# IDE/ATA Controller Using Altera MAX Series

2014.09.22

AN-495



Altera® MAX® II, MAX V, and MAX 10 devices can be used in this example application.

Storage devices, such as the floppy drive, CD-ROM drive, and hard disk drive, are connected to the computer through the IDE/ATA interface. This design example illustrates the implementation of an IDE/ATA controller using a supported Altera device through which a host computer or microprocessor system can connect to a standard Integrated Drive Electronics (IDE) device.

**Related Information**

- **Design Example for MAX II**
  Provides the design files for this application note (AN 495).
- **Design Example for MAX 10**
  Provides the MAX 10 design files for this application note (AN 495).
- **Power Management in Portable Systems Using MAX II CPLDs**
- **MAX II CPLD Design Guidelines**
- **Information Technology - AT Attachment with Packet Interface - 5 (ATA/ATAPI-5)**
  Provides more information about different PIO and DMA modes, PIO read and write waveforms and timing specifications, and the various internal registers of the IDE device.

## IDE/ATA Controller and Interface

When controllers and hard drives had proprietary technologies, a controller from one manufacturer did not work well with a hard drive from another manufacturer. The IDE was created to standardize the use of hard drives in computers. This was based on a concept of combining the controller and the hard drive, thereby reducing interface costs and making firmware implementations easier. The controller residing on a chip provided the means for transferring data to or from the host computer.

This IDE controller, also known as the ATA (Advanced Technology Attachment) controller, is an asynchronous parallel interface between a host microprocessor system and a standard IDE device. Therefore, this can be called a host adapter because it provides a way to connect a complete IDE device to the host.

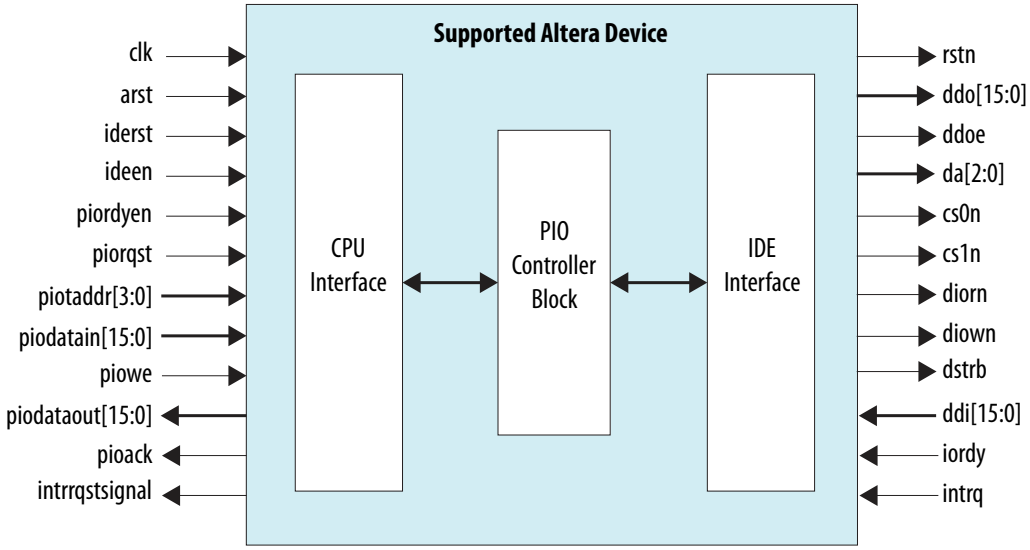
From the time of its inception, the ATA interface has been upgraded frequently and newer versions have been introduced. This design example implements an IDE controller compatible with the ATA-5 interface. The ATA-5 standard supports the following modes of operation:

- The PIO mode
- The DMA mode

Although the ATA-5 standard supports two modes, this design is restricted to only the PIO mode (mode 0) and with only one device connected to the controller (master).

**Figure 1: Basic Block Diagram of an IDE/ATA Interface**

The IDE/ATA interface consists of three blocks—the CPU interface block that receives commands from the CPU, the PIO controller block that contains the PIO state machine, and the IDE interface block that generates the signals required by the IDE device to carry out data transfer with the host (computer).




# IDE/ATA Controller Using Supported Altera Devices

The IDE interface supports two modes of data transfer—the PIO mode and the DMA mode. This design example is restricted to mode 0 of PIO data transfer.

