ATA-3 IP-Core

OCIDEC (OpenCores IDE Controller)

Specification

Author: Richard Herveille
RHerveille@OpenCores.org

Rev. 0.1
April 10, 2001

Revision History

Rev.	Date	Author	Description
0.1		Richard Herveille	First Draft

Contents

1

Introduction

The OCIDEC (OpenCores IDE Controller) is a WISHBONE rev.B2 compliant ATA/ATAPI-5 host implementation. The ATA (AT Attachment) interface, also known as IDE (Integrated Drive Electronics) interface, provides a simple interface to low cost non-volatile memories like hard-disk drives, DVD players, CDROM players/writers, CompactFlash and PC-Card devices.

Three different versions of the core are currently available. All versions are backward software and function compatible. Software can detect which version is implemented by reading the Device-ID and Revision Number from the status register, thus making it possible to handle all cores with a single device driver. This gives a designer/system integrator the ability to trade off complexity/resource usage to performance/feature set. See table below for feature specific to each core.

	OCIDEC-1	OCIDEC-2	OCIDEC-3
Features	Smallest core PIO compatible timing only	Small core PIO transfer support only PIO dataport fast timing per device	PIO and DMA transfer support PIO settings per device DMA settings per device
Intended use	Single CompactFlash/PCCard systems System requiring simple ATA capabilities	Dual CompactFlash/PCCard systems System requiring Fast ATA capabilities	Harddisk/CDROM interface System requiring full ATA capabilities
Gate count	Approx. 4Kgates	Approx.4.6Kgates	Approx.14Kgates

Features:

- **Software and Function backwards compatible cores**
- **Common PIO compatible timing settings for all connected devices**
- **Fast PIO dataport timing settings per connected device**
- **Singleword/Multiword timing settings per connected device**
- **PIO write PingPong enhancement**
- **Automatic Big endian versus Little endian conversion**
- **DMA read/write buffers**
- **WISHBONE rev.B2 compliant**
- **OpenCores WISHBONE DMA engine compatible**

2

IO ports & Parameters

2.1 Core Parameters

Parameter	Default	Description	Device
TWIDTH	8	Internal counter width	ALL
PIO_mode0_T1	6	PIO mode0 Address valid to DIOR-/DIOW- setup	ALL
PIO_mode0_T2	28	PIO mode0 DIOR-/DIOW- pulse width	ALL
PIO_mode0_T4	2	PIO mode0 DIOW- data hold	ALL
PIO_mode0_Teoc	23	PIO mode0 end of cycle time	ALL
DMA_mode0_Tm	4	DMA mode0 CS(1:0) valid to DIOR-/DIOW-	OCIDEC3
PIO_mode0_Td	21	DMA mode0 DIOR-/DIOW- asserted time	OCIDEC3
PIO_mode0_Teoc	21	DMA mode0 end of cycle time	OCIDEC3

TWIDTH
The number of bits the internal counters use. When TWIDTH = 8 the core will use 8bit wide counters. For slow clock inputs this number can be reduced. For a 100MHz clock input, for example, TWIDTH can be set to 5.

The procedure for calculating TWIDTH starts with calculating the initial mode0 timing parameters. These values are the largest numbers the counters need to contain. The next step is to take the log2 from the largest value and round the result to the next highest integer. At 100MHz the value is 28, for PIO_mode0_T2. Log2(28) = 4.8, rounding this to the next highest integer result in 5.

TIMINGS
See appendix A ‘ATA Timings’ for a detailed description of all timing parameters.

2.2 WISHBONE interconnect signals

Port	Width	Direction	Description	Device
CLK_I	1	Input	Master clock input	ALL
RST_I	1	Input	Synchronous active high reset	ALL
NRESET	1	Input	Asynchronous active low reset	ALL
ADR_I	5	Input	Lower address bits	ALL
DAT_I	32	Input	Data towards the core	ALL

DAT_O	32	Output	Data from the core	ALL
SEL_I	4	Input	Byte select signals	ALL
WE_I	1	Input	Write enable input	ALL
STB_I	1	Input	Strobe signal/Core select input	ALL
CYC_I	1	Input	Valid bus cycle input	ALL
ACK_O	1	Output	Bus cycle acknowledge output	ALL
RTY_O	1	Output	Bus cycle retry output	ALL
ERR_O	1	Output	Bus cycle error output	ALL
INTA_O	1	Output	Interrupt request signal output	ALL
DMA_req	1	Output	DMA request to external DMA engine	OCIDEC3
DMA_ack	1	Input	DMA acknowledge from external DMA engine	OCIDEC3

2.2.1 CLK_I
All internal logic is registered to the rising edge of the [CLK_I] clock input. The frequency range over which the core can operate depends on the technology used and the transfer modes which need to be supported. The implementation dependant upper frequency limited is caused by the 8bit wide internal counters. They limit the maximum frequency to 880MHz (T2=255), see internal registers for more information. The lower frequency is limited by the PIO transfer bandwidth to approximately 10MHz.

