Spartan-3AN FPGA In-System Flash User Guide

For Spartan®-3AN FPGA applications that read or write data to or from the In-System Flash memory after configuration

UG333 (v2.1) January 15, 2009

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
02/26/07	1.0	Initial release.
08/29/07	1.1	Updated throughout. Removed legacy commands not recommended for designs. Added “MultiBoot Configuration Bitstream Guidelines”. Simplified discussion in Chapter 7, “Power Management”.
09/24/07	1.1.1	Added Caution to Chapter 1, “Overview and SPI_ACCESS Interface,”, that SPI_ACCESS is not currently supported in simulation.
04/22/08	2.0	Updated to reflect current simulation model and timing characteristics.
01/15/09	2.1	Added “Related Materials and References,” page 14 to Chapter 1, “Overview and SPI_ACCESS Interface”. Updated and clarified default SIM_FACTORY_ID in Table 1-3. Corrected highest byte number in Figure 2-5.

Table of Contents

Chapter 9: Security Register

Appendix A: Optional Power-of-2 Addressing Mode

Chapter 1

Overview and SPI_ACCESS Interface

Note: This user guide only applies to Spartan®-3AN FPGA designs that access or modify the in-system Flash ***after*** configuration. This user guide is not required for applications that only use the in-system Flash to configure the FPGA. For Spartan-3AN FPGA configuration information, see UG332: *Spartan-3 Generation Configuration User Guide.*

Spartan-3AN FPGAs include abundant In-System Flash (ISF) memory. The ISF memory array appears to a Spartan-3AN FPGA application as SPI-based serial Flash memory. The ISF memory is primarily designed to automatically configure the FPGA when power is applied or whenever the PROG_B pin is pulsed Low. However, the ISF memory array is large enough to store…

- two complete, uncompressed FPGA configuration bitstreams. Using the MultiBoot feature, the FPGA application can selectively choose between the two designs or reserve one image as a fail-safe image for live in-system Flash updates.
- additional nonvolatile data for the FPGA application, such as MicroBlaze™ processor code, serial numbers, Ethernet MAC IDs, graphic images, message templates, and so on.

In-System Flash Summary

Table 1-1, page 8 summarizes the key attributes and capabilities of the ISF memory. The remainder of this user guide describes these features and capabilities in greater detail.

The table also summarizes the amount of Flash memory available to the FPGA application, depending on the number of design options.

- How many FPGA configuration bitstreams are stored in the ISF array?
  - Most applications store a single FPGA configuration bitstream, leaving the remaining space for nonvolatile user data.
  - Optionally, each Spartan-3AN FPGA can store two uncompressed MultiBoot configuration images, which reduces the amount of Flash memory available to the application.
- Does the FPGA application use the ISF memory's Sector Protect or Sector Lockdown features to protect ISF memory contents?
  - Without using the sector-based data protection features, user application data can be stored in the next available page location following the FPGA bitstream (page aligned).
  - If the application uses the sector-based data protection features, then user application data is typically aligned to the next sector boundary (sector aligned).

Table 1-1: **In-System Flash Memory Summary**

Description		Spartan-3AN FPGA				
		3S50AN	3S200AN	3S400AN	3S700AN	3S1400AN
In-System Flash (ISF) memory bits		1,081,344 (1M+)	4,325,376 (4M+)	4,325,376 (4M+)	8,650,752 (8M+)	17,301,504 (16M+)
SRAM page buffers		1	2	2	2	2
Default Addressing Mode page size (bytes)		264	264	264	264	528
Optional Power-of-2 Addressing Mode page size (bytes)		256	256	256	256	512
Pages		512	2,048	2,048	4,096	4,096
Blocks		64	256	256	512	512
Sectors		4	8	8	16	16
Pages per Block		8	8	8	8	8
Pages per Sector		128	256	256	256	256
Bytes per Block		2,112	2,112	2,112	2,112	4,224
Bytes per Sector		33,792	67,584	67,584	67,584	135,168
FPGA configuration bitstream size (uncompressed)		437,312	1,196,128	1,886,560	2,732,640	4,755,296
Pages required for FPGA bitstream, always starting at page 0	Default	208	567	894	1,294	1,126
	Power-of-2	214	585	922	1,335	1,161
Page Aligned User Data (maximizes available data space but limits Sector Protect, Sector Lockdown features)						
Pages available for user application beyond FPGA configuration bitstream, data aligned to next page boundary, Default Addressing Mode		304	1,481	1,154	2,802	2,970
Total Flash memory bits available for user application, Default Addressing Mode		642,048 (627K) (0.61M)	3,127,872 (3,054K) (2.98M)	2,437,248 (2,380K) (2.32M)	5,917,824 (5,779K) (5.64M)	12,545,280 (12,251K) (11.96M)
Sector Aligned User Data (user data aligned to sectors for Sector Protect, Sector Lockdown features)						
Sectors required per uncompressed FPGA bitstream		2	3	4	6	5
Sectors available for user application beyond FPGA configuration bitstream, aligned to next sector boundary		2	5	4	10	11
Total bits available for user application in remaining sectors, Default Addressing Mode		540,672 (528K) (0.51M)	2,703,360 (2,640K) (2.57M)	2,162,688 (2,112K) (2.06M)	5,406,720 (5,280K) (5.15M)	11,894,784 (11,616K) (11.34M)
MultiBoot FPGA Configuration						
Maximum number of uncompressed MultiBoot FPGA configuration images		2	2	2	2	2
Total sectors available for user application, beyond MultiBoot FPGA configuration bitstreams		0	2	0	4	6
Total Flash memory bits available for user application, beyond MultiBoot FPGA configuration bitstreams, sector aligned, Default Addressing Mode		0	1,081,344 (1,056K) (1.03M)	0	2,162,688 (2,112K) (2.06M)	6,488,064 (6,336K) (6.18M)

Accessing In-System Flash Memory After Configuration

SPI_ACCESS Design Primitive

After the FPGA configures, the application loaded into the FPGA can access the ISF memory using a special design primitive called SPI_ACCESS, shown in Figure 1-1. All data accesses to and from the ISF memory are performed using an SPI serial protocol. Neither the Spartan-3AN FPGA itself nor the SPI_ACCESS primitive includes a dedicated SPI master controller. Instead, the control logic is implemented using the FPGA's programmable logic resources. The SPI_ACCESS primitive essentially connects the FPGA application to the In-System Flash memory array.

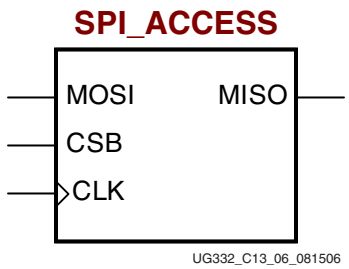

Figure 1-1: **SPI_ACCESS Primitive (only available on Spartan-3AN FPGAs)**

Table 1-2 describes the connections to the SPI_ACCESS primitive. The serial data lines are names relating to the logic that drives the data. The FPGA application is always the Master of each SPI transaction; the ISF memory is always the Slave.

Table 1-2: **SPI_ACCESS Primitive Connections**

Port Name	Direction	Function
MISO	Output	Master Input, Slave Output. Serial data output from the ISF memory array back to the FPGA logic.
MOSI	Input	Master Output, Slave Input. Serial data input to the ISF memory array from the FPGA logic.
CSB	Input	Active-Low chip-enable to ISF memory array, driven by FPGA logic.
CLK	Input	Clock input to ISF memory array, driven by FPGA logic.

Table 1-3 describes the available attributes for the SPI_ACCESS primitive.

Table 1-3: **SPI_ACCESS Primitive Attributes**

Attribute	Type	Allowed Values	Default	Description
SIM_DEVICE	String	"3S50AN", "3S200AN", "3S400AN", "3S700AN" or "3S1400AN"	"UNSPECIFIED"	Specifies the target device so that the proper size SPI Memory is used. This attributes required to be set.
SIM_USER_ID	64-byte Hex Value	Any 64-byte hex value	All locations default to 0xFF	Specifies the programmed USER ID in the Security Register for the SPI Memory

Table 1-3: **SPI_ACCESS Primitive Attributes *(Continued)***

Attribute	Type	Allowed Values	Default	Description
SIM_MEM_FILE	String	Specified file and directory name	"NONE"	Optionally specifies a hex file containing the initialization memory content for the SPI Memory.
SIM_FACTORY_ID	64-byte Hex Value	Any 64-byte Hex Value	All locations default to 0x00	Specifies the unique identifier value in the Security Register for simulation purposes (the actual hardware value will be specific to the particular device used). See "Security Register" in Chapter 9.
SIM_DELAY_TYPE	String	"ACCURATE", "SCALED"	"SCALED"	Scales down some timing delays for faster simulation run. "ACCURATE" = timing and delays consistent with datasheet specs. "SCALED" = timing numbers scaled back to run faster simulation, behavior not affected.

