low as possible.

In case of uncorrectable error, this is by default signaled by giving an AHB error response to the master. This behavior can be disabled by configuration and the core will in that case “respond in form” with the corrupted data. A separate sideband signal for uncorrectable errors is also available for custom handling at the SoC level. An uncorrectable error register is included in the core where software can see where the last (if any) uncorrectable error occurred, for diagnosis.

In case of an uncorrectable error detected during a read-modify-write cycle, the modify-write part is cancelled and the original contents of the address (containing error) is left untouched, to be found later by the scrubber or a read access.

1.5.4 EDAC details

The implemented EDAC is based on a Reed-Solomon code implemented over the GF2^4 Galois field, with the capability to correct two symbol errors, one symbol corresponding to four binary bits of data. The code has theoretically 11 data symbols and 4 check symbols, however it is shortened to 8 data symbols and 4 check symbols.

The field generating polynomial for the Galois field is a^ 4+a+1, and each nibble of data in and out of the controller is interpreted as powers of alpha, so {b3,b2,b1,b0} = b3*a^3+b2*a^2+b1*a^1+b0. The Reed-Solomon code in this field is defined by the generating polynomial (x+a^6)(x+a^7)(x+a^8)(x+a^9).

To support correction of byte-wide errors, the code is interleaved by a factor of two, resulting in a code with 2*8 data symbols * 4 bit = 64 data bits and 2*4 check symbols * 4 bits/symbol = 32 check bits. The interleaved code can then correct two different byte-wide errors in the data.

1.5.5 Internal self-checking

The external memory EDAC contains an internal self-check function where the data after decoding is re-encoded and compared with the original read-in data and checkbits. If they differ in some other location than one where an error was corrected, then the EDAC will signal an internal consistency error.

The core can be configured to automatically retry the read transfer when this happens. If the retry is disabled, or if there is another consistency error when the retry is made, this will be treated as an uncorrectable error (section 1.5.3) and in addition a consistency error IRQ is signaled.

1.5.6 Diagnostic interface

The controller supports a diagnostic interface where the checkbits and data bits can be read and written separately with the EDAC bypassed to allow error injection. Accesses to the diagnostic area can be freely mixed with normal memory access which simplifies many test scenarios. Further details are in the back-end description, section 1.3.9.

1.5.7 Initialization

When EDAC is used, the memory must be initialized with correct checkbits. An initialization function that clears the entire memory is implemented in the front-end. This can be set up to initialize with a custom pattern, or to initialize each address with the row/column number of that address.

The controller can be set up to perform a read back immediately after the initialization to verify that the expected data is returned.

1.5.8 Scrubbing

The controller has a built-in scrubber implemented in the front-end side. This reads through all of the memory at a programmable rate, and if any correctable errors are detected, then the corrected data is written back to the same address. The scrubber is also used as the basis for the SEFI handling functionality.

1.5.9 Correctable error counting

There are two separate sets of counters that count statistics on correctable errors, the byte lane error counters, and the address error counters. Both of these will be updated whenever the scrubber finds a correctable error. Note that these are only updated by scrubber accesses and not updated when a regular access triggers a correctable error. The reason for that is to avoid large up-counts if the same error location is read multiple times, which could interfere with the statistics and falsely trigger the SEFI detection logic.

1.5.10 Byte lane error counters

The controller has 12 wrapping counters that track the number of correctable errors that the scrubber finds in each byte of the DDR data bus, plus an additional counter called the watermark that tracks the value of the slowest moving counter. These counters are combined to implement a statistical SEFI detection scheme, whích triggers when the byte lane with the most correctable errors has 64 errors more than the byte lane with the least amount of correctable errors.

The watermark is implemented simply by starting at zero after reset like the other counters, and each cycle checking if it is equal to any of the 12 byte lane counters. If the watermark is not equal to any of the counters, then the slowest moving counter must have incremented and the watermark is also incremented to track it.

If one byte lane has a much higher frequency of errors than the others, then that byte lane's error counter will eventually wrap around and count up to the watermark value. That is used as an indication that a SEFI has occurred on that byte lane, which is managed by the SEFI handling state machine.

An equivalent scheme would be to have byte lane counters where you subtract one from each counter once all counters are non-zero and trigger when one counter reaches 64. Each counter minus the watermark (modulo 64) can be viewed as such a counter.

Note that these counters only count errors found by the scrubber, in order to guarantee proportional checking of errors across the whole address space, as otherwise repeated reads of one address with a correctable error could quickly increment a specific counter.

1.5.11 SEFI handling

A special state machine is implemented to attempt SEFI recovery when detected by the byte lane error counters. Note that due to the powerful EDAC and other features implemented in the controller, a single SEFI plus additional SEUs in parallel can be handled implicitly by the controller, the purpose of the SEFI recovery features is to achieve a faster repair in case of recoverable SEFI. The SEFI recovery actions can be disabled in the controller if unwanted, and the SEFI detection logic can then be used in a passive mode for diagnostic purposes instead.

When a SEFI is detected using the byte lane error counters, the SEFI handler it will first request a recalibration of the memory, and a re-write of the memory mode registers from the controller back-end. When this is complete, the scrubber will be setup to perform an accelerated scrub run over the whole memory range in order to quickly regenerate the check bits in case the SEFI was temporary.

After the regeneration, the scrubber will again run in the normal pace. If a full cycle (from start adddress to end address) of the scrubber is performed without the SEFI detection triggering on the byte lane that had the SEFI, then it is considered to be 'healed' from the SEFI. If the SEFI detection again triggered, it is considered to have a permanent SEFI. When that happens, the controller will issue a permanent SEFI interrupt. and can also optionally be configured to disable the output drivers for that byte-lane in the backend. The controller will continue tracking errors in the byte lane and may find it to have healed at a later stage.

1.5.12 PHY-based SEFI detection

The controller supports an optional sideband signal from the PHY indicating that it had a read failure on a byte lane, if available in the implementation then this can be used to trigger the SEFI handling functionality independently of the error counting. A dedicated interrupt also exists for this condition (back-end error signaled by PHY).

1.5.13 Address error counters

A second set of correctable error counters updated by the scrubber are also provided. These are saturating signed counters that will increase or decrease depending on which address the error occurred on. One counter is implemented for each address bit in the memory address space (CS/bank/row/column). If a correctable error is found where the corresponding address bit is 0, that counter is decremented, while if the address bit was 1, that counter is incremented. As long as errors are occurring randomly over the address space, these counters will vary randomly around zero, but if more errors happens when an address bit has one value, the corresponding counter will tend towards its maximum or minimum value.

The main purpose of these counters are for identifying the likely address when a SEFI is detected, and also to collect statistics over time to detect if some parts of memory are weaker than others.

1.6 Front-end to back-end interface implementation

1.6.1 Command FIFO format

Each port of the front-end sends commands to the back-end using a FIFO. The command words of the FIFO have been designed to match closely (but not exactly) with the commands going out to the memory.

The command words containing the following fields:

- Command ID (4-bit tag)
- 3-bit command word, similar to the (RASn,CASn,WEn) signals of the DDR memory interface
- Chip-select ID (number of bits depending on number of CS supported)
- Bank address bits not implied by port ID

- Address bits (16 bits)
- A special read-modify-write flag (1 bit)
- Prefetch flag (1 bit)

1.6.2 Commands

In normal operation, the commands row-open (011), read (101), write (100), and row-close (010) are issued.

The row-open command opens a specific CS, bank and row. Note that each port can only have one row open. If a new row-open command comes in when a port is open, the other open row will first be closed by the back-end.

The read command causes a read burst to be performed by the back-end. A row must have been selected already using the row-open command. The read out data is normally written into the read-data FIFO, unless the RMW flag or the prefetch flag is set. Note that the CS and bank fields are ignored for this command (these fields have special use for precharging, see below)

The write command performs a write burst to memory, analogous to the read command. The write data must have been placed in the write data FIFO before sending this command.

The row close command will close the currently opened row for the port, if no port is opened it will have no effect. Note that since the back-end does not require this between row-open commands, this is only needed when the port is idle to ensure that the row is not kept opened indefinitely. This also gives the front-end responsibility to manage the row-open policy of the controller.

Having a separate command to select/open the row has some benefits for performance. The front-end can be optimized to transfer the CS,bank,row immediately, and other work such as determining the exact column and transferring write data into the FIFO can be done in parallel with the back-end opning the row.

1.6.3 Response FIFO format

The response FIFO provides a response word for each command. This consists of:

- Command ID (4-bit)
- Result code (3-bit)
- Correctable error byte lane mask (12-bit)

The result code can be one of: OK (000), uncorrectable error (001), EDAC consistency error (010), TBD (011,100,101,110), internal fault detected (111)

1.6.4 Read-data FIFO format

The read-data FIFO words consists of:

- Data word 128-bit
- Correctable error mask (48-bit)
- Uncorrectable error indicator (2-bit)
- Last of burst indicator (1 bit)

Note that the data word is always 128 bits regardless of the width setting of the DDR memory, therefore each read burst will generate 2 or 4 words in the FIFO.

Which byte lanes had the correctable errors can be seen in the response word.

1.6.5 Write-data FIFO format

The write-data FIFO words consists of:

- Data word 128-bit
- Data byte valid 16-bit

The data word is always 128 bits wide regardless of the width setting of the DDR memory, therefore each write command must have 2 or 4 words in this FIFO corresponding to the data. The front-end must always pad data to full bursts by adding words with all valid bits set to 0.

1.6.6 Read-modify-write accesses

The controller allows read-modify-write accesses (8/16/32-bit writes) to be queued up internally without stalling the AHB bus. They are performed without doing round-trips up to the AHB front-end which allows them to be done much more efficiently.

Read-modify-write accesses are commanded by:

1. in the command FIFO, put a row-open command to the command FIFO (unless the desired row is already open)

2. in the command FIFO, put a read command for the desired column with the RMW flag bit set to 1. This will make the back-end store the first word of the read burst into an internal buffer.

3. in the write data FIFO, store the partial codeword with corresponding mask as the first word of the write burst, and then pad with additional words with all bytes masked to make a full burst. (this can be done in parallel with steps 1 and 2)

4. in the command FIFO, put a write command to the same column with RMW flag set to 0.

5. once the RMW is completed a result word will be put into the response FIFO

1.6.7 Prefetching

In order to command a prefetch, set the prefetch bit to 1 in the commmand word. The read out data will not be stored in the read data FIFO but in a separate two-port prefetch buffer RAM. The RAM might hold multiple bursts, CS,Bank and the top bits of the address part of the read command word that are otherwise ignored are used as address bits into the prefetch buffer in this case.

1.6.8 Register interface

The configuration registers in the back-end are accessed using a simple handshaked interface, allowing one write or one read at a time. Only one register interface is implemented and is internally multiplexed in the front-end between the user ports and the SEFI handler state machine.

1.7 Implementation

1.7.1 Code structure

The core can be built either as part of a GRLIB design, or stand-alone. For this there are two different top level entities, ftaddr_gr which uses the GRLIB AMBA record, and the techmap library for FIFOs and buffers, and ftaddr_sa, which is a stand-alone implementation that only depends on the IEEE standard libraries and uses a custom technology-specific VHDL file for FIFOs.

The main part of the controller logic is shared between the two versions and are contained in the entities ftaddr_fe_ahb (AHB-clocked part), ftaddr_be (DFI-clocked part), ftaddr_edac (EDAC sub-block of ftaddr_be) and the package ftaddr_int.

The GRLIB top level instantiates the syncfifo_2p component from the techmap library to implement FIFOs, and syncram_2p to implement dual-ported memory. To select technology you pass in a tech generic.

The stand-alone top level instead instantiates a block called ftaddr_sa_fifo to implement the FIFOs. The ftaddr_sa_fifo entity can either implement a FIFO based on flip flops or on RAM blocks. When

RAM blocks are selected then those are instantitated from the ftaddr_sa_techspec_fifobuf entity, this needs to be provided for the target technology.

When building the main top-level or one of the standard logic wrapper together with the RTL fifo with flip flops, the IP core has no external dependencies, except for the standard IEEE VHDL libraries. In other configurations it will depend on the core GRLIB libraries grlib.stdlib, grlib.amba and techmap.gencomp libraries.

An additional wrapper also exists called ftaddr_sawrap, this takes the stand-alone version and gives it the same interface as the GRLIB version. The purpose is if the exact same code of the stand-alone version is desired inside a GRLIB design.

1.7.2 Clocking and reset

The controller takes two clocks as input, one AHB clock used to clock the front-end ports, and one DFI clock used to clock the controller back-end. The inputs and output signals of the core all belong to one of these two clock domains.

The core has two synchronous reset inputs, one for each clock domain. The two should both be asserted for a number of cycles in both domains, long enough to reset each domain and then to allow all synchronization registers to stabilize. After that the resets can be raised in any order.

A dynsync signal is included in the controller that allows skipping a synchronization stage if the clock domains are synchronous to each other. Note that this should not change during operation. It is currently only supported in the ftaddr_sa_fifo stand-alone implementation, for other implementations it is ignored and full synchronization is always used. Note also that using dynsync requires synchronous constraining between the clock domains.

1.7.3 Buffer memories, stand-alone version

The internal buffers consists in both directions of a control (command or response) asynchronous FIFO and a data FIFO. In the read direction there is also an additional FIFO for the read prefetch.

In the stand-alone version, the command, response and read-data FIFOs are based on flip flops, while write and prefetch FIFOs are based on SRAM.

The command and read-data FIFOs are implemented using flip-flops with a minimum-latency scheme so that the data is ready on the same cycle that the write counter has been transferred instead of requiring an extra cycle for reading the buffer as is required for a standard SRAM based FIFO. An additional cycle can be removed from these FIFOs if the clock domains are synchronous, using the dynsync feature.

The response FIFO is implemented with flip-flops but using a standard scheme with an extra cycle for reading from the buffer. The write data and prefetch data FIFOs are implemented with SRAM using a traditional asynchronous FIFO approach.

The FIFOs must be of the first-word-fall-through type, where data and valid come out on the same cycle.

1.7.4 Guard gates, stand-alone version

Guard gates to prevent glitches propagating between the clock domains are implemented with the ftaddr_sa_cdcguard, in turn depending on the ftaddr_sa_nand2 block, which is a two-input NAND model. In order to avoid this from being optimized out, the command “set_boundary_optimization [get_designs ftaddr_sa_nand2] false” can be used in Design Compiler.

1.7.5 EDAC pipelining

Extra EDAC pipeline stages are currently not supported, as the pipelining currerntly implemented satisfies the performance requirements.

1.7.6 Clock gating

The IP core is designed with a design style that allows for automatic clock gate insertion by the synthesis tool.

1.7.7 Performance and area

Trial synthesis of the controller was done to C65SPACE using DC Topo with configuration 4 ports, 128 bits AHB, stand-alone version with all features included, black boxed SRAMs. This met timing at 400 MHz frequency on both AHB and DDR domains and produced a standard cell area (not including the SRAM) of 0.46 mm^2, and at 550 MHz frequency with an area of 0.51 mm^2

The controller has also been built with all features including EDAC, for an Altera Stratix4 FPGA with AHB clock 100 MHz and AFI/DDR clock 125 MHz.

1.7.8 Porting to other technology, stand-alone version

There are several possible strategies for porting the controller to a new technology.

The simplest strategy is to build the controller with only flip flops for memories using the dffonly generic, it then becomes inherently portable to any technology by changing the target library of the synthesis tool.

The recommended strategy for porting the controller using SRAM buffers, is to take the existing c65space technology specific implementation (ftaddr_sa_techspec_c65space.vhd) and modifying the RAM instantiations to equivalent RAMs in the target technology. The error correction implemented in this file can tolerate at most one error per RAM addreess.

1.7.9 Porting to new technology, GRLIB version

The GRLIB version is ported to a new technology by implementing a technology mapping for the desired technology and then changing the controllers memtech generic to use this techmap.

1.8 PHY specific implementation characteristics

1.8.1 Generic DFI implementation

The generic DFI implementation is intended to interface a DFI2.1.1 compliant DDR2/DDR3 PHY with the following limitations:

- The PHY must use a 1:1 frequency ratio DFI interface
- The PHY does not use “MC evaluation” mode for training. Note that it is impossible to implement a generic MC evaluation calibration algorithm since the meaning of the delays are not defined in DFI. For using MC evaluation mode, a PHY specific implementation is required.
- The DFI timing parameters are compatible. The timing parameters for the controller are listed in table 9.

The trden, twrlat and twrdata timings that the PHY supports are supplied to the controller through VHDL generics (genphy_trden, genphy_twrlat, and genphy_twrdata). The timings can be set either absolut or relative to the CAS latency depending on what the PHY requires so that they will always be correctly set when updating the CAS latency. The timings can also be overridden from software at run time through the PHY timing registers.

The update interfaces (both MC initiated and PHY initiated) are supported by the controller. The MC update request is asserted while waiting for periodic refresh command to finish.

Note that the DFI parity buses are not used, the regular data bus is used also for ECC/parity bits.

The controller offers some optional extra signals that may be used if the PHY supports it, or may be ignored otherwise:

- xdfi_term_en - Usable to enable local termination on the controller side during reads. Active high, default not used (always 0). The timing and usage of this signal is controlled via the ODT internal timing register.
- xdfi_rderr - Indicate that data on a specific byte lane was not received. Tie to all-0 if not used.
- xdfi_softrst - Active high “soft-reset” to reset the state machines of the PHY to idle state while idle.
- xdfi_phyctrl - Generic control signals tied to PHY generic control signal register.

If desired and supported by the PHY, the control and address signal buses may be duplicated throught the ctrldup and csdup generics

1.8.2 ISD65 PHY specific implementation

The ISD65 PHY specific implementation is tuned for the PHY for C65SPACE developed by ISD S.A.

This implementation adds calibration algorithms for gate training, write leveling, and read leveling. Note that the PHY has separated ports for gate coarse delay and gate fine delay. The gate coarse delay should be connected to the 6 most significant bits, and the gate fine delay to the 6 least significant bits of dfi_rdlvl_gate_delay for each byte lane.

This implementation does not use generics for DFI timings, but has hard-coded default values to match the PHY. They may still be overridden by software.

When using this implementation, the generics to the controller shall be set according to table 6 and non-DFI signals shall be connected according to table 7.

Calibration for the ISD65 PHY is described in section 1.16.

Table 6. Required VHDL generic settings for ISD65 PHY implementation

Generic	Setting	Comment
phyimpl	1	ISD65 constant
ddrbits	96	96-bit wide interface only supported
numrwen	1	
numrdlvlphy	1	
numrdlvlmc	2	
numwrlvlphy	1	
numwrlvlmc	12	
rdblvlbits	48	6 per data bit
rdglvlbits	12	6 bits coarse delay, 6 bits fine delay
wrlvlbits	6	
phyctrlbits	11	May be set higher for system-specific extra bits

Table 7. Non-DFI port connections for ISD65 PHY implementation

FTADDR Port	Connection on PHY	Comment
xdfi_term_en	i_odt_en(N)	Drive all bits on PHY input bus with this value
xdfi_rderr	o_rd_err	
xdfi_softrst	Not connected	
xdfi_phyctrl(0)	i_sync_clk_edge_mode(N)	Drive all bits on PHY input bus with this value
xdfi_phyctrl(1)	i_sync_clk_edge(N)	Drive all bits on PHY input bus with this value
xdfi_phyctrl(6:2)	i_sync_delta(5N+4 : 5N)	Drive same value for each byte lane
xdfi_phyctrl(7)	i_pdt_rtt_high(N)	Drive all bits on PHY input bus with this value
xdfi_phyctrl(8)	i_proga(N)	
xdfi_phyctrl(9)	i_progb(N)	
xdfi_phyctrl(10)	i_loopback_en	
xdfi_bl_resetn	Not connected	Memory reboot not supported
xdfi_bl_cke	Not connected	Memory reboot not supported
xdfi_bl_pdn	Not connected	Memory reboot not supported
xdfi_clk_zero	Not connected	Memory reboot not supported

1.8.3 Altera UniPhy implementation

The Altera implementation is intended for FPGA prototyping. It has been tested on Stratix4 with DDR2 memory.