2.2.2 RST_I
The active high synchronous reset input [RST_I] forces the core to restart. All internal registers are preset and all state-machines are set to an initial state.

2.2.3 nReset:
The active low asynchronous reset input [nRESET] forces the core to restart. All internal registers are preset and all state-machines are set to an initial state.

nReset is not a WISHBONE compatible signal. It is provided for FPGA implementations. Since most FPGAs provide a dedicated reset path using [nRESET] instead of [RST_I] can result in lower cell-usage and higher performance. Either use [nRESET] or [RST_I].

2.2.4 ADR_I
The address array input [ADR_I] is used to pass a binary coded address to the core. The most significant bit is at the higher number of the array.

2.2.5 DAT_I
The data array input [DAT_I] is used to pass binary data from the MASTER to the core. All data transfers are 32bit wide, except for PIO transfers, which are 8 or 16bit wide.

2.2.6 DAT_O
The data array output [DAT_O] is used to pass binary data from the core to the MASTER. All data transfers are 32bit wide, except for PIO transfer accesses, which are 8 or 16bit wide.

2.2.7 SEL_I
The byte select array input [SEL_I] indicates where valid data is placed on the [DAT_I] input array during writes to the core, and where it is expected on the [DAT_O] output array during reads from the core. The cores require all accesses to be 32bit wide [SEL_I(3:0)] = '1111'b, except for PIO transfer accesses where 16bit wide accesses are allowed [SEL_I(1:0)] = '11'b.

2.2.8 WE_I
The write enable input [WE_I], when asserted indicates whether the current bus cycle is a read or write cycle. The signal is asserted during write cycles and negated during read cycles.

2.2.9 STB_I
The strobe input [STB_I] is asserted when the core is being addresses. The core only responds to WISHBONE signals when [STB_I] is asserted, except for [RST_I] and [nRESET] reset signals, which are always responded to.

2.2.10 CYC_I
The cycle input [CYC_I], when asserted indicates that a valid bus cycle is in progress. The logical AND function of [STB_I] and [CYC_I] indicates a valid transfer cycle to/from the core.

2.2.11 ACK_O
The acknowledge output [ACK_O], when asserted indicates the normal termination of a valid bus cycle.

2.2.12 RTY_O
The retry output [RTY_O], when asserted indicates the termination of a valid bus cycle without the completion of the cycle. The OCIDEC-3 controller asserts RTY_O when the host issues a PIO transfer access, but the controller is busy. The controller is busy when either DMAtip or PWPPF is set in the status register. If the MASTER interface does support retry bus cycles it should back off and retry the access at a later moment. If the MASTER interface does not support retry bus cycles it can ignore the RTY_O signal. The controller will handle the request when it has completed the current transfer(s) and complete the bus cycle normally.

2.2.13 ERR_O
The error output [ERR_O], when asserted indicates an abnormal termination of a bus cycle. The [ERR_O] output signal is asserted when:

1) The host tries to access the controller's internal registers using non 32bit aligned data; i.e. SEL_I(3:0) not equal to 1111b.
2) The host issues a PIO transfer using byte-wide or non 32bit aligned data; i.e. SEL_I(1:0) not equal to 11b.
3) The host tries to access a DMA buffer using non 32bit aligned data; i.e. SEL_I(3:0) not equal to 1111b (OCIDEC-3 only).

2.2.14 INTA_O
The interrupt request output [INTA_O], when asserted indicates that a connected device needs servicing. The [INTA_O] output is a relay from the [INTRQ] ATA interface signal.

2.2.15 DMA_req
The DMA request output [DMA_req], when asserted indicates that the controller wants to transfer data between the MASTER and the core via DMA. The [DMA_req] signal is asserted when the controller's DMA Transmit Buffer is empty during a DMA write transfer, or when the controller's DMA Receive Buffer Contains data. This signal should be connected to an external DMA engine, like the OpenCores WISHBONE DMA core.

2.2.16 DMA_ack
The DMA acknowledge input [DMA_ack], when asserted indicates the termination of the current DMA cycle.

2.3 ATA signals

Port	Width	Direction	Description	Device
RESETn	1	Output	IDE hardware reset	ALL
DDi	16	Input	Device Data (from ATA devices)	ALL
DDo	16	Output	Device Data (towards ATA devices)	ALL
DDoe	1	Output	DD output enable	ALL
DA	3	Output	Device Address	ALL
CS0n	1	Output	Chip Select0	ALL
CS1n	1	Output	Chip Select1	ALL
DMARQ	1	Input	DMA request	OCIDEC3
DMACKn	1	Output	DMA acknowledge	OCIDEC3
DIORn	1	Output	Device IO read	ALL
DIOWn	1	Output	Device IO write	ALL
IORDY	1	Input	IO channel ready	ALL
INTRQ	1	Input	Device interrupt	ALL

The following section describes the ATA interface signals as described in the ATA specifications. When there is a difference in nomenclature between the ATA specs and the core signals, both are given.