HDL Instantiation Examples

The SPI_ACCESS design primitive must be instantiated in an HDL design; it cannot be inferred by the logic synthesis software.

Caution! Only a subset of the commands available in hardware are supported in simulation for the SPI_ACCESS primitive. Please see the Simulation Support section for a list of these commands.

VHDL

The SPI_ACCESS primitive requires that the Xilinx Unisim Library be declared. Instantiate the SPI_ACCESS component and connect it to the other signals in the design.

Xilinx Unisim Library

The Xilinx Unisim library includes definitions for all the Spartan-3AN FPGA design primitives, including the SPI_ACCESS primitive. Declare the Unisim library before the **`entity`** declaration.

```
library UNISIM;
use UNISIM.VComponents.all;
entity XXXX is
```

Instantiate SPI_ACCESS Primitive

Instantiate the SPI_ACCESS design primitive after the **`architecture`** declaration. Connect each of the four SPI_ACCESS ports to a signal name in the FPGA application.

```
architecture Behavioral of XXXX is
begin
...
SPI_ACCESS_inst: SPI_ACCESS
generic map (
   SIM_DEVICE => "3S700AN"
)
port map (
   MISO => MISO_signal,  -- 1-bit SPI output data
   MOSI => MOSI_signal,  -- 1-bit SPI input data
```

```
   CSB   => CSB_signal,    -- 1-bit SPI chip enable
   CLK   => CLK_signal     -- 1-bit SPI clock input
);
-- End of SPI_ACCESS_inst instantiation
```

Verilog

Using Verilog, simply connect the SPI_ACCESS design primitive to signal names within the FPGA application.

```
SPI_ACCESS #(
.SIM_DEVICE("3S700AN")
) SPI_ACCESS_inst (
   .MISO(MISO_signal),  // 1-bit SPI output data
   .MOSI(MOSI_signal),  // 1-bit SPI input data
   .CSB(CSB_signal),    // 1-bit SPI chip enable
   .CLK(CLK_signal)     // 1-bit SPI clock input
);
// End of SPI_ACCESS_inst instantiation
```

SPI Transactions

The SPI_ACCESS primitive uses a typical SPI serial transaction protocol. By default, the interface supports SPI "Mode 3" transfers.

Caution! Only a subset of the commands available in hardware are supported in simulation for the SPI_ACCESS primitive. Please see the Simulation Support section for a list of these commands.

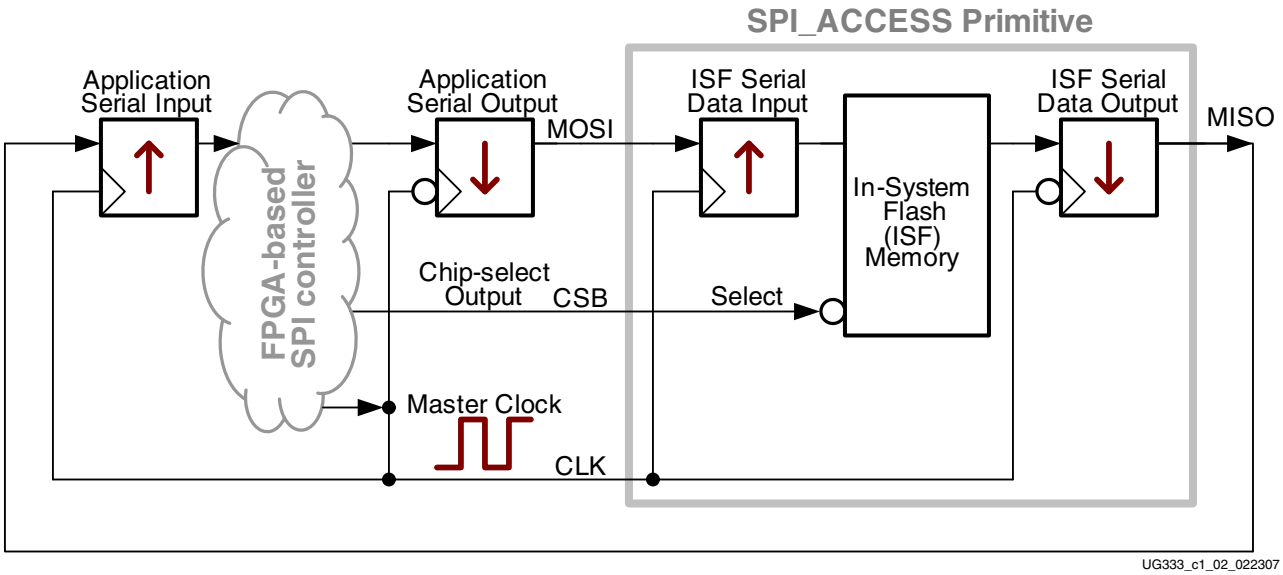

Figure 1-2: **FPGA-to-SPI_ACCESS Interface and Active Clock Edges**

- All transactions are controlled by a SPI Master controller, built using FPGA logic.
- All transactions are synchronized by the CLK input on the SPI_ACCESS primitive. The FPGA application generates the CLK signal.
- The SPI protocol uses both the rising and falling edges of the CLK signal.

- The FPGA application selects the ISF memory by driving the SPI_ACCESS CSB input Low when the CLK input is High and de-selects the ISF memory by driving the CSB input High.
- When CSB returns High, the current command is terminated.
- The SPI Master, which is the FPGA application, starts every transaction by supplying a command code or command sequence to the SPI_ACCESS MOSI input.
- The CSB select input must be Low throughout the duration of a transaction. It cannot change during the middle of an operation. Some ISF memory operations, such as erasing a page or sector, continue automatically even after the SPI transaction is complete.
- The FPGA application supplies data to the ISF memory via the SPI_ACCESS MOSI input, clocked on the falling edge of CLK.
- The ISF memory captures any data supplied on the SPI_ACCESS MOSI input using the rising edge of CLK.
- The ISF memory presents data or status on the SPI_ACCESS MISO output using the falling edge of CLK.
- The FPGA application captures any data supplied by the ISF memory on the SPI_ACCESS MISO output using the rising edge of CLK.
  - Because MISO output data changes every falling edge of CLK, the FPGA can also capture data on the falling edge of CLK to simplify the FPGA application.
- All data, commands, and address information are supplied serially, ordered from most-significant bit to least-significant bit.
- When the CSB select input is High to deselect the ISF memory, the SPI_ACCESS MISO output is High.

Example Detailed Command Sequence

Figure 1-3 provides a detailed example of a command sequence that reads the Status Register. This example shows the relation of the various control signals and the clock edges when data appears and is captured.

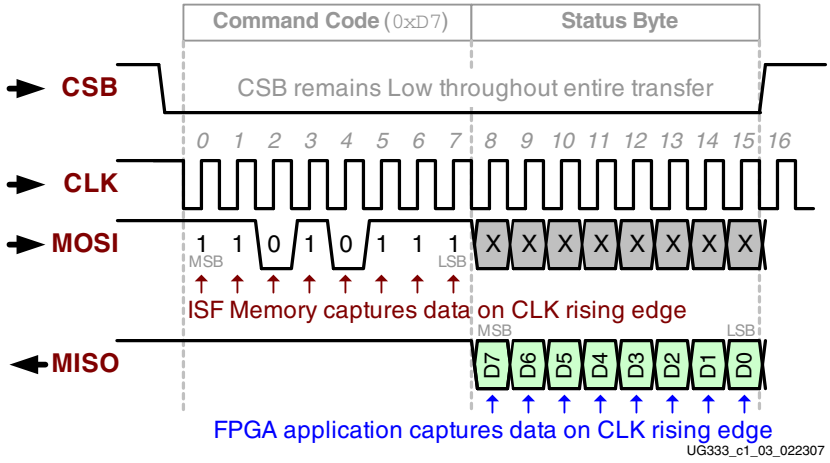

Figure 1-3: **Status Register Read Command Sequence**

1. The FPGA application starts the command by driving the SPI_ACCESS CSB input Low, while the CLK input is High. Subsequently, the CSB input must remain Low throughout the entire transfer.
2. The Status Register Read command code is `0xD7`, as detailed in Table 6-7, page 66. As mentioned in "SPI Transactions," all data is transferred most-significant bit first. Consequently, the FPGA application clocks in the binary pattern "`11010111`" on the MOSI input, synchronized to the falling edge of CLK.
3. The ISF memory captures the command sequence on the rising edge of CLK, as indicated.
4. Before data appears, the MISO output is High.
5. After all eight command bits are transferred, the ISF memory provides the current Status Register contents on the SPI_ACCESS MISO output. Again, the data appears most-significant bit first, synchronized to the falling edge of CLK.
6. The FPGA application captures the ISF status information on the rising edge of CLK.
7. After receiving the last bit of status information, the FPGA application drives the CSB input High to end the transaction.

