

AMBA Bus DMA Controller Specification

Version 1.00
Gareth Morris
28th March 2001

Table of Contents

Table of Contents	i
Index of Figures	ii
1 Introduction.....	1
2 Functional Overview.....	2
3 Architecture Overview (DMA Controller)	3
3.1 DMA Controller State Machine	4
3.2 Description of External Signals	5
3.2.1 Normal AMBA Signals.....	5
3.2.2 Arbitration AMBA Signals	5
3.2.3 Additional non-AMBA signals	5
3.3 Memory Map.....	6
3.4 Timing	7
4 Internal Architecture (Bus Arbiter).....	8
5 References	9
6 Glossary	10

Index of Figures

Figure 1: Simplified internal architecture of the single chip on board satellite computer.....	1
Figure 2: Proposed top level architecture of DMA controller.....	3
Figure 3: Diagram of proposed state machine for DMA controller.....	4
Figure 4: Basic timing diagram of DMA controller DMA controller.....	7

1 Introduction

This document describes the specification of a DMA (Direct Memory Access) controller for peripherals using an AMBA bus. The controller is to be implemented in VHDL as part of Surrey Satellite Technology's single chip on board computer project. This computer will be installed into small satellites in order to maximise the payload. The eventual aim is to put all the satellite systems onto a single chip.

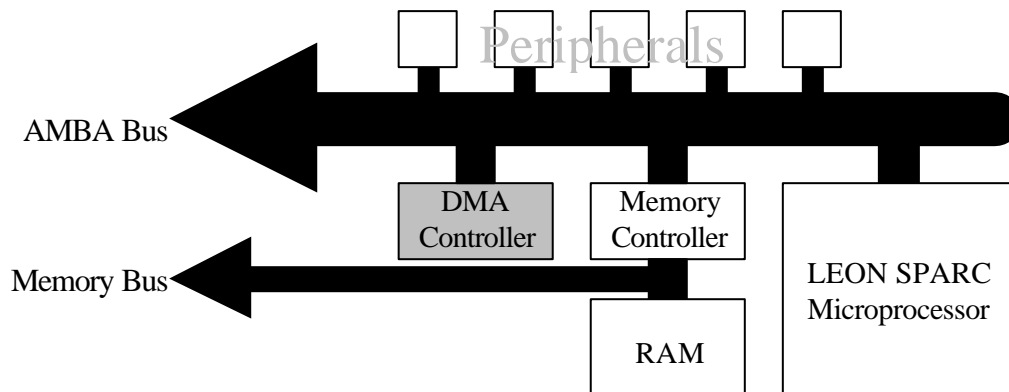


Figure 1: Simplified internal architecture of the single chip on board satellite computer

The diagram (based loosely on an image in Ref: 2) above shows an overview of the architecture of the single chip satellite, which resides on a Xilinx 'Virtex' FPGA. At the heart of the system is a LEON SPARC microprocessor. This is a public-domain microprocessor based on the SPARC V8 model developed in VHDL by the European Space Agency (see <http://www.gaisler.com> for more details).

This microprocessor is able to access peripherals (also on chip) via a special bus for embedded applications, called AMBA (Advanced Microprocessor Bus Architecture), which is specified by the custom microprocessor company ARM. To access memory LEON must use a special memory controller – also on the AMBA bus.

In order to function efficiently (for data capture, telemetry etc) it is necessary for the AMBA bus peripherals to directly write blocks of data to memory. This function is performed by a DMA controller, which will be created by this project. In common with all the other modules of this project the controller is to be written in VHDL and targeted to a Xilinx XCV800 FPGA device, for prototyping.

2 Functional Overview

The LEON microprocessor is a ‘master’ device on the AMBA bus – this means that it is able to initiate read and write transfers to all devices on the bus. The other peripherals (except the microprocessor) are ‘slave’ devices. This means that they are only able to respond to read or write requests.

