ESA GRLIB IP Core's Manual

Version eval-1.0, May 2005

Jiri Gaisler, Edvin Catovic, Roland Weigand (ESA)

Gaisler Research, 2005.

Table of contents

1 Introduction

1.1 Scope

This doument describes specific IP cores provided by the European SPace Agency inside the GRLIB IP library. When applicable, the cores use the GRLIP plug&play configuration method as described in the ‘GRLIB User’s Manual’. These cores has been ported from the LEON2 VHDL model, and are provided under the GNU LGPL license.

1.2 IP core overview

The tables below lists the provided IP cores and their AMBA plug&play device ID. All cores use vendor ID 0x04 (ESA).

TABLE 1. Serial communications

Name	**Function**	**Device ID**	**License**
L2UART	LEON2 Generic UART	0x008	LGPL

TABLE 2. PCI functions

Name	**Function**	**Device ID**	**License**
PCIARB	PCI arbiter with optional APB interface	0x010	LGPL

TABLE 3. PCI functions

Name	**Function**	**Device ID**	**License**
PCIARB	PCI arbiter with optional APB interface	0x010	LGPL

2 Memory Controller

2.1 Overview

The memory controller handles a memory bus hosting PROM, memory mapped I/O devices, asynchronous static ram (SRAM) and synchronous dynamic ram (SDRAM). The controller acts as a slave on the AHB bus. The function of the memory controller is programmed through memory configuration registers 1, 2 & 3 (MCR1, MCR2 & MCR3) through the APB bus. The memory bus supports four types of devices: prom, sram, sdram and local I/O. The memory bus can also be configured in 8- or 16-bit mode for applications with low memory and performance demands. The controller decodes three address spaces (PROM, I/O and RAM) whose mapping is determined through VHDL-generics.

Chip-select decoding is done for two PROM banks, one I/O bank, five SRAM banks and two SDRAM banks. Figure 1 shows how the connection to the different device types is made.

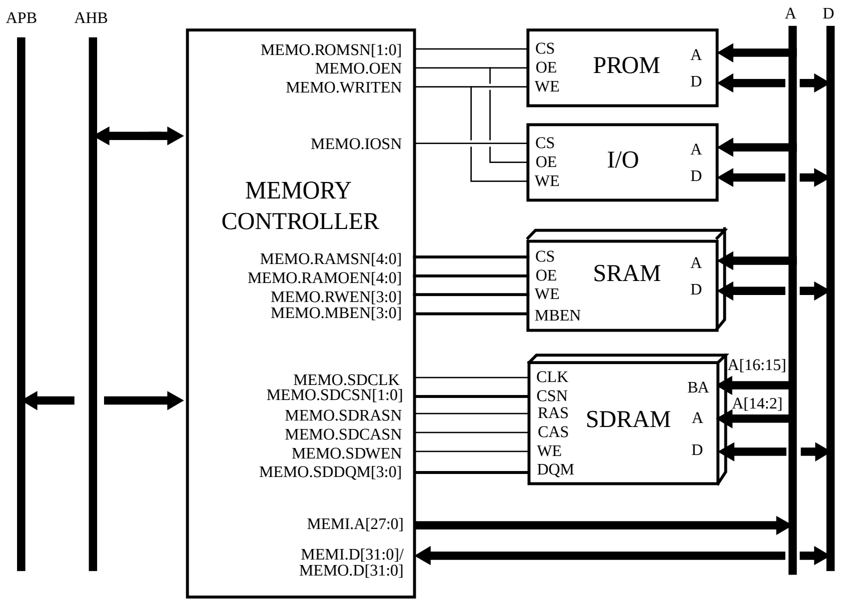

Figure 1. Memory controller conected to AMBA bus and different types of memory devices

2.2 PROM access

Accesses to prom have the same timing as RAM accesses, the differences being that PROM cycles can have up to 15 waitstates.

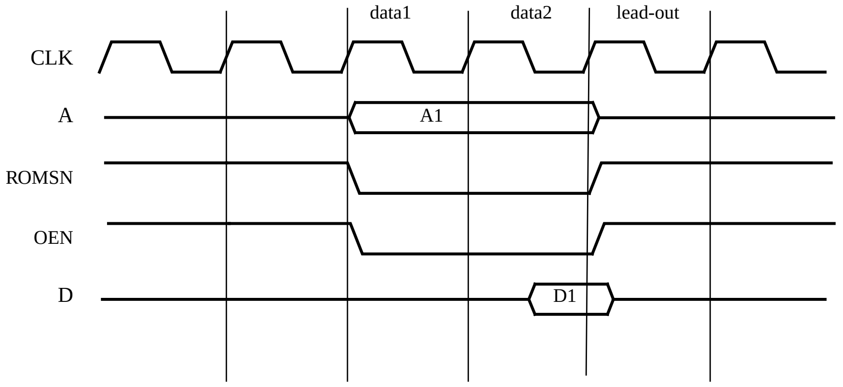

Figure 2. Prom read cycle

Two PROM chip-select signals are provided, MEMO.ROMSN[1:0]. MEMO.ROMSN[0] is asserted when the lower half of the PROM area as addressed while MEMO.ROMSN[1] is asserted for the upper half. When the VHDL model is configured to boot from internal prom, MEMO.ROMSN[0] is never asserted and all accesses to the lower half of the PROM area are mapped on the internal prom.

2.3 Memory mapped I/O

Accesses to I/O have similar timing to ROM/RAM accesses, the differences being that a additional waitstates can be inserted by de-asserting the MEMI.BRDYN signal. The I/O select signal (MEMO.IOSN) is delayed one clock to provide stable address before MEMO.IOSN is asserted.

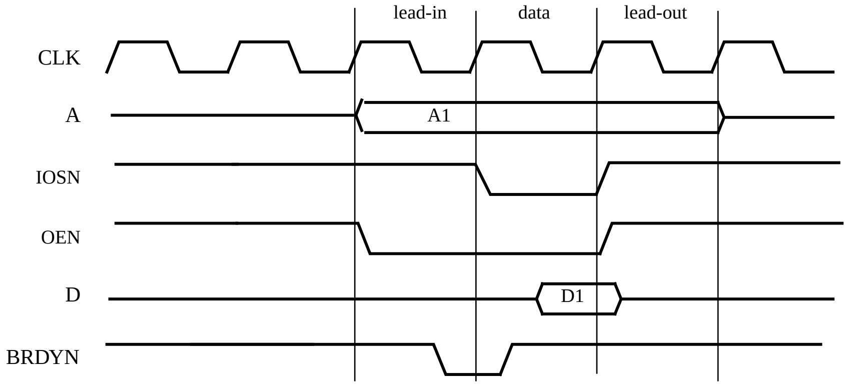

Figure 3. I/O read cycle

2.4 SRAM access

The SRAM area can be up to 1 Gbyte, divided on up to five RAM banks. The size of banks 1-4 (MEMO.RAMSN[3:0]is programmed in the RAM bank-size field (MCR2[12:9]) and can be set in binary steps from 8 Kbyte to 256 Mbyte. The fifth bank (MEMO.RAMSN[4]) decodes the upper 512 Mbyte. A read access to SRAM consists of two data cycles and between zero and three wait-states. Accesses to MEMO.RAMSN[4] can further be stretched by de-asserting MEMI.BRDYN until the data is available. On non-consecutive accesses, a lead-out cycle is added after a read cycle to prevent bus contention due to slow turn-off time of memories or I/O devices. Figure 4 shows the basic read cycle waveform (zero waitstate).

