esa ESTEC

PCI Core/ AMBA Bus Interface

D/TOS-ESM/ELV/158

E. Lama Vaquero
November 2000

Preface

The subject of this project is the design of an interface between a PCI Core designed 'in-house' and an AMBA bus.

The PCI Core / AMBA Bus Interface has been designed as a VHDL model and has been simulated with Modelsim EE-SE 5.3c. No synthesis has been done at the moment.

The subjects covered in each chapter are outlined below:

Chapter 1 is an introduction to the project. It explains the motivation behind this work and describes the PCI Core and the AMBA bus, which are both connected to the system designed.

Chapter 2 gives an overview of the whole system designed and introduces the three main blocks composing it: the Configuration, the Master and the Target Interfaces.

In chapter 3 the Configuration Interface is studied. This chapter explains the requirements needed and the design chosen.

Chapter 4 deals with the Slave Interface. The requirements needed for the PCI Interface and for the AHB bus are outlined and the design is explained in detail. A section has been included to show all the restrictions found.

The same structure as in chapter 4 has been followed in chapter 5 with the Master Interface.

Chapter 6 concludes the report. Some modifications that may be required in future versions are outlined together with some problems related to the PCI Interface that are still to be solved.

Contents

Chapter 1: Introduction

The aim of this project is to develop an interface between two widely used commercial buses: the AMBA bus and the PCI bus.

Prior to the starting of this work, a PCI Interface had been developed in a previous project. This PCI Interface was designed by Riccardo Locatelli [1] as a connection between a PCI bus and a generic back-end interface and his design has been used so that the final system to be implemented would be simpler than designing the whole interface between the two buses.

The new system developed is the block between the AMBA bus and the PCI Interface shown in black double squares in figure 1.1, making it work as a back-end application for the PCI Interface.

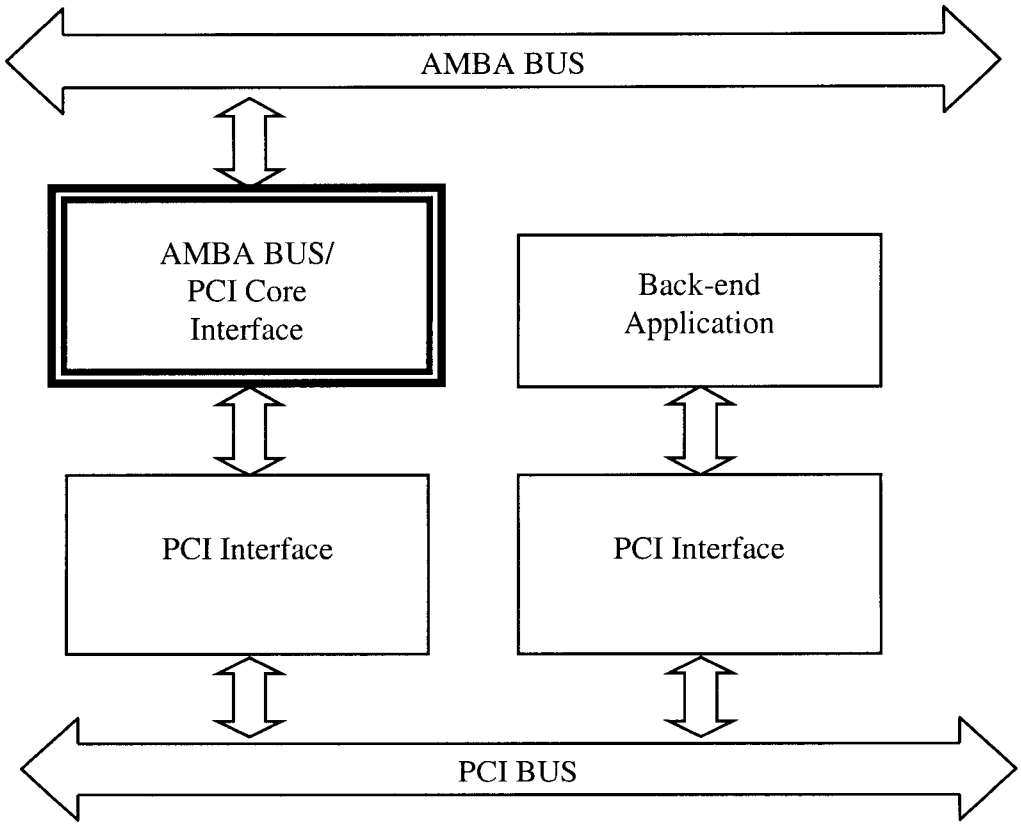

Figure 1.1. Interconnection of the AMBA bus / PCI Core Interface.

The PCI Interface was designed to connect a back-end application to the PCI bus. This functionality has been used to connect it to another interface and thus create a

nexus between the AMBA and PCI buses that would make the transfer of information between the two buses possible.

1.1. PCI Interface

The PCI Interface [1] was implemented by Riccardo Locatelli. It was designed to be a very simple interface for the back-end application in such a way that the exchange of information would be done on easier basis than having to handle with the whole PCI protocol.

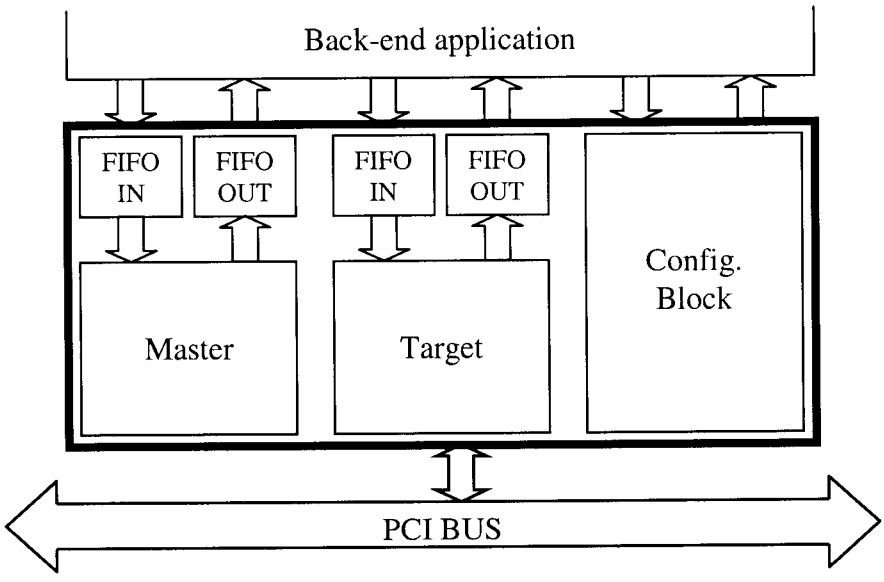

Figure 1.2. PCI Interface between a PCI Bus and a back-end interface.

Figure 1.2 shows the typical connection of the PCI Interface to a PCI bus on one side and to a back-end application on the other. The three blocks that compose the PCI Core are the Master, the Target and the Configuration block.

The back-end application can be a Master device, a Target device or both. That means separate data paths must be provided so that Master and Target parts of the same application can operate independently.

The Master block in the PCI Core is intended to interface with the Master side of the back-end application (if any), and the Target block in the PCI Core is designed to interface with the Target side of the back-end application (again, if any).

For the Master and Target blocks there are input/output synchronisation FIFO's that also serve as data buffering between both systems. The exchange of information is done through these FIFO's in the form of 40-bit words.

The functions of the three blocks composing the PCI Core are:

- The Master block receives/send the data from/to the Master side of the back-end application, and is the one initiating the transaction on the PCI bus.

- The Target block receives/sends the data from/to the Target side of the back-end application, and is the one responding to the transfer initiated by a master application.
- The configuration block contains the configuration space registers and handles the read and write cycles to them. The configuration cycles can be performed from both sides: direct configuration via the PCI bus or local configuration through back-end.

1.2. AMBA buses

The Advanced Microcontroller Bus Architecture (AMBA) specification [2] defines an on-chip communications standard for designing high-performance embedded microcontrollers.

Three distinct buses are defined within the AMBA specification:

- **Advanced High-performance Bus (AHB)**: The AHB acts as the high-performance system backbone bus. AHB supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macrocell functions. AHB is also specified to ensure easy of use in an efficient design flow using synthesis and automated test techniques.

- **Advanced Peripheral Bus (APB)**: AMBA APB is optimised for minimal power consumption and reduced interface complexity to support peripheral functions.

- **Advanced System Bus (ASB)**: AMBA ASB is an alternative system bus suitable for use where the high-performance features of AHB are not required. ASB also supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macrocell functions.

Only the two first ones, AHB and APB, will be used in this design. The reason is that the next version of LEON [3] will use the AMBA bus to transfer data between the cache controllers, peripheral functions and memory controller: it will use the APB bus to access the PCI configuration registers from the CPU, and the AHB bus to transfer data between the PCI core and the CPU's memory controller. Using the AMBA on-chip bus will simplify attachment of future cores by offering a well-defined interface.

Chapter 2: Design overview

Figure 2.1 shows the system designed. It consists of three independent modules: the Slave Interface, the Master Interface and the Configuration Interface.

It is connected on one side to two of the AMBA Buses: The APB is used to access the PCI configuration registers from the CPU and the AHB to transfer data between the PCI core and the CPU's memory controller.

And on the other side it connects to the PCI Interface. The Configuration Interface is directly connected to the Configuration Block in the PCI Interface, the Master Interface is connected through the synchronisation FIFOs to the Target Block in the PCI Interface, and the Slave Interface is connected, also through synchronisation FIFOs, to the Master Block in the PCI Interface.

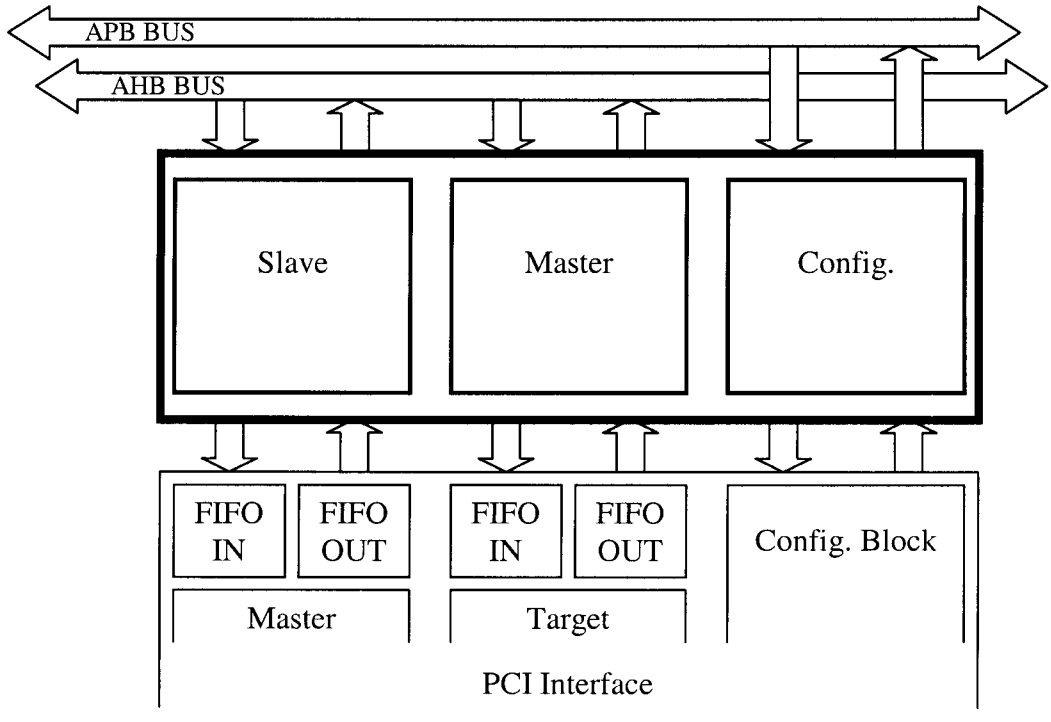

Figure 2.1. Connection of the PCI Core / AMBA Bus Interface.

The naming convention is that the names are given according to the bus they are related to. So in a transaction started in the AMBA bus, the slave interface would receive the data and send them to the Master Block in the PCI Interface that would then start the transaction in the PCI bus (see figure 2.2).

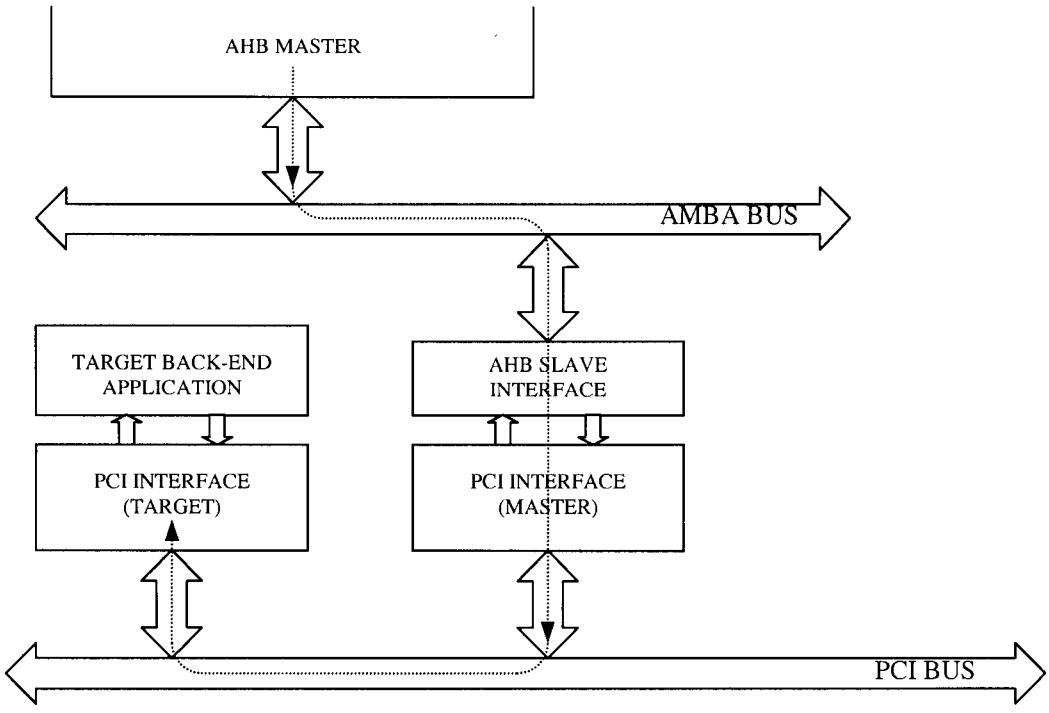

Figure 2.2. Transfer from an AMBA Master to a PCI Target.

