IDE/ATA Interface Controller with WISHBONE

June 2010 Reference Design RD1095

Introduction

Integrated Drive Electronics (IDE) is one of the most popular data bus interfaces for PCs. The IDE interface links a computer motherboard's data paths to the computer's disk storage devices. This interface is known by many different names including ATA, ATA/ATAPI, and EIDE. Although the IDE controller is built into the hard drive, an interface controller is required to bridge the hard disk's internal controller with the rest of the system. This function is often referred to as an IDE/ATA interface controller. This reference design implements a generic IDE interface controller compliant with the ATA/ATAPI-5 Standard. It is also a WISHBONE-compliant host controller that provides a simple interface to low-cost, non-volatile memories such as hard disk drives, CD-ROM players/writers and CompactFlash and PC card devices. This document and the design are based on the OpenCores ATA/ATAPI-5 core.

Features

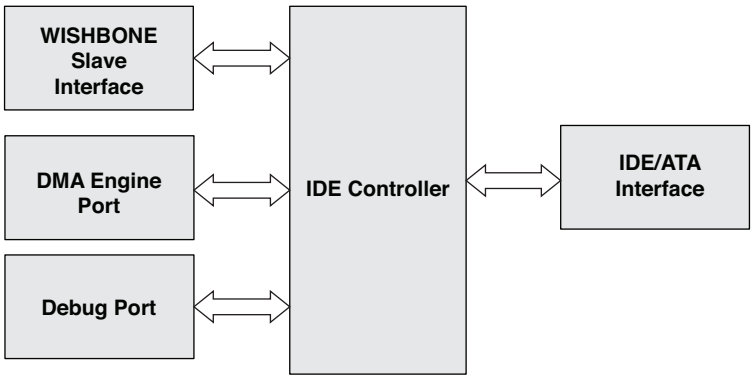
- ATA/ATAPI Revision 5 compliant design
- Common PIO (Programmed Input Output) compatible timing setting for all connected devices
- Fast PIO dataport timing settings for connected devices
- Single-word/multi-word timing settings for connected devices
- PIO mode that supports PINGPONG read and write
- Automatic big endian versus little endian conversion
- DMA read and write buffer
- WISHBONE DMA engine compatible
- WISHBONE Revsion B2 compliant
- 32-bit host interface
- Operation in a wide range of input clock frequencies

Functional Description

The IDE interface controller can interface with a host controller and two hard disk drives. The two hard disks are configured as master and slave so that they can share the same signal lines from the controller. The identification of master and slave devices is handled by setting the CSEL signal and configuring the DEVICE/HEAD register to the appropriate values.

The processor implements high-level applications and the software drive for the IDE device while the interface controller core performs low-level, time-intensive tasks. These tasks include the programming of internal registers for a hard disk, transferring data in burst mode, and buffering data to and from the processor and hard disks.

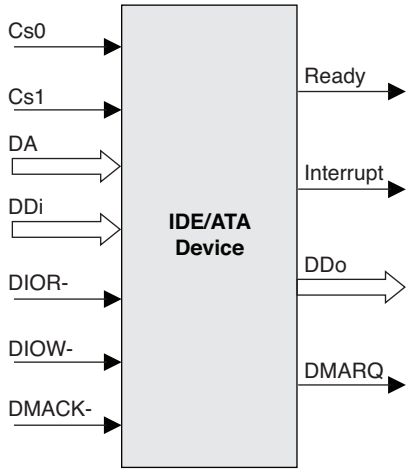
Figure 1. Interface Diagram



Various interfaces provided by this design are shown in Figure 1. The interfaces can be divided into four types.

1. **WISHBONE Slave Interface** – A standard slave WISHBONE interface connects the host processor with an IDE interface controller. It supports a 32-bit data-like input/output signal, 6-bit address signal, read/write enable signal, control signals, interrupt signal, and global signals such as clock and reset. The configuration of all controller registers, and the read/write of the ATA device registers can be done through this interface.
2. **DMA Engine Port** – The controller can access external memory through this port in DMA transfer mode.
3. **Debug Port** – This interface is used to monitor the transmit buffer and receive FIFO in a simulation procedure. These ports are optional in this design.
4. **IDE/ATA Interface** – The IDE/ATA interface is an asynchronous data communication bus. All data transfer is controlled by the host. Figure 2 shows the interface of the IDE/ATA bus.