2.3.1 CS(1:0)- [CS(1:0)n]
These are the chip select signals from the host used to select the Command Block or Control Block registers. When DMACK- is asserted, CS0- and CS1- shall be negated and the transfers shall be 16bit wide.

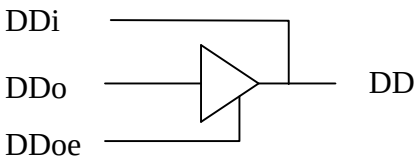
2.3.2 DA(2:0)

This is the 3bit binary coded address asserted by the host to access a register or data port on the device.

2.3.3 DD(15:0)

This is an 8 or 16bit bi-directional data interface between the host and the device. The lower 8bits are used for 8bit register transfers. Data transfers are 16bits wide except for CompactFlash devices that implement 8bit data transfers

The DDi, DDo and DDoe must be connected to external tri-state buffers to form the DD signals, as shown below.



This can be accomplished using the following VHDL code:

```
DD  <= DDo when (DDoe = '1') else (others => 'Z');
DDi <= DD;
```

2.3.4 DIOR- [DIORn]

DIOR- is the strobe signal asserted by the host to read device registers or the data port.

2.3.5 DIOW- [DIOWn]

DIOW- is the strobe signal asserted by the host to write device registers or the data port.

2.3.6 DMACK- [DMACKn]

The host shall use this signal in response to DMARQ to initiate DMA transfers.

OCIDEC-1 and OCIDEC-2 do not support DMA transfers. This signal, however, is connected on ATA devices that do support DMA transfers. This signal must be pulled-up.

2.3.7 DMARQ

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

OCIDEC-1 and OCIDEC-2 do not support DMA transfers. This signal, however, is connected on ATA devices that do support DMA transfers. In this case, this signal must be left unconnected.

2.3.8 INTRQ

The selected device uses this signal to interrupt the host system when interrupt pending is set.

2.3.9 IORDY

This signal is used to extend the host transfer cycle of any host register access (either read or write) when the device is not ready to respond to a data transfer request.

For PIO mode 3 and above, IORDY shall always be used.

2.3.10 RESET- [RESETn]

The host uses this signal, referred to as hardware reset, to reset the connected devices.

3

Registers

3.1 Core Registers list

Name	Address	Width	Access	Description	Device
CTRL	0x00	31	R/W	Control register	ALL
STAT	0x01	31	R/W	Status register	ALL
PCTR	0x02	31	R/W	PIO compatible timing register	ALL
PFTR0	0x03	31	R/W	PIO fast timing register device 0	OCIDEC2
PFTR1	0x04	31	R/W	PIO fast timing register device 1	OCIDEC2
DTR0	0x05	31	R/W	DMA timing register device 0	OCIDEC3
DTR1	0x06	31	R/W	DMA timing register device 1	OCIDEC3
DTxDB	0x0F	31	W	DMA Transmit Data Buffer	OCIDEC3
DRxDB	0x0F	31	R	DMA Receive Data Buffer	OCIDEC3

3.2 Control Register [CTRL]

Bit #	Access	Description	Device
31:16	R/W	*Reserved*	
15	R/W	DMAen, DMA enable	OCIDEC3
14	R/W	*Reserved*	
13	R/W	DMAdir, DMA direction	OCIDEC3
12:10	R/W	*Reserved*	
9	R/W	BeLeC1, Big Endian Little Endian conversion device1	OCIDEC3
8	R/W	BeLeC0, Big Endian Little Endian conversion device0	OCIDEC3
7	R/W	IDEen, IDE enable	ALL
6	R/W	FTE1, Fast Timing device1 enable	OCIDEC2
5	R/W	FTE0, Fast Timing device0 enable	OCIDEC2
4	R/W	PWPP, PIO write Ping-Pong enable	OCIDEC3
3	R/W	IORDYen_ft1, Fast Timing device1 IORDY enable	OCIDEC2
2	R/W	IORDYen_ft0, Fast Timing device0 IORDY enable	OCIDEC2
1	R/W	IORDYen_ct, Compatible timing IORDY enable	ALL
0	R/W	ARST, ATA Reset	ALL

Reset Value: 0x0001

3.2.1 DMA

The DMA enable bit should be set (‘1’) if and only if the DMA timing registers for both devices are programmed. The controller responds to the ATA device’s DMARQ line only when this bit is set.

3.2.2 DMAdir
The DMA direction bit set the direction of data transfer during DMA transfers. When set (‘1’) the direction of data transfer is from host to device, i.e. writing to device. When cleared (‘0’) the direction of data transfer is from device to host, i.e. reading from device. Do not change this bit when DMA transfer is in progress.

3.2.3 BeLeC
When set (‘1’) Big endian versus Little endian conversion during DMA transfers is enabled for the selected device.

3.2.4 IDEen
The IDE enable bit should be set (‘1’) after the initial reset procedure (see power-on and hardware protocol). The controller responds to PIO transfer request only when this bit is set.

3.2.5 FTE
The fast timing bits should be set (‘1’) if the device supports fast timing and if Fast Timing Registers are programmed.