In the same way, when a transaction is started in the PCI bus, the Target block in the PCI Interface would receive the data that would transmit to the Master interface. The Master Interface would then start the transaction in the AMBA bus (see figure 2.3).

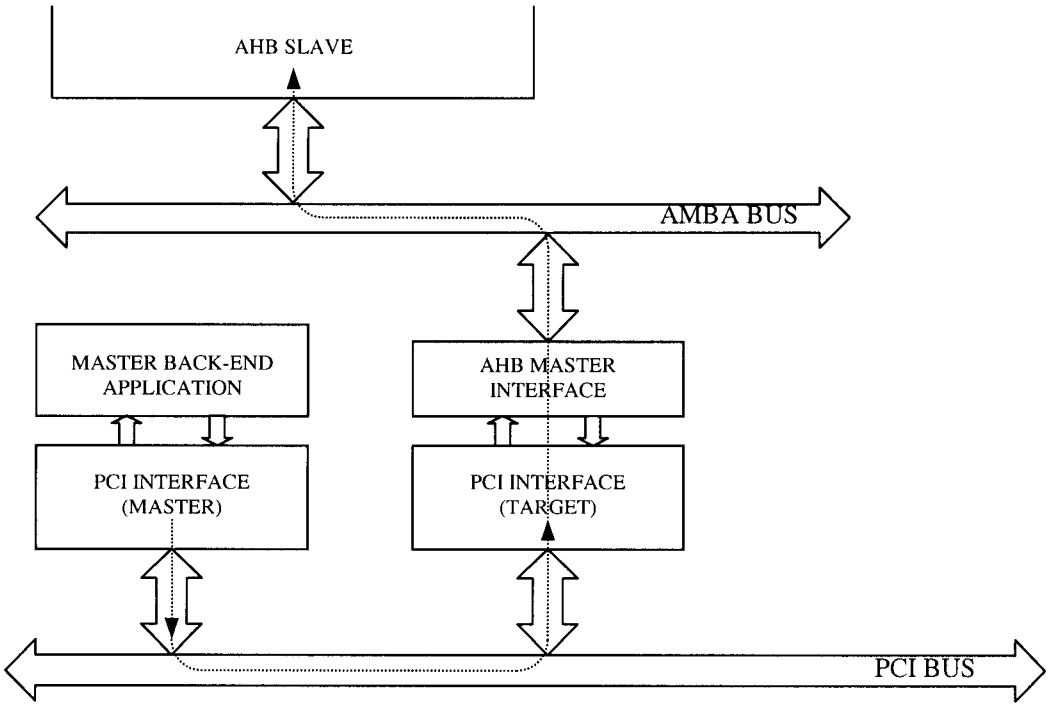

Figure 2.3. Transfer from a PCI Master to an AMBA Slave.

Chapter 3: Configuration Interface

The PCI Configuration Registers in the Configuration Space of the PCI Core can be accessed through the PCI Bus in a direct access or through the Local Configuration Interface in a back-end access [1][4].

The block designed, named Configuration Interface, is shown in figure 3.1. It is an APB Slave block that connects the APB Bus to the Configuration Block in the PCI Core.

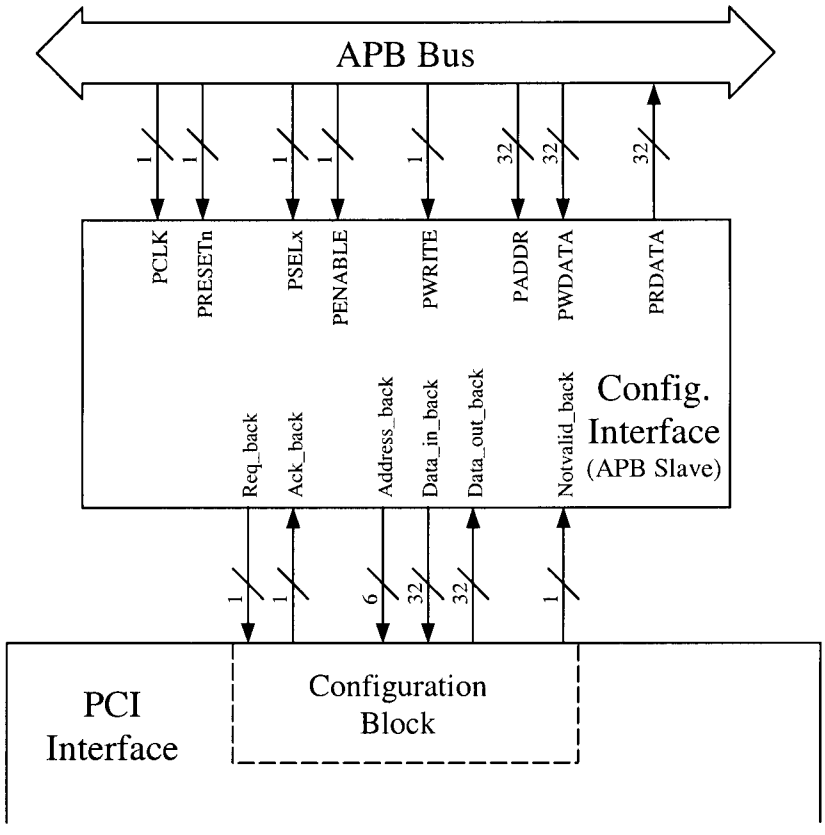

Figure 3.1. Connection of the Configuration Interface Module.

3.1. PCI Interface requirements.

As can be observed in figure 3.1, the address bus in the PCI Interface has a width of 6 bits. This is because the configuration Space in the PCI Core consists of 256 bytes

divided into 32-bit lines. 8 bits are needed to address it, but given that the accesses are done in groups of 4 bytes, the two lowest bits are discarded giving an amount of 6 bits to address the whole configuration space.

The timing requirements for back-end configuration accesses are those shown in figures 3.2 and 3.3.

On a write cycle, the Address_back and Data_in_back signals should contain the right address and data when the acknowledge signal (Ack_back) is asserted. It is recommended to set the data when requesting the bus (Req_back). The writing of the registers occurs when both Req_back and Ack_back are asserted.

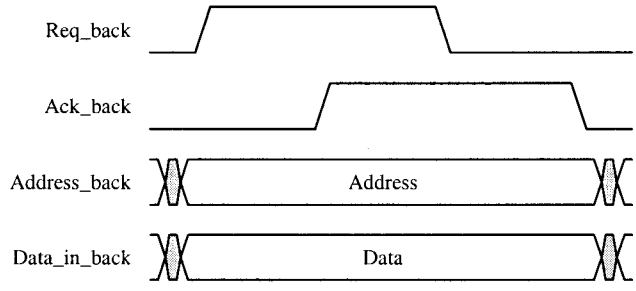

Figure 3.2. Back-end Write Access to the Configuration Space in the PCI Core.

On a read cycle, back-end has to drive the Address_back bus with the right address value and the corresponding data can be read on the Data_out_back. When the signal Notvalid_back is asserted, it means that a direct PCI access is occurring and data read are not valid.

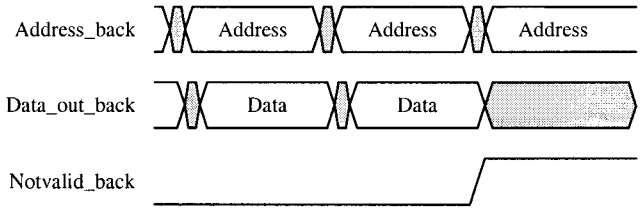

Figure 3.3. Back-end Read Access to the Configuration Space in the PCI Core.

3.2. APB requirements.

The timing requirements for APB accesses are those shown in figures 3.4 and 3.5. Both read and write cycles take two clock cycles to complete.

For a write transfer (figure 3.4) the data can be latched at the following points:
- on either rising edge of PCLK, when PSEL is HIGH.
- On the rising edge of PENABLE, when PSEL is HIGH

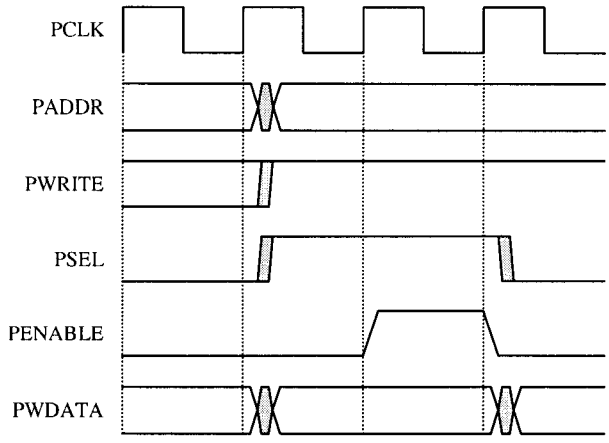

Figure 3.4. APB Write Transfer.

For read transfers (figure 3.5) the data can be driven on to the data bus when PWRITE is LOW and both PSELx and PENABLE are HIGH. PADDR is used to determine which register should be read.

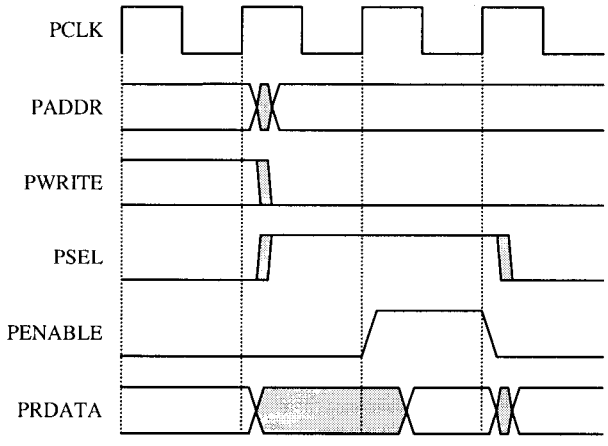

Figure 3.5. APB Read transfer.

3.3. Design.

Taking into account the PCI Interface and APB specifications, the design implemented is that shown in figure 3.6.

On a write cycle, the bus request (Req_back), address (Address_back) and data (Data_in_back) are latched into the PCI Interface. This design would work only if the acknowledgement is immediate, which will happen if no cycle is being attempted from the PCI Bus. If configuration can happen simultaneously from APB Bus and from PCI Bus, higher priority is given to direct (PCI) access by the PCI Interface. To check whether the write cycle has taken place correctly, a read cycle can follow.

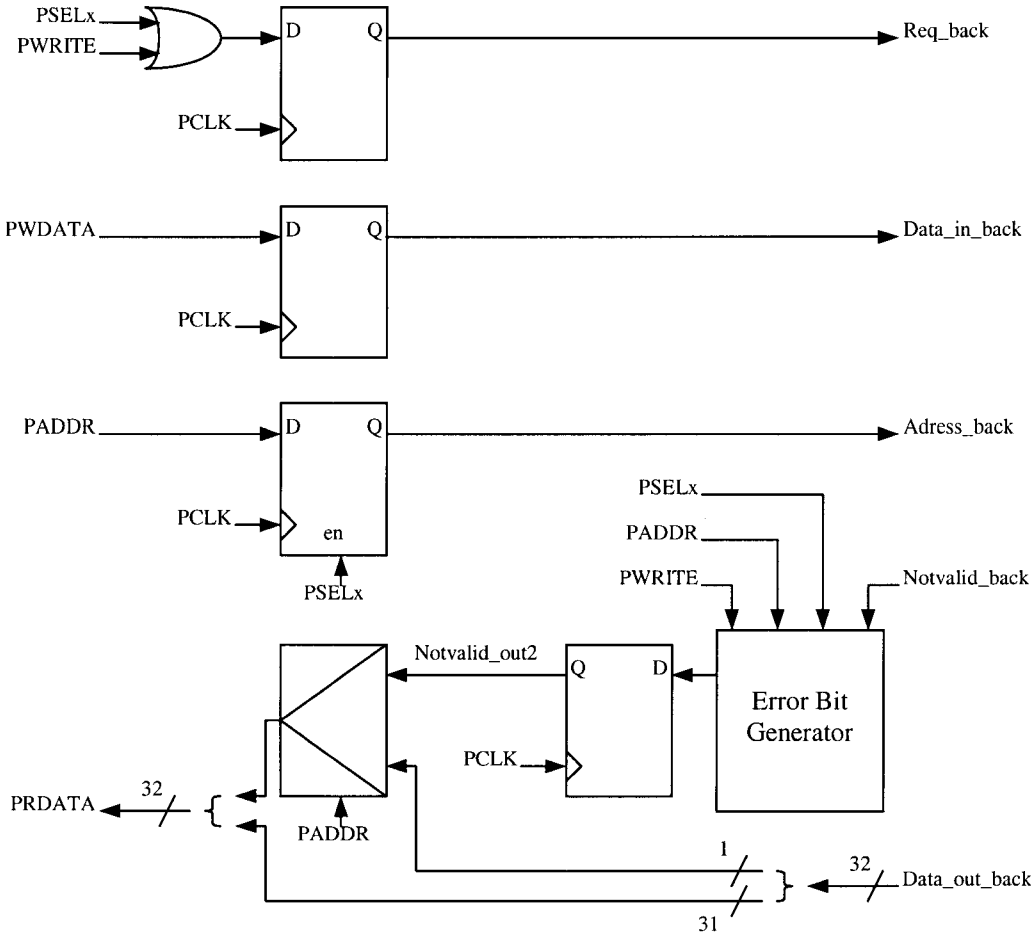

Figure 3.6. Structure of the Configuration Interface.