Figure 2. IDE BUS Signal



The signals on the left side are the input signals of the hard disk and the signals on the right are the output signals. The signals Cs0, Cs1 and DA denote one asynchronous address. The DDi and DDo signals indicate the data signal to/from the hard disk. The read and write enable signals are DIOR- and DIOW- respectively. The ready signal indicates that the hard disk is in the idle state and it can receive data or commands from the host. The interrupt signal is used to indicate the completion of one operation or that an error has occurred. The DMARQ and DMACK- signals are used in the DMA transfer procedure.

Table 1 describes the signals used in this design.

Table 1. IDE/ATA Interface Controllwer Signal Descriptions

Port	Direction	Width	Description
WISHBONE Interface			
wb_clk_i	Input	1	Clock input
wb_rst_i	Input	1	Synchronous active high reset
arst_i	Input	1	Input asynchronous reset
wb_adr_i	Input	5	Input address bits
wb_dat_i	Input	32	Input data towards the core
wb_dat_o	Output	32	Output data from the core
wb_sel_i	Input	4	Input byte select signals
wb_we_i	Input	1	Input write enable input
wb_stb_i	Input	1	Input strobe signal/core select input
wb_cyc_i	Input	1	Valid bus cycle input
wb_ack_o	Output	1	Output bus cycle acknowledge output
wb_rty_o	Output	1	Output bus cycle retry output
wb_err_o	Output	1	Output bus cycle error output
wb_inta_o	Output	1	Output Interrupt request signal output
DMA Interface			
DMA_req	Output	1	DMA request signal to external DMA engine
DMA_ack	Input	1	DMA acknowledge from external DMA engine
IDE/ATA Interface			
resetn_pad_o	Output	1	Output IDE hardware reset
dd_pad_i	Input	16	Input device data (from ATA devices)
dd_pad_o	Output	16	Output device data (towards ATA devices)
dd_padoe_o	Output	1	Output device data output enable
da_pad_o	Output	3	Output device address
cs0n_pad_o	Output	1	Output chip select0
cs1n_pad_o	Output	1	Output chip select1
dmarq_pad_i	Input	1	Input DMA request
dmackn_pad_o	Output	1	Output DMA acknowledge
diorn_pad_o	Output	1	Output Device I/O read
diown_pad_o	Output	1	Output Device I/O write
iordy_pad_i	Input	1	Input I/O channel ready
intrq_pad_i	Input	1	Input device interrupt

Table 2 lists the parameters that need to be configured in the initial state.

Table 2. Default Parameters

Parameter	Default	Description
ARST_LVL	1	Asynchronous reset level
TWIDTH	8	Internal counter width
PIO_mode0_T1	6	PIO mode0 address valid to DIOR-/DIOW- setup
PIO_mode0_T2	28	PIO mode0 DIOR-/DIOW- pulse width
PIO_mode0_T4	2	PIO mode0 DIOW- data hold
PIO_mode0_Teoc	23	PIO mode0 end of cycle time
DMA_mode0_Tm	4	DMA mode0 CS(1:0) valid to DIOR-/DIOW-
DMA_mode0_Td	21	DMA mode0 DIOR-/DIOW- asserted time
DMA_mode0_Teoc	21	DMA mode0 end of cycle time

Data Transfer Protocol and IDE Internal Register

The IDE/ATA Interface Controller protocol supports three data transfer modes. Each mode has different data transfer timing parameters.

The first data transfer mode is register data transfer. This is the most simple method for transferring data. It is used primarily for configuring registers.

The second mode is the PIO data transfer protocol. In PIO mode, the transfer of data is based on one or more sectors. When a data block is transferred completely, the hard disk generates an interrupt signal to report the result to the host. PIO mode has five transfer speeds. Table 3 lists the transfer cycle values and their corresponding transfer speeds.

Table 3. PIO Mode Transfer Timing Parameters and Speeds

PIO Transfer Mode	Theoretical Values	
	Cycle Time (ns)	Transfer Rate (Mbps)
0	600	3.33
1	383	5.22
2	240	8.33
3	180	11.11
4	120	16.67

The third transfer mode is the multi-word DMA data transfer protocol. The advantage of the DMA transfer is that it does not require processor intervention and can exchange data directly with external memory. There are two differentDMA transfer modes. One is the multi-word mode and the other is the ultra DMA mode. Ultra DMA mode samples data on the positive and negative edges of the DIOR- or DIOW-. This design does not support ultra DMA mode. For DMA transfer, there are two directions for transferring data between the host and the device: writing data to the hard disk or reading data from the hard disk. DMA mode offers the transfer cycle values and speeds listed in Table 4.

