UniBoard DDR3 firmware module

	Organisatie / Organization	Datum / Date
Auteur(s) / Author(s): Daniel van der Schuur	ASTRON	21 November 2011
Controle / Checked: Eric Kooistra	ASTRON	
Goedkeuring / Approval: Andre Gunst	ASTRON	
Autorisatie / Authorisation: **Handtekening / Signature** Andre Gunst	ASTRON	

Distribution list:

Group:	Others:
Andre Gunst Eric Kooistra Harm-Jan Pepping	Gijs Schoonderbeek Sjouke Zwier Harro Verkouter (JIVE) Jonathan Hargreaves (JIVE) Salvatore Pirruccio (JIVE)

Document history:

Revision	Date	Author	Modification / Change
0.1	2011-09-09	Daniel van der Schuur	Draft
0.2	2011-09-12	Daniel van der Schuur	Moved sections, added schematics
0.3	2011-09-19	Daniel van der Schuur	Updated resource usage section
0.4	2011-10-10	Daniel van der Schuur	Added test results for 1066MT/s; Included sections on write FIFO flushing and packet support.
0.5	2011-10-18	Daniel van der Schuur	Improved layout; Added info on chip select signals; Added references to howto-files.
0.6	2011-11-14	Daniel van der Schuur	Added more details on entity I/O ports; Added info on basic operation to section 2.3.
0.7	2011-11-17	Daniel van der Schuur	Updated diagnostics register map. Deleted section about diagnostics data width.
0.8	2011-11-21	Daniel van der Schuur	Added table containing ddr3 control and status signals; Mentioning actual word sizes where applicable.

Table of contents:

Terminology:

CTLR	Controller
DIAG	Diagnostics (VHDL module)
DP	Data Path (VHDL module)
DDR	Double Data Rate
DUT	Device Under Test
EOP	Start Of Packet
FIFO	First In First Out
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
HDL	Hardware Description Language
HPC	High Performance Controller
IO	Input Output
MISO	Master In Slave Out
MM	Memory-Mapped
MOSI	Master Out Slave In
MT/s	MegaTranfers per Second
Nof	Number of
PHY	Physical layer
PIO	Parallel IO
PRSG	Pseudo Random Sequence Generator
RAM	Random Access Memory
RL	Ready Latency
RTL	Register Transfer Level
SISO	Source In Sink Out
SNK	Sink
SODIMM	Small Outline Dual In-line Memory Module
SOP	Start Of Packet
SOPC	System On a Programmable Chip (Altera)
SOSI	Source Out Sink In
SRC	Source
ST	Streaming

References:

1. 'DP Streaming Module Description', ASTRON-RP-382, Eric Kooistra
2. https://svn.astron.nl/UniBoard_FP7/UniBoard/trunk, the UniBoard FP7 SVN repository ($UNB)
3. $UNB/Firmware/docs/howto/How_to_develop_Nios2_software.txt
4. $UNB/Firmware/docs/howto/How_to_use_Modelsim.txt
5. $UNB/Firmware/docs/howto/How_to_use_the_UniBoard_compilation_scripts.txt
6. $UNB/Firmware/modules/Lofar/diag
7. 'Quartus II Handbook', quartusii_handbook.pdf, www.altera.com
8. 'Altera Stratix IV Device Handbook', July 2010, , www.altera.com
9. 'UniBoard Board Design', ASTRON RP-316, Gijs Schoonderbeek, Sjouke Zwier
10. www.altera.com, "Avalon Interface Specifications", mnl_avalon_spec.pdf
11. 'External Memory Interface Handbook', July 2010, www.altera.com

1 Introduction

1.1 Purpose

The ddr3 firmware module can be used to access the DDR3 SODIMMs connected to the UniBoard FPGAs. Using this module, one can write and read back a data stream (as defined in [1]) to and from DDR3 memory. Currently the module supports a maximum data rate of 1066MT/s. By default, the module's data rate is 800MT/s. The module has been validated at both 800Mt/s and 1066MT/s. During these tests, its write efficiency was found to be 92%, while its read efficiency is 94%.

1.2 Module overview

An overview of the ddr3 module is shown in Figure 1. Altera's ALTMEMPHY MegaFunction contains a user interface to its internal High Performance Controller II, and the PHY interface to the (SODIMM) memory module itself. The ddr3 module provides streaming (ST) user interfaces to the read and the write sides of the memory controller. Optionally, it can trigger a writing sequence on reception of an SOP. Its internal driver accepts a start address and an end address so the user can set the desired address range. A ddr3 register instance (ddr3_reg) enables the user to access status and control signals through an MM interface.

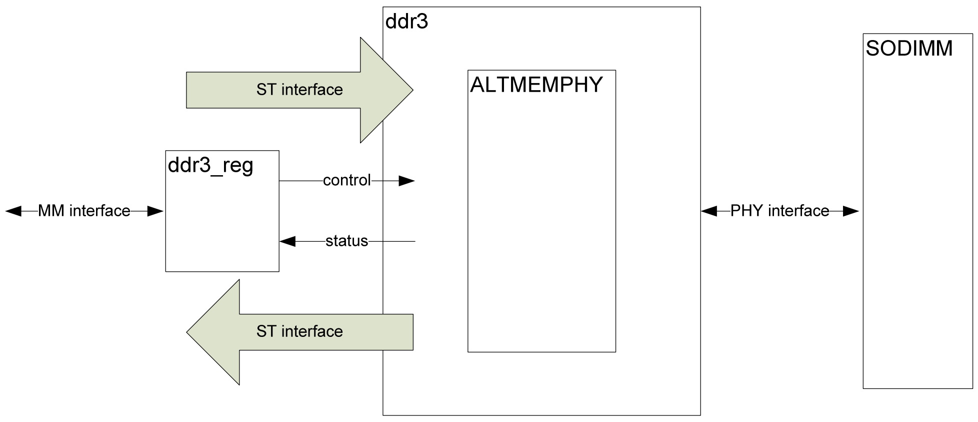

Figure 1: ddr3 module interfaces

The user data width at the write input and read output is variable between 8 and 256 bits. A test design, unb_ddr3, is provided to interface the ddr3 module to a diagnostics module. Both modules support software control, which is described in chapter 2. Chapter 3 covers the clock domains, hardware interfaces and parameters of the ddr3 module. An application of software control of the ddr3 module and the diagnostics module can be found in chapter 4. The final chapters provide details on the HDL implementation and verification.

2 Software interface

2.1 Operation

Figure 2 shows the firmware setup to test the ddr3 module. MM registers are used by a processor (e.g. NIOS II) to control the ddr3 module, and to read out its status. In addition, the diagnostics module [6] is used to verify the correct functioning of a ddr3 module. The data itself is not written or read via the MM interface, as the data path flows through the module via its streaming interfaces. The contents of the MM registers made available via ddr3_reg are explained in paragraph 2.3.

