MAX 10 User Flash Memory User Guide

UG-M10UFM
2015.05.04

101 Innovation Drive
San Jose, CA 95134
www.altera.com

Contents

MAX 10 User Flash Memory Overview

1

2015.05.04

UG-M10UFM Subscribe Send Feedback

Altera MAX® 10 FPGAs offer a user flash memory (UFM) block that stores non-volatile information.

The UFM provides an ideal storage solution that you can access using the Avalon Memory Mapped (Avalon-MM) slave interface to UFM.

The UFM block also offers the following features.

Features	Capacity
Endurance	Counts up to 10,000 program/erase cycles
Data retention (after 10,000 program/erase cycles)	• 20 years at 85 °C • 10 years at 100 °C
Maximum operating frequency	• Serial interface: 7.25 MHz • Parallel interface: 116 MHz
Data length	Stores data of up to 32 bits length in parallel

MAX 10 UFM Architecture and Features

2

2015.05.04

UG-M10UFM

The UFM architecture of MAX 10 devices is a combination of soft and hard IPs. You can only access the UFM using the Altera On-Chip Flash IP core in the Quartus II software.

UFM and CFM Array Size

Each array is organized as various sectors. You can erase each page or sector independently. The Altera On-Chip Flash IP core also gives you access to configuration flash memory (CFM) based on your specification in the parameter editor.

Table 2-1: UFM and CFM Array Size

This table lists the dimensions of the UFM and CFM arrays.

Device	Pages per Sector					Page Size (Kb)	Total User Flash Memory Size (Kb) (1)	Total Configuration Memory Size (Kb)
	UFM1	UFM0	CFM2	CFM1	CFM0			
10M02	3	3	0	0	34	16	96	544
10M04	0	8	41	29	70	16	1248	2240
10M08	8	8	41	29	70	16	1376	2240
10M16	4	4	38	28	66	32	2368	4224
10M25	4	4	52	40	92	32	3200	5888
10M40	4	4	48	36	84	64	5888	10752
10M50	4	4	48	36	84	64	5888	10752

UFM Memory Organization Map

The address scheme changes based on the configuration mode you specify in the Altera On-Chip Flash parameter editor.

The following tables show the dynamic UFM and CFM support based on different configuration mode and MAX10 FPGA variant.

(1) The maximum possible value, which is dependent on the mode you select.

Table 2-2: Dynamic Flash Size Support: Flash and Analog Variants

Configuration	UFM1	UFM0	CFM2	CFM1	CFM0
Dual compressed images	UFM space	UFM space	—	—	—
Single uncompressed image	UFM space	UFM space	UFM space		—
Single compressed image	UFM space	UFM space	UFM space	UFM space	
Single uncompressed image with memory initialization	UFM space	UFM space	—	—	—
Single compressed image with memory initialization	UFM space	UFM space	—	—	—

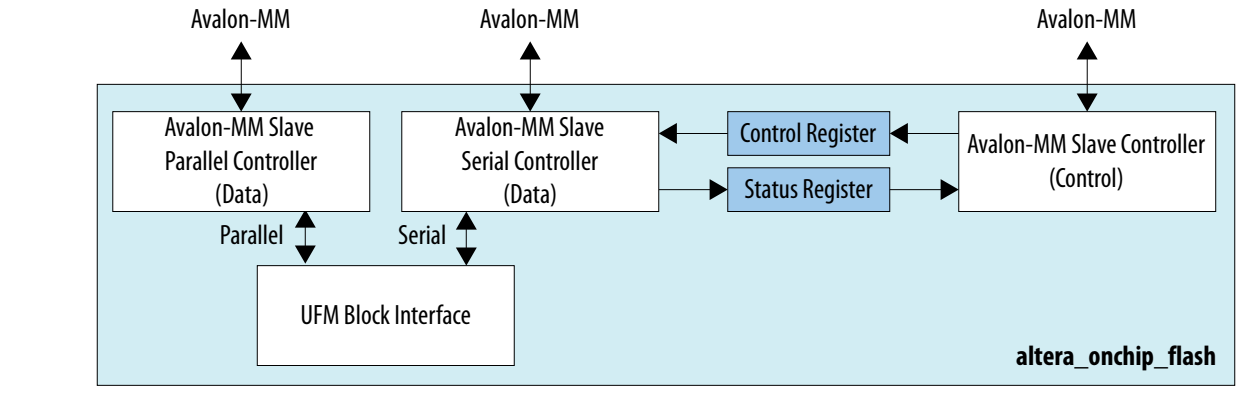
Table 2-3: Dynamic Flash Size Support: Compact Variant

Configuration	UFM1	UFM0	CFM2	CFM1	CFM0
Dual compressed images	Not available				
Single uncompressed image	UFM space	UFM space	—	—	—
Single compressed image	UFM space	UFM space	—	—	—
Single uncompressed image with memory initialization	Not available				
Single compressed image with memory initialization	Not available				

UFM Block Diagram

This figure shows the top level view of the Altera On-Chip Flash IP core block diagram. The Altera On-Chip Flash IP core supports both parallel and serial interfaces for all MAX 10 FPGAs, except for 10M02 devices. 10M02 devices only allow serial interface.

Figure 2-1: Altera On-Chip Flash IP Core Block Diagram



This IP block has two Avalon-MM slave controllers:

- Data—a wrapper of the UFM block that provides read and program accesses to the flash.
- Control—the CSR and status register for the flash, which is required only for program and erase operations.

These figures show the detailed overview of the Avalon-MM interface during read and program (write) operation.

Figure 2-2: Altera On-Chip Flash IP Core Avalon-MM Slave Read and Program (Write) Operation in Parallel Mode

This diagram shows the standard interface for all 10M04, 10M08, 10M16, 10M25, 10M40, and 10M50 devices in parallel mode.

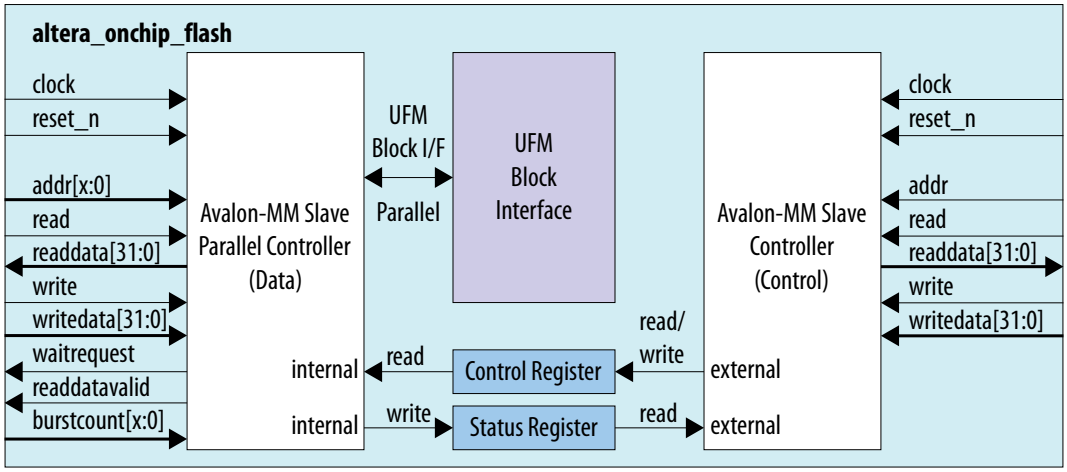
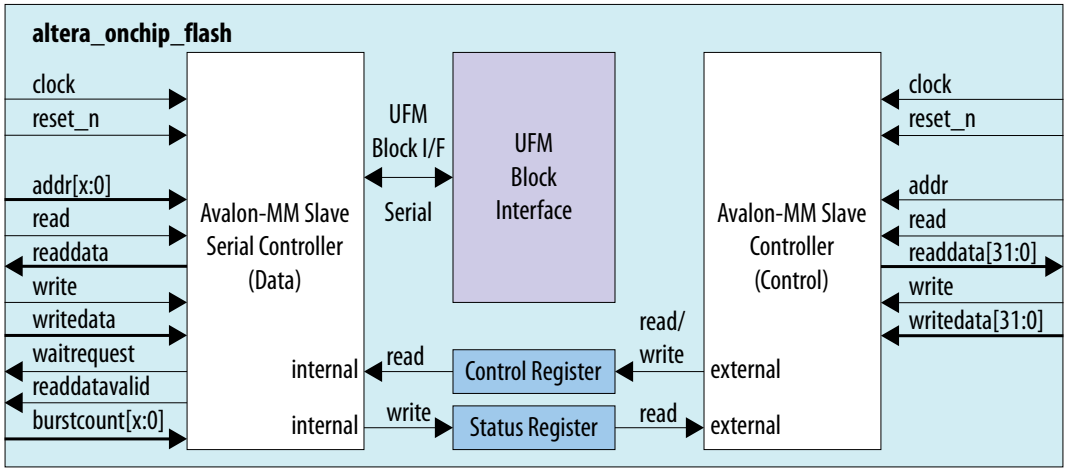


Figure 2-3: Altera On-Chip Flash IP Core Avalon-MM Slave Read and Program (Write) Operation in Serial Mode

This diagram shows the standard interface for all MAX 10 devices in serial mode.



These figures show the detailed overview of the Avalon-MM interface during read only operation.