3.2.6 PWPP
The PIO Write PingPong bit should always be set (‘1’). It enables a performance enhancement feature for PIO transfers. Clearing this bit provides the same PIO functionality and performance as the OCIDEC-2 device.

3.2.7 IORDY
The IORDY bits are:
1) Set (‘1’) depending on the current mode and capabilities of the drive, according to the device’s capabilities for PIO Mode 2.
2) Always set for PIO Modes 3 and above.

3.2.8 ARST
The ATA reset bit controls the RESET- line status. When set (‘1’) the RESETn line is asserted (low level) and all connected ATA devices are reset and the OCIDEC-3’s DMA buffers are flushed. This bit is set during a core reset (either [RST_I] or [nRESET] asserted), effectively resetting the connected device when the core is reset.

3.3 Status register [STAT]

Bit #	Access	Description	Device
31:28	R	Device ID	ALL
27:24	R	Revision Number	ALL
23:16	R	*Reserved*	
15	R	DMAtip, DMA transfer in progress	OCIDEC3
14:10	R	*Reserved*	
10	R	DRBE, DMA receive buffer empty	OCIDEC3

9	R	DTBF, DMA transmit buffer full	OCIDEC3
8	R	DMARQ, DMARQ line status	OCIDEC3
7	R	PIOtip, Programmed IO transfer in progress	ALL
6	R	PWPPF, PIO write Ping-Pong full	
5:1	R	*Reserved*	
0	R/W	IDEIS, IDE Interrupt status When set to '1', indicates that a device asserted its interrupt line.	ALL

3.3.1 Device ID
The Device ID contains the number of the implemented OCIDEC core.
OCIDEC-1, Device ID = 0x01
OCIDEC-2, Device ID = 0x02
OCIDEC-3, Device ID = 0x03
Software can use this number to install the correct device driver for the implemented core.

3.3.2 Revision Number
The Revision Number for all core types is currently 0x00.

3.3.3 DMAtip
The DMA Transfer In Progress flag is set ('1') when the core is currently performing a DMA transfer.

3.3.4 DRBE
The DMA Receive Buffer Empty flag is set ('1') when the receive fifo is empty.

3.3.5 DTBF
The DMA Transmit Buffer Full flag is set ('1') when the transmit buffer is full.

3.3.6 DMARQ
The DMARQ flag reflects the status of the ATA-DMARQ line.

3.3.7 PIOtip
The Programmed IO Transfer In Progress flag is set ('1') when the core is currently performing a PIO transfer.

3.3.8 PWPPF
The PIO Write PingPong Full flag is set ('1') when the PIO write pingpong system is full.

3.3.9 IDEIS
The IDE Interrupt Status flag reflects the ATA-INTRQ line. Software must clear this bit by writing a '0' to it. If the Interrupt Status Bit is cleared by writing a '0' to it, while the interrupt line is still asserted, this bit remains '0' until a new edge is detected on the interrupt line.

3.4 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-

Reset value: All registers are filled with the PIO_mode0 timings, described in the 'Core Parameters' section.

This is the slowest mode all connected devices can handle. OCIDEC-1 uses this mode for all PIO transfer accesses to the connected devices. OCIDEC-2 and above use this mode for all accesses to the connected devices, except for data-register accesses.

The ATA specs refer to these settings as 'Register Transfer Timing Parameters'.
See Appendix A 'ATA Timings' for a detailed description of all timing parameters.

3.5 PIO Fast Timing Register Device 0 [PFTR0]
PIO Fast Timing Register 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-

Reset value: All registers are filled with the PIO_mode0 timings, described in the 'Core Parameters' section.

OCIDEC-2 and above cores use these timing settings for PIO transfer accesses to the ATA device's data-register. The core internally keeps track of the DEV bit, set in the 'Device/Head' register. When the DEV bit is set ('1'), device1 is selected. The core will use the PFTR1 timing settings to access the selected device. When the DEV bit is cleared ('0'), device0 is selected. The core will use the PFTR0 timing settings to access the selected device.

The ATA specs refer to these settings as 'PIO Data Transfer Timing Parameters'.
See Appendix A 'ATA Timings' for a detailed description of all timing parameters.

3.6 DMA Timing Register Device 0 [DTR0]
DMA Timing Register 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, CS(1:0) valid to DIOR-/DIOW-

Reset value: All registers are filled with the DMA_mode0 timings, described in the 'Core Parameters' section.

OCIDEC-3 cores use these timing settings for all Single- and Multiword DMA transfer accesses to the ATA device's data-port. The core internally keeps track of the DEV bit, set in the 'Device/Head' register. When the DEV bit is set ('1'), device1 is selected. The core will use the DTR1 timing settings to access the selected device. When the DEV bit is cleared ('0'), device0 is selected. The core will use the DTR0 timing settings to access the selected device.

The ATA specs refer to these settings as 'Multiword DMA Data Transfer Timing Parameters'. See Appendix A 'ATA Timings' for a detailed description of all timing parameters.