The DMA controller is required to initiate read and write transfers to the Memory Controller (a slave device) and also other peripherals on the bus. For this reason the DMA controller is also required to be a ‘master’ device.

As there will be more than one AMBA bus master (DMA and CPU) in the system, the AMBA specification requires the presence of a ‘Bus Arbiter’. The bus arbiter selects which master is to have rights to the bus at any one instant in time. An arbiter already exists for the LEON processor, but does not take into account the DMA controller. Therefore a new arbiter will be also be implemented as a part of the project.

Initially the DMA controller is programmed by the LEON microprocessor with details of the next DMA transfer. This information is stored in write only registers internal to the DMA controller and includes the following information: base address for the transfer source, base address in memory to where the data is transferred, and the size of the transfer. This means that the DMA controller will also be a slave on the AMBA bus in order that the LEON can access these registers.

The DMA controller waits for the ‘AHB_Slv_DMAREQ’ line to be activated and takes over the AMBA bus once the microprocessor has relinquished it. It checks its internal registers to obtain details of the transfer. The DMA controller reads data from the source, stores it in an internal buffer, and then writes it out to the memory controller, word by word until the transfer is complete. The DMA controller then voluntarily releases usage of the AMBA bus and activates the ‘AHB_Slv_DMAACK’ line in order to indicate to the LEON processor that the transfer was completed.

If at anytime the LEON microprocessor requires usage of the AMBA bus it is able to take priority over the DMA controller. The DMA controller checks it has control of the bus at each stage, if it does not the controller waits for the bus to become available. Once LEON has finished using the bus the transfer continued from where it left off.

Finally an additional control between the LEON microprocessor and the DMA controller exists called DMA_DISABLE. When this is activated the controller is completely removed from the bus by using tri-state buffers. This is to ensure the microprocessor has complete control in all possible scenarios.

3 Architecture Overview (DMA Controller)

The internal architecture is very loosely based on the classic DMA controller – the Intel 8237 (Ref: 3), and is shown in the form of a diagram below:

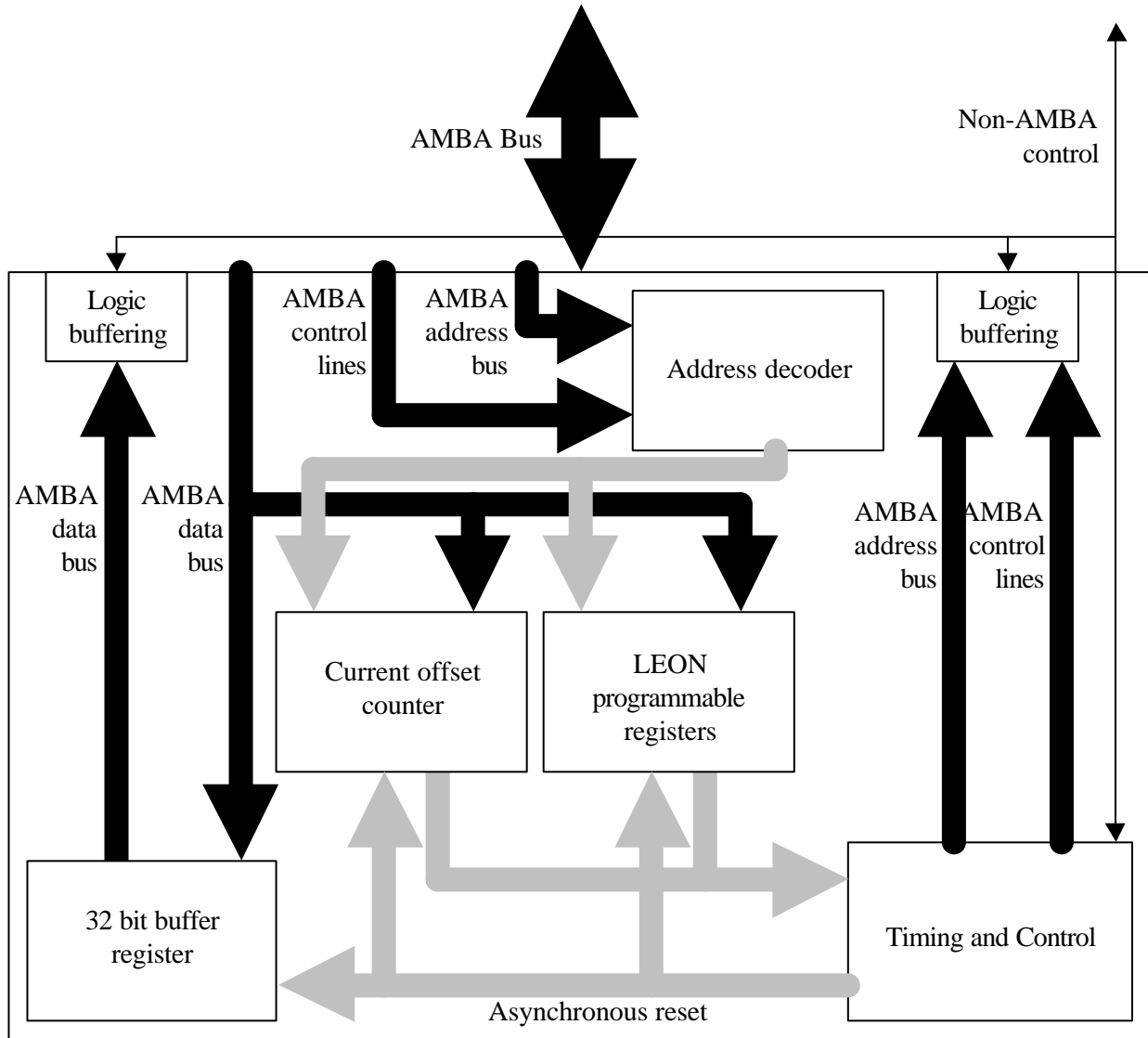


Figure 2: Proposed top level architecture of DMA controller

The architecture proposed for the DMA controller may be split into the above functional units. The logic buffering has been shown pictorially as being two separate modules, but in reality these are combined into 1. The most complex of these units is the timing and control block, which consists of a large state machine which is described in the diagram on the next page. The address decoder derives clock enable signals from the AMBA bus and operates in AMBA bus slave mode. Also present are a number of 32 bit registers which store details of the transfer in progress.