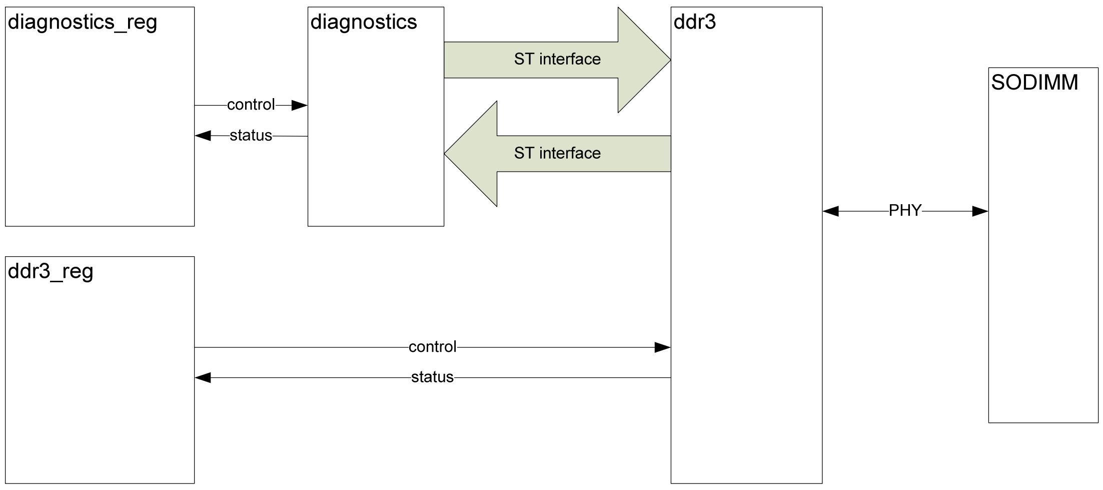

Figure 2: diagnostics module used to test ddr3 module

A typical MM write-read sequence starts by preparing the diagnostics module. The diagnostics module is used to generate a test data stream on its source side to write to the ddr3 module, and to verify the data stream as it is read back from the ddr3 module through its sink side. To prepare the diagnostics module, the user has to set its diagnostics mode to either counter or pseudo random mode, and enable its source and sink side. Data will start to stream from its source towards the ddr3 module as soon as the ddr3 module indicates to be ready via its streaming interface. On the sink side, verification of data will start as soon as valid data is presented on the read side of the ddr3 module.
To start a ddr3 write sequence, the user has to set the address range and set WR_NOT_RD to ‘1’ to write. When the controller is ready, the write sequence is started by writing a ‘1’ to bit 0 of the enable register. The streaming data, generated by the diagnostics module, will be stored in the SODIMM until the end address is reached. As the test data stream flows from the source, the number of sourced words (the number of words accepted by the connected sink – the ddr3 module) is counted and stored in the SRC_CNT register.
The ‘done’ register will indicate a completed sequence. The same order applies to reading back the written data. As valid data is being read back, it is streamed into the sink of the diagnostics module. During or after completion of the read sequence, the diagnostics results can be read out. Paragraph 2.3 lists the contents of the diagnostics registers.

2.2 Operation of the ddr3 module

Regardless of the connected streaming source (write side) and sink (read side), a basic ddr3 operation is performed as follows:

0) After power up, wait until INIT_DONE is high;
1) If the current operation is not the first, wait until DONE is high;
2) Set the address range (START_ADDR and END_ADDR);
3) Set WR_NOT_RD to indicate write or read sequence. During a write sequence, any valid data available on the streaming write port will be written to DDR3 memory until the end address is reached. During a read sequence, data is read back from DDR3 memory up to the end address and is available on the streaming read port.
4) Start the sequence by writing a ‘1’ to the ENABLE register.
5) = 1) Wait until DONE is high before continuing with the next sequence, e.g. reading back the written data.

The used ddr3 status and control signals, made available via a memory mapped interface using ddr3_reg, are explained in the next paragraph.

2.3 MM registers

This paragraph lists the contents of the memory mapped registers. The word size is 32 bits unless stated otherwise. The ddr3 register map is shown in Table 1.

Name	Address (words)	Size (words)	Read/ Write	Description
ENABLE	0	1	W	Enables ddr3 module by writing a ‘1’ to bit 0.
WR_NOT_RD	1	1	W	bit 0: '1' = write; '0' = read.
DONE	2	1	R	bit 0: '1' indicates user defined sequence has been completed.
INIT_DONE	3	1	R	bit 0: '1' after initialization sequence is completed.
CTLR_RDY	4	1	R	bit 0: Ready signal output by Altera's High Performance Controller II. An asserted CTLR_RDY means INIT_DONE is also high. Also, when DONE is asserted, CTLR_RDY should be asserted as well. If CTLR_RDY stays low, an error has occurred.
START_ADDR	5	1	W	Start address. Any start address is allowed.
END_ADDR	6	1	W	End address. >= START_ADDR. Resolution is 4 column addresses (4*64=256 bits), meaning that even in case the set range spans less than the resolution, the number of written or read column addresses starting from START_ADDR is still 4. An exception is when the start address plus the resolution would exceed the highest addressable address.

Table 1: ddr3_reg

Table 2 lists the available MM registers to control the diagnostics module and read out its status. All register bits except those in the SRC_CNT and SNK_CNT registers apply to one diagnostics stream. In case of the DDR3 module, only one 256-bit stream is used, so only bit 0 of most registers is used.

Name	Address (words)	Size (words)	Read/ Write	Description
SRC_EN	0	1	W	Source enable, bit 0 corresponds to stream 0.
SRC_MD	1	1	W	Source mode – 0 for PRBS, 1 for counter mode, bit 0 corresponds to stream 0.
SRC_CNT_CLR	2	1	W	Source count clear, bit 0 corresponds to stream 0.
SRC_CNT	3	16	R	Source count, 32 bits per stream. Source count of stream 0 available at offset 0.
SNK_EN	19	1	W	Sink enable, bit 0 corresponds to stream 0.
SNK_MD	20	1	W	Sink mode– 0 for PRBS, 1 for counter mode, bit 0 corresponds to stream 0.
SNK_CNT_CLR	21	1	W	Sink count clear, bit 0 corresponds to stream 0.
SNK_CNT	22	16	R	Sink count, 32 bits per stream. Sink count of stream 0 available at offset 0.
SNK_DIAG_VAL	38	1	R	Sink diag valid, bit 0 corresponds to stream 0.
SNK_DIAG_RES	39	1	R	Sink diag result, bit 0 corresponds to stream 0.

Table 2: diagnostics_reg

The number of words that are verified is counted and kept in the SNK_CNT register. To read the diagnostics result at the sink side, the register bits in Table 3 and Table 4 are used. These tables provide more details of the SNK_DIAG_RES and SNK_DIAG_VAL registers listed in Table 2. A non-zero diag result (valid when SNK_DIAG_VAL = '1') indicates errors.