On a read cycle the output data can be read shortly after the address has been driven into the PCI Interface. The Notvalid_back is used to indicate that the result of a read cycle is not correct. This information cannot be passed to the APB as there is no error line in this bus, so its value is stored in a register. To read this error bit, the eighth line in the address bus should be set to high. If there has been a mistake since the last time the error bit was read, the output value would be '1'.

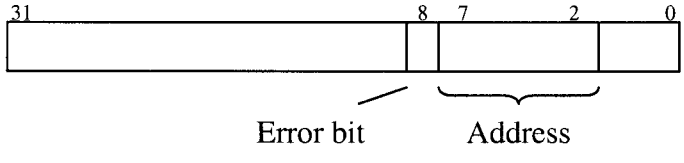

Figure 3.7. Structure of the 32-bit address bus sent from the APB.

Chapter 4: Slave Interface

The Slave Interface connects the AHB bus and the Master's FIFO in the PCI Core. Seen from the AHB bus it works as an AHB slave module and seen from the PCI interface it works as a Master Back-end application.

When a Master module in the AHB bus initiates a transaction, the information reaches the AHB Slave (Slave interface in figure 4.1.) which will transform it to the PCI Interface standards in order to initiate the transaction in the PCI bus as a bus master.

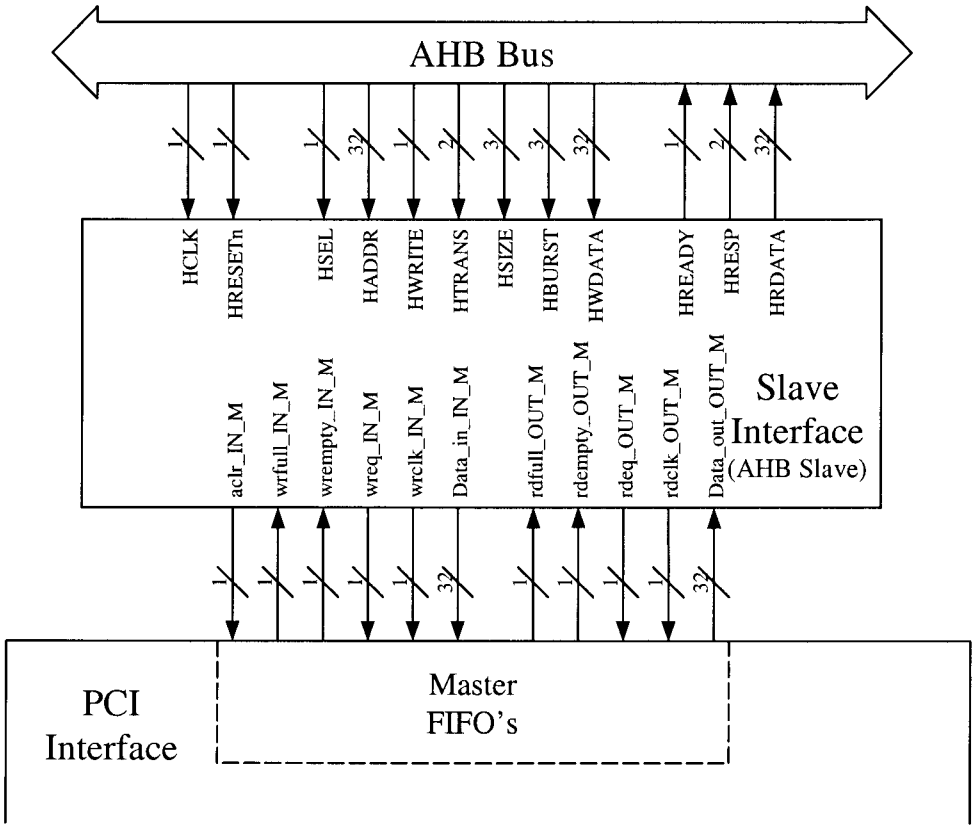

Figure 4.1. Connection of the Slave Interface Module.

4.1. PCI Interface requirements.

The information should be sent into the FIFO's is in the format of 40-bit words. In these words, the address, data and control information should be organised in a

determined way depending on the transfer cycle. The FIFO's store the information in the Data_in lines on the positive edge of the clock (rdclk). The examples shown in this section do not cover all the possible cases. For a complete list of bit combinations refer to documents [1] and [4].

4.1.1. Single write cycle

On a single write cycle the first word should contain the command, the address and the transfer type, and the second word the byte enable lines, the data and the transfer type. The PCI interface should then give the transfer result through the Data_out lines.

An example of a single write cycle is shown below:

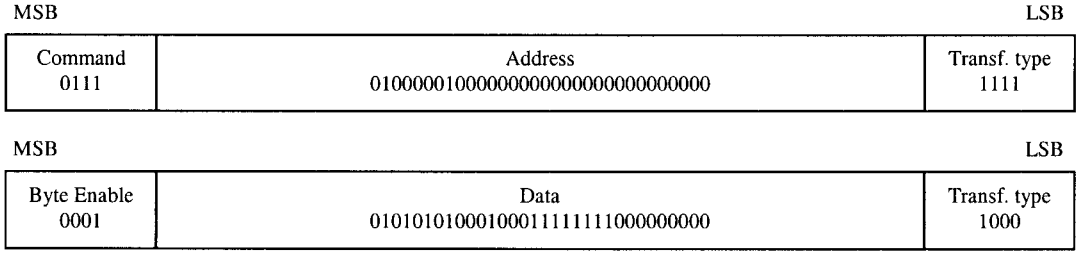

Figure 4.2. Words sent from the Slave interface to the Master's FIFO's.

The first word sent to the master's FIFO's contains the command "0111" that corresponds to a Memory Write cycle followed by the address and by the transfer type "1111" that corresponds to a Single access.

The second word contains the Byte Enable lines (inverted logic) followed by the data to be written and by the transfer type "1000" that indicates the last data condition.

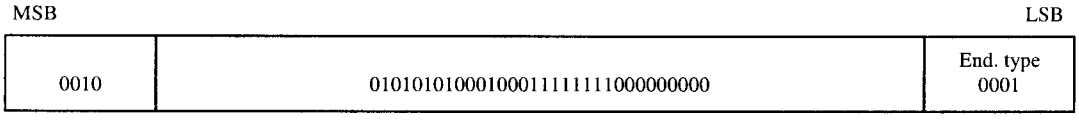

Figure 4.3. Word sent from the Master's FIFO's to the Slave interface.

The master's FIFO's would then send the status word that contains on the four most significant bits the Data Bit, Lock Info, First Data and Parity Error lines and on the four least significant bits the end type "0001" that corresponds to End Completion.

4.1.2. Burst write cycle

On a burst write cycle the first word should contain the command, the address and the transfer type, and the folowing words the byte enable lines, the data and the transfer type. The PCI interface should then give the transfer result through the Data_out lines.

An example of a burst write cycle is shown below:

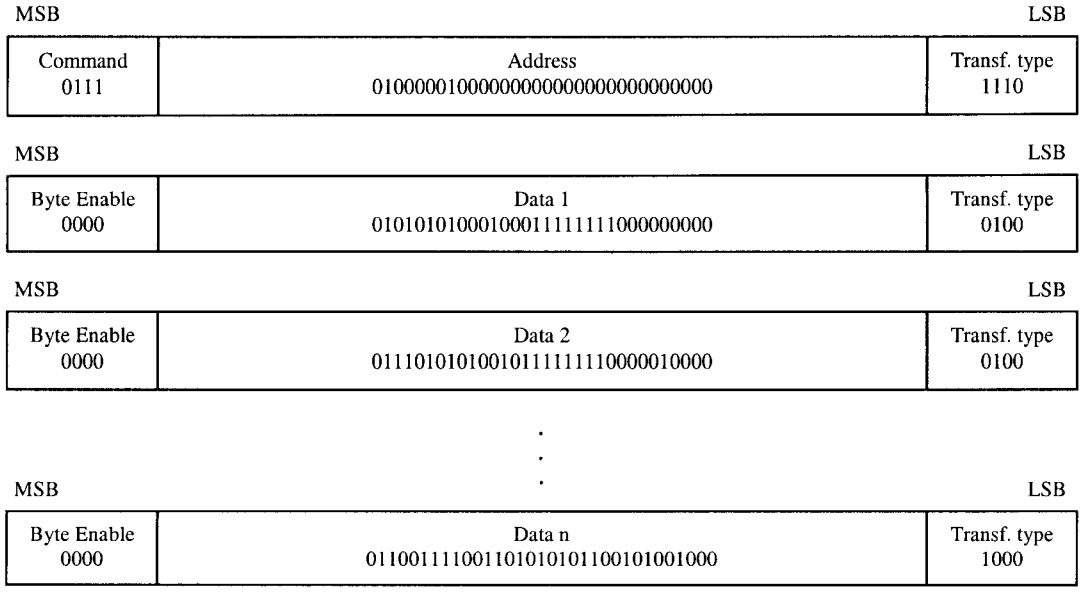

Figure 4.4. Words sent from the Slave interface to the Master's FIFO's.

The first word sent to the master's FIFO's contains the command "0111" that corresponds to a Memory Write cycle followed by the address and by the transfer type "1110" that corresponds to a Burst access.

The following words contains the Byte Enable lines (inverted logic) followed by the data to be written and by the transfer type "0100" that indicates the data condition. On the last word, the transfer type is changed to "1000" that indicates the 'last data' condition.

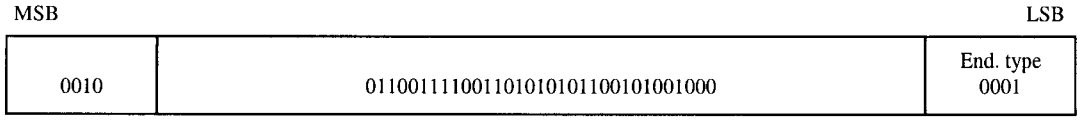

Figure 4.5. Word sent from the Master's FIFO's to the Slave interface.

The master's FIFO's would then send the status word that contains on the four most significant bits the Data Bit, Lock Info, First Data and Parity Error lines and on the four least significant bits the end type "0001" that corresponds to End Completion, as in the Single Write Cycle.

4.1.3. Single prefetchable read cycle

On a single read cycle to a prefetchable address the only word sent to the master's FIFO's should contain the command, the address and the transfer type. The PCI interface should then give the data together with the transfer result through the Data_out lines.

An example of a single read cycle to a prefetchable address is shown below:

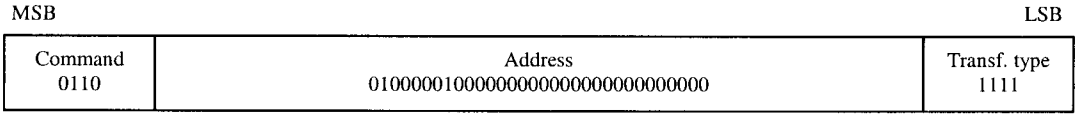

Figure 4.6. Word sent from the Slave interface to the Master's FIFO's.

The word sent to the master's FIFO's contains the command "0110" that corresponds to a Memory Read cycle followed by the address to be read and by the transfer type "1111" that corresponds to a Single access.

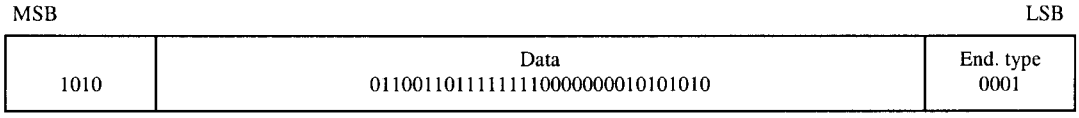

Figure 4.7. Word sent from the Master's FIFO's to the Slave interface.

The master's FIFO's would then send the status word that contains: on the four most significant bits the Data Bit, Lock Info, First Data and Parity Error lines; on the 32 mid-bits the data read; and on the four least significant bits the end type "0001" that corresponds to End Completion.

4.1.4. Single not-prefetchable read cycle

On a single read cycle to a not-prefetchable address the first word should contain the command, the address and the transfer type, and the second word the byte enable lines and the transfer type. The PCI interface should then give the data together with the transfer result through the Data_out lines.

An example of a single read cycle to a not-prefetchable address is shown below:

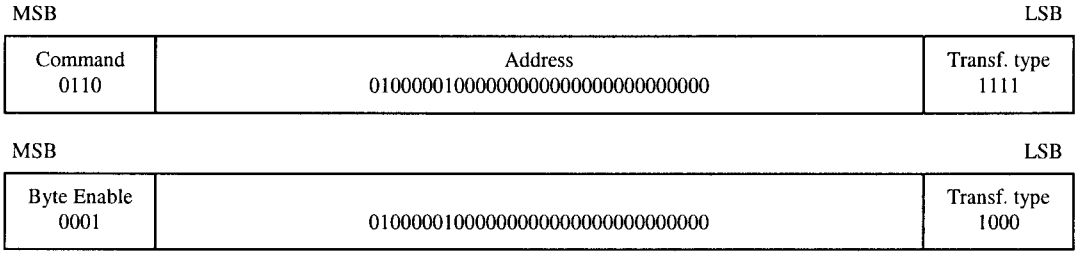

Figure 4.8. Words sent from the Slave interface to the Master's FIFO's.

The first word sent to the master's FIFO's contains the command "0110" that corresponds to a Memory Read cycle followed by the address and by the transfer type "1111" that corresponds to a Single access.

The following words contains the Byte Enable lines (inverted logic) in the four most significant bits and in the least significant bits the transfer type "0100" that indicates the last data condition.

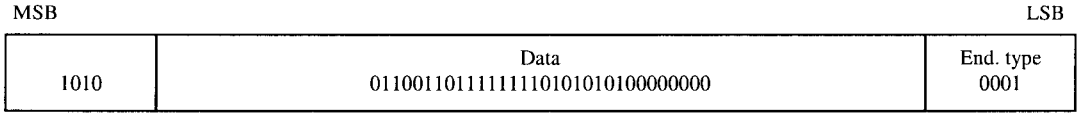

Figure 4.9. Word sent from the Master's FIFO's to the Slave interface.