The PHY is built using the Quartus II “DDR2 SDRAM Controller with UniPHY” MegaWizard flow, with the “Generate PHY only” option. It must be built with 1:1 frequency ratio (“Rate on Avalon-MM interface” set to “Full”). The burst length must be set to 8.

Some specific precautions need to be taken with this PHY:

- The controller should not reprogram the mode registes, and in particular not change the CAS latency. All period mode register reprogramming functions must be disabled, and the CAS latency setting in the controller must be set to the same as set up when building the PHY.
- Accessing a non-implemented rank will result in the PHY locking up. This is avoided by setting up the NCS field properly in the AHB address decode register and enabling out-of-range AHB error.

The interface used is AFI which is very similar to DFI. Two dedicated ports are added to the controller to support this PHY, afi_dqs_burst and afi_wlat.

The connections are listed in table 8.

Table 8. Port connections for UNIPHY implementation

FTADDR port	Connection on PHY	Comment
dfi_cs_n	afi_cs_n	
dfi_bank	afi_ba	
dfi_address	afi_addr	Connect lowest bits if afi_addr width lower
dfi_ras_n	afi_ras_n	
dfi_cas_n	afi_cas_n	
dfi_we_n	afi_we_n	
dfi_cke	afi_cke	
dfi_odt	afi_odt	
dfi_reset_n	Not connected	
dfi_wrdata	afi_wdata	
dfi_wrdata_en(0)	afi_wdata_valid	Connect bit 0 of wrdata_en to all bits
dfi_wrdata_mask	afi_dm	
dfi_rddata_en	afi_rdata_en, afi_rdata_en_full	
dfi_rddata	afi_rdata	
dfi_rddata_valid	afi_rdata_valid	
dfi_ctrlupd_ack	Not connected	Feed constant 0 into controller
dfi_phyupd_req	Not connected	Feed constant 0 into controller
dfi_phyupd_type	Not connected	Feed constant “00” into controller
dfi_dram_clk_disable	dram_clk_disable	
dfi_init_complete	afi_cal_success	
dfi_rdlvl_mode	Not connected	Feed constant 0 into controller
dfi_rdlvl_req	Not connected	Feed constant 0 into controller
dfi_rdlvl_gate_mode	Not connected	Feed constant 0 into controller
dfi_rdlvl_gate_req	Not connected	Feed constant 0 into controller
dfi_rdlvl_resp	Not connected	Feed constant 0 into controller
dfi_wrlvl_mode	Not connected	Feed constant 0 into controller
dfi_wrlvl_req	Not connected	Feed constant 0 into controller
dfi_wrlvl_resp	Not connected	Feed constant 0 into controller
xdfi_term_en	Not connected	
xdfi_rderr	Not connected	Feed constant 0 into controller
xdfi_softrst	Not connected	
xdfi_phyctrl	Not connected	
afi_dqs_burst	afi_dqs_burst	
afi_wlat	afi_wlat	

1.9 DFI Spec Sheet

Below is a specification sheet of the controller in the same format as section 7.0 of the DFI specification. (note that version 2.1.1 of the specification has an error where min and max have been swapped).

Table 9. DFI Settings specification for FTADDR controller

Terms	Min	Max	Comment
DFI clock frequency	N/A		Synchronous logic, max frequency depends on implementation technology. 1:1 clock ratio.
DFI Address width	16	16	
DFI Bank Width	3	3	
DFI Control Width	1	1	
DFI Chip Select Width	1	8	Selected via numcs generic
DFI Data Width	64	256	Selected via ddrbits generic (times 2). Since maximum 2*96 bits are used, additional data bits are padded with zero/ignored.
DFI Data Enable Width	1	-	Set via numrwen generic, all bits driven to same value.
DFI Read Data Valid Width	1	-	Set via numrwen generic, only bit 0 is used.
DFI Read Leveling Delay Width	1	-	Set via rdblvlbits generic.
DFI Read Leveling Gate Delay Width	1	-	Set via rdglvlbits generic
DFI Read Leveling MC IF Width	1	-	Set via numrdlvlmc generic
DFI Read Leveling PHY IF Width	1	-	Set via numrdlvlphy generic
DFI Read Leveling Response Width	32	96	Selected via ddrbits generic (always same width as data bus)
DFI Write Leveling Delay Width	1	-	Set via wrlvlbits generic
DFI Write Leveling MC IF Width	1	-	Set via numwrlvlmc generic
DFI Write LEveling PHY IF Width	1	-	Set via numwrlvlphy generic
DFI Write Leveling Response Width	32	96	Selected via ddrbits generic (always same width as data bus)

Table 10. Timing parameter settings for FTADDR controller

Parameter	Min	Max	Comment
t_{ctrl_delay}	0	5	
t_{phy_wrdata}	0	11-tphy_wrlat	Set via genphy_twrdata generic
$t_{phy\_wrdelay}$	N/A		Frequency ratio systems only
t_{phy_wrlat}	0	11	Set via genphy_twrlat generic
t_{phy_rdlat}	1	-	Controller waits indefinitely for read data. System level limitations for refresh etc.
t_{rddata_en}	0	11	Set via genphy_trden generic
$t_{ctrlupd\_interval}$	500	-	Depends on refresh rate settings, one update offer per refresh.
$t_{ctrlupd\_min}$	3	514	Depends on T_{RFC} setting of controller
$t_{ctrlupd\_max}$	3	514	Depends on T_{RFC} setting of controller
t_{phyupd_type0}	0	-	
t_{phyupd_type1}	0	-	
t_{phyupd_type2}	0	-	

Table 10. Timing parameter settings for FTADDR controller

Parameter	Min	Max	Comment
t_{phyupd_type3}	0	-	
t_{phyupd_resp}	100	-	System dependent. Controller will in worst case wait for any already issued read or write commands followed by a precharge all to complete.
$t_{dram\_clk\_disable}$	0	10	
$t_{dram\_clk\_enable}$	0	10	
$t_{init\_complete}$	N/A		dfi_init_start not implemented
t_{init_start}	N/A		dfi_init_start not implemented
$t_{phy\_paritylat}$	N/A		dfi_init_start not implemented
t_{rdlvl_dll}	N/A		Only for MC evaluation mode
t_{rdlvl_en}	1	16	
t_{rdlvl_load}	N/A		Only for MC evaluation mode
t_{rdlvl_max}	-	-	Controller waits indefinitely for rdlvl_resp
t_{rdlvl_resp}	100	-	System dependent. Controller will in worst case wait for any already issued read or write commands followed by a precharge all to complete.
$t_{rdlvl\_resplat}$	1	15	
t_{rdlvl_rr}	16	16	
t_{wrlvl_dll}	N/A		Only for MC evaluation mode
t_{wrlvl_en}	1	16	
t_{wrlvl_load}	N/A		Only for MC evaluation mode
t_{wrlvl_max}	-	-	Controller waits indefinitely for wrlvl_resp
t_{wrlvl_resp}	100	-	System dependent. Controller will in worst case wait for any already issued read or write commands followed by a precharge all to complete.
$t_{wrlvl\_resplat}$	1	14	
t_{wrlvl_ww}	15	15	
t_{lp_resp}	N/A		Low-power interface not implemented
t_{lp_wakeup}	N/A		Low-power interface not implemented

1.10 Registers

The controller provides a 1024 byte memory-mapped register area. This is split into two halves, the first for configuration registers residing in the back-end, and the second for configuration registers residing in the front-end. For forward compatibility, reserved fields should be written either with 0 or with the last read-out value, and reserved registers should not be accessed at all.

Table 11. Configuration registers for FTADDR controller

Register address offset	Name	R/W	Reset value (dynrst=0)	Notes
0x000	Feature set register	R	*	1
0x004	Back-end status register	R	0x00000000	
0x008	Memory configuration register 1	R/W	0x808FFFFF	
0x00C	Memory configuration register 2	R/W	0xFFFFFFC0	
0x010	Memory configuration register 3	R/W	0x00076475	
0x014	Memory configuration register 4	R/W	0x00000009	
0x018	Service configuration register 1	R/W	0x0FF01512	
0x01C	Service configuration register 2	R/W	0x1900FFFF	
0x020	PHY timing register 1	R/W	*	1
0x024	PHY timing register 2	R/W	*	1
0x028	Diagnostic control register 1	R/W	*	2
0x02C	Diagnostic control register 2	R/W	*	2
0x030	Diagnostic status register	R	*	2
0x034	Diagnostic checkbit register	R/W	*	2
0x038	Diagnostic data register 1	R/W	*	2
0x03C	Diagnostic data register 2	R/W	*	2
0x040	ODT configuration register, rank #0	R/W	0x00000000	
0x044	ODT configuration register, rank #1	R/W	0x00000000	3
0x048	ODT configuration register, rank #2	R/W	0x00000000	3
0x04C	ODT configuration register, rank #3	R/W	0x00000000	3
0x050	ODT configuration register, rank #4	R/W	0x00000000	3
0x054	ODT configuration register, rank #5	R/W	0x00000000	3
0x058	ODT configuration register, rank #6	R/W	0x00000000	3
0x05C	ODT configuration register, rank #7	R/W	0x00000000	3
0x060	ODT external timing register	R/W	0x00008006	
0x064	ODT internal timing register	R/W	0x00000000	
0x068	Command register	W	0x00000000	
0x06C	Sleep mode configuration register	R/W	0x00000000	
0x070	Back-end EDAC configuration register	R/W	0x80000000	
0x074	Service time counter register	R/W	0x0FF340FF	4
0x078	PHY indirect address register	R/W	0x00000000	
0x07C	PHY indirect data register	R/W	*	1
0x080	PHY generic control register	R/W	*	1
0x084	Training time counter register	R/W	0xFFFF0200	4
0x088	Back-end FIFO error counter register	R/W	*	2
0x08C-0x1FC	RESERVED			
0x200	AHB address decode register	R/W	0x0000E6AA	
0x204	AHB access configuration register	R/W	0x4200000F	

Table 11. Configuration registers for FTADDR controller

Register address offset	Name	R/W	Reset value (dynrst=0)	Notes
0x208	Prefetch configuration register	R/W	0x03FFFFFF	
0x20C	Scrubber configuration register 1	R/W	0x00000000	
0x210	Scrubber configuration register 2	R/W	0x0000FFFF	
0x214	IRQ pending register	R/W	0x00000000	
0x218	IRQ enable register	R/W	0x00000000	
0x21C	Scrubber UE error register	R/W	*	2
0x220	Scrubber CE byte lane counter register 1	R	0x00000000	
0x224	Scrubber CE byte lane counter register 2	R	0x00000000	
0x228	Scrubber CE byte lane counter register 3	R	0x00000000	
0x22C	Scrubber CE byte lane counter register 4	R	0x00000000	
0x230	Scrubber CE address counter register 1	R	*	2
0x234	Scrubber CE address counter register 2	R	*	2
0x238	Scrubber CE address counter register 3	R	*	2
0x23C	Scrubber CE address counter register 4	R	*	2
0x240	Access CE location register, port #0	R	*	2
0x244	Access UE location register, port #0	R	*	2
0x248	Access CE location register, port #1	R	*	2, 3
0x24C	Access UE location register, port #1	R	*	2, 3
0x250	Access CE location register, port #2	R	*	2, 3
0x254	Access UE location register,port #2	R	*	2, 3
0x258	Access CE location register, port #3	R	*	2, 3
0x25C	Access UE location register, port #3	R	*	2, 3
0x260	Prefetch status register, port #0	R	*	2
0x264	Prefetch status register, port #1	R	*	2, 3
0x268	Prefetch status register, port #2	R	*	2, 3
0x26C	Prefetch status register, port #3	R	*	2, 3
0x270	Prefetch bank/CS register	R	*	2
0x274	Scrubber start address register	R/W	0x00000000	
0x278	Scrubber end address register	R/W	0x73FFFFFF	
0x27C	Front-end FIFO error counter register	R/W	*	2
0x280	Init pattern register 1	R/W	0x00000000	
0x284	Init pattern register 2	R/W	0x00000000	
0x288	Init pattern register 3	R/W	0x00000000	
0x28C	Init pattern register 4	R/W	0x00000000	
0x290	Scrubber position register, port #0	R	0x00000000	
0x294	Scrubber position register, port #1	R	0x00000000	3
0x298	Scrubber position register, port #2	R	0x00000000	3
0x29C	Scrubber position register, port #3	R	0x00000000	3
0x2A0-0x3FC	RESERVED			

Note 1: Reset value depends on IP configuration options

Note 2: Some fields in register are not reset

Note 3: Whether register is implemented depends on IP configuration options (port/CS count)

Note 4: Register starts counting directly after releasing reset, read-out value may be different

In addition to the AHB registers, an additional register space is accessible by the PHY indirect address and data registers. This provides access to PHY-specific registers and their address mappings are tabulated in table 12 below.

Table 12. PHY-specific configuration registers for FTADDR controller

Indirect address	Name	R/W	Reset value	Notes
Generic PHY implementation (phyimpl=0)				
0x00-0xFF	RESERVED			
ISD65 PHY implementation (phyimpl=1)				
0x00	Manual calibration register 1	R/W	0x00000000	
0x01	Manual calibration register 2	W	0x00000000	
0x02-0x04	Gate coarse delay register 1-3	R/W	*	1
0x05-0x07	Gate fine delay register 1-3	R/W	*	1
0x08-0x0A	Write leveling delay register 1-3	R/W	*	1
0x0B-0x22	Read delay register 1-24	R/W	*	1
0x23-0x25	Write leveling response register 1-3	R	*	1
0x26-0x28	Read leveling response register 1-3	R	*	1
0x29	MPR bit position register 1	RW	0x00000000	
0x2A	MPR bit position register 2	RW	0x00000000	
0x2B	Training configuration register	RW	0x0000003F	
0x2C-0xFF	RESERVED			
Altera UniPHY implementation (phyimpl=2)				
0x00-0xFF	RESERVED			

Note 1: Some fields in register are not reset

1.10.1 Feature set register

Table 13. 0x000 - FTADDR feature set register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PHYIMPL								RESERVED							
*								0							
r								r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES	ECCS	PARS	NCSMAX			NPORT		DDRWIDTH							
0	1	1	*			*		*							
r	r	r	r			r		r							

31 : 24	PHY implementation ID (PHYIMPL). See section 1.3.11 for possible values.
23 : 15	Reserved
14	ECC support, set to 1 if external memory EDAC is supported
13	Parity support, set to 1 if parity checking mode is supported
12 : 10	Number of chip selects (external banks) supported, minus one.
	0=1 bank, 1= 2 banks, ..., 7 = 8 banks supported
9 : 8	Number of AHB ports on controller, log2 format. 0=1 port, 1=2 ports 2=4 ports, 3=reserved
7 : 0	Width of DDR data bus, in bits.

1.10.2 Backend status register

Table 14. 0x004 - FTADDR backend status register

31	26	25	16
INTERROR		RESERVED	
000000		0	
r		r	

15	5	4	0
RESERVED		BESTATE	
0		*	
r		r	

31 : 26 Internal error. Diagnostic bits set only on data path inconsistency conditions that should ‘never happen’. See section 1.3.13 for further details.

25 : 5 Reserved

4 : 0 Current back-end state

0 = Down, waiting to be enabled

1 = Enabled, waiting for minimum time before beginning init

2 = Performing init sequence

3 = Initial gate training

4 = Initial data eye training

5 = Initial write leveling

6 = Normal state

7 = In service interval

8 = Returning from service interval, reopening rows

9 = Manual mode

10 = Closing rows for entering service interval

11 = Performing auto-refresh

12 = Rewriting mode registers

13 = Incremental gate training

14 = Incremental data eye training

15 = Incremental write leveling

16 = Performing diagnostic access

17 = Performing automatic retry

18 = Memories in self-refresh mode

19 = Performing memory byte lane reboot

20-31 = Reserved (currently unused)

1.10.3 Memory configuration register 1

Table 15. 0x008 - FTADDR memory configuration register 1

31	30	29	28	27	26	25 22	21 20	19 16
DDRT	HWID	ENX4	PWRU	SKIP-CALIB	SKIP-INIT	CASLAT	WLAT	DRAS (6:3)
1	0	0	0	0	0	10	00	1111
rw	rw	rw	rw	rw	rw	rw	rw	rw

15 13	12 6	5 3	2 0
DRAS (2:0)	DRP	DFAW	DRRD
111	1111111	111	111
rw	rw	rw	rw

31 DDR memory type (0=DDR2, 1=DDR3)

30 Half width mode selection (0=full width, 1=half width)

29 Half byte mode selection (0=full byte 1= half byte)

28 Power up (PWRU), set to 1 to start init sequence.

27 Skip calibration, set to 1 before (or at the same time as) setting PWRU to skip calibration

26 Skip initalization, set to 1 before (or at the same time as) setting PWRU to skip initialization

25 : 22 CAS latency (CASLAT), programmed into the mode registers of the memory devices

For DDR2: 0-1=Reserved, 2=CL2, 3=CL3, 4=CL4, 5=CL5, 6=CL6, 7-15=Reserved

For DDR3: 0=Reserved, 1=CL5, 2=CL6, 3=CL7, 4=CL8, 5=CL9, ..., 10=CL14, 11-15=Reserved

21 : 20 CAS write latency for DDR3 (WLAT), setting has no effect when DDR2 is used

0=WL5, 1=WL6, 2=WL7, 3=WL8, 4=WL9, 5=WL10, 6=WL11, 7=WL12

19 : 13 Delay setting for tRAS (DRAS)

12 : 6 Delay setting for tRP (DRP)

5 : 3 Delay setting for tFAW (DFAW)

2 : 0 Delay setting for tRRD (DRRD)

1.10.4 Memory configuration register 2

Table 16. 0x00C - FTADDR memory configuration register 2

31 30	29 25	24 20	19 16
RESERVED	DWTR	DWR	DRTP
0	11111	11111	1111
r	rw	rw	rw

15 12	11 3	2	1	0
DRCD	DRFC	DLLDIS	RDIMM	T2
1111	111111111	0	0	0
rw	rw	rw	rw	rw

31 : 30 Reserved

29 : 25 Delay setting for tWTR (DWTR)

24 : 20 Delay setting for tWR (DWR)

19 : 16 Delay setting for tRTP (DRTP)

15 : 12 Delay setting for tRCD (DRCD)

11 : 3 Delay setting for tRFC (DRFC)

2 DLL disable setting for external memories (1=DLL disabled, 0=DLL enabled)

1 Registered DIMM mode (0=regular memory, 1=registered DIMM)

0 2T signaling mode enable (0=1T signaling, 1=2T signaling)

1.10.5 Memory configuration register 3

Table 17. 0x010 - FTADDR memory configuration register 3

31	30	20	19	16
M3OVR	RESERVED		DRTW	
0	0		*	
rw	r		rw	

15	12	11	8	7	4	3	0
DRTRXCS		DWTWXCS		DRTWXCS		DWTRXCS	
0110		0100		*		*	
rw		rw		rw		rw	

31 Memory configuration register 3 override (M3OVR).

This must be set to 1 to configure the other fields in this register, otherwise they are automatically set to default values as indicated below.