Bits	Field name	Description
[32..nof_streams]	unused	unused
[nof_streams-1 .. 0]	SNK_DIAG_VAL	Diag valid indicator bits for each stream. A '1' indicates that the corresponding diag result is valid, else '0'. Bit 0 corresponds to stream 0.

Table 3: SNK_DIAG_VAL register bits

Bits	Field name	Description
[32..nof_streams]	unused	unused
[nof_streams-1 .. 0]	SNK_DIAG_RES	Diag result for each stream. A '0' indicates that the corresponding diag result is OK, else '1'. Bit 0 corresponds to stream 0.

Table 4: SNK_DIAG_RES register bits

3 Hardware interface

3.1 Clock domains

Figure 3 shows an overview of the clock domains that exist in the ddr3 module. The clocks and their descriptions are listed below in Table 5. The listed frequencies apply to the default ddr3 settings that provide a transfer rate of 800MT/s.

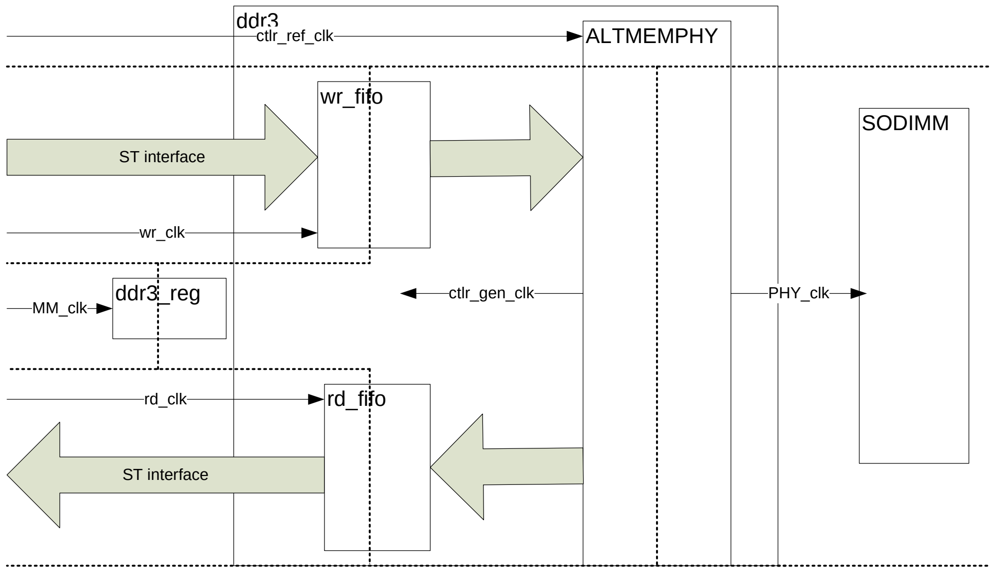

Figure 3: ddr3 clock domains

The user clocks that are used to clock data into the write FIFO (WR_CLK) and data out of the read FIFO (RD_CLK) can be connected to the internally generated controller clock (CTLR_GEN_CLK), as it is available as entity output. In addition, a double frequency version (CTLR_GEN_CLK_2X) is generated by the controller and provided as output.

Name	Frequency (MHz)	Description
CTLR_REF_CLK	200	Controller reference clock. The PHY_CLK and CTLR_GEN_CLK signals are derived from this clock.
CTLR_GEN_CLK	200	Controller generated clock. This clock connects to the read side of the write FIFO and the write side of the read FIFO.
CTLR_GEN_CLK_2X	400	Controller generated (double frequency) clock
PHY_CLK	400	PHY clock to clock the memory module.
WR_CLK	user	Clocks the write side of the write FIFO.
RD_CLK	user	Clocks the read side of the read FIFO.
MM_CLK	user	Clock for the Memory Mapped interface.

Table 5: ddr3 clock signals

3.2 Parameters

Available parameters when instantiating the ddr3 module are listed in Table 6.

Generic	Type	Description
g_phy	NATURAL	Reserved.
g_ddr	t_c_ddr3_phy	Record type containing the specifications of the DDR3 PHY interface.
g_mts	NATURAL	MegaTransfers per Second. Supported: • 800MT/s; • 1066MT/s.
g_wr_data_w	NATURAL	Data width on the write side of the write FIFO. Supported are powers of two between 8..256.
g_rd_data_w	NATURAL	Data width on the read side of the read FIFO. Supported are powers of two between 8..256.
g_wr_fifo_depth	NATURAL	Depth of the write FIFO. Expressed in number of words as seen from the read side of the write FIFO. For correct operation, this value should be equal to or greater than 16, and in addition be greater than the maximum burst size. For highest performance, it should be deep enough to still have a *maximum burst size* number of words stored after completion of the previous burst access.
g_rd_fifo_depth	NATURAL	Depth of the read FIFO. Expressed in number of words as seen from the write side of the read FIFO. This value should be equal or greater than 16 plus c_ddr3_ctrl_nof_latent_reads, which is the maximum number of words the ddr3 module could output after issuing the last read command.
g_wr_use_ctrl	BOOLEAN	When write FIFO flushing is enabled, setting this generic to TRUE adds packet detection to the write side of the DDR3 controller. This means that the flushing of the write FIFO is disabled upon reception of an EOP. This effectively fills the write FIFO on an SOP, preventing the DDR3 modules from being written with incomplete packets.

Table 6: ddr3 parameters

3.3 Interface signals

The interface signals of the ddr3 module are shown in Figure 4. Table 7 lists the general specifications of these interfaces, while the next paragraphs provide more detailed information. The status and control signals of the ddr3 module can be accessed via a memory mapped interface, using ddr3_reg, or can be used without ddr3_reg. The functions of the status and control signals between ddr3_reg and the ddr3 module are listed in Table 8.

Interface	Type	Description
wr_sosi	t_dp_sosi	Streaming data write port, source to sink (ddr3 module)
wr_siso	t_dp_siso	Flow control from ddr3 write port, sink to source
rd_sosi	t_dp_sosi	Streaming data read port, source (ddr3 module) to sink
rd_siso	t_dp_siso	Flow control to ddr3 read port, sink to source
mm_mosi	t_mem_mosi	ddr3 control (MM wrdata, see Table 9) via MM bus (requires ddr3_reg)
mm_miso	t_mem_miso	ddr3 status (MM rddata, see Table 9) via MM bus (requires ddr3_reg)
phy_ou	t_ddr3_phy_ou	Signals output by ALTMEMPHY, to SODIMM via FPGA pins.
phy_io	t_ddr3_phy_io	Bidirectional signals between ALTMEMPHY and SODIMM via FPGA pins.
phy_in	t_ddr3_phy_in	Signals from SODIMM to ALTMEMPHY, via FPGA pins.