The master's FIFO's would then send the status word that contains: on the four most significant bits the Data Bit, Lock Info, First Data and Parity Error lines; on the 32 mid-bits the data read; and on the four least significant bits the end type "0001" that corresponds to End Completion.

4.1.5. Burst prefetchable read cycle

On a burst read cycle to a prefetchable address the word sent to the master's FIFO's should contain the command, the address and the transfer type. The PCI interface should then start giving data starting from the address provided. At any time the master sends the stop request to finish the transfer and the slave sends the last data together with the transfer result.

An example of a burst read cycle to a prefetchable address is shown below:

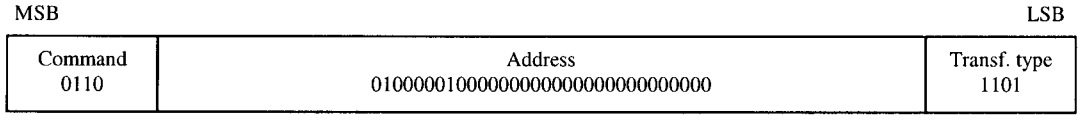

Figure 4.10. Word sent from the Slave interface to the Master's FIFO's.

The word sent to the master's FIFO's contains the command "0110" that corresponds to a Memory Read cycle followed by the address to be read and by the transfer type "1101" that corresponds to a Burst Prefetchable access.

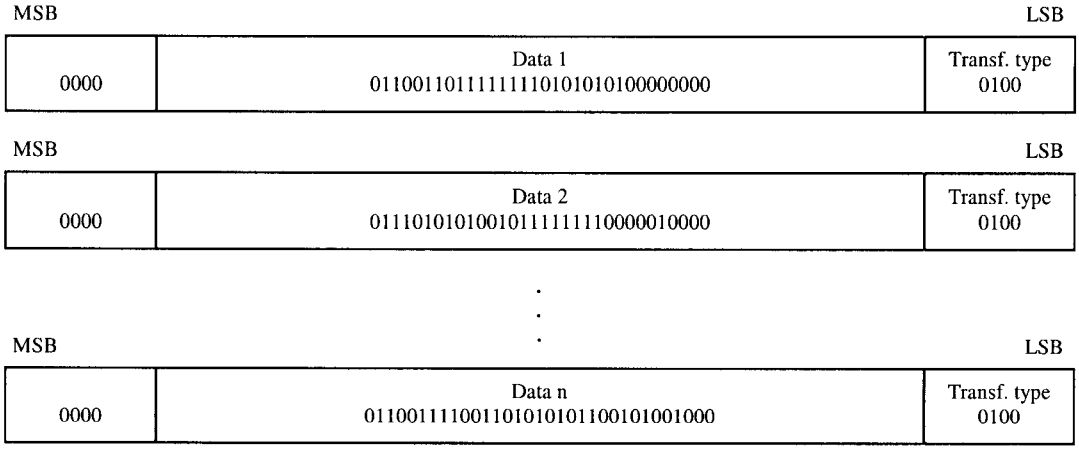

Figure 4.11. Words sent from the Master's FIFO's to the Slave interface.

The words sent from the Master's FIFO's contains the data in the mid 32 bits and the transfer type "0100" that indicates the data condition.

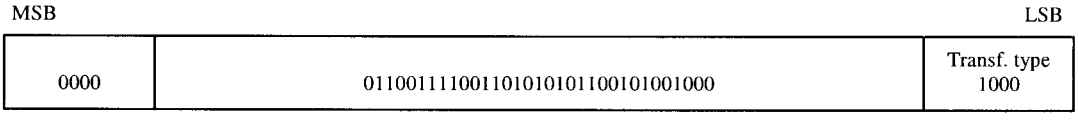

Figure 4.12. Word sent from the Slave interface to the Master's FIFO's.

The data will continue to be sent by the FIFO until a Stop signal is received. To stop the burst read access the Slave Interface sends in the least significant bits the Transfer type "1000" that indicates the last data condition.

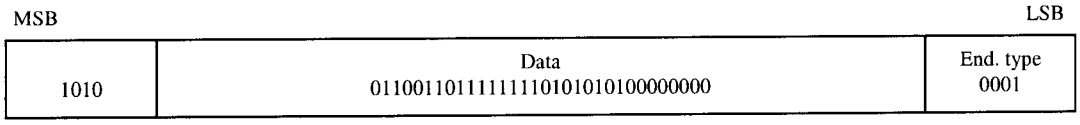

Figure 4.13. Word sent from the Master's FIFO's to the Slave interface.

The master's FIFO's would then send the status word that contains: on the four most significant bits the Data Bit, Lock Info, First Data and Parity Error lines; on the 32 mid-bits the data read; and on the four least significant bits the end type "0001" that corresponds to End Completion.

4.1.6. Burst not-prefetchable read cycle

On a burst read cycle to a not-prefetchable address the first word should contain the command, the address and the transfer type. The following words will be a handshake between Master and Target: the Slave Interface sends the Byte enable lines and receives the data on the corresponding lines. On the last transfer the PCI interface gives the data together with the transfer result.

An example of a burst read cycle to a not-prefetchable address is shown below:

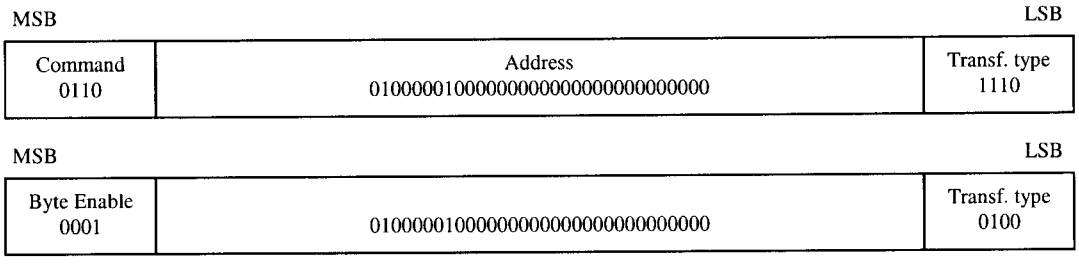

Figure 4.14. Words sent from the Slave interface to the Master's FIFO's.

The first word sent to the master's FIFO's contains the command "0110" that corresponds to a Memory Read cycle followed by the address and by the transfer type "1110" that corresponds to a Burst access.

The following word contains the Byte Enable lines (inverted logic) in the four most significant bits and in the least significant bits the transfer type "0100" that indicates the data condition.

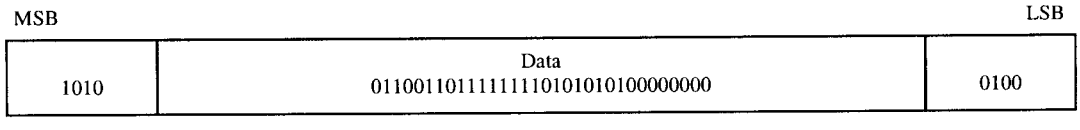

Figure 4.15. Word sent from the Master's FIFO's to the Slave interface.

The Master's FIFO's then sends the data in the appropriate mid lines and on the four most significant bits the Data Bit, Lock Info, First Data and Parity Error lines.

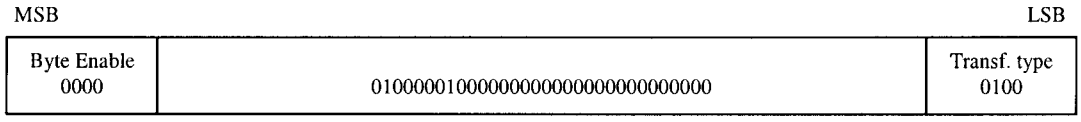

Figure 4.16. Words sent from the Slave interface to the Master's FIFO's.

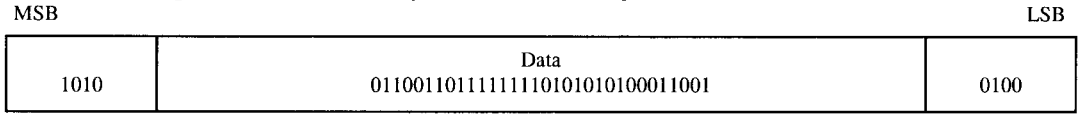

Figure 4.17. Word sent from the Master's FIFO's to the Slave interface.

.
.
.

This exchange of information will continue until the Last Data condition is sent to the Master's FIFO's (Transf. type 1000).

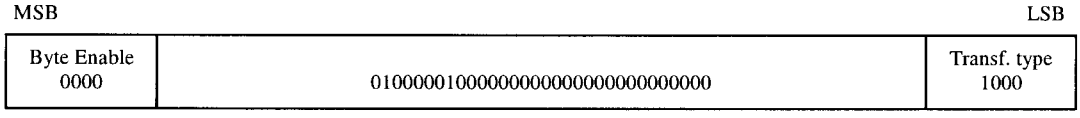

Figure 4.18. Words sent from the Slave interface to the Master's FIFO's.

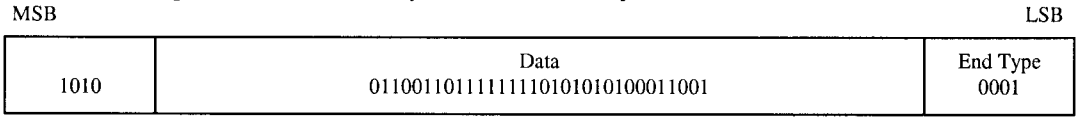

Figure 4.19. Word sent from the Master's FIFO's to the Slave interface.

The master's FIFO's would then send the status word that contains: on the four most significant bits the Data Bit, Lock Info, First Data and Parity Error lines; on the 32 mid-bits the data read; and on the four least significant bits the end type "0001" that corresponds to End Completion.

4.2. AHB requirements.

The AHB Slave Interface is connected to the AHB Bus through the following lines [2]:

HCLK, HRESET System clock and reset.

HSEL This is the selection signal. A slave must only sample the address and control signals when HSEL and HREADY are HIGH, indicating that the current transfer is completing.

HWRITE This line indicates that the current transfer is a write cycle when HIGH or a read cycle when LOW.

HSIZE This signal indicates the size of the transfer, as shown in the following table:

HSIZE	Size	Description
000	8 bits	Byte
001	16 bits	Half word
010	32 bits	Word
011	64 bits	
100	128 bits	4-word line
101	256 bits	8-word line
110	512 bits	
111	1024 bits	

HPROT This is the protection control signal. Not all masters will be capable of generating accurate protection information, therefore it is recommended that slaves do not use the HPROT signals unless strictly necessary.

HTRANS This signal indicates the cycle type. There are four possible values:

HTRANS	Cycle	Description
00	IDLE	Indicates that no data transfer is required.
01	BUSY	Allows the bus master to insert IDLE cycles in the middle of burst of transfers.
10	NONSEQ	Indicates the first transfer on a burst or a single transfer.
11	SEQ	The remaining transfers in a burst are 'sequential' and the address is related to the previous transfer.

HBURST This signal indicates the transfer type. The possible values of this signal are summarised in the following table:

HBURST	Transfer	Description
000	SINGLE	Single transfer
001	INCR	Incrementing burst of unspecified length
010	WRAP4	4-beat wrapping burst
011	INCR4	4-beat incrementing burst
100	WRAP8	8-beat wrapping burst
101	INCR8	8-beat incrementing burst
110	WRAP16	16-beat wrapping burst
111	INCR16	16-beat incrementing burst

HADDR This is the bus where the address lines for every transfer are driven.

HWDATA This is the where the bus master will write the data on write transfers and hold it stable throughout the whole cycle.

HRDATA This is the read data bus driven by the appropriate slave during read transfers.

HREADY This signal is used to extend the data phase. When LOW this signal causes wait states to be inserted into the transfer and allows extra time for the slave to provide or sample data.

HRESP This is the response signal used to shows the status of the transfer. It can have four possible values:

HRESP	Response	Description
00	OKAY	The OKAY response is used to indicate that the transfer is progressing normally. When HREADY goes HIGH this indicates the transfer has completed successfully.
01	ERROR	The ERROR response indicates that a transfer error has occurred and the transfer has been unsuccessful.
10	RETRY	Both the RETRY and SPLIT transfer responses indicate that the transfer cannot complete immediately, but the bus master should continue to attempt the transfer.
11	SPLIT	

Figure 4.20 shows the timing of the above signals in a basic transfer with a wait state cycle which extends the transfer allowing additional time for completion.

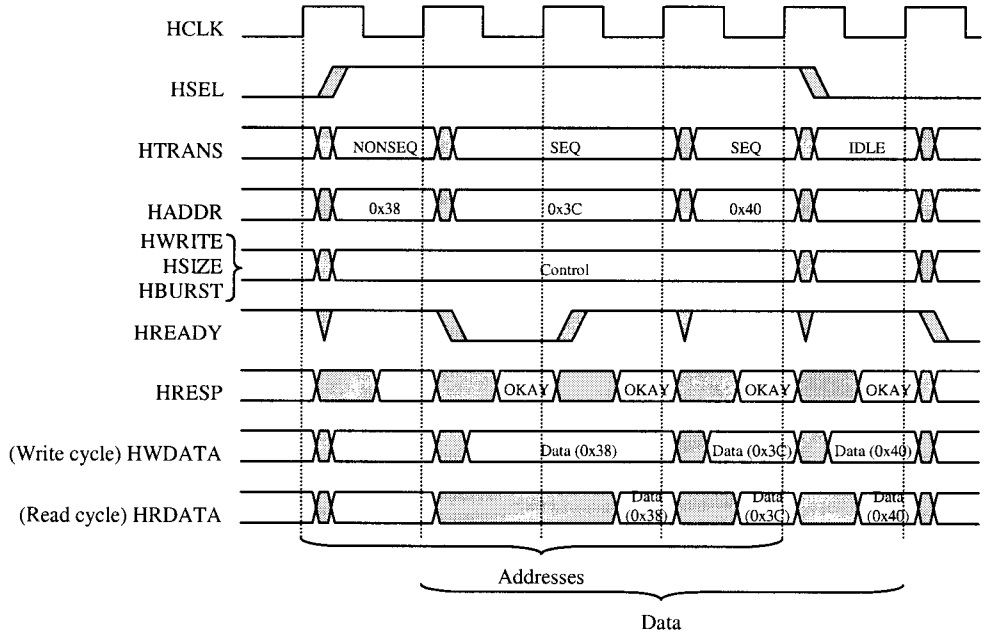

Figure 4.20. Signals sent to and received from the Slave on a basic AHB transfer.