30 : 20 Reserved

19 : 16 Delay between read-to-write commands to same CS, by default set to CL-WL+6

15 : 12 Delay between read-to-read commands to different CS, by default set to 6 cycles

11 : 8 Delay between write-to-write commands to different CS, by default set to 4 cycles

7 : 4 Delay between read-to-write commands to different CS, by default set to CL-WL+6

3 : 0 Delay between write-to-read commands to different CS, by default set to WL-CL+6

1.10.6 Memory configuration register 4

Table 18. 0x014 - FTADDR memory configuration register 4

31	20	19	16
DATA BYTE DISABLE		RESERVED	
000000000000		0	
rw		r	

15	4	3	0
RESERVED		DMOD	
0		1001	
r		rw	

31 : 20 Data byte disable mask. Set bit to 1 to disable corresponding byte lane (if supported by PHY).

19 : 4 Reserved

3 : 0 Delay setting for tMOD (DMOD)

1.10.7 Service configuration register 1

Table 19. 0x018 - FTADDR service configuration register 1

31	30 29	28	27 20	19 16
SHOLD	RESERVED	FREOP	MRREPIVAL	REFIVAL (7:4)
0	0	0	11111111	0000
rw	r	rw	rw	rw

15 12	11 0
REFIVAL (3:0)	SRVBASEPER
0001	010100010010
rw	rw

31	Hold-off service intervals, set to 1 to delay triggering any refresh or mode reprog cycles
30 : 29	Reserved
28	Force always reopening rows after service interval (see section 1.3.3)
27 : 20	Mode register reprogramming interval, in multiples of base period, minus one Setting this register to all-ones disables mode register reprogramming
19 : 12	Refresh interval, in multiples of base period, minus one Setting this register to all-ones disables periodic refresh cycles
11 : 0	Service interval base period in DFI clock cycles, minus one

1.10.8 Service configuration register 2

Table 20. 0x01C - FTADDR service configuration register 2

31 24	23 16
U100MULT	RESERVED
00011001	0
rw	r

15 8	7 0
FTRAINIVAL	ITRAINIVAL
11111111	11111111
rw	rw

31 : 24	Multiple of service interval base periods needed to get to 100 us, minus one. This counter is used for the training intervals in this register. The value for this register and the base period is also used for timing of init sequence delays.
23 : 16	Reserved
15 : 8	Full re-training interval, in multiples of 10 ms units (100*U100MULT*SRVBASEPER), minus one Setting this register to all-ones disables full re-training
7 : 0	Incremental re-training interval, in multiples of 10 ms units (100*U100MULT*SRVBASEPER), minus one. Setting this register to all-ones disables incremental training

1.10.9 PHY timing register 1

Table 21. 0x020 - FTADDR PHY timing register 1

31	30	16
PTOR	RESERVED	
0	0	
rr	r	

15	12	11	0
RESERVED		RDEN_MASK	
0		*	
r		rw	

31 PHY timing register override (PTOR)

This is set to 1 to allow manual configuration of the fields in PHY timing register 1 and 2, otherwise they are set up automatically.

30 : 12 Reserved

11 : 0 Read-enable timing mask. This controls when the read-enable signal to the PHY (dfi_rddata_enable) is sent out relative to the read command. If bit 0 is set, the read-enable signal is asserted the same time as the command, if bit 1 is set then the read-enable is asserted one cycle after, and so on.

Only one bit in the mask should be set for DFI compliant operation, however it may in some cases be useful for debugging to set multiple bits to stretch the dfi_rddata_enable longer than the actual burst.

By default (if PTOR is not set) this will be set automatically to an expected good value based on CAS latency values and which PHY implementation is used.

1.10.10 PHY timing register 2

Table 22. 0x024 - FTADDR PHY timing register 2

31	21	20	16
RESERVED		WRDATA_MASK (8:4)	
0			
r			

15	12	11	9	8	0
WRDATA_MASK (3:0)		RESERVED		WREN_MASK	
*		0		*	
rw		r		rw	

31 : 21 Reserved

20 : 12 Write data timing mask. Controls the timing of delivery of write data to the PHY relative to the write command, analogous to the RDEN_MASK.

By default (if PTOR in PHY timing register 1 is not set), this is set automatically to the expected good value based on CAS/write latency settings and PHY implementation.

11 : 9 Reserved

8 : 0 Write enable mask. Controls the timing of the dfi_wrdata_en signal to the PHY relative to the write command, analogous to RDEN_MASK.

By default (if PTOR in PHY timing register 1 is not set) this is set automatically to the expected good value based on CAS/write latency settings and PHY implementation.

1.10.11 Diagnostic access control register 1

Table 23. 0x028 - FTADDR diagnostic access control register 1

31	24	23	20	19	18	16
RESERVED		DIAGCS		RES	DIAGBANK	
0		(nr)		0	(nr)	
r		rw		r	rw	

15	0
DIAGROW	
(nr)	
rw	

31 : 24 Reserved

23 : 20 Chip select number for next diagnostic access (DIAGCS)

Actual width of this field depends on number of CS lines implemented.

19 Reserved

18 : 16 Bank number used for next diagnostic access (DIAGBANK)

15 : 0 Row number used for next diagnostic access (DIAGROW)

This register is also used to control the address lines in manual mode.

1.10.12 Diagnostic access control register 2

Table 24. 0x02C - FTADDR diagnostic access control register 2

31	30	29	28	27	26	25	24	23	16
DODG	DGWR	MANE	MRASN	MCASN	MWEN	MRDC	MWRC	MANUAL_CSN	
0	(nr)	0	(nr)	(nr)	(nr)	0	0	11111111	
rw	rw	rw	rw	rw	rw	w	w	rw	

15	14	12	11	0
MGTRD	RESERVED		DIAGCOL	
0	0		(nr)	
w	r		rw	

31 Do diagnostic access (DODG)

Write 1 to this field to start a single diagnostic access corresponding to DIAGCS,DIAG-BANK,DIAGCOL,DGWR fields. Self clearing. Poll DIAGDONE to detect completion.

30 Write/Read control for diagnostic access. Set to 1 to make next diagnostic access a write, set to 0 to make it a read.

29 Manual mode request. Set this field to 1 to request the controller to go into manual mode and clear when done . This is a request signal, poll the Backend status register to see when controller is ready.

28 Manual mode RASN signal control

27 Manual mode CASN signal control

26 Manual mode WEN signal control

25 Manual mode read pipeline activation. Write 1 to this the same time as asserting the manual_csn with a read command to fetch the read data into the diagnostic data register.

24 Manual mode write pipeline activation. Write 1 to this field the same time as asserting the manual_csn with a write command to do a write burst with the data from the diagnostic data register.

23 : 16 Manual chip select assertion. If one or more bits in this field are written then the corresponding chip select lines will be asserted for one cycle. This field is self-clearing and returns immediately to all-ones state.

15 Manual gate training read sequence. This will lower one of the MANUAL_CSN bits (which one is selected by the DIAGCS field) two times with 4 cycles apart. The first time the MRDC bit is also written to one.

14 : 12 Reserved

11 : 0 Column number for diagnostic access.

1.10.13 Diagnostic access status register

Table 25. 0x030 - FTADDR diagnostic access status register

31	30 25	24	23 16
DDONE	RESERVED	DGUE	DIAGCEMASK (23:16)
0	0	(nr)	(nr)
r	r	r	r

15 0
DIAGCEMASK (15:0)
(nr)
r

31 Diagnostic access done (DDONE)

Poll this bit after writing DODG, when it is set to 1 the access has completed, diagnostic data registers have been updated and DGUE/DGCEMASK fields are valid.

30 : 25 Reserved

24 Diagnostic access uncorrectable EDAC error detected (DGUE)

23 : 0 Diagnostic access correctable EDAC error mask (DIAGCEMASK)

One bit per nibble read asserted if the EDAC corrected an error on that nibble.

1.10.14 Diagnostic checkbit register

Table 26. 0x034 - FTADDR diagnostic checkbit register

31 16
DIAGCB (31:16)
(nr)
rw

15 0
DIAGCB (15:0)
(nr)
rw

31 : 0 Diagnostic checkbit register. Holds the 32-bit checkbit part of the codeword before or after diagnostic access

1.10.15 Diagnostic data register 1

Table 27. 0x038 - FTADDR diagnostic data register 1

31 16
DIAGDATA1 (31:16)
(nr)
rw

15 0
DIAGDATA1 (15:0)
(nr)
rw

31 : 0 Diagnostic data register 1. Holds the high half of the 64-bit data part of the codeword before or after diagnostic access

1.10.16 Diagnostic data register 2

Table 28. 0x03C - FTADDR diagnostic data register 2

31	16
DIAGDATA2 (31:16)	
(nr)	
rw	

15	0
DIAGDATA2 (15:0)	
(nr)	
rw	

31 : 0 Diagnostic data register 2. Holds the low half of the 64-bit data part of the codeword before or after diagnostic access

1.10.17 ODT configuration register for CS #N

Note: 1-8 separate registers with the same format is implemented, one for each chip select

Table 29. 0x040-0x05C, FTADDR ODT configuration register for CS #N

31 27	26	25 24	23 21	20 19	18 17	16
RESERVED	ODTWL	RTTWL	RTT	RTTWR	DIC	RLODT
0	1	01	000	00	00	0
r	rw	rw	rw	rw	rw	rw

15 8	7 0
RDODT	WRODT
00000000	00000000
rw	rw

31 : 27 Reserved

26 Enable ODT during write leveling (DDR3 only)

25 : 24 RTT setting during write leveling (DDR3 only)

23 : 21 RTT setting for programming into mode register

20 : 19 RTT_WR setting for programming into mode register (DDR3 only)

18 : 17 DIC setting for programming into mode register

16 Enable local on-chip termination when reading from this chip select.

15 : 8 Mask of which ODT signals to enable when reading from this chip select

7 : 0 Mask of which ODT signals to enable when writing to this chip select

1.10.18 ODT external timing register

Table 30. 0x060 - FTADDR ODT external timing register

31	24	23	0
ODTDEFAULT		ODTRDMASK (12:5)	
00000000		00000000	
rw		rw	

13	12	11	1	0
ODTRDMASK (4:1)	RES	ODTWRMASK(12:1)		RES
1000	0	00000000110		0
rw	r	rw		r

31 : 24 ODT default value

Mask specifying which ODT signals are enabled when neither read nor write ODT configuration has been activated.

23 : 13 ODT read timing mask.

Bit mask specifying the timing relative to a read command when the read ODT configuration for that chip select (based on the per-CS ODT registers) is activated. This mask is set automatically based on current CAS latency settings and can only be manually set if the PTOR field in PHY timing register 1 is set.

12 Reserved

Controller does not support asserting ODT simultaneously with the read command, therefore bit 0 of the timing mask is not implemented.

11 : 1 ODT write timing mask

Bit mask specifying the timing relative to a write command, when the write ODT configuration for that chip select (based on the per-CS ODT registers) is activated. This mask is set automatically based on current CAS latency settings and can only be manually set if the PTOR field in PHY timing register 1 is set.

0 Reserved

Controller does not support asserting ODT simultaneously with the write command, therefore bit 0 of the ODT write timing mask is not implemented.

1.10.19 ODT internal timing register

Table 31. 0x064 - FTADDR ODT internal timing register

31	24	23	16
RESERVED		LODTRMASK (11:4)	
*		00000000	
r		rw	

15	12	11	0
LODTRMASK (3:0)		RESERVED	
0000		0	
rw		r	

31 : 24 Reserved

23 : 12 Local ODT read timing mask.

Bit mask specifying the timing relative to a read command when the internal on-chip termination is activated. This mask is set analogous to the RDEN_MASK.

11 : 0 Reserved

1.10.20 Command register

Table 32. 0x068 - FTADDR command register

31										20	19	18	17	16
SET BYTE LANE DISABLE											RES	MREB	DOZQL	DOZQS
0											0	0	0	0
ws											r	ws	ws	ws

15	14	11	10	9	8	7	6	5	4	3	2	1	0
SPRST	TRLANE		BLTR	BLTG	BLTW	INCTR	INCTG	INCTW	FTRR	FTRG	FTRW	DOMR	DOREF
0	r0		0	0	0	0	0	0	0	0	0	0	0
ws	rw		ws	ws	ws	ws	ws	ws	ws	ws	ws	ws	ws

31 : 20	Disable byte lane. Writing 1 to one of the bits in this field sets the corresponding bit in the byte lane disable register in memory configuration register 4 (see 1.10.6). Reading returns a copy of the byte lane disable register.
19	Reserved
18	Write 1 to trigger memory reboot of single byte lane selected by bits 14:11, if supported by controller. The read value of this field will be 1 after writing it until the action has been performed.
17	Write 1 to trigger a single DDR3 ZQ Calibration long command. The read value of this field will be 1 after writing it until the action has been performed.
16	Write 1 to trigger a single DDR3 ZQ Calibration short command. The read value of this field will be 1 after writing it until the action has been performed.
15	Write 1 to this bit to trigger a soft reset of the PHY (if supported). The read value of this field will be 1 after writing it until the action has been performed.
14 : 11	Selects which byte lane to target for byte lane re-training or memory reboot: 0=bits 7:0, 1=bits 15:8, ..., 11 = bits 95:88, 12-15 = reserved This field is only written when one or more of the bits 10:8 or 18 is also written with 1. The field needs to have its value kept until the re-training and re-boot actions have completed.
10	Write 1 to trigger read data eye training on single byte lane selected by bits 14:11. The read value of this field will be 1 after writing it until the action has been performed.
9	Write 1 to trigger gate training on single byte lane selected by bits 14:11. The read value of this field will be 1 after writing it until the action has been performed.
8	Write 1 to trigger write leveling on single byte lane selected by bits 14:11. The read value of this field will be 1 after writing it until the action has been performed.
7	Write 1 to trigger full read data eye training. The read value of this field will be 1 after writing it until the action has been performed.
6	Write 1 to trigger full gate training. The read value of this field will be 1 after writing it until the action has been performed.
5	Write 1 to trigger full write leveling. The read value of this field will be 1 after writing it until the action has been performed.
4	Write 1 to trigger incremental read data eye training. The read value of this field will be 1 after writing it until the action has been performed.
3	Write 1 to trigger incremental gate training. The read value of this field will be 1 after writing it until the action has been performed.
2	Write 1 to trigger incremental write leveling. The read value of this field will be 1 after writing it until the action has been performed.
1	Write 1 to trigger a mode register reprogramming. The read value of this field will be 1 after writing it until the action has been performed.
0	Write 1 to trigger an auto-refresh. The read value of this field will be 1 after writing it until the action has been performed.

1.10.21 Sleep mode configuration register

Table 33. 0x06C - FTADDR sleep mode configuration register

31	16
RESERVED	
0	
r	

15	4	3	1	0
RESERVED		SREFTHRES		ASRFE
0		000		0
r		rw		rw

31 : 4 Reserved

3 : 1 Self refresh threshold (SREFTHRES). Number of idle refresh periods needed in order to trigger going to self-refresh mode, minus one. Only used if SLPEN is set.

0 Automatic self-refresh enable. If set to 1 the controller will automatically put memories in self-refresh mode when idle for a longer time.

1.10.22 EDAC configuration register

Table 34. 0x070 - FTADDR backend EDAC and FIFO error counter register

31	30	29	28	27	26	16
ECCMODE		ERET	RDEEN	DRMW	RESERVED	
10		0	0	0	0	
rw		rw	rw	rw	r	

15	4	3	0
RESERVED		RETTEST	
0		0000	
r		rw	

31 : 30 Error Correction Code Mode

10 = full EDAC, 01 = parity, 00 = none

29 Automatic retry on EDAC internal errors

28 Enable read-error signal from PHY (if supported by PHY) to trigger byte lane failure messages to the front-end.

27 Disable read-modify-write and perform partially masked writes instead. Should only be used when EDAC is disabled (ECCMODE=00)

26 : 4 Reserved

3 : 0 Retry test. Writing a 1 to bit 3 and a count of 1-7 to bits 2:0 will trigger an EDAC internal error flag once in order to test the automatic retry feature enabled by bit 29. The count determines how many read data words from the PHY are received before the test is triggered. Writing 0 to bit 3 or writing 0 to bits 2:0 will have no effect and the field gets cleared.

1.10.23 Service time counter register

This register provides access to the internal running counters of the back-end that trigger the periodic service functions. Writing to this register is mainly intended for testability purposes (for example to produce a specific timing corner case for testing) and is not expected to be used in a typical application.

Table 35. 0x074 - FTADDR service time counter register

31 29	28	27 20	19 16
RESERVED	SITICK	MRREPCTR	REFCTR(7:4)
0	0	00000000	0000
r	rw	rw	rw

15 12	11 0
REFCTR(3:0)	SICTR
0000	000000000000
rw	rw

31 : 29	Reserved
28	Service interval tick, set to 1 when service interval counter reached zero and was reset to the programmed interval.
27 : 20	Mode register reprogramming counter. Decreased when SITICK is set.
19 : 12	Refresh counter. Decreased when SITICK is set.
11 : 0	Service interval counter. Decreased each cycle.

1.10.24 PHY indirect address register

Table 36. 0x078 - FTADDR PHY indirect address register

31 20	19 16
PHY READ ERROR INDICATOR	RESERVED
000000000000	0
r	r

15 8	7 0
RESERVED	PHY REGISTER ADDRESS
0	00000000
r	rw

31 : 20	Phy read error status indicator, read only. This indicates for which byte lanes the PHY has signaled a read error. The bits are cleared on reset or when the corresponding byte lane is disabled via the byte lane disable mask.
19 : 8	Reserved
7 : 0	PHY indirect register address. This field controls which PHY-internal register is accessed via the PHY indirect data register

1.10.25 PHY indirect data register

Table 37. 0x07C - FTADDR PHY indirect data register

31	16
PHY REGISTER DATA(31:16)	
*	
rw	
15	0
PHY REIGSTER DATA(15:0)	
*	
rw	

31 : 0 PHY register data, read/write. Contents of the register depends on which PHY is implemented and which register has been selected in the PHY indirect address register.

1.10.26 PHY generic control register

Table 38. 0x080 - FTADDR PHY generic control register

31	16
PHY GENERIC CONTROL(31:16)	
*	
rw	
15	0
PHY GENERIC CONTROL(15:0)	
*	
rw	

31 : 0 Generic control bus intended to control PHY or system-specific pseudo-static signals.

Number of bits implemented in the controller depends on the *phyctrlbits* generic.

1.10.27 Training time counter register

This register provides access to the internal running counters of the back-end that trigger the periodic training functions, and is also used for power-up and initialization sequence delays. Writing to this register is mainly intended for testability purposes (for example to produce a specific timing corner case for testing) and is not expected to be used in a typical application.

Table 39. 0x084 - FTADDR training time counter register

31	24	23	16
FTRAINCTR		ITRAINCTR	
00000000		00000000	
rw		rw	

15	14	8	7	0
TICK10	M10CTR		U100CTR	
0	0000000		00000000	
rw	rw		rw	

31 : 24 Full incremental training counter value, decreased each 10 millisecond tick.

23 : 16 Periodic incremental training counter value. decreased each 10 millisecond tick.

15 10 millisecond tick, set for one cycle when 10 millisecond counter wraps.

14 : 8 10 millisecond counter, decreased when 100 microsecond counter wraps.

7 : 0 100 microsecond counter, decreased when SITICK is set.