Table 7: interface signals

Interface	Type	Description
ctlr_rdy	STD_LOGIC	Indicates whether the internal high performance controller (HPC II) is ready ('1') or not. As a non-ready HPC should only occur during a read or write sequence, this signal is included only for debugging purposes.
ctlr_init_done	STD_LOGIC	'1' after initialization sequence is completed.
dvr_done	STD_LOGIC	'1' indicates user defined sequence has been completed.
dvr_en	STD_LOGIC	Enables read or write sequence. Sequence depends on set address range, dvr_wr_not_rd and whether dvr_done is high.
dvr_wr_not_rd	STD_LOGIC	'1' = write; '0' = read.
dvr_start_addr	t_ddr3_addr	Start address. See 6.1.2.
dvr_end_addr	t_ddr3_addr	End address. See 6.1.2.

Table 8: ddr3 status and control signals

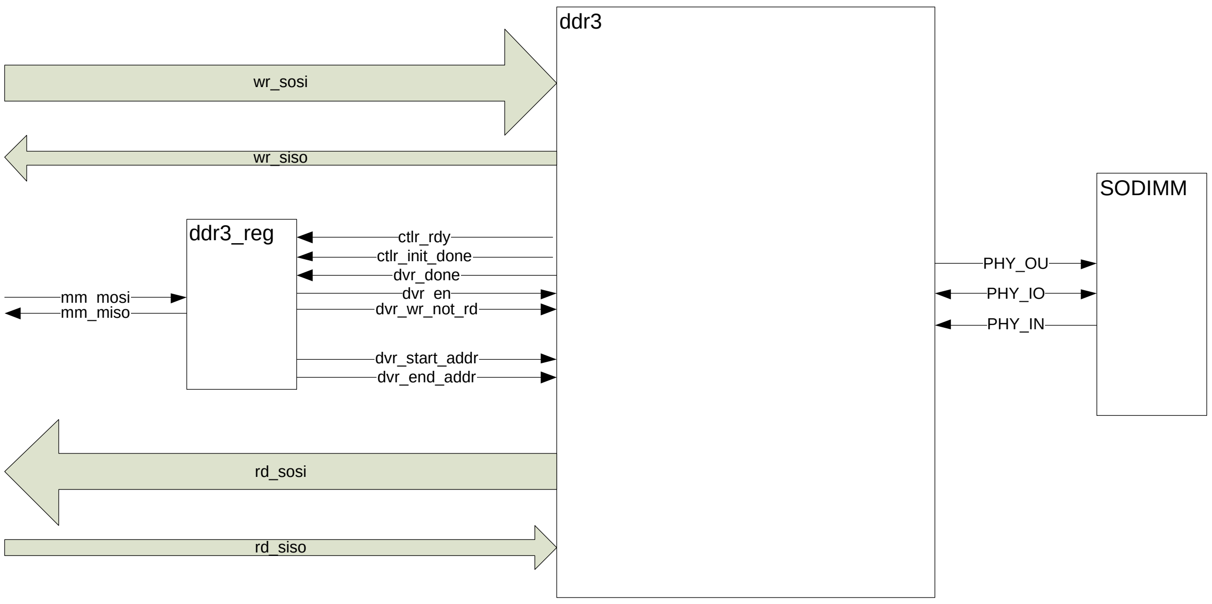

Figure 4: interface signals

3.3.1 MM interface

Table 9 lists the defined interface signals for the Memory Mapped registers.

Signal	Type	Description
rddata[31..0]	MISO	Read data word
address[3..0]	MOSI	Byte address range
wrdata[31..0]	MOSI	Write data word
wr	MOSI	Write enable
rd	MOSI	Read enable

Table 9: MM interface signals

3.3.2 ST interfaces

Streaming interfaces connect to the write side (Table 10) of the write FIFO, and the read side (Table 11) of the read FIFO.

Signal	Type	Description
ready	SISO	Used for flow control, the ready latency (RL) is 1.
data[255..0]	SOSI	Data word, used width: g_wr_data_w-1..0
valid	SOSI	Data valid signal
SOP	SOSI	Start Of Packet (packet detection requires g_wr_use_ctrl = TRUE)
EOP	SOSI	End Of Packet (packet detection requires g_wr_use_ctrl = TRUE)

Table 10: ST interface: write side

Signal	Type	Description
ready	SISO	Used for flow control, the ready latency (RL) is 1.
data[255..0]	SOSI	Data words, used width: g_rd_data_w-1..0
valid	SOSI	Data valid signal

Table 11: ST interface: read side

3.3.3 PHY interface

The PHY interface consists of the I/O pins and signals that connect the ALTMEMPHY to the physical DDR3 SODIMM memory module on the board. Three VHDL record types are used to combine the IN, OUT and INOUT PHY pins into a record of their own. These records are defined in ddr3_pkg.vhd.

4 Application

4.1 Design

Design unb_ddr3 instantiates up to two ddr3 modules, indicated in Figure 5 as ddr_0 and ddr_1. The streaming interfaces of each module connect to the corresponding diagnostics module, which provides a generated data stream on the write interface of the ddr3 modules, and verify the read back data stream through the streaming read ports. The data consists of a random bit sequence or a counter sequence, depending on the set mode. The width of the data is configurable between 8 and 256 bits.
Both the ddr3 modules and the diagnostics modules have their own MM status and control registers, that are written and read by the NIOS II processor in the SOPC.