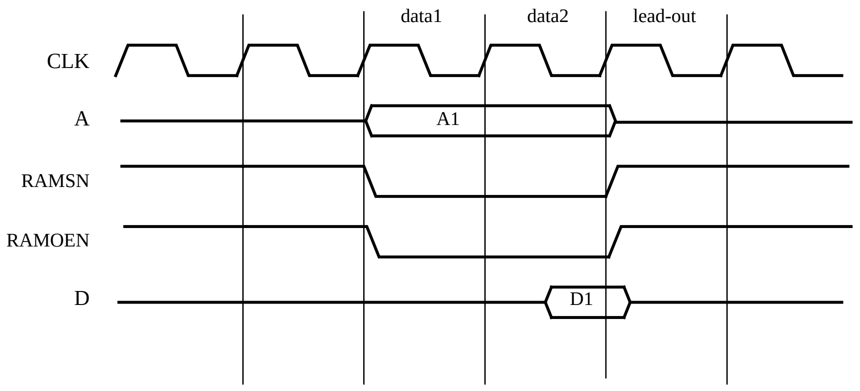

Figure 4. Static ram read cycle (0-waitstate)

For read accesses to MEMO.RAMSN[4:0], a separate output enable signal (MEMO.RAMOEN[n]) is provided for each RAM bank and only asserted when that bank is selected. A write access is similar to the read access but takes a minimum of three cycles:

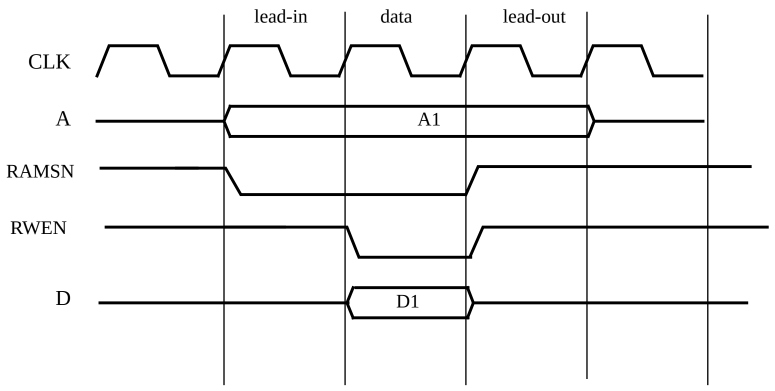

Figure 5. Static ram write cycle

Through an (optional) feed-back loop from the write strobes, the data bus is guaranteed to be driven until the write strobes are de-asserted. Each byte lane has an individual write strobe to allow efficient byte and half-word writes. If the memory uses a common write strobe for the full 16- or 32-bit data, the read-modify-write bit MCR2 should be set to enable read-modify-write cycles for sub-word writes.

A drive signal vector for the data I/O-pads is provided which has one drive signal for each data bit. It can be used if the synthesis tool does not generate separate registers automatically for the current technology. This can remove timing problems with output delay.

2.5 8-bit and 16-bit PROM and SRAM access

To support applications with low memory and performance requirements efficiently, it is not necessary to always have full 32-bit memory banks. The SRAM and PROM areas can be individually configured for 8- or 16-bit operation by programming the ROM and RAM size fields in the memory configuration registers. Since read access to memory is always done on 32-bit word basis, read access to 8-bit memory will be transformed in a burst of four read cycles while access to 16-bit memory will generate a burst of two 16-bits reads. During writes, only the necessary bytes will be writen. Figure 6 shows an interface example with 8-bit PROM and 8-bit SRAM. Figure 7 shows an example of a 16-bit memory interface.

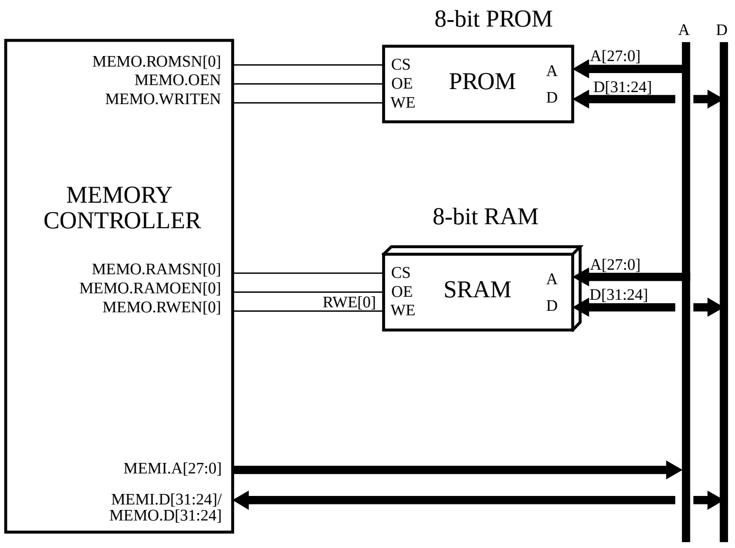

Figure 6. 8-bit memory interface example

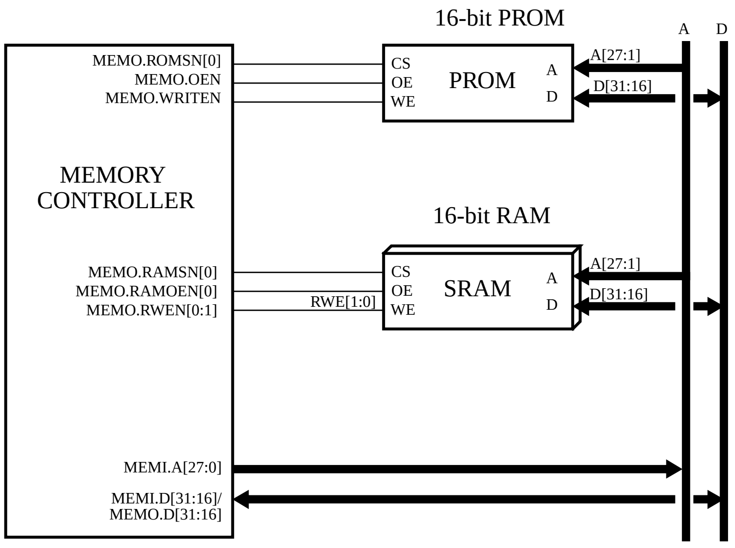

Figure 7. 16-bit memory interface example

2.6 Burst cycles

To improve the bandwidth of the memory bus, accesses to consecutive addresses can be performed in burst mode. Burst transfers will be generated when the memory controller is accessed using an AHB burst request. These includes instruction cache-line fills, double loads and double stores. The timing of a burst cycle is identical to the programmed basic cycle with the exception that during read cycles, the lead-out cycle will only occurs after the last transfer.