1.10.28 Back-end FIFO error counter register

Table 40. 0x088 - FTADDR back-end FIFO error counter register

31	24	23	22	21	20	19	18	17	16
RESERVED		CM3UE	CM3CE		WD3UE	WD3CE		CM2UE	CM2CE (1)
0		(nr)	(nr)		(nr)	(nr)		(nr)	(nr)
r		wc	wc		wc	wc		wc	wc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CM2CE (0)	WD2UE	WD2CE		CM1UE	CM1CE		WD1UE	WD1CE		CM0UE	CM0CE		WD0UE	WD0CE	
(nr)	(nr)	(nr)		(nr)	(nr)		(nr)	(nr)		(nr)	(nr)		(nr)	(nr)	
wc	wc	wc		wc	wc		wc	wc		wc	wc		wc	wc	

31 : 24	Reserved
23	Uncorrectable error occurred on command FIFO for port #3
22 : 21	Correctable error saturating counter for command FIFO for port #3
20	Uncorrectable error occurred on write-data FIFO for port #3
19 :18	Correctable error saturating counter for write-data FIFO for port #3
17	Uncorrectable error occurred on command FIFO for port #2
16 : 15	Correctable error saturating counter for command FIFO for port #2
14	Uncorrectable error occurred on write-data FIFO for port #2
13 : 12	Correctable error saturating counter for write-data FIFO for port #2
11	Uncorrectable error occurred on command FIFO for port #1
10 : 9	Correctable error saturating counter for command FIFO for port #1
8	Uncorrectable error occurred on write-data FIFO for port #1
7 : 6	Correctable error saturating counter for write-data FIFO for port #1
5	Uncorrectable error occurred on command FIFO for port #0
4 : 3	Correctable error saturating counter for command FIFO for port #0
2	Uncorrectable error occurred on write-data FIFO for port #0.
1 : 0	Correctable error saturating counter for write-data FIFO for port #0.

Note that error counters are only implemented if the port is implemented and the corresponding FIFO implements ECC. Counters are not reset and should be written with ones to be cleared.

1.10.29 AHB address decode register

Table 41. 0x200 - FTADDR AHB address decode register

31	30														16
OORER	RESERVED														
0	0														
rw	r														

15	13	12	9	8	5	4	2	1	0
NCS		CSBASE		BANKBASE		ROWBASE		COLBASE	
111		0011		0101		010		10	
rw		rw		rw		rw		rw	

31 Out-of-range AHB error.

Set to 1 to trigger an AHB error response if an address above the amount of installed memory is made (determined based on CSBASE and NCS).

30 : 16 Reserved

15 : 13 Number of chip selects (ranks / external banks) installed, minus one. Used for out-of-range detection and for the scrubber.

12 : 9 AHB address bits used to determine chip select (CSBASE)

0: HADDR(24 : 22), (nports*4) MiB data/rank

1: HADDR(25 : 23), (nports*8) MiB data/rank

...

9: HADDR(31 : 31), (nports * 2) GiB data/rank (only two ranks used)

10: All addresses map to first chip select, (nports * 4) GiB data/rank

11-15: Reserved

See section 1.2.8 for guidance on setting up this field

8 : 5 AHB address bits used to determine internal bank index (BANKBASE)

Same range as CSBASE.

Note: If no address bits are needed to determine internal bank (number of ports on controller equals number of internal banks on memory), this should be set equal to CSBASE.

See section 1.2.8 for guidance on setting up this field

4 : 2 AHB address bits used to determine row (ROWBASE)

0: HADDR(..: 10), 1 KiB/row (over whole data bus)

1: HADDR(.. : 11), 2 KiB/row (over whole data bus)

...

5: HADDR(.. : 15), 32 KiB/row (over whole data bus)

6-7: Reserved

See section 1.2.8 for guidance on setting up this field

1 : 0 AHB address bits used to determine column (COLBASE)

0: HADDR(N:1), 16 data bits/column (hwidth=1, enx4=1)

1: HADDR(N:2), 32 data bits/column (hwidth=1,enx4=0 or hwidth=0,enx4=1)

2: HADDR(N:3), 64 data bits/column (hwidth=0, enx4=0)

3: Reserved

Note this field must be set corresponding to the backend hwidth/enx4 configuration for correct controller operation.

1.10.30 AHB access configuration register

Table 42. 0x204 - FTADDR AHB access configuration register

31 28	27 24	23 19	18 16
PRELIMHI	PREFLIMLO	RESERVED	WQUEUEMAX
0100	0010	r	000
rw	rw	0	rw

15 13	12 5	4 3	2	1	0
ROWIDLECNT	ROWOTM	ROWOPOL	UEERR	UESIG	CESIG
000	00000000	01	1	1	1
rw	rw	rw	rw	rw	rw

31 : 28 High prefetch threshold, controls how many read commands, plus 1, should be issued ahead of time by the prefetcher. Must not be set higher than the capacity of the prefetch buffer. Only used if prefetching is enabled in the prefetch configuration register.

27 : 24 Low prefetch threshold, controls at what fill level of the prefetch buffer the prefetcher will fetch more data. If set to 0 the prefetcher will not fetch more data until the prefetch buffer is empty, setting it higher will allow the prefetching latency to be masked by remaining data already in the prefetch buffer. Only used if prefetching is enable the prefetch configuration register.

23 : 19 Reserved

18 : 16 Maximum writes allowed to be queued up inside the controller before giving wait states.

Setting the field to 0 disables the maximum, and it is then limited only by FIFO depth.

15 : 13 Row idle cycle count. This allows, in addition to other row policy settings in this register, to controls the minimum number of AHB cycles, plus 1, the AHB port will stay in idle state with a row opened before sending a row-close hint to the back-end.

12 : 5 Row open cycle count. This allows, in addition to the other row policy settings in this register, to set a minimum number of AHB cycles between opening a row and sending a row-close hint to the back-end. This can be matched to the tRAS configured in the back-end.

4 : 3 Front-end row close policy:

00 = aggressive, send row-close hint as early as possible. (for reads, before even receiving data)
01 = standard, send row-close hint whenever AHB port is in idle state.
10 = delayed, as standard but in addition waits for writes queued up in the command FIFO to be emptied by the back-end before send.
11 = keep rows open as long as possible, do not send row-close hints to back-end

2 Enable AHB error responses on uncorrectable EDAC errors when reading

1 Enable sideband signal for uncorrectable error when reading

This setting has no functional impact, except for masking the UE output signal

0 Enable sideband signal for correctable error when reading

This setting has no functional impact, except for masking the CE output signal

1.10.31 Prefetch configuration register

Table 43. 0x208 - FTADDR prefetch configuration register

31	26	25	16
RESERVED		PREFTO	
0		1111111111	
r		rw	

15	0
PREFETCH MASTER MASK	
1111111111111111	
rw	

31 : 26 Reserved

25 : 16 Prefetch timeout, if prefetch data has not been fetched within the specified number of AHB clock cycles, the prefetch data is discarded.

15 : 0 Prefetch enable mask per master.

Each register bit controls whether the corresponding master will trigger the prefetching unit.

Less bits may be implemented depending on the controller configuration.

1.10.32 Scrubber configuration register 1

Table 44. 0x20C - FTADDR scrubber configuration register 1

31	28	27	24	23	20	19	16
INITEN		READBACK		SCRUBPEND		RESERVED	
0000		0000		0000		0	
rw		rw		rw		r	

15	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		RGALL	IEXD	IADR	SFCAL	BESFI	SFIDBL	SEFISTATE		SHOLD	SEFIEN	SCRUBCNT		SCREN
0		0	0	0	0	0	0	00		0	0	00		0
r		rw	rw	rw	rw	rw	rw	r		rw	rw	rw		rw

31 : 28 Initialization mode start, one bit per AHB port. When this bit is set to 1, the port will perform a full initialization run through the whole memory. The bit is cleared after completion.

Reads 1 if currently performing initialization.

27 : 24 Read-back start, one bit per AHB port. When this bit is set to 1, the port will perform a read-back and compare with expected initialization data (corresponding to current configuration).

This can be set to 1 at the same time as the INITEN field in order to trigger a initialization followed back-to-back with a read-back without allowing any other accesses in between.

23 : 20 Scrubber iteration pending, one bit per AHB port. This reads 1 when the periodic scrubber counter has expired, and a scrubber iteration is pending. This can be set to 1 manually by writing, and will in that case trigger a single iteration of the scrubber.

19 : 13 Reserved

12 SEFI handler regenerate all. If set to 0, regeneration will only scrub between scrubber start and scrubber end address, if set to 1 all memory will be scubbed during regeneration.

11 Initialize with existing data mode enable.

When performing initialization (using INITEN field of SEFI handler and init register) and this field is set to 1, use read-write cycles to keep the existing data content and only update the checkbits using read-write cycles.

Note that read-back after initialization will return errors after initializing the data in this mode, since the data contents will not match the init pattern.

10 Initialize with address mode enable.

When set to 1, the lower bits of the initialization and read-back pattern will not be taken from the initialization pattern registers but instead be set to the current rank,bank,row and column of the address being written.

9 Perform PHY re-calibration on byte lane when a SEFI is detected on that byte lane.

Table 44. 0x20C - FTADDR scrubber configuration register 1

8	Set to 1 to handle byte lane errors signaled by PHY as a detected SEFI. If set to 0, byte lane errors are not handled as a detected SEFI but are still reported through the BEREI interrupt. Note that this only has effect if the read-error signal is supported by the PHY and the PHY read error is enabled in the back-end's EDAC configuration register (see 1.10.22)
7	Set to 1 to disable the byte lane in the PHY when a SEFI is detected on that byte lane.
6 : 5	Current SEFI handling state machine state: 00 = Idle, scanning for SEFI 01 = SEFI detected, reprogramming mode registers 10 = SEFI detected, performing regeneration process 11 = Reserved for future use (currently never used)
4	SEFI handler hold off. While set to 1, triggered SEFI will not start regeneration process until cleared. If set to 1 while regeneration ongoing, the regeneration will be suspended until cleared.
3	SEFI handler enabled If set to 0, a detected SEFI will set the byte lane mask but not trigger any regeneration action. If set to 1, a detected SEFI will trigger the SEFI state machine to perform reprogramming and regeneration.
2 : 1	Length of each scrub iteration in 8-column bursts, minus one.
0	Enable scrubber

1.10.33 Scrubber configuration register 2

Table 45. 0x210 - FTADDR scrubber configuration register 2

31	28	27	16
RESERVED		SEFI BYTE LANE MASK	
0		000000000000	
r		r	

15	0
SCRUBBER INTERVAL	
1111111111111111	
rw	

31 : 28	Reserved
27 : 16	Mask of SEFI status for each byte lane, 0=normal 1=SEFI
15 : 0	Scrubber interval, time in AHB cycles between each scrubber iteration, minus one

1.10.34 IRQ pending register

Table 46. 0x214 - FTADDR IRQ pending register

31	16
RESERVED	
0	
r	

15 12	11	10	9	8	7 6 5 4	3 2 1 0
RESERVED	PSFI	BEREI	BESCI	SFI	UEI	CEI
0	0	0	0	0	0000	0000
r	wc	wc	wc	wc	wc	wc

31 : 12	Reserved
11	Permanent SEFI detected. Write 1 to clear
10	Backend read error signaled by PHY. Write 1 to clear
9	Backend EDAC self-check triggered. Write 1 to clear.
8	SEFI detected IRQ ocurred. Write 1 to clear.
7 : 4	Uncorrectable error IRQ occurred, uncorrectable error address register is valid. Write 1 to clear. A separate IRQ bit for each AHB port is implemented
3 : 0	Correctable error IRQ occurred, correctable error address register is valid. Write 1 to clear. A separate IRQ bit for each AHB port is implemented.

1.10.35 IRQ enable register

This register controls whether the interrupt line from the controller is asserted when the corresponding bit in the IRQ pending register goes from 0 to 1. If the IRQ is disabled in this register, then the IRQ pending bit will still be set in the same way, but the interrupt line will not be asserted.

Table 47. 0x218 - FTADDR IRQ enable register

31	16
RESERVED	
0	
r	

15 12	11	10	9	8	7 6 5	4	3 2 1	0
RESERVED	PSFIE	BEREIE	BESCIE	SFIE	RESERVED	UEIE	RESERVED	CEIE
0				0	0	0	0	0
r				wc	r	wc	r	wc

31 : 12	Reserved
11	Permanent SEFI IRQ enable
10	Backend read error IRQ enable
9	Backend EDAC self-check IRQ enable
8	SEFI detected IRQ enable
7 : 5	Reserved
4	Uncorrectable error IRQ enable. Enables/disables the IRQ for all AHB ports
3 : 1	Reserved
0	Correctable error IRQ enable. Enables/disables the IRQ for all AHB ports

1.10.36 Scrubber UE register

Table 48. 0x21C - FTADDR scrubber UE register

31	30	29 27	26 24	23 16
SCRUE	RBERR	SECSNO	SEBANK	SEROW(15:8)
(nr)	(nr)	(nr)	(nr)	(nr)
wc	wc	r	r	r

15 8	7 0
SEROW(7:0)	SECOLHI
(nr)	(nr)
r	r

31	Set to 1 when uncorrectable error was detected while scrubbing on one of the ports. Must be cleared before another error can be logged in this register.
30	Set to 1 when an error during initialization readback was detected on one of the ports. Must be cleared before another error can be logged in this register.
29 : 27	Chip select number of scrubber uncorrectable error or readback error. Only valid when SCRUE or RBERR fields in this register are set.
26 : 24	Internal bank number of scrubber uncorrectable error or readback error. Only valid when SCRUE or RBERR fields in this register are set.
23 : 8	Row number of scrubber uncorrectable error or readback error. Only valid wen SCRUE or RBERR fields in this register are set.
7 : 0	Top bits (11:4) of column number where scrubber uncorrectable error or readback error occurred. Only valid when SCRUE or RBERR fields in this register are set.

1.10.37 Scrubber CE byte lane counter register 1,2,3

Register 1 holds counters for bits 95:64 (check bits), register 2 holds counters for bits 63:32 (high part of data bits), register 3 holds counters for bits 31:0. All registers are cleared by writing to the BLCLR bit in byte lane error counter register 4.

Table 49. 0x220-0x228, FTADDR byte lane error counter register N (N=1..3)

31 30	29 24	23 22	21 16
RESERVED	LANE 4N+3 ERROR COUNTER	RESERVED	LANE 4N+2 ERROR COUNTER
0	000000	0	000000
r	r	r	r

15 14	13 8	7 6	5 0
RESERVED	LANE 4N+1 ERROR COUNTER	RESERVED	LANE 4N+0 ERROR COUNTER
0	000000	0	000000
r	r	r	r

31 : 30	Reserved
29 : 24	Wrapping scrubber corrected error counter for bits 95:88 / 63:56 / 31:24
23 : 22	Reserved
21 : 16	Wrapping scrubber corrected error counter for bits 87:80 / 55:48 / 23:16
15 : 14	Reserved
13 : 8	Wrapping scrubber corrected error counter for bits 79:72 / 47:40 / 15:8
7 : 6	Reserved
5 : 0	Wrapping scrubber corrected error counter for bits 71;64 / 39:32 / 7:0

1.10.38 Scrubber CE byte lane counter register 4

Table 50. 0x22C - FTADDR byte lane error counter register 4

31	30	16
BLCLR	RESERVED	
0	0	
w	r	

15	6	5	0
RESERVED		BYTE LANE WATERMARK	
0		000000	
r		r	

31	Byte lane counter clear (BLCLR). Writing 1 to this bit clears the byte lane counters and resets the water mark
30 : 6	Reserved
5 : 0	Byte lane error counter watermark. This tracks the slowest moving of the byte lane error counters.

1.10.39 Scrubber CE address counter register 1

Table 51. 0x230 - FTADDR address error counter register 1

31	30	28	27	24	23	20	19	16
AECLR	RESERVED		ADCTRCS2		ADCTRCS1		ADCTRCS0	
0	0		(nr)		(nr)		(nr)	
w	r		r		r		r	

15	12	11	8	7	4	3	0
RESERVED		ADCTRBA2		ADCTRBA1		ADCTRBA0	
0		(nr)		(nr)		(nr)	
r		r		r		r	

31	Address error counter clear (AECLR), writing 1 to this field clears all the counters in the address error counter registers.
30 : 28	Reserved
27 : 24	Saturating signed counter for chip select bit 2 (raised for CE in CS#7,6,5,4 lowered for CS#3,2,1,0) Not implented if support for 4 or less chip selects implemented.
23 : 20	Saturating signed counter for chip select bit 1 (raised for CE in CS#7,6,3,2, lowered for CS#5,4,1,0) Not implemented if support for 2 or less chip selects implemented
19 : 16	Saturating signed counter for chip select bit 0 (raised for CE in CS#7,5,3,1, lowered for CS#6,4,2,0) Not implemented if support for 1 chip select implemented
15 : 12	Reserved
11 : 8	Saturating signed counter for bank address bit 2 (raised if BA2=1, lowered if BA2=0 in address)
7 : 4	Saturating signed counter for bank address bit 1 (raised if BA1=1, lowered if BA1=0 in address)
3 : 0	Saturating signed counter for bank address bit 0 (raised if BA0=1, lowered if BA0=0 in address)

1.10.40 Scrubber CE address counter register 2,3

Table 52. 0x234-0x238, FTADDR address error counter register 2/3

31 28	27 24	23 20	19 16
ADCTRROW15 / 7	ADCTRROW14 / 6	ADCTRROW13 / 5	ADCTRROW 12 / 4
(nr)	(nr)	(nr)	(nr)
r		r	
15 12	**11 8**	**7 4**	**3 0**
ADCTRROW 11 / 3	ADCTRROW10 / 2	ADCTRROW9 / 1	ADCTRROW8 / 0
(nr)	(nr)	(nr)	(nr)
r	r	r	r

31 : 28 Saturating signed counter for row address bit 15 / 7
27 : 24 Saturating signed counter for row address bit 14 / 6
23 : 20 Saturating signed counter for row address bit 13 / 5
19 : 16 Saturating signed counter for row address bit 12 / 4
15 : 12 Saturating signed counter for row address bit 11 / 3
11 : 8 Saturating signed counter for row address bit 10 / 2
7 : 4 Saturating signed counter for row address bit 9 / 1
3 : 0 Saturating signed counter for row address bit 8 / 0

1.10.41 Scrubber CE address counter register 4

Table 53. 0x23C, FTADDR address error counter register 4

31 28	27 24	23 20	19 16
ADCTRCOL11	ADCTRCOL10	ADCTRCOL9	ADCTRCOL8
(nr)	(nr))	(nr)	(nr)
r		r	
15 12	**11 8**	**7 4**	**3 0**
ADCTRCOL7	ADCTRCOL6	ADCTRCOL5	ADCTRCOL4
(nr)	(nr)	(nr)	(nr)
r	r	r	r

31 : 28 Saturating signed counter for column address bit 11 (physically on AD13)
27 : 24 Saturating signed counter for column address bit 10 (physically on AD11)
23 : 20 Saturating signed counter for column address bit 9
19 : 16 Saturating signed counter for column address bit 8
15 : 12 Saturating signed counter for column address bit 7
11 : 8 Saturating signed counter for column address bit 6
7 : 4 Saturating signed counter for column address bit 5
3 : 0 Saturating signed counter for column address bit 4

1.10.42 Access CE/UE location register, port 0,1,2,3

For each AHB port two registers are implemented, the first holding the address of a correctable error, the second holding the address of an uncorrectable error. These are only valid if the corresponding IRQ pending bit is set, and will not be updated until the IRQ pending bit has been cleared.

Table 54. 0x240-0x25C - FTADDR AHB CE/UE location register, port 0,1,2,3

31	16
CE/UE ADDRESS (31:16)	
(nr)	
r	

15	0
CE/UE ADDRESS (15:0)	
(nr)	
r	

31 : 0 AHB address of access with error.