Table 4. DMA Mode Transfer Timing Parameters and Speeds

DMA Transfer Mode	Theoretical Values	
	Cycle Time (ns)	Transfer Rate (MBps)
0	480	4.2
1	150	13.3
2	120	16.7

The host's control over the hard disk is implemented through the reading and writing of two sets of registers to and from the hard disk. One set of registers is defined as the command register, the other is the control/diagnostic register. These two register sets are the internal registers of the hard disk. Table 5 lists all internal registers.

Table 5. Internal Registers of an IDE Device

Address					Name and Definition	
CS1 (cs1n_pad_o)	**CS0 (cs0n_pad_o)**	**DA2 (da_pad_o[2])**	**DA1 (da_pad_o[1])**	**DA0 (da_pad_o[0])**	**Read (diorn_pad_o)**	**Write (diown_pad_o)**
Command Registers						
1	0	0	0	0	Data register	Data register
1	0	0	0	1	Error register	Feature register
1	0	0	1	0	Sector count register	Sector count register
1	0	0	1	1	Sector number register	Sector number register
1	0	1	0	0	Cylinder low register	Cylinder low register
1	0	1	0	1	Cylinder high register	Cylinder high register
1	0	1	1	0	Device/head register	Device/head register
1	0	1	1	1	Status register	Command register
Control and Diagnostic Register						
0	1	1	1	0	Status register (alternate)	Device control register
DMA Port (DMA Mode)						
1	1	*	*	*	Data port	Data port

Data transfer will not begin until all internal registers of the IDE device are configured. This design provides a CTRL register in addition to the IDE internal registers. Through the CTRL register, the host can select the transfer mode to be either PIO mode or DMA mode. After the transfer mode is selected, the user begins to configure the Command parameter register set. The Feature register, Sector Count register, Sector Number register, Cylinder low register, Cylinder high register and Device/Head register must be configured before the configuration of the Command register. All IDE internal registers are eight bits wide.

The Command register stores a command that is compliant with the ATA/ATAPI specification. Different transfer types have different command codes and command parameters. Users can refer to the specification to select the corresponding command code and parameters. The hard disk executes according to the contents of the Command register.

The Status register stores the results of the command operation. The host can read the Status register to determine the status of the hard disk and whether the former command has been received correctly. The Status register contains the following bits that reflect the result:

- **BSY** – Seventh bit of the Status register, indicates whether the hard disk is busy.
- **DRDY** – Sixth bit of the Status register, indicates whether the hard disk is ready.
- **DF** – Fifth bit of the Status register, indicates the hard disk is in fault state.
- **DRQ** – Third bit of the Status register, indicates one data request is being sent by the hard disk.

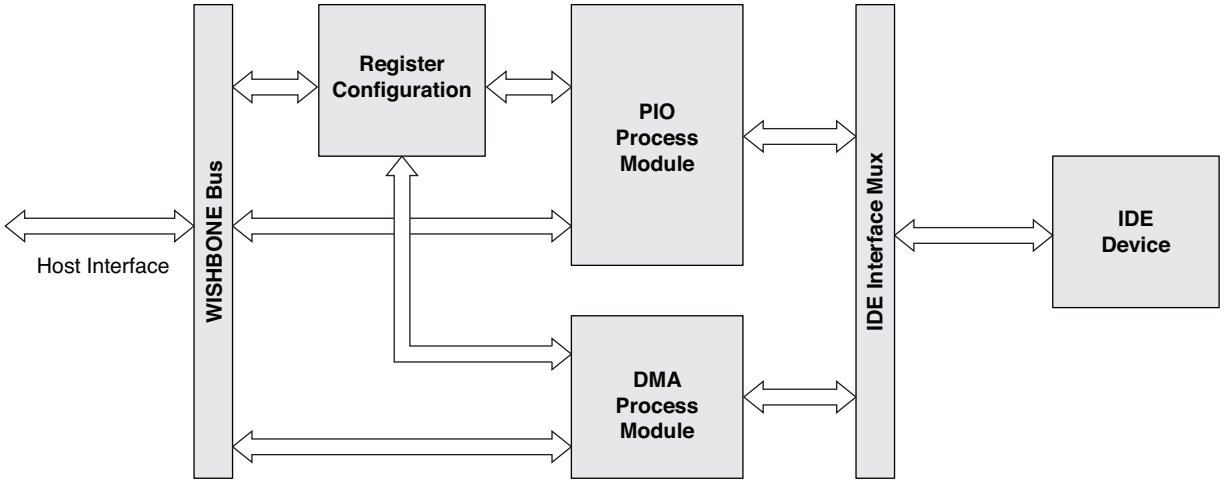
The Alternate Status Register contains the same information as the Status register. Reading it does not clear a pending interrupt.