Figure 2-4: Altera On-Chip Flash IP Core Avalon-MM Slave Read Only Operation in Parallel Mode

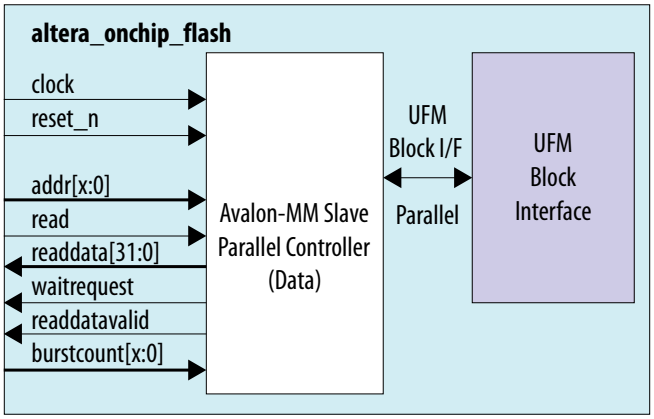
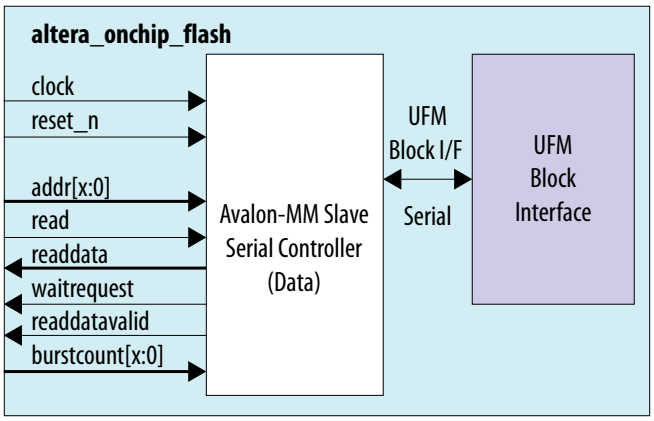


Figure 2-5: Altera On-Chip Flash IP Core Avalon-MM Slave Read Only Operation in Serial Mode



UFM Operating Modes

The UFM block offers the following operating modes:

- Read
- Burst read
- Program (Write)
- Sector erase
- Page erase
- Sector write protection

You can choose one of the following access modes in the Altera On-Chip Flash parameter editor to read and control the operations.

- Read and program mode—this mode allows both data and control slave interface. This mode is applicable for both UFM and CFM sectors.
- Read only mode—this mode allows only data slave interface, and restricted to only read operations. This mode is applicable for both UFM and CFM sectors.
- Hidden—this mode does not allow any read or program (write) operations. This mode is applicable only for CFM sectors.

The following table shows the comparison between parallel and serial modes.

Table 2-4: Comparison between Parallel Mode and Serial Mode

Feature	Parallel Mode	Serial Mode
Avalon-MM Data Interface	Parallel mode with 32-bit data bus	Serial mode with 32 bits based burst count
Access Mode	• Read and program • Read only • Hidden	• Read and program • Read only • Hidden
Read Mode	• Incrementing burst read • Wrapping burst read	Incrementing burst read only
Program (Write) Operation	Single 32-bit parallel program operation	Single 32-bit serial program operation

MAX 10 UFM Design Considerations

3

2015.05.04

UG-M10UFM Subscribe Send Feedback

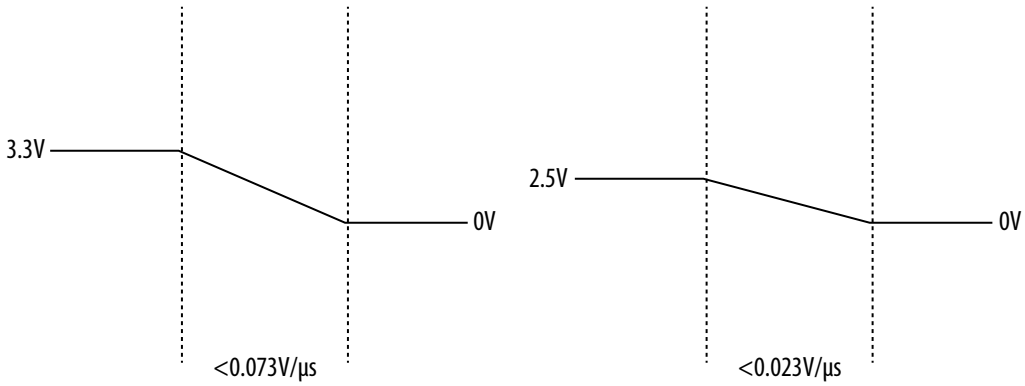
There are several considerations that require your attention to ensure the success of your designs. Unless noted otherwise, these design guidelines apply to all variants of this device family.

Guideline: UFM Power Supply Requirement

During UFM and CFM operations, make sure to follow the maximum slew rate requirement for power supply ramp down. This setting prevents device damage in case of power loss.

Table 3-1: Maximum Slew Rate Requirement

Device	Maximum Slew Rate
Single-supply device	0.073V/µs
Multi-supply device	0.023V/µs



Single-Supply Device **Multi-Supply Device**

Guideline: Program and Read UFM with JTAG

You can program UFM using JTAG interface version IEEE Standard 1149.1.

The JTAG interface supports Jam™ Standard Test and Programming Language (STAPL) Format File (**.jam**), Programmer Object File (**.pof**), and JAM Byte Code File (**.jbc**).

Guideline: UFM Content Initialization

You can initialize the UFM content using Altera software.

The initial memory content supports Memory Initialization File (**.mif**), and Hexadecimal (Intel-Format) File (**.hex**).

You can initialize the UFM content using either one of the following ways:

- Set the initial memory content through the Altera On-Chip Flash IP core.
- Set the initial memory content through the **Convert Programming File** tool in the Quartus II software when you convert **.sof** to **.pof**.

Guideline: Erase Before Program

Make sure to erase the flash location before you perform a program (write) operation.

MAX 10 UFM Implementation Guides

4

2015.05.04

UG-M10UFM Subscribe Send Feedback

Altera On-Chip Flash IP Core

The Altera IP core design flow helps you get started with any Altera IP core.

Introduction to Altera IP Cores

Altera® and strategic IP partners offer a broad portfolio of off-the-shelf, configurable IP cores optimized for Altera devices. The Quartus® II software installation includes the Altera IP library. You can integrate optimized and verified Altera IP cores into your design to shorten design cycles and maximize performance. You can evaluate any Altera IP core in simulation and compilation in the Quartus II software. The Quartus II software also supports integration of IP cores from other sources. Use the IP Catalog to efficiently parameterize and generate synthesis and simulation files for a custom IP variation.

The Altera IP library includes the following categories of IP cores:

- Basic functions
- DSP functions
- Interface protocols
- Low power functions
- Memory interfaces and controllers
- Processors and peripherals

Note: The IP Catalog (**Tools** > **IP Catalog**) and parameter editor replace the MegaWizard™ Plug-In Manager for IP selection and parameterization, beginning in Quartus II software version 14.0. Use the IP Catalog and parameter editor to locate and paramaterize Altera and other supported IP cores.

Related Information

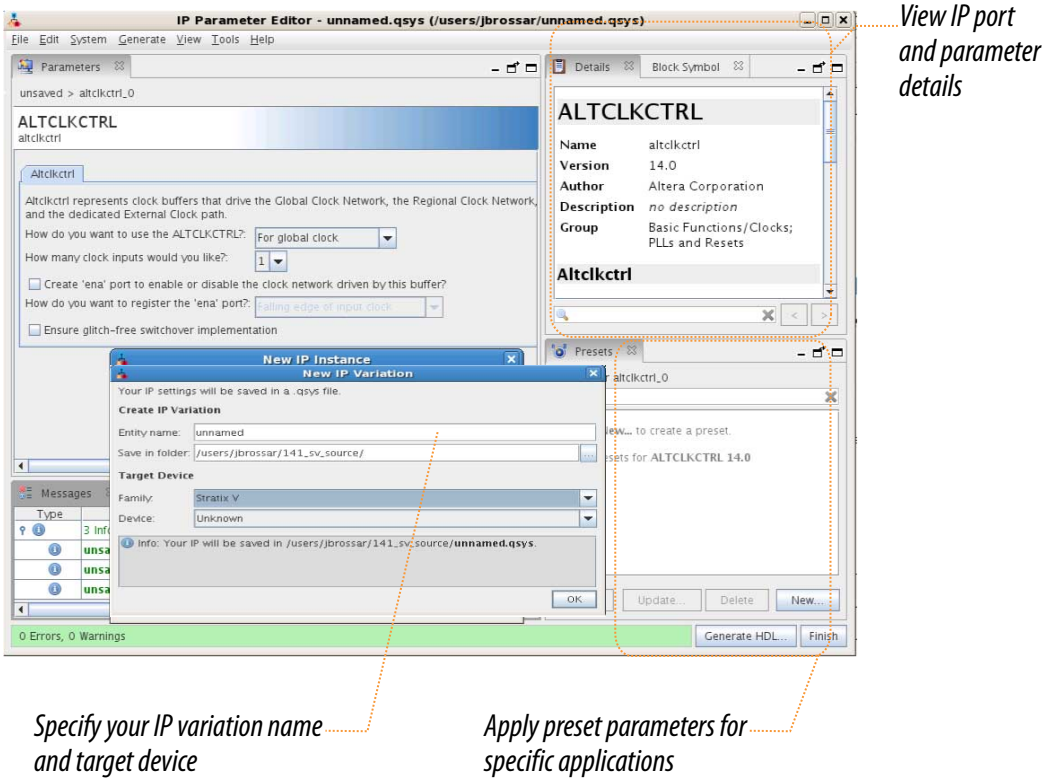
- **IP User Guide Documentation**
- **Altera IP Release Notes**

Specifying IP Core Parameters and Options

You can quickly configure a custom IP variation in the parameter editor. Use the following steps to specify IP core options and parameters in the parameter editor. Refer to *Specifying IP Core Parameters and Options (Legacy Parameter Editors)* for configuration of IP cores using the legacy parameter editor.