Simulation Support

The SPI_ACCESS simulation model supports only a subset of the total commands that can be run in hardware. The commands that are supported in the model are shown below in Table 1-4. These have been tested and verified to work in the model and on silicon. All other commands are not supported in the simulation model, though they will work as expected in hardware. For more information on the SPI_ACCESS primitive and simulation model, please refer to the ISE® software, version 10.1 Synthesis and Simulation Design Guide at:

http://toolbox.xilinx.com/docsan/xilinx10/books/docs/sim/sim.pdf

or the Spartan-3A Libraries Guide at:

http://toolbox.xilinx.com/docsan/xilinx10/books/docs/spartan3a_hdl/spartan3a_hdl.pdf

Table 1-4: **SPI_ACCESS Simulation Supported Commands**

Command	Common Application	Hex Command Code
Fast Read	Reading a large block of contiguous data, if CLK frequency is above 33 MHz	0x0B
Random Read	Reading bytes from randomly-addressed locations, all read operations at 33 MHz or less	0x03
Status Register Read	Check ready/busy for programming commands, result of compare, protection, addressing mode, etc.	0xD7
Information Read	Read JEDEC Manufacturer and Device ID	0x9F
Security Register Read	Performs a read on the contents of the security register.	0x77
Security Register Program	Programs the User-Defined Field in the Security Register	0x9B
Buffer Write	Write data to SRAM page buffer; when complete, transfer to ISF memory using Buffer to Page Program command	Buffer1- 0x84 Buffer2- 0x87

Table 1-4: **SPI_ACCESS Simulation Supported Commands *(Continued)***

Command	Common Application	Hex Command Code
Buffer to Page Program with Built-in Erase	First erases selected memory page and programs page with data from designated buffer	Buffer1- 0x83 Buffer2- 0x86
Buffer to Page Program without Built-in Erase	Program a previously erased page with data from designated buffer	Buffer1- 0x88 Buffer2- 0x89
Page Program Through Buffer with Erase	Combines **Buffer Write** with **Buffer to Page Program with Built-in Erase** command	Buffer1- 0x82 Buffer2- 0x85
Page to Buffer Compare	Verify that the ISF memory array was programmed correctly	Buffer1- 0x60 Buffer2- 0x61
Page to Buffer Transfer	Transfers the entire contents of a selected ISF memory page to the specified SRAM page buffer	Buffer1- 0x53 Buffer2- 0x55
Sector Erase	Erases any unprotected, unlocked sector in the main memory	0x7C
Page Erase	Erases any individual page in the ISF memory array	0x81

Related Materials and References

UG332: *Spartan-3 Generation Configuration User Guide*

- Includes specific information on configuration from the internal SPI Flash in Chapter 10, and general information on configuration including MultiBoot.

UG331: *Spartan-3 Generation FPGA User Guide*

- Includes general FPGA information such as architecture, software, and packaging.

DS557: *Spartan-3AN FPGA Data Sheet*

- Includes ISF and FPGA timing specifications

Also see the Spartan-3AN FPGA Starter Kit for implementation examples, including using the ISF after configuration.

Spartan-3AN FPGA Starter Kit page

Spartan-3A/3AN FPGA Starter Kit Documentation

UG334: *Spartan-3A/3AN FPGA Starter Kit User Guide*

Spartan-3A/3AN FPGA Starter Kit Reference Designs

Chapter 2

In-System Flash Memory Architecture

Block Diagram

The Spartan®-3AN FPGA In-System Flash (ISF) memory consists of a main nonvolatile, reprogrammable Flash memory array and one or two SRAM page buffers, as shown in Figure 2-1. The Flash memory array is organized into Pages, Blocks, and Sectors. The smallest erasable unit is a Page. Page size depends on both the device and the addressing mode. The one or two SRAM page buffers simplify system interfaces and data programming. All Spartan-3AN FPGAs, except the XC3S50AN FPGA, have two SRAM page buffers. The XC3S50AN FPGA has a single such buffer.

All accesses to the ISF from the FPGA application are through a four-wire SPI interface, as described in "SPI_ACCESS Design Primitive," page 9.

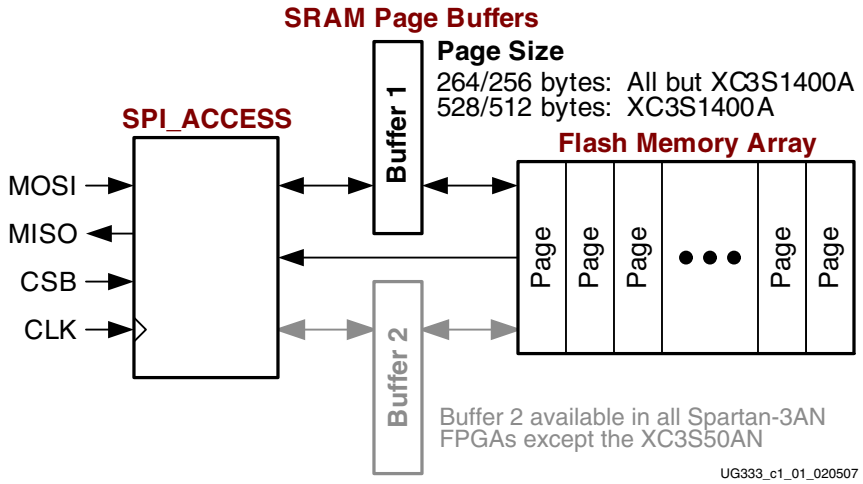

Figure 2-1: **Internal SPI Flash Block Diagram**

Flash Memory Array

For optimal flexibility, the In-System Flash (ISF) memory is divided into three levels of granularity, as shown in Figure 2-2. Figure 2-2 shows a specific example for the XC3S700AN. Other Spartan-3AN FPGA family members are similar but either have a different number of sectors or a different page size. The number of sectors and page size for other Spartan-3AN FPGAs is listed in Table 2-1, page 16.

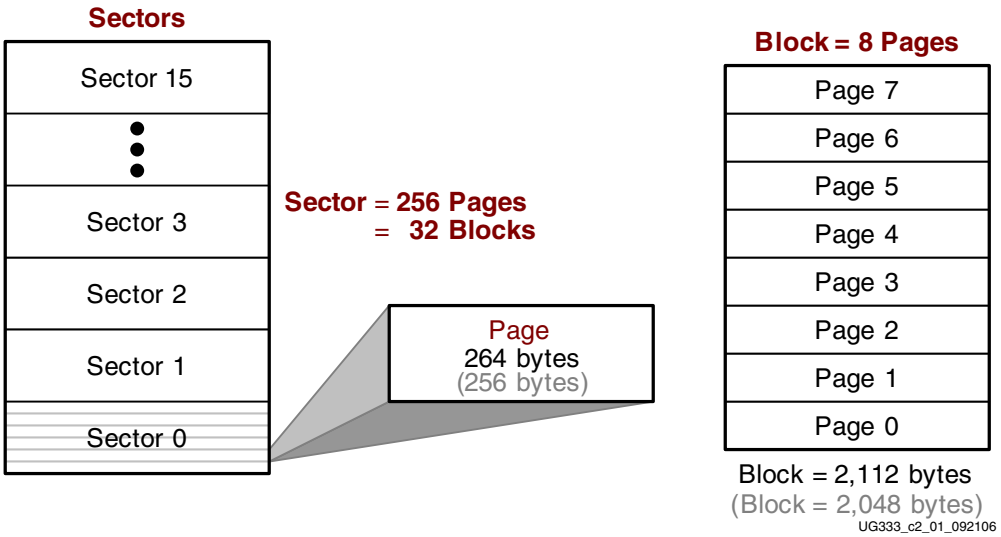

Figure 2-2: **XC3S700AN Internal SPI Flash Memory Hierarchy**

Table 2-1: **Spartan-3AN FPGA Memory Architecture**

Device	Total Flash Bits	Sectors	Sector Size	Pages per Sector	Total Pages	Page Size (Bytes)
XC3S50AN	1,081,344	4	33K	128	512	264
XC3S200AN	4,325,376	8	66K	256	2,048	264
XC3S400AN	4,325,376	8	66K		2,048	264
XC3S700AN	8,650,752	16	66K		4,096	264
XC3S1400AN	17,301,504	16	132K		4,096	528

The most basic construct within the hierarchy is a memory *Page*. By default, a Page consists of 264 bytes, except on the XC3S1400AN FPGA, which has a larger page size of 528 bytes. The page size is reduced when the Optional Power-of-2 Addressing Mode is selected, as shown in gray in Figure 2-2. The SRAM page buffer(s), shown in Figure 2-1, are large enough to hold an entire page image. A page is the smallest erasable element within the ISF memory, while a byte is the smallest readable and writable element within an SRAM page buffer.