2.7 8- and 16-bit I/O access

Similar to the PROM/RAM areas, the I/O area can also be configured to 8- or 16-bits mode. However, the I/O device will NOT be accessed by multiple 8/16 bits accesses as the memory areas, but only with one single access just as in 32-bit mode. To accesses an I/O device on a 16-bit bus, LDUH/STH instructions should be used while LDUB/STB should be used with an 8-bit bus.

2.8 SDRAM access

2.8.1 General

Synchronous dynamic RAM (SDRAM) access is supported to two banks of PC100/PC133 compatible devices. This is implemented by a special version of the SDCTRL SDRAM controller core from Gaisler Research, which is optionally instantiated as a sub-block. The SDRAM controller supports 64M, 256M and 512M devices with 8 - 12 column-address bits, and up to 13 row-address bits. The size of the two banks can be programmed in binary steps between 4 Mbyte and 512 Mbyte. The operation of the SDRAM controller is controlled through MCFG2 and MCFG3 (see below). Both 32- and 64-bit data bus width is supported, allowing the interface of 64-bit DIMM modules. The memory controller can be configured to use either a shared or separate bus connecting the controller and SDRAM devices.

2.8.2 Address mapping

The two SDRAM chip-select signals are decoded. SDRAM area is mapped into the upper half of the RAM area defined by BAR2 register. When the SDRAM enable bit is set in MCFG2, the controller is enabled and mapped into upper half of the RAM area as long as the SRAM disable bit is not set. If the SRAM disable bit is set, all access to SRAM is disabled and the SDRAM banks are mapped into the lower half of the RAM area.

2.8.3 Initialisation

When the SDRAM controller is enabled, it automatically performs the SDRAM initialisation sequence of PRECHARGE, 2x AUTO-REFRESH and LOAD-MODE-REG on both banks simultaneously. The controller programs the SDRAM to use page burst on read and single location access on write.

2.8.4 Configurable SDRAM timing parameters

To provide optimum access cycles for different SDRAM devices (and at different frequencies), some SDRAM parameters can be programmed through memory configuration register 2 (MCFG2) The programmable SDRAM parameters can be seen in table 4:

TABLE 4. SDRAM programmable timing parameters

Function	Parameter	range	unit
CAS latency, RAS/CAS delay	t_{CAS}, t_{RCD}	2 - 3	clocks
Precharge to activate	t_{RP}	2 - 3	clocks

TABLE 4. SDRAM programmable timing parameters

Function	Parameter	range	unit
Auto-refresh command period	t_{RFC}	3 - 11	clocks
Auto-refresh interval		10 - 32768	clocks

Remaining SDRAM timing parameters are according the PC100/PC133 specification.

2.9 Refresh

The SDRAM controller contains a refresh function that periodically issues an AUTO-REFRESH command to both SDRAM banks. The period between the commands (in clock periods) is programmed in the refresh counter reload field in the MCFG3 register. Depending on SDRAM type, the required period is typically 7.8 or 15.6 µs (corresponding to 780 or 1560 clocks at 100 MHz). The generated refresh period is calculated as (reload value+1)/sysclk. The refresh function is enabled by setting bit 31 in MCFG2.

2.9.1 SDRAM commands

The controller can issue three SDRAM commands by writing to the SDRAM command field in MCFG2: PRE-CHARGE, AUTO-REFRESH and LOAD-MODE-REG (LMR). If the LMR command is issued, the CAS delay as programmed in MCFG2 will be used, remaining fields are fixed: page read burst, single location write, sequential burst. The command field will be cleared after a command has been executed. Note that when changing the value of the CAS delay, a LOAD-MODE-REGISTER command should be generated at the same time.

2.9.2 Read cycles

A read cycle is started by performing an ACTIVATE command to the desired bank and row, followed by a READ command after the programmed CAS delay. A read burst is performed if a burst access has been requested on the AHB bus. The read cycle is terminated with a PRE-CHARGE command, no banks are left open between two accesses.

2.9.3 Write cycles

Write cycles are performed similarly to read cycles, with the difference that WRITE commands are issued after activation. A write burst on the AHB bus will generate a burst of write commands without idle cycles in-between.

2.9.4 Address bus connection

The memory controller can be configured to either share the address and data buses with the SRAM, or to use separate address and data buses. When the buses are shared, the address bus of the SDRAMs should be connected to A[14:2], the bank address to A[16:15]. The MSB part of A[14:2] can be left unconnected if not used. When separate buses are used, the SDRAM address bus should be connected to SA[12:0] and the bank address to SA[14:13].

2.9.5 Data bus

SDRAM can be connected to the memory controller through the common or separate data bus. If the separate bus is used the width is configurable to 32 or 64 bits. 64-bit data bus allows the 64-bit SDRAM devices to be connected using the full data capacity of the devices. 64-bit SDRAM devices can be connected to 32-bit data bus if 64-bit data bus is not available but in this case only half the full data capacity will be used. There is a drive signal vector and separate data vector available for SDRAM. The drive vector has one drive signal for each data bit. These signals can be used to remove timing problems with the output delay when a separate SDRAM bus is used. SDRAM bus signals are described in chapter 2.10, for configuration options refer to chapter 2.14.

2.9.6 Clocking

The SDRAM clock typically requires special synchronisation at layout level. For Xilinx and Altera device, the GR Clock Generator can be configured to produce a properly synchronised SDRAM clock. For other FPGA targets, the GR Clock Generator can produce an inverted clock.

2.10 Registers

The memory controller is programmed through tree registers mapped into APB address space.

TABLE 5. Memory controller registers

Register	APB Address offset
MCFG1	0x0
MCFG2	0x4
MCFG3	0x8

2.10.1 Memory configuration register 1 (MCFG1)

Memory configuration register 1 is used to program the timing of rom and local I/O accesses.

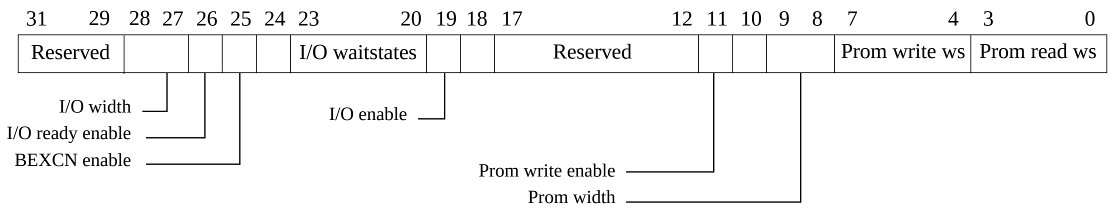

Figure 8. Memory configuration register 1

[3:0]: Prom read waitstates. Defines the number of waitstates during prom read cycles (“0000”=0, “0001”=1,... “1111”=15).