1. In the IP Catalog (**Tools** > **IP Catalog**), locate and double-click the name of the IP core to customize. The parameter editor appears.
2. Specify a top-level name for your custom IP variation. The parameter editor saves the IP variation settings in a file named <*your_ip*>**.qsys**. Click **OK**.
3. Specify the parameters and options for your IP variation in the parameter editor, including one or more of the following. Refer to your IP core user guide for information about specific IP core parameters.
   - Optionally select preset parameter values if provided for your IP core. Presets specify initial parameter values for specific applications.
   - Specify parameters defining the IP core functionality, port configurations, and device-specific features.
   - Specify options for processing the IP core files in other EDA tools.
4. Click **Generate HDL**, the **Generation** dialog box appears.
5. Specify output file generation options, and then click **Generate**. The IP variation files generate according to your specifications.
6. To generate a simulation testbench, click **Generate** > **Generate Testbench System**.
7. To generate an HDL instantiation template that you can copy and paste into your text editor, click **Generate** > **HDL Example**.
8. Click **Finish**. The parameter editor adds the top-level **.qsys** file to the current project automatically. If you are prompted to manually add the **.qsys** file to the project, click **Project** > **Add/Remove Files in Project** to add the file.
9. After generating and instantiating your IP variation, make appropriate pin assignments to connect ports.

Figure 4-1: IP Parameter Editor



Files Generated for Altera IP Cores

The Quartus II software generates the following IP core output file structure:

Figure 4-2: IP Core Generated Files

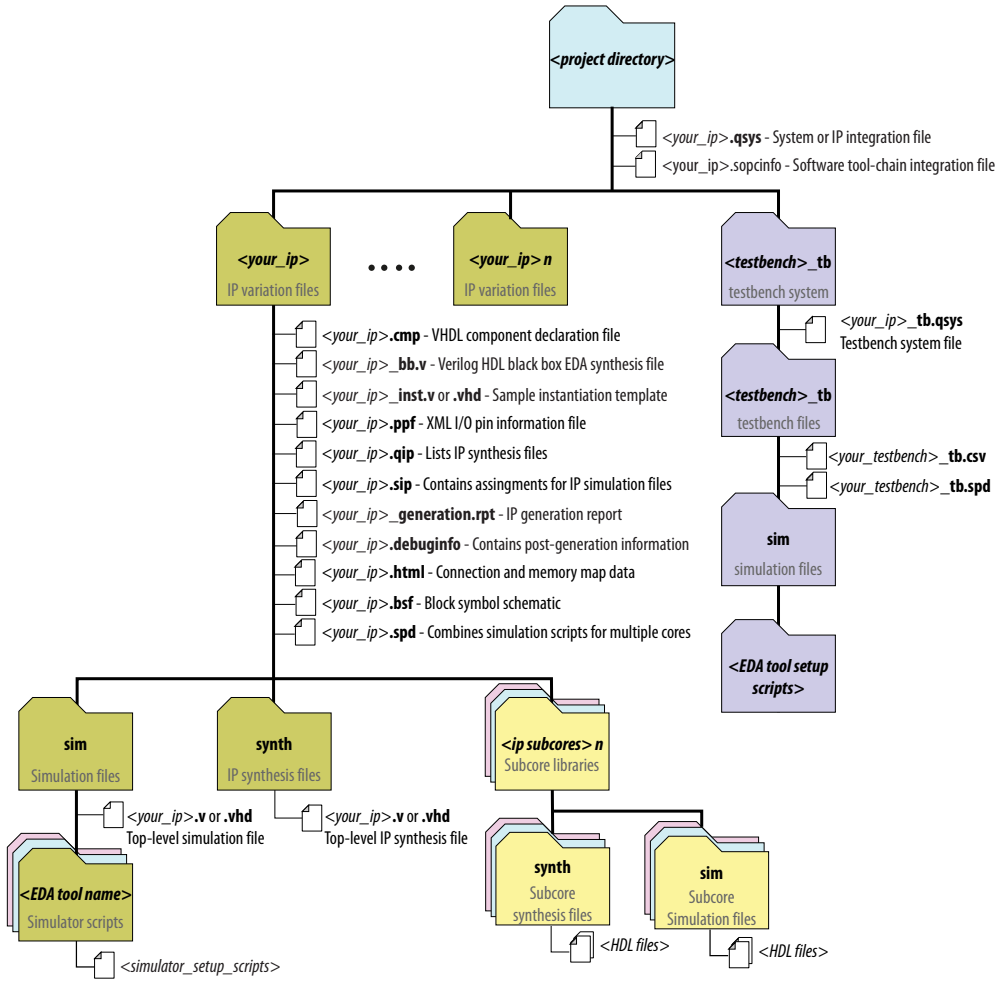


Table 4-1: IP Core Generated Files

File Name	Description
<*my_ip*>.qsys	The Qsys system or top-level IP variation file. <*my_ip*> is the name that you give your IP variation.
<*system*>.sopcinfo	Describes the connections and IP component parameterizations in your Qsys system. You can parse its contents to get requirements when you develop software drivers for IP components. Downstream tools such as the Nios II tool chain use this file. The **.sopcinfo** file and the **system.h** file generated for the Nios II tool chain include address map information for each slave relative to each master that accesses the slave. Different masters may have a different address map to access a particular slave component.

File Name	Description
<*my_ip*>.cmp	The VHDL Component Declaration (**.cmp**) file is a text file that contains local generic and port definitions that you can use in VHDL design files.
<*my_ip*>.html	A report that contains connection information, a memory map showing the address of each slave with respect to each master to which it is connected, and parameter assignments.
<*my_ip*>_generation.rpt	IP or Qsys generation log file. A summary of the messages during IP generation.
<*my_ip*>.debuginfo	Contains post-generation information. Used to pass System Console and Bus Analyzer Toolkit information about the Qsys interconnect. The Bus Analysis Toolkit uses this file to identify debug components in the Qsys interconnect.
<*my_ip*>.qip	Contains all the required information about the IP component to integrate and compile the IP component in the Quartus II software.
<*my_ip*>.csv	Contains information about the upgrade status of the IP component.
<*my_ip*>.bsf	A Block Symbol File (**.bsf**) representation of the IP variation for use in Quartus II Block Diagram Files (**.bdf**).
<*my_ip*>.spd	Required input file for `ip-make-simscript` to generate simulation scripts for supported simulators. The **.spd** file contains a list of files generated for simulation, along with information about memories that you can initialize.
<*my_ip*>.ppf	The Pin Planner File (**.ppf**) stores the port and node assignments for IP components created for use with the Pin Planner.
<*my_ip*>_bb.v	You can use the Verilog black-box (**_bb.v**) file as an empty module declaration for use as a black box.
<*my_ip*>.sip	Contains information required for NativeLink simulation of IP components. You must add the **.sip** file to your Quartus project.
<*my_ip*>_inst.v or **_inst.vhd**	HDL example instantiation template. You can copy and paste the contents of this file into your HDL file to instantiate the IP variation.
<*my_ip*>.regmap	If the IP contains register information, the .**regmap** file generates. The .**regmap** file describes the register map information of master and slave interfaces. This file complements the .**sopcinfo** file by providing more detailed register information about the system. This enables register display views and user customizable statistics in System Console.

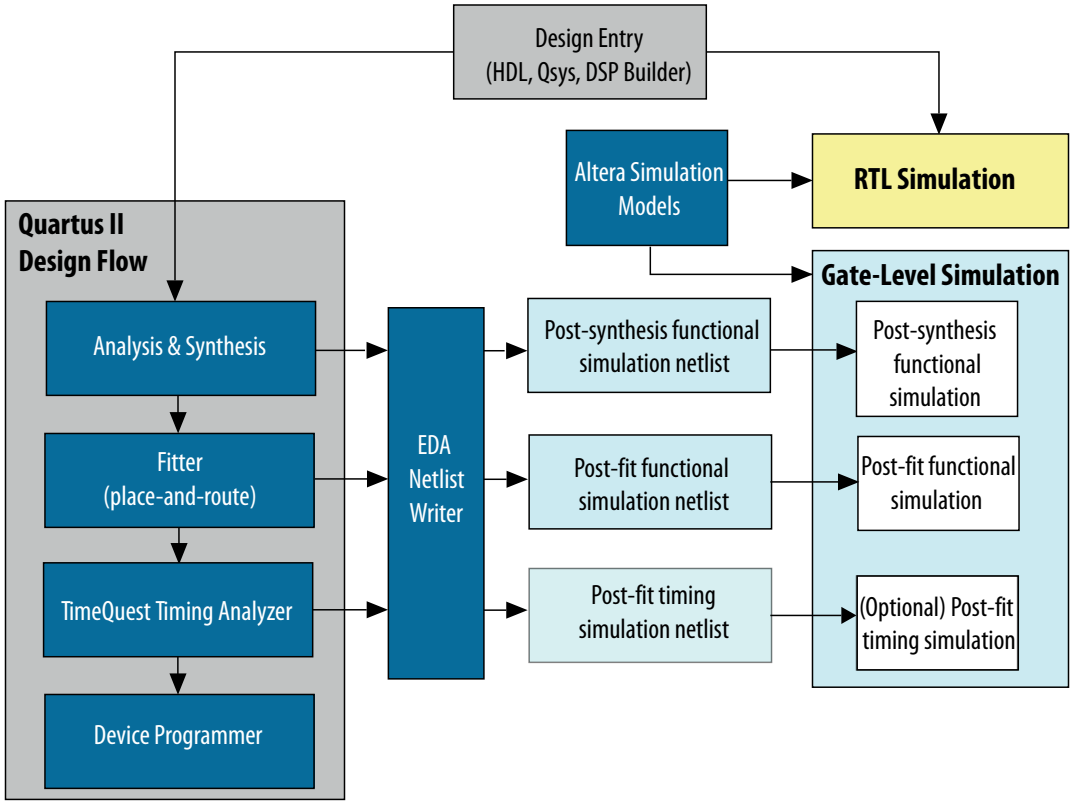
File Name	Description
<*my_ip*>.svd	Allows HPS System Debug tools to view the register maps of peripherals connected to HPS within a Qsys system. During synthesis, the **.svd** files for slave interfaces visible to System Console masters are stored in the **.sof** file in the debug section. System Console reads this section, which Qsys can query for register map information. For system slaves, Qsys can access the registers by name.
<*my_ip*>.v or **<*my_ip*>.vhd**	HDL files that instantiate each submodule or child IP core for synthesis or simulation.
mentor/	Contains a ModelSim® script **msim_setup.tcl** to set up and run a simulation.
aldec/	Contains a Riviera-PRO script **rivierapro_setup.tcl** to setup and run a simulation.
/synopsys/vcs **/synopsys/vcsmx**	Contains a shell script **vcs_setup.sh** to set up and run a VCS® simulation. Contains a shell script **vcsmx_setup.sh** and **synopsys_ sim.setup** file to set up and run a VCS MX® simulation.
/cadence	Contains a shell script **ncsim_setup.sh** and other setup files to set up and run an NCSIM simulation.
/submodules	Contains HDL files for the IP core submodule.
<*child IP cores*>/	For each generated child IP core directory, Qsys generates **/synth** and **/sim** sub-directories.

