ALTASMI_PARALLEL Megafunction User Guide

Introduction

The ALTASMI_PARALLEL megafunction provides access to erasable programmable configurable serial (EPCS) devices through parallel data input and output ports. An EPCS device is a serial configuration device that can be used to perform an active serial (AS) configuration on supported Altera® devices. During an AS configuration, the Altera device is the master and the EPCS device is the slave.

> If you are unfamiliar with the Altera megafunction or the MegaWizard™ Plug-In Manager, refer to the *Megafunction Overview User Guide*.

> For more information about AS configuration, refer to the Active Serial Configuration page on the Altera website.

The ALTASMI_PARALLEL megafunction implements a basic active serial memory interface (ASMI). To use this megafunction, you do not need to know the details of the serial interface and the read and write protocol of an EPCS device. You can perform the following tasks with the ALTASMI_PARALLEL megafunction:

- Specify the type of EPCS device to use

- Read the EPCS silicon identification (device identification)

- Protect a certain sector in the EPCS device from write or erase

- Read the data at a specified address from the EPCS device

- Perform single-byte write to the EPCS device

- Perform page write to the EPCS device

- Read the status of the EPCS device

- Erase a specified sector on the EPCS device

- Erase memory in bulk on the EPCS device

Device Family Support

The ALTASMI_PARALLEL megafunction is supported in the following devices:

- Arria® GX device onwards

- Cyclone® series of devices

- HardCopy® III device onwards

- Stratix® II device onwards

General Description

This section describes an overview of the ALTASMI_PARALLEL megafunction.

Figure 1 shows a typical block diagram of the ALTASMI_PARALLEL megafunction.

Figure 1. ALTASMI_PARALLEL Block Diagram
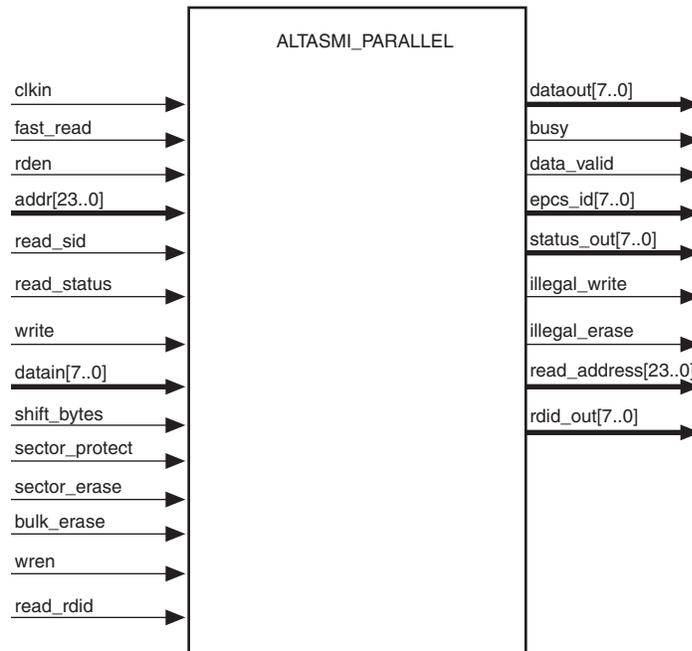
For more information about the ports and parameters of this megafunction, refer to "Ports and Parameters" on page 19.

The memory in the EPCS device is divided into two sections:

- **Configuration memory**—contains the bitstream of the configuration data
- **General purpose memory**—used for any application-specific storage

You can use the ALTASMI_PARALLEL megafunction to access the general purpose memory portion of the EPCS devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) through the supported Altera devices (refer to Figure 2).

⚠ CAUTION Do not access the configuration memory in the EPCS device. Doing so risks corrupting the configuration bits.

Figure 2. Accessing General Purpose Memory in Altera Devices
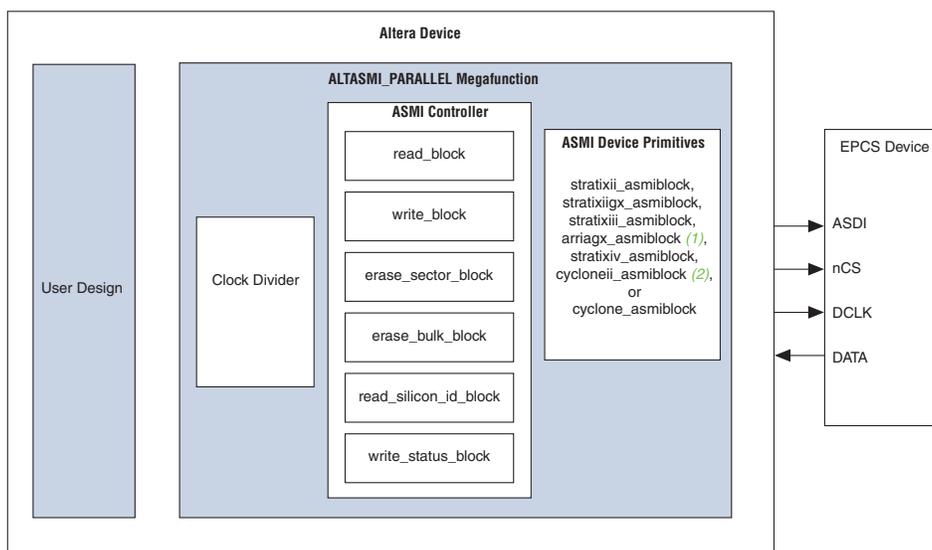
Notes to Figure 2:

(1)   The arriagx_asmi primitive is used in Arria II and Arria II GX synthesis.

(2)   The cycloneii_asmi primitive is used in Cyclone II and Cyclone III synthesis.

> For more information about features, memory array organization, and operation codes of the EPCS device, refer to the *Altera Configuration Device* chapter in volume 2 of the *Configuration Handbook*.

ALTASMI_PARALLEL Megafunction Operations and Timing Requirements

This section describes all the operations and timing requirements of the ALTASMI_PARALLEL megafunction. Understanding the operations help you to implement the ALTASMI_PARALLEL megafunction with the functions you desire.

The following shows the supported operations listed from the highest priority to the lowest. The megafunction executes the operation with the highest priority when more than one operations are requested at once. The rest is ignored.

■ "Read Memory Capacity ID from the EPCS Device" on page 4 (available for EPCS16, EPCS64, and EPCS128 devices only)

■ "Read Silicon ID from the EPCS Device" on page 5 (available for EPCS1, EPCS4, EPCS16, and EPCS64 devices only)

■ "Read Data from the EPCS Device" on page 6

■ "Fast Read Data from the EPCS Device" on page 7 (available for EPCS16, EPCS64, and EPCS128 devices only)

■ "Write Data to the EPCS Device" on page 9

■ "Erase Memory in a Specified Sector on the EPCS Device" on page 12

☞ You cannot verify the functionality of this megafunction through simulation. The timing diagrams in this section show the expected results in the hardware and are not the actual results from the simulation.

The general timing requirement for all operations is the `clkin` signal must toggle at the appropriate frequency range at all times. The megafunction uses the `clkin` signal to feed the EPCS device and to perform internal processing. For a read operation, the `clkin` signal can toggle at a maximum frequency of 20 MHz. For a fast read operation, the frequency of the `clkin` signal can toggle at a maximum of 25 MHz.

For more information about the supported frequency and EPCS devices timing, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 1 of the *Configuration Handbook*.

Altera recommends checking the `busy` signal before sending a new command. When the `busy` signal is deasserted, allow two clock cycles before sending a new signal. This delay allows the circuit to reset itself before executing the next command.

The following sections describe each operation in the ALTASMI_PARALLEL megafunction.

Read Memory Capacity ID from the EPCS Device

This section explains in detail the operation and timing requirement for reading the memory capacity ID from the EPCS device. Use the `read_rdid` signal to instruct the megafunction to read the memory capacity ID from the EPCS device.

Figure 3 shows an example of the latency when the ALTASMI_PARALLEL megafunction is executing the read command.