[7:4]: Prom write waitstates. Defines the number of waitstates during prom write cycles (“0000”=0, “0001”=1,... “1111”=15).

[9:8]: Prom width. Defines the data with of the prom area (“00”=8, “01”=16, “10”=32).

[10]: Reserved

[11]: Prom write enable. If set, enables write cycles to the prom area.

[17:12]: Reserved

[19]: I/O enable. If set, the access to the memory bus I/O area are enabled.

[23:20]: I/O waitstates. Defines the number of waitstates during I/O accesses (“0000”=0, “0001”=1, “0010”=2,..., “1111”=15).

[25]: Bus error (BEXCN) enable.

[26]:Bus ready (BRDYN) enable.

[28:27]: I/O bus width. Defines the data with of the I/O area (“00”=8, “01”=16, “10”=32).

During power-up, the prom width (bits [9:8]) are set with value on MEMI.BWIDTH inputs. The prom waitstates fields are set to 15 (maximum). External bus error and bus ready are disabled. All other fields are undefined.

2.10.2 Memory configuration register 2 (MCFG2)

Memory configuration register 2 is used to control the timing of the SRAM and SDRAM.

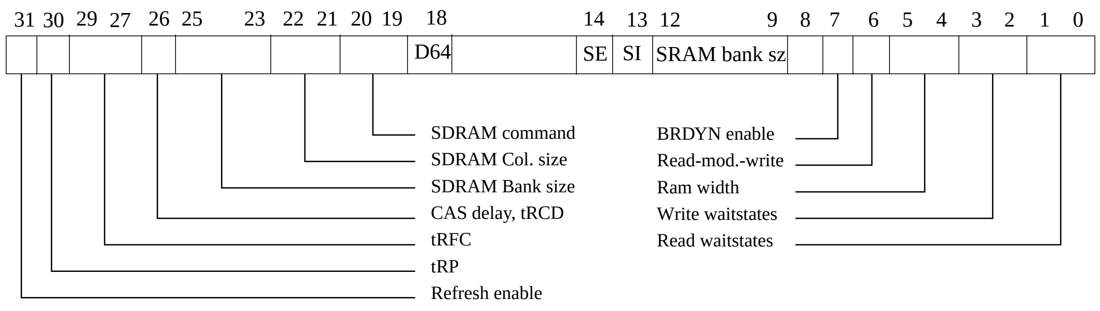

Figure 9. Memory configuration register 2

[1:0]: Ram read waitstates. Defines the number of waitstates during ram read cycles (“00”=0, “01”=1, “10”=2, “11”=3).

[3:2]: Ram write waitstates. Defines the number of waitstates during ram write cycles (“00”=0, “01”=1, “10”=2, “11”=3).

[5:4]: Ram with. Defines the data with of the ram area (“00”=8, “01”=16, “1X”= 32).

[6]: Read-modify-write. Enable read-modify-write cycles on sub-word writes to 16- and 32-bit areas with common write strobe (no byte write strobe).

[7]: Bus ready enable. If set, will enable BRDYN for ram area

[12:9]: Ram bank size. Defines the size of each ram bank (“0000”=8 Kbyte, “0001”=16 Kbyte... “1111”=256 Mbyte).

[13]: SI - SRAM disable. If set together with bit 14 (SDRAM enable), the static ram access will be disabled.

[14]: SE - SDRAM enable. If set, the SDRAM controller will be enabled.

[18]: 64-bit data bus (D64) - Reads ‘1’ if memory controller is configured for 64-bit data bus, otherwise ‘0’. Read-only.

[20:19] SDRAM command. Writing a non-zero value will generate an SDRAM command: “01”=PRECHARGE, “10”=AUTO-REFRESH, “11”=LOAD-COMMAND-REGISTER. The field is reset after command has been executed.

[22:21]: SDRAM column size. “00”=256, “01”=512, “10”=1024, “11”=4096 when bit[25:23]= “111”, 2048 otherwise.

[25:23]: SDRAM banks size. Defines the banks size for SDRAM chip selects: “000”=4 Mbyte, “001”=8 Mbyte, “010”=16 Mbyte .... “111”=512 Mbyte.

[26]: SDRAM CAS delay. Selects 2 or 3 cycle CAS delay (0/1). When changed, a LOAD-COMMAND-REGISTER command must be issued at the same time. Also sets RAS/ CAS delay (tRCD).

[29:27]: SDRAM t_{RFC} timing. t_{RFC} will be equal to 3 + field-value system clocks.

[30]: SDRAM t_{RP} timing. t_{RP} will be equal to 2 or 3 system clocks (0/1).

[31]: SDRAM refresh. If set, the SDRAM refresh will be enabled.

2.10.3 Memory configuration register 3 (MCFG3)

MCFG3 is contains the reload value for the SDRAM refresh counter.

31 27	26 12	11 0
RESERVED	SDRAM refresh reload value	RESERVED

Figure 10. Memory configuration register 3

The period between each AUTO-REFRESH command is calculated as follows:

$t_{REFRESH}$ = ((reload value) + 1) / SYSCLK

2.11 Using MEMI.BRDYN

The MEMI.BRDYN signal can be used to stretch access cycles to the I/O area and the ram area decoded by MEMO.RAMSN[4]. The accesses will always have at least the pre-programmed number of waitstates as defined in memory configuration registers 1 & 2, but will be further stretched until MEMI.BRDYN is asserted. MEMI.BRDYN should be asserted in the cycle preceding the last one. The use of MEMI.BRDYN can be enabled separately for the I/O and RAM areas.

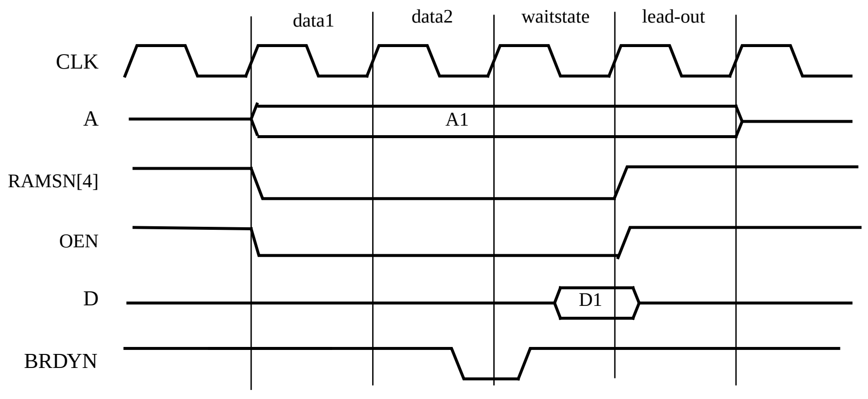

Figure 11. RAM read cycle with one BRDYN controlled waitstate

2.12 Access errors

An access error can be signalled by asserting the MEMI.BEXCN signal, which is sampled together with the data. If the usage of MEMI.BEXCN is enabled in memory configuration register 1, an error response will be generated on the internal AMBA bus. MEMI.BEXCN can be enabled or disabled through memory configuration register 1, and is active for all areas (PROM, I/O an RAM).