Simulating Altera IP Cores in other EDA Tools

The Quartus II software supports RTL and gate-level design simulation of Altera IP cores in supported EDA simulators. Simulation involves setting up your simulator working environment, compiling simulation model libraries, and running your simulation.

You can use the functional simulation model and the testbench or example design generated with your IP core for simulation. The functional simulation model and testbench files are generated in a project subdirectory. This directory may also include scripts to compile and run the testbench. For a complete list of models or libraries required to simulate your IP core, refer to the scripts generated with the testbench. You can use the Quartus II NativeLink feature to automatically generate simulation files and scripts. NativeLink launches your preferred simulator from within the Quartus II software.

Figure 4-3: Simulation in Quartus II Design Flow



Note: Post-fit timing simulation is supported only for Stratix IV and Cyclone IV devices in the current version of the Quartus II software. Altera IP supports a variety of simulation models, including simulation-specific IP functional simulation models and encrypted RTL models, and plain text RTL models. These are all cycle-accurate models. The models support fast functional simulation of your IP core instance using industry-standard VHDL or Verilog HDL simulators. For some cores, only the plain text RTL model is generated, and you can simulate that model. Use the simulation models only for simulation and not for synthesis or any other purposes. Using these models for synthesis creates a nonfunctional design.

Related Information

Simulating Altera Designs

UFM Avalon-MM Operating Modes

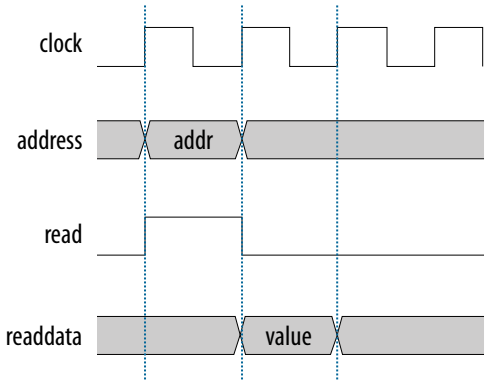
The UFM operating modes use Avalon-MM interface.

UFM Read Status and Control Register

You can access the control register value through the Avalon-MM control slave interface.

Figure 4-4: Read Status and Control Register

The figure below shows the timing diagram for the read status and control register.



To use the control register, assert the `read` signal and send the control register address to the control slave address.

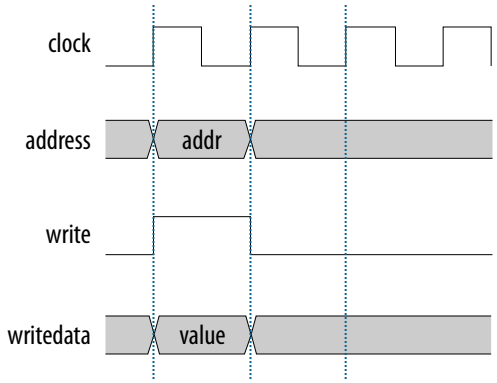
The flash IP core then sends the register value through the `readdata` bus.

UFM Write Control Register

You can program (write) the control register value through Avalon-MM control slave interface.

Figure 4-5: Program (Write) Control Register

The figure below shows the timing diagram for the program control register.



To program the control register, assert the `write` signal.

The flash IP core then sends address `0x01` (control register) and `writedata` (register value) to control the slave interface.

UFM Program (Write) Operation

The UFM offers a single 32-bit program (write) operation.

To perform a UFM program operation, follow these steps:

1. Disable the write protection mode. Write 0 into the write protection register for the sector of the given data through the Avalon-MM control interface.
2. Program the following data into flash through the Avalon-MM data interface.
   - Address: legal address (from Avalon-MM address map)
   - Data: user data

   Set burstcount to 1 (parallel mode) or 32 (serial mode).
3. The flash IP core sets the `busy` field in the status register to `2'b10` when the program operation is in progress.
4. If the operation goes well, the flash IP core sets the write successful field in the status register to `1'b1` or write successful. The flash IP core sets the write successful field in the status register to `1'b0` (failed) if one of the following conditions takes place:
   - The burst count is not equal to 1 (parallel mode) or 32 (serial mode).
   - The given address is out of range.
   - The sector protection mode or write protection mode of the corresponding sector is not clear (the value is not `1'b0`).
5. Repeat the earlier steps if you want to perform another program operation.
6. You have to enable back the write protection mode when the program operation completes. Write 1 into the write protection register for the corresponding sector through the Avalon-MM control interface.

Note: Check the status register after each write to make sure the program operation is successful (write successful).

Figure 4-6: Program Operation in Parallel Mode

The figure below shows the write data timing diagram in parallel mode.

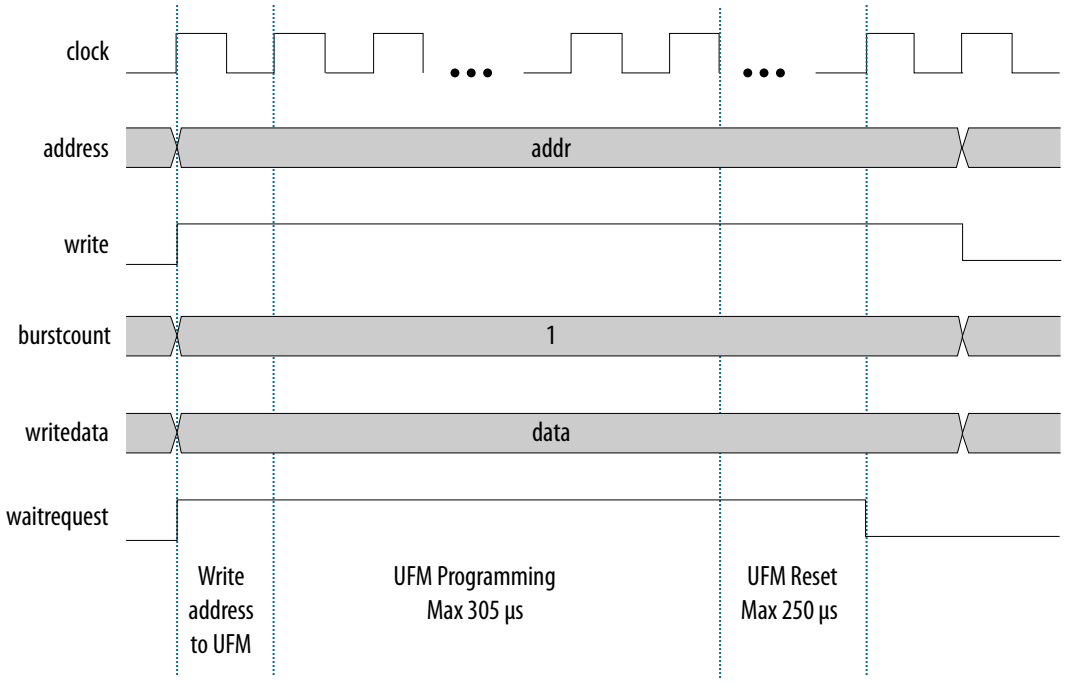
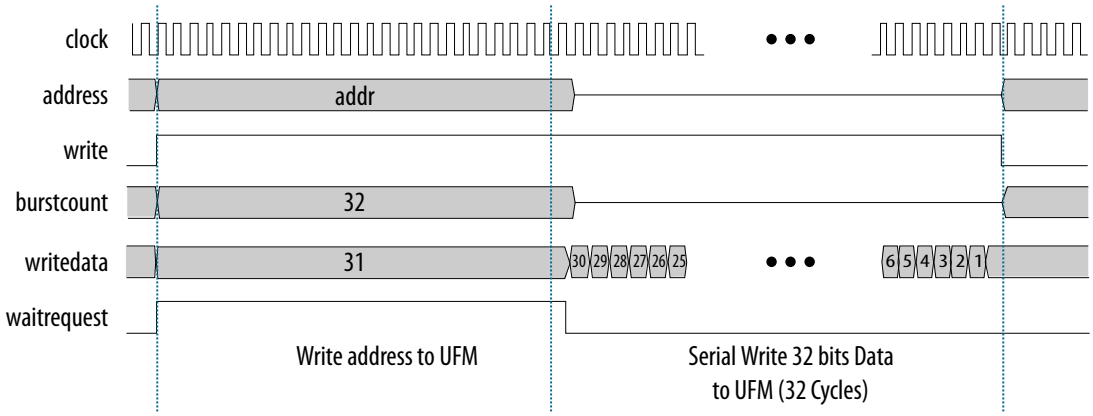


Figure 4-7: Program Operation in Serial Mode

The figure below shows the write data timing diagram in serial mode.