Pages are also grouped into a larger structure called a *Block*, which consists of 8 pages. The Block Erase command provides an intermediate solution between the fast-but-small Page Erase command and the slower-but-larger Sector Erase command.Finally, pages and blocks are further combined into *Sectors*. A sector typically consists of 256 contiguous pages or 32 contiguous blocks, except on the XC3S50AN, which has 128 pages per sector. Sectors have additional control options. Sectors can be selectively write-protected by the FPGA application, a featured called Sector Protection. Similarly, sectors can be permanently locked down, preventing the contents from ever being erased or modified, using a feature called Sector Lockdown. The sector controls similarly affect any data bytes, pages, and blocks within the sector.

The ISF memory hierarchy directly impacts the ISF commands and addressing, as described in subsequent sections.

Sector 0 is further subdivided into two, individually protected sub-sectors, as shown in Figure 2-3. While the combined Sector 0 is the same size as all other sectors, Sector 0a is always 8 pages while Sector 0b represents the remaining pages in Sector 0. Consequently, the commands that operate on Sector 0 require slightly different addressing and control than the same commands used on other sectors.

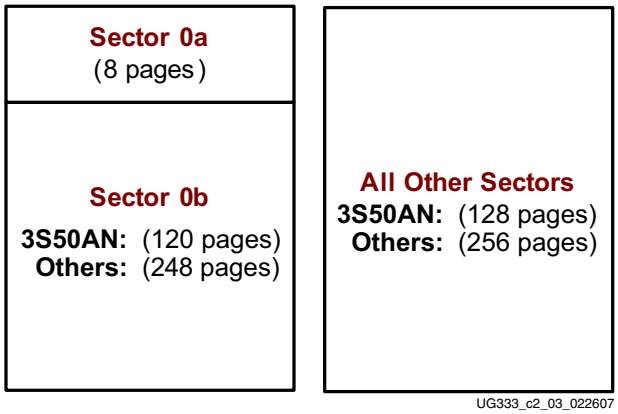

Figure 2-3: **Sector 0 is Sub-divided into Two Smaller Sub-sectors**

Spartan-3AN FPGA applications generally never make use of the split Sector 0 structure, but applications must be aware that it exists in order to erase or protect Sector 0.

Addressing Overview

All commands that require an address use a 24-bit address field to select a sector, a block, a page, or a byte location within a page. However, the In-System Flash memory supports two different possible addressing schemes.

Addressing Modes

- All Spartan-3AN FPGAs, as delivered, use the Default Addressing Mode, detailed in Table 2-2. All Xilinx® software primarily supports this mode.
- With an additional, special programming step, described in Appendix A, "Optional Power-of-2 Addressing Mode", Spartan-3AN FPGAs support a slightly different addressing mode, which may be more natural for some applications.

The Default Addressing Mode provides roughly 3% more total memory bits. Figure 2-4, page 18 shows the Flash memory array for the XC3S700AN FPGA, plus a detailed, expanded diagram of a page within the Flash array. The diagram also describes how the 24 bits in a command address field select a sector, block, page, or byte within a memory page. Figure 2-5 is a similar diagram specific to the XC3S1400AN, which uses a larger page size.

Using the Default Addressing Mode, each memory page is slightly larger than the typical power-of-2 page size found in other memories. For example, all Spartan-3AN FPGAs except the XC3S1400AN use a 264-byte page, while the XC3S1400AN memory pages are double the size at 528 bytes. The "extra" bits are useful for a variety of applications.

- More total nonvolatile memory for FPGA configuration or data storage applications.
- Page pointers, linked address pointers, attributes, and status indicators for a Flash-based file system.
- Error detection/correction bits for extreme applications.

- Counters, to track the number of program/erase cycles per page.