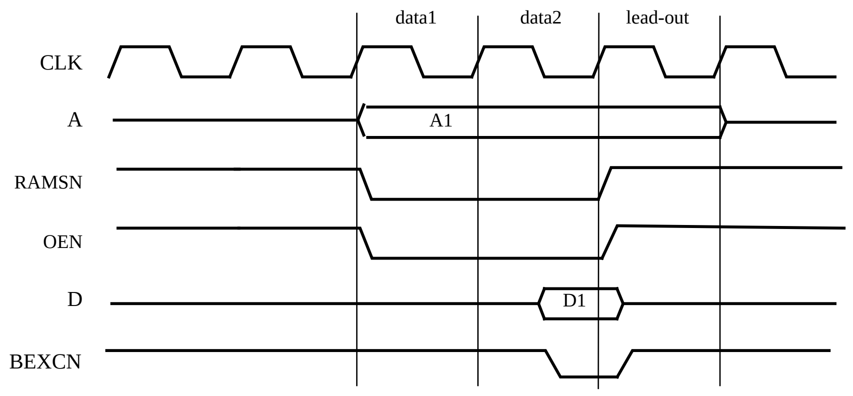

Figure 12. Read cycle with BEXCN

2.13 Attaching an external DRAM controller

To attach an external DRAM controller, MEMO.RAMSN[4] should be used since it allows the cycle time to vary through the use of MEMI.BRDYN. In this way, delays can be inserted as required for opening of banks and refresh.

2.14 Configuration options

The Memory controller has the following configuration options (VHDL generics):

TABLE 6. Memory controller configuration options (VHDL generics)

Generic	Function	Allowed range	Default
hindex	AHB slave index	1 - NAHBSLV-1	0
pindex	APB slave index	0 - NAPBSLV-1	0
romaddr	ADDR filed of the AHB BAR0 defining PROM address space. Default PROM area is 0x0 - 0x1FFFFFFF.	0 - 16#FFF#	16#000#
rommask	MASK filed of the AHB BAR0 defining PROM address space.	0 - 16#FFF#	16#E00#
ioaddr	ADDR filed of the AHB BAR1 defining I/O address space. Default I/O area is 0x20000000 - 0x2FFFFFFF.	0 - 16#FFF#	16#200#
iomask	MASK filed of the AHB BAR1 defining I/O address space.	0 - 16#FFF#	16#E00#
ramaddr	ADDR filed of the AHB BAR2 defining RAM address space. Default RAM area is 0x40000000-0x7FFFFFFF.	0 - 16#FFF#	16#400#
rammask	MASK filed of the AHB BAR2 defining RAM address space.	0 -16#FFF#	16#C00 #
paddr	ADDR filed of the APB BAR configuration registers address space.	0 - 16#FFF#	0
pmask	MASK filed of the APB BAR configuration registers address space.	0 - 16#FFF#	16#FFF #
wprot	RAM write protection.	0 - 1	0
invclk	Inverted clock is used for the SDRAM.	0 - 1	0
fast	Enable fast SDRAM address decoding.	0 - 1	0
romasel	log2(PROM address space size) - 1. E.g. if size of the PROM area is 0x20000000 romasel is log2(2^29)-1 = 28.	0 - 31	28
sdrasel	log2(RAM address space size) - 1. E.g if size of the RAM address space is 0x40000000 sdrasel is log2(2^30)-1= 29.	0 - 31	29
srbanks	Number of SRAM banks.	0 - 5	4
ram8	Enable 8-bit PROM and SRAM access.	0 - 1	0
ram16	Enable 16-bit PROM and SRAM access.	0 - 1	0
sden	Enable SDRAM controller.	0 - 1	0
sepbus	SDRAM is located on separate bus.	0 - 1	1
sdbits	32 or 64 -bit SDRAM data bus.	32, 64	32
oepol	Select polarity of drive signals for data pads. 0 = active low, 1 = active high.	0 - 1	0

2.15 Vendor and device id

The module has vendor id 0x04 (ESA) and device id 0x00F. For description of vendor and device ids see GRLIB IP Library User’s Manual.

2.16 Signal description

Memory controller signals are described in table 7.

TABLE 7. Memory controller signal description.

Signal name	Field	Type	Function	Active
CLK	N/A	Input	Clock	-
RST	N/A	Input	Reset	Low
MEMI	DATA[31:0]	Input	Memory data	High
	BRDYN	Input	Bus ready strobe	Low
	BEXCN	Input	Bus exception	Low
	WRN[3:0]	Input	SRAM write enable feedback signal	Low
	BWIDTH[1:0]	Input	Prom width. This value is written in Prom Width field of MCFG0 register.	High
	SD[31:0]	Input	SDRAM separate data bus	High
MEMO	ADDRESS[27:0]	Output	Memory address	High
	DATA[31:0]	Output	Memory data	-
	SDDATA[63:0]	Output	Sdram memory data	-
	RAMSN[4:0]	Output	SRAM chip-select	Low
	RAMOEN[4:0]	Output	SRAM output enable	Low
	IOSN	Output	Local I/O select	Low
	ROMSN[1:0]	Output	PROM chip-select	Low
	OEN	Output	Output enable	Low
	WRITEN	Output	Write strobe	Low
	WRN[3:0]	Output	SRAM write enable	Low
	MBEN[3:0]	Output	Byte enable	Low
	BDRIVE[3:0]	Output	Drive byte lanes on external memory bus. Controls I/O-pads connected to external memory bus.	Low/High
	VBDRIVE[31:0]	Output	Vectored I/O-pad drive signals.	Low/High
	SVBDRIVE[63:0]	Output	Vectored I/O-pad drive signals for separate sdram bus.	Low/High
	READ	Output	Read strobe	High
	SA[14:0]	Output	SDRAM separate address bus	High
AHBSI	*	Input	AHB slave input signals	-
AHBSO	*	Output	AHB slave output signals	-
APBI	*	Input	APB slave input signals	-
APBO	*	Output	APB slave output signals	-
WPROT	WPROTHIT	Input	Unused	-
SDO	SDCASN	Output	SDRAM column address strobe	Low
	SDCKE[1:0]	Output	SDRAM clock enable	High
	SDCSN[1:0]	Output	SDRAM chip select	Low
	SDDQM[7:0]	Output	SDRAM data mask	Low
	SDRASN	Output	SDRAM row address strobe	Low
	SDWEN	Output	SDRAM write enable	Low

* see GRLIB IP Library User’s Manual

2.17 Library dependencies

Table shows libraries that the memory controller module depends on.

TABLE 8. Library dependencies

Library	Package	Imported unit(s)	Description
GRLIB	AMBA	Signals	AHB signal definitions
GAISLER	MEMCTRL	Signals Components	Memory bus signals definitions SDMCTRL component
ESA	MEMORYCTRL	Component	Memory controller component declaration

2.18 Memory controller instantiation

This examples shows how a memory controller can be instantiated. The example design contains an AMBA bus with a number of AHB components connected to it including the memory controller. The external memory bus is defined on the example designs port map and connected to the memory controller. System clock and reset are generated by GR Clock Generator and Reset Generator.