3.7 DMA Transmit Data Buffer [DTxDB]

Bit #	Access	Description
31:16	W	Transmit Data1(15:8)
23:16	W	Transmit Data(7:0)
15:8	W	Transmit Data2(15:8)
7:0	W	Transmit Data2(7:0)

Reset value: 0x00

The DMA Transmit Data Buffer contains the data to be written to the ATA devices using DMA transfer accesses. The core will wait until this register has been filled, before starting a DMA write transfer. The core expands the 32bit DMA Transmit Data into two consecutive 16bit DMA transfers. This implies that all DMA transfers must be a multiple of 32bit, i.e. the number of cycles per DMA transfer must be a multiple of two.

The core can perform automatic Big Endian versus Little Endian conversions. When the BeLeC bit is set ('1') for the selected device, the core performs endian conversion. It transfers Data2 first and then Data1, with the MSB and LSB swapped.
First DMA cycle: Data2(7:0) => DD(15:8), Data2(15:8) => DD(7:0)
Second DMA cycle: Data1(7:0) => DD(15:8), Data1(15:8) => DD(7:0)
When the BeLeC bit is cleared ('0') for the selected device, the core does not perform endian conversion. It transfers Data1 first and then Data2, without the MSB and LSB swapped.
First DMA cycle: Data1(15:8) => DD(15:8), Data1(7:0) => DD(7:0)
Second DMA cycle: Data2(15:8) => DD(15:8), Data2(7:0) => DD(7:0)

When the host writes to the buffer while it is full (DMATxFull = ‘1’), the core acknowledges the cycle, but discards the new data.

The buffer is flushed when the ARST bit is set (‘1’).

3.8 DMA Receive Data Buffer

The DMA Receive Buffer is a 7double-word deep FIFO. It contains the data read from the ATA devices during a DMA read transfer. The core combines two consecutive 16bit DMA read transfers into a 32bit double-word and stores this in the DMA Receive Buffer. This implies that all DMA transfers must be a multiple of 32bit, i.e. the number of cycles per DMA transfer must be a multiple of two. The core will continue transferring data until the FIFO is full, or the ATA devices release the DMARQ line.

The core can perform automatic Big Endian versus Little Endian conversions. When the BeLeC bit is set (‘1’) for the selected device, the core performs endian conversion. It stores the first received word in the lower word of the receive buffer and the next received word in the upper word of the receive buffer, with the MSB and LSB swapped.

DMA Transmit Data Buffer contents, BeLeC = ‘1’

Bit #	Access	Description
31:16	R	Second received word, DD(7:0)
23:16	R	Second received word, DD(15:8)
15:8	R	First received word, DD(7:0)
7:0	R	First received word, DD(15:8)

Reset value: 0x00

When the BeLeC bit is cleared (‘0’) for the selected device, the core does not perform endian conversion. It stores the first received word in the upper word of the receive buffer and the next received in the lower word of the receive buffer, without swapping the MSB and the LSB.

DMA Transmit Data Buffer contents, BeLeC = ‘0’

Bit #	Access	Description
31:16	R	First received word, DD(15:8)
23:16	R	First received word, DD(7:0)
15:8	R	Second received word, DD(15:8)
7:0	R	Second received word, DD(7:0)

Reset value: 0x00

When the host reads from the buffer while it is empty (DMATRxEmpty = ‘1’), the core acknowledges the cycle, but the data read is invalid.

The buffer is flushed when the ARST bit is set (‘1’).

3.9 ATA IO Registers list

Name	Address	Width	Access
Alternate Status Register	0x1E	8	R
Command Register	0x17	8	W
Cylinder High Register	0x15	8	R/W
Cylinder Low Register	0x14	8	R/W
Data Register	0x10	16	R/W
Device Control Register	0x1E	8	W
Device/Head Register	0x16	8	R/W
Error Register	0x11	8	R
Features Register	0x11	8	W
Sector Count Register	0x12	8	R/W
Sector Number Register	0x13	8	R/W
Status Register	0x17	8	R

The ATA IO registers are all accessed using PIO transfers. When an access is made to an 8bit register, the data is expected on DD_I(7:0) for a write access and presented on DD_O(7:0) for a read access. When an access is made to a 16 bit register, the data is expected on DD_I(15:0) for a write access and presented on DD_O(15:0) for a read access.

The ATA IO Registers are addressed using the CS0-, CS1- and DA(2:0) lines. These lines are mapped into the core's address range, making them transparent for any software wanting to access them. The registers are mapped into the 0x10 to 0x1F address range, according to the following scheme.

CS0- <= ADR_I(3)
CS1- <= not ADR_I(3)
DA(2:0) <= ADR_I(2:0)

CS(1:0)- reflect the ADR_(3) signal state. CS0- is asserted (low level) when ADR_I(3) is negated ('0'). CS1- is asserted (low level) when ADR_I(3) is asserted ('1'). DA(2:0) reflect the ADR_I(2:0) state.