1.10.43 Prefetch status register, port N

Table 55. 0x260-0x26C, FTADDR prefetch status register for port #N

31	30	29 25	24 16
PFACT	PFDRN	PREF FETCH POSITION	PREF READOUT POSITION
0	0	(nr)	(nr)
r	r	r	r

15	0
PREFETCH ROW	
(nr)	
r	

31 Prefetcher activated status. 1=activated, 0=idle

30 Prefetcher draining status, 1=draining, 0=normal

This will be set after a master accessing a different address than predicted by the prefetcher before all prefetch data in flight has been discarded

29 : 25 Current fetch position of the prefetcher, column bits 7:3. Only valid when PFACT is 1.

The other, higher, column bits can be deducted from the read position

24 : 16 Current read position of the prefetcher, column bits 11:3. Only valid when PFACT is 1.

15 : 0 Current prefetcher row. Only valid when PFACT is 1.

Note: 1-4 separate registers with the same format is implemented, one for each AHB port. Additional bank/CS address bits are in the prefetch bank register shared between all the ports.

1.10.44 Prefetch bank status register

Table 56. 0x270 - FTADDR prefetch bank status register

31 30	29 27	26 24	23 22	21 19	18 16
RESERVED	PREF3CS	PREF3IBANK	RESERVED	PREF2CS	PREF2IBANK
0	(nr)	(nr)	0	(nr)	(nr)
r	r	r	r	r	r

15 14	13 11	10 8	7 6	5 3	2 0
RESERVED	PREF1CS	PREF1IBANK	RESERVEVD	PREF0CS	PREF0IBANK
0	(nr)	(nr)	0	(nr)	(nr)
r	r	r	r	r	r

31 : 30 Reserved

29 : 27 Port 3 prefetcher current chip select number. Only valid when PFACT is set in prefetch status register for port 3. Only implemented in 4-port configuration

26 : 24 Port 3 prefetcher internal bank number high bits (lower bits implied “11” by port number). Only valid when PFACT is set in prefetch status register for port 3. Only implemented in 4-port configuration (1 bit wide).

23 : 22 Reserved

21 : 19 Port 2 prefetcher current chip select number. Only valid when PFACT is set in prefetch status register for port 2. Only implemented in 4-port configuration

18 : 16 Port 2 prefetcher internal bank number high bits (lower bits implied “10” by port number). Only valid when PFACT is set in prefetch status register for port 2. Only implemented in 4-port configuration (1 bit wide).

15 : 14 Reserved

13 : 11 Port 1 prefetcher current chip select number. Only valid when PFACT is set in prefetch status register for port 1. Only implemented in 4-port and 2-port configurations.

10 : 8 Port 1 prefetcher internal bank number high bits (lower bits implied “1” or “01” by port number). Only valid when PFACT is set in prefetch status register for port 1. Field is 0-2 bits wide depending on number of ports configured.

7 : 6 Reserved

5 : 3 Port 0 prefetcher current chip select number. Only valid when PFACT is set in prefetch status register for port 0.

2 : 0 Port 0 prefetcher internal bank number high bits (in 2/4-port configurations, lower bits implied “0” or “00” by port number). Only valid when PFACT is set in prefetch status register for port 0. Field is 1-3 bits wide depending on number of ports configured.

1.10.45 Scrubber start address register

Table 57. 0x274 - FTADDR scrubber start address register

31	30 28	27 25	24 16
RES	SCSTCS	SCSTIBANK	SCSTROW(15:7)
0	000	000	000000000
r	rw	rw	rw

15 9	8 0
SCSTROW(6:0)	SCSTCOL
0000000	000000000
rw	rw

31 Reserved

30 : 28 Scrubber start address chip select

27 : 25 Scrubber start address internal bank. For 2/4 port configurations, this field holds only the high internal bank bits, since the lower bits are inferred from the port number.

24 : 9 Scrubber start address row

8 : 0 Scrubber start address column number bits 11:3. Bits 2:0 of the starting column are always 0 by design.

1.10.46 Scrubber end address register

Table 58. 0x278 - FTADDR scrubber end address register

31	30 28	27 25	24 16
RES	SCEDCS	SCEDIBANK	SCEDROW(15:7)
0	111	111	111111111
r	rw	rw	rw

15 9	8 0
SCEDROW(6:0)	SCEDCOL
1111111	111111111
rw	rw

31 Reserved

30 : 28 Scrubber end address chip select

27 : 25 Scrubber end address internal bank. For 2/4 port configurations, this field holds only the high internal bank bits, since the lower bits are inferred from the port number.

24 : 9 Scrubber end address row

8 : 0 Scrubber end address column number bits 11:3. Bits 2:0 of the starting column are always 0 by design.

1.10.47 Frontend FIFO error counter register

Table 59. 0x27C - FTADDR frontend FIFO error counter register

31	30	29 28	27	26	25 24	23	22	21 20	19	18	17 16
RES	RS3UE	RS3CE	RD3UE	PF3UE	RDPF3CE	RES	RS2UE	RS2CE	RD2UE	PF2UE	RDPF2CE
0	(nr)	(nr)	(nr)	(nr)	(nr)	0	(nr)	(nr)	(nr)	(nr)	(nr)
r	wc	wc	wc	wc	wc	r	wc	wc	wc	wc	wc
15	14	13 12	11	10	9 8	7	6	5 4	3	2	1 0
RES	RS1UE	RS1CE	RD1UE	PF1UE	RDPF1CE	RES	RS0UE	RS0CE	RD0UE	PF0UE	RDPF0CE
0	(nr)	(nr)	(nr)	(nr)	(nr)	0	(nr)	(nr)	(nr)	(nr)	(nr)
r	wc	wc	wc	wc	wc	r	wc	wc	wc	wc	wc

31	Reserved
30	Uncorrectable error occurred on response FIFO for port #3
29 : 28	Correctable error counter for response FIFO for port #3
27	Uncorrectable error occurred on read-data FIFO for port #3
26	Uncorrectable error occurred on prefetch-data FIFO for port #3
25 :24	Correctable error saturating counter for read-data and prefetch-data FIFOs for port #3
23	Reserved
22	Uncorrectable error occurred on response FIFO for port #3
21 : 20	Correctable error counter for response FIFO for port #3
19	Uncorrectable error occurred on read-data FIFO for port #3
18	Uncorrectable error occurred on prefetch-data FIFO for port #3
17 :16	Correctable error saturating counter for read-data and prefetch-data FIFOs for port #3
15	Reserved
14	Uncorrectable error occurred on response FIFO for port #3
13 : 12	Correctable error counter for response FIFO for port #3
11	Uncorrectable error occurred on read-data FIFO for port #3
10	Uncorrectable error occurred on prefetch-data FIFO for port #3
9 :8	Correctable error saturating counter for read-data and prefetch-data FIFOs for port #3
7	Reserved
6	Uncorrectable error occurred on response FIFO for port #3
5 : 4	Correctable error counter for response FIFO for port #3
3	Uncorrectable error occurred on read-data FIFO for port #3
2	Uncorrectable error occurred on prefetch-data FIFO for port #3
1 :0	Correctable error saturating counter for read-data and prefetch-data FIFOs for port #3

Note that error counters are only implemented if the port is implemented and the corresponding FIFO implements ECC. Counters are not reset and should be written with ones to be cleared.

1.10.48 Initialization pattern register 1,2,3,4

Table 60. 0x280-0x28C - FTADDR initialization pattern register N

31	16
INITIALIZATION PATTERN DATA (31:16)	
*	
rw	

15	0
INITIALIZATION PATTERN DATA(15:0)	
*	
rw	

31 : 0 Data to use for initialization and readback operation.

1.10.49 Scrubber position register, port N

Table 61. 0x290-0x29C - FTADDR scrubber position register

31	30 28	27 25	24 16
SCWR	SCCURCS	SCCURIBANK	SCCURROW(15:7)
0	000	000	000000000
r	r	r	r

15 9	8 0
SCCURROW(6:0)	SCCURCOL
0000000	000000000
r	r

31 Wrapped bit. Indicates that the scrubber for this port has completed a full iteration.

30 : 28 Scrubber current chip select

27 : 25 Scrubber current internal bank. For 2/4 port configurations, this field holds only the high internal bank bits, since the lower bits are inferred from the port number.

24 : 9 Scrubber current row

8 : 0 Scrubber current column number bits 11:3

1.10.50 ISD65 PHY specific registers

The registers in this section are only implemented in the ISD65 implementation. They are accessed via the indirect PHY register interface.

Table 62. PHY register 0x00 - Manual calibration register 1

31	30	16
RILCK	RESERVED	
0	0	
rw	r	

15	4	3	2	1	0
RESERVED		RLEN	REDGE	RGLEN	WLEN
0		0	0	0	0
r		rw	rw	rw	rw

31	Register interface lock. Set to 1 to prevent leveling state machines from changing the fields of this register
30 : 4	Reserved
3	Manual control of dfi_rdlvl_en
2	Manual control of dfi_rdlvl_edge
1	Manual control of dfi_rdlvl_gate_en
0	Manual control of dfi_wrlvl_en

Table 63. PHY register 0x01 - Manual calibration register 2

31	16
RESERVED	
0	
r	

15	6	5	4	3	2	1	0
RESERVED		RLLD		WLLD	WLSB	URLR	UWLR
0		00		0	0	0	0
r		w		w	w	w	w

31 : 6	Reserved
5 : 4	Manual control of dfi_rdlvl_load
3	Manual control of dfi_wrlvl_load
2	Manual control of dfi_wrlvl_strobe
1	Update rdlvl_resp. Set this field to 1 to trigger sampling of dfi_rdlvl_resp
0	Update wrlvl_resp. Set this field to 1 to trigger sampling of dfi_wrlvl_resp

Table 64. PHY register 0x02-0x04 - Gate coarse delay register 1-3

31 30	29 24	23 18	17 16
RESERVED	GCD4/9	GCD3/8	GCD2/7(5:4)
0	(nr)	(nr)	(nr)
r	rw	rw	rw

15 12	11 6	5 0
.GCD2/7(3:0)	GCD1/6/11	GCD0/5/10
(nr)	(nr)	(nr)
rw	rw	rw

31 : 30 Reserved

29 : 24 Gate coarse delay for, register 1: byte lane 4, register 2: byte lane 9. register 3: reserved.

23 : 18 Gate coarse delay for, register 1: byte lane 3, register 2: byte lane 8. register 3: reserved.

17 : 12 Gate coarse delay for, register 1: byte lane 2, register 2: byte lane 7, register 3: reserved

11 : 6 Gate coarse delay for, register 1: byte lane 1, register 2: byte lane 6, register 3: byte lane 11

5 : 0 Gate coarse delay for, register 1: byte lane 0, register 2: byte lane 5, register 3: byte lane 10

Table 65. PHY register 0x05-0x07 - Gate fine delay register 1-3

31 30	29 24	23 18	17 16
RESERVED	GFD4/9	GFD3/8	GFD2/7(5:4)
0	(nr)	(nr)	(nr)
r	rw	rw	rw

15 12	11 6	5 0
.GFD2/7(3:0)	GFD1/6/11	GFD0/5/10
(nr)	(nr)	(nr)
rw	rw	rw

31 : 30 Reserved

29 : 24 Gate fine delay for, register 1: byte lane 4, register 2: byte lane 9. register 3: reserved.

23 : 18 Gate fine delay for, register 1: byte lane 3, register 2: byte lane 8. register 3: reserved.

17 : 12 Gate fine delay for, register 1: byte lane 2, register 2: byte lane 7, register 3: reserved

11 : 6 Gate fine delay for, register 1: byte lane 1, register 2: byte lane 6, register 3: byte lane 11

5 : 0 Gate fine delay for, register 1: byte lane 0, register 2: byte lane 5, register 3: byte lane 10

Table 66. PHY register 0x08-0x0A - Write leveling delay register 1-3

31 30	29 24	23 18	17 16
RESERVED	WLD4/9	WLD3/8	WLD2/7(5:4)
0	(nr)	(nr)	(nr)
r	rw	rw	rw

15 12	11 6	5 0
.WLD2/7(3:0)	WLD1/6/11	WLD0/5/10
(nr)	(nr)	(nr)
rw	rw	rw

31 : 30 Reserved

29 : 24 Gate coarse delay for, register 1: byte lane 4, register 2: byte lane 9. register 3: reserved.

23 : 18 Gate coarse delay for, register 1: byte lane 3, register 2: byte lane 8. register 3: reserved.

17 : 12 Gate coarse delay for, register 1: byte lane 2, register 2: byte lane 7, register 3: reserved

11 : 6 Gate coarse delay for, register 1: byte lane 1, register 2: byte lane 6, register 3: byte lane 11

5 : 0 Gate coarse delay for, register 1: byte lane 0, register 2: byte lane 5, register 3: byte lane 10

Table 67. PHY register 0x0B-0x22 - Read delay register 1-24

31 28	27 21	20 16
RESERVED	RLD3	RLD2(6:2)
0	(nr)	(nr)
r	rw	rw

15 14	13 7	6 0
.RLD2(1:0)	RLD1	RLD07
(nr)	(nr)	(nr)
rw	rw	rw

31 : 28 Reserved

27 : 21 Read data delay for data bit 3, 7, 11, ..., 95.

20 : 14 Read data delay for data bit 2, 6, 10, ..., 94.

13 : 7 Read data delay for data bit 1, 5, 9, ..., 93.

6 : 0 Read data delay for data bit 0, 4, 8, ..., 92.

Table 68. PHY register 0x23-0x25 - Write leveling response register 1-3

31 16
WRITE LEVELING RESPONSE (31:16)
0
r

15 0
WRITE LEVELING RESPONSE (15:0)
(nr)
r

31 : 0 Register 1: Write leveling response for bits 31:0
Register 2: Write leveling response for bits 63:32
Register 3: Write leveling response for bits 95:64

Table 69. PHY register 0x26-0x28 - Read leveling response register 1-3

31 16
READ LEVELING RESPONSE (31:16)
(nr)
r

15 0
READ LEVELING RESPONSE (15:0)
(nr)
r

31 : 0 Register 1: Read leveling response for bits 31:0
Register 2: Read leveling response for bits 63:32
Register 3: Read leveling response for bits 95:64

Table 70. PHY register 0x29 - MPR bit position register 1

31	24	23	21	20	18	17	16
RESERVED		MB7		MB6		MB5(2:1)	
0		000		000		00	
r		rw		rw		rw	

15	14	12	11	9	8	6	5	3	2	0
MB5(0)	MB4		MB3		MB2		MB1		MB0	
0	000		000		000		000		000	
rw	rw		rw		rw		rw		rw	

31 : 24 Reserved

23 : 21 MPR result bit position for byte lane 7

20 : 18 MPR result bit position for byte lane 6

17 : 15 MPR result bit position for byte lane 5

14 : 12 MPR result bit position for byte lane 4

11 : 9 MPR result bit position for byte lane 3

8 : 6 MPR result bit position for byte lane 2

5 : 3 MPR result bit position for byte lane 1

2 : 0 MPR result bit position for byte lane 0

Table 71. PHY register 0x2A - MPR bit position register 2

31	24	23	21	20	18	17	16
MPREN	RESERVED						
1	0						
rw	r						

15	14	12	11	9	8	6	5	3	2	0
RES	MB11		MB10		MB9		MB8		MB7	
0	000		000		000		000		000	
r	rw		rw		rw		rw		rw	

31 MPR enable. If set to 1 the DDR3 MPR register will be used for incremental read leveling, otherwise regular reads with a test pattern will be used. As the MPR register is only guaranteed to return the result on one bit, the bit for the result on each byte lane is configured via the MB fields.

20 : 15 Reserved

14 : 12 MPR result bit position for byte lane 11

11 : 9 MPR result bit position for byte lane 10

8 : 6 MPR result bit position for byte lane 9

5 : 3 MPR result bit position for byte lane 8

2 : 0 MPR result bit position for byte lane 7

Table 72. PHY register 0x2B - Training configuration register

31	16
RESERVED	
0	
r	

15	6	5	4	3	2	1	0
RESERVED		RIEN	GIEN	WIEN	RFEN	GFEN	WFEN
0		1	1	1	1	1	1
r		rw	rw	rw	rw	rw	rw

31 : 6	Reserved
5	Incremental read eye training enable. If this is set to 0, the full read eye training willl bbe skipped and no action is taken when incremental read eye training is requested by the controller.
4	Incremental gate training enable. If this is set to 0, the full gate training will be skipped and no action is taken when incremental gate training is requested by the controller.
3	Incremental write leveling enable. If this is set to 0, the incremental write leveling will be skipped and no action is taken when incremental write leveling is requested by the controller.
2	Read eye training enable. If this is set to 0, the full read eye training willl bbe skipped and no action is taken when read eye training is requested by the controller.
1	Gate training enable. If this is set to 0, the full gate training will be skipped and no action is taken when gate training is requested by the controller.
0	Write leveling enable. If this is set to 0, the full write leveling will be skipped and no action is taken when write leveling is requested by the controller

1.11 Vendor and device identifiers

When the GRLIB version of the IP core's top level is used, the core will supply plug-n-play information to the system through the AHB output record. This controller is identified with vendor ID 0x001 (Cobham Gaisler) and device ID NNN. The version described in this document is version 0.

1.12 Configuration options

1.12.1 Configuration options for stand-alone version

Table 73 shows the configuration options (VHDL generics) for the stand-alone version of the core.

Table 73. Configuration options for stand-alone version (ftaddr_sa)

Generic	Function	Allowed range	Default
ahbbits	Width of AHB read/write data buses and maximum access size.	32, 64, 128	128
ddrbits	Width of DDR data bus. This is the width of the external data bus, so the DFI data buses will have double the width.	32 - 96	96
nports	Number of implemented AHB ports	1-4	1
nahbmst	Number of AHB masters on each bus (detemines maximum HMASTER value)	1 - 16	16
numcs	Number of chip select signals implemented	1 - 8	1
ctrldup	Option to create duplicate identical copies of memory control signals. Set to 1 for single copy corresponding to standard DFI interface.	1 - 4	1
numrwen	Width of dfi_rddata_en, dfi_rddata_valid, dfi_wrdata_en signals. Corresponds to ‘DFI Data Enable Width’ and ‘DFI Read Data Enable Width’ in DFI standard.	1 - N	1
numrdlvlphy	Width of dfi_rdlvl_req and dfi_rdlvl_gate_req signals. Corresponds to ‘DFI Read Leveling PHY IF Width’ in DFI standard	1 - N	1
numrdlvlmc	Width of dfi_rdlvl_en, dfi_rdlvl_gate_en, dfi_rdlvl_edge, dfi_rdlvl_load signals. Corresponds to ‘DFI Read Leveling MC IF Width’ in DFI standard.	1 - N	1
numwrlvlphy	Width of dfi_wrlvl_req signal. Corresponds to ‘DFI Write Leveling PHY IF Width’ in DFI standard.	1 - N	1
numwrlvlmc	Width of dfi_wrlvl_en signal. Corresponds to ‘DFI Write Leveling MC IF Width’ in DFI standard	1 - N	1
rdblvlbits	Width per bit of dfi_rdlvl_delay and width per byte of dfi_wrlvl_delay vector. Corresponds to ‘DFI Read Leveling Delay Width’ and ‘DFI Write Leveling Delay Width’ in DFI standard	1 - N	1
rdglvlbits	Width per byte lane of dfi_rdlvl_gate_delay signal. Corresponds to ‘DFI Read Leveling Gate Delay Width’ in DFI standard	1 - N	1
rdgflvlbits	Width per byte lane of dfi_rdlvl_gate_fdelay signal.	1 - N	1
wrlvlbits	Width per byte lane of dfi_wrlvl_delay signal	1 - N	1
phyimpl	PHY implementation ID. See section 1.3.11.	0 - PHY_MAX	0
genphy_trden	Specifies trddata_en parameter for generic DFI implementation. Add 100 to specify CAS latency relative delay.	0-11, 95-110	0
genphy_twrlat	Specifies tphy_wrlat parameter for generic DFI implementation. Add 100 to specify write latency relative delay.	0-11, 95-110	0
genphy_twrdata	Specifies twrdata parameter for generic DFI implementation.	0 - 11	0
dynrst	Enable reset values control from input signals	0 - 1	0
phyctrlbits	Number of bits implemented in generic PHY control register	0 - 32	0

1.12.2 Configuration options for GRLIB version

Tables 74 shows the configuration options (VHDL generics) of the GRLIB version of the core.