Figure 3. Reading Memory Capacity ID　*(Note 1)*
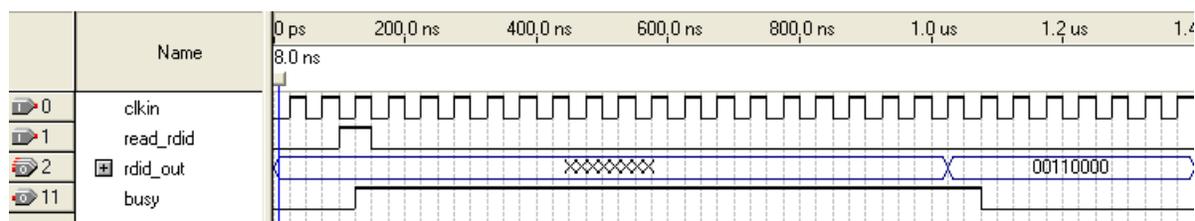
Note to Figure 3:

(1) The latency shown does not correctly indicate the true processing time. It only illustrates the command.

The megafunction registers the `read_rdid` signal on the rising edge of the `clkin` signal. After the megafunction registers the `read_rdid` signal, the `busy` signal is asserted to indicate that the read command is in progress.

Ensure that the memory capacity ID appears on the `rdid_out[7..0]` signal before the `busy` signal is deasserted. This allows you to sample the `rdid_out[7..0]` signal as soon as the `busy` signal is deasserted.

The `rdid_out[7..0]` signal holds the value of the memory capacity ID until the device is reset. Therefore, this read command should be executed only once.

☞ To meet setup and hold time requirements, assert the `read_rdid` signal anytime between the rising edges of the `clkin` signal, and keep the `read_rdid` signal asserted for at least one full clock cycle. Ensure that the `read_rdid` signal assertion does not coincide with the rising edges of the `clkin` signal.

If you keep the `read_rdid` signal asserted while the `busy` signal is deasserted after the megafunction has finished processing the read command, the megafunction re-registers the `read_rdid` signal as a value of one and carries out the command again. Therefore, before the megafunction deasserts the `busy` signal, you should deassert the `read_rdid` signal before the `busy` signal is deasserted.

Read Silicon ID from the EPCS Device

This section explains in detail the operation and timing requirement for reading the silicon ID from the EPCS device. Use the `read_sid` signal to instruct the megafunction to read the silicon ID from the EPCS device.

Figure 4 shows an example of the latency when the ALTASMI_PARALLEL megafunction is executing the read command.

Figure 4. Reading Silicon ID *(Note 1)*
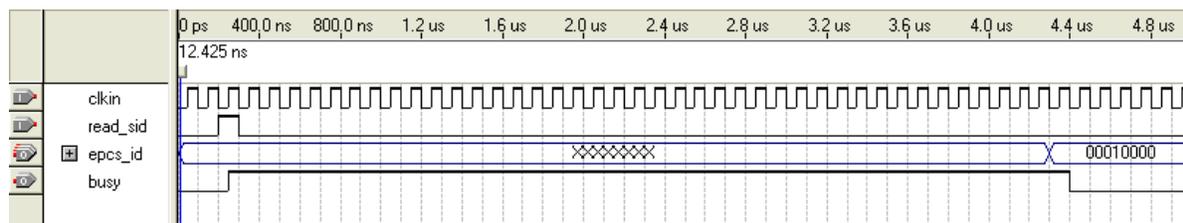
Note to Figure 4:

(1)   The latency shown does not correctly indicate the true processing time. It only illustrate the command.

The megafunction registers the `read_sid` signal on the rising edge of the `clkin` signal. After the megafunction registers the `read_sid` signal, it asserts the `busy` signal to indicate that the read command is in progress.

Ensure that the silicon ID appears on the `epcs_id[7..0]` signal before the `busy` signal is deasserted. Therefore, you can sample the `epcs_id[7..0]` signal as soon as the `busy` signal is deasserted.

The `epcs_id[7..0]` signal holds the value of the silicon ID until the device is reset. Therefore, this command should be executed only once.

☞ To meet setup and hold time requirements, assert the `read_sid` signal anytime between the rising edges of the `clkin` signal, and keep the `read_sid` signal asserted for at least one full clock cycle. Ensure that the `read_sid` signal assertion does not coincide with the rising edges of the `clkin` signal.

If you keep the read_sid signal asserted while busy signal is deasserted and the megafunction has finished processing the read command, the megafunction re-registers the read_sid signal as a value of one and carries out another read command. Therefore, before the megafunction deasserts the busy signal, you should deassert the read_sid signal.

Read Data from the EPCS Device

This section explains in detail the operation and timing for reading from the EPCS device. Use the read signal to instruct the megafunction to read data from the EPCS device. The ALTASMI_PARALLEL megafunction supports two types of read data operation: multiple-byte and single-byte read.

Figure 5 shows an example of the latency when the ALTASMI_PARALLEL megafunction is executing multiple-byte read comand, while Figure 6 shows an example of single-byte read command.

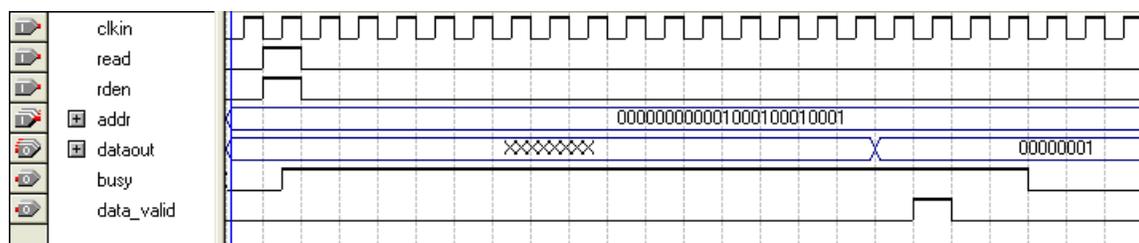**Figure 5.** Reading Multiple-Byte *(Note 1)*
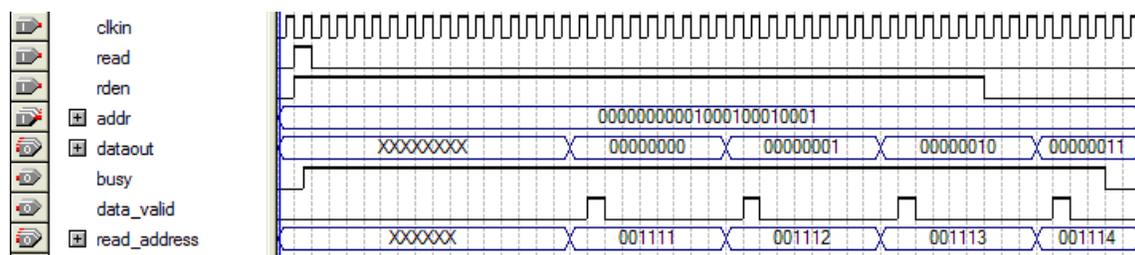
Note to Figure 5:

(1)   The latency shown does not correctly indicate the true processing time. It illustrates the command only.

Figure 6. Reading Single-Byte *(Note 1)*

Note to Figure 6:

(1)   The latency shown does not correctly indicate the true processing time. It illustrates the command only.

The megafunction registers the read signal on the rising edge of the clkin signal. After the megafunction receives the read command, it asserts the busy signal to indicate that the read command is in progress.

Ensure that the read address appears on the addr[23..0] signal before asserting the read signal. The rden signal must also be asserted to enable the read operation.

The first data byte then appears on the dataout[7..0] signal. The megafunction then asserts the data_valid signal for one clock cycle, which indicates that the dataout[7..0] signal contains a new valid data.

If you enable the `read_address[23..0]` port in the MegaWizard Plug-In Manager, the memory address for each data byte that appears on `dataout[7..0]` signal is reflected on this port.

If you want to continue reading sequential data from the EPCS device, the `rden` signal must remain asserted. This condition allows you to read every memory address from the EPCS device with a single read command.

For every eight `clkin` signal clock cycles, a new data byte from the next address appears on the `dataout[7..0]` signal with its corresponding memory address on the `read_address[23..0]` signal . The `data_valid` signal is asserted for one clock cycle after the new data byte is out on the `dataout[7..0]` signal. Use the `data_valid` signal as an indication to capture the new data byte.

After the second-to-last byte of data to be read appears on the `dataout[7..0]` signal, and the `data_valid` signal is asserted, deassert the `rden` signal to indicate the end of the read command. A new byte from the next address then appears on the `dataout[7..0]` signal, and the `data_valid` signal is reasserted before the megafunction stops processing. Only then does the megafunction deassert the `busy` signal.