Memory controller decodes default memory areas: PROM area is 0x0 - 0x1FFFFFFF, I/O-area is 0x20000000-0x3FFFFFFF and RAM area is 0x40000000 - 0x7FFFFFFF. SDRAM controller is enabled. SDRAM clock is synchronized with system clock by clock generator.

```
library ieee;
use ieee.std_logic_1164.all;

library grlib;
use grlib.amba.all;
use grlib.tech.all;
library gaisler;
use gaisler.memctrl.all;
use gaisler.pads.all;   -- used for I/O pads
library esa;
use esa.memoryctrl.all;

entity mctrl_ex is
  port (
    clk : in std_ulogic;
    resetn : in std_ulogic;
    pllref : in  std_ulogic;

    -- memory bus
    address  : out   std_logic_vector(27 downto 0); -- memory bus
    data     : inout std_logic_vector(31 downto 0);
    ramsn    : out   std_logic_vector(4 downto 0);
    ramoen   : out   std_logic_vector(4 downto 0);
    rwen     : inout std_logic_vector(3 downto 0);
    romsn    : out   std_logic_vector(1 downto 0);
    iosn     : out   std_logic;
    oen      : out   std_logic;
    read     : out   std_logic;
    writen   : inout std_logic;
    brdyn    : in    std_logic;
    bexcn    : in    std_logic;
-- sdram i/f
    sdcke    : out std_logic_vector ( 1 downto 0);  -- clk en
    sdcsn    : out std_logic_vector ( 1 downto 0);  -- chip sel
    sdwen    : out std_logic;                       -- write en
    sdrasn   : out std_logic;                       -- row addr stb
    sdcasn   : out std_logic;                       -- col addr stb
    sddqm    : out std_logic_vector (7 downto 0);  -- data i/o mask
    sdclk    : out std_logic;                       -- sdram clk output
    sa       : out std_logic_vector(14 downto 0); -- optional sdram address
    sd       : inout std_logic_vector(63 downto 0) -- optional sdram data
    );
end;

architecture rtl of mctrl_ex is

  -- AMBA bus (AHB and APB)
  signal apbi  : apb_slv_in_type;
  signal apbo  : apb_slv_out_vector := (others => apb_none);
  signal ahbsi : ahb_slv_in_type;
  signal ahbso : ahb_slv_out_vector := (others => ahbs_none);
  signal ahbmi : ahb_mst_in_type;
  signal ahbmo : ahb_mst_out_vector := (others => ahbm_none);

  -- signals used to connect memory controller and memory bus
  signal memi : memory_in_type;
  signal memo : memory_out_type;

  signal sdo : sdram_out_type;

  signal wprot : wprot_out_type;  -- dummy signal, not used
  signal clkm, rstn : std_ulogic; -- system clock and reset

-- signals used by clock and reset generators
  signal cgi : clkgen_in_type;
  signal cgo : clkgen_out_type;

  signal gnd : std_ulogic;

begin

  -- Clock and reset generators
  clkgen0 : clkgen generic map (clk_mul => 2, clk_div => 2, sdramen => 1,
                                tech => virtex2, sdinvclk => 0)
  port map (clk, gnd, clkm, open, open, sdclk, open, cgi, cgo);

  cgi.pllctrl <= "00"; cgi.pllrst <= resetn; cgi.pllref <= pllref;

  -- Memory controller
  mctrl0 : mctrl generic map (srbanks => 1, sden => 1)
    port map (rstn, clkm, memi, memo, ahbsi, ahbso(0), apbi, apbo(0), wprot, sdo);

  -- memory controller inputs not used in this configuration
  memi.brdyn <= ’1’; memi.bexcn <= ’1’; memi.wrn <= "1111";
  memi.sd <= sd;

  -- prom width at reset
  memi.bwidth <= "10";

  -- I/O pads driving data memory bus data signals
  datapads : for i in 0 to 3 generate
      data_pad : iopadv generic map (width => 8)
      port map (pad => data(31-i*8 downto 24-i*8),
                o => memi.data(31-i*8 downto 24-i*8),
                en => memo.bdrive(i),
                i => memo.data(31-i*8 downto 24-i*8));
  end generate;

  -- connect memory controller outputs to entity output signals
  address <= memo.address; ramsn <= memo.ramsn; romsn <= memo.romsn;
  oen <= memo.oen; rwen <= memo.wrn; ramoen <= "1111" & memo.ramoen(0);
  sa <= memo.sa;
  writen <= memo.writen; read <= memo.read; iosn <= memo.iosn;
  sdcke <= sdo.sdcke; sdwen <= sdo.sdwen; sdcsn <= sdo.sdcsn;
  sdrasn <= sdo.rasn; sdcasn <= sdo.casn; sddqm <= sdo.dqm;
end;
```

3 LEON2 Generic UART

3.1 Overview

The Generic UART is a universal asynchronous receiver/transmitter, originally developed for the LEON2 processor. The UART support data frames with 8 data bits, one optional parity bit and one stop bit. To generate the bit-rate, each UART has a programmable 12-bits clock divider. Hardware flow-control is supported through the RTSN/CTSN hand-shake signals. Figure 13 shows a block diagram of the UART.

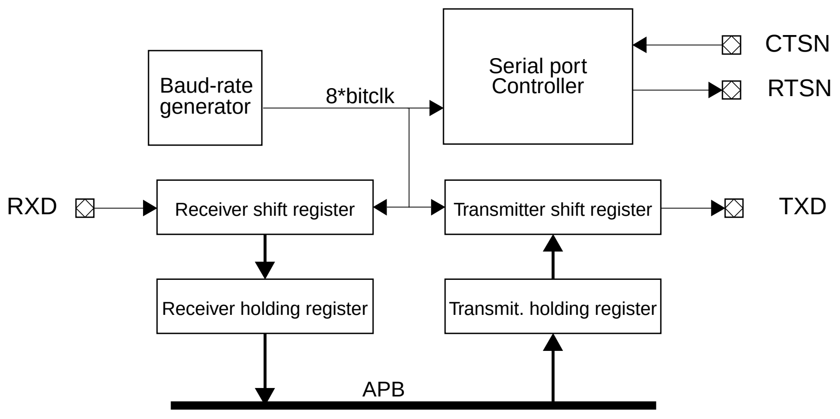

Figure 13. LEON2 UART block diagram

3.2 Configuration options

The UART has the following configuration options (VHDL generics):

TABLE 9. UART configuration options (VHDL generics)

Generic	Function	Allowed range	Default
pindex	APB slave index	0 - NAPBSLV-1	0
paddr	ADDR filed of the APB BAR.	0 - 16#FFF#	0
pmask	MASK filed of the APB BAR.	0 - 16#FFF#	16#FFF#
console	Prints output form the UART on console during VHDL simulation and speeds up simulation by always returning ‘1’ for Data Ready bit of UART Status register. Does not effect synthesis.	0 - 1	0
pirq	Index of the interrupt line.	0 - NAHBIRQ-1	0

3.3 Vendor and device id

The module has vendor id 0x04 (ESA) and device id 0x008. For description of venor and device IDs, see GRLIB IP Library User’s Manual.