Following is a short description of each register, taken from the ATA/ATAPI specs. See the ATA/ATAPI specifications for more information about these registers.

3.9.1 Alternate Status Register

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

3.9.2 Command Register

The Command register contains the command code being sent to the device.Command execution begins immediately after this register is written. The contents of the Command Block registers become parameters of the command when this register is written. Writing this register clears any pending interrupt condition.

3.9.3 Cylinder High Register
The Cylinder High Register content is command dependent.

3.9.4 Cylinder Low Register
The Cylinder Low Register becomes a command parameter when the Command register is written.

3.9.5 Data Register
PIO data transfers are processed by a series of reads or writes to the Data Register.

3.9.6 Device Control Register
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.

3.9.7 Device/Head Register
The Device/Head Register contains the selected drive bit [DEV]. OCIDEC-2 and above cores keep track of the contents of this register to select the required timing settings.

3.9.8 Error Register
The Error Register contains the status for the current command.

3.9.9 Features Register
The Features Register content is command dependant.

3.9.10 Sector Count Register
The Sector Count Register content is command dependant.

3.9.11 Sector Number Register
The Sector Number Register content is command dependant.

3.9.12 Status Register
The Status Register contains the device status. The register's contents are updated to reflect the current state of the device and the progress of any command being executed by the device. Reading the Status Register clears any pending interrupt. The host should not read the Status Register when an interrupt is expected as this may clear the interrupt pending before the INTRQ can be recognized.

4

Operation

4.1 Power-on and hardware reset protocol

The RESET- signal is controlled by the [ARST] bit in the Control Register [CTRL]. When the [ARST] bit is set (‘1’), the RESET- line is asserted (low level). When the [ARST] bit is cleared (‘0’), the RESET- line is negated (high level). The host can set and clear this bit by writing to the Control Register. It is also set after a core reset caused by the assertion of [RST_I] or [nRESET]. When the RESET- signal is asserted, the connected devices execute the hardware reset protocol. The host should respond as described below.

1) Assert RESET- for at least 25us
2) Negate RESET- and wait at least 2ms
3) Read the ATA Status Register or the Alternate Status Register
4) Wait for the busy flag [BSY] to be cleared.
5) Perform an IDENTIFY DEVICE or IDENTIFY PACKET DEVICE command for each connected device.
6) Read the device parameters from each connected device.
7) Program the core’s timing registers depending on the data read from the device(s).

4.2 PIO Transfer Access

A write to or a read from an address in the 0x10 to 0x1F range initiates a PIO write- or read transfer respectively. The PIO registers of the devices are mapped into this range. When the core detects a read or write access in this address range, it executes a PIO Transfer cycle as shown in figure 4.1.

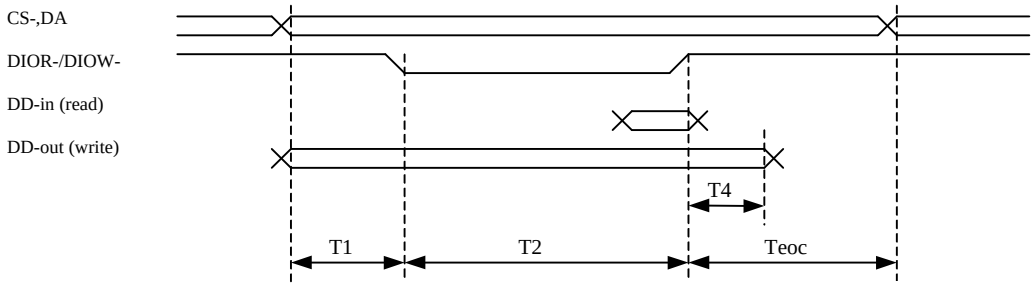


Figure 4.1 PIO Transfer Cycle

The cores use the values entered in the Compatible Timing Register for all accesses to the ATA devices, except for OCIDEC-2 and above cores, which use the Fast Timing Register for Data Register accesses.

4.3 PIO Fast Data Register Transfers

A Fast Data Register Transfer is a read or write access to the ATA Data Register, located at address 0x10. The access cycle is identical to any other PIO transfer access, as shown in figure 4.1. The only difference is that the cycle time is shorter, allowing a higher data-rate when compared to the Compatible Timing.

OCIDEC-2 and above cores support this mode. These cores have a Fast Timing Register for each connected device. When enabled, by setting the appropriate Fast Timing Enable bit (FATEx), the cores will use the Fast Timing Registers' settings to access the data register. The cores select the timing settings depending on the currently select device. The cores know which device is selected by tracking write accesses to the ATA Device/Head Register, located at address 0x16. During a write access to this register the cores internally store the value of the DEV bit. Since any combination of devices can be attached to the ATA interface (no devices, only device0, only device1, device0 & device1), it is impossible for the cores to know the setup. It is therefore important to perform at least one write to the Device/Head register to correctly set the internal DEV bit.