For a single-byte read, simply assert the `rden` signal for one clock cycle in conjunction with the `read` signal, or deassert the `rden` signal any time before the first data appears on the `dataout[7..0]` signal, and the `data_valid` signal asserts for the first time.

Monitor the `data_valid` signal and sample the `dataout[7..0]` signal only when the `data_valid` signal is a value of one.

After the read operation is completed, the `dataout[7..0]` signal holds the value of the last byte read until you issue a new read command or reset the device.

☞ The `read`, `rden`, and `addr[7..0]` signals must adhere to setup and hold time requirements for the `clkin` signal. These signals should remain stable at the rising edge of the `clkin` signal.

Fast Read Data from the EPCS Device

This section explains in detail the operation and timing for fast reading of data from the EPCS device. Use the `fast_read` signal to instruct the megafunction to read the silicon ID from the EPCS device. The ALTASMI_PARALLEL megafunction supports two types of fast read data operation: multiple-byte and single-byte operation.

Figure 7 shows an example of the latency when the ALTASMI_PARALLEL megafunction is executing multiple-byte fast read command, while Figure 8 shows an example of single-byte read command.

Figure 7. Fast Reading Multiple-Byte *(Note 1)*
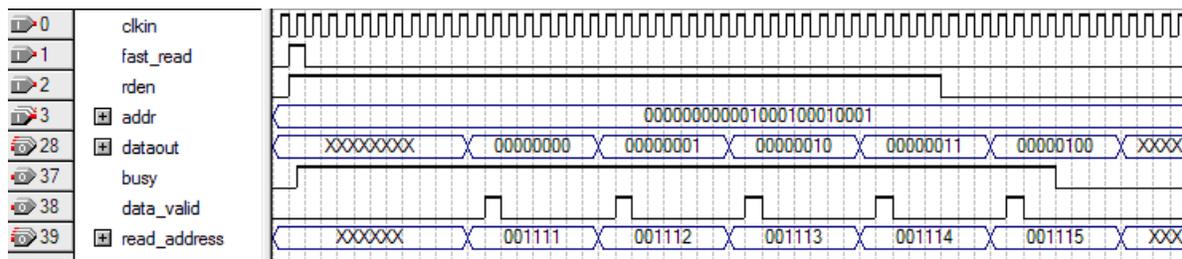
Note to Figure 7:

(1)   The latency shown does not correctly indicate the true processing time. It only illustrates the command.

Figure 8. Fast Reading a Single-Byte *(Note 1)*
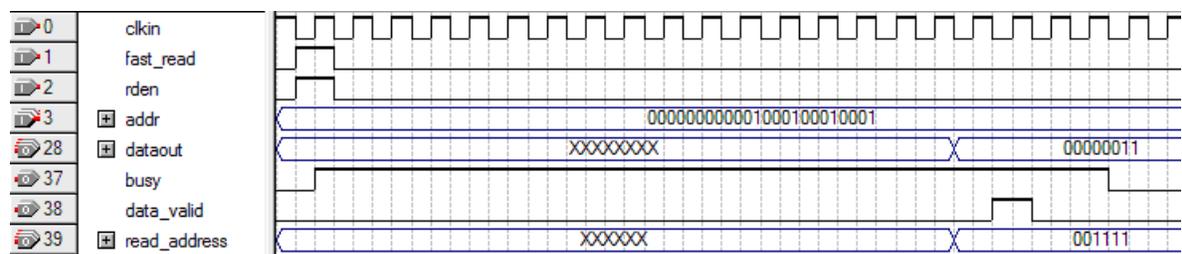
Note to Figure 8:

(1)   The latency shown does not correctly indicate the true processing time. It only illustrates the command.

The fast read command is the same as the read command, with the following exceptions:

■   The fast read command produces the first byte of data on the dataout[7..0] port eight cycles later than it appears for the read command.

■   The fast read command is available on EPCS16, EPCS64, and EPCS128 devices only.

■   The fast read command can run up to 25 MHz clock frequency.

■   The fast read and the read commands are mutually exclusive—you can use only one of them in each megafunction instantiation.

■   The fast read and read operations are mutually exclusive. You can only do either read or fast read operation at a time. The fast read operation is a replacement for the read operation at higher than 20 MHz clock frequency.

The megafunction registers the fast_read signal on the rising edge of the clkin signal. For the megafunction to register the read command, ensure that the memory address appears on the addr[23..0] signal before the fast_read signal is asserted. The rden signal must also be asserted to enable the fast read command.

After the megafunction registers the `fast_read` signal, the `busy` signal is asserted to indicate that the fast read command is in progress. The data appears on the `dataout[7..0]` signal. The first valid byte of fast read data appears eight clock cycles later than it appears in a normal read command. Also, after the first byte, subsequent bytes appear sequentially, similar to any multiple-byte normal read operation. Therefore, the fast read operation performs faster than the read operation. The megafunction asserts the `data_valid` signal for one clock cycle, to indicate `dataout[7..0]` contains a new valid data.

If you enable the `read_address[23..0]` signal in the MegaWizard Plug-In Manager, the read address for each data byte on `dataout[7..0]` signal appears on the `read_address[23..0]` signal.

Assert the `rden` signal until you have finished reading sequential data from the EPCS device. This condition allows you to read every memory address from the EPCS device with a single read command.

The data from the next address appears on the `dataout[7..0]` signal and its memory address appears on the `read_address[23..0]` signal at every eight `clkin` clock cycles. The `data_valid` signal is asserted for one clock cycle after the new data byte appears on the `dataout[7..0]` signal. Use the `data_valid` signal as an indication to capture the new data byte.

When the second-to-last byte of data to be read appears on the `dataout[7..0]` signal, and the `data_valid` is asserted, deassert the `rden` signal to indicate the end of the fast read command. The final data byte appears on the `dataout[7..0]` signal, the `data_valid` is reasserted, and then the megafunction deasserts the `busy` signal.

For a single-byte fast read operation, assert the `rden` and the `fast_read` signals for a single clock cycle, or deassert the `rden` at any time before the first data byte appears on the `dataout[7..0]` signal, and the `data_valid` signal is asserted for the first time.

Monitor the `data_valid` signal to ensure you sample the `dataout[7..0]` signal only when the `data_valid` signal is asserted.

After the fast read operation is complete, the `dataout[7..0]` signal holds the value of the last byte read until you issue a new fast read command or reset the device.

☞ The `fast_read`, `rden`, and `addr[7..0]` signals must adhere to setup and hold time requirements for the `clkin` signal. These signals should remain stable at the rising edge of the `clkin` signal.

Write Data to the EPCS Device

This section explains in detail the operation and timing requirement for writing data to the EPCS device. The ALTASMI_PARALLEL megafunction supports two types of write operation: single-byte write and page-write.

Single-Byte Write Operation

Figure 9 shows an example of the latency when the ALTASMI_PARALLEL megafunction is performing a single-byte write operation.

Figure 9. Writing a Single-Byte *(Note 1)*
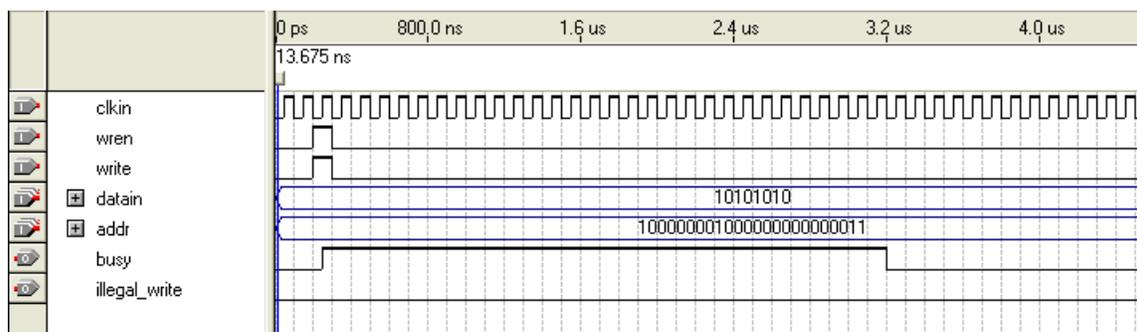
Note to Figure 9:

(1)   The latency shown does not reflect the true processing time. It only illustrates the command.