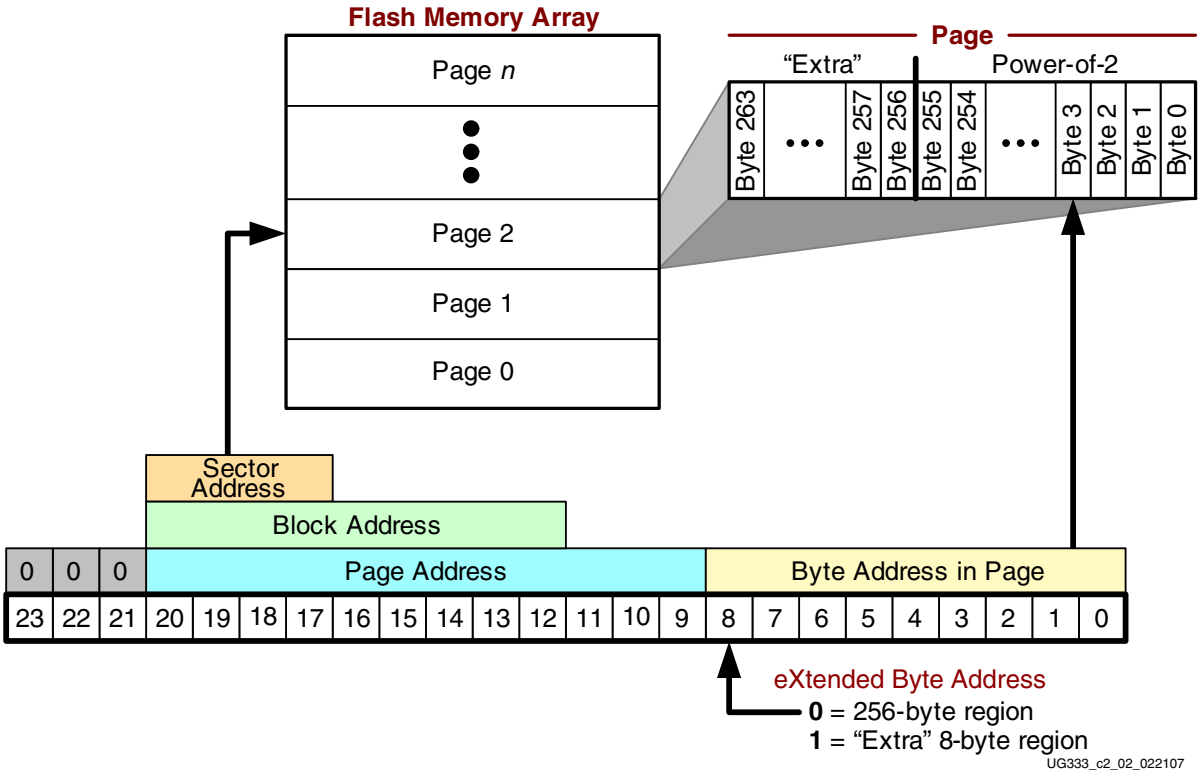

Figure 2-4: **Default Addressing Mode for XC3S700AN FPGA**

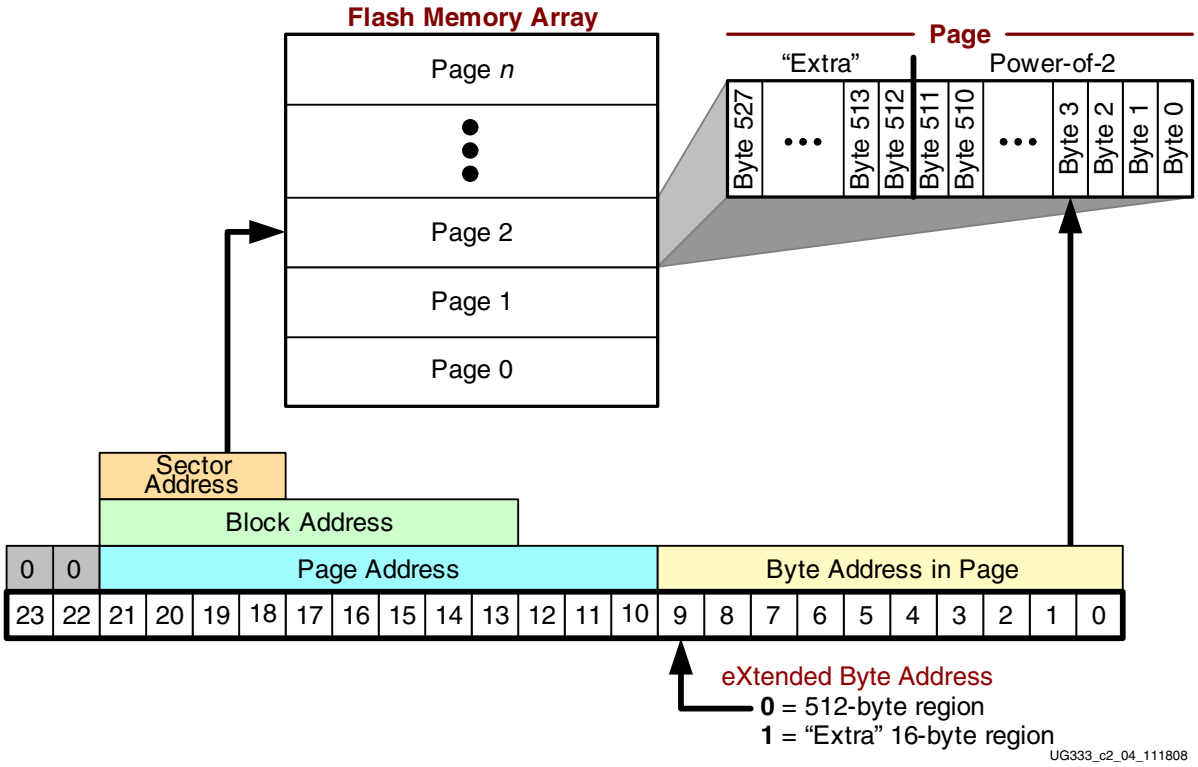

Figure 2-5: **Default Addressing Mode, XC3S1400AN FPGA**

Default Addressing Mode

In the default addressing mode, specific memory bytes are addressed by page and by the specific byte location within that page. As shown in Table 2-2, the number of pages and the page size varies by FPGA part number.

Table 2-2: **Default Addressing Mode**

FPGA	High Address								Middle Address								Low Address							
	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
3S50AN	0	0	0	0	0	0	**Sector**		Don't Care bits															
							SA1	SA0	Sector 0a Sector 0b				X	X	X	X	X	X	X	X	X	X	X	X
							Block Address						Don't Care bits											
							BL5	BL4	BL3	BL2	BL1	BL0	X	X	X	X	X	X	X	X	X	X	X	X
							Page Address (512 pages)										**Byte Address** (264 bytes)							
							PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	BA8	BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0
3S200AN 3S400AN	0	0	0	0	**Sector**				Don't Care bits															
					SA2	SA1	SA0		Sector 0a Sector 0b				X	X	X	X	X	X	X	X	X	X	X	X
					Block Address								Don't Care bits											
					BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0	X	X	X	X	X	X	X	X	X	X	X	X
					Page Address (2,048 pages)												**Byte Address** (264 bytes)							
					PA10	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	BA8	BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0
Bit Location	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
3S700AN	0	0	0	**Sector**					Don't Care bits															
				SA3	SA2	SA1	SA0		Sector 0a Sector 0b				X	X	X	X	X	X	X	X	X	X	X	X
				Block Address									Don't Care bits											
				BL8	BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0	X	X	X	X	X	X	X	X	X	X	X	X
				Page Address (4,096 pages)													**Byte Address** (264 bytes)							
				PA11	PA10	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	BA8	BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0
3S1400AN	0	0	**Sector**						Don't Care bits															
			SA3	SA2	SA1	SA0			Sector 0a Sector 0b				X	X	X	X	X	X	X	X	X	X	X	X
			Block Address										Don't Care bits											
			BL8	BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0		X	X	X	X	X	X	X	X	X	X	X	X
			Page Address (4,096 pages)												**Byte Address** (528 bytes)									
			PA11	PA10	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	BA9	BA8	BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0
Bit Location	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Delivered State

Each Spartan-3AN FPGA is delivered with an erased ISF memory. Engineering Samples may have a pre-programmed pattern, and should be erased and reprogrammed with the FPGA bitstream used in the end application.

- All Flash memory locations are erased and byte locations contain `0xFF`.
- Spartan-3AN FPGAs use the "Default Addressing Mode," page 19.
- The "Sector Protection Register," page 74 is programmed with `0x00`, indicating that all memory sectors are unprotected.
- The Unique Identifier field in the "Security Register," page 83 is permanently factory programmed with a value that is unique to every Spartan-3AN FPGA.
- The User-Defined Field in the "Security Register," page 83 is erased, and all locations are `0xFF`.

Memory Allocation Tables

The following tables describe how the ISF memory is allocated to FPGA configuration bitstreams and the portions that are available for page-aligned and sector-aligned data. The tables assume that the ISF memory contains two FPGA configuration bitstreams, the primary bitstream loaded at power-up and a secondary MultiBoot bitstream, starting at the next sector boundary. If the application does not require a MultiBoot bitstream, then the allocated ISF memory can be used for data storage. The physical address is shown for both the Default Addressing Mode and for the Optional Power-of-2 Addressing Mode. Consequently, the starting page number is different between the addressing modes for any user data that shares a sector with the end of an FPGA configuration bitstream.

- ISF memory allocation for the XC3S50AN FPGA is provided in Table 2-3, page 21.
- ISF memory allocation for the XC3S200AN FPGA is provided in Table 2-4, page 22.
- ISF memory allocation for the XC3S400AN FPGA is provided in Table 2-5, page 23.
- ISF memory allocation for the XC3S700AN FPGA is provided in Table 2-6, page 24.
- ISF memory allocation for the XC3S1400AN FPGA is provided in Table 2-7, page 25.

Table 2-3: **XC3S50AN In-System Flash Memory Allocation**

Allocation	Sector		Default Addressing Mode		Optional Power-of-2 Addressing Mode	
			Page	Address	Page	Address
Bitstream	0	a	0	0x00_0000	0	0x00_0000
		b	...	0x00_1000	...	0x00_0800
	1		207	0x01_9E00	213	0x00_D500
First available user data space (page aligned)	1		208	0x01_A000	214	0x00_D600
			...	. . .	...	. . .
			255	0x01_FE00	255	0x00_FF00
2nd MultiBoot Bitstream, or available for user data space	2		256	0x02_0000	256	0x01_0000
	...		...	. . .	...	. . .
	3		463	0x03_9E00	469	0x01_D500
Second available user data space (page aligned)	3		464	0x03_A000	470	0x01_D600
			...	. . .	...	. . .
			511	0x03_FE00	511	0x01_FF00

Table 2-4: **XC3S200AN In-System Flash Memory Allocation**

Allocation	Sector		Default Addressing Mode		Optional Power-of-2 Addressing Mode	
			Page	Address	Page	Address
Bitstream	0	a	0	0x00_0000	0	0x00_0000
		b	...	0x00_1000	...	0x00_0800
	...		...	. . .	...	. . .
	2		566	0x04_6C00	584	0x02_4800
First available user data space (page aligned)	2		567	0x04_6E00	585	0x02_4900
			...	. . .	...	. . .
			767	0x05_FE00	767	0x02_FF00
2nd MultiBoot Bitstream, or available for user data space	3		768	0x06_0000	768	0x03_0000
	...		...	. . .	...	. . .
	5		1,334	0x0A_6C00	1,352	0x05_4800
Second available user data space (page aligned)	5		1,335	0x0A_6E00	1,353	0x05_4900
			...	. . .	...	. . .
			1,535	0x0B_FE00	1,535	0x05_FF00
User data space (sector aligned)	6		1,536	0x0C_0000	1,536	0x06_0000
			...	. . .	...	. . .
			1,791	0x0D_FE00	1,791	0x06_FF00
	7		1,792	0x0E_0000	1,792	0x07_0000
			...	. . .	...	. . .
			2,047	0x0F_FE00	2,047	0x07_FF00