3.4 Operation

3.4.1 Transmitter operation

The transmitter is enabled through the TE bit in the UART control register. When ready to transmit, data is transferred from the transmitter holding register to the transmitter shift register and converted to a serial stream on the transmitter serial output pin (TXD). It automatically sends a start bit followed by eight data bits, an optional parity bit, and one stop bits (figure 14). The least significant bit of the data is sent first

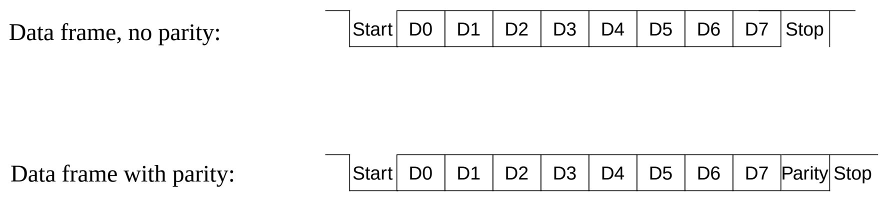

Figure 14. UART data frames

Following the transmission of the stop bit, if a new character is not available in the transmitter holding register, the transmitter serial data output remains high and the transmitter shift register empty bit (TSRE) will be set in the UART control register. Transmission resumes and the TSRE is cleared when a new character is loaded in the transmitter holding register. If the transmitter is disabled, it will continue operating until the character currently being transmitted is completely sent out. The transmitter holding register cannot be loaded when the transmitter is disabled.

If flow control is enabled, the CTSN input must be low in order for the character to be transmitted. If it is deasserted in the middle of a transmission, the character in the shift register is transmitted and the transmitter serial output then remains inactive until CTSN is asserted again. If the CTSN is connected to a receivers RTSN, overrun can effectively be prevented.

3.4.2 Receiver operation

The receiver is enabled for data reception through the receiver enable (RE) bit in the USART control register. The receiver looks for a high to low transition of a start bit on the receiver serial data input pin. If a transition is detected, the state of the serial input is sampled a half bit clocks later. If the serial input is sampled high the start bit is invalid and the search for a valid start bit continues. If the serial input is still low, a valid start bit is assumed and the receiver continues to sample the serial input at one bit time intervals (at the theoretical centre of the bit) until the proper number of data bits and the parity bit have been assembled and one stop bit has been detected. The serial input is shifted through an 8-bit shift register where all bits have to have the same value before the new value is taken into account, effectively forming a low-pass filter with a cut-off frequency of 1/8 system clock.

During reception, the least significant bit is received first. The data is then transferred to the receiver holding register (RHR) and the data ready (DR) bit is set in the USART status register. The parity, framing and overrun error bits are set at the received byte boundary, at the same time as the receiver ready bit is set. If both receiver holding and shift registers contain an un-read character when a new start bit is detected, then the character held in the receiver shift register will be lost and the overrun bit will be set in the UART status register. If flow control is enabled, then the RTSN will be negated (high) when a valid start bit is detected and the receiver holding register contains an un-read character. When the holding register is read, the RTSN will automatically be reasserted again.

3.4.3 Baud-rate generation

Each UART contains a 12-bit down-counting scaler to generate the desired baud-rate. The scaler is clocked by the system clock and generates a UART tick each time it underflows. The scaler is reloaded with the value of the UART scaler reload register after each underflow. The resulting UART tick frequency should be 8 times the desired baud-rate. If the EC bit is set, the scaler will be clocked by the UARTI.EXTCLK input rather than the system clock. In this case, the frequency of UARTI.EXTCL must be less than half the frequency of the system clock.

3.4.4 Loop back mode

If the LB bit in the UART control register is set, the UART will be in loop back mode. In this mode, the transmitter output is internally connected to the receiver input and the RTSN is connected to the CTSN. It is then possible to perform loop back tests to verify operation of receiver, transmitter and associated software routines. In this mode, the outputs remain in the inactive state, in order to avoid sending out data.

3.4.5 Interrupt generation

The UART will generate an interrupt under the following conditions: when the transmitter is enabled, the transmitter interrupt is enabled and the transmitter holding register moves from full to empty; when the receiver is enabled, the receiver interrupt is enabled and the receiver holding register moves from empty to full; when the receiver is enabled, the receiver interrupt is enabled and a character with either parity, framing or overrun error is received.

3.5 UART registers

The UART is programmed through tree registers mapped into APB address space.

TABLE 10. Uart registers

Register	APB Address offset
UART Data register	0x0
UART Status register	0x4
UART Control register	0x8
UART Scaler register	0xC

3.5.1 UART Data Register

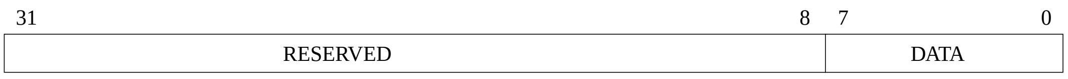

Figure 15. UART data register

[7:0]: Receiver holding register (read access

[7:0]: Transmitter holding register (write access)

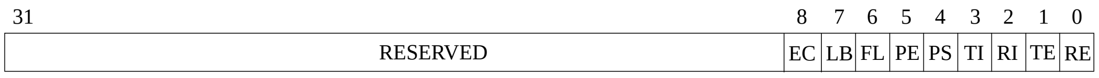

Figure 16. UART control register

0: Receiver enable (RE) - if set, enables the receiver.

1: Transmitter enable (TE) - if set, enables the transmitter.

2: Receiver interrupt enable (RI) - if set, enables generation of receiver interrupt.

3: Transmitter interrupt enable (TI) - if set, enables generation of transmitter interrupt.
4: Parity select (PS) - selects parity polarity (0 = even parity, 1 = odd parity)
5: Parity enable (PE) - if set, enables parity generation and checking.
6: Flow control (FL) - if set, enables flow control using CTS/RTS.
7: Loop back (LB) - if set, loop back mode will be enabled.
8: External Clock (EC) - if set, the UART scaler will be clocked by UARTI.EXTCLK

3.5.2 UART Status Register

31 … 7	6	5	4	3	2	1	0
RESERVED	FE	PE	OV	BR	TH	TS	DR

Figure 17. UART status register

0: Data ready (DR) - indicates that new data is available in the receiver holding register.
1: Transmitter shift register empty (TS) - indicates that the transmitter shift register is empty.
2: Transmitter hold register empty (TH) - indicates that the transmitter hold register is empty.
3: Break received (BR) - indicates that a BREAK has been received.
4: Overrun (OV) - indicates that one or more character have been lost due to overrun.
5: Parity error (PE) - indicates that a parity error was detected.
6: Framing error (FE) - indicates that a framing error was detected.