For single-byte write operation or when the PAGE_SIZE parameter is a value of one, the shift_bytes signal is not required. Ensure that the data byte is available on the datain[7..0] signal and the memory address is available on the addr[23..0] signal before setting the write and wren signals to one.

If wren signal is a value of zero, the write operation is not carried out and the busy signal remains deasserted. If the me mory region is protected (you can set this in the EPCS status register), then the write operation does not proceed, and the busy signal is deasserted. The megafunction then asserts the illegal_write signal for two clock cycles to indicate that the command has been aborted. The write, datain[7..0], and addr[23..0] signals are registered on the rising edge of the clkin signal.

After the megafunction receives the write command, it asserts the busy signal to indicate that the write operation is in progress. The busy signal stays asserted while the EPCS device is writing the data byte into the flash memory.

For more information about the write operation timing, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 1 of the *Configuration Handbook*

If you keep both the wren and write signals asserted while the busy signal is deasserted after the megafunction has finished processing the write command, the megafunction re-registers the wren and write signals as a value of one and carries out another write command. Therefore, before the megafunction deasserts the busy signal, you should deassert the wren and write signals.

Page-Write Operation

The page-write operation rules are more complicated than the single-byte write operation because you must shift the data bytes on the datain[7..0] signal.

Figure 10 shows an example of the page-write operation when the PAGE_SIZE parameter is a value of eight, while Figure 11 shows an example of writing four bytes of data.

Figure 10. Page-Write Operation: Example 1
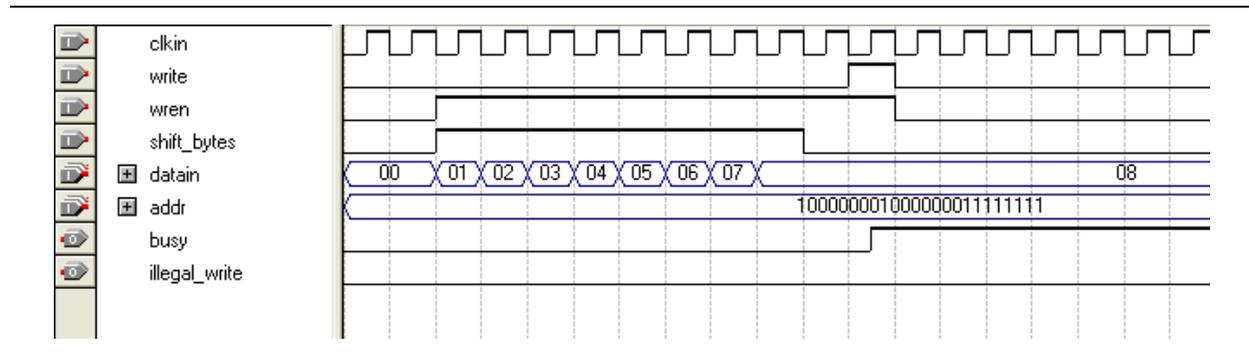
Figure 11. Page-Write Operation: Example 2
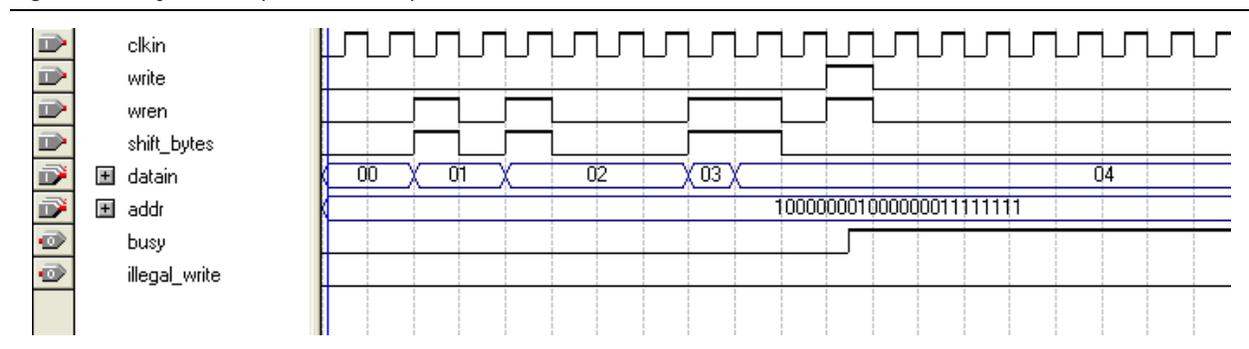
The page-write sequence is executed in two stages: stage 1 and stage 2.

For stage 1, you must assert the `wren` and `shift_bytes` signals to enable the megafunction to sample the data byte at `datain[7..0]` signal and to store the byte internally in the page-write buffer. The megafunction samples `datain[7..0]` signal at the rising edge of the `clkin` signal.

You do not need to ensure that a new data byte is available with each clock cycle; however, you can use the `shift_bytes` signal to control when the megafunction takes in a new data byte. Each time a new data byte is ready at `datain[7..0]` signal, assert the `shift_bytes` signal for one clock cycle to enable the megafunction to sample the data. The `wren` signal should be set to a value of one.

Continue controlling the `shift_bytes` and `wren` signals until all the data bytes are shifted into the page-write buffer for writing.

You can write any number of data bytes less than the `PAGE_SIZE` parameter value set in the MegaWizard Plug-In Manager.

☞ If you send more data bytes than the `PAGE_SIZE` parameter value, only the last (equivalent to `PAGE_SIZE` value) number of bytes is written to the EPCS device. The first few bytes are discarded. This behavior is consistent with the EPCS device itself.

☞ The `shift_bytes`, `wren`, and `datain[7..0]` ports must adhere to setup and hold time requirements for the `clkin` signal. These ports should remain stable at the rising edge of the `clkin` signal.

For stage 2, you must ensure that the start memory address to be written appears on the addr[23..0] signal before you assert the write signal. When you have completed sending all data bytes, assert the write signal to indicate to the megafunction that the internal write can proceed. The megafunction registers both the write and addr[23..0] ports on the rising edge of the clkin signal. You need to only send the start memory address to be written to. The EPCS device handles the address increment internally.

⚠ CAUTION
If the eight least significant address bits of the addr[7..0] are not all zero, sent data that continues beyond the end of the current page is not written into the next page. Instead, this data is written at the start memory address of the same page (from the address whose eight least significant address bits are all 0).

The megafunction passes the user-supplied data and the memory address as it is to the EPCS device. To avoid unexpected rearrangement of data order by the EPCS write operation, use a PAGE_SIZE of 256 bytes, and execute page-write operations at the start of each page boundary (where the addr[7..0] bits are all 0).

As soon as the megafunction receives the write command, the busy signal is asserted.

The busy signal remains asserted while the EPCS device is writing into the memory.

👣 For more information about the write operation timing, refer to *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet*.

If wren signal is a value of zero, the write operation is not carried out, and the busy signal remains deasserted.

If the memory region is protected (you can set this in the EPCS status register), the write operation does not proceed, and the busy signal is deasserted. The megafunction then asserts the illegal_write signal for two clock cycles to indicate that the write operation has been aborted.

If you keep both the wren and write signals asserted while the busy signal is deasserted after the megafunction has finished processing the write command, the megafunction re-registers the wren and write signals as a value of one, and carries out another write command. Therefore, before the megafunction deasserts the busy signal, you should deassert the wren and write signals.

☞ The SCFIFO megafunction is used as the storage buffer for the page write operation. This allows you to select the RAM or LEs as the storage buffer.

Erase Memory in a Specified Sector on the EPCS Device

This section explains in detail the operation and timing requirement for erasing the memory in a specified sector in the EPCS device. Use the sector_erase signal to instruct the megafunction to erase memory in a specified sector on the EPCS device.

Figure 12 shows an example of the latency when the ALTASMI_PARALLEL megafunction is executing the erase memory command.