The Device Control Register contains the software reset [SRST] and the interrupt enable [nIEN] bits. When the Device Control register is written, both devices respond to the write, regardless of which device is selected.

Design Module Description

The modules in this design are shown in Figure 3. This reference design can be divided into three modules: the register configuration module, PIO data transfer module, and DMA data transfer module. The modules and user registers are described below.

Figure 3. Internal Block Diagram



Register Configuration Module

The primary function of this module is to implement the slave WISHBONE interface and to read/write all user registers. It manages all registers according to the requirements of the host. Based on the contents of the CTRL register, it generates different control signals for the PIO process module and DMA process module.

Table 6 provides a description of each of the seven registers used in this design.

Table 6. User Registers

Name	Address	Width	Access	Description
CTRL	0x00	32	Read/Write	Control register
STAT	0x01	32	Read	Status register
PCTR	0x02	32	Read/Write	PIO compatible timing register
PFTR0	0x03	32	Read/Write	PIO fast timing register device0
PFTR1	0x04	32	Read/Write	PIO fast timing register device1
DTA0	0x05	32	Read/Write	DMA timing register device0
DTA1	0x06	32	Read/Write	DMA timing register device1
DTxDB	0x07	32	Write	DMA transmit data buffer
DRxDB	0x07	32	Read	DMA receive data buffer

CTRL Register

The 32-bit CTRL register is used to control all transfer operations between the host and the hard disk. The 15th bit is the DMA enable signal. When the user wants to set the transfer mode as DMA mode, it must enable this bit first. The 13th bit is the DMA transfer direction control signal. Through the setting of this bit, the transfer direction (read/write) can be determined. The 8th and 9th bits are the big endian/little endian conversion bits for device0 and device1. A '1' set for these bits indicates the big endian/little endian conversion in the DMA transfer procedure. The 7th bit is the IDE enable signal. When the user wants to set the transfer mode as IDE mode, this bit must be set. The 4th bit is the PINGPONG enable signal. Enabling this bit provides a performance enhancement for the PIO transfer. In PIO transfer mode, if the selected device supports fast timing and the corresponding registers are programmed, the host must set the 6th and 5th bits. In setting these two bits, the controller will generate the timing

control signal according to PFTR0 or PFTR1. The first bit of this register is the reset signal. The user can reset all signals to the default state by setting this bit. All other bits of this register are the reserved bits for future applications.

Table 7. CTRL Register

Bit #	Access	Description
31:16	R/W	Reserved
15	R/W	DMAen, DMA enable signal
14	R/W	Reserved
13	R/W	DMAdir, DMA direction
12:10	R/W	Reserved
9	R/W	BeLeC1, big endian/little endian conversion device1
8	R/W	BeLeC0, big endian/little endian conversion device0
7	R/W	IDE enable signal
6	R/W	Fast timing device1 enable
5	R/W	Fast timing device0 enable
4	R/W	PIO write PINGPONG enable
3	R/W	Fast timing device1 IORDY enable
2	R/W	Fast timing device0 IORDY enable
1	R/W	Compatible timing IORDY enable
0	R/W	Reset signal

STAT Register

The 32-bit STAT register shows the status of the controller in the data transfer procedure. This register can be read to determine the state of the transmit buffer, the receive FIFO, and the hard disk status registers. The 15th bit indicates that the controller is doing DMA data processing. The 10th bit indicates the full of the receive FIFO and the 9th bit indicates the empty state of the transmit FIFO. The 8th bit records the DMA request signal from the hard disk. The 7th bit indicates that the PIO transfer in process. The 6th bit indicates write PINGPONG is full and the host must wait until PINGPONG is empty. The first bit is the IDE interrupt signal that indicates an interrupt asserted by the hard disk. The user can look at this bit to know if the hard disk sends a request to the host in an abnormal case. All other bits are the reserved bits.

Table 8. STAT Register

Bit #	Access	Description
31:16	R	Reserved
15	R	DMAtip, DMA transfer in process
14:11	R	Reserved
10	R	DMA receive FIFO empty
9	R	DMA transmit fifo full
8	R	DMARQ line status
7	R	PIOtip, PIO transfer in process
6	R	PIO Write PINGPONG full
5:1	R	Reserved
0	R	IDE interrupt signal status

PIO Compatible Timing Register (PCTR)

This register stores the default and slowest timing parameters. The IDE/ATA specifications refer to these settings as “Register Transfer Timing Parameters”.