UFM Sector Erase Operation

The sector erase operation allows the UFM to erase by sectors.

To perform a UFM sector erase operation, follow these steps:

1. Disable the write protection mode. Write 0 into the write protection register for the sector through the Avalon-MM control interface.
2. Write the appropriate bits into the control register to select the sector erase location. The flash IP core stores the sector erase address and initiates the sector erase operation.

   Note: The IP core only accepts the sector erase address when it is in IDLE state;`busy` field at status register is`2'b00`. If the IP core is busy, it will ignore the sector erase address.
3. The flash IP core sets the `busy` field in the status register to `2'b01` when the erase operation is in progress.
4. The flash IP core then asserts the `waitrequest` signal if there are any new incoming read or write commands from the data interface.
5. The flash IP core erases the sector. It stores the physical flash erase result in the erase successful field in the status register when the sector erase operation completes.

   Note: The maximum erase time is 350 ms.
6. The flash IP core sets the erase successful field in the status register to `1'b0` (failed) if one of the following conditions takes place:
   - You send an illegal sector number.
   - The sector protection mode or write protection mode of the corresponding sector is not clear (the value is not `1'b0`)
7. Repeat the earlier steps if you want to perform another sector erase operation.
8. You have to enable back the write protection mode when the sector erase operation completes. Write 1 into the write protection register for the corresponding sector through the Avalon-MM control interface.

   Note: Check the status register after each erase to make sure the erase operation is successful (erase successful).

UFM Page Erase Operation

The page erase operation allows the UFM to erase by pages.

To perform a UFM page erase operation, follow these steps:

1. Disable the write protection mode. Write 0 into the write protection register for the sector through the Avalon-MM control interface.
2. Write the appropriate bits into the control register to select the page erase location. The flash IP core stores the page erase address and initiates the page erase operation.

   Note: The IP core only accepts the page erase address when the IP is in IDLE state; `busy` field at status register is`2'b00`. If the IP core is busy, it will ignore the page erase address.
3. The flash IP core sets the `busy` field in the status register to `2'b01` when the erase operation is in progress.
4. The flash IP core then asserts the `waitrequest` signal if there are any new incoming read or write commands from the data interface.
5. The flash IP core erases the page. It stores the physical flash erase result in the erase successful field in the status register when the page erase operation completes.

   Note: The maximum erase time is 350 ms.
6. The flash IP core sets the erase successful field in the status register to `1b'0` (failed) if you send an illegal address.
7. Repeat the earlier steps if you want to perform another page erase operation.
8. You have to enable back the write protection mode when the page erase operation completes. Write 1 into the write protection register for the corresponding page through the Avalon-MM control interface.

   Note: Check the status register after each erase to make sure the erase operation is successful (erase successful).

UFM Read Operation

The UFM offers a single 32-bit read operation.

To perform a read operation, the address register must be loaded with the reference address where the data is or is going to be located in the UFM.

To perform a UFM read operation, follow these steps:

1. Assert the `read` signal to send the legal data address to the data slave interface.
2. Set the burstcount to 1 (parallel mode) or 32 (serial mode).
3. The flash IP core asserts the `waitrequest` signal when it is busy.
4. The flash IP core asserts the `readdatavalid` signal and sends the data through the `readdata` bus.
5. The flash IP core sets the `busy` field in the status register to `2'b11` when the read operation is in progress.
6. If the operation goes well, the flash IP core sets the read successful field in the status register to `1'b1` or read successful. It sets the read successful field in the status register to `1'b0` (failed) and returns empty flash if you try to read from an illegal address or protected sector.

The following figures show the timing diagrams for the read operations for the different MAX 10 devices in parallel and serial modes.

Figure 4-8: Read Operation for 10M04, 10M08, 10M16 and 10M25 Devices in Parallel Mode

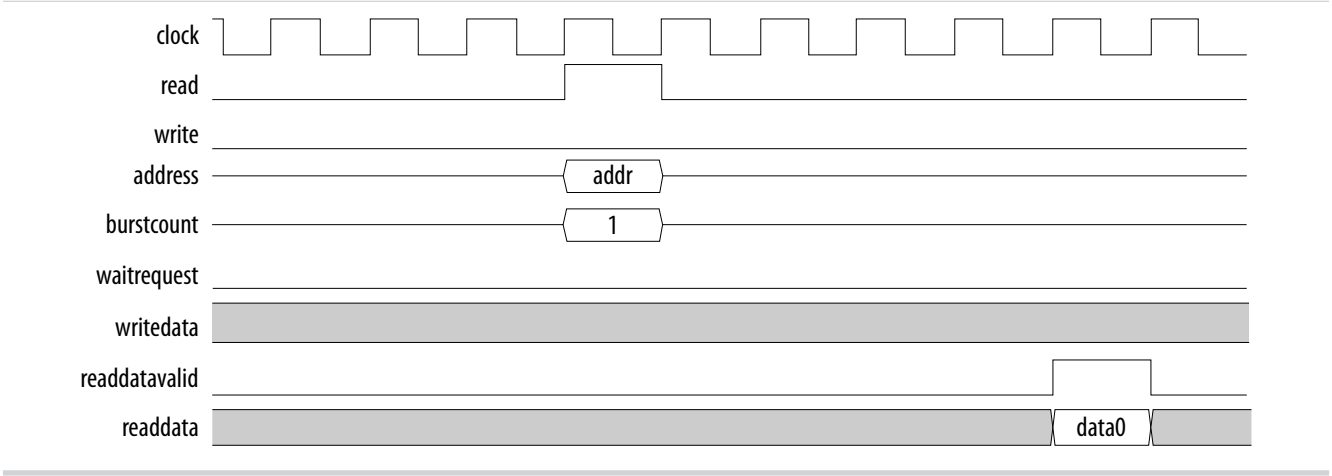


Figure 4-9: Read Operation for 10M40 and 10M50 Devices in Parallel Mode

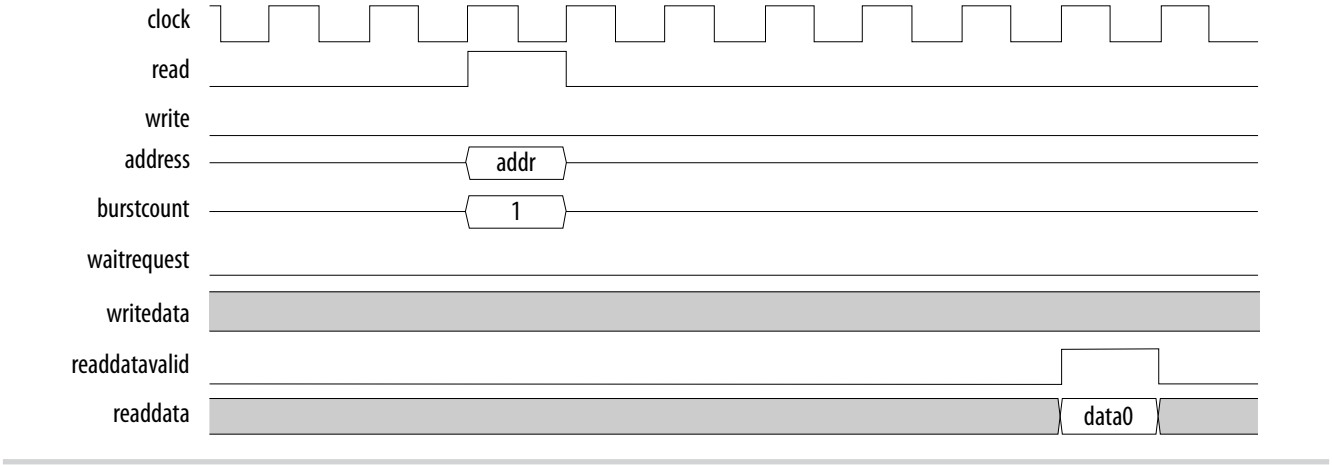
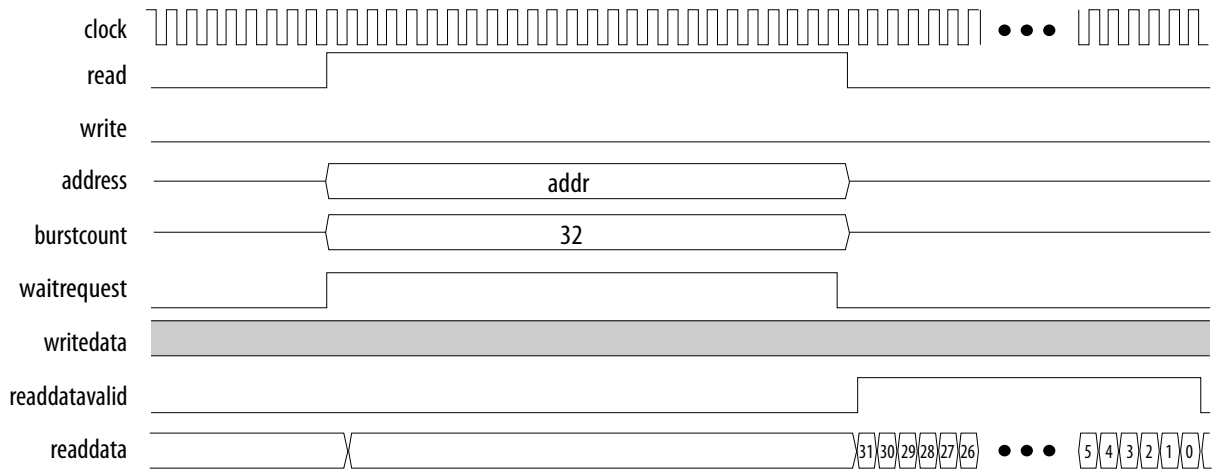


Figure 4-10: Read Operation for MAX 10 Devices in Serial Mode



UFM Burst Read Operation

The burst read operation is a streaming 32-bit read operation.