Figure 12. Erasing Memory in a Specified Sector *(1)*
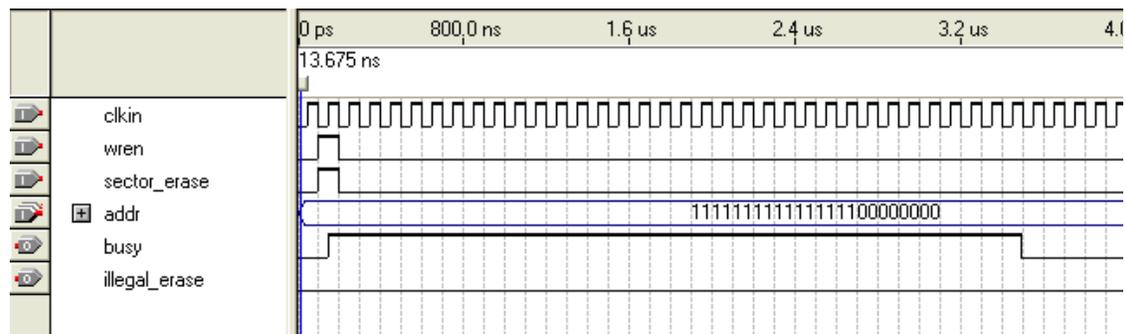
Note to Figure 12:

(1) The latency shown does not correctly reflect the true processing time. It illustrates the command only.

The megafunction registers the `sector_erase` signal on the rising edge of the `clkin` signal. The address placed on the `addr[23..0]` signal can be any valid address in the sector that you can erase.

Ensure that the memory address to be erased appears on the `addr[23..0]` signal before setting the `wren` and `sector_erase` signals to a value of one. After the megafunction receives the sector erase command, it asserts the `busy` signal as long as the sector is being erased.

For more information about the write operation timing, refer to *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet*.

If `wren` signal is a value of zero, then the sector erase operation is carried out, and the `busy` signal remains deasserted.

If the memory region is protected (specified in the EPCS status register), the erase operation cannot proceed, and the `busy` signal is deasserted. The `illegal_erase` port is then asserted for two clock cycles to indicate that the erase operation has been aborted.

If you keep the `wren` and `sector_erase` signals asserted while the `busy` signal is deasserted after the megafunction has finished erasing the memory, the megafunction re-registers the `wren` and `sector_erase` signals as a value of one and carries out another sector erase operation. Therefore, before the megafunction deasserts the `busy` signal, you should deassert the `wren` and `sector_erase` signals.

Erase Memory in Bulk on the EPCS Device

This section explains in detail the operation and timing for erasing the memory in bulk on the EPCS device. Use the `bulk_erase` signal to instruct the megafunction to erase memory in bulk on the EPCS device.

Figure 13 shows an example of the latency when the ALTASMI megafunction is executing the erase memory in bulk command.

This command erases all the memory on the EPCS device, including the configuration data portion. You must use this command with caution.

Figure 13. Erasing Memory in Bulk
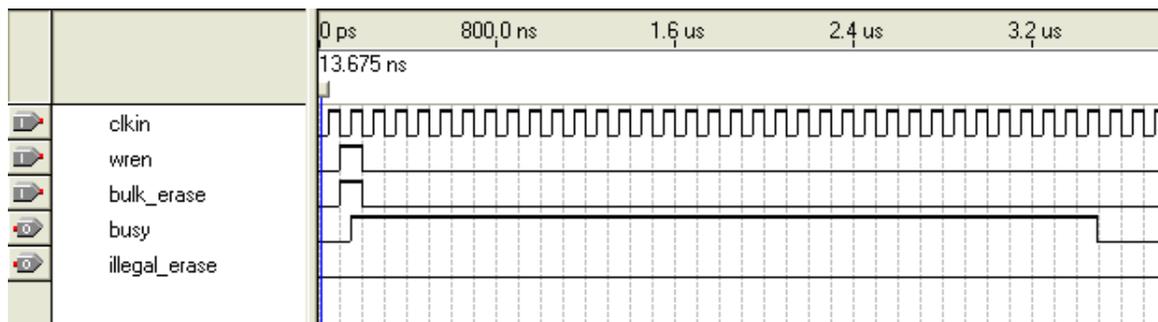
Note to Figure 13:

(1)   The latency shown does not correctly reflect the true processing time. It only illustrates the command.

If the `wren` signal is a value of one, the megafunction registers the `bulk_erase` signal at the rising edge of the `clkin` signal. The megafunction asserts the `busy` signal as soon as it receives the `bulk_erase` signal. The `busy` signal remains asserted for as long as it takes to erase the entire EPCS memory.

For more information about the serial configuration device timing parameters for write operation, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 1 of the *Configuration Handbook*.

If the `wren` signal is a value of zero, then the `bulk_erase` signal is not carried out, and the `busy` signal remains deasserted.

Also, if any of the memory regions are protected (you can set this in the EPCS status register), then the erase operation does not proceed, and the `busy` signal is deasserted. The `illegal_erase` port is then asserted for two clock cycles to indicate that the erase operation has been aborted.

☞   If you keep both the `wren` and `bulk_erase` ports asserted while the `busy` signal is deasserted after the megafunction has finished erasing memory in bulk command, the megafunction re-registers the `wren` and `bulk_erase` signals as a value of one and carries out another bulk erase operation. Therefore, before the megafunction deasserts the `busy` signal, you should deassert the `wren` and `bulk_erase` signals.

Protect a Sector on the EPCS Device

This section explains in detail the operation and timing requirement for protecting a sector in the EPCS device. Use the `sector_protect` signal to instruct the megafunction to protect a sector on the EPCS device.

Figure 14 shows an example of the latency when the ALTASMI_PARALLEL megafunction is executing the sector protect command.

Figure 14. Protecting a Sector
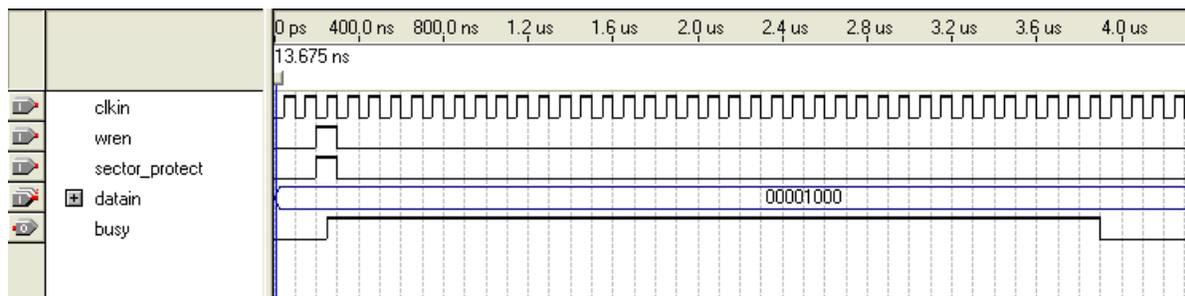
Note to Figure 14:

(1) The latency shown does not correctly reflect the true processing time. It illustrates the command only.

This command is used to write the EPCS status register to set the block protection bits. The block protection bits show which sectors are protected from write or erase, and provide protection in addition to that provided by the wren signal.

You can set the block protection bits in the EPCS status register to protect those sectors that contain configuration data, and are not intended for general-purpose memory usage.

For more information about block protection bits in the EPCS devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 1 of the *Configuration Handbook*.

You should ensure that the 8-bit code is available on the datain[7..0] signal before asserting the sector_protect and wren signals. The megafunction registers the sector_protect signal at the positive edge of the clkin signal.

The megafunction asserts the busy signal as soon as it receives the sector_protect signal. The busy signal remains asserted while the EPCS status register is written.

If the wren signal is a value of zero, the sector_protect signal is not carried out, and the busy signal remains deasserted.

☞ If you keep the wren and sector_protect signals asserted while the busy signal is deasserted after the megafunction has finished processing the sector protect command, the megafunction re-registers the wren and sector_protect signals as a value of one and carries out another write status register operation. Therefore, before the megafunction deasserts the busy signal, you should deassert the sector_protect signal.

Out of the 8 bits, only bits 2 to 3, or 2 to 4 (depending on the EPCS device) are used for block protection. The rest of the bits have other meanings for the ASMI operation, and cannot be overwritten by the sector protect operation. The following list shows some of the block protection levels available through ASMI:

- All sectors

- Upper-half of the memory

- Upper-quarter

- Half of the upper-quarter

Read Status Register of the EPCS Device

This section explains in detail the operation and timing requirement for reading status register in the EPCS device. Use the read_status signal to instruct the megafunction to read the status register of the the EPCS device.

Figure 15 shows an example of the latency when the ALTASMI_PARALLEL megafunction is executing the read status register command.