Table 9. PIO Compatible Timing Register (PCTR)

Bit #	Access	Description
31:24	R/W	Teoc, end of cycle time
23:16	R/W	T4, DIOW- data hold
15:8	R/W	T2, DIOR-/DIOW- pulse width
7:0	R/W	T1, address valid to DIOR-DIOW-

PIO Fast Timing Register Device 0 [PFTR0] and Device 1[PFTR1]
The IDE/ATA specificationss refer to these settings as “PIO Data Transfer Timing Parameters”. The user needs to select the corresponding register according to the parameter of the Device/Head register.

Table 10. PIO Fast Timing Register Device 0 [PFTR0] and Device 1[PFTR1]

Bit #	Access	Description
31:24	R/W	Teoc, end of cycle time
23:16	R/W	T4, DIOW- data hold
15:8	R/W	T2, DIOR-/DIOW- pulse width
7:0	R/W	T1, address valid to DIOR-DIOW-

DMA Timing Register Device 0 [DTR0] and Device 1 [DTR1]
The IDE/ATA specificationss refer to these settings as “Multiword DMA Data Transfer Timing Parameters”.

Table 11. DMA Timing Register Device 0 [DTR0] and Device 1 [DTR1]

Bit #	Access	Description
31:24	R/W	Teoc, end of cycle time
23:16	R/W	Reserved
15:8	R/W	Td, DIOR-/DIOW- pulse width
7:0	R/W	Tm, address valid to DIOR-DIOW-

DMA Transmit Data Buffer
The DMA Transmit Data Buffer contains the data to be written into the IDE/ATA device using DMA transfer accesses.

Table 12. DMA Transmit Data Buffer

Bit #	Access	Description
31:24	R/W	Transfer data1(15:8)
23:16	R/W	Transfer data1(7:0)
15:8	R/W	Transfer data2(15:8)
7:0	R/W	Transfer data2(7:0)

DMA Receive Data Buffer
When the host reads from the buffer while it is empty (DMATRxEmpty = ‘1’), the core acknowledges the cycle, but the read data is invalid. The buffer is flushed when the ARST bit is set (‘1’).

Table 13. DMA Receive Data Buffer

Bit #	Access	Description
31:24	R/W	Second receive DD(7:0)
23:16	R/W	Second receive DD(15:8)
15:8	R/W	First receive DD(7:0)
7:0	R/W	First receive DD(15:8)

PIO Process Module

This module is used for data transfer using the PIO mode. This mode has two types of data transfer: register transfer protocol and data transfer protocol.

Register transfer protocol is used in the configuration of the IDE/ATA internal register. All the data are transferred with an 8-bit width at the lowest speed. The corresponding timing parameters for PIO Transfer Mode 0 are given in Table 14.

Table 14. PIO Timing Parameters

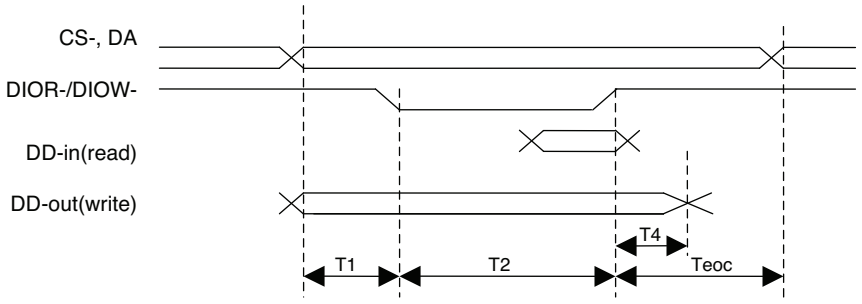
PIO Transfer Mode	Cycle Time (ns)	PIO Mode			
		T1 (ns)	T2 (ns)	T4 (ns)	Teoc (ns)
0	600	70	290	30	240
1	383	50	125	20	208
2	240	30	100	15	110
3	180	30	80	10	70
4	120	25	70	10	25

The 16-bit PIO mode transfers data at a higher speed than register data transfer speed. It uses the data register to transfer data to the hard disk. The contents of the internal registers define the start boundary of the data transfer and the number of required sectors. This data transfer mode has four different data transfer timing parameters that correspond to the PIO modes 0, 1, 2, 3 and 4 in Table 14.

According to the speed of the selected hard disk, the host can configure different timing parameters for the controller. This module generates the corresponding IDE/ATA interface signals to the hard disk.

The read/write timing diagram for PIO transfer mode is shown in Figure 4.