4.4 PIO Write PingPong

The OCIDEC-3 core features a PIO write transfer access enhancement, called PIO Write PingPong. The basic idea is that it is not necessary to keep the host bus busy, during a PIO write transfer, until the data is transferred to the ATA device. Instead the data and address is stored internally, the host bus is released and the ATA write transfer is started. The OCIDEC-3 core can store two consecutive write accesses. When a third write access is initiated before the first has completed, or when a read access is initiated the host bus is stalled, until the first write access or the read access completes.

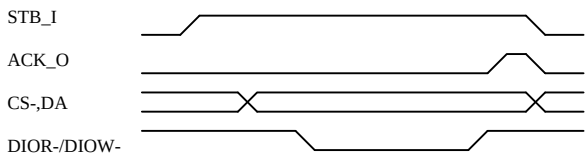


Figure 4.2 PIO transfer without pingpong enabled

Figure 4.2 shows the bus cycle for a normal PIO transfer as executed by OCIDEC-1 and OCIDEC-2 cores. The host bus is released when the data has been read from/written to the ATA device.

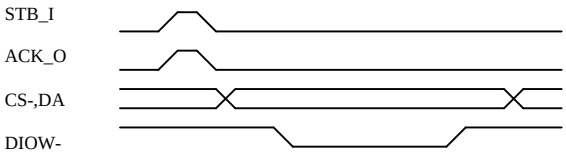


Figure 4.3 PIO write transfer with pingpong enabled

Figure 4.3 shows the bus cycle for a PIO write transfer with PIO Write PingPong enabled. The host bus is released as soon as the data and address have been stored internally. The write cycle completes normally.

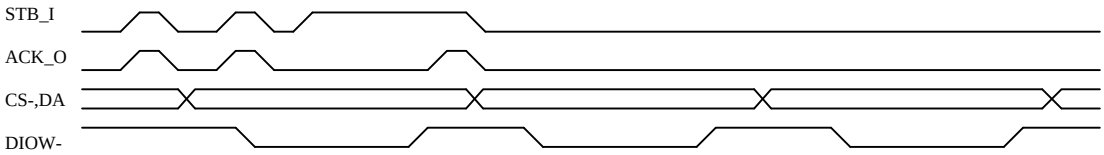


Figure 4.4 PIO multiple write transfers with pingpong enabled

Figure 4.4 shows how multiple PIO write transfers are affected by the pingpong system. Because the pingpong system can store two consecutive write cycles, the host bus is released as soon as the data and address of the first two cycles are stored internally. During the third cycle, the host bus is released as soon as there is place available in the pingpong system.

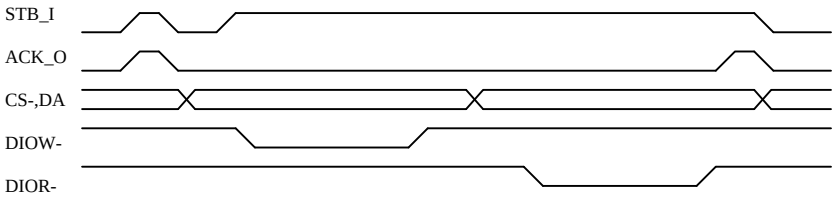


Figure 4.5 PIO write and read transfers with pingpong enabled

Figure 4.5 shows how a read transfer is affected by the pingpong system. If there is room in the pingpong system, a read transfer is stored internally and the host bus is released. During a read transfer the host bus needs to be hold until the data has been read from the ATA device. The core has to wait until the write transfer completes before the read transfer can be initiated. This could result in the host bus being hold for a long time, especially if two write cycles are stored in the pingpong system. To avoid this, software can read the PIO Write PingPong Full flag (PWPPF) in the status register before initiating a write or read transfer.

The core notifies the WISHBONE MASTER when a PIO transfer access is requested while the pingpong system is full, by asserting the bus retry output [RTY_O]. If the MASTER supports bus retry cycles, it should back-off and retry the cycle later. If the MASTER does not support retry cycles the [RTY_O] output can be ignored, and the software should check whether the pingpong system is full or not.

4.5 DMA Transfer Access

The ATA device initiates a DMA transfer by asserting the DMARQ line. It does so in response to READ DMA, WRITE DMA, READ DMA QEUED, WRITE DMA QEUED and PACKET commands. When the DMA Timing Registers are programmed, and the DMA enable [DMAen] bit is set (‘1’), the core responds to the assertion of DMARQ by starting a DMA transfer cycle as shown in figure 4.6. Either the device or the host can terminate the transfer cycle. The device terminates the cycle by negating DMARQ, the host terminates the cycle by negating DMACK-.