Table 2-5: **XC3S400AN In-System Flash Memory Allocation**

Allocation	Sector		Default Addressing Mode		Optional Power-of-2 Addressing Mode	
			Page	Address	Page	Address
Bitstream	0	a	0	0x00_0000	0	0x00_0000
		b	...	0x00_1000	...	0x00_0800
	...		...	. . .	...	. . .
	3		893	0x06_FA00	921	0x03_9900
First available user data space (page aligned)	3		894	0x06_FC00	922	0x03_9A00
			...	. . .	...	. . .
			1,023	0x07_FE00	1,023	0x03_FF00
2nd MultiBoot Bitstream, or available for user data space	4		1,024	0x08_0000	1,024	0x04_0000
	...		...	. . .	...	. . .
	7		1,917	0x0E_FA00	1,945	0x07_9900
Second available user data space (page aligned)	7		1,918	0x0E_FC00	1,946	0x07_9A00
			...	. . .	...	. . .
			2,047	0x0F_FE00	2,047	0x07_FF00

Table 2-6: **XC3S700AN In-System Flash Memory Allocation**

Allocation	Sector		Default Addressing Mode		Optional Power-of-2 Addressing Mode	
			Page	Address	Page	Address
Bitstream	0	a	0	0x00_0000	0	0x00_0000
		b	...	0x00_1000	...	0x00_0800
	...		...	. . .	...	. . .
	5		1,293	0x0A_1A00	1,334	0x05_3600
First available user data space (page aligned)	5		1,294	0x0A_1C00	1,335	0x05_3700
			...	. . .	...	. . .
			1,535	0x0B_FE00	1,535	0x05_FF00
2nd MultiBoot Bitstream, or available for user data space	6		1,536	0x0C_0000	1,536	0x06_0000
	...		...	. . .	...	. . .
	11		2,829	0x16_1A00	2,890	0x0B_4A00
Second available user data space (page aligned)	11		2,830	0x16_1C00	2,871	0x0B_4B00
			...	. . .	...	. . .
			3,071	0x17_FE00	3,071	0x0B_FF00
User data space (sector aligned)	12		3,072	0x18_0000	3,072	0x0C_0000
			...	. . .	...	. . .
			3,327	0x19_FE00	3,327	0x0C_FF00
	13		3,328	0x1A_0000	3,328	0x0D_0000
			...	. . .	...	. . .
			3,583	0x1B_FE00	3,583	0x0D_FF00
	14		3,584	0x1C_0000	3,584	0x0E_0000
			...	. . .	...	. . .
			3,839	0x1D_FE00	3,839	0x0E_FF00
	15		3,840	0x1E_0000	3,840	0x0F_0000
			...	. . .	...	. . .
			4,095	0x1F_FE00	4,095	0x0F_FF00

Table 2-7: **XC3S1400AN In-System Flash Memory Allocation**

Allocation	Sector		Default Addressing Mode		Optional Power-of-2 Addressing Mode	
			Page	Address	Page	Address
Bitstream	0	a	0	0x00_0000	0	0x00_0000
		b	...	0x00_2000	...	0x00_1000
	...		...	...	...	...
	4		1,125	0x11_9400	1,160	0x09_1000
First available user data space (page aligned)	4		1,126	0x11_9800	1,161	0x09_1200
			...	...	...	...
			1,279	0x13_FC00	1,279	0x09_FE00
2nd MultiBoot Bitstream, or available for user data space	5		1,280	0x14_0000	1,280	0x0A_0000
	...		...	...	...	...
	9		2,405	0x25_9400	2,440	0x13_1000
Second available user data space (page aligned)	9		2,406	0x25_9800	2,441	0x13_1200
			...	...	...	...
			2,559	0x27_FC00	2,559	0x13_FE00
User data space (sector aligned)	10		2,560	0x28_0000	2,560	0x14_0000
			...	...	...	...
			2,815	0x2B_FC00	2,815	0x15_FE00
	11		2,816	0x2C_0000	2,816	0x16_0000
			...	...	...	...
			3,071	0x2F_FC00	3,071	0x17_FE00
	12		3,072	0x30_0000	3,072	0x18_0000
			...	...	...	...
			3,327	0x33_FC00	3,327	0x19_FE00
	13		3,328	0x34_0000	3,328	0x1A_0000
			...	...	...	...
			3,583	0x37_FC00	3,583	0x1B_FE00
	14		3,584	0x38_0000	3,584	0x1C_0000
			...	...	...	...
			3,839	0x3B_FC00	3,839	0x1D_FE00
	15		3,840	0x3C_0000	3,840	0x1E_0000
			...	...	...	...
			4,095	0x3F_FC00	4,095	0x1F_FE00

MultiBoot Configuration Bitstream Guidelines

The following guidelines are recommended when storing multiple configuration files in the In-System Flash (ISF) memory.

Align to Flash Sector Boundaries

Spartan-3AN FPGA MultiBoot addressing is flexible enough to allow a bitstream to begin at any byte boundary. However, ideally, a Spartan-3AN FPGA MultiBoot configuration image should be aligned to a sector boundary. This way, one FPGA bitstream can be updated without affecting others in the Flash. Aligning to an ISF sector boundary provides the additional advantage of allowing independent protection or lockdown of the bitstreams.

Additional Memory Space Required for DCM_WAIT

Multiple configuration images should be spaced more than 5 ms apart so that the second configuration file does not interfere with a delayed start-up following programming with the first configuration file. Start-up can be delayed by waiting for lock from one or more Digital Clock Managers (DCMs), a slow or missing STARTUP user clock, or holding the DONE pin Low.

Each DCM provides an option setting that, during configuration, causes the FPGA to wait for the DCM to acquire and lock to its input clock frequency before the DCM allows the FPGA to finish the configuration process. The lock time, which is specified in the Spartan-3AN FPGA data sheet, depends on the DCM mode, and the input clock frequency.

Even if the FPGA is waiting for one or more DCMs to lock before completing configuration, the FPGA's configuration controller continues searching for the next synchronization word. If two adjacent MultiBoot images are placed one immediately following the other, and the first FPGA bitstream contains a DCM with the DCM_WAIT option set, then potential configuration problems can occur. If the controller sees the synchronization word in the second FPGA bitstream before completing the current configuration, it starts interpreting data from the second bitstream. However, the FPGA's configuration logic may complete the current configuration even though the FPGA has read data from the second bitstream.

> ***Caution!*** FPGA applications that use the DCM_WAIT option on a DCM must ensure sufficient spacing between MultiBoot configuration images!

Spacing MultiBoot bitstreams sufficiently apart in memory prevents the FPGA from ever seeing the second synchronization word. For more details, see UG332: Spartan-3 Generation Configuration User Guide, Chapter 14, *Reconfiguration and MultiBoot*.

User Data Storage Guidelines

The following guidelines are recommended when storing user data in the In-System Flash (ISF) memory.

1. Do not intermix user data with the last page of FPGA configuration bitstream data. Intermixing user data and configuration data makes updating the FPGA bitstream more difficult.
2. If using the Sector Protection or Sector Lockdown features, do not intermix user data in the sectors that contain the FPGA bitstream(s).

See the "Memory Allocation Tables," page 20 for specific locations for each Spartan-3AN FPGA and for both addressing modes.

Chapter 3

Read Commands

The Spartan®-3AN FPGA application reads In-System Flash (ISF) memory data either directly from the main Flash memory or from either one of the SRAM data buffers by issuing the appropriate command code. Table 3-1 summarizes and compares the various supported read commands. Some read commands offer multiple forms. One form is best for reading large blocks of contiguous data while the other form is better for reading single bytes from randomly-address locations. Typically, the commands with faster data transfer also have a longer initial latency and require one or more "don't care" bytes after sending the appropriate read command and 24-bit address.

Table 3-1: **Summary of In-System Flash Memory Read Commands**

Read Command	Best Application	Hex Command Code	Max CLK Frequency	Extra Initial Latency	Read From	Minimum Read Size	Maximum Read Size	Affects SRAM Page Buffers	Simulation Support
Fast Read	Reading a large block of contiguous data, if CLK frequency is above 33 MHz	0x0B	50	8 cycles	Flash array	1 byte	Entire array	No	Yes
Random Read	Reading bytes from randomly-addressed locations, all read operations at 33 MHz or less	0x03	33	None	Flash array	1 byte	Entire array	No	Yes
Page to Buffer Transfer	Transfers the entire contents of a selected ISF memory page to the specified SRAM page buffer	Buffer 1 (0x53) Buffer 2 (0x55)	33	None	SRAM page buffer	One page	One page	Yes	Yes
Buffer Read	Reading multiple contiguous bytes from the SRAM page buffer	Buffer 1 (0xD4) Buffer 2 (0xD6)	50	8 cycles	SRAM page buffer	1 byte	One page	No	No
Buffer Read (Low Freq)	Randomly reading bytes from the SRAM page buffer	Buffer 1 (0xD1) Buffer 2 (0xD3)	33	None	SRAM page buffer	1 byte	One page	No	No

Notes:

1. The Buffer 2 commands are not available in the XC3S50AN because it has only one SRAM page buffer.

Fast Read

The Fast Read command is best for longer, sequential read operations. This is the same command that the FPGA issues during configuration. This command is also best for code shadowing applications, where the FPGA application copies a large amount of code or data into external SRAM or DDR SDRAM for a MicroBlaze™ processor. Although it has longer initial latency than the Random Read command, the Fast Read command supports a CLK clock frequency up to 50 MHz.