The burst read operation offers the following modes:

- Data incrementing burst read—allows a maximum of 128 burst counts.
- Data wrapping burst read—has fixed burst counts of 2 (10M04/08) and 4 (10M16/25/40/50)

To perform a UFM burst read operation, follow these steps:

1. Assert the `read` signal and send the legal burstcount and legal data addresses to the data interface.
2. The flash IP core asserts the `waitrequest` signal when it is busy.
3. The flash IP core then asserts the `readdatavalid` signal and sends the data through the `readdata` bus.

   Note: For data wrapping burst read operation, if the address reaches the end of the flash, it wraps back to the beginning of the flash and continues reading.
4. The flash IP core sets the `busy` field in the status register to `2'b11` or `busy_read` when the read operation is in progress.
5. If the operation goes well, the flash IP core sets the read successful field in the status register to `1'b1` or read successful. It sets the read successful field in the status register to `1'b0` (failed) and changes all empty flash to 1 if you try to read from an illegal address or protected sector.

UFM Data Incrementing Burst Read

The following figures show the timing diagrams for the data incrementing burst read operations for the different MAX 10 devices.

Figure 4-11: Incrementing Burst Read Operation for 10M04 and 10M08 Devices in Parallel Mode

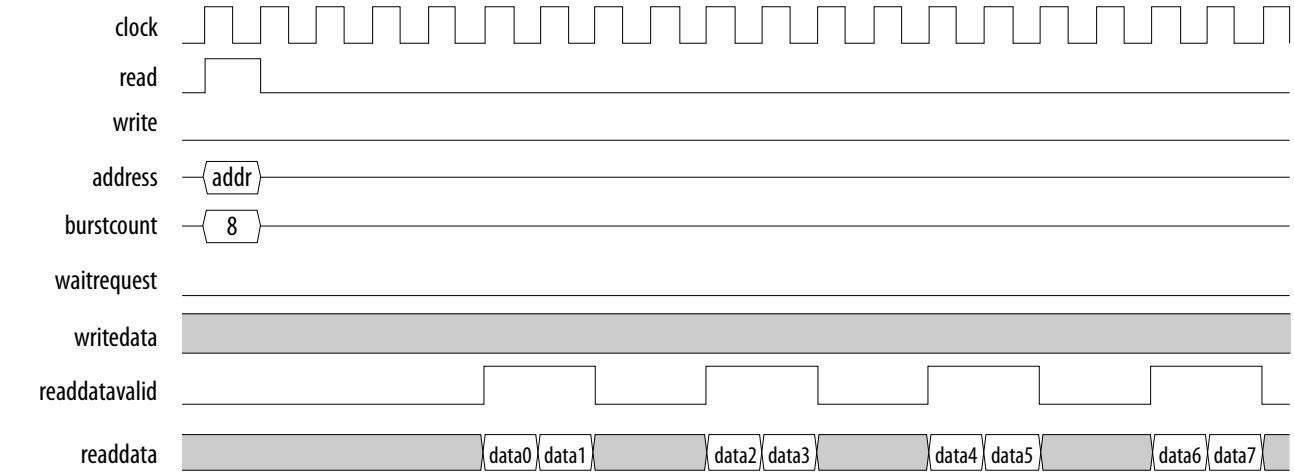


Figure 4-12: Incrementing Burst Read Operation for 10M16 and 10M25 Devices in Parallel Mode

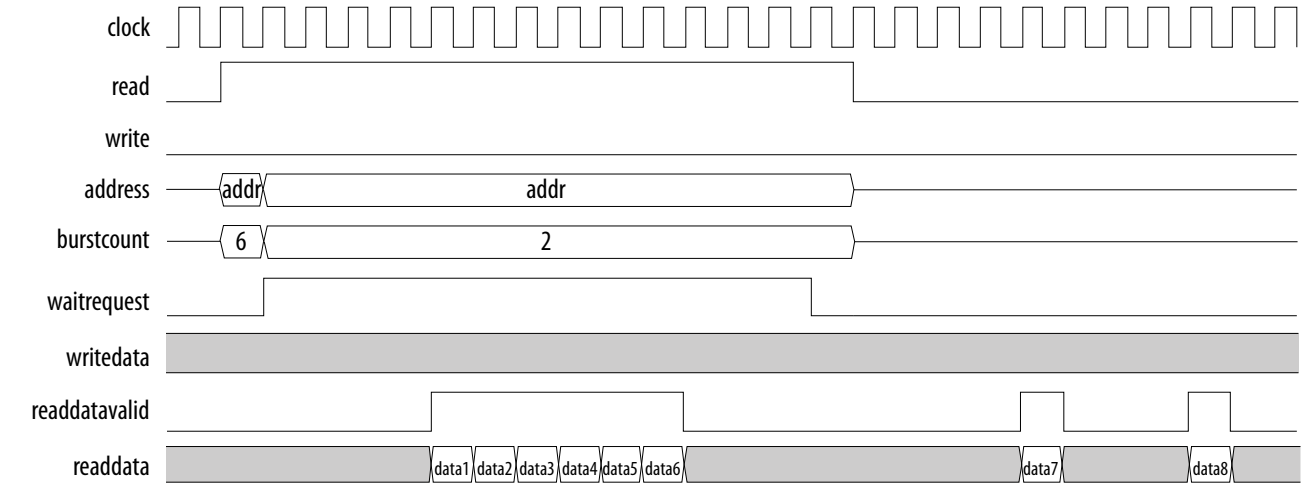


Figure 4-13: Incrementing Burst Read Operation for 10M50 Devices in Parallel Mode

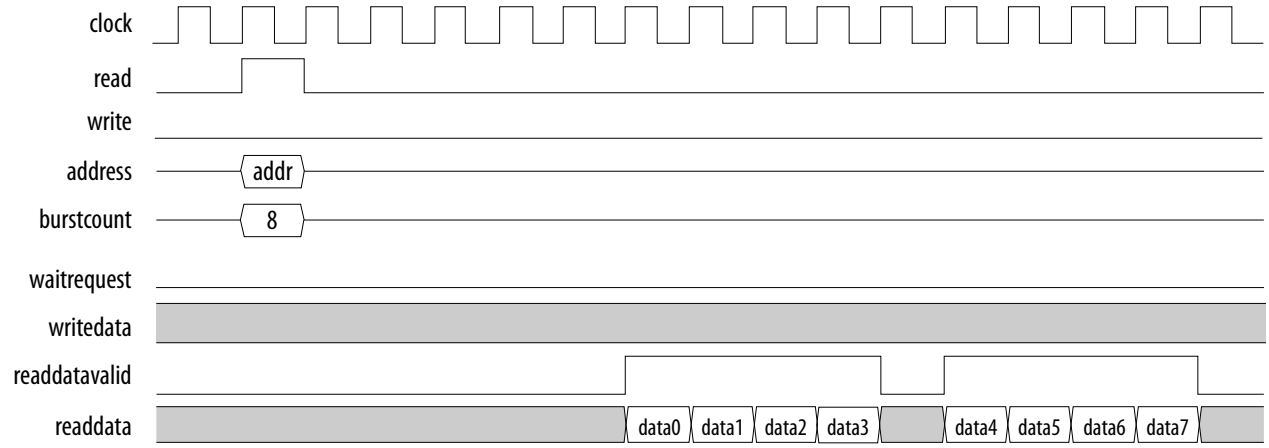


Figure 4-14: Unaligned Address Incrementing Burst Read Operation for 10M50 Devices in Parallel Mode