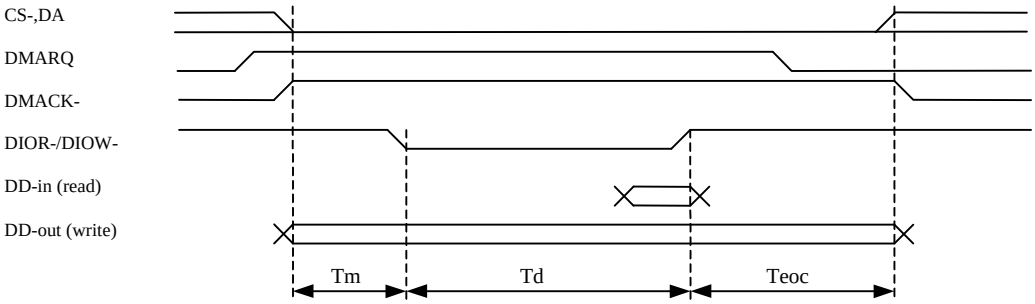


Figure 4.6 DMA Transfer Cycle, terminated by device

The direction of the data transfer is controlled by the command issued to the ATA devices and the DMA Direction [DMAdir] bit in the control register. When set (‘1’) the core’s response is a DMA write cycle, when cleared (‘0’) the core’s response is a DMA read cycle. Setting the DMAdir bit to write (‘1’) while a READ DMA (QEUED) command is issued, or setting the DMAdir bit to read (‘0’) while a WRITE DMA (QEUED) command is issued leads to unpredictable results.

To reduce host bus usage, the DMA buffers are 32bits wide. Because of this each DMA transfer must be a multiple of 32bits, 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, like the OpenCore’s WISHBONE DMA engine. It is possible to use the core without an external DMA engine (Pseudo-DMA sequence). In this case software can track the status of the DMARQ line in the Status Register. 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, software can decide whether to read from or write to the DMA buffers.

4.6 DMA Write Cycle

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 no (more) data in the DMA Transmit Buffer, the core 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).
Because the core has no knowledge about the amount of data to transfer, 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.

4.7 DMA Read Cycle

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 DMACK-, 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.

4.8 DMA Big Endian versus Little Endian conversion

The core can perform automatic big versus little endian conversion during DMA transfers. This feature can be enabled per device, by setting (‘1’) the appropriate BeLeC bits in the control register. Figure 4.7 shows a 32bit DMA WRITE without endian conversion. Figure 4.8 shows a 32bit DMA WRITE with endian conversion. In both examples the data to be written is 0x12345678.

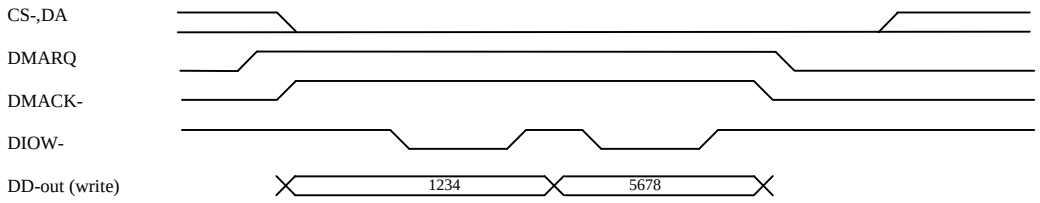


Figure 4.7 32bit DMA write transfer, without endian conversion

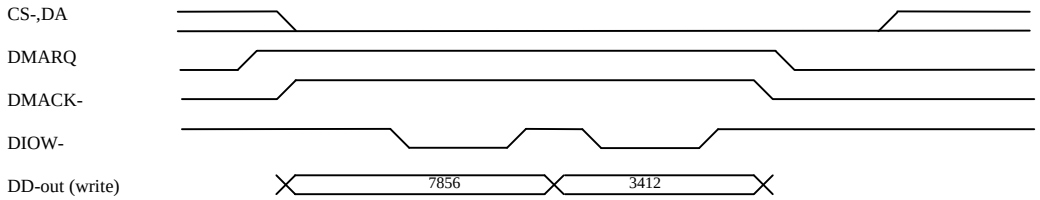


Figure 4.8 32bit DMA write transfer, with endian conversion

5

Architecture

This section should describe the architecture of the block. A block diagram should be included to describe the top level of the design.

Why deviate from standard implementations ?

Standard implementations like Intel's 82801 I/O Controller Hub are based on fixed clock architectures, especially the PCI bus using a 33MHz clock. This core is intended to be used in many different architectures with as many different clock frequencies. Using independent timing registers results in greater flexibility and better performance for a given clock frequency.

Appendix A

ATA Timings

PIO:
Teoc = (T0 – T1 – T2) or T9 or T2i whichever is greater.

All timings are in ns per clock cycle minus 2, rounded to the next highest integer.
For example T2 should be 100ns, the input clock frequency is 32MHz:

T2 = (100ns * 32MHz) –2 = 0.24 => 1

If the result of the equations is negative, then the result will be zero.
For example T1 should be 30ns, the input clock frequency is 32MHz:

T2 = (30ns * 32MHz) –2 = -1.0 => 0

Dma:
Teoc = (T0 – Td – Tm) or Tkw whichever is greater.

All timings are in ns per clock cycle minus 2, rounded to the next highest integer.
For example Td should be 80ns, the input clock frequency is 100MHz:

Td = (80ns * 100MHz) –2 = 6