Figure 4. PIO Transfer Mode Timing



According to the PIO transfer protocol, T1 is the minimum period of the address (CS-, DA) before the read and write pulses (DIOR-, DIOW-) is valid. T2 is the minimum hold time of the DIOR-/DIOW- signal. T4 is the minimum time for the data hold time in the writing procedure. Teoc is the maximum period for the data bus release in reading procedure.

This reference design uses a 100 MHz clock as the processing clock. The T1, T2, T4 and Teoc time delays are implemented with four different counters working at this 100 MHz clock frequency.

The function of this module can be summarized as follows:

1. Select the CS signals and DA signals for the ATA device as the asynchronous address.
2. Wait T1 time, set the DIOR- and DIOW- valid once the data is ready.
3. Wait T2 time, check the IORDY signal and wait for the valid of IORDY.
4. If IORDY is valid, latch the data for the read operation. For the write operation, hold the data for T4 time.
5. Wait for the end of one read or write cycle, then wait for the next read or write request from the host.

When using PIO mode to transfer data, the operation flow is as follows.

1. Set the cs1n_pad_o(Cs1) at the high level; set the cs0n_pad_o(Cs0) with low level.
2. Configure the internal register and set the da_pad_o (DA) signal from 3'h1 to 3'h7.
3. Configure the timing parameter register of the selected device.
4. Set the IDE enable and PINGPONG enable by configuring CTRL.
5. Look up the status of the hard disk.
6. Configure the data register and set the da_pad_o (DA) signal to 3'h0.
7. Transfer data to the hard disk through the data register.

DMA Process Module

The DMA process module is used to implement the multi-word DMA mode. Using DMA transfer mode, the controller can access the external memory directly and build a bridge between the external memory and the hard disk. All data transfer is 16 bits. This data transfer type has three data transfer speeds and timing parameters. Table 15 describes the timing parameters for different transfer speeds.

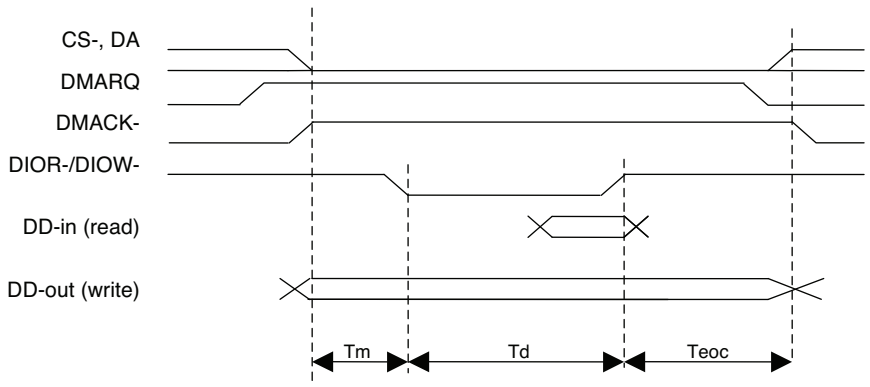
Table 15. DMA Timing Parameters

PIO Transfer Mode	Cycle Time (ns)	DMA Mode		
		Tm (ns)	Td (ns)	Teoc (ns)
0	480	50	50	215
1	150	30	80	40
2	120	25	70	25

According to the selected hard disk speed, the host configures the different timing parameters to the timing registers of this controller.

The DMA transfer mode timing is shown in Figure 5.

Figure 5. DMA Transfer Mode Timing



Tm, Td and Teoc have definitions similar to the PIO timing parameters.

DMARQ is used for DMA data transfers between the host and the device, when the device is ready to transfer data to or from the host. For multi-word DMA transfers, DIOR- and DIOW- control the direction of data. This signal is used in a handshake manner with DMACK-. The device waits until the host asserts DMACK- before negating DMARQ and re-asserting DMARQ if there is more data to transfer. The host uses DMACK- in response to DMARQ to initiate DMA transfers.

For the DMA transfer mode, the device terminates the cycle by negating DMARQ; the host terminates the cycle by negating DMACK-.

For the DMA transfer process, this module includes one Tx buffer and one Rx FIFO. The Tx buffer is used to store data to be transmitted for writing to the hard disk. Rx FIFO is used to store the received data from the hard disk in the read operation.

All the DMA transmit buffers and the Rx FIFO are 32 bits wide to reduce the cost of reading and writing. Therefore, each DMA transfer must be a multiple of 32 bits (i.e. the number of cycles per DMA transfer must be a multiple of two). The core is designed to work with an external DMA engine. When the DMARQ flag is set, the ATA-DMARQ line is asserted, indicating the ATA devices request a DMA transfer. By checking the DMA Receive Buffer Empty [DRBE] and the DMA Transmit Buffer Full [DTBF] in the Status register, the software or the host can decide whether to read from or write to the DMA buffers.