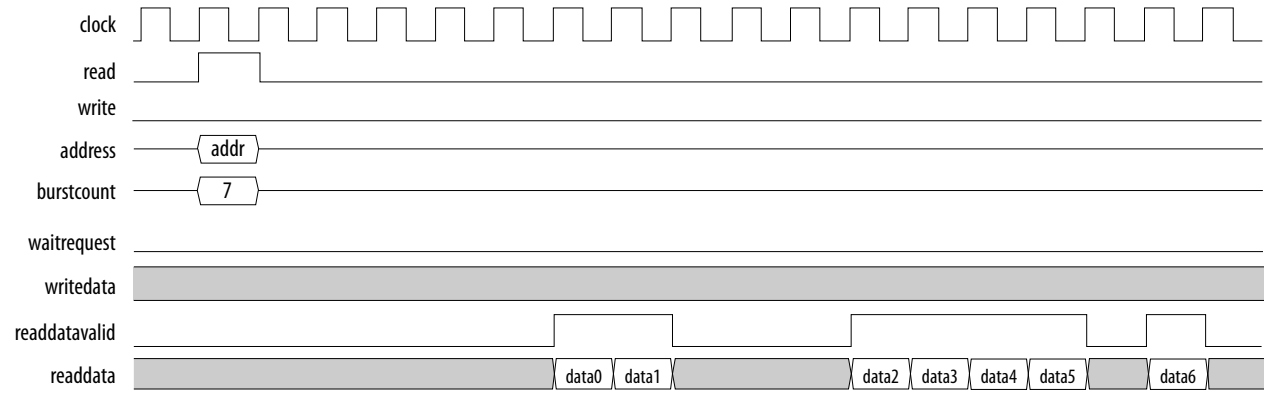
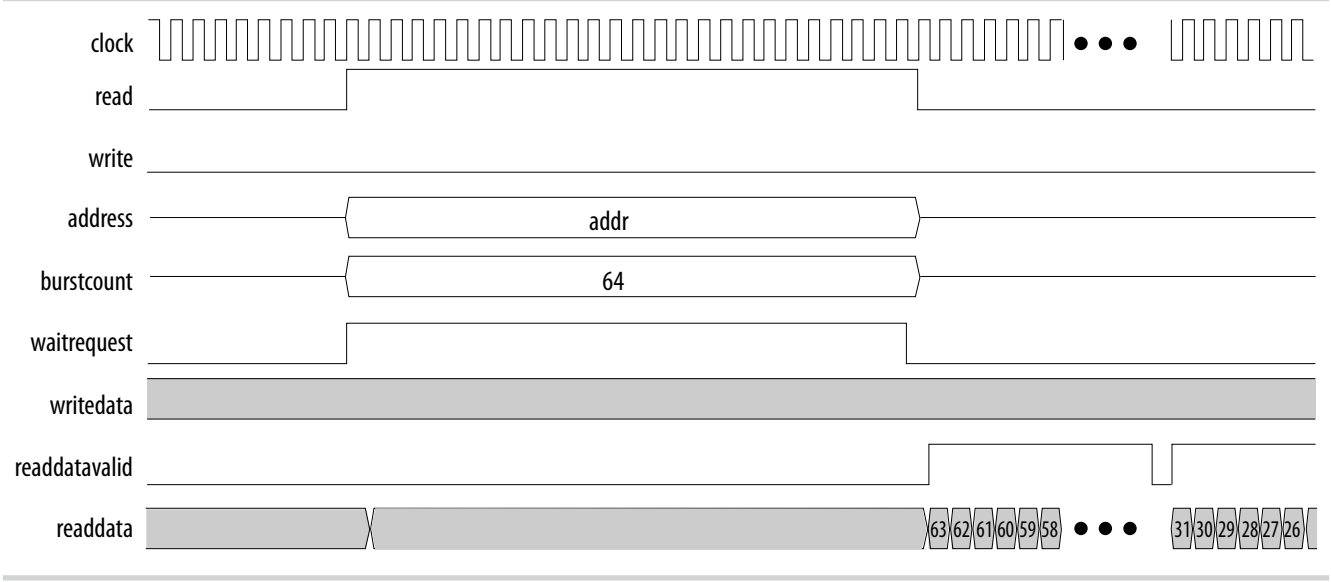


Figure 4-15: Incrementing Burst Read Operation for MAX 10 Devices in Serial Mode



UFM Data Wrapping Burst Read

The UFM IP supports data wrapping when it receives an unaligned address.

Note: Wrapping burst read is available only for parallel interface.

Table 4-2: Data Wrapping Support for MAX 10 Devices

Device	Data Register Length	Flash IP Data Bus Width	Fixed Supported Burst Count	Data Wrapping
10M04, or 10M08	32	64	2	The address wraps back to the previous boundary after 64 bits or 2 cycles. For example, for a wrapping in a 32-bit data interface: 1. Start address is 0×01 2. Address sequence will be 0×01, then back to address 0×00
10M16, 10M25, 10M40, or 10M50	32	128	4	The address wraps back to the previous boundary after 128 bits or 4 cycles. For example, for a wrapping in a 32-bit data interface: 1. Start address is 0×02 2. Address sequence will be 0×02 and 0×03, then back to address 0×00 and 0×01

The following figures show the timing diagrams for the data wrapping burst read operations for the different MAX 10 devices.

Figure 4-16: Wrapping Burst Read Operation for 10M04 and 10M08 Devices

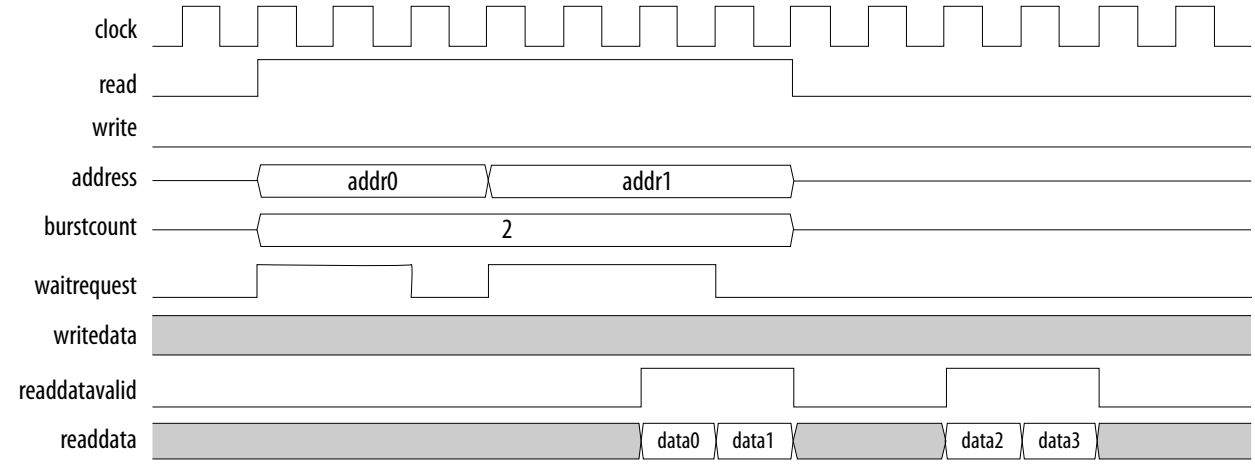


Figure 4-17: Wrapping Burst Read Operation for 10M16 and 10M25 Devices

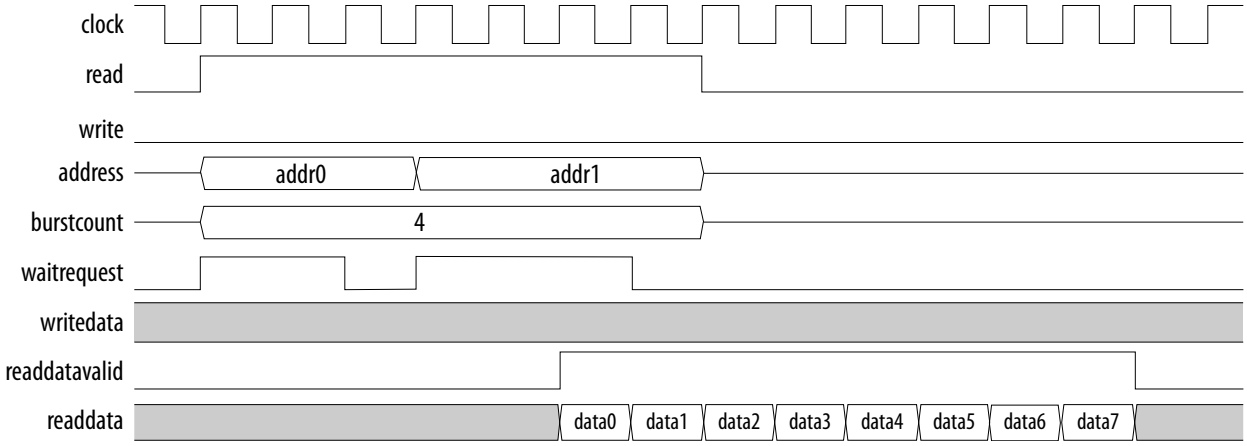
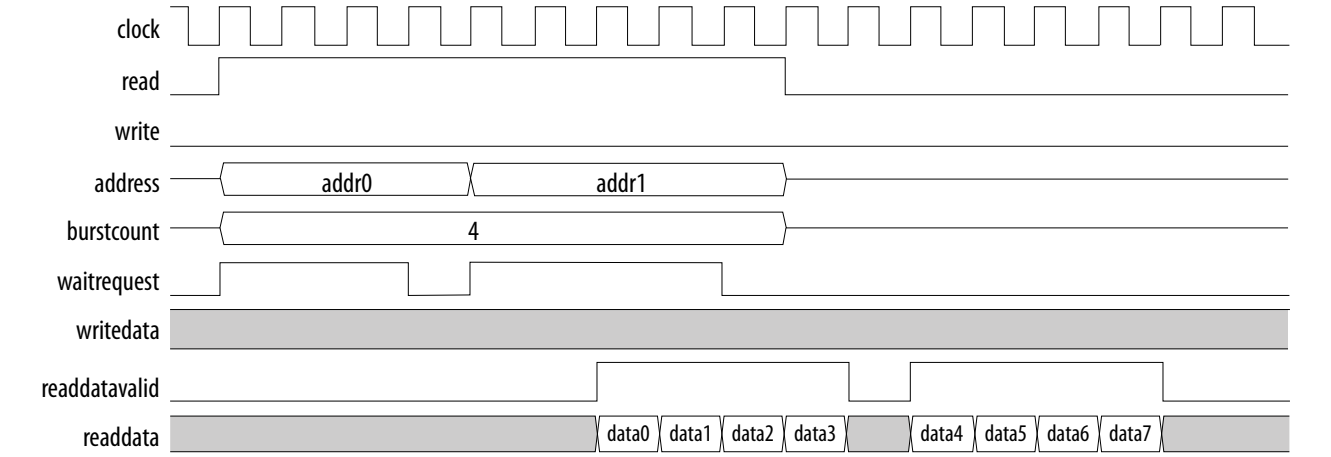


Figure 4-18: Wrapping Burst Read Operation for 10M40 and 10M50 Devices



Altera On-Chip Flash IP Core References

5

2015.05.04

UG-M10UFM

This section provides information about the Altera On-Chip Flash IP Core parameters, signals, and registers.

Altera On-Chip Flash Parameters

The following table lists the parameters for the Altera On-Chip Flash IP core.