**Table 1: Description of Interface Signals**

Lists the brief description of the various signals as shown in the Basic Block Diagram of an IDE/ATA Interface figure.

| Signal | Size | Direction | Description |
|---|---|---|---|
| `clk` | 1 | Input | Same as the clock of the processor. This example works at a clock frequency of 100 MHz. |
| `arst` | 1 | Input | Asynchronous active-low reset to reset the controller. |
| `iderst` | 1 | Input | Active-high signal to reset the IDE device. |
| `ideen` | 1 | Input | Active-high signal to enable the IDE device. |
| `pioiordyen` | 1 | Input | Active-high signal to enable the `IORDY` signal from the IDE device. |

| Signal | Size | Direction | Description |
|---|---|---|---|
| piorqst | 1 | Input | Active-high signal to start a PIO data transfer cycle. |
| pioaddr[3:0] | 4 | Input | 4-bit bus to select the device address and the chip select signals of the IDE device. |
| piodatain[15:0] | 16 | Input | 16-bit bus to send the data to the IDE device. |
| piowe | 1 | Input | Active-high signal to set the direction of data transfer.<br>• piowe is 1—write operation<br>• piowe is 0—read operation |
| intrrqstsignal | 1 | Output | Signal to interrupt the CPU. |
| pioack | 1 | Output | Signal to indicate the end of a PIO read/ write cycle. |
| piodataout [15:0] | 16 | Output | 16-bit bus to hold the data read from the IDE device. |
| rstn | 1 | Output | Active-low signal to reset the IDE device. |
| ddo [15:0] | 16 | Output | 16-bit data bus that transfers the data sent by the CPU to the device. The lower 8 bits are used for 8-bit data transfers. |
| da [2:0] | 3 | Output | 3-bit active-high signal. Contains the binary coded address asserted by the host to access a register or data port in the device. |
| cs0n | 1 | Output | Active-low chip select signals from the host used to select the command block or control block registers.<br>• cs0 is 0—select control register<br>• cs1 is 0—select command register<br>• cs0 or cs1 is 0 or 1—the device ignores transitions on DIOR-/DIOW- and release the data bus. |
| cs1n | 1 | | |
| diorn | 1 | Output | Active-low strobe signal asserted by the host to read device registers or the data port. |
| diown | 1 | Output | Active-low strobe signal asserted by the host to write to the device registers or the data port. |
| dstrb | 1 | Output | data-in strobe signal from the device. The rising edge of dstrb latches the data from the device into the host. |

| Signal | Size | Direction | Description |
|---|---|---|---|
| ddi [15:0] | 16 | Input | 16-bit data bus that contains the data read from the IDE device. |
| iordy | 1 | Input | This signal is negated to extend the host transfer cycle of any host register access (read or write) when the device is not ready to respond to a data transfer request. Optional for mode 0 but required for higher modes. |
| intrq | 1 | Input | Used by the selected device to notify the host of an event. The device internal interrupt pending state is set when such an event occurs. |
| ddoe | 1 | Output | Used to read data from the IDE device. The data on the ddi bus is put on the data lines piodataout after the signal is brought low. |

## CPU Interface Block

The CPU interface block receives the signals from the host CPU and stores them in the internal registers of the supported Altera device. Depending on the information in these internal registers, the PIO controller goes through the various states of the PIO read and write operations.

## PIO Controller Block

The PIO controller block contains the PIO state machine. Whenever the host sends a read or write request, the state machine goes through the appropriate read or write states, resulting in data transfer.

A module called the *piomodecontroller* in this block determines the states through which the state machine should proceed. Another module called the *runoncecounter* loads the count value to generate the required delays as per the read or write timing requirements. The count value is loaded through parameters called piomodet1, piomodet2, piomodet4, and piomodeteoc.

The default values loaded in this design example are for a processor operating at a frequency of 100 MHz. These parameter values may be modified for a processor operating at a different frequency. A third module called the *updowncounter* decrements the count loaded on each clock pulse, thus generating the required delay.

## IDE Interface Block

This IDE interface block generates the appropriate interfacing signals so that the data is loaded into or from the addressed internal register of the IDE device. The internal register selected depends on the value of the da[2:0], cs0n, and cs1n lines. A read or write operation is indicated by the diorn and the diown lines, respectively.

# Implementation

The detailed description of the implementation is based on the MAX II devices. This application can also be implemented in MAX V and MAX 10 devices.

This design example can be implemented using a MAX II device such as the EPM570 device or any other MAX II device with the required general-purpose I/O (GPIO) pin count and LEs.

This design example has been implemented in Verilog and successful operation has been demonstrated using the MDN-B2 demo board, as described in this application note.

# Acknowledgments

Design example adapted for Altera MAX 10 FPGAs by:

Orchid Technologies Engineering and Consulting, Inc.

Maynard, Massachusetts 01754

TEL: 978-461-2000

WEB: **www.orchid-tech.com**

EMAIL: **info@orchid-tech.com**

# Document Revision History

| Date | Version | Changes |
|---|---|---|
| September 2014 | 2014.09.22 | Added MAX V and MAX 10 devices. |
| December 2007 | 1.0 | Initial release. |