3.1 DMA Controller State Machine

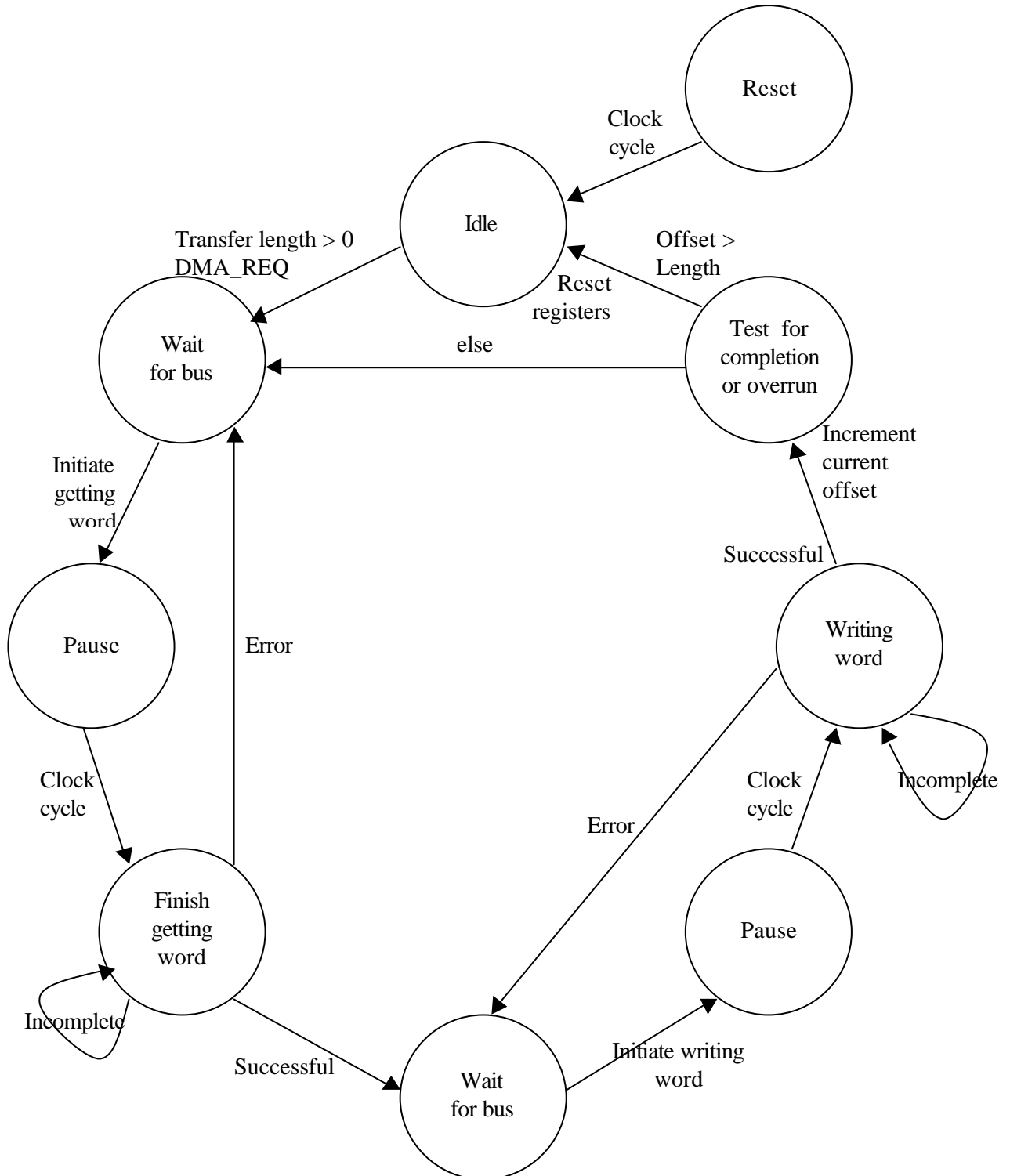


Figure 3: Diagram of proposed state machine for DMA controller

3.2 Description of External Signals

Active HIGH unless otherwise stated (Ref:1).

3.2.1 Normal AMBA Signals

HCLK	- This clock times all bus transfers	
HRESETn	- Active LOW. Resets the system and bus.	
HADDR[31:0]	- 32bit system address bus	
HTRANS[1:0]	- Transfer type	- See AMBA Spec. Non-sequential mode always used (10) in case bus control is lost
HWRITE	- Transfer direction. HIGH=write, LOW=read.	
HSIZE[2:0]	- Size of the transfer (byte, half word, word etc...)	- Set for 32bit operation(010)
HBURST[2:0]	- Burst type	- Set to 000, single transfer
HPROT[3:0]	- Protection control	- not used
HWDATA[31:0]	- Write to slave data bus	
HSELx	- Slave select (simply a decode of the address bus)	- not used
HRDATA[31:0]	- Read from slave data bus	
HREADY	- When HIGH indicates the bus is free	
HRESP[1:0]	- Transfer response (OK, error, retry etc)	

3.2.2 Arbitration AMBA Signals

The small x indicates a different signal for each master. This project will have LEON and DMA masters.

So called 'locked' transfers are not supported as the LEON microprocessor always has priority over bus usage, so the DMA may not lock the bus for its own use.