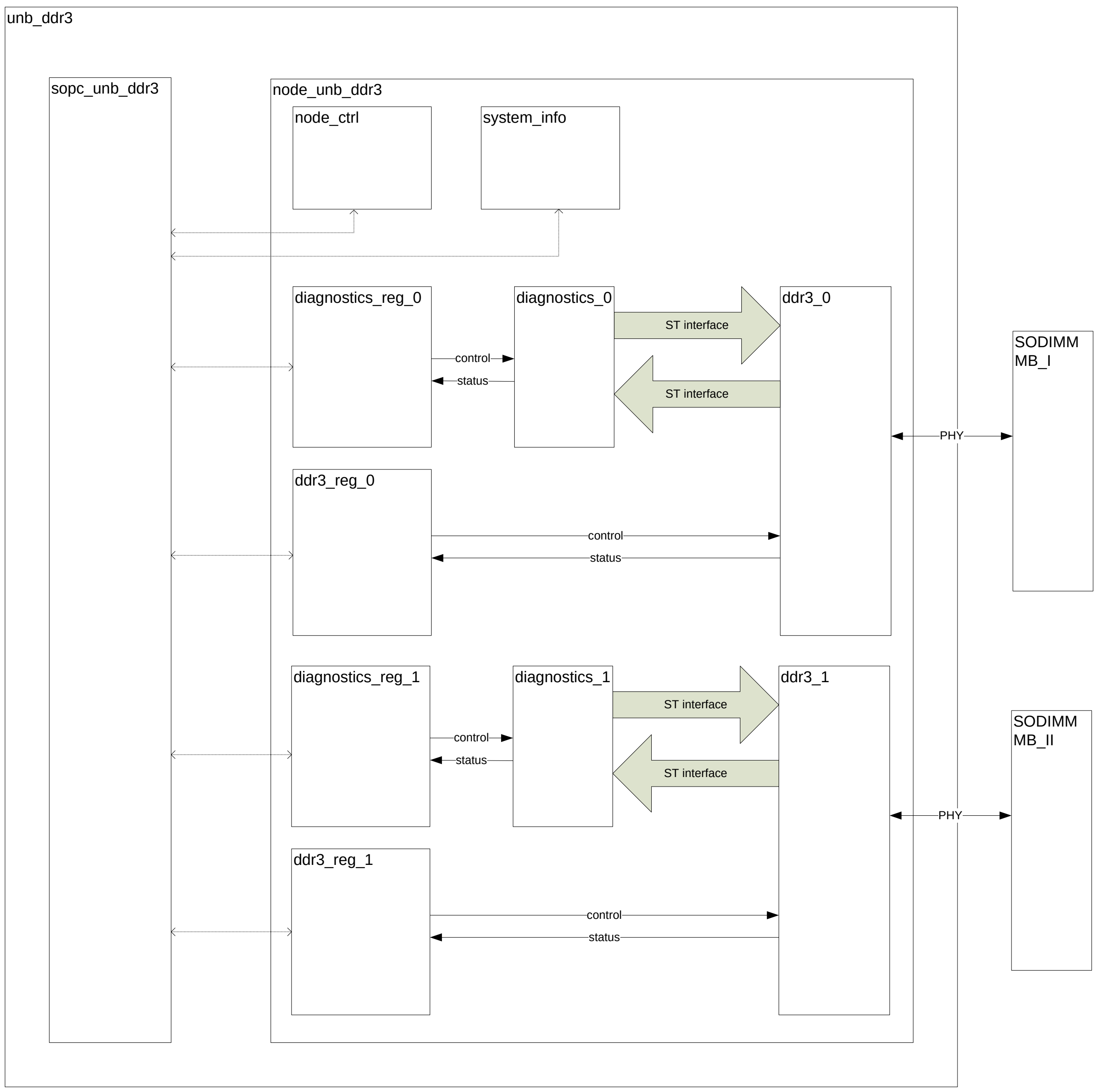

Figure 5: Design: unb_ddr3

4.2 Software

The C-application (main.c – see Appendix 9.2) that runs on the NIOS II processor in the unb_ddr3 design performs a number of test iterations on one or both modules. A test iteration comprises of (in order):

- Setting up the test by:
  - Setting the address range to be tested;
  - Setting the diag mode to PRBS;
  - Enabling the source and sink sides of the diagnostics module;
  - Waiting for init_done to go high.
- Starting a WRITE sequence:
  - Setting the ddr3 driver to WRITE;
  - Clearing the diagnostics sourced and sinked word counts;
  - Enabling the ddr3 driver.
- Waiting for the sequence to be completed (DONE);
- Logging:
  - The number of sourced words when write sequence is done;
  - The duration (in seconds) of this write sequence.
- Starting a READ sequence:
  - Setting the ddr3 driver to READ;
  - Enabling the ddr3 driver.
- Waiting for the sequence to be completed (DONE);
- Interpret the test results:
  - Calculating the number of expected read back words from given address range;
  - Comparing this to the actual number of read back (sinked) words that is logged by the diagnostics module;
  - Read the diagnostics valid and result bits;
  - Log the duration (in seconds) of this read sequence.
- Update the summary that is periodically printed to the screen via JTAG.

4.3 Running on hardware

The design, merged with main.c, can be run on all 8 UniBoard FPGAs simultaneously. However, the printf() output might disrupt the slow JTAG connections. To prevent this, the C-application includes a define to set the delay after which the application starts printing to the screen.

5 Design

5.1 Architecture

Figure 6 shows the internal architecture of the ddr3 module. Its main components are the used IP block – an ALTMEMPHY instance – and the driver that controls it. User data is fed into the write side FIFO via a streaming interface. The write data flow to the ALTMEMPHY is controlled by the driver that toggles the SISO.ready signal on the read side of the write FIFO. The number of words available on the read side of the write FIFO is fed into the driver, so the driver can forward this number to the ALTMEMPHY as the current burst size.

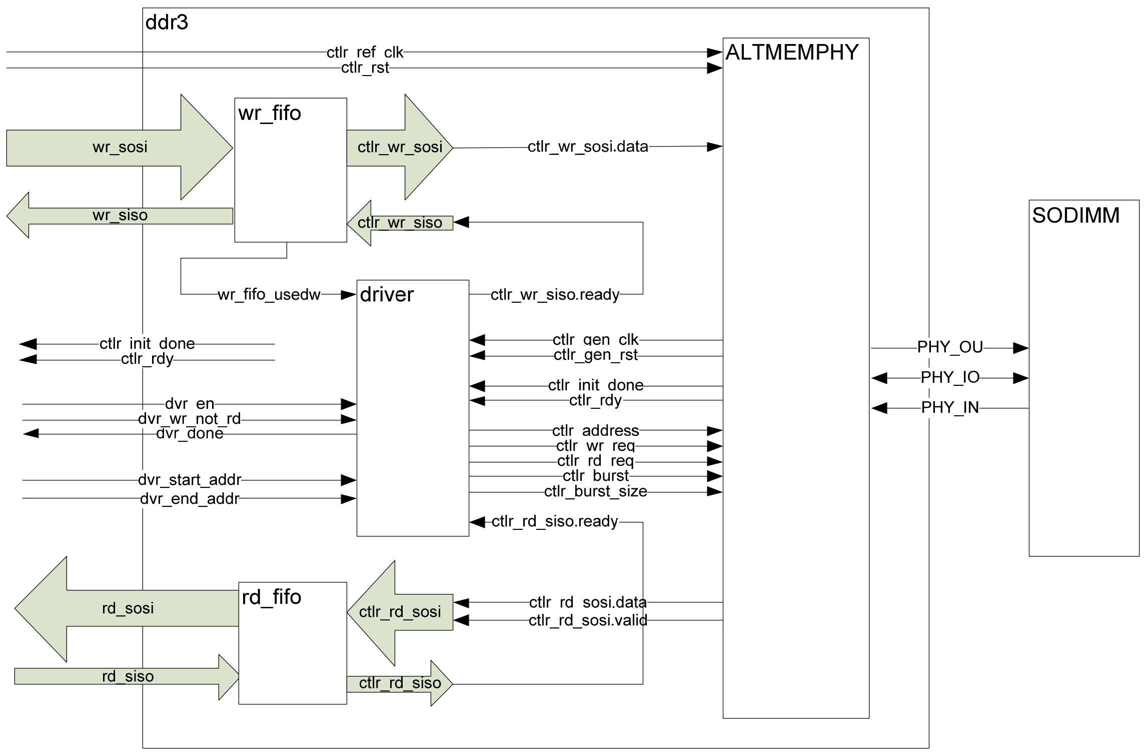

Figure 6: Basic architecture of the ddr3 module

The read burst size depends on the size of the read FIFO. The ready signal originating from the sink that connects to the read side is also fed into the driver and, when in read mode, triggers a read sequence.