The Fast Read command sequentially reads a continuous stream of data directly from Flash memory bypassing the SRAM page buffers, as shown in Figure 3-1. The command specifies an initial starting byte address in the ISF memory. The ISF memory incorporates an internal address pointer that automatically increments on every clock cycle, allowing one continuous read operation without requiring additional address sequences.

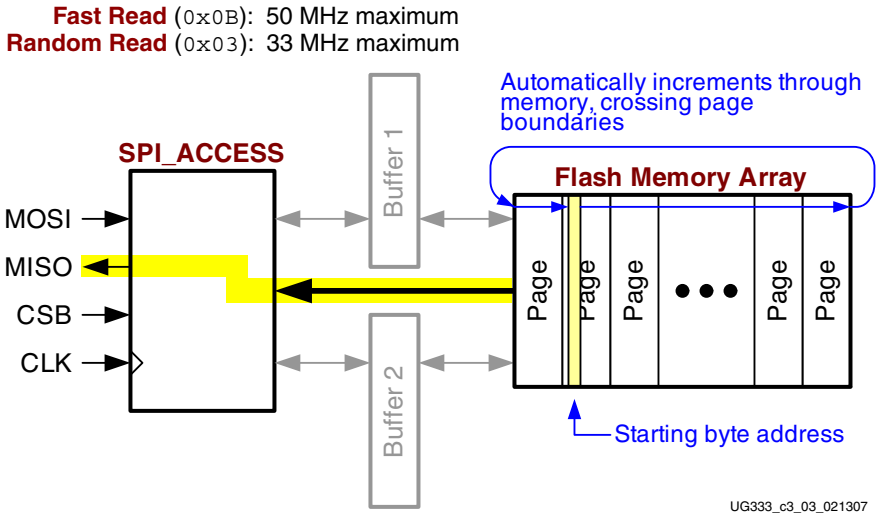

Figure 3-1: **Fast Read and Random Read Commands**

To perform a Fast Read command, summarized in Table 2-2, page 19 and shown in detail in Figure 3-2, the FPGA application must perform the following actions.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the Fast Read command code, 0x0B, most-significant bit first.

Table 3-2: **Fast Read (0x0B) Command Summary**

Pin	Command	24-bit Starting Page and Byte Address			Don't Care Byte	ISF Memory Data Bytes (most-significant bit first)		
		High Address	Middle Address	Low Address				
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	...	Byte n+6
MOSI	0x0B	**Default Addressing:** See Table 2-2, page 19 **Optional Power-of-2 Addressing:** See Table A-3, page 89			XX	XX	...	XX
MISO	High					Data Byte +0	...	Data Byte +*n*

Notes:

1. The Fast Read command Is supported in simulation.

- Similarly, serially clock in a 24-bit page and byte starting address.
  - The starting byte location can be anywhere in the ISF memory array, located on any page, as shown in Figure 3-1.
  - If using default addressing, see Table 2-2, page 19.
  - If using power-of-2 addressing, see Table A-3, page 89.

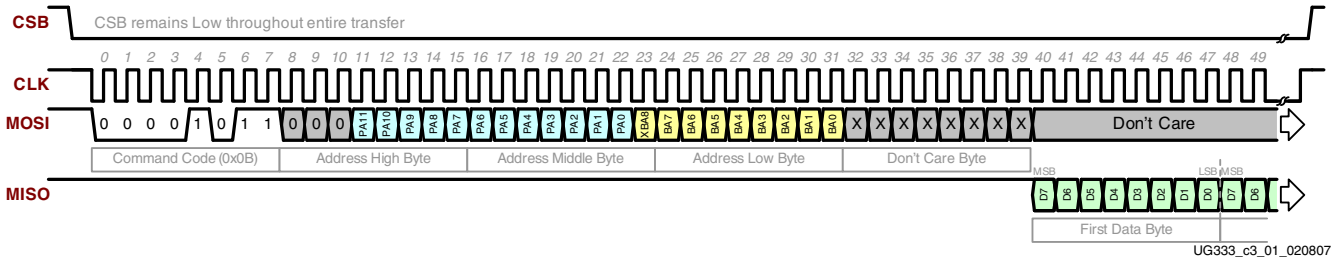

Figure 3-2: **Fast Read Command Waveform (XC3S700AN, Default Address Mode)**

- Provide eight additional CLK cycles. At this point, the data supplied on the MOSI input does not matter.
- On the next falling CLK edge, the requested data serially appears on the MISO output port.
  - Data is clocked out serially, most-significant bit first.
  - While CSB is Low, new data appears on the MISO output on every subsequent falling CLK edge. The ISF memory automatically increments the implied address counter through contiguous memory locations, as highlighted in Figure 3-1, regardless of the address mode.
- To end the data transfer, deassert CSB High on the falling edge of CLK.

The CSB signal must remain Low throughout the entire data transfer—when writing the command code, the 24 address bits, the 8 "don't care" bits, and when reading the dummy bytes and data bytes.

Upon reaching the end of a memory page, the ISF memory continues reading at the beginning of the next page, as shown in Figure 3-1. There is no added delay when crossing a page boundary. After reading the last bit in the memory array, the ISF continues reading but returns to the beginning of the first page of memory. Again, there is no added delay when wrapping around from the end of the array to the beginning of the array. A Low-to-High transition on CSB terminates the read operation and the MISO output pin returns High.

The Fast Read command bypasses both SRAM page buffers; the contents of the buffers remain unchanged.

Random Read

The Random Read command is best for smaller, random read operations. It has less initial latency than the Fast Read command, but only operates up to 33 MHz, less than the maximum frequency possible with the Fast Read command.

The Random Read command sequentially reads a continuous stream of data directly from Flash memory bypassing the SRAM page buffers, as shown in Figure 3-1. The command specifies an initial starting byte address in the ISF memory. The ISF memory incorporates an internal address pointer that automatically increments on every clock cycle, allowing one continuous read operation without requiring additional address sequences.

To perform a Random Read command, summarized in Table 3-3 and shown in detail in Figure 3-3, the FPGA application must perform the following actions.

Table 3-3: **Random Read (0x03) Command Summary**

Pin	Command	24-bit Starting Page and Byte Address			ISF Memory Data Bytes (most-significant bit first)		
		High Address	Middle Address	Low Address			
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5		Byte n+5
MOSI	0x03	**Default Addressing:** See Table 2-2, page 19 **Optional Power-of-2 Addressing:** See Table A-3, page 89			XX	...	XX
MISO	High				Data Byte +0	...	Data Byte +*n*

Notes:

1. The Random Read command is supported in simulation.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the Random Read command code, 0x03, most-significant bit first.
- Similarly, serially clock in a 24-bit starting byte address.
  - The starting byte location can be anywhere in the ISF memory array, located on any page, as shown in Figure 3-1.
  - If using the default address scheme, see Table 2-2, page 19.
  - If using power-of-2 addressing, see Table A-3, page 89.