3.5.3 UART Scaler Register

31 … 12	11 … 0
RESERVED	SCALER RELOAD VALUE

Figure 18. UART scaler reload register

3.6 Signal description

The UART signals are described in table 11.

TABLE 11. UART signal description.

Signal name	Field	Type	Function	Active
RST	N/A	Input	Reset	Low
CLK	N/A	Input	Clock	-
APBI	*	Input	APB slave input signals	-
APBO	*	Output	APB slave output signals	-
UARTI	RXD	Input	UART receiver data	High
	CTSN	Input	UART clear-to-send	High
	EXTCLK	Input	Use as alternative UART clock	-
UARTO	RTSN	Output	UART request-to-send	High
	TXD	Output	UART transmit data	High

* see GRLIB IP Library User’s Manual

3.7 Library dependencies

Table 12 shows libraries that should be used when instantiating an APB UART.

TABLE 12. Library dependencies

Library	Package	Imported unit(s)	Description
GRLIB	AMBA	Types	APB signal definitions
GAISLER	UART	Types	Type declarations
ESA	MISC	Component	Component declaration

3.8 UART instantiation

This examples shows how the UART can be instantiated.

```
library ieee;
use ieee.std_logic_1164.all;

library grlib;
use grlib.amba.all;
library gaisler;
use gaisler.uart.all;
library esa;
use esa.misc.all;

entity apbuart_ex is
  port (
    clk : in std_ulogic;
    rstn : in std_ulogic;

    -- UART signals
    rxd   : in  std_ulogic;
    txd   : out std_ulogic
    );
end;

architecture rtl of apbuart_ex is
    -- APB signals
  signal apbi  : apb_slv_in_type;
  signal apbo  : apb_slv_out_vector := (others => apb_none);

  -- UART signals
  signal uarti : uart_in_type;
  signal uarto : uart_out_type;

begin

  uart0 : l2uart
  generic map (pindex => 1, paddr => 1,  pirq => 2, console => 1)
  port map (rstn, clk, apbi, apbo(1), uarti, uarto);

  -- UART input data
  uarti.rxd <= rxd;

  -- APB UART inputs not used in this configuration
  uarti.ctsn <= ’0’; uarti.extclk <= ’0’;

  -- connect APB UART output to entity output signal
  txd <= uarto.txd;
end;
```

4 PCI arbiter

4.1 Overview

PCIARB is n arbiter for the PCI bus, according to the PCI specification version 2.1. It is configurable for 4, 8, 16 or 32 agents, with 4 as default. The arbiter uses nested round-robbing policy in two priority levels. The priority assignment is either hard-coded or programmable through an APB interface.

4.2 Operation

4.2.1 Scheduling algorithm

The arbiter uses the algorithm described in the implementation note of section 3.4 of the PCI standard. The bus is granted by two nested round-robbing loops, where an agent number and a priority level is assigned to each agent. The agent number determines which pair of REQ/GNT lines are used. Agents are counted from 0 to NB_AGENTS-1. All agents in one level have equal access to the bus through a round-robbing policy. All agents of level 1 as a group have access equal to each agent of level 0. Re-arbitration occurs, when frame_n is asserted, as soon as any other master has requested the bus, but only once per transaction.

With programmable priorities, the priority level of all agents except NB_AGENTS-1 is programmable via APB. In a 256 byte APB address range, the priority level of agent N is accessed via the address 0x80 + 4*N. The APB read returns 0 on all non-implemented addresses, and the address bits (1:0) are not decoded. The constant ARB_LVL_C in pci_arb.vhd is the reset value.

4.2.2 Time-out

The “broken master” time-out is another reason for re-arbitration (section 3.4.1 of the PCI standard). Grant is removed from an agent, which has not started a cycle within 16 cycles after request (and grant). Reporting of such a ‘broken’ master is not implemented.

4.2.3 Turn-over

A turn-over cycle is required by the standard, when re-arbitration occurs during idle state of the bus. Notwithstanding to the standard, “idle state” is assumed, when FRAMEN is high for more than 1 cycle.

4.2.4 Bus parking

The bus is parked to agent 0 after reset, it remains granted to the last owner, if no other agent requests the bus. When another request is asserted, re-arbitration occurs after one turnover cycle.

4.2.5 Lock

Lock is defined as a resource lock by the PCI standard. The optional bus lock mentioned in the standard is not considered here and there are no special conditions to handle when LOCKN is active during in arbitration.

4.2.6 Latency

Latency control in PCI is via the latency counters of each agent. The arbiter does not perform any latency check and a once granted agent continues its transaction until its grant is removed AND its own latency counter has expired. Even though, a bus re-arbitration occurs during a transaction, the hand-over only becomes effective, when the current owner deasserts FRAMEN .

4.3 Configuration

The arbiter can be configured to NB_AGENTS = 4, 8, 16 or 32. A priority level (0 = high, 1 = low) is assigned to each device. Exception is agent NB_AGENTS-1, which has always lowest priority.

The priority levels are hard-coded, when APB_PRIOS = 0. In this case, the APB ports (APBI/APBO) are unconnected. The constant ARB_LVL_C must then be set to appropriate values.

When APB_PRIOS = 1, the levels are programmable via the APB-address 0x80. Bit 31 (leftmost) = master 31 . . bit 0 (rightmost) = master 0. Bit NB_AGENTS-1 is don't care at write and reads 1. Bits NB_AGENTS to 31, if existing, are dont care and read 0. The constant ARB_LVL_C is then the reset value.

The PCI arbiter has the following configuration options (VHDL generics):

TABLE 13. Memory controller configuration options (VHDL generics)

Generic	Function	Allowed range	Default
pindex	APB slave index	0 - NAPBSLV-1	0
paddr	ADDR filed of the APB BAR configuration registers address space.	0 - 16#FFF#	0
pmask	MASK filed of the APB BAR configuration registers address space.	0 - 16#FFF#	16#FFF#
nb_agents	Number of agents	4, 8, 16 or 32	4
apb_en	Enable programming through APB	0 - 1	1

4.4 Vendor and device id

The module has vendor id 0x04 (ESA) and device id 0x010. For description of venor and device ids see GRLIB IP Library User's Manual.

4.5 Signal description

Memory controller signals are described in table 14.

TABLE 14. Memory controller signal description.

Signal name	Type	Function	Active
CLK	Input	PCI Clock	-
RST_N	Input	PCI Reset	Low
REQ_N[0 to n-1]	Input	PCI Request signals	Low
FRAME_N	Input	PCI FRAME	Low
GNT_N[0 to n-1]	Output	PCI Grant signals	Low
PCLK	Input	APB Clock	-
PRST_N	Input	APB Reset	Low
APBI*	Input	APB slave input signals	-
APBO*	Output	APB slave output signals	-

* see GRLIB IP Library User's Manual

4.6 Library dependencies

Table shows libraries that the memory controller module depends on.

TABLE 15. Library dependencies

Library	**Package**	**Imported unit(s)**	**Description**
GRLIB	AMBA	Signals	AHB signal definitions