The transfer will start when the HSEL signal indicates that the slave has been selected and the HREADY signal is set to HIGH. The address, the cycle type and control signals will be sampled on the positive edge of the following clock. If this was a write cycle the AHB master would provide the data to be written on the following clock unless the HREADY signal is set to zero which serves the AHB Slave to insert wait cycles. If, on the contrary, this was a read cycle the AHB slave would have to provide the data before the rising edge of the following clock, unless the HREADY signal is set to zero.

On the same clock edge where the data is written or read, the AHB Slave drives the HRESP signal to report the status of the transfer. The transfer is finished when the slave is not selected any longer or when the HTRANS signal indicates a IDLE state or another NONSEQ status meaning a new transfer is starting.

When the answer to a transfer is not an OKAY response, a two-cycle response is required [2], one of them while HREADY is set to LOW and the second one while it is set to HIGH.

4.3. Design.

Figure 4.21 shows the schematics of the Slave Interface module. Underlined are the input and output lines, the rest being internal signals.

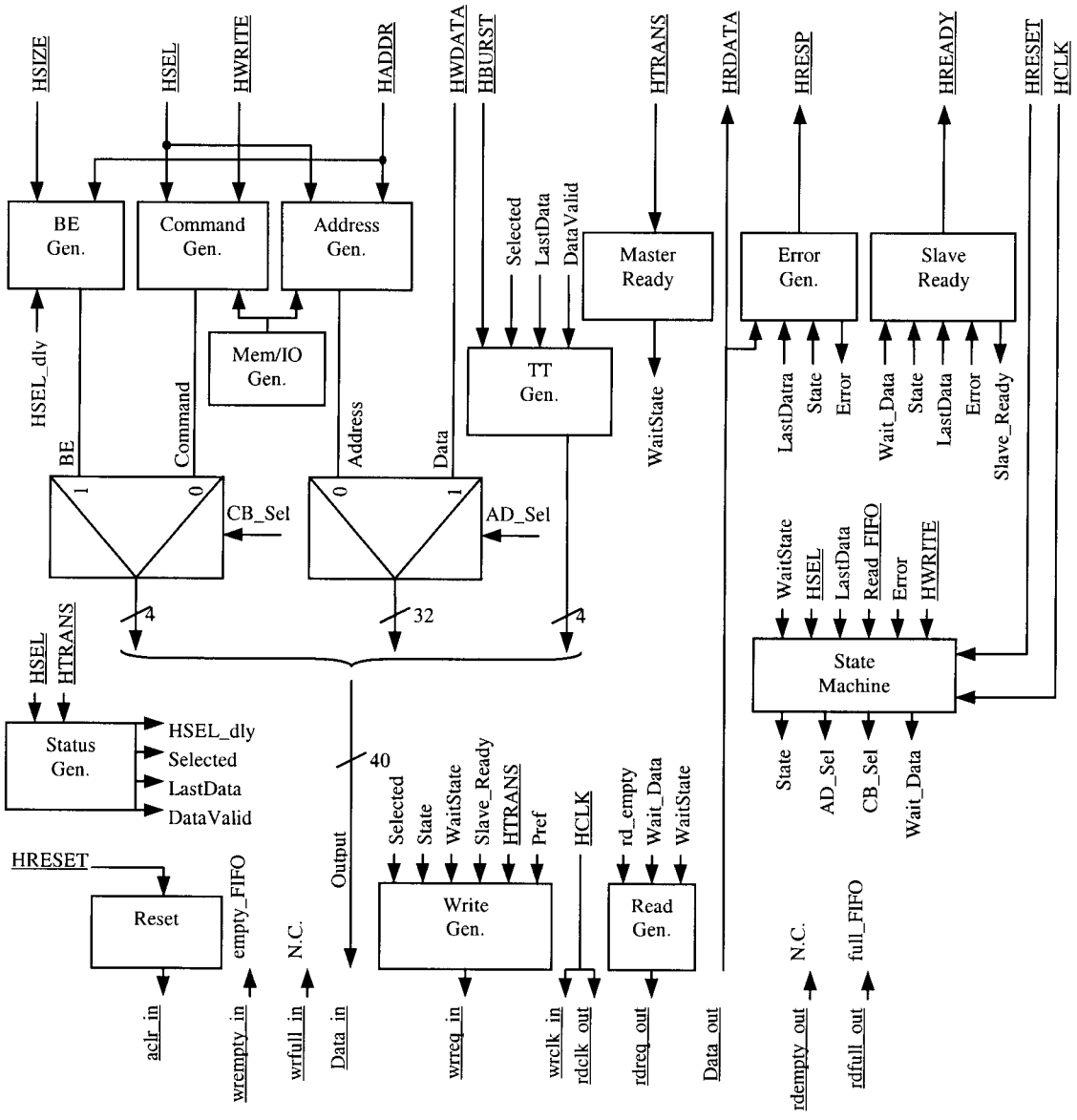

Figure 4.21. Structure of the AHB Slave Interface.

The blocks named BE Generator, Command Generator, Address Generator and TT Generator transform the data from the AHB bus into the 40-bit words to be sent to the PCI Interface. These blocks work asynchronously in order to have a fast conversion. The synchronous State Machine then selects the combination of output bits by means of two multiplexors. At the same time the Read and Write Generators set the read and write requests for the Master's FIFO's. The data given back from the PCI Interface in

a read cycle is sent directly to the AHB and on the last cycle of the read/write transfer the last word from the FIFO is analysed by the Error Generator and sent back to the AHB bus.

A more detailed description of the AHB Slave Interface blocks follows.

4.3.1. Status Generator

From the HSEL and HTRANS inputs, this block generates some internal signals used by other blocks:

HSEL_dly This is the HSEL signal delayed on one clock cycle. The HSEL signal is only active during the address phase but not during the data phase that occurs one clock later. The signal HSEL_dly will be active during the data phase.

Selected This signal will be active while there is an address or data phase taking place.

LastData It indicates that the last data phase of the transfer is taking place.

DataValid This signal indicates that data is being transferred.

4.3.2. BE Generator

This block generates the byte enable lines (in inverted logic) to be sent into the FIFO's. These lines have to be transferred to the FIFO's on the second word, that is why the signals have been delayed in one clock cycle. The byte enable lines are generated when the Slave is selected according to the following table:

HSIZE	HADDR[1:0]	BE	Description
BYTE	11	0111	8-bit transfer and offset = 3
BYTE	10	1011	8-bit transfer and offset = 2
BYTE	01	1101	8-bit transfer and offset = 1
BYTE	00	1110	8-bit transfer and offset = 0
HALFWORD	10	0011	16-bit transfer and offset = 2
HALFWORD	01	1100	16-bit transfer and offset = 0
OTHERS		0000	32+ bit transfer

4.3.3. Mem/IO Generator

To be implemented...

This block implements the 'Mem/IO' signal that indicates whether the transfer is addressed to the Memory or to the I/O Space.

4.3.4. Command Generator

This block generates the Command lines when the Slave is selected according to the following table:

HWRITE	Mem/IO	Command	Description
0	0	0010	I/O Read Command
0	1	0110	Memory Read Command
1	0	0011	I/O Write Command
1	1	0111	Memory Write Command

4.3.5. Address Generator

In the PCI bus the information contained in the two low order address bits (AD[1:0]) varies by the address space. In the I/O Address Space, all 32 AD lines are used to provide a full address [referencia a PCI Local Bus Specification] while in the Memory Address Space, the address is complemented with the byte enable lines sent in the following word.

4.3.6. Pref. and Lock Generator

To be implemented...

This block implements the 'Pref' and 'Lock' signals that indicate respectively whether the transfer is to a prefetchable or not-prefetchable address and whether the transaction should be locked.

4.3.7. TT Generator

This block implements the transfer type bits sent/received in the four least significant bits of the word. They are generated according to the following table:

LastData	DataValid	HBURST	Pref	TT	Description
1	X	X	X	1000	Last data / Stop
0	1	X	X	0100	Data valid
0	0	SINGLE	X	0011	Address single
0	0	OTHERS	1	0011	Address Burst Prefetch.
0	0	OTHERS	0	0111	Address Burst not-Prefetch

On the AHB Bus there are different burst accesses (INCR, WRAP4, INCR4,...) [2] but the PCI cannot distinguish them, that is why they all transform into a general Burst Address access.

4.3.8. Multiplexors

The multiplexors select between the Byte Enable and the Command lines and between the Address and Data lines. On every transfer to the PCI bus, the first cycle contains the Command and Address lines and the second cycle the data and the byte enable lines. The State Machine generates the selection lines.

4.3.9. Read Generator

A read request is sent to the Master's FIFO to read the words coming from the PCI Interface. This Read_FIFO signal is generated in the State S2 of the State Diagram and in the State S3 on a prefetchable read cycle provided there is data to be read in the FIFO and no WaitState signal is active.

4.3.10. Write Generator

The information received via the AMBA bus has to be transformed into words to be sent to the Master's FIFO. The write request signal will be generated to write the word containing the address in S0, to write the data in a write transfer (S1), and to write the byte enable lines in a not-prefetchable read cycle (S2, S3), provided the slave is selected and no wait signal is active.

4.3.11. Error Generator

This block generates the HRESP lines. It will give the error message ("01") when the message received in the last transfer is different from 'End Completion' ("0001") and will maintain it for two cycles, as the error message is a two-cycle response.

4.3.12. Master Ready

This block will indicate by inserting wait states when the master starting the transaction is not ready.

4.3.13. Slave Ready

The Slave Ready block generates one internal signal 'Slave_Ready' and one external signal 'HREADY'. They both indicate when the slave interface is ready and have the same logical value except in the case of consecutive transfers. In this case, the internal signal indicates that the slave is ready to continue with the transaction but externally a not-ready signal is given during one clock cycle due to the dual bus system of the AMBA bus. In the Master's FIFO the last data of one transfer and the address of the next transfer is given in two different words while in the AMBA bus they arrive on the same clock cycle, that is why a not-ready cycle is sent to the AMBA master.

4.3.14. State Machine

Figure 4.22 shows the state diagram implemented for the Slave Interface. When the Slave is selected (HSEL = 1) a write (S1) or a read (S2) transfer will start. After a write access, a read cycle is necessary to get the word containing the information about the transfer result.

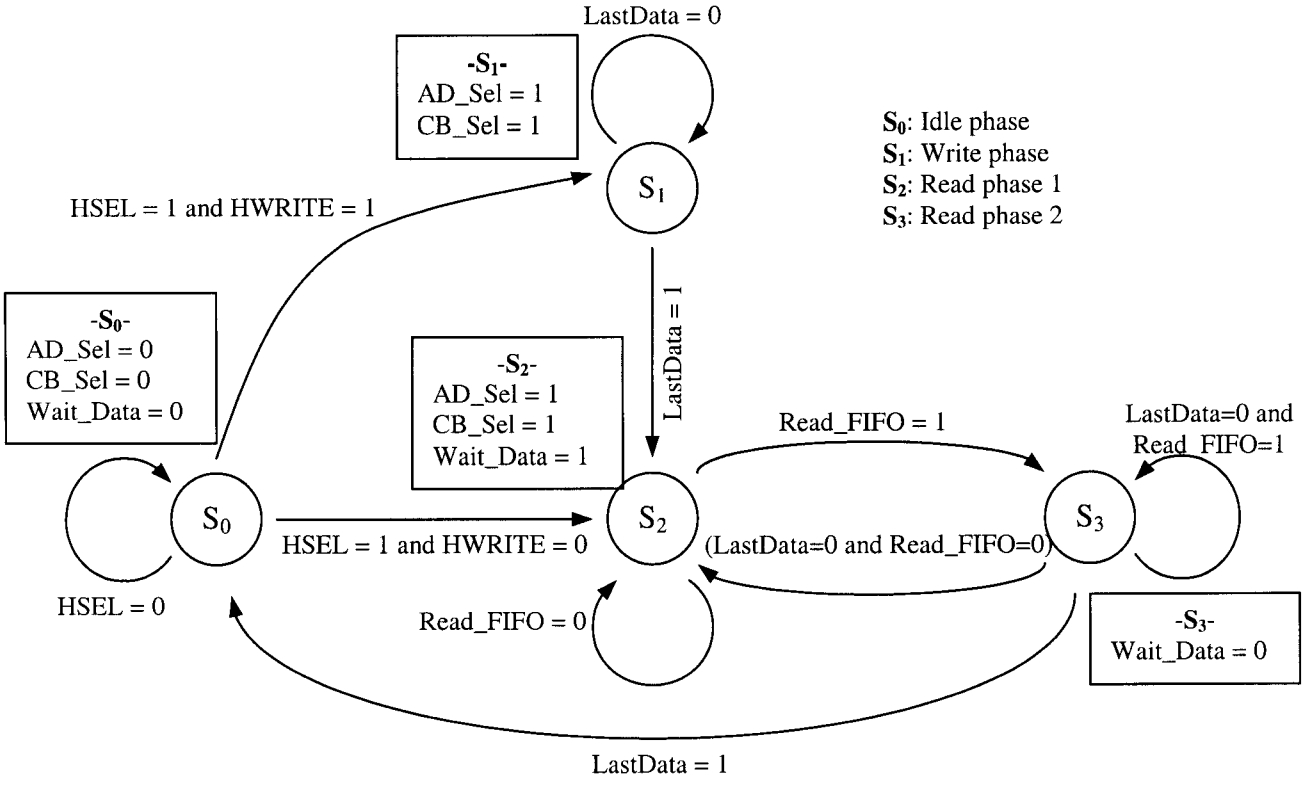

Figure 4.22. State Machine in the Slave Interface.

4.4. Design restrictions

It has to be taken into account that the information lines in the AMBA and PCI buses are not exactly the same. This gives some restrictions in the interconnection of the two.