5.2 ALTMEMPHY Megafunction settings

The settings of the HPCII used with the ALTMEMPHY include the maximum burst size and the command cue depth. A burst size of 64 words was chosen because it significantly increased performance with respect to a previously tested burst size of 8, without taking up too many additional resources. In addition, a second ALTMEMPHY parameter was changed: the command queue depth was increased from 4 to 8. This did not increase performance at all; it merely used slightly more resources. This is explained by the fact that the unb_ddr3 design in which the ddr3 module was tested writes and reads large address ranges, with maximum burst sizes. This means that the command queue does not need to be that deep, as the controller can already schedule the accesses efficiently because of these large burst sizes.

6 Implementation

This chapter describes the implementation of the ddr3 module, specifically the driver that drives the HPC II.

6.1 Bursting

The ALTMEMPHY's High Performance Controller II supports a maximum burst size of 64 256-bit words. To achieve maximum efficiency, the ddr3 driver posts write commands with burst sizes of 64 words or, if 64 words are not available, the number of words that are present in the write FIFO. Read commands are always for bursts of 64 words unless a burst of 64 would exceed the end address, in which case the burst size equals the number of addresses up to the end address. The latter also applies to write bursts.

6.1.1 Burst size

When posting a burst read or write request to the controller, the driver must provide one address from which the burst access will start. This means that the next address to read or write is the current address incremented by the current burst size. Table 12 shows this for three subsequent burst accesses.

Current address	Burst size	Next current address
0	64	64
64	50	114
114	17	131

Table 12: Next address determined by burst size, decimal representation

These are however simplified representations of the addresses. The actual addresses depend on the address resolution.

6.1.2 Address resolution

The data width of the controller side of the ALTMEMPHY is 256 bits. However, the data bus on the PHY side of the ALTMEMHY is 64 bits wide. This means that one address on the PHY side spans 64 bits, and that one address on the controller side spans 4 times 64 bits. So, without using byte masking, the 'resolution' of one address on the controller side equals four real addresses. Table 13 shows the same burst accesses as Table 12, however this time accounting for the address resolution of 4.

Current address	Burst size	Next current address
0	64	64*4=256
256	50	256+50*4=456
456	17	456+17*4=524

Table 13: Next address, accounting for resolution, decimal representation

The driver uses the passed g_ddr generic to determine the number of chip select lines, bank address width, row address width and column address width. These specifications are used to fill in the values of a ddr3 address type (t_ddr3_addr) record from the 'raw' address. Below, the record type declaration is shown using address widths (actually these are extracted from g_ddr) that apply to the 4GB SODIMM used on the UniBoard:

```
TYPE t_ddr3_addr IS RECORD
  chip   : STD_LOGIC_VECTOR(g_ddr.cs_w -1 DOWNTO 0); -- g_ddr.cs_w  = 2
  bank   : STD_LOGIC_VECTOR(g_ddr.ba_w -1 DOWNTO 0); -- g_ddr.ba_w  = 3
  row    : STD_LOGIC_VECTOR(g_ddr.row_w-1 DOWNTO 0); -- g_ddr.row_w = 15
  column : STD_LOGIC_VECTOR(g_ddr.col_w-1 DOWNTO 0); -- g_ddr.col_w = 10
END RECORD;
```

In Table 14 some addresses are listed in 3 representations. The number of addresses needed to address 4GB equals 2^([logical chip address width]+ba_w+row_w+col_w)= 2^29 = 536.870.912. Note that the logical chip address width (e.g. 1 bit can select one of 2 chips) is used – not the number of actual chip select lines (cs_w). This logical chip address is maintained throughout the design and fed into the ALTMEMPHY, which converts it to a physical chip address consisting of separate chip select lines.
With 2^29 addresses, the highest possible address is 536.870.911 or chip 1, bank 7, row 32.767, column 1023. However, since the address resolution is 4, the highest address on the controller side is 4 addresses down: 536.870.908 or chip 1, bank 7, row 32.767, column 1020. During the last access, column 1020, 1021, 1022 and 1023 are read or written.

Address (decimal)	**Address (std_logic_vector)**	**Address (t_ddr3_addr)**
5	0 000 000000000000000 0000000101	Chip: 0 Bank: 0 Row: 0 Col: 5
301.988.869	1 000 111111111111111 0000000101	Chip: 1 Bank: 0 Row: 32767 Col: 5
536.870.908	1 111 111111111111111 11111111**00**	Chip: 1 Bank: 7 Row: 32767 Col: **1020**

Table 14: Address representations

6.2 Finite state machine

The state machine in Figure 7 gets its input signals from the controller side of the ALTMEMPHY, the read and write FIFOs, and the user. It controls the READY flow control signal for the write side, read and write requests, and burst size.

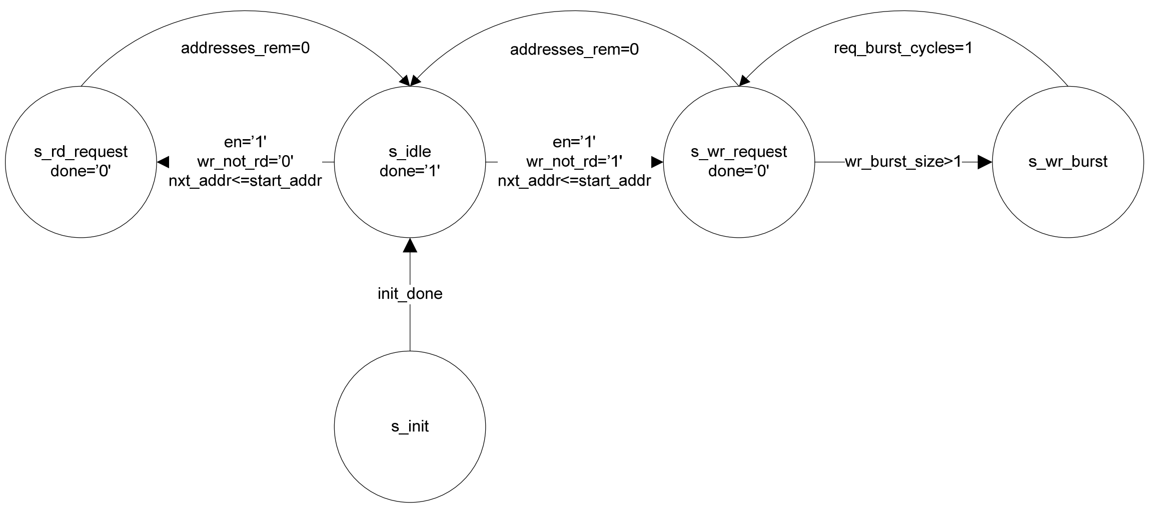

Figure 7: Driver FSM

The main difference between reading from and writing to the DDR3 controller is the need to clock in valid data on a write request, and in case of a burst access, also during the cycles *following* the write request (S_WR_BURST).