Table 74. Configuration options for GRLIB version (ftaddr_gr)

Generic	Function	Allowed range	Default
hindex0	AHB master index for port 0	0 - NAHBMST-1	0
hindex1	AHB master index for port 1	0 - NAHBMST-1	0
hindex2	AHB master index for port 2	0 - NAHBMST-1	0
hindex3	AHB master index for port 3	0 - NAHBMST-1	0
haddr0	ADDR field of the AHB BAR0 defining the SDRAM address area on port 0.	0 - NAPBSLV-1	0
hmask0	MASK field of the AHB BAR0 defining the SDRAM address area on port 0.	0 - 16#FFF#	0
haddr1	ADDR field of the AHB BAR0 defining the SDRAM address area on port 1.	0 - NAPBSLV-1	0
hmask1	MASK field of the AHB BAR0 defining the SDRAM address area on port 1.	0 - 16#FFF#	0
haddr2	ADDR field of the AHB BAR0 defining the SDRAM address area on port 2.	0 - NAPBSLV-1	0
hmask2	MASK field of the AHB BAR0 defining the SDRAM address area on port 2.	0 - 16#FFF#	0
haddr3	ADDR field of the AHB BAR0 defining the SDRAM address area on port 3.	0 - NAPBSLV-1	0
hmask3	MASK field of the AHB BAR0 defining the SDRAM address area on port 3.	0 - 16#FFF#	0
ioaddr0	ADDR field of the AHB BAR1 defining I/O address space on port 0 where control registers are mapped.	0 - 16#FFF#	16#000#
iomask0	MASK field of the AHB BAR1 defining I/O address space on port 0.	0 - 16#FFF#	16#FFF#
ioaddr1	ADDR field of the AHB BAR1 defining I/O address space on port 1 where control registers are mapped.	0 - 16#FFF#	16#000#
iomask1	MASK field of the AHB BAR1 defining I/O address space on port 1.	0 - 16#FFF#	16#FFF#
ioaddr2	ADDR field of the AHB BAR1 defining I/O address space on port 2 where control registers are mapped.	0 - 16#FFF#	16#000#
iomask2	MASK field of the AHB BAR1 defining I/O address space on port 2.	0 - 16#FFF#	16#FFF#
ioaddr3	ADDR field of the AHB BAR1 defining I/O address space on port 3 where control registers are mapped.	0 - 16#FFF#	16#000#
iomask3	MASK field of the AHB BAR1 defining I/O address space on port 3.	0 - 16#FFF#	16#FFF#
hirq	Index of the interrupt line.	0 - NAHBIRQ-1	0
tech	Technology value for FIFO implementation.	0 - NTECH	inferred
ahbbits	Width of AHB read/write data buses and maximum access size.	32, 64, 128	AHBDW
ddrbits	Width of DDR data bus. This is the width of the external data bus, so the DFI data buses will have double the width.	32 - 96	96
nports	Number of implemented AHB ports	1-4	1
nahbmst	Number of AHB masters on each bus (detemines maximum HMASTER value)	1 - 16	16

Table 74. Configuration options for GRLIB version (ftaddr_gr)

Generic	Function	Allowed range	Default
numcs	Number of chip select signals implemented	1 - 8	1
ctrldup	Option to create duplicate identical copies of memory control signals. Set to 1 for single copy corresponding to standard DFI interface.	1 - 4	1
numrwen	Width of dfi_rddata_en, dfi_rddata_valid, dfi_wrdata_en signals. Corresponds to 'DFI Data Enable Width' and 'DFI Read Data Enable Width' in DFI standard.	1 - N	1
numrdlvlphy	Width of dfi_rdlvl_req and dfi_rdlvl_gate_req signals. Corresponds to 'DFI Read Leveling PHY IF Width' in DFI standard	1 - N	1
numrdlvlmc	Width of dfi_rdlvl_en, dfi_rdlvl_gate_en, dfi_rdlvl_edge, dfi_rdlvl_load signals. Corresponds to 'DFI Read Leveling MC IF Width' in DFI standard.	1 - N	1
numwrlvlphy	Width of dfi_wrlvl_req signal. Corresponds to 'DFI Write Leveling PHY IF Width' in DFI standard.	1 - N	1
numwrlvlmc	Width of dfi_wrlvl_en signal. Corresponds to 'DFI Write Leveling MC IF Width' in DFI standard	1 - N	1
rdblvlbits	Width per byte laneof dfi_rdlvl_delay vector. Corresponds to 'DFI Read Leveling Delay Width' in DFI standard	1 - N	1
rdglvlbits	Width per byte lane of dfi_rdlvl_gate_delay signal. Corresponds to 'DFI Read Leveling Gate Delay Width' in DFI standard	1 - N	1
wrlvlbits	Width per byte lane of dfi_wrlvl_delay signal. Corresponds to 'DFI Write Leveling Delay Width' in DFI standard	1 - N	1
phyimpl	PHY implementation ID. See section 1.3.11.	0 - PHY_MAX	0
genphy_trden	Specifies trddata_en parameter for generic DFI implementation. Add 100 to specify CAS latency relative delay.	0-11, 95-110	0
genphy_twrlat	Specifies tphy_wrlat parameter for generic DFI implementation. Add 100 to specify write latency relative delay.	0-11, 95-110	0
genphy_twrdata	Specifies twrdata parameter for generic DFI implementation.	0 - 11	0
fifoftmask	Bit mask specifying which syncfifo_2p instances should have the FT generic set to 1. Sum of: 1: Command FIFOs 2: Response FIFOs 4: Write data FIFOs 8: Read data FIFOs 16: Prefetch read data FIFOs	0-31	31
fifoinfmask	Bit mask specifying which syncfifo_2p instances should have the tech generic set to inferred instead of tech value given in generic. Same encoding as fifoftmask.	0-31	0
dynrst	Enable reset values control from input signals	0 - 1	0
phyctrlbits	Number of bits implemented in generic PHY control register	0 - 32	0

1.13 Signal descriptions

1.13.1 Stand-alone version

Table 75 shows the interface signals (VHDL ports) of the stand-alone version of the core.

Table 75. Signal descriptions for stand-alone version (ftaddr_sa)

Signal name	Type	Function	Active
AHB_CLK	Input	AHB clock	Rising
AHB_RSTN	Input	Reset input for AHB clock domain	Low
AHB_HSEL[nports-1:0]	Input	AMBA AHB control and data signals. Each signal has a separate copy for each port, for a signal width W, port 0 is stored in indexes (W-1):0, port 1 in indexes (2W-1):W, and so forth. When HSEL is asserted, an additional signal HSEL_REG qualifies if the access is to the register area or to the memory.	High
AHB_HSEL_REG[nports-1:0]	Input		High
AHB_HADDR[nports*32-1:0]	Input		-
AHB_HTRANS[nports*2-1:0]	Input		-
AHB_HSIZE[nports*3-1:0]	Input		-
AHB_HBURST[nports*3-1:0]	Input		-
AHB_HWRITE[nports-1:0]	Input		-
AHB_HWDATA[nports*ahbbits-1:0]	Input		-
AHB_HPROT[nports*4-1:0]	Input		-
AHB_HMASTER[nports*4-1:0]	Input		-
AHB_HMASTLOCK[nports-1:0]	Input		-
AHB_HREADY_IN[nports-1:0]	Input		High
AHB_HREADY[nports-1:0]	Output		High
AHB_HRESP[nports*2-1:0]	Output		-
AHB_HRDATA[nports*ahbbits-1:0]	Output		-
AHB_CE[nports-1:0]	Output	EDAC corrected error signal for each port	High
AHB_UE[nports-1:0]	Output	EDAC uncorrectable error signal for each port	High
IRQ_OUT	Output	Interrupt request output from controller. High for single AHB clock cycle when asserted.	High
DYNSYNC	Input	Dynamic synchronization control signal, see section 1.7.2. This is a pseudo-static signal that should only change during reset.	-
DFI_CLK	Input	DFI interface clock	Rising
DFI_RSTN	Input	Reset input for DFI clock domain	Low
DFI_CS_N[ctrldup*numcs-1:0]	Output	DFI interface memory control signals. Forwarded to the memory devices by the PHY. DFI_RESET_N is used for DDR3 only. If ctrldup is set higher than one, multiple identical copies of the output signals are generated.	Low
DFI_BANK[ctrldup*3-1:0]	Output		-
DFI_ADDRESS[ctrldup*16-1:0]	Output		-
DFI_RAS_N[ctrldup-1:0]	Output		Low
DFI_CAS_N[ctrldup-1:0]	Output		Low
DFI_WE_N[ctrldup-1:0]	Output		Low
DFI_CKE[ctrldup*numcs-1:0]	Outpu		High
DFI_ODT[ctrldup*numcs-1:0]	Output		High
DFI_RESET_N[ctrldup*numcs-1:0]	Output		Low
DFI_WRDATA[2*ddrbits-1:0]	Output	DFI memory write interface. All bits of DFI_WRDATA_EN are driven with identical values.	-
DFI_WRDATA_EN[numrwen-1:0]	Output		High
DFI_WRDATA_MASK[ddrbits/4-1:0]	Output		-

Table 75. Signal descriptions for stand-alone version (ftaddr_sa)

Signal name	Type	Function	Active
DFI_RDDATA_EN[numrwen-1:0]	Output	DFI memory read interface. All bits of DFI_RDDATA_EN are driven with identical values. Only bit 0 of DFI_RDDATA_VALID is used.	High
DFI_RDDATA[2*ddrbits-1:0]	Input		-
DFI_RDDATA_VALID[numrwen-1:0]	Input		High
DFI_CTRLUPD_REQ	Output	DFI update interface	High
DFI_CTRLUPD_ACK	Input		High
DFI_PHYUPD_REQ	Input		High
DFI_PHYUPD_TYPE[1:0]	Input		-
DFI_PHYUPD_ACK	Output		High
DFI_DATA_BYTE_DISABLE[ddrbits/8-1:0]	Output	DFI data byte disable signal	High
DFI_DRAM_CLK_DISABLE	Output	DFI clock output disable signal	High
DFI_INIT_COMPLETE	Input	DFI initialization complete signal from PHY	High
DFI_RDLVL_MODE[1:0]	Input	DFI read leveling (gate and eye training) interface.	-
DFI_RDLVL_REQ[numrdlvlphy-1:0]	Input		High
DFI_RDLVL_EN[numrdlvlmc-1:0]	Output		High
DFI_RDLVL_GATE_MODE[1:0]	Input		-
DFI_RDLVL_GATE_REQ[numrdlvlphy-1:0]	Input		High
DFI_RDLVL_GATE_EN[numrdlvlmc-1:0]	Output		High
DFI_RDLVL_CS_N[numcs-1:0]	Output		Low
DFI_RDLVL_EDGE[numrdlvlmc-1:0]	Output		-
DFI_RDLVL_DELAY[(ddrbits/8)*rdblvlbits-1:0]	Output		-
DFI_RDLVL_GATE_DELAY[(ddrbits/8)*rdglvlbits-1:0]	Output		
DFI_RDLVL_LOAD[numrdlvlmc-1:0]	Output		High
DFI_RDLVL_RESP[(ddrbits-1:0]	Input		
DFI_WRLVL_MODE[1:0]	Input	DFI write leveling interface	-
DFI_WRLVL_REQ[numwrlvlphy-1:0]	Input		High
DFI_WRLVL_EN[numwrlvlmc-1:0]	Output		High
DFI_WRLVL_CS_N[numcs-1;0]	Output		Low
DFI_WRLVL_DELAY[(ddrbits/8)*rdblvlbits-1:0]	Output		-
DFI_WRLVL_LOAD[numwrlvlmc-1:0]	Output		High
DFI_WRLVL_STROBE[numwrlvlmc-1:0]	Output		High
DFI_WRLVL_RESP[ddrbits-1:0]	Input		-
XDFI_TERM_EN	Output	Enable for local termination on DQ bus. (PHY specific signal)	High
XDFI_RDERR[(ddrbits/8)-1:0]	Input	Byte lane error detection signal from PHY PHY-specific signal, tie low if unused	High
XDFI_SOFTRST	Output	Soft reset signal to PHY (PHY-specific signal)	High
XDFI_PHYCTRL[31:0]	Output	Generic PHY control register output	-
XDFI_BL_RESETN[(ddrbits/8)-1:0]	Output	RESETN and CKE signals per byte lane, to use instead of regular DFI signals when implementing a memory system with reboot support (see section 1.3.12)	Low
XDFI_BL_CKE[(ddrbits/8)-1:0]	Output		High

Table 75. Signal descriptions for stand-alone version (ftaddr_sa)

Signal name	Type	Function	Active
XDFI_BL_PDN[(ddrbits/8)-1:0]	Output	Output signal for power cycling control per byte lane, only for use when implementing a memory system with reboot support (see section 1.3.12)	Low
XDFI_CLK_ZERO	Output	Output signal for forcing clock outputs low, only for use when implementing a memory system with reboot support (see section 1.3.12)	High
AFI_DQS_BURST	Output	Additional control signal needed when interfacing AFI PHY	High
AFI_WLAT[5:0]	Input	AFI write latency signal, only used when interfacing AFI PHY	-
RSTVAL008[35:0]	Input	Reset values for configuration registers, used only if dynrst generic is set. The name corresponds to the offset in the register area of the register. Bits 31:0 are used as the reset value if dynrst is set. When bit 32 is set, and dynrst is set, this forces the register permanently to the supplied reset value and the corresponding register bits can be optimized out by the synthesis. Bit 35:33 are currently unused.	-
RSTVAL00C[35:0]	Input		-
RSTVAL010[35:0]	Input		-
RSTVAL014[35:0]	Input		-
RSTVAL018[35:0]	Input		-
RSTVAL01C[35:0]	Input		-
RSTVAL020[35:0]	Input		-
RSTVAL024[35:0]	Input		-
RSTVAL040[35:0]	Input		-
RSTVAL044[35:0]	Input		-
RSTVAL048[35:0]	Input		-
RSTVAL04C[35:0]	Input		-
RSTVAL050[35:0]	Input		-
RSTVAL054[35:0]	Input		-
RSTVAL058{35:0]	Input		-
RSTVAL05C[35:0]	Input		-
RSTVAL060[35:0]	Input		-
RSTVAL064[35:0]	Input		-
RSTVAL06C[35:0]	Input		-
RSTVAL070[35:0]	Input		-
RSTVAL080[35:0]	Input		-
RSTVAL200[35:0]	Input		-
RSTVAL204[35:0]	Input		-
RSTVAL208[35:0]	Input		-
RSTVAL20C[35:0]	Input		-
RSTVAL210[35:0]	Input		-
RSTVAL218[35:0]	Input		-
RSTVAL274[35:0]	Input		-
RSTVAL278[35:0]	Input		-
RSTVAL280[35:0]	Input		-
RSTVAL284[35:0]	Input		-
RSTVAL288[35:0]	Input		-
RSTVAL28C[35:0]	Input		-

1.13.2 GRLIB version

Table 75 shows the interface signals (VHDL ports) of the GRLIB version of the core.

Table 76. Signal descriptions for GRLIB version (ftaddr_gr)

Signal name	Type	Function	Active
AHB_CLK	Input	AHB clock	Rising
AHB_RSTN	Input	Reset input for AHB clock domain	Low
AHBSI[nports-1:0]	Input	AMBA AHB signal records.	-
AHBSO[nports-1:0]	Output	Port 0 signals are contained in ahbsi[0]/ahbso[0], port 1 in ahbsi[1]/ahbso[1], and so on. All ports are clocked by the same AHB clock but are otherwise independent. The ports are not required to be on the same AHB bus. The IRQ line is driven on AHB port 0 only.	-
AHB_CE[nports-1:0]	Output	EDAC corrected error signal for each port	High
AHB_UE[nports-1:0]	Output	EDAC uncorrectable error signal for each port	High
DYNSYNC	Input	Dynamic synchronization control signal, see section 1.7.2. This is a pseudo-static signal that should only change during reset.	-
DFI_CLK	Input	DFI interface clock	Rising
DFI_RSTN	Input	Reset input for DFI clock domain	Low
DFI_CS_N[ctrldup*numcs-1:0]	Output	DFI interface memory control signals. Forwarded to the memory devices by the PHY. DFI_RESET_N is used for DDR3 only. If ctrldup is set higher than one, multiple identical copies of the output signals are generated.	Low
DFI_BANK[ctrldup*3-1:0]	Output		-
DFI_ADDRESS[ctrldup*16-1:0]	Output		-
DFI_RAS_N[ctrldup-1:0]	Output		Low
DFI_CAS_N[ctrldup-1:0]	Output		Low
DFI_WE_N[ctrldup-1:0]	Output		Low
DFI_CKE[ctrldup*numcs-1:0]	Outpu		High
DFI_ODT[ctrldup*numcs-1:0]	Output		High
DFI_RESET_N[ctrldup*numcs-1:0]	Output		Low
DFI_WRDATA[2*ddrbits-1:0]	Output	DFI memory write interface. All bits of DFI_WRDATA_EN are driven with identical values.	-
DFI_WRDATA_EN[numrwen-1:0]	Output		High
DFI_WRDATA_MASK[ddrbits/4-1:0]	Output		-
DFI_RDDATA_EN[numrwen-1:0]	Output	DFI memory read interface. All bits of DFI_RDDATA_EN are driven with identical values. Only bit 0 of DFI_RDDATA_VALID is used.	High
DFI_RDDATA[2*ddrbits-1:0]	Input		-
DFI_RDDATA_VALID[numrwen-1:0]	Input		High
DFI_CTRLUPD_REQ	Output	DFI update interface	High
DFI_CTRLUPD_ACK	Input		High
DFI_PHYUPD_REQ	Input		High
DFI_PHYUPD_TYPE[1:0]	Input		-
DFI_PHYUPD_ACK	Output		High
DFI_DATA_BYTE_DISABLE[ddrbits/8-1:0]	Output	DFI data byte disable signal	High
DFI_DRAM_CLK_DISABLE	Output	DFI clock output disable signal	High
DFI_INIT_COMPLETE	Input	DFI initialization complete signal from PHY	High

Table 76. Signal descriptions for GRLIB version (ftaddr_gr)