One of these restrictions we find with the lock signal. The lock information is given by the Master starting the transaction in the AHB bus to the ARBITRER, but it does not arrive to the Slave side. This means that it cannot be passed to the PCI bus.

A similar situation occurs with the 'Memory/IO' and 'Prefetchable' signals. This information should be provided to the PCI in a bus transfer to indicate whether it is a Memory or an Input/Output access and whether it is to a Prefetchable or to a not-prefetchable address. But this information is not present in the AMBA bus.

Another restriction is the error messages which are different in both buses. While on the AHB bus the result of the transfer is given on every cycle of the transfer, on the

PCI bus it is given only at the end of it. The decision has been taken to send the OK message while the transaction is taking place and on the last cycle of the transfer wait for the OK or Error response form the Master's FIFO and send the corresponding message to the AHB. The retry message has been transformed into an Error message, the reason being that if a Retry message would be sent, the AHB master would only repeat the last cycle instead of the whole transfer as expected by the PCI Interface.

Chapter 5: Master Interface

The Master Interface receives the words from the FIFOs in the PCI Interface and generates the appropriate signals and data to send to the AHB Bus. On a read cycle it also receives the data from the AHB Bus and generates the words to be written to the FIFOs in the PCI Interface.

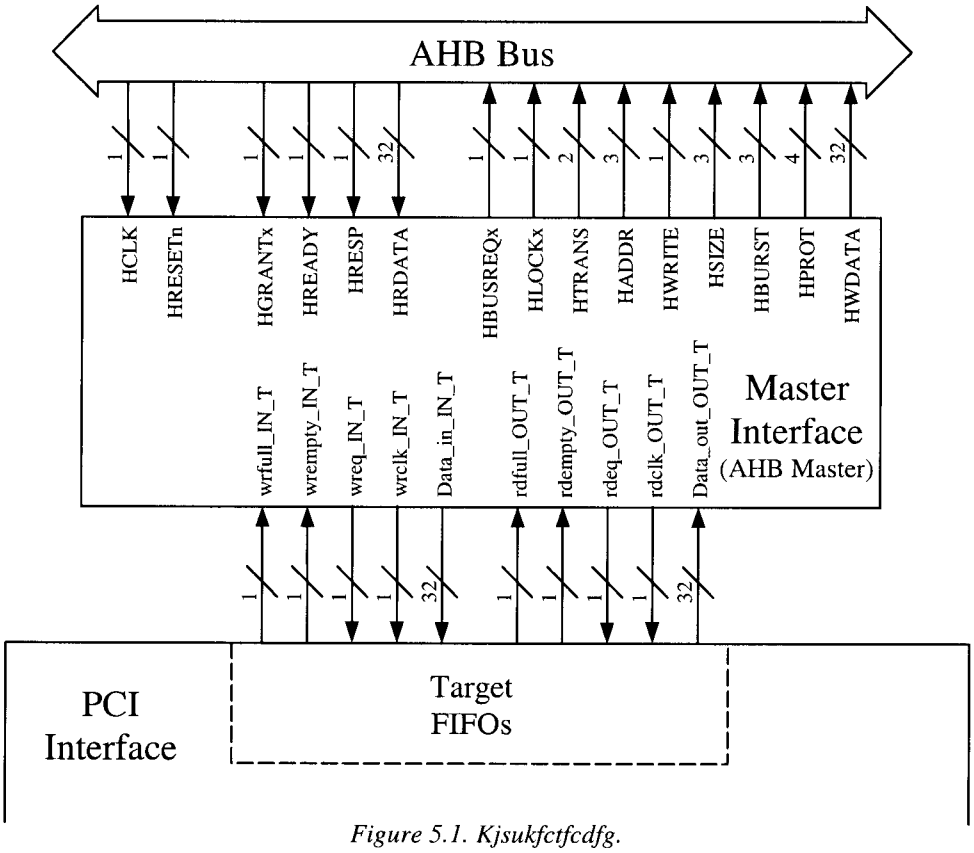

Figure 5.1. Kjsukfctfcdfg.

5.1. PCI Interface requirements.

The Target's FIFOs in the PCI Interface will receive the 40-bit words and will transfer them to the Master Interface. Some examples of different cycles follow [1][4].

5.1.1. Single write cycle

On a single write the FIFO will send a first word to the Master Interface that will contain the Lock request, the Info progress indicating a 'single write' cycle ("000"), the address and the command indicating a 'memory write' cycle ("0111") as shown in figure 5.2.

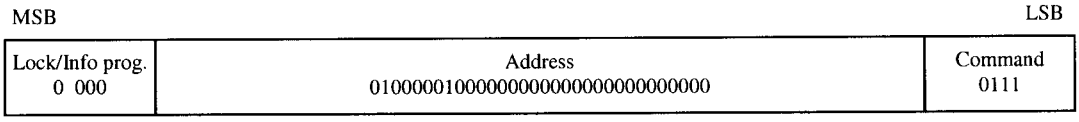

Figure 5.2. Word sent from the Target's FIFOs to the Master interface.

The second and last word will contain the Lock request, the Info progress ("110") indicating the Last Data condition, the data to be written and the Byte Enable lines (see figure 5.3).

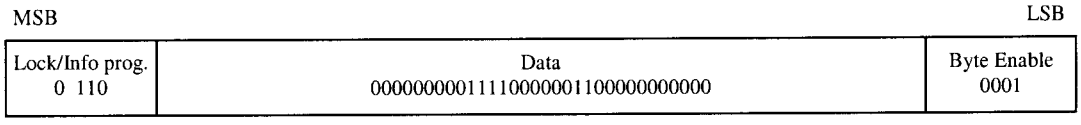

Figure 5.3. Word sent from the Target's FIFOs to the Master interface.

5.1.2. Burst write cycle

The first word sent by the FIFO in a burst write cycle will be the same as in the single write cycle with the exception of the info progress that will now indicate a 'burst write' cycle ("001") as shown in figure 5.4.

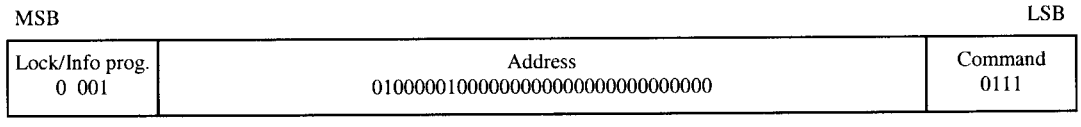

Figure 5.4. Word sent from the Target's FIFOs to the Master interface.

The following words will contain the data to be written with the info progress indicating 'Data valid' ("101") and the transfer will end when the last word is sent with the info progress "Last Data' ("110").

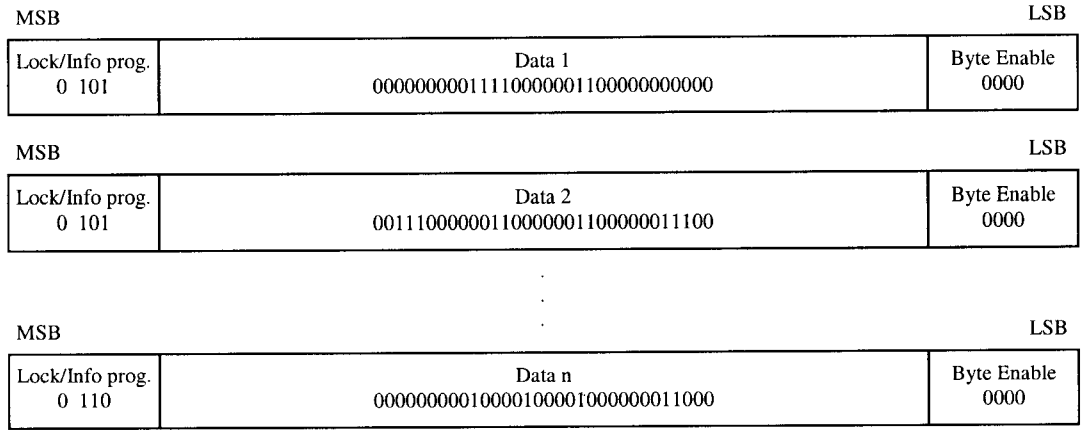

Figure 5.5. Words sent from the Target's FIFOs to the Master interface.

5.1.3. Single prefetchable read cycle

On a single read cycle to a prefetchable address the first word received from the Target's FIFO will contain the Lock request, the Info progress, the address and the command. It will then wait for the requested data to be sent. Once the data received, the Target's FIFO will send a 'Last Data' or 'Stop' message to the Master Interface. An example of such cycle is shown bellow:

The first word received from the Target's FIFO will contain: the Lock request in the most significant bit, the Info progress ("010" indicating Single Read cycle) in the three following most significant bits, the address and the command ("0110" in this case indicating Memory read cycle), as shown in figure 5.6.

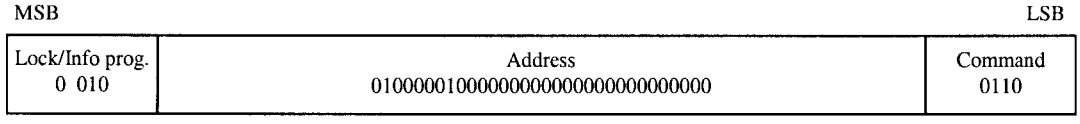

Figure 5.6. Word sent from the Target's FIFOs to the Master interface.

The FIFO will wait then for the data to be delivered in the 32 least significant bits of the word (see figure 5.7).

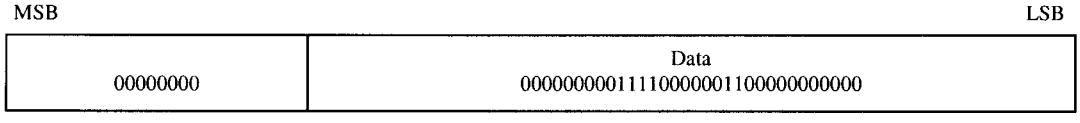

Figure 5.7. Word sent from the Master interface to the Target's FIFOs.

The FIFO will send a last word indicating on the 'Info progress' bits that that was the last word to be sent ("110").

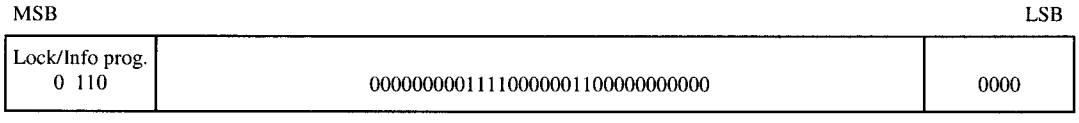

Figure 5.8. Word sent from the Target's FIFOs to the Master interface.

5.1.4. Single not-prefetchable read cycle

On a single read cycle to a not-prefetchable address the first word received will contain the Lock request, the Info progress, the address and the command and the second word will contain the Byte Enable lines. It will then wait for the data requested. An example of such cycle is shown bellow:

The first word received from the Target's FIFO will contain: the Lock request in the most significant bit, the Info progress ("010" indicating Single Read cycle) in the three following most significant bits, the address and the command ("0110" in this case indicating Memory read cycle), and in the second word the four least significant bits will contain the Byte Enable lines, as shown in figure 5.9.

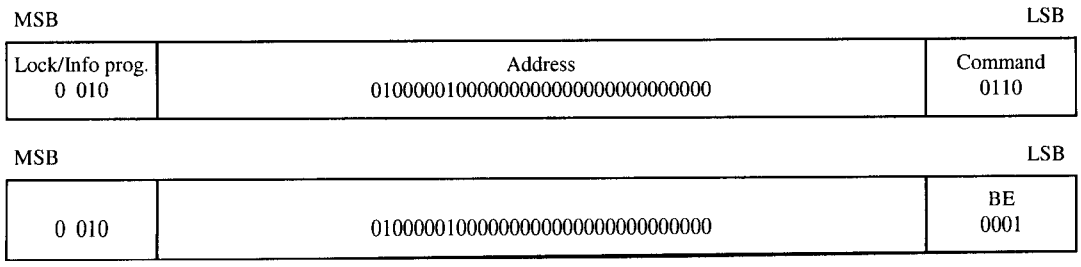

Figure 5.9. Words sent from the Target's FIFOs to the Master interface.

The FIFO will wait then for the data to be delivered in the 32 least significant bits of the word (see figure 5.10).

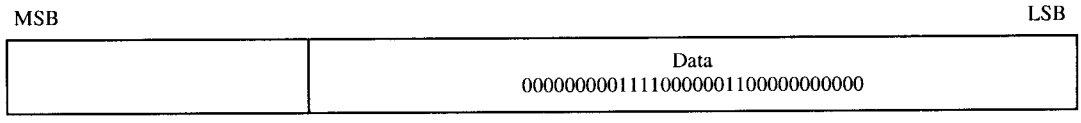

Figure 5.10. Word sent from the Master interface to the Target's FIFOs.

5.1.5. Burst prefetchable read cycle

On a burst read cycle to a prefetchable address the first word received from the Target's FIFO will contain the Lock request, the Info progress, the address and the command. It will then wait for the requested words. To stop the transfer the Target's FIFO will send a last word with the Stop message. An example of such cycle is shown bellow:

The first word received from the Target's FIFO will contain: the Lock request in the most significant bit, the Info progress ("011" indicating Burst Read cycle) in the three following most significant bits, the address and the command ("0110" in this case indicating Memory read cycle), as shown in figure 5.11.

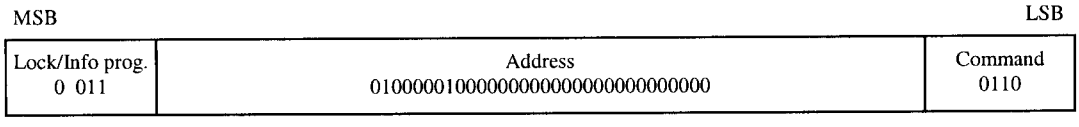

Figure 5.11. Word sent from the Target's FIFOs to the Master interface.

The FIFO will then wait for the data to be delivered in the 32 least significant bits of the word (see figure 5.12).