6.2.1 Ready latency

Another important aspect is the controller ready (CTLR_RDY) output signal. When this signal is low, the controller cannot accept any more requests or data. This causes gaps in the sequence of requests, the write data and the read back data. The gaps in the requests caused by a de-asserted CTLR_RDY signal are dealt with by the FSM. The gaps in the data are supported by the used streaming interface, however for the write side a ready latency **RL=0** is used. The write FIFO includes a latency adapter to lower the ready latency on its source side to zero. This is necessary to be able to clock a valid data word onto the controller write data bus as soon as CTLR_RDY is asserted. The following paragraphs discuss the states in more detail.

6.2.2 S_IDLE

Once the initialization stages of the DDR3 modules and controller have completed, the FSM goes into the idle state. Once the user has set the desired address range, the WR_NOT_RD bit and enable bit, the start address is assigned and the FSM goes into the first request state.

6.2.3 S_WR_REQUEST

When the controller is ready, and valid data exists on the write bus (RL=0), the driver issues a write burst request to the controller. During this cycle, the first word of the burst data sequence is already sent to the controller and the ready signal to the read side of the write FIFO is asserted to acknowledge this, so valid data will be present on the next cycle. The burst size equals the number of words available in the write FIFO or the maximum burst size of 64, but only if a burst of such size will not exceed the end address. If it would, the burst size is made equal to the remaining number of addresses to write. As said, the first word is already written during this cycle. The remaining words of the burst access will be clocked in during S_WR_BURST.

6.2.4 S_WR_BURST

This state reads the number of required burst cycles that was determined during the request state. During every cycle on which the controller is ready, one valid data word is clocked in. After the last word of the burst access is clocked in, the FSM returns to the request state which, if there are any more addresses to write, will issue another burst write request.

6.2.5 S_RD_REQUEST

A read request is very similar to a write request: a burst read of 1 to 64 can be posted. The difference is that no valid data words need to be clocked in, so if one burst read request has been accepted by the controller (CTLR_READY = ‘1’), a new burst request can be posted on the very next cycle during which the controller is ready. The burst size is determined in the same way as during the write request state, in order not to exceed the set end address.

6.3 Mixed width FIFOs

The ddr3 module utilizes mixed width FIFOs (dp_fifo_dc_mixed_widths) that are available from the DP library. Both FIFOs are of this type, and allow the user to set different data widths on the FIFO read side and the FIFO write side. Figure 8 shows an example of the user sides of the FIFOs set to 32 bits. Although the mixed width FIFOs support it, the ddr3 modules keeps their controller side data widths to a fixed 256 bits. The wr_fifo_nof_usedw signal applies to the read side of the write FIFO, so it always indicates the number of 256-bit words.

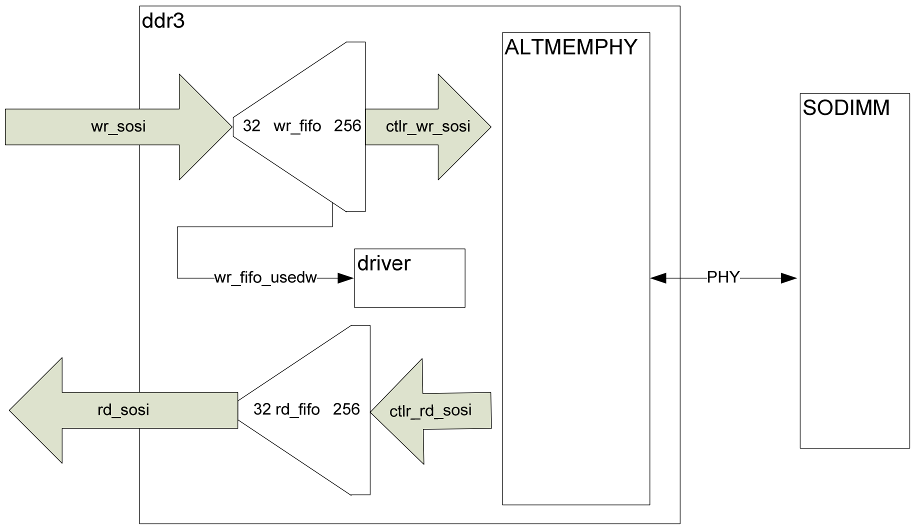

Figure 8: Mixed width FIFOs

Naturally, the average data rate on both sides of the FIFO is the same, but is dictated by the data width and the clock frequency on the user side. In this example, using the same clock on both sides of the FIFOs, the average data rate will be 1/8th of the data rate one would have when setting the user data width to 256 bits.

6.4 Write FIFO flushing

The ddr3 module features a dvr_flush input to allow the user to flush the write FIFO. This is required to prevent gaps in a continuous (unstoppable) stream – during initialization, the ddr3 module's write FIFO (and possibly other FIFO's in the stream) can fill up and data will be thrown away. This is an issue, however as soon as the ddr3 module is initialized, the stream continues and pushes old data forward which will be written into DDR3. To prevent this, the user can assert dvr_flush to keep flushing the write size FIFO during initialization or read sequences that are long enough to affect the stream on the write side. This is done by artificially keeping the ready signal on the read side of the write FIFO asserted, thereby continuously clocking out and discarding the data.
De-asserting dvr_flush causes the write FIFO to fill up starting from the first valid word that enters the FIFO. As mentioned in chapter 4, setting generic g_wr_use_ctrl to TRUE waits for an EOP before FIFO flushing is really disabled. This way, the very first word that enters the write FIFO will do so on an SOP.

6.5 Synthesis

6.5.1 Required Megafunctions

Before a design containing the ddr3 module can be synthesized, both variations of the ALTMEMPHY must be generated:

- aphy_4g_800.vhd;
- aphy_4g_1066.vhd

These files can be found in the IP directory (see chapter 9). Both are required because Quartus will try to analyse all included (.qip) files. For simulation, only the instantiated variation needs to be generated.