HBUSREQx	- Master bus request signal	
HLOCKx	- Locked transfer request signal	- not used
HGRANTx	- Bus granted signal	
HMASTER[3:0]	- Master currently performing a transfer	- 0=LEON 1=DMA
HMASTLOCK	- Indicates a locked transfer is being performed	- not used
HSPLITx[15:0]	- Split completion used by Split capable slaves	- not used

3.2.3 Additional non-AMBA signals

AHB_Slv_DMAACK	- Signal from DMA Controller indicating end of DMA transfer
AHB_Slv_DMAREQ	- Signal from LEON or Slave peripheral requesting a DMA transfer

3.3 Memory Map

The base address of the DMA controller's programmable registers has been defined as <to be decided>

Byte offset from base address	Register name	Details
+\$00	Inc_Addr	The base address of the incoming data, 32 bits wide
+\$04	Out_Addr	The base address of the outgoing data, 32 bits wide
+\$08	Trans_Len	Number of words of data to be transferred, 32 bits wide
+\$12 -> \$23	Reserved	Reserved for future expansion

3.4 Timing

Based on AMBA bus spec (Ref:1)

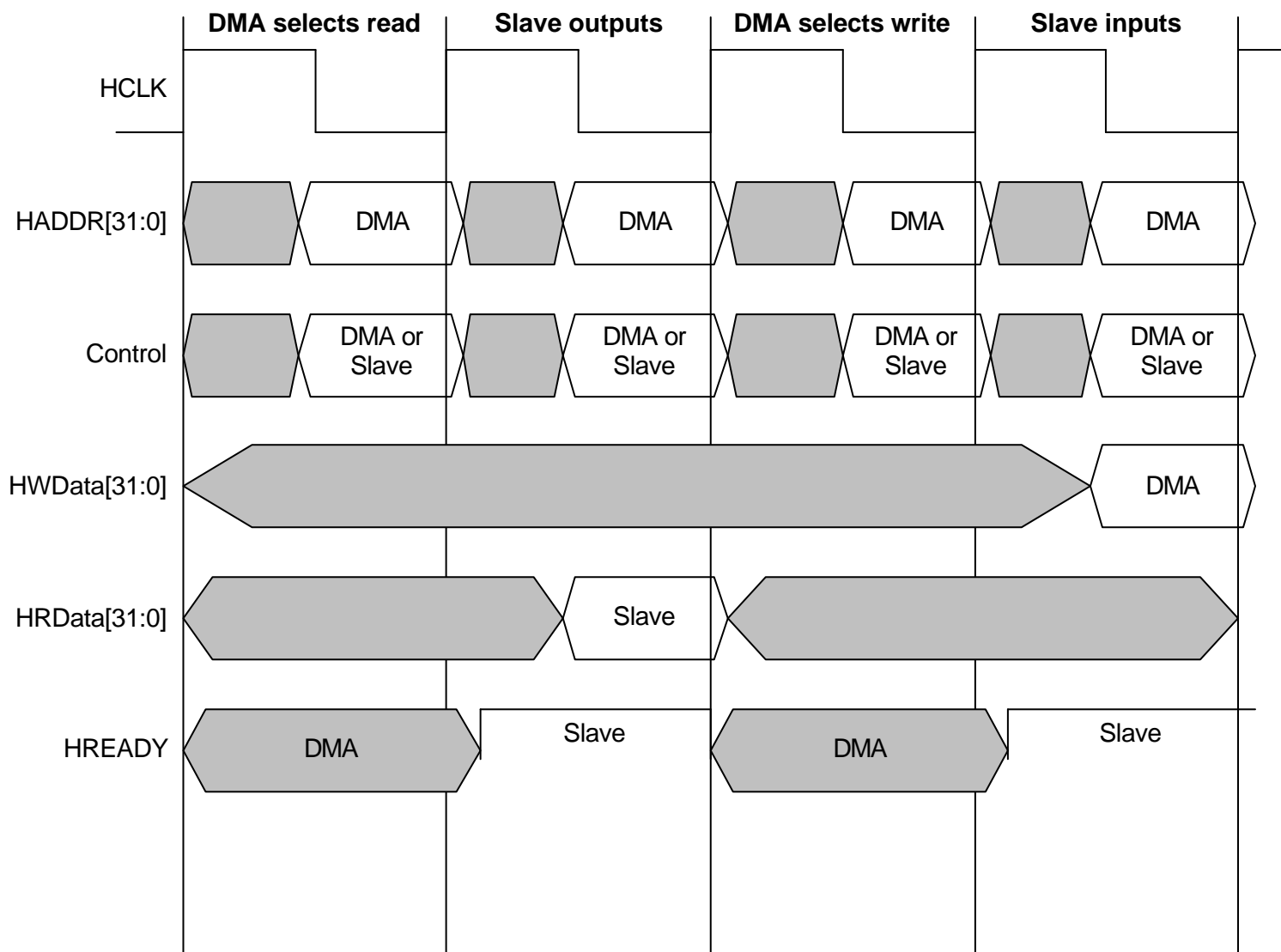


Figure 4: Basic timing diagram of DMA controller DMA controller

The control lines shown on this diagram comprise of the following signals:

- HTRANS** - Set by DMA controller
- HWRITE** - Set by DMA controller
- HSIZE** - Set by DMA controller
- HBRURST** - Set by DMA controller
- HRESP** - Set by Slave

If HREADY is not asserted to high by the slave device as shown on the diagram, but is pulled low, then the DMA controller will not change the state of the bus that cycle, and check the status of the signal on the next HCLK rising edge.

4 Internal Architecture (Bus Arbiter)

The bus arbiter sits on the AMBA bus and selects which master device has control of the bus. The LEON microprocessor already has an arbiter, which instantly grants the LEON microprocessor the bus. This will be removed and the new arbiter will take its place. The arbiter will operate the following arbitration scheme:

- The LEON microprocessor has control of the bus.
- If the DMA controller requests the bus usage is instantly granted to the it - if the LEON microprocessor's request line is de-asserted (the DMA controller only requests bus usage after it has been set-up by the microprocessor)
- If LEON then requests usage of the bus it is instantly granted.
- If the DMA controller de-asserts the AMBA bus request line control passes to the LEON microprocessor.
- If the LEON microprocessor de-asserts the AMBA bus request line it has no effect.

It should be possible to implement such a scheme using purely combinatorial logic.

5 References

- | | | |
|---|--------------|----------|
| 1. AMBA Bus Specification | Revision 2.0 | ARM Ltd. |
| 2. LEON-1 VHDL Model Description | Version 2.2a | ESA |
| 3. 8237a High Performance DMA Controller Data Sheet | Revision 005 | Intel |

6 Glossary

<i>AMBA</i>	- Advanced Microcontroller Bus Architecture. A system bus standard devised by the ARM corporation for embedded applications.
<i>ARM</i>	- Manufacturer of RISC processors for embedded applications.
<i>Bus Arbiter</i>	- Device for AMBA which controls bus access.
<i>DMA</i>	- Direct Memory Access. A method whereby peripherals can access main system memory without using processor resources.
<i>ESA</i>	- European Space Agency. Organisation which manages Europe's space activities
<i>FPGA</i>	- Field Programmable Gate Array. RAM based Semiconductor device which is programmable for different digital logic circuits
<i>LEON</i>	- Microprocessor adhering to the SPARC V8 standard written in VHDL by Jiri Gaisler of the ESA.
<i>SPARC V8</i>	- Scalable Processor ARChitecture version 8. A standard for RISC based microprocessors.
<i>VHDL</i>	- VHSIC Hardware Description Language. A software language for describing digital logic performance
<i>VHSIC</i>	- Very High Speed Integrated Circuit
<i>Virtex</i>	- State of the art FPGA
<i>Xilinx</i>	- Semiconductor Corporation manufacturing FPGAs