Figure 15. Reading a Status Register
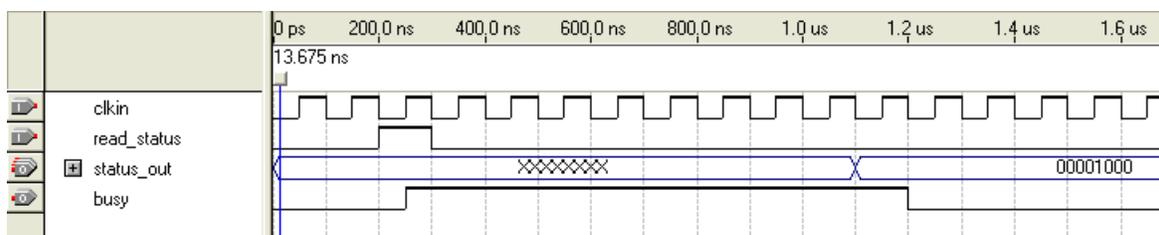
Note to Figure 15:

(1) The latency shown does not correctly reflect the true processing time. It illustrates the command only.

The megafunction registers the read_status signal on the rising edge of the clkin signal. After the megafunction receives the read_status signal, it asserts the busy signal to indicate that the read command is in progress. To prevent the megafunction from re-registering the command and executing it again, deassert the read_status signal before the busy signal is deasserted.

The megafunction ensures that the 8-bit status register value is available on the status_out[7..0] signal before deasserting the busy signal. You can sample the status_out[7..0] signal as soon as the busy signal is deasserted.

You must decode the 8-bit status register value to find out which sectors are protected.

For more information about the meaning of the status register value in the EPCS devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 1 of the *Configuration Handbook*.

The status_out[7..0] signal holds the value of the status register from the last read status command. The contents of the status register may have changed (via a sector protect command, for example). Therefore, before sampling the status_out[7..0] signal, you should issue a new read status command.

Parameter Settings

This section provides descriptions of the options available on the **Parameter Settings, EDA,** and **Summary** pages of the MegaWizard Plug-In Manager for the ALTASMI_PARALLEL megafunction. The ALTASMI_PARALLEL megafunction is listed under the **I/O** category in the MegaWizard Plug-In Manager.

For more information on how to start the MegaWizard Plug-In Manager, refer to the "Overview" section of the *Megafunction Overview User Guide*.

The options contained on page 2 and page 2a of the MegaWizard Plug-In Manager are the same for all supported device families. For more information, refer to the "Device Family Support" section of the *Megafunction Overview User Guide*.

The device family and the EPCS type that you select on page 2 and page 2a on the MegaWizard Plug-In Manager determine the ports and parameters that are available on page 3. Table 1 provides description of the options available on page 3 of the MegaWizard Plug-In Manager.

Table 1. ALTASMI_PARALLEL MegaWizard Plug-In Manager Page Option and Description (Page 3) (Part 1 of 3)

Option	Description
What EPCS type do you want to use?	Specify the EPCS type you want to use. Types available are **EPCS1**, **EPCS4**, **EPCS16**, **EPCS64**, and **EPCS128** devices.
Enable 'read silicon ID' operation	Enables the ability to read the silicon ID of the EPCS device with an active-high input signal named `read_sid`. When this signal is asserted, the megafunction reads the silicon ID of the EPCS device. After it is read, the 8-bit silicon ID appears on the `epcs_id[7..0]` signal until the device is reset.
	This option is only available for EPCS1, EPCS4, EPCS16, and EPCS64 devices. For more information, refer to "Read Silicon ID from the EPCS Device" on page 5.
Enable 'read status' operation	Enables the ability to read the port status using an active-high input signal named `read_status`. When this signal is asserted, the megafunction reads the EPCS status register. As it is read, the 8-bit value appears on the `status_out[7..0]` signal.
	For more information, refer to "Read Status Register of the EPCS Device" on page 16.
Enable 'read identification' operation	Enables the ability to read the memory capacity ID of the EPCS device with an active-high input signal named `read_rdid`. When this signal is asserted, the megafunction reads the memory capacity ID of the EPCS device. The 8-bit ID appears on the `rdid_out[7..0]` signal until the device is reset.
	This option is only available for EPCS16, EPCS64, and EPCS128. For more information, refer to "Read Memory Capacity ID from the EPCS Device" on page 4.

Table 1. ALTASMI_PARALLEL MegaWizard Plug-In Manager Page Option and Description (Page 3) (Part 2 of 3)

Option	Description
Enable 'write' operation	Enables the ability to write to the EPCS device with an active-high input signal named `write`. When this port is asserted, the megafunction writes the data from the `datain[7..0]` signal (for single-byte write) or from the page-write buffer (for page-write) to the address that appears on the `addr[23..0]` port, and to subsequent addresses for page-write. In page-write mode, you must use the `shift_byte` signal to shift in data bytes before asserting the write signal. For more information, refer to "Write Data to the EPCS Device" on page 9.
Enable 'write' operation **(What 'write' mode do you want to use?)**	Specify the writing mode to the EPCS device (**single byte write** or **page write**). This option is only available when you turn on the **Enable 'write' operation** option.
Enable 'write' operation **(What is the 'page write' size?)**	Specify the size (in bytes) of the page that you want to write to the EPCS device (**2, 4, 8, 16, 32, 64, 128,** or **256 bytes**). This option is only available when you choose **page write** mode from the **What 'write' mode do you want to use?** option.
Enable 'write' operation **(Where do you want the 'page write' data to be stored?)**	Specifies the location where the page write data is stored before writing to the EPCS device (**RAM blocks** or **Logic elements**). If you want to save more space in the FPGA, select **RAM blocks**. This option is only available when you choose **page write** mode from the **What 'write' mode do you want to use?** option
Enable 'sector protect' operation	Enables the ability to protect sectors in the EPCS device from write and erase operations with an active-high input port named `sector_protect`. When this port is asserted, the megafunction reads the block protection code value on the `datain[7..0]` signal and writes it to the EPCS status register. To protect specific memory sectors, you must send their block protection code to the `datain[7..0]` signal. For more information, refer to "Protect a Sector on the EPCS Device" on page 14.
Enable 'bulk erase' operation	Enables the ability to erase all the memory of the EPCS device, including the configuration data portion with an active-high input signal named `bulk_erase`. When this signal is asserted, the megafunction implements a full erase that sets all the memory bits of the EPCS device to a value of one. For more information, refer to "Erase Memory in Bulk on the EPCS Device" on page 13.
Enable 'sector erase' operation	Enables the ability to erase a certain sector in the EPCS memory with an active-high input signal named `sector_erase`. When the signal is asserted, the megafunction implements a full erase of the sector. The sector to be erased is indicated by the value of the `addr[23..0]` signal. The value of the signal can be any valid address within the sector. For more information, refer to "Erase Memory in a Specified Sector on the EPCS Device" on page 12.

Table 1. ALTASMI_PARALLEL MegaWizard Plug-In Manager Page Option and Description (Page 3) (Part 3 of 3)

Option	Description
Enable 'fast read' operation	Enables the ability to perform a fast read operation with an active-high input signal named `fast_read`. When this signal is asserted, the megafunction performs a fast read from the memory address that appears on the `addr[23..0]` signal. Each data byte appears on the `dataout[7..0]` signal as it is read.
	This option is only available for EPCS16, EPCS64, and EPCS128 devices.The fast read operation replaces the normal settings.
	The `fast_read` signal supports single-byte fast read and sequential fast read. If a write or erase operation is in progress (the `busy` signal is asserted), the fast read command is ignored. The fast read operation occurs only when allowed by the `rden` signal. For more information, refer to "Fast Read Data from the EPCS Device" on page 7.
Use 'wren' input port	Enables write and erase operations to the EPCS memory with an active-high input signal named `wren`. If this signal is asserted, the write and erase operations are enabled, and disabled if the signal is deasserted. If the `wren` signal is not used, all write and erase operations are automatically enabled when the command appears on the relevant megafunction input port. The affected commands are write, sector protect, bulk erase, and sector erase.
	This option is only available when you turn on the **Enable 'write' operation, Enable 'sector protect' operation, Enable 'bulk erase' operation,** or **Enable 'sector erase' operation** option.
Use 'read_address' output port	This signal holds the address from which data is being read. This signal works together with the `dataout[7..0]` signal. As data appears on `dataout[7..0]`, the address from which the data byte was read appears on the read-address output port.