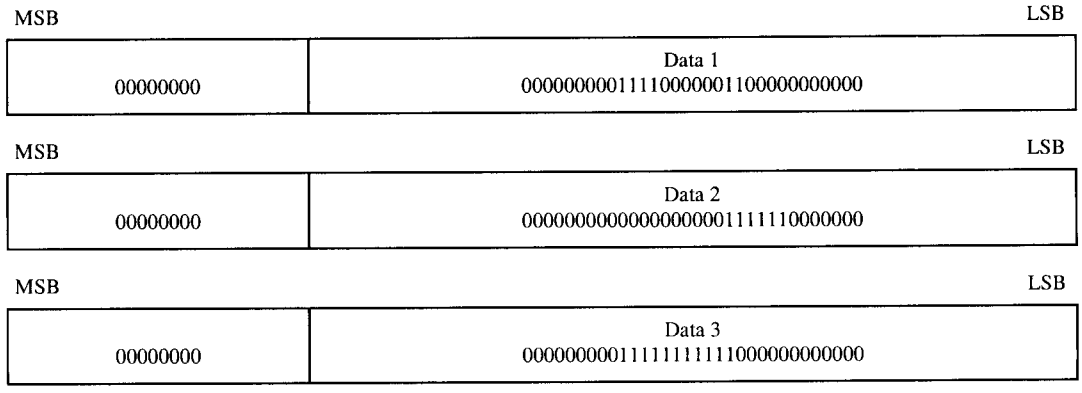

Figure 5.12. Word sent from the Master interface to the Target's FIFOs.

The Master Interface should continue to send data until a stop message is received form the Target's FIFO. The stop message will be sent in the 'Info progress' bits ("110") as shown in figure 5.13.

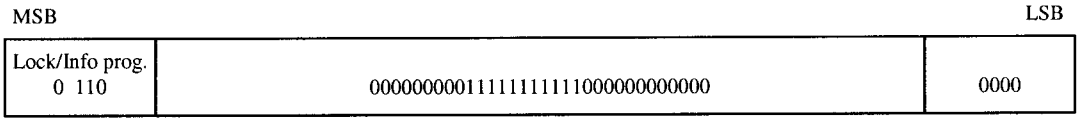

Figure 5.13. Word sent from the Target's FIFOs to the Master interface.

5.1.6. Burst not-prefetchable read cycle

On a burst read cycle to a not-prefetchable address the first word received will contain the Lock request, the Info progress, the address and the command and the second word will contain the Byte Enable lines. It will then wait for the data requested and send the Byte Enable lines for the next word. This sequence will continue until a 'Last Data' message is sent. An example of such cycle is shown bellow:

The first word received from the Target's FIFO will contain: the Lock request in the most significant bit, the Info progress ("011" indicating Burst Read cycle) in the three following most significant bits, the address and the command ("0110" in this case indicating Memory read cycle), and in the second word the four least significant bits will contain the Byte Enable lines, as shown in figure 5.14.

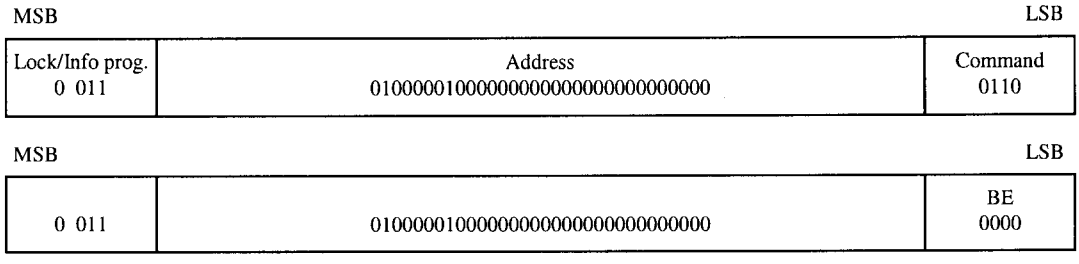

Figure 5.14. Words sent from the Target's FIFOs to the Master interface.

The FIFO will wait then for the data to be delivered in the 32 least significant bits of the word (see figure 5.15).

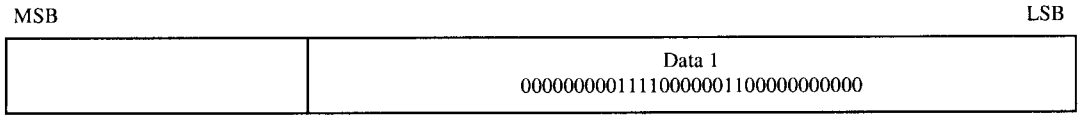

Figure 5.15. Word sent from the Master interface to the Target's FIFOs.

This sequence will continue as long as the 'Last Data' message is received in the 'Info progress' bits ("110") as shown in figure 5.16.

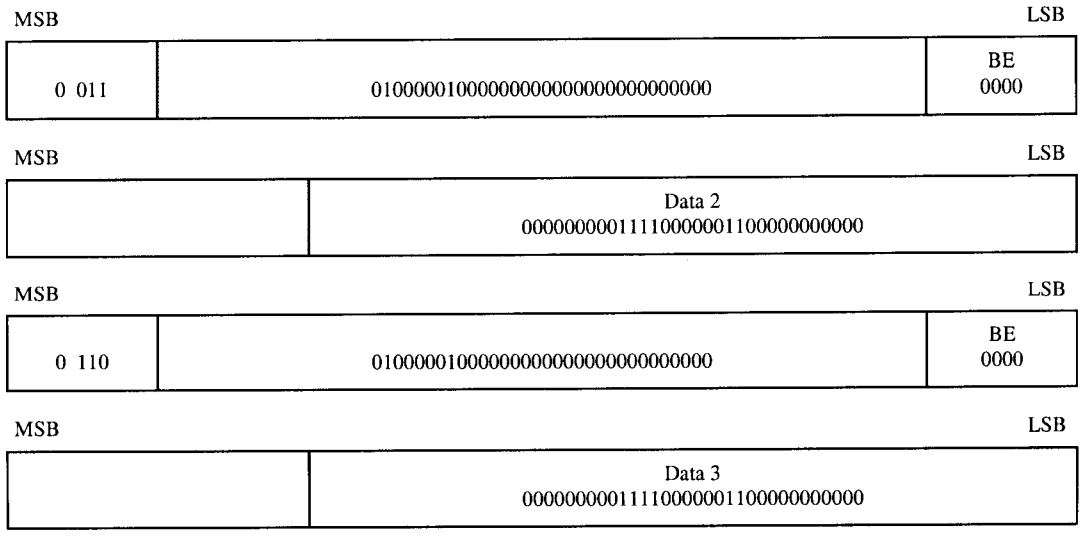

Figure 5.16. Words sent from the Target's FIFOs to the Master interface with the BE lines and words sent from the Master interface to the Target's FIFOs with the data requested.

5.2. AHB requirements.

The AHB Master Interface is connected to the AHB Bus through the following lines [2]:

HCLK, HRESET System clock and reset.

HGRANT The grant signal is generated by the arbiter and indicates that the appropriate master is currently the highest priority master requesting the bus.

HREADY This signal is used to extend the data phase. When LOW this signal causes wait states to be inserted into the transfer and allows extra time for the slave to provide or sample data.

HRESP This is the response signal used to shows the status of the transfer. It can have four possible values:

HRESP	Response	Description
00	OKAY	The OKAY response is used to indicate that the transfer is progressing normally. When HREADY goes HIGH this indicates the transfer has completed successfully.
01	ERROR	The ERROR response indicates that a transfer error has occurred and the transfer has been unsuccessful.
10	RETRY	Both the RETRY and SPLIT transfer responses indicate that the transfer cannot complete immediately, but the bus master should continue to attempt the transfer.
11	SPLIT	

HRDATA This is the read data bus driven by the appropriate slave during read transfers.

HBUSREQ The bus request signal is used by a bus master to request access to the bus.

HLOCK The lock signal is asserted by a master at the same time as the bus request signal. This indicates to the arbiter that the master is performing a number of indivisible transfers.

HTRANS This signal indicates the cycle type. There are four possible values:

HTRANS	Cycle	Description
00	IDLE	Indicates that no data transfer is required.
01	BUSY	Allows the bus master to insert IDLE cycles in the middle of burst of transfers.
10	NONSEQ	Indicates the first transfer on a burst or a single transfer.
11	SEQ	The remaining transfers in a burst are 'sequential' and the address is related to the previous transfer.

HADDR This is the bus where the address lines for every transfer are driven.

HWRITE This line indicates that the current transfer is a write cycle when HIGH or a read cycle when LOW.

HSIZE This signal indicates the size of the transfer, as shown in the following table:

HSIZE	Size	Description
000	8 bits	Byte
001	16 bits	Half word
010	32 bits	Word
011	64 bits	
100	128 bits	4-word line
101	256 bits	8-word line
110	512 bits	
111	1024 bits	

HBURST This signal indicates the transfer type. The possible values of this signal are summarised in the following table:

HBURST	Transfer	Description
000	SINGLE	Single transfer
001	INCR	Incrementing burst of unspecified length
010	WRAP4	4-beat wrapping burst
011	INCR4	4-beat incrementing burst
100	WRAP8	8-beat wrapping burst
101	INCR8	8-beat incrementing burst
110	WRAP16	16-beat wrapping burst
111	INCR16	16-beat incrementing burst

HPROT This is the protection control signal. Not all masters will be capable of generating accurate protection information, therefore it is recommended that slaves do not use the HPROT signals unless strictly necessary.

HWDATA This is the where the bus master will write the data on write transfers and hold it stable throughout the whole cycle.

Figure 5.17 shows the timing of the above signals in a basic transfer with a wait state cycle which extends the transfer allowing additional time for completion.

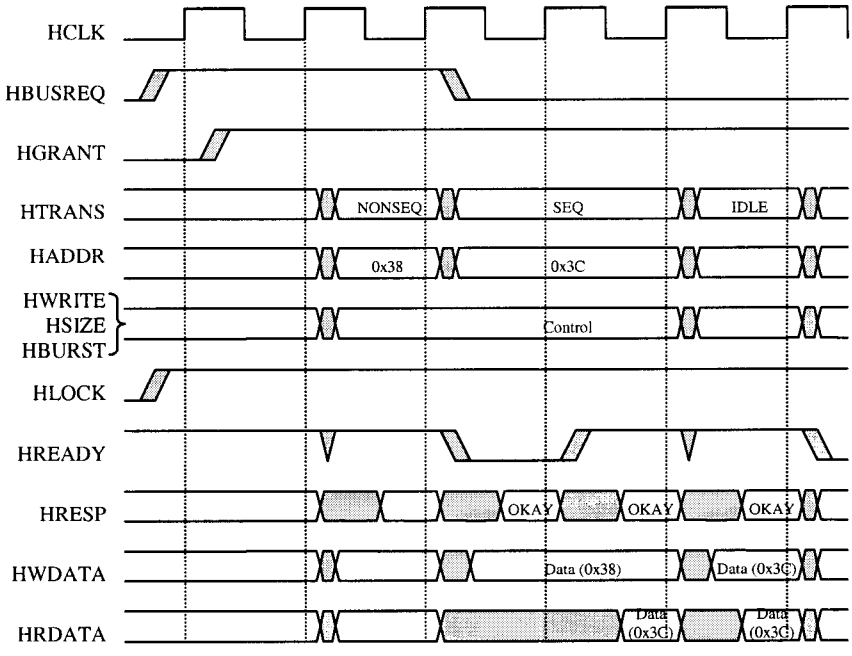

Figure 5.17. Signals sent and received from the Master on a basic AHB transfer.

The transfer is started by the AHB Master asserting a request signal to the arbiter. The transfer can commence when the master is granted the bus.

A granted bus master starts an AMBA AHB transfer by driving the address and control signals. These signals provide information on the address, direction and width of the transfer, as well as an indication if the transfer forms part of a burst.

A write data bus is used to move data from the AHB master to an AHB slave, while a read data bus is used to move data from an AHB slave to the AHB master.

For write operations the bus master will hold the data stable throughout the extended cycles. For read transfers the slave does not have to provide valid data until the transfer is about to complete. On every cycle the AHB Slave will report the status of the transfer by using the HRESP lines.

5.3. Design.

The Master interface has been designed taking into account the requirements explained in sections 5.1 and 5.2. An extra block has been designed between the Target's FIFOs and the Master Interface named 'Bit Organiser' (see figure 5.18).

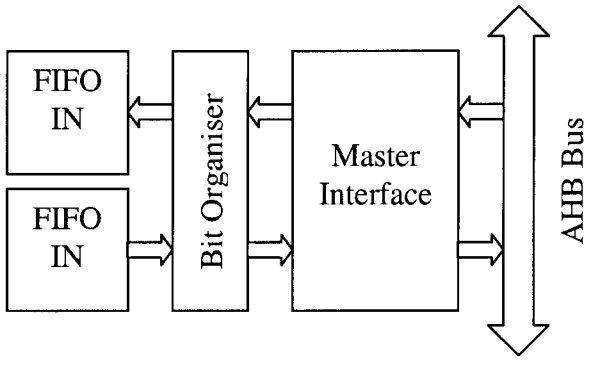

Figure 5.18. Position of the Bit organiser Block.

The position of the bits in the PCI Interface is not the same throughout the transfer cycles. For example the Master Block sends the command in the four most significant bits and the Target Block receives it in the four least significant bits. The same happens with the position of the data read in a read cycle: the Target Block must place the data in the 32 lowest bits while the master receives it in the 32 middle bits.

To avoid confusion, in the PCI Core / AMBA Bus Interface the bits will occupy always the same position in the word. For that reason a complementary block named 'Bit Structure' has been include. This block will keep the format of the data sent and received to/from the Master Interface the same as that sent and received to/from the Slave Interface. That format is the one shown in figure 5.19.

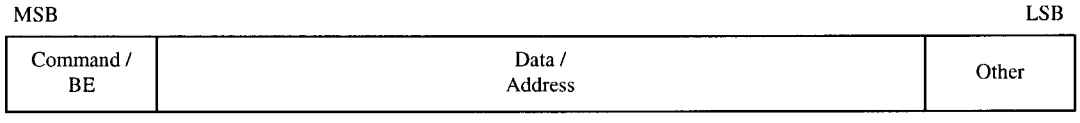

Figure 5.19. Format of the 40-bit words in the PCI Core / AMBA Bus Interface.