If DMA mode is used as the data transfer mode, the operation flow is as follows:

1. Configure the Command register set in the hard disk.
2. Set the Cs0 signal and Cs1 signal to ‘11’.
3. Set the DMAen bit and the DMAdir bit by configuring the CTRL register.
4. Wait for the validation of the DMARQ line.
5. Initiate the DMA transfer road according to the data transfer direction.
6. Program the DMA timing registers.
7. Transfer data between the external memory and the hard disk.

The core starts a DMA write cycle when the DMAdir bit is set (‘1’) and the DMACK line is asserted. The core asserts the DMACK- signal in response to the DMARQ line as soon as there is data in the DMA Transmit Buffer. When there is not enough data in the DMA Transmit Buffer, the controller postpones the DMA cycle and asserts the WISHBONE DMA_req signal. When no new data has been written into the DMA Transmit Buffer before the end of the current cycle, the DMACK- signal is negated.

When new data has been written into the DMA Transmit Buffer, the core extends the DMA sequence and continues with a new cycle (multi-word DMA transfer). Since the controller has no knowledge about how much data should be transferred, it is the host’s responsibility to keep track of this. Either by programming the external DMA engine to transfer the required amount of samples when an external DMA engine is used, or by counting the number of transfers when using pseudo-DMA transfers.

The core starts a DMA read cycle when the DMAdir bit is cleared (‘0’) and the DMACK- line is asserted. The core asserts DMACK- in response to the DMARQ line when the DMA Receive Buffers are not full. The core asserts the WISHBONE DMA_req signal as soon as data is available in the DMA receive buffers. When the buffers become full the core negates the DMACK- line, until the host empties the receive buffers by reading from the DMA buffer address. If the ATA devices still have the DMARQ line asserted, the core asserts the DMACK- line again and continues the DMA transfer.

Test Bench Description

The test bench for this design includes the following modules. The functions of each module are described below.

Test_bench_top.v and Tests.v

The top of the test bench generates the clock and reset signals and also includes five different test vectors.

1. **Io_test1** – Tests read/write of registers, includes the user registers of the controller and the internal registers of the IDE hard disk.
2. **Io_test2** – Tests the different PIO transfer modes.
3. **Int_test** – Tests the initialization procedure.
4. **Rst_test** – Tests the software reset function.
5. **DMA_test** – Tests the DMA read and write functions.

To test DMA mode transfer, a DMA slave engine must work with the host controller to complete all DMA operations. To simplify the design of the DMA engine, this test bench uses a task to emulate the slave Pseudo-DMA engine function. The task includes two test vectors: writing data to the hard disk and reading data from the hard disk. If a write request of the pseudo-DMA is received, it can read data from one register memory and transfer data to the Tx buffer. After one sector of data is transferred, this simple pseudo-DMA engine responds to the controller with one acknowledge signal. If a read request is received, it can read data from the controller and transfer it to the external memory. After one sector of data is read, the pseudo-DMA engine responds to the acknowledge signal. This simple task is implemented in the test.v module.

Ata_device.v

This module is a simple IDE/ATA device simulation model. It emulates the behavior of the hard disk.

Wb_mast_model.v and Wb_model_define.v

These two files emulate CPU behavior and implement all WISHBONE reading and writing tasks.

HDL Simulation and Verification

Figure 6 is the RTL simulation waveform of the PIO transfer mode.

