

Multiplexing SDIO Devices Using Altera MAX Series

2014.09.22

AN-509



Subscribe



Send Feedback

This design example shows the capability of Altera® MAX® II, MAX V and MAX 10 devices to implement a Secure Digital (SD) or Secure Digital Input Output (SDIO) device multiplexer. The SD or SDIO has the capability to multiplex two or more SD/SDIO devices to an SD/SDIO host equipped with a single SD interface. This document also illustrates how the select line of this multiplexer can be controlled through an I²C interface.

The supported Altera devices are an excellent choice to implement SD/SDIO device multiplexers. The supported devices' low power, high I/O pin count, and ability to tolerate any possible power-on sequence on their core and I/O banks make them an ideal choice in this application.

Related Information

- [Design Example for MAX II](#)
Provides the MAX II design files for this application note (AN 509).
- [Design Example for MAX 10](#)
Provides the MAX 10 design file for this application note (AN 509).
- [AN428: MAX II CPLD Design Guidelines](#)
- [AN422: Power Management in Portable Systems Using MAX II CPLDs](#)

Using Multiple SD Devices

It may often be required for an SD host controller with a single SD interface to support more than one SD device. The SD protocol and standards recommend doing this using one of following two methods:

- Use a bidirectional multiplexer between the SD host and the multiple SD devices and use this multiplexer to multiplex the data lines. This is performed while the clock line is connected to each of the multiple SD devices.
- Retain the data lines connected to the multiple SD devices while multiplexing the unidirectional clock line.

This design example uses the second method described above.

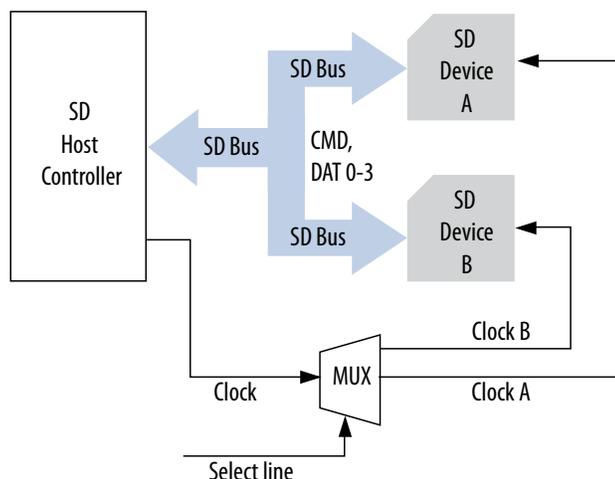
Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, Enpirion, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

ISO
9001:2008
Registered

ALTERA
now part of Intel

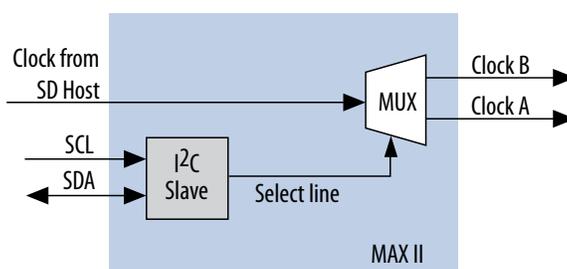
Figure 1: Implementing a Clock-Based SD MUX



I²C Select Line Multiplexer Using MAX II Devices

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.

The “MUX” in Figure 1 represents a unidirectional multiplexer that is implemented using a MAX II device. The select line for the multiplexer is controlled through an I²C interface, which is also implemented in the same device. The multiplexer is thus an I²C slave, and the clock line from the host controller is either connected to the SD Device A or the SD Device B, depending on the I²C data received.

Figure 2: I²C Select Line MUX Implementation in a MAX II Device

The I²C interface implementation in the MAX II device (I²C slave) has a 7-bit address and follows the general I²C protocol. The start signal is sent by the master, followed by the 7-bit address and an R/W bit. When the address broadcast on the I²C bus matches with the slave device’s address, an ACK (acknowledge) signal is sent by the device. This is followed by a DATA byte, as per the select line selection that is required by the master. This is then followed by another ACK signal by the slave. The end of a session of data transaction is when the Stop (P) signal is sent by the master.

Table 1: I²C Interface Pin Description

Signal	Purpose	Direction
SCL	I ² C Clock	Output
SDA	I ² C Serial Data	Bidirectional

Figure 3: I²C Signal Format



S = Start (SCLK high, SDA high to low)
 R/W = Read/Write (1 for Read, 0 for Write)
 ACK = Acknowledgement (SDA held low by receiver)
 P = Stop (SCLK high, SDA low to high)
 Default Slave Address = 0000000 (00h)

This design example can be implemented with the EPM240G or EPM240 device or any other MAX II device. It is demonstrated in an I²C bus environment. Implementation involves using this design example source code and allocating I²C bus lines, the SD host controller clock line, the eject line, and the clock lines for the SD Device A and SD Device B. An LED indicator that is used to indicate the current selection is connected to an output port assigned to display the select line status. This SD multiplexer is demonstrated on the MDN-B2 demo board and with the help of an I²C simulator. The simulator is created using a PC parallel port and interfacing hardware to create an I²C-compliant two-wire bus. The MDN-B2 features two SD sockets to accommodate two SD devices (to be multiplexed) and an SD card-shaped PCB adapter that fits into a standard SD socket of an SD host.