Table 5-1: Altera On-Chip Flash IP Core Parameters

Parameters	Default Value	Description
Data interface	Parallel	Allows you to select the type of interface. You can choose parallel or serial. **Note:** 10M02 devices support only serial interface.
Read burst mode	Incrementing	Allows you to select the type of read burst mode. You can choose incrementing or wrapping. Incrementing mode: Burstcount range is 1, 2, 4, 7, ... 128 Wrapping mode: Burstcount fixed to 2 or 4 **Note:** Serial mode supports only incrementing mode.
Read burst count	2	Allows you the flexibility to adjust the burst count bus width. • Parallel mode: This setting represents the maximum burst count number. • Serial mode: This setting supports stream read and represents the words to be read for each read operation. The Avalon-MM interface burst count bus width is equal to 32*read burst count.

Parameters	Default Value	Description
Configuration mode	Single uncompressed image	Allows you to select the configuration mode. You can choose one of these options: • Dual compressed images • Single uncompressed image: Accesses CFM2 sector as UFM • Single compressed image: Accesses CFM2 and CFM1 sectors as UFM • Single uncompressed image with memory initialization • Single compressed image with memory initialization
Flash Memory	—	The sector ID, address range value, and flash type are generated dynamically by hardware **.tcl** based on the device and configuration mode you select. Indicates the address mapping for each sector and adjusts the **Access Mode** for each sector individually. **Note:** Only CFM sectors support **Hidden** access mode.
Clock frequency	116.0 MHz	Key in the appropriate clock frequency in MHz. The maximum frequency is 116.0 MHz for parallel interface and 7.25 MHz for serial interface.
Initialize flash content	Off	Turn on this option to initialize the flash content.
Enable non-default initialization file	Off	Turn on this option to enable your preferred initialization file. If you choose to have a non-default file, type the filename or select the **.hex** file using the browse button.
User created hex or mif file	—	This option is only available if you turn on **Enable non-default initialization file**. Assign your own **.hex** or **.mif** filename.
User created dat file for simulation	—	This option is only available if you turn on **Enable non-default initialization file**. Assign your own simulation filename.

Altera On-Chip Flash Signals

The following table lists the signals for the Altera On-Chip Flash IP core.

Table 5-2: Avalon-MM Slave Input and Output Signals for Parallel and Serial Modes.

Signal	Width	Direction	Description
Clock and Reset			
`clock`	1	Input	System clock signal that clocks the entire peripheral.

Signal	Width	Direction	Description
reset_n	1	Input	System synchronous reset signal that resets the entire peripheral. The IP core asserts this signal asynchronously. This signal becomes synchronous in the IP core after the rising edge of the clock.
Control			
avmm_csr_addr	1	Input	Avalon-MM address bus that decodes registers.
avmm_csr_read	1	Input	Avalon-MM read control signal. The IP core asserts this signal to indicate a read transfer. If present, the readdata signal is required.
avmm_csr_readdata	32	Output	Avalon-MM read back data signal. The IP core asserts this signal during read cycles.
avmm_csr_write	1	Input	Avalon-MM write control signal. The IP core asserts this signal to indicate a write transfer. If present, the writedata signal is required.
avmm_csr_writedata	32	Input	Avalon-MM write data bus. The bus master asserts this bus during write cycles.
Data			
avmm_data_addr	User-defined	Input	Avalon-MM address bus that indicates the flash data address. The width of this address depends on your selection of device and configuration mode.
avmm_data_read	1	Input	Avalon-MM read control signal. The IP core asserts this signal to indicate a read transfer. If present, the readdata signal is required.
avmm_data_readdata	• Parallel mode: 32 • Serial mode: 1	Output	Avalon-MM read back data signal. The IP core asserts this signal during read cycles.
avmm_data_write	1	Input	Avalon-MM write control signal. The IP core asserts this signal to indicate a write transfer. If present, the writedata signal is required.
avmm_data_writedata	• Parallel mode: 32 • Serial mode: 1	Input	Avalon-MM write data bus. The bus master asserts this bus during write cycles.
avmm_data_waitrequest	1	Output	The IP core asserts this bus to pause the master when the IP core is busy during read or write operations.
avmm_data_readdata-valid	1	Output	The IP core asserts this signal when the readdata signal is valid during read cycles.

Signal	Width	Direction	Description
avmm_data_burstcount	User-defined	Input	The bus master asserts this signal to initiate a burst read operation. • In write operations, the burstcount is always fixed to 1 for parallel mode and 32 for serial mode. • In incrementing burst read mode, the supported read burstcount range: Parallel mode: 1-2^(burstcount width-1) Serial mode: 1-128*32 • In wrapping burst read mode (parallel mode only), the supported read burstcount is fixed to 2 and 4. 10M04, and 10M08: 2 10M16, 10M25, 10M40 and 10M50: 4

Altera On-Chip Flash Registers

The following table lists the address mapping and registers for the Altera On-Chip Flash IP core.

Table 5-3: Altera On-Chip Flash Control Address Mapping

Register	Address	Access	Description
Status Register	0×00	Read only	Stores the status and result of recent operations and sector protection mode.
Control Register	0×01	Read/Program	Stores the following information: • Page erase address • Sector erase address • Sector write protection mode

Table 5-4: Altera On-Chip Flash Status Register

Bit Offset	Field	Default Value	Description
1–0	busy	2'b00	2'b00 IDLE 2'b01 BUSY_ERASE 2'b10 BUSY_WRITE 2'b11 BUSY_READ
2	rs (read successful)	1'b0	1'b0 Read failed 1'b1 Read successful

Bit Offset	Field	Default Value	Description
3	ws (write successful)	1'b0	1'b0 Write failed 1'b1 Write successful
4	es (erase successful)	1'b0	1'b0 Erase failed 1'b1 Erase successful
5	sp (UFM1 protection bit)	—	The IP core sets these bits based on the specified device and configuration mode. If the IP core sets one of these bits, you cannot read or program on the specified sector.
6	sp (UFM0 protection bit)	—	
7	sp (CFM2 protection bit)	—	
8	sp (CFM1 protection bit)	—	
9	sp (CFM0 protection bit)	—	
31–10	dummy (padding)	—	All of these bits are set to 1.

Table 5-5: Altera On-Chip Flash Control Register

Bit Offset	Field	Default Value	Description
19–0	pe (page erase address)	All 1's	Sets the page erase address to initiate a page erase operation. The IP core only accepts the page erase address when it is in IDLE state. Otherwise, the page address will be ignored. The legal value is any available address. The IP core erases the corresponding page of the given address.

<table>
<tr><th>Bit Offset</th><th>Field</th><th>Default Value</th><th>Description</th></tr>
<tr><td>22–20</td><td>se (sector erase address)</td><td>3'b111</td><td>Sets the sector erase address to initiate a sector erase operation. The IP core only accepts the sector erase address when it is in IDLE state. Otherwise, the page address will be ignored.<table><tr><td>3'b001</td><td>UFM1</td></tr><tr><td>3'b010</td><td>UFM0</td></tr><tr><td>3'b011</td><td>CFM2</td></tr><tr><td>3'b100</td><td>CFM1</td></tr><tr><td>3'b101</td><td>CFM0</td></tr><tr><td>3'b111</td><td>Not set</td></tr><tr><td>Other values</td><td>Illegal address</td></tr></table>Note: If you set both sector address and page address at the same time, the sector erase address gets the priority. The IP core accepts and executes the sector erase address and ignores the page erase address.</td></tr>
<tr><td>23</td><td>wp (UFM1 write protection)</td><td>1</td><td rowspan="5">The IP core uses these bits to protect the sector from write and erase operation. You must clear the corresponding sector write protection bit before your program or erase the sector.<table><tr><td>1'b0</td><td>Disable write protected mode</td></tr><tr><td>1'b1</td><td>Enable write protected mode</td></tr></table></td></tr>
<tr><td>24</td><td>wp (UFM0 write protection)</td><td>1</td></tr>
<tr><td>25</td><td>wp (CFM2 write protection)</td><td>1</td></tr>
<tr><td>26</td><td>wp (CFM1 write protection)</td><td>1</td></tr>
<tr><td>27</td><td>wp (CFM0 write protection)</td><td>1</td></tr>
<tr><td>31–28</td><td>dummy (padding)</td><td>—</td><td>All of these bits are set to 1.</td></tr>
</table>

Additional Information for MAX 10 UFM User Guide

A

2015.05.04

UG-M10UFM

Subscribe

Send Feedback

Document Revision History for Content MAX 10 User Flash Memory User Guide

Date	Version	Changes
May 2014	2015.05.04	• Changed *write* to industry-standard term *program*. • Added a note to the *UFM and CFM Array Size* section that the total UFM size is the maximum possible value, which is dependent on the selected mode. • Added design consideration information about the maximum slew rate requirement for power supply ramp down. • Added design consideration information about erasing the flash location before performing a program operation.

101 Innovation Drive, San Jose, CA 95134

Date	Version	Changes

December 2014	2014.12.15	• Added support for serial interface.
		• Added maximum operating frequency of 7.25 MHz for serial interface.
		• Updated the UFM block diagram to include serial interface.
		• Added design consideration information about creating initial memory content using the IP core, and programming UFM using JTAG interface version IEEE Standard 1149.1.
		• Added new timing diagrams for read and write operations in serial mode.
		• Added information for the new serial interface related GUI parameters, signals, and registers.
		• Added information for the following new Avalon-MM slave interface signals for serial mode: `addr`, `read`, `readdata`, `write`, `writedata`, `waitrequest`, `readdatavalid`, and `burstcount`.
		• Added information for the following new parameters:
		  • **Data Interface** that allows you to choose between **Parallel** and **Serial** interface.
		  • **Configuration Scheme** and **Configuration Mode** that replace **Dual Images**. The new parameters include all supported configuration modes.
		  • **Read Burst Count** that allows the burstcount width to be auto-adjusted.

September 2014	2014.09.22	Initial release.