Signal name	Type	Function	Active
DFI_RDLVL_MODE[1:0]	Input	DFI read leveling (gate and eye training) interface.	-
DFI_RDLVL_REQ[numrdlvlphy-1:0]	Input		High
DFI_RDLVL_EN[numrdlvlmc-1:0]	Output		High
DFI_RDLVL_GATE_MODE[1:0]	Input		-
DFI_RDLVL_GATE_REQ[numrdlvlphy-1:0]	Input		High
DFI_RDLVL_GATE_EN[numrdlvlmc-1:0]	Output		High
DFI_RDLVL_CS_N[numcs-1:0]	Output		Low
DFI_RDLVL_EDGE[numrdlvlmc-1:0]	Output		-
DFI_RDLVL_DELAY[(ddrbits/8)*rdblvlbits-1:0]	Output		-
DFI_RDLVL_GATE_DELAY[(ddrbits/8)*rdglvlbits-1:0]	Output		
DFI_RDLVL_LOAD[numrdlvlmc-1:0]	Output		High
DFI_RDLVL_RESP[ddrbits-1:0]	Input		
DFI_WRLVL_MODE[1:0]	Input	DFI write leveling interface	-
DFI_WRLVL_REQ[numwrlvlphy-1:0]	Input		High
DFI_WRLVL_EN[numwrlvlmc-1:0]	Output		High
DFI_WRLVL_CS_N[numcs-1;0]	Output		Low
DFI_WRLVL_DELAY[(ddrbits/8)*rdblvl-bits-1:0]	Output		-
DFI_WRLVL_LOAD[numwrlvlmc-1:0]	Output		High
DFI_WRLVL_STROBE[numwrlvlmc-1:0]	Output		High
DFI_WRLVL_RESP[ddrbits-1:0]	Input		-
XDFI_TERM_EN	Output	Enable for local termination on DQ bus. (PHY specific signal)	High
XDFI_RDERR[(ddrbits/8)-1:0]	Input	Byte lane error detection signal from PHY PHY-specific signal, tie low if unused	High
XDFI_SOFTRST	Output	Soft reset signal to PHY (PHY-specific signal)	High
XDFI_PHYCTRL[31:0]	Outpu	Generic PHY control register output	-
XDFI_BL_RESETN[(ddrbits/8)-1:0]	Output	RESETN and CKE signals per byte lane, to use instead of regular DFI signals when implementing a memory system with reboot support (see section 1.3.12)	Low
XDFI_BL_CKE[(ddrbits/8)-1:0]	Output		High
XDFI_BL_PDN[(ddrbits/8)-1:0]	Output	Output signal for power cycling control per byte lane, only for use when implementing a memory system with reboot support (see section 1.3.12)	Low
XDFI_CLK_ZERO	Output	Output signal for forcing clock outputs low, only for use when implementing a memory system with reboot support (see section 1.3.12)	High
AFI_DQS_BURST	Output	Additional control signal needed when interfacing AFI PHY	High
AFI_WLAT[5:0]	Input	AFI write latency signal, only used when interfacing AFI PHY	-

Table 76. Signal descriptions for GRLIB version (ftaddr_gr)

Signal name	Type	Function	Active
RSTVAL008[35:0]	Input	Reset values for configuration registers, used only if dynrst generic is set. The name corresponds to the offset in the register area of the register. Bits 31:0 are used as the reset value if dynrst is set. When bit 32 is set, and dynrst is set, this forces the register permanently to the supplied reset value and the corresponding register bits can be optimized out by the synthesis. Bit 35:33 are currently unused.	-
RSTVAL00C[35:0]	Input		-
RSTVAL010[35:0]	Input		-
RSTVAL014[35:0]	Input		-
RSTVAL018[35:0]	Input		-
RSTVAL01C[35:0]	Input		-
RSTVAL020[35:0]	Input		-
RSTVAL024[35:0]	Input		-
RSTVAL040[35:0]	Input		-
RSTVAL044[35:0]	Input		-
RSTVAL048[35:0]	Input		-
RSTVAL04C[35:0]	Input		-
RSTVAL050[35:0]	Input		-
RSTVAL054[35:0]	Input		-
RSTVAL058{35:0]	Input		-
RSTVAL05C[35:0]	Input		-
RSTVAL060[35:0]	Input		-
RSTVAL064[35:0]	Input		-
RSTVAL06C[35:0]	Input		-
RSTVAL070[35:0]	Input		-
RSTVAL080[35:0]	Input		-
RSTVAL200[35:0]	Input		-
RSTVAL204[35:0]	Input		-
RSTVAL208[35:0]	Input		-
RSTVAL20C[35:0]	Input		-
RSTVAL210[35:0]	Input		-
RSTVAL218[35:0]	Input		-
RSTVAL274[35:0]	Input		-
RSTVAL278[35:0]	Input		-
RSTVAL280[35:0]	Input		-
RSTVAL284[35:0]	Input		-
RSTVAL288[35:0]	Input		-
RSTVAL28C[35:0]	Input		-

1.14 Library dependencies

Tables 77 and 78 shows libraries used when instantiating the core (VHDL libraries).

Table 77. Library dependencies for instantiating stand-alone top level

Library	Package	Imported unit(s)	Description
GAISLER	FTADDR_SAPKG	Component	Component declaration for std-logic version

Table 78. Library dependencies for instantiating GRLIB top level

Library	Package	Imported unit(s)	Description
GRLIB	AMBA	Signals	Signal record definitions
GAISLER	FTADDR_PKG	Component, Types	Component declaration for GRLIB version
TECHMAP	GENCOMP	Constants	Constants for tech generic

1.15 Component declaration

The component declaration for the stand-alone and GRLIB versions are provided below.

```
component ftaddr_sa is
   generic (
     ahbbits        : integer := 128;
     ddrbits        : integer := 96;
     nports         : integer := 1;
     nahbmst        : integer := 16;
     numcs          : integer := 1;
     ctrldup        : integer := 1;
     csdup          : integer := 1;
     numrwen        : integer := 1;
     numrdlvlphy    : integer := 1;
     numrdlvlmc     : integer := 1;
     numwrlvlphy    : integer := 1;
     numwrlvlmc     : integer := 1;
     rdblvlbits     : integer := 1;
     rdglvlbits     : integer := 1;
     wrlvlbits      : integer := 1;
     phyimpl        : integer := 0;
     genphy_trden   : integer := 98;
     genphy_twrlat  : integer := 100;
     genphy_twrdata : integer := 0;
     dynrst         : integer := 0;
     phyctrlbits    : integer := 0;
     dffonly        : integer := 0;
     simopts        : integer := 0
     );
   port (
     -- AMBA clock/reset
     ahb_clk               : in  std_ulogic;
     ahb_rstn              : in  std_ulogic;
     -- AMBA ports
     ahb_hsel              : in  std_logic_vector(nports-1 downto 0);
     ahb_hsel_reg          : in  std_logic_vector(nports-1 downto 0);
     ahb_haddr             : in  std_logic_vector(nports*32-1 downto 0);
     ahb_htrans            : in  std_logic_vector(nports*2-1 downto 0);
     ahb_hsize             : in  std_logic_vector(nports*3-1 downto 0);
     ahb_hburst            : in  std_logic_vector(nports*3-1 downto 0);
     ahb_hwrite            : in  std_logic_vector(nports-1 downto 0);
     ahb_hwdata            : in  std_logic_vector(nports*ahbbits-1 downto 0);
     ahb_hprot             : in  std_logic_vector(nports*4-1 downto 0);
     ahb_hmaster           : in  std_logic_vector(nports*4-1 downto 0);
     ahb_hmastlock         : in  std_logic_vector(nports-1 downto 0);
     ahb_hready_in         : in  std_logic_vector(nports-1 downto 0);
     ahb_hready            : out std_logic_vector(nports-1 downto 0);
     ahb_hresp             : out std_logic_vector(nports*2-1 downto 0);
     ahb_hrdata            : out std_logic_vector(nports*ahbbits-1 downto 0);
     -- EDAC sideband signals
     ahb_ce                : out std_logic_vector(nports-1 downto 0);
     ahb_ue                : out std_logic_vector(nports-1 downto 0);
     -- Interrupt output
     irq_out               : out std_ulogic;
     -- Sync configuration
     dynsync               : in  std_ulogic;
     -- DFI clock/reset
     dfi_clk               : in  std_ulogic;
     dfi_rstn              : in  std_ulogic;
```

```
      -- DFI command interface
      dfi_cs_n              : out std_logic_vector(csdup*numcs-1 downto 0);
      dfi_bank              : out std_logic_vector(ctrldup*3-1 downto 0);
      dfi_address           : out std_logic_vector(ctrldup*16-1 downto 0);
      dfi_ras_n             : out std_logic_vector(ctrldup-1 downto 0);
      dfi_cas_n             : out std_logic_vector(ctrldup-1 downto 0);
      dfi_we_n              : out std_logic_vector(ctrldup-1 downto 0);
      dfi_cke               : out std_logic_vector(csdup*numcs-1 downto 0);
      dfi_odt               : out std_logic_vector(csdup*numcs-1 downto 0);
      dfi_reset_n           : out std_logic_vector(csdup*numcs-1 downto 0);
      dfi_wrdata            : out std_logic_vector(2*ddrbits-1 downto 0);
      dfi_wrdata_en         : out std_logic_vector(numrwen-1 downto 0);
      dfi_wrdata_mask       : out std_logic_vector(ddrbits/4-1 downto 0);
      dfi_rddata_en         : out std_logic_vector(numrwen-1 downto 0);
      dfi_rddata            : in  std_logic_vector(2*ddrbits-1 downto 0);
      dfi_rddata_valid      : in  std_logic_vector(numrwen-1 downto 0);
      -- DFI update interface
      dfi_ctrlupd_req       : out std_ulogic;
      dfi_ctrlupd_ack       : in  std_ulogic;
      dfi_phyupd_req        : in  std_ulogic;
      dfi_phyupd_type       : in  std_logic_vector(1 downto 0);
      dfi_phyupd_ack        : out std_ulogic;
      -- DFI init/power-down singals
      dfi_data_byte_disable : out std_logic_vector(ddrbits/8-1 downto 0);
      dfi_dram_clk_disable  : out std_logic_vector(csdup*numcs-1 downto 0);
      dfi_init_complete     : in  std_ulogic;
      -- DFI training/leveling interface
      dfi_rdlvl_mode        : in  std_logic_vector(1 downto 0);
      dfi_rdlvl_req         : in  std_logic_vector(numrdlvlphy-1 downto 0);
      dfi_rdlvl_en          : out std_logic_vector(numrdlvlmc-1 downto 0);
      dfi_rdlvl_gate_mode   : in  std_logic_vector(1 downto 0);
      dfi_rdlvl_gate_req    : in  std_logic_vector(numrdlvlphy-1 downto 0);
      dfi_rdlvl_gate_en     : out std_logic_vector(numrdlvlmc-1 downto 0);
      dfi_rdlvl_cs_n        : out std_logic_vector(csdup*numcs-1 downto 0);
      dfi_rdlvl_edge        : out std_logic_vector(numrdlvlmc-1 downto 0);
      dfi_rdlvl_delay       : out std_logic_vector((ddrbits/8)*rdblvlbits-1 downto 0);
      dfi_rdlvl_gate_delay  : out std_logic_vector((ddrbits/8)*rdglvlbits-1 downto 0);
      dfi_rdlvl_load        : out std_logic_vector(numrdlvlmc-1 downto 0);
      dfi_rdlvl_resp        : in  std_logic_vector(ddrbits-1 downto 0);
      dfi_wrlvl_mode        : in  std_logic_vector(1 downto 0);
      dfi_wrlvl_req         : in  std_logic_vector(numwrlvlphy-1 downto 0);
      dfi_wrlvl_en          : out std_logic_vector(numwrlvlmc-1 downto 0);
      dfi_wrlvl_cs_n        : out std_logic_vector(csdup*numcs-1 downto 0);
      dfi_wrlvl_delay       : out std_logic_vector((ddrbits/8)*wrlvlbits-1 downto 0);
      dfi_wrlvl_load        : out std_logic_vector(numwrlvlmc-1 downto 0);
      dfi_wrlvl_strobe      : out std_logic_vector(numwrlvlmc-1 downto 0);
      dfi_wrlvl_resp        : in  std_logic_vector(ddrbits-1 downto 0);
      -- PHY specific signals
      xdfi_term_en          : out std_ulogic;
      xdfi_rderr            : in std_logic_vector((ddrbits/8)-1 downto 0) := (others => '0');
      xdfi_softrst          : out std_ulogic;
      xdfi_phyctrl          : out std_logic_vector(31 downto 0);  -- Generic PHY control
register
      xdfi_bl_resetn        : out std_logic_vector(ddrbits/8-1 downto 0);
      xdfi_bl_cke           : out std_logic_vector(ddrbits/8-1 downto 0);
      xdfi_bl_pdn           : out std_logic_vector(ddrbits/8-1 downto 0);
      xdfi_clk_zero         : out std_ulogic;
      afi_dqs_burst         : out std_ulogic;
      afi_wlat              : in  std_logic_vector(5 downto 0) := "000000";
      -- Reset values
      rstval008 : in std_logic_vector(35 downto 0) := x"000000000";
      rstval00C : in std_logic_vector(35 downto 0) := x"000000000";
      rstval010 : in std_logic_vector(35 downto 0) := x"000000000";
      rstval014 : in std_logic_vector(35 downto 0) := x"000000000";
      rstval018 : in std_logic_vector(35 downto 0) := x"000000000";
      rstval01C : in std_logic_vector(35 downto 0) := x"000000000";
      rstval020 : in std_logic_vector(35 downto 0) := x"000000000";
      rstval024 : in std_logic_vector(35 downto 0) := x"000000000";
      rstval040 : in std_logic_vector(35 downto 0) := x"000000000";
      rstval044 : in std_logic_vector(35 downto 0) := x"000000000";
      rstval048 : in std_logic_vector(35 downto 0) := x"000000000";
```

```
    rstval04C : in std_logic_vector(35 downto 0) := x"000000000";
    rstval050 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval054 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval058 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval05C : in std_logic_vector(35 downto 0) := x"000000000";
    rstval060 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval064 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval06C : in std_logic_vector(35 downto 0) := x"000000000";
    rstval070 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval080 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval200 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval204 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval208 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval20C : in std_logic_vector(35 downto 0) := x"000000000";
    rstval210 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval218 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval274 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval278 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval280 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval284 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval288 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval28C : in std_logic_vector(35 downto 0) := x"000000000"
    );
end component;

component ftaddr_gr is
  generic (
    hindex0     : integer := 0;
    hindex1     : integer := 0;
    hindex2     : integer := 0;
    hindex3     : integer := 0;
    haddr0      : integer := 0;
    hmask0      : integer := 0;
    haddr1      : integer := 0;
    hmask1      : integer := 0;
    haddr2      : integer := 0;
    hmask2      : integer := 0;
    haddr3      : integer := 0;
    hmask3      : integer := 0;
    ioaddr0     : integer := 16#000#;
    iomask0     : integer := 16#FFC#;
    ioaddr1     : integer := 16#000#;
    iomask1     : integer := 16#FFC#;
    ioaddr2     : integer := 16#000#;
    iomask2     : integer := 16#FFC#;
    ioaddr3     : integer := 16#000#;
    iomask3     : integer := 16#FFC#;
    hirq        : integer := 0;
    tech        : integer := inferred;
    ahbbits     : integer := AHBDW;
    ddrbits     : integer := 96;
    nports      : integer := 1;
    nahbmst     : integer := 16;
    numcs       : integer := 1;
    ctrldup     : integer := 1;
    csdup       : integer := 1;
    numrwen     : integer := 1;
    numrdlvlphy : integer := 1;
    numrdlvlmc  : integer := 1;
    numwrlvlphy : integer := 1;
    numwrlvlmc  : integer := 1;
    rdblvlbits  : integer := 1;
    rdglvlbits  : integer := 1;
    wrlvlbits   : integer := 1;
    phyimpl     : integer := 0;
    genphy_trden   : integer := 98;
    genphy_twrlat  : integer := 100;
    genphy_twrdata : integer := 0;
    fifoftmask  : integer := 16#1F#;
    fifoinfmask : integer := 0;
    dynrst      : integer := 0;
```

```
      phyctrlbits : integer := 0
      );
    port (
      -- AMBA clock/reset
      ahb_clk                : in std_ulogic;
      ahb_rstn               : in std_ulogic;
      -- AMBA ports
      ahbsi                  : in  ahb_slv_in_vector_type(nports-1 downto 0);
      ahbso                  : out ahb_slv_out_vector_type(nports-1 downto 0);
      -- EDAC sideband signals
      ahb_ce                 : out std_logic_vector(nports-1 downto 0);
      ahb_ue                 : out std_logic_vector(nports-1 downto 0);
      -- Sync configuration
      dynsync                : in std_ulogic;
      -- DFI clock/reset
      dfi_clk                : in  std_ulogic;
      dfi_rstn               : in  std_ulogic;
      -- DFI command interface
      dfi_cs_n               : out std_logic_vector(csdup*numcs-1 downto 0);
      dfi_bank               : out std_logic_vector(ctrldup*3-1 downto 0);
      dfi_address            : out std_logic_vector(ctrldup*16-1 downto 0);
      dfi_ras_n              : out std_logic_vector(ctrldup-1 downto 0);
      dfi_cas_n              : out std_logic_vector(ctrldup-1 downto 0);
      dfi_we_n               : out std_logic_vector(ctrldup-1 downto 0);
      dfi_cke                : out std_logic_vector(csdup*numcs-1 downto 0);
      dfi_odt                : out std_logic_vector(csdup*numcs-1 downto 0);
      dfi_reset_n            : out std_logic_vector(csdup*numcs-1 downto 0);
      dfi_wrdata             : out std_logic_vector(2*ddrbits-1 downto 0);
      dfi_wrdata_en          : out std_logic_vector(numrwen-1 downto 0);
      dfi_wrdata_mask        : out std_logic_vector(ddrbits/4-1 downto 0);
      dfi_rddata_en          : out std_logic_vector(numrwen-1 downto 0);
      dfi_rddata             : in  std_logic_vector(2*ddrbits-1 downto 0);
      dfi_rddata_valid       : in  std_logic_vector(numrwen-1 downto 0);
      -- DFI update interface
      dfi_ctrlupd_req        : out std_ulogic;
      dfi_ctrlupd_ack        : in  std_ulogic;
      dfi_phyupd_req         : in  std_ulogic;
      dfi_phyupd_type        : in  std_logic_vector(1 downto 0);
      dfi_phyupd_ack         : out std_ulogic;
      -- DFI init/power-down signals
      dfi_data_byte_disable  : out std_logic_vector(ddrbits/8-1 downto 0);
      dfi_dram_clk_disable   : out std_logic_vector(csdup*numcs-1 downto 0);
      dfi_init_complete      : in  std_ulogic;
      -- DFI trainig/leveling interface
      dfi_rdlvl_mode         : in  std_logic_vector(1 downto 0);
      dfi_rdlvl_req          : in  std_logic_vector(numrdlvlphy-1 downto 0);
      dfi_rdlvl_en           : out std_logic_vector(numrdlvlmc-1 downto 0);
      dfi_rdlvl_gate_mode    : in  std_logic_vector(1 downto 0);
      dfi_rdlvl_gate_req     : in  std_logic_vector(numrdlvlphy-1 downto 0);
      dfi_rdlvl_gate_en      : out std_logic_vector(numrdlvlmc-1 downto 0);
      dfi_rdlvl_cs_n         : out std_logic_vector(csdup*numcs-1 downto 0);
      dfi_rdlvl_edge         : out std_logic_vector(numrdlvlmc-1 downto 0);
      dfi_rdlvl_delay        : out std_logic_vector((ddrbits/8)*rdblvlbits-1 downto 0);
      dfi_rdlvl_gate_delay   : out std_logic_vector((ddrbits/8)*rdglvlbits-1 downto 0);
      dfi_rdlvl_load         : out std_logic_vector(numrdlvlmc-1 downto 0);
      dfi_rdlvl_resp         : in  std_logic_vector(ddrbits-1 downto 0);
      dfi_wrlvl_mode         : in  std_logic_vector(1 downto 0);
      dfi_wrlvl_req          : in  std_logic_vector(numwrlvlphy-1 downto 0);
      dfi_wrlvl_en           : out std_logic_vector(numwrlvlmc-1 downto 0);
      dfi_wrlvl_cs_n         : out std_logic_vector(csdup*numcs-1 downto 0);
      dfi_wrlvl_delay        : out std_logic_vector((ddrbits/8)*wrlvlbits-1 downto 0);
      dfi_wrlvl_load         : out std_logic_vector(numwrlvlmc-1 downto 0);
      dfi_wrlvl_strobe       : out std_logic_vector(numwrlvlmc-1 downto 0);
      dfi_wrlvl_resp         : in  std_logic_vector(ddrbits-1 downto 0);
      -- PHY specific signals
      xdfi_term_en           : out std_ulogic;
      xdfi_rderr             : in std_logic_vector((ddrbits/8)-1 downto 0) := (others =>
'0');
      xdfi_softrst           : out std_ulogic;
      xdfi_phyctrl           : out std_logic_vector(31 downto 0);  -- Generic PHY control
register
```

```
    xdfi_bl_resetn         : out std_logic_vector(ddrbits/8-1 downto 0);
    xdfi_bl_cke            : out std_logic_vector(ddrbits/8-1 downto 0);
    xdfi_bl_pdn            : out std_logic_vector(ddrbits/8-1 downto 0);
    xdfi_clk_zero          : out std_ulogic;
    afi_dqs_burst          : out std_ulogic;
    afi_wlat               : in  std_logic_vector(5 downto 0) := "000000";
    -- Reset values
    rstval008 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval00C : in std_logic_vector(35 downto 0) := x"000000000";
    rstval010 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval014 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval018 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval01C : in std_logic_vector(35 downto 0) := x"000000000";
    rstval020 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval024 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval040 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval044 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval048 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval04C : in std_logic_vector(35 downto 0) := x"000000000";
    rstval050 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval054 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval058 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval05C : in std_logic_vector(35 downto 0) := x"000000000";
    rstval060 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval064 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval06C : in std_logic_vector(35 downto 0) := x"000000000";
    rstval070 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval080 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval200 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval204 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval208 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval20C : in std_logic_vector(35 downto 0) := x"000000000";
    rstval210 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval218 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval274 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval278 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval280 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval284 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval288 : in std_logic_vector(35 downto 0) := x"000000000";
    rstval28C : in std_logic_vector(35 downto 0) := x"000000000"
    );
end component;
```

1.16 Appendix A: Calibration for ISD65 PHY

1.16.1 Write Leveling

Write leveling is only used for DDR3 memories. Write leveling allows to compensate for the DQS/CK skew caused by the fly-by topology of routing. DDR3 standard has a dedicated mode for write leveling which simplifies the operation significantly.