Page 4 of the MegaWizard Plug-In Manager, allows you to specify options for stimulation and timing and resource estimation. This page normally lists the simulation libraries required for functional simulation by third-party tools. However, the ALTASMI_PARALLEL megafunction does not have simulation model files, and cannot be simulated.

Page 5 of the MegaWizard Plug-In Manager allows you to specify the generated file types. The Variation file contains wrapper code in the HDL you specified on page 2a. You can optionally generate Pin Planner ports PPF file (**.ppf**), AHDL Include file (<*function name*>**.inc**), VHDL component declaration file (<*function name*>**.cmp**), Quartus II symbol file (<*function name*>**.bsf**), Instantiation template file (<*function name*>**.v**), and Verilog HDL black box file (<*function name*>**_bb.v**). If you selected the **Generate netlist** option on page 4, the file for the synthesis area and timing estimation netlist (<*function name*>**_syn.v**) is also available. A gray checkmark indicates a file that is automatically generated, and a red checkmark indicates generation of an optional file.

Ports and Parameters

This chapter describes the ports and parameters for the ALTASMI_PARALLEL megafunction. These ports and parameters are available to customize the ALTASMI_PARALLEL megafunction according to your application.

The parameter details are only relevant if you bypass the MegaWizard Plug-In Manager interface and use the megafunction as a directly parameterized instantiation in your design. The details of these parameters are hidden from the MegaWizard Plug-In Manager interface.

For the most current information about the ports and parameters for this megafunction, refer to the latest version of the Quartus II Help.

Table 2 shows the list of input ports. Table 3 shows the list of output ports. Table 4 shows the ALTASMI_PARALLEL megafunction parameters.

Table 2. Input Ports for ALTASMI_PARALLEL Megafunction (Part 1 of 2)

Port Name	Condition	Size	Description
addr[]	Required	24 bit	Contains the value of the EPCS memory address to be read from, written to, and erased from.
bulk_erase	Optional	1 bit	Active-high port that is used to execute the bulk erase operation. If asserted, the megafunction performs a full-erase operation that sets all memory bits of the EPCS device to '1', which includes the general purpose memory of the EPCS device.
clkin	Required	1 bit	Input clock port for the ASMI block. In general, the clkin signal must toggle at the appropriate frequency range at all times. The megafunction uses the signal to feed the EPCS device and to perform internal processing. For more information about the EPCS clock frequency specification, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 1 of the *Configuration Handbook*.
datain[]	Optional	8 bit	Parallel input data of 1-byte length for write and sector protect operations.
fast_read	Optional	1 bit	Active-high port that is used to execute the fast read operation. If asserted, the megafunction performs a fast read operation from a memory address value that appears on the addr[23..0] port. You should use the fast_read port together with the rden port. For more information about the fast read operation, refer to the "Fast Read Data from the EPCS Device" on page 7.
rden	Required	1 bit	Active-high port that is used to allow read and fast read operations to be performed as long as it stays asserted. For more information about read and fast read operations, refer to the "ALTASMI_PARALLEL Megafunction Operations and Timing Requirements" on page 3.
read	Required	1 bit	Active-high port that is used to execute the read operation. If asserted, the megafunction performs a read operation from a memory address value that appears on the addr[23..0] port. You should use the read port together with the rden port. The read port is disabled if the fast_read port is used. For more information about the read operation, refer to the "Read Data from the EPCS Device" on page 6.
read_rdid	Optional	1 bit	Active-high port that is used to execute the read memory capacity ID operation. If asserted, the megafunction proceeds to read the memory capacity ID of the EPCS device, and the value of the memory capacity ID appears at the rdid_out[7..0] port. For more information about the read memory capacity ID operation, refer to the "Read Memory Capacity ID from the EPCS Device" on page 4.

Table 2. Input Ports for ALTASMI_PARALLEL Megafunction (Part 2 of 2)

Port Name	Condition	Size	Description
read_sid	Optional	1 bit	Active-high port that is used to execute the read silicon ID operation. If asserted, the megafunction proceeds to read the silicon ID of the EPCS device, and the value of the silicon ID appears at the `epcs_id[7..0]` port. For more information about the read silicon ID operation, refer to the "Read Silicon ID from the EPCS Device" on page 5.
read_status	Optional	1 bit	Active-high port that is used to execute the read EPCS status register operation. If asserted, the megafunction reads the status register of the EPCS decvice, and outputs the value at the `status_out[7..0]` port. You can use the `read_status` port to determine which memory sector on the EPCS device is read-only. For more information about the read EPCS status register operation, refer to the "Read Status Register of the EPCS Device" on page 16.
sector_erase	Optional	1 bit	Active-high port that is used to execute the sector erase operation. If asserted, the megafunction starts erasing the memory sector on the EPCS device based on the memory address value at the `addr[23..0]` port. The value can be any valid memory address within the sector to be erased. For more information about the sector erase operation, refer to the "Erase Memory in a Specified Sector on the EPCS Device" on page 12.
sector_protect	Optional	1 bit	Active-high port that is used to execute the sector protect operation. If asserted, the megafunction takes the value of the `datain[7..0]` port and writes to the EPCS status register. The status register contains the block protection bits that represent the memory sector to be protected. For details about the sector protect operation, refer to the "Protect a Sector on the EPCS Device" on page 14. For more information about block protection bits in the EPCS devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 1 of the *Configuration Handbook*.
shift_bytes	Optional	1 bit	Active-high port that is used to shift data bytes during the write operation. You must use this port together with the write port during the page-write operation. The megafunction samples and shifts the data in the `datain[7..0]` port at the rising edge of the `clkin` signal, as long as the `shift_bytes` signal is asserted. You should continue shifting the required bytes into the EPCS device until the megafunction finishes sampling and storing the data internally. For more information about the write operation, refer to the "Write Data to the EPCS Device" on page 9.
wren	Optional	1 bit	Active-high port that is used to allow write and erase operations to be performed as long as it stays asserted. If this port is not generated, the megafunction automatically allows all write and erase operations to be performed. Use this port together with the `write`, `sector_protect`, `bulk_erase`, and `sector_erase` ports. For more information about the write and erase operations, refer to the "ALTASMI_PARALLEL Megafunction Operations and Timing Requirements" on page 3.
write	Optional	1 bit	Active-high port that is used to execute the write operation. If asserted, the megafunction writes the data from the `datain[7..0]` port (for single-byte write), or from the page-write buffer (for page-write), to the memory address specified in the `addr[23..0]` port (and to the subsequent addresses for page write operation). In page-write operation, you must use the `shift_bytes` port to shift in data bytes before asserting the write port. For more information about the write operation, refer to the "Write Data to the EPCS Device" on page 9.

Table 3. Output Ports for ALTASMI_PARALLEL Megafunction (Part 1 of 2)

Port	Required	Size	Description
busy	Required	1 bit	Port that indicates the megafunction is performing valid instruction. The busy signal goes high when the megafunction is executing valid instruction, and goes low once the instruction is completed.
data_valid	Required	1 bit	Port that indicates that the dataout[7..0] port contains a valid data byte read from the EPCS memory. You should sample the dataout[7..0] port only when the data_valid signal is high. For more information about the write operation, refer to the "Write Data to the EPCS Device" on page 9.
dataout[]	Required	8 bit	Port that contains the data byte read from the EPCS memory during read operation. This port holds the value of the last data byte read until the device is reset, or until a new read operation is carried out. You should sample the dataout[7..0] port only when the data_valid signal is high. For more information about the read operation, refer to the "Read Data from the EPCS Device" on page 6.
epcs_id[]	Optional	8 bit	Port that contains the silicon ID of the EPCS device once the read silicon ID operation is completed. This port holds the value of the silicon ID until you reset the device. You should sample the epcs_id[7..0] port after the busy signal goes low. For more information about the read silicon ID operation, refer to the "Read Silicon ID from the EPCS Device" on page 5.
illegal_erase	Optional	1 bit	Port that indicates that an erase instruction has been set to a protected sector on the EPCS memory. This port is compulsory when the sector_erase port, bulk_erase port, or both, are specified. The illegal_erase signal goes high to indicate that the megafunction has aborted the erase instruction. The signal pulses high for two clock cycles—one clock cycle before, and one clock cycle after the busy signal goes low. You should monitor this port to detect the status of an erase operation. For more information about the erase operation, refer to the "ALTASMI_PARALLEL Megafunction Operations and Timing Requirements" on page 3.
illegal_write	Optional	1 bit	Port that indicates that a write instruction is targeting a protected sector on the EPCS memory. This port is compulsory when the write port is specified. The illegal_write signal goes high to indicate that the megafunction has aborted a write instruction. The signal pulses high for two clock cycles—one clock cycle before, and one clock cycle after the busy signal goes low. You should monitor this port to detect the status of a write operation. For more information about the write operation, refer to the "Write Data to the EPCS Device" on page 9.
rdid_out[]	Optional	8 bit	Port that contains the memory capacity ID of the EPCS device once the read memory capacity ID operation is completed. This port holds the value until you reset the device. You should sample the rdid_out[7..0] port after the busy signal goes low. For details about the read memory capacity ID operation, refer to the "Read Memory Capacity ID from the EPCS Device" on page 4.