Figure 6. PIO Transfer Mode

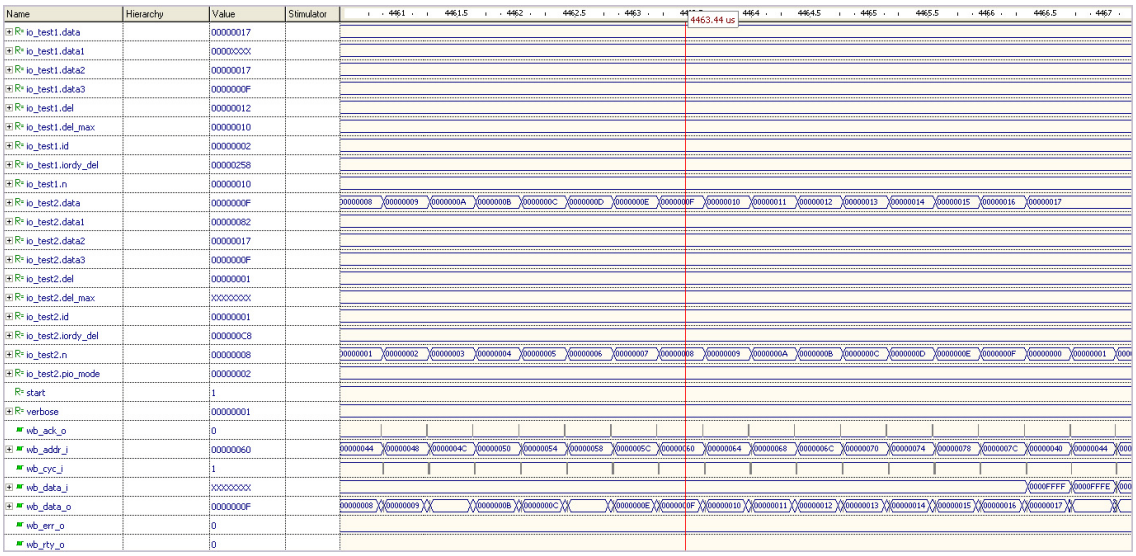
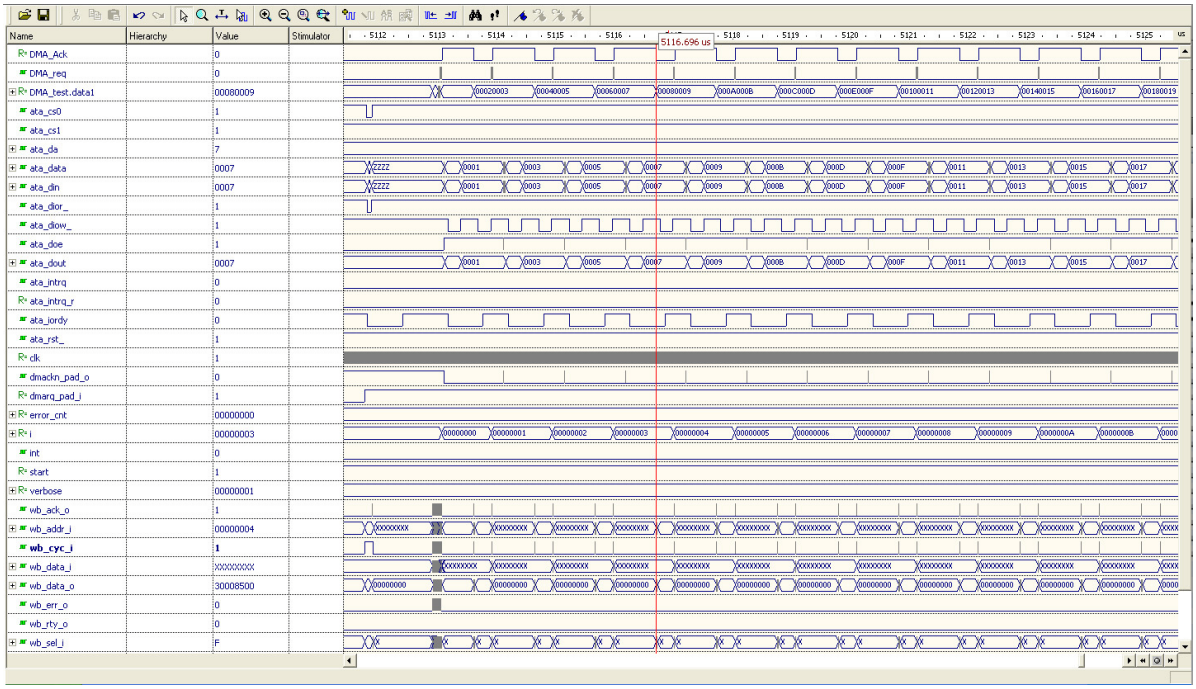


Figure 7 is the RTL simulation waveform of the DMA transfer mode.

Figure 7. DMA Transfer Mode



Implementation

Table 16. Performance and Resource Utilization

Device Family	Speed Grade	Utilization (LUTs)	f_{MAX} (MHz)	I/Os	Architecture Resources
MachXO™ [1]	-5	752	>100	194	N/A

1. Performance and utilization characteristics are generated using LCMXO2280C-5FT324C, with ispLEVER® 8.1 software. When using this design in a different device, density, speed, or grade, performance and utilization may vary.

References

- Information Technology - AT Attachment with packet Interface -5(ATA/ATAPI-5)
- WISHBONE System-on-Chip Interconnection Architecture for Portable IP Cores
- ATA/ATAPI-5 Core from OpenCores (Author: Richard Herveille)

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
June 2010	01.0	Initial release.