The utility program, which uses the parallel port, and its hardware interface with the MAX II-based multiplexer and provide the SDA and SCL connections as required on an I²C two-wire system. When implemented, this design allows the I²C master (or the control panel of the Maxim utility) to control the select line of the MAX II-based multiplexer on the MDN-B2. The following table lists the EPM240G pin assignments for this design example.

Table 2: EPM240G Pin Assignments for Design Example Using the MDN-B2 Demo Board

Assign unused pins As input-tristated in the Quartus[®] II software. You must also enable the Auto Open-Drain setting on the SCLK and SDA pins. To do this, on the Assignments menu, click Settings and then select Analysis and Synthesis Settings to enable the Auto Open-Drain setting. These settings are followed by a compilation cycle.

Signal	Pin	Signal	Pin
APD_inhibit	Pin 14	APD_inhibit_inv	Pin 12
eject	Pin 64	SCLK	Pin 39
SDA	Pin 40	sdA_clock	Pin 2

Signal	Pin	Signal	Pin
sdB_clock	Pin 7	sd_host_clock	Pin 28
sel	Pin 76	—	—

Related Information

[Dallas Semiconductor's Maxim application note AN3230](#)

Provides more information about setting up an I2C environment.

Design Demonstration on the MDN-B2 Demo Board

To demonstrate this design on the MDN-B2 demo board, perform the following steps:

1. Turn on the power to the demo board (using slide switch SW1).
2. Download the design onto the MAX II device through the JTAG header JP5 on the demo board and a conventional programming cable (ByteBlaster™ II or USB-Blaster™).
3. Keep SW4 on the demo board pressed before and during the start of the programming process.
4. Once completed, turn off the power and remove the JTAG connector.

Setting up a Parallel Port Driven I²C Environment on PC

To set up a parallel port driven I2C environment on your PC, perform the following steps:

1. Download a software utility, such as the Maxim parallel port utility, to communicate with the slave in the I²C defined protocol. Install the parallel port software. (The `ParDS2W.exe` program is used in this example.)
2. You must install a parallel port driver to enable access to the parallel port in Windows XP or Windows 2000 for this parallel port utility.
3. After installation, you must configure the Direct-IO program. Open the Windows control panel and click the Direct IO icon.
4. Enter the `Begin` and `End` addresses of your parallel port (normally, this is 378 through 37F; however, confirm your PC's parallel port address by looking at the settings in Control Panel/System/Hardware/Device Manager/Ports/ECP Printer Port (LPT)/Resources.
5. Configure the parallel port to ECP by changing the BIOS settings when you start your PC.
6. Next, select the Security tab of the Direct IO control panel and browse to the directory path of the `ParDS2W.exe` program.
7. Click **Open** and then click **Add** to add the program. The path of this utility is shown in the **Allowed Processes** field. Click **OK**.
8. Attach the parallel port I²C dongle that is supplied along with the MDN-B2 demo board. Use an extension cord, if necessary, to extend the parallel port connection closer to your demo board.
9. Attach the 4-pin socket on the pigtail of the I²C parallel port dongle to the I2C header (JP3) of the demo board so that the red mark on the socket meets pin 1 on the JP3 header.
10. Open the `ParDS2W` program, select the appropriate parallel port address of your PC (as seen when configuring Direct IO) and set the 2-Wire Device Address to `00h`.
11. Finally, you can test the I²C setup on the **Test Circuit** tab to see if you have a Test PASS message in the Status window. If you do, the I²C environment is set.

Related Information

[Parallel port driver from Direct-IO](#)

Setting up SD Host/Card Reader to Work with The MDN-B2 Demo Setup

To set up the SD host/card reader to work with the MDN-B2 demo setup, perform the following steps:

1. Use a general purpose USB SD card reader modified to access the card detect line from its SD socket.
2. Ensure that the USB cable to the SD card reader is disconnected and the power switch on the MDN-B2 is turned off.
3. Fix the SD card-shaped PCB adapter in the SD card reader's SD socket, and then insert two SD memory card devices in the two SD sockets on the MDN-B2.
4. Connect the card detect line tapped from the SD card reader to the MDN-B2 board through pin 5 on JP8.
5. Turn on the MDN-B2 and then connect the USB cable to the SD card reader.
"My Computer" on the PC to which the card reader is connected shows one of the SD memory cards.
6. Using the parallel port utility, perform a write operation into the I²C slave.
Observe the corresponding changes in the "My Computer" listing of the appropriate SD memory card that was multiplexed.
7. To do a write I²C operation, press **Start** followed by **Write Byte**. Enter hex byte 00 in the field adjacent to **Write Data** and click **Write Data**.
8. This will select one of the SD devices. Similarly, writing FF will select the other SD device. LED D12 on the MDN-B2 will change its state accordingly.

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	<ul style="list-style-type: none">• Added MAX V and MAX 10 devices.• Updated template.• Restructured document.
December 2007	1.0	Initial release.