6.5.2 Resources

Table 15 summarizes the resource usage of one ddr3 module. The ALTMEMPHY takes up most of the ALMs by far, while M9K block occupation (no M144Ks are used) is only partially due to the ALTMEMPHY. The module's write FIFO takes up 8 M9Ks, because its default depth is 256 words of 256 bits wide. This translates to 8 M9Ks of 36 bits wide, providing the width of 256bits and a depth of 256 words. The read FIFO also occupies 8 M9Ks, even though its default depth is only 128 words. This is caused by the fact that the required width of 256bits is accomplished by using 8 M9Ks of 36bits wide each. The total available depth is 256 words, however only 128 are used – hence the read FIFO uses only half the amount of block memory bits the write FIFO uses.

	ALTMEMPHY	ddr3 VHDL			ddr3 total
		RD FIFO	WR FIFO	Driver	
ALMs	6924 (7,6%)	107 (<1%)	356 (<1%)	137 (<1%)	7524 (8,25%)
M9Ks	13 (1,1%)	8 (<1%)	8 (<1%)	0	29 (2,3%)
Block mem bits	104Ki	64Ki	32Ki	0	200Ki

Table 15: ddr3 module resource usage

The percentages in parentheses relate to the total resources available in an Altera Stratix IV EP4SGX230 FPGA. Percentages are not provided for the amount of block memory bits as block memory usage is generally better represented as occupied M9K blocks.

6.5.3 Synthesis for 1066MT/s or 800MT/s

Synthesizing one or more ddr3 modules in a design requires the following two actions:

- Selecting g_mts to be one of the following:
  - 800;
  - 1066.
- Setting the correct reference CLK in the design's SDC file:
  - 200MHz as the 800MT/s reference clock;
  - 266,67MHz as the 1066MT/s reference clock.

7 Verification

Simulation of the ddr3 module requires a memory model. A memory model is generated when running the ALTMEMPHY Megafunction. Details on this memory model are provided in the next paragraph.

7.1 Altera memory model

Two types of memory models are generated by the Megafunction:

- Full memory model
- Associative array

The full memory model allocates memory for all the complete address range the ddr3 module uses. Unless simulation takes place on a machine with a large amount of RAM, simulation of large memories is not possible. Another downside is that the written data is not visible in the wave window.
The associative array allocates 2K of addresses, and this array can be viewed in the ModelSim wave window. Each array element contains 155 bits. Table 1 lists the contents of these array elements.

Bit range	Width	Contents
154..129	26 bits	address
128..001	128 bits	data
0	1 bit	'1' when data has been written to this array element.

Table 16: Associative array element contents

7.2 Simulation

Verification (refer to [3], [4], [5] for general information regarding development, simulation and the NIOS II) of the ddr3 module is done using the diagnostics module to generate and verify data. This basic scheme is used in three levels:

- tb_ddr3
  - uses VHDL stimuli
- tb_node_unb_ddr3
  - uses VHDL functions to access ddr3 module via dedicated MM bus
  - uses VHDL functions to access diagnostics module via dedicated MM bus
- tb_unb_ddr3
  - conducts the same MM accesses through one shared MM bus using a C application (main.c) running on a NIOS II processor

8 Validation

8.1 Logging

When unb_ddr3.sof is loaded onto one or more FPGAs, the following type of output can be expected after the NOF_TICKS_JTAG_DELAY, and will be updated periodically:

```
Module  Src          Snk          IWR  PASS     FAIL     ERRCODE   tWR   tRD
======  ===          ===          ===  ====     ====     =======   ===   ===
MB_I     134217728   134217728    2    0004438  0000000  0         3217  3133
MB_II    134217728   134217728    2    0004438  0000000  0         3217  3133

FPGA temp :   037 degrees C
Runtime     :  6360 seconds
```

Table 17 gives more information on the JTAG output of unb_ddr3.

Name	**Comment**
MB_I MB_II	Memory Bank I Memory Bank II
Src	Number of sourced (written) 256-bit words per test iteration
Snk	Number of sinked (read) 256-bit words per test iteration
IWR	Indicates the current state: Idle (0) Writing (1) Reading (2)
PASS	Number of OK test iterations
FAIL	Number of failed test iterations
ERRCODE	Error code of last test iteration: 0) OK 1) Source/sink count mismatch 2) Calculated/actual sink count mismatch 3) Invalid diag result 4) Diag result indicates errors (non-zero) Higher error codes override lower error codes, e.g. if an error code 4 is indicated, a lower error code can have occurred also.
tWR	Total time (seconds) of all write operations so far combined, excluding software overhead
tRD	Total time (seconds) of all read operations so far combined, excluding software overhead
FPGA temp	FPGA temperature read out periodically
Runtime	Total runtime, including software overhead

Table 17: unb_ddr3 JTAG output

The source and sink counts in this case apply to 4GB modules: (134.217.728 * 256 bits) / 8 = 4GB.

8.2 Results

The unb_ddr3 design was tested on all 8 nodes. The following parameters were used:

- Data rate: 800MT/s;
  - Writing and reading back 4GiB per module, looped;
  - 62 hours;
  - MB_I and MB_II tested simultaneously;
  - User (diagnostics) data width of 256 bits.

- Data rate: 1066MT/s;
  - Writing and reading back 4GiB per module, looped;
  - 16 hours;
  - MB_I and MB_II tested simultaneously;
  - User (diagnostics) data width of 256 bits.

During both tests, all read back data was verified OK (zero errors). An additional test was performed for several hours, however this time with a diagnostics data width of 32 bits instead of 256. The results were OK, and as expected (see section 6.3), it took 8 times as long to perform the same number of test iterations.

8.2.1 Efficiency

The ddr3 module's efficiency can be derived from the logged test times. The following calculations apply to the 1066MT/s test, but the resulting efficiency will be the same for the 800MT/s version.

8.2.1.1 Theoretical maximum data rate

(HPCII user interface clock frequency * data width) = 266.667MHz*256b = 68267Mb/s = 63.58 Gib/sec

8.2.1.2 Measured data rate:

- 54915 iterations (wr + rd)
  - tWr 30150
  - tRd 29283

Data rate, wr:	(4 GiB * 54915 iterations) / 30150 s = 7.28557GiB/sec = 58.28Gib/sec = 92%
Data rate, rd:	(4 GiB * 54915 iterations) / 29283 s = 7.50128GiB/sec = 60.01Gib/sec = 94%

9 Appendix – list of files

9.1 ddr3

The ddr3 directory contains the source code for the ddr3 module and its components, the optional ddr3_reg and a test bench for the ddr3 module. It also contains an IP folder where the variation files for the two versions of the ALTMEMPHY are stored. It is located at:

$UNB/Firmware/modules/ddr3

9.2 unb_ddr3

The files for synthesis and simulation are in the following directory:

$UNB/Firmware/designs/unb_ddr3

In addition, a C main program is located at:

$UNB/Firmware/software/apps/unb_ddr3