Table 3. Output Ports for ALTASMI_PARALLEL Megafunction (Part 2 of 2)

Port	Required	Size	Description
read_address[]	Optional	24 bit	Port that contains the memory address of the EPCS to be read from. You should use this port together with the dataout[7..0] port. For more information about the read operation, refer to the "Read Data from the EPCS Device" on page 6.
status_out[]	Optional	8 bit	Port that contains the value of the EPCS status register once the read status register operation is completed. This port holds the value until you execute another reading status register operation, or until you reset the device. To obtain the most recent value of the status register, you must perform a read status register operation before sampling the status_out[7..0] port. You should sample the port only after the busy signal goes low. For more information about the read status register operation, refer to the "Read Status Register of the EPCS Device" on page 16.

Table 4. Parameters for ALTASMI_PARALLEL Megafunction (Part 1 of 2)

Parameter	Type	Desription
EPCS_TYPE	String	Specifies the EPCS device type used. Values are **EPCS1**, **EPCS4**, **EPCS16**, **EPCS64**, or **EPCS128**. The default is **EPCS4**.
INTENDED_DEVICE_FAMILY	String	Specifies the device family you intend to use. This parameter is used for modeling and behavioral simulation purposes, as each device family has its ows ASMI primitive. The default is **UNUSED**.
LPM_HINT	String	This parameter allows you to assign Altera-specific parameters in VHDL Design Files (**.vhd**). The default is **UNUSED**.
LPM_TYPE	String	This parameter identifies the library of parameterized modules (LPM) entity name in VHDL Design Files. For example, LPM_TYPE="ALTASMI_PARALLEL".
PAGE_SIZE	Integer	Specifies the maximum number of byte that is sent during one single write instruction. Valid values range from **1** to **256**. This parameter is required if the write port is specified. The default is **1**.
PORT_BULK_ERASE	String	Specifies if the bulk_erase port is generated. Values are **PORT_USED** and **PORT_UNUSED**. The default is **PORT_UNUSED**. If you intend to use the bulk_erase port, you must set this parameter to **PORT_USED** .
PORT_FAST_READ	String	Specifies if the fast_read port is generated. Values are **PORT_USED** and **PORT_UNUSED**. The default is **PORT_UNUSED**. ifI you intend to use the fast_read port, you must set this parameter to **PORT_USED**.
PORT_ILLEGAL_ERASE	String	Specifies if the illegal_erase port is generated. Values are **PORT_USED** and **PORT_UNUSED**. The default is **PORT_UNUSED**. If the PORT_BULK_ERASE parameter or the PORT_SECTOR_ERASE parameter is set to **PORT_USED,** you must set this parameter to **PORT_USED**.
PORT_ILLEGAL_WRITE	String	Specifies if the illegal_write port is generated. Values are **PORT_USED** and **PORT_UNUSED**. The default is **PORT_UNUSED**. If the PORT_WRITE parameter is set to **PORT_USED**, you must set this parameter to **PORT_USED**.

Table 4. Parameters for ALTASMI_PARALLEL Megafunction (Part 2 of 2)

Parameter	Type	Desription
PORT_RDID_OUT	String	Specifies if the rdid_out port is generated. Values are **PORT_USED** and **PORT_UNUSED**. The default is **PORT_UNUSED**. If the PORT_READ_RDID parameter is set to **PORT_USED**, you must set this parameter to **PORT_USED** .
PORT_READ_ADDRESS	String	Specifies if the read_address port is generated. Values are **PORT_USED** and **PORT_UNUSED**. The default is **PORT_UNUSED**. If you intend to use the read_address port, you must set this parameter to **PORT_USED**.
PORT_READ_RDID	String	Specifies if the read_rdid port is generated. Values are **PORT_USED** and **PORT_UNUSED**. The default is **PORT_UNUSED**. If you intend to use the read_rdid port, you must set this parameter to **PORT_USED** .
PORT_READ_SID	String	Specifies if the read_sid port is generated. Values are **PORT_USED** and **PORT_UNUSED**. The default is **PORT_UNUSED**. If you intend to use the read_sid port, you must set this parameter to **PORT_USED**.
PORT_READ_STATUS	String	Specifies if the read_status port is generated. Values are **PORT_USED** and **PORT_UNUSED**. The default is **PORT_UNUSED**. If you intend to use the read_status port, you must set this parameter to **PORT_USED**.
PORT_SECTOR_ERASE	String	Specifies if the sector_erase port is generated. Values are **PORT_USED** and **PORT_UNUSED**. The default is **PORT_UNUSED**. If you intend to use the sector_erase port, you must set this parameter to **PORT_USED**.
PORT_SECTOR_PROTECT	String	Specifies if the sector_protect port is generated. Values are **PORT_USED** and **PORT_UNUSED**. The default is **PORT_UNUSED**. If you intend to use the sector_protect port, you must set this parameter to **PORT_USED**.
PORT_SHIFT_BYTES	String	Specifies if the shift_bytes port is generated. Values are **PORT_USED** and **PORT_UNUSED**. The default is **PORT_UNUSED**. If you intend to use the shift_bytes port, you must set this parameter to **PORT_USED**.
PORT_WREN	String	Specifies if the wren port is generated. Values are **PORT_USED** and **PORT_UNUSED**. The default is **PORT_UNUSED**. If you intend to use the wren port, you must set this parameter to **PORT_USED**.
PORT_WRITE	String	Specifies if the write port is generated. Values are **PORT_USED** and **PORT_UNUSED**. The default is **PORT_UNUSED**. If you intend to use the write port, you must set this parameter to **PORT_USED**.
USE_EAB	String	Specifies the location to store the page-write data, on RAM or logic elements (LEs). Use this parameter only if the write port is specified, the PAGE_SIZE parameter is greater than 1. Values are **ON** or **OFF**. The default is **ON**. Using LEs as storage consumes more space on the device, but frees up the RAM. If you use one M4K memory block as storage, it is recommended to set this parameter to **ON**.

Document Revision History

The following table shows the revision history for this user guide.

Date	Document Version	Changes Made
July 2009	3.0	Updated for Quartus® II 9.1 release, specifically: ■ Removed "Device Family Support" ■ Added new information in "Introduction" on page 1 ■ Added "Parameter Settings" on page 17 ■ Added link to *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* ■ Updated to include information about `read_rdid` signal ■ Updated Figure 2 on page 3 to include Arria® II GX and Stratix IV ■ Added Figure 1 on page 2 ■ Removed "How to Contact Altera" and "Typographic Conventions" sections.
October 2007	2.4	Updated for Quartus® II 7.2 release, specifically: ■ Updated for new MegaWizard™ Plug-In Manager pages ■ Updated to include information about new `fast_read` command
May 2007	2.3	Updated for Quartus II 7.1 release, specifically: ■ Added Arria™ GX to list of supported devices in "Device Family Support" ■ Added Figure 1–2 ■ Updated Figures 1-2, 2-2, 2-3, 2-4, and 2-5
March 2007	2.2	■ Removed Table 1-1 and added a list of supported devices. ■ Updated for the Quartus II software version 7.0 by adding support for Cyclone® III device.
December 2006	2.1	Updated device family support to include Stratix III.
June 2006	2.0	Updated for Quartus II 6.0 release; specifically: ■ Updated all screen shots ■ Added the section "How to Use the Megafunction" on page 2–15
November 2005	1.0	Initial Release

101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support
www.altera.com/support

I.S. EN ISO 9001