Write leveling algorithm:

1. Write leveling is enabled by writing to MR1 register.

2. “dfi_wrlvl_en” is asserted to switch the PHY to write leveling mode.

3. Smallest delay value is loaded to “dfi_wrlvl_delay” for each byte lane. This value determines how much the DQS and DQSN signals will be delayed that goes out from the PHY to the DDR memories.

4. dfi_wrlvl_store is asserted for one cycle and then a number of cycles determined by the PHY specification is waited to sample the dfi_wrlvl_response value for all the byte lanes.

5. The byte lanes in which a high response is received is considered to be done and the delay value for those lanes are fixed.

6. As soon as not all the byte lanes are done or delay value has not reached to the maximum available value, the delay is incremented by one and the steps from 4. is repeated. If all the byte lanes are done or maximum delay value is reached continue to the next step.

7. If one or more byte lanes is not done after the maximum delay value is reached, those byte lanes are considered to be erroneous.

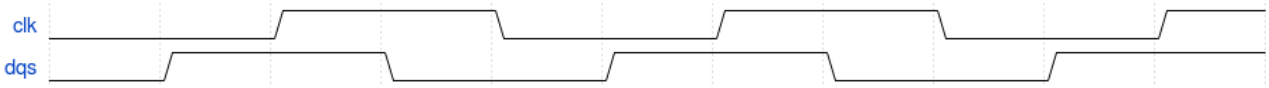
8. “dfi_wrlvl_en” is deasserted to prepare the PHY to regular operation mode.

9. Write leveling is disabled by writing to MR1 register.

Write leveling operation is done in parallel for all the byte lanes requested by the DDR controller.

Following waveforms illustrates a case in which before write leveling CLK is delayed compared to DQS arriving to the inputs of the DDR memory.

Before write leveling:



After write leveling:

1.16.2 Incremental Write Leveling

Incremental write leveling is only used for DDR3 memories and it is optional. It can allow more safe operation if there is a big variation in the delays caused by supply voltage, temperature etc. since it is done periodically, the algorithm is designed to take as short time as possible. Note that the calibration interval is under user control, and it is the user’s responsibility to ensure the incremental calibration is run often enough to avoid memory interface malfunction.

Algorithm:

1. Write leveling is enabled by writing to MR1 register.

2. “dfi_wrlvl_en” is asserted to switch the PHY to write leveling mode.

3. “dfi_wrlvl_delay” is incremented by one and a sample is taken (step 4 in Write Leveling). After that “dfi_wrlvl_delay” is decremented by one and another sample is taken. This way if no change is needed in the delay value, calibration finite state machine can directly finish without updating the dfi_wrlvl_delay value again.

4. Action is determined depending on the values sampled in the previous step.

For each byte lane:

*dfi_wrlvl_delay-1 = ‘0’ & dfi_wrlvl_delay = ‘1’ --> No action is needed.

*dfi_wrlvl_delay-1 = ‘1’ & dfi_wrlvl_delay = ‘1’ --> dfi_wrlvl_delay value needs to be decremented by one.

*dfi_wrlvl_delay-1 = ‘0’ & dfi_wrlvl_delay = ‘0’ --> dfi_wrlvl_delay value needs to be incremented by one.

*dfi_wrlvl_delay-1 = ‘1’ & dfi_wrlvl_delay = ‘0’ --> No action is needed. Theoretically, this case would be expected to never happen, but in practice it might happen due to random timing jitter when taking the two samples if the write strobe is very close the transition region, hence already very close to desired point.

5. If none of the byte lanes requires any action go to the next step directly. If any of the dfi_wrlvl_delay value is changed than update the delay values and go to next step.

6. “dfi_wrlvl_en” is deasserted to prepare the PHY to regular operation mode.

7. Write leveling is disabled by writing to MR1 register.

Incremental write leveling operation is done in parallel for all the byte lanes requested by the DDR controller.

1.16.3 Read Leveling

Read leveling consist of *Gate Calibration* and *Data-Eye Calibration*. Both of them are used with DDR2 and DDR3 memories, and both operations are important for read operations to work correctly.

1.16.4 Gate Calibration

Gate calibration determines the start point when the DQS/DQSN signals arriving from DDR memories are treated as valid during read operation. This prevents a value to be sampled when the data bus is in Hi-Z mode. It is most desirable to identify the start point somewhere close to the middle of preamble window of DQS/DQSN during gate calibration. If gate calibration goes wrong and sets the start point after the first rising edge of DQS then the read data sequence will be wrong.

PHY provides two different delay types for gate calibration. One is called coarse delay value in which every step corresponds to a quarter of the clock cycle. The second one is called fine delay value in which every step corresponds to ~40-80 ps delay value (depending on the manufacturing, temperature, voltage etc.). Both of the delay types are used during gate calibration in order to have a reliable operation.

Gate calibration algorithm:

1. Issue an open row command for further read commands in the calibration for. For DDR3 memories enable dataflow from MPR by writing to MR3 register.

2. “dfi_rdlvl_gate_en” signal is asserted to prepare the PHY for gate leveling operation.

3. Coarse delay value is set to four clock cycles for all the byte lanes. When the delay value is minimum there is a high chance of sampling a Hi-Z value hence sampling start somewhere in the middle of read operation and goes backwards to find the preamble.

4. Issue two read commands back-to-back. Since the sampling is done somewhere in the middle, two read commands are issued back-to-back in order not fall a point after the read operation ends.

5. Read back the responses. Response is the value of the DQS signal during the rising edge of gate enable signal.

5a. If a response is zero increment the coarse delay value by one and go back to step 4.

5b. If a response is one, decrement the coarse delay value by one and go forward to step 6.

5c. If the response is zero even though the smallest coarse delay value is reached mark the byte lane as erroneous and assume the operation for that byte lane is complete and go to step 9.

6. Issue two read commands back-to-back.

6a. If a response is zero, try to find a one response with incrementing the fine delay. Increment fine delay by two. Go to step 7.

6b. If a response is one, decrement coarse delay value by one and go to step 6.

7. Issue two read commands back-to-back.

7.a. If a response is one, the algorithm can proceed next step to determine the beginning of the preamble. Since we are already at the 50% of the preamble the coarse delay is reduced by 3 so that the following samples are always taken at the 25% of the preamble. This reduces the risk of sampling the edge of the DQS signal which can give erroneous results. After coarse delay is reduced by 3 go to step 8.

7.b If a response is zero, increment the fine grain delay by 2 and go back to step 7.

8. Issue two read commands back-to-back.

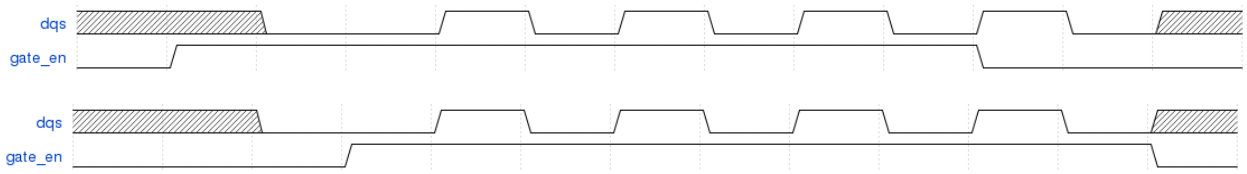
8.a If a response is zero that means preamble is found. At this point coarse delay is incremented by 3 to set the final gate enable to 50%. The reason to increment by 3 is that, after dfi_rldlv_gate_en signal is deasserted ISD PHY decrements coarse delay by two. So that increment by three actually corresponds to increment by one. Go to step 9.

8.b If a response is one, decrement coarse delay value by 4 (full cycle) and repeat step 8.

9. Wait until all the byte lanes are complete. Deassert “dfi_rdlvl_gate_en” to prepare PHY for regular operation. Close the open row..

Gate calibration is done in parallel for all the byte lanes requested by the DDR controller.

Following waveforms illustrates a case in which before gate calibration *gate_en* is enabled when DQS is in Hi-Z mode. After calibration gate_en is enabled approximately at the middle of preamble window which allows a correct read operation.



1.16.5 Incremental Gate Calibration

Incremental gate calibration can be used for both DDR2 and DDR3 and it is optional. It can allow more safe operation if there is a big variation in the delays caused by supply voltage, temperature etc. since it is done periodically, the algorithm is designed to take as short time as possible. Note that the calibration interval is under user control, and it is the user’s responsibility to determine if incremental calibration is needed and ensure the incremental calibration is run often enough to avoid memory interface malfunction.

This algorithm is exactly same as the regular gate calibration described in the previous section. But the coarse delay value starts from the delay value determined by the initial gate calibration. This way the algorithm converges much faster compared to the initial training.

1.16.6 Data-Eye Calibration

The edges of default DQS signal coming out of the DDR memories can not be directly used to sample the DQ because the actual data is available in a window which resides in between the rising and falling edges of the DQS signal and the data is undefined for a certain period after and before each edge of the DQS signal. The purpose of data-eye operation is to shift DQS signal a quarter cycle in order to align the edges of DQS to the middle of data window.

A specific data pattern is written to a part of the DDR controller in order for algorithm to run correctly. Initial data-eye calibration is done for all the available byte lanes and since the DDR controller is not accessed yet there is no problem of data corruption. During SEFI handling an initial date-eye calibration can be optionally enabled while trying to recover from SEFI. In this case the data-eye calibration is done only for the byte that have received SEFI. If the recovery goes successful the section which is overwritten on that byte lane is recovered during scrubbing.

Data-eye calibration is done per byte-lane in which all the bits in a byte lane is calibrated in parallel. Once a byte-lane is finished next byte-lane is calibrated. The reason not all byte lanes are calibrated in parallel is to reduce the number of flip-flops.

Data-eye calibration algorithm:

1. Open a row and write access pattern of zeros and ones which would fit to a one full read burst operation.

2. Assert “dfi_rdlvl_en” signal to prepare the PHY for data-eye calibration operation.

3. Set the value of “dfi_rdlvl_delay” to zero for each bit in the byte-lane.

4. Issue a read command to sample DQ output with the first falling edge of DQS.

5a. If the response is zero set the temporary delay as the current dfi_rdlvl_delay value, after that increment dif_rdlvl_delay value and go to step 6.

5b. If the response is one increment the dfi_rdlvl_delay value is one go to step 4.

6. Issue a read command.

7. Increment the dfi_rdlvl_delay value and go back to step 6 continuously and collect the responses in a shift register. If at any point a one is sampled go back to state 4. If all the shift register is filled with zeros that means we hit a valid data window. The temporary delay value is the minimum delay point (rdlvl_delay_min_value). Increment dfi_rdlvl_delay value by one and go to step 8.

8. Issue a read command.

9a. If a one is sampled that means we reach to the end of the current valid data window. This value is considered as maximum delay value (rdlvl_delay_value_max). In order to align DQS in the middle of DQ window calculate the final delay as (rdlvl_delay_value_max + rdlvl_delay_min_value)/2. Set this delay value as the final delay value and go to step 10.

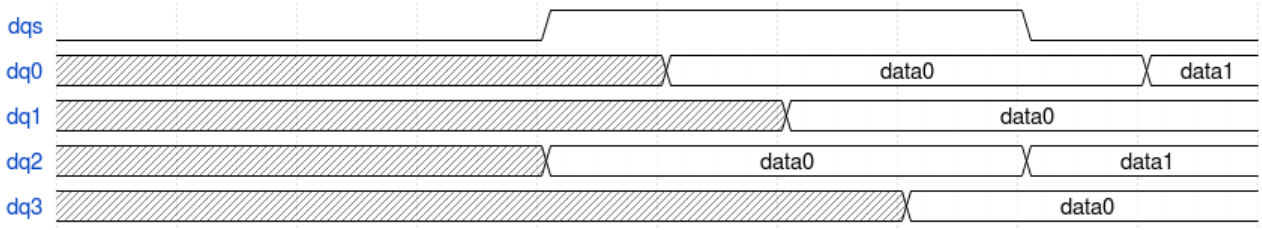
9b. If a zero is sampled go to step 8.

10. If all the bits in the byte-lane is calibrated switch to the next byte-lane and go to step 3. If all the byte-lanes are calibrated then the entire data-eye calibration operation is finished.

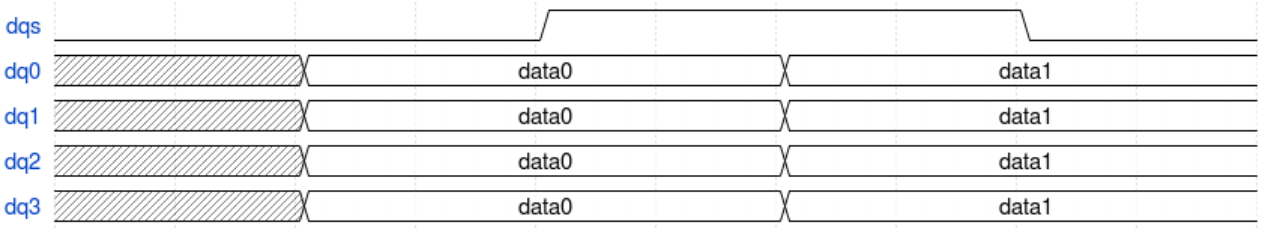
11. Deassert the “dfi_rdlvl_en” signal in order to PHY to return regular operation mode. For DDR2 memories close the open row. For DDR3 memories disabel data flow from MPR mode by writing to MR3 register.

Following waveforms illustrates a case in which before data-eye calibration DQS edges either samples Hi-Z or wrong data. After the data-eye calibration DQS edges are samples the middle of correct data instance.

Before data-eye calibration:



After data-eye calibration:



1.16.7 Incremental Data-Eye Calibration

Incremental data-eye calibration is optional and can allow more safe operation if there is a big variation in the delayed caused by supply voltage, temperature etc. Note that the calibration interval is under user control, and it is the user's responsibility to determine if incremental calibration is needed and ensure the incremental calibration is run often enough to avoid memory interface malfunction.

Incremental data-eye calibration can be done to DDR3 memories using MPR mode without any data corruption. But DDR2 memories require part of the memory to be overwritten (AHB address 0x0 - 0x40), hence the application must not make use of this address range if incremental data-eye calibration is done with DDR2 memory.

MPR mode in DDR3 outputs a predefined data pattern in one of the bits. This bit is used for incremental calibration and change in this bit is applied to all the remaining bits in the byte lane assuming the drift in the bits of the same byte lane has a high correlation. The position of the bit in which predefined output is driven is programmable through calibration registers, this allows to use MPR mode for all types of DDR3 memories irrespective from the connection of the bits of DDR device to the DDR controller. To simplify the hardware the calibration is also done for a single bit on a byte lane and change is propagated to other bits if the incremental training is done for DDR2 memories also. It is also possible to do DDR3 incremental calibration without MPR mode like the DDR2 but this is not recommended since a memory area will be overwritten as indicated in the previous paragraph.

Since the algorithm is based on a single bit from each byte lane the incremental calibration algorithm is done in parallel for all the byte lanes.

Algorithm:

1. If DDR3 with MPR mode is selected, enable dataflow from MPR by writing to MR3 register. Make sure MPR bit position is set correctly in the calibration module registers. If DDR2 is used or MPR mode is disabled open a row and write the predefined pattern as the regular data-eye calibration.

2. Assert "dfi_rdlvl_en" signal to prepare the PHY for data-eye calibration operation.

3. Save the "dfi_rdlvl_delay" value for the data-flow bits in the byte-lane and set the "dfi_rdlvl_delay" value to rdlvl_delay_value_max which was determined in the initial data-eye training. Issue a read command to sample DQ.

4. Set "dfi_rdlvl_delay" value to rdlvl_delay_value-max-1. Issue a read command to sample DQ.

5. Action is determined depending on the values sampled in steps 3 and 4.

For each byte lane:

rdlvl_delay_value_max-1 = ‘0’ & rdlvl_delay_value_max = ‘1’ --> No action is needed.

rdlvl_delay_value_max-1 = ‘1’ & rdlvl_delay_value_max = ‘1’ --> “dfi_rdlvl_delay” value needs to be decremented by one. This is done for all the bits in the specific byte lane. In addition “rdlvl_delay_value_max” is also decremented by one to prepare for the next instance of incremental data-eye training.

rdlvl_delay_value_max-1 = ‘0’ & rdlvl_delay_value_max = ‘0’ --> “dfi_rdlvl_delay” value needs to be incremented by one. This is done for all the bits in the specific byte lane. In addition “rdlvl_delay_value_max” is also incremented by one to prepare for the next instance of incremental data-eye training.

rdlvl_delay_value_max-1 = ‘1’ & rdlvl_delay_value_max = ‘0’ --> No action is needed. Theoretically, this case would be expected to never happen, but in practice it might happen due to random timing jitter when taking the two samples if the write strobe is very close the transition region, hence already very close to desired point.

6. Update “dfi_rdlvl_delay” on the PHY according to the results from previous step.

7. Deassert the “dfi_rdlvl_en” signal in order to PHY to return regular operation mode. Disable data flow from MPR mode if the calibration is done with MPR mode. Otherwise close the open row.