If in future versions of the PCI Core the structure of the bits is kept throughout the whole transfer, this block can be skipped.

Figure 5.20 shows the schematics of the Master Interface module. Underlined are the input and output lines, the rest being internal signals.

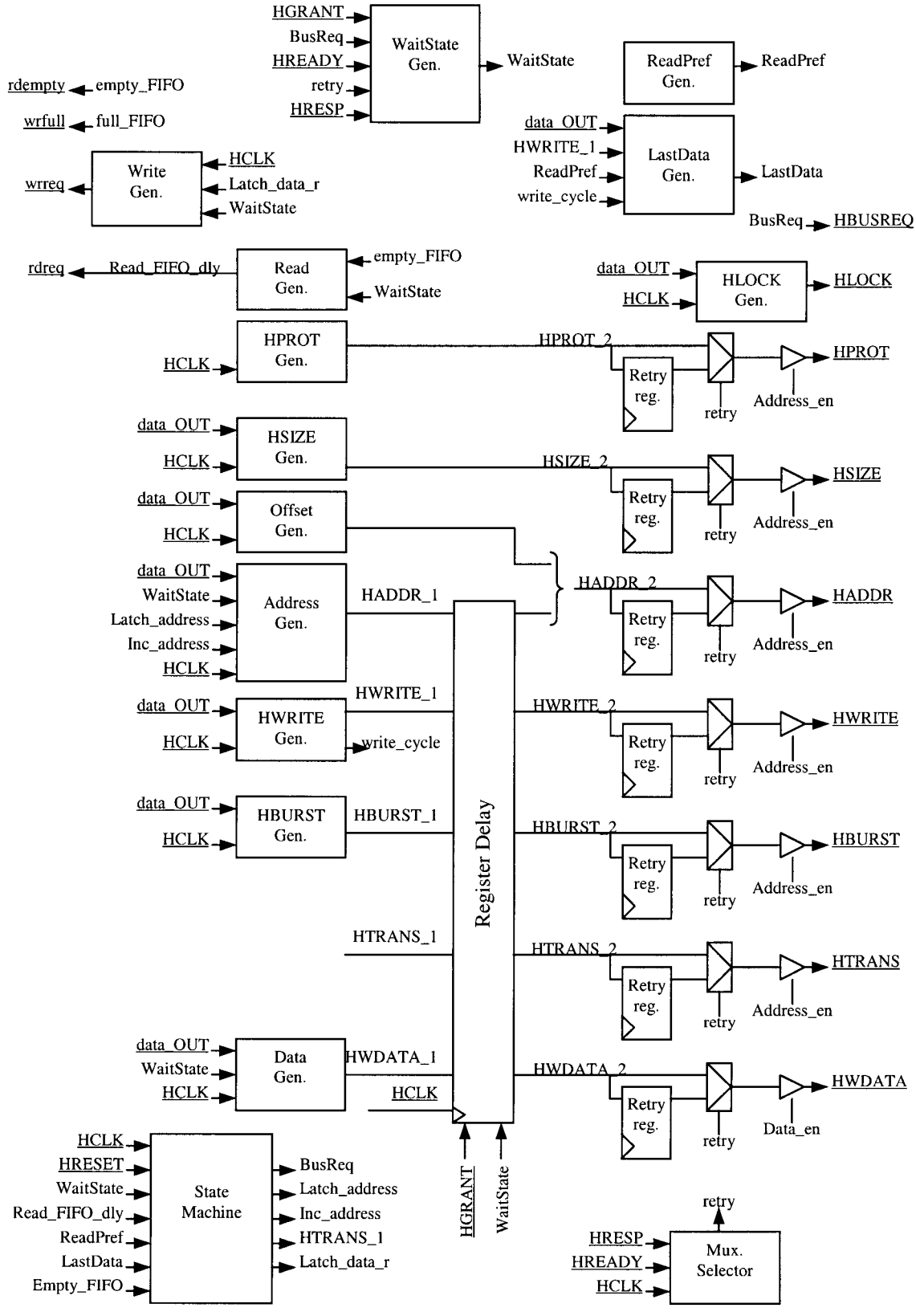

Figure 5.20. Structure of the AHB Master Interface.

The different generator blocks create the signals to be sent to the AHB bus from the words received form the Target's FIFOs. These signals are then delayed one or two clock periods depending on the order they are received and the order they have to be sent. They are also saved in a register in case the AHB Slave asks for a retry cycle.

A more detailed description of the Master Interface blocks follows.

5.3.1. HSIZE Generator

This block generates the HSIZE lines that indicate the size of the transfer. They will be generated from to the Byte Enable lines received from the Target's FIFO according to the following table:

BE	HSIZE	Description
0111	BYTE	8-bit transfer
1011	BYTE	8-bit transfer
1101	BYTE	8-bit transfer
1110	BYTE	8-bit transfer
0011	HALFWORD	16-bit transfer
1100	HALFWORD	16-bit transfer
0000	OTHERS	32+ bit transfer

5.3.2. Offset Generator

This block generates the two least significant bits in the Data bus. They will be generated from to the Byte Enable lines received from the Target's FIFO according to the following table:

BE	Offset	Description
0111	11	Offset = 3
1011	10	Offset = 2
1101	01	Offset = 1
1110	00	Offset = 0
0011	10	Offset = 2
1100	00	Offset = 0
0000	00	Offset = 0

5.3.3. Address Generator

This block generates the signals to be sent in the HADDR lines. This address has to be incremented in consecutive cycles so it has to main functions: latch the address coming for the Target's FIFO when a new transfer starts, and increment this address on the following cycles.

5.3.4. HLOCK Generator

The Lock signal will come directly from the first FIFO word in the most significant bit. It just needs to be transferred to the AHB.

5.3.5. HWRITE Generator

This block generates the HWRITE signal which indicates whether the transfer corresponds to a write or to a read access. This information will be given by the Target's FIFO in the 'Info Progress' bits according to the following table:

Info Progress	HWRITE	Description
000	1	Single write
001	1	Burst write
010	0	Single read
011	0	Burst read

5.3.6. HBURST Generator

This block generates the signal indicating the transfer type. In the AHB there are eight transfer types while only two in the PCI Interface. This means that a 'Single transfer' in the PCI Interface will correspond to a 'Single transfer' in the AHB and a 'Burst transfer in the PCI Interface will correspond to an 'Incrementing burst of unspecified length' in the AHB.

5.3.7. Data Generator

The HWRITE lines will be directly those coming from the Target's FIFO.

5.3.8. Write Generator

This block generates the 'wrreq' signal to be sent to the Target's FIFO to store a word in it. If will be set to one on a single read or burst read cycle.

5.3.9. Read Generator

This block generated the 'rdreq' signal to be sent to the Target's FIFO to read a word from it. It will be set to one when there is data to be read and no Wait State cycle is taking place.

5.3.10. Wait State Generator

This block generates Wait State cycles whenever the AHB Master or the AHB Slave are not ready to continue with the transfer. It will also insert a Wait State cycle when a retry cycle is taking place.

5.3.11. Read Prefetchable Generator

To be implemented...

This block implements the 'ReadPref' signal that indicates whether the transfer is to a prefetchable or not-prefetchable address.

5.3.12. Last Data Generator

This block generates the LastData signal that indicates that the last data phase of the transfer is taking place.

5.3.13. HPROT Generator

No level of protection has been implemented in this version.

5.3.14. Multiplexor Selector

This block generates the multiplexor selector signal. This signal selects between the general output lines on a normal cycle and those from the retry register whenever a retry cycle is taking place.

5.3.15. State Machine

Figure 5.21 shows the state diagram implemented for the Master Interface. The transfer would start when the output FIFO indicates it is not empty, meaning there is data to be read. At that moment it will go to state S1 and then to state S2 if it was a burst write cycle or to state S3 if it was a burst read cycle.

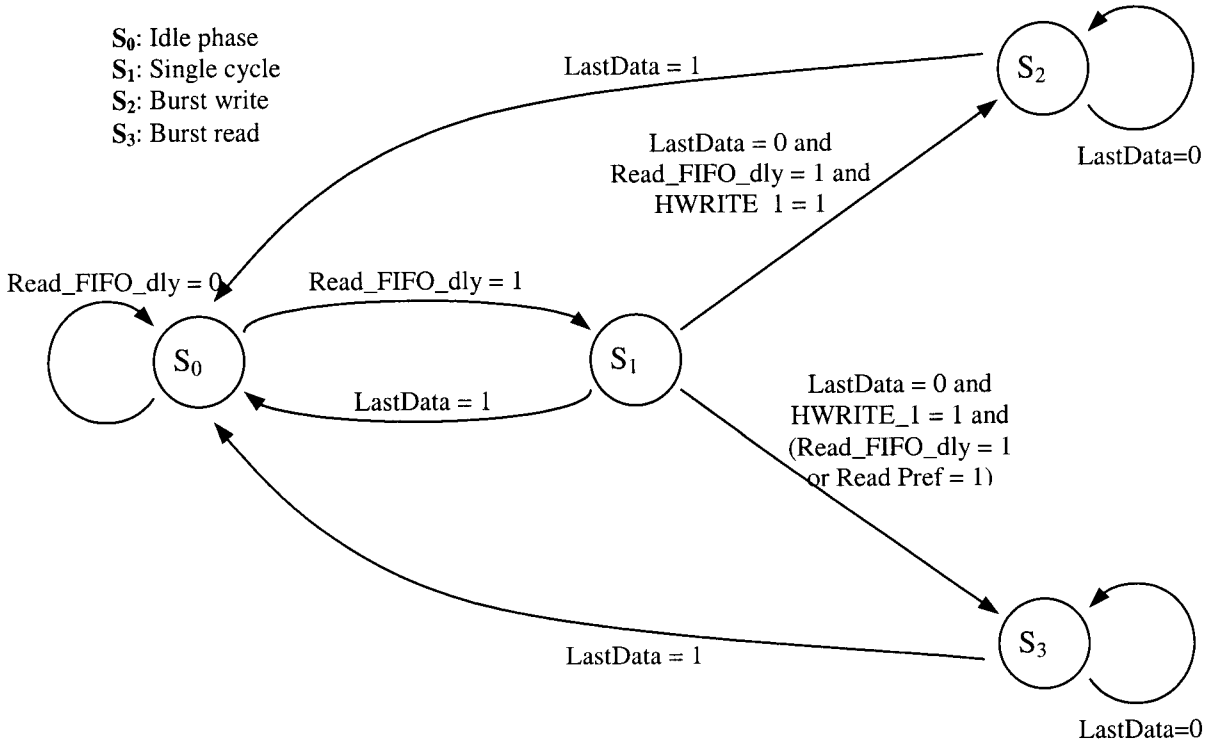

Figure 5.21. State Machine in the Master Interface

5.4. Design restrictions.

It has to be taken into account that the information lines in the AMBA and PCI busses are not exactly the same. This gives some restrictions in the interconnection of the two.

One of these restrictions can be found in the OK, Retry and Error signals provided through the HRESP lines: These signals that the AHB Slave sends to the Master Interface cannot be passed to the PCI Interface. The Target side of the PCI Interface will tell the PCI Master that the transfer went ok when the words are delivered to the synchronisation FIFO, not waiting for the information coming for the Master Interface.

Another restriction can be found with the Prefetchable information. Once a transfer is started, the Target side of the PCI Interface access the Configuration block to get the information of whether it is addressed to a prefetchable or to a not-prefetchable address. It then sends the appropriate word(s) to the Output synchronisation FIFO. The problem is that the Prefetchable information is not passed to the back-end application (the Master Interface in our case) and thus it does not know how many data to expect.

Chapter 6: Conclusions

The time diagrams obtained with the simulation of the PCI Core / AMBA Bus Interface designed prove its correct functioning. These simulations have included single and burst read/write transfers, retry cycles, bus grant taken before the transfer is completed, target or master not ready, et cetera.

There are nevertheless some features that have not been implemented in this version of the design and that are explained bellow:

1.- Wrap cycles.
The wrapping burst cycles that exist in AHB transfers do not exist in PCI transfers. To fit a wrapping burst cycle into a PCI transfer, it should be split into two different transfers. This has not yet been implemented in the Slave Interface.

2.- Split capabilities.
In this first version no split capabilities have been implemented in the Slave Interface.

3.- PCI / AMBA address conversion.
The addresses received from the PCI Interface are passed directly into the AHB and vice versa. This should be changed, as the addresses in the two busses may not be the same.

4.- Extensive testing.
The system designed has been tested with the synchronisation FIFOs. The next step would be to test it together with the PCI Interface, the PCI bus and the AHB, after which modifications may be needed in the current design.

There are also some problems related to the PCI Interface that are still to be solved:

1.- Burst read cycles.
The burst read cycles in the PCI Interface are not working correctly at the moment when this document is being written. For that reason they have not been implemented according to the FIFO's input/output but according to the documentation provided. This means that once the problem will be solved in the PCI Interface, modifications may be needed in the PCI Core / AMBA Bus Interface.

2.- I/O, Prefetchable and Lock generators.
The PCI Interface needs to be provided with information about whether the transfer is to an I/O or to a memory address, to a Prefetchable or not-prefetchable address and whether it is a lock access or not. Unfortunately the AHB does not provide the Slave

Interface with such information and thus it cannot be passed to the PCI Interface. A way should be found to generate these signals.

3.- Error transmission.
As explained in section 5.4, the PCI Interface will not wait for the OK/Error message form the Master Interface. It will instead assume that the transfer has been OK when the PCI Core writes successfully the corresponding data into the synchronisation FIFOs. It would be necessary to find a way to transmit the error information to the PCI Interface.

References

[1] Riccardo Locatelli, *"Master/Target PCI VHDL Core"*, E.W.P. 2047, ESA 1999.

[2] AMBA Bus Specification.

[3] J. Gaisler, *"LEON functional description"*, TOS-ESD/JG/501, ESA 1998.

[4] E. Lama Vaquero, *"PCI Interface"*, D/TOS-ESM/ELV/157, ESA 2000.