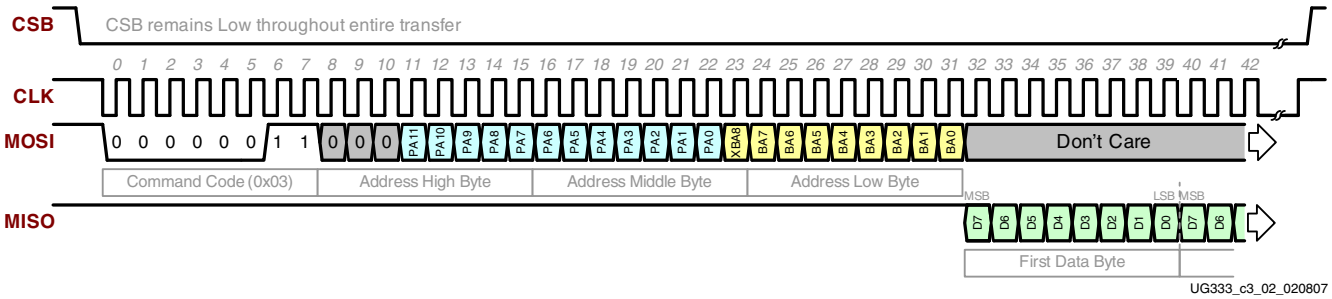

Figure 3-3: **Random Read Command Waveform (XC3S700AN, Default Address Mode)**

- No dummy byte is required for the Random Read command. At this point, the data supplied on the MOSI input does not matter.
- On the next falling CLK edge, the requested data serially appears on the MISO output port.
  - Data is clocked out serially, most-significant bit first.
  - While CSB is Low, new data appears on the MISO output on every subsequent falling CLK edge. The ISF memory automatically increments the implied address counter through contiguous memory locations, as highlighted in Figure 3-1, regardless of the address mode.
- To end the data transfer, deassert CSB High on the falling edge of CLK.

The CSB signal must remain Low throughout the entire data transfer—when writing the command code, the 24 address bits, and when reading the data bytes.

Upon reaching the end of a memory page, the ISF memory continues reading at the beginning of the next page, as shown in Figure 3-1. There is no added delay when crossing a page boundary. After reading the last bit in the memory array, the ISF continues reading but returns to the beginning of the first page of memory. Again, there is no added delay when wrapping around from the end of the array to the beginning of the array. A Low-to-High transition on CSB terminates the read operation and the MISO output pin returns High.

The Fast Read command bypasses both SRAM page buffers; the contents of the buffers remain unchanged.

Page to Buffer Transfer

The Page to Buffer Transfer command copies the entire contents of an ISF memory page into the specified SRAM page buffer, as shown in Figure 3-4. The contents of the ISF memory page are unaffected.

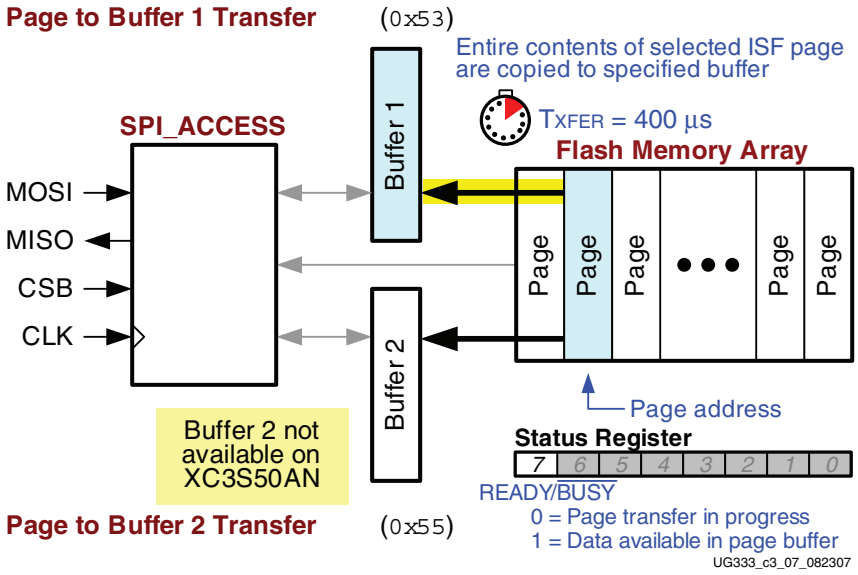

Figure 3-4: **Page to Buffer Transfer Command**

To issue a Page to Buffer Transfer command, the FPGA application must perform the following actions.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the appropriate Page to Buffer Transfer command code, shown in Table 3-4, most-significant bit first.

Table 3-4: **Page to Buffer Transfer Command Summary**

Pin	Command	24-bit Page Address		
		High Address	Middle Address	Low Address
		ISF Page Address		Byte Address Unused
	Byte 1	Byte 2	Byte 3	Byte 4
MOSI	**Page to Buffer 1 Transfer** 0x53 **Page to Buffer 2 Transfer**(1) 0x55	**Default Addressing:** See Table 5-3, page 53 **Power-of-2 Addressing:** See Table A-3, page 89		Don't Care XX

Notes:

1. The Buffer 2 command is not available in the XC3S50AN because it has only one SRAM page buffer.
2. The Page to Buffer Transfer command is supported in simulation.

- Similarly, serially clock in a 24-bit page address.
  - Only the Page Address is required. Any byte address values are ignored.
  - If using the default address scheme, see Table 2-2, page 19.
  - If using power-of-2 addressing, see Table A-3, page 89.
- To end the data transfer, deassert CSB High on the falling edge of CLK. Subsequently, the entire contents of the addressed ISF memory page is transferred to the selected SRAM page buffer.

The page transfer operation is internally self-timed and does not require CLK to toggle. The page is transferred in 400 µs or less, specified as symbol T_{XFER} in the Spartan-3AN FPGA data sheet and shown in Table 3-5. During the transfer, the READY/BUSY bit (bit 7) in the Status Register is '0' and returns to '1' when the transfer completes. The FPGA application can monitor this bit to determine when the transfer is complete or the application can wait 400 µs or more before accessing the data in the specified SRAM page buffer.

Table 3-5: **Page to Buffer Transfer, T_{XFER}**

Symbol	Description	FPGA	Typ	Max	Units
T_{XFER}	Page to Buffer Transfer Time	All	—	400	µs

While the page transfer is in progress, the FPGA can access any other portion of the ISF memory that is not actively involved in the transfer, including any of the following commands.

- Read from or write to the other SRAM page buffer, not involved in the present operation.
  - Buffer Read
  - Buffer Write
- Status Register Read
- Information Read

Buffer Read

The FPGA application can independently access the SRAM data buffers separately from the ISF memory array, as shown in Figure 3-5. The Buffer Read command sequentially reads data directly from the selected buffer. When reading data from the buffer, first load data into the buffer from an ISF memory page using the Page to Buffer Transfer command.

Caution! The Buffer 2 Read command is not supported on the XC3S50AN FPGA because it contains only one buffer.

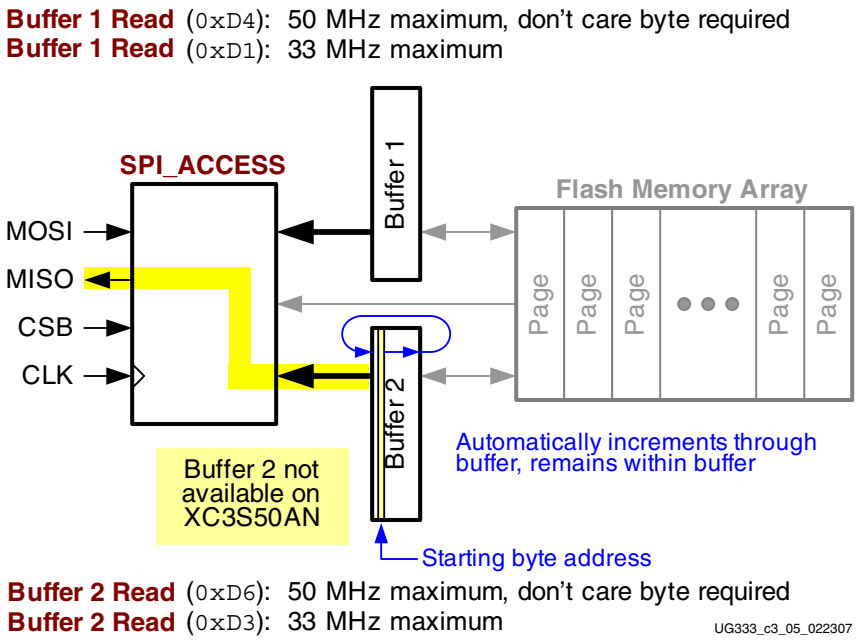

Figure 3-5: **Buffer Read Command**

There are two versions of the Buffer Read command. The version that operates up to 50 MHz, shown in Table 3-6, is best for reading multiple contiguous bytes from the SRAM page buffer. This command requires eight "don't care" bits after specifying the 24-bit address.

Table 3-6: **Buffer Read Command Summary (High Frequency, up to 50 MHz)**

		24-bit Starting Byte Address			Don't Care Byte	Page Buffer Data Bytes (most-significant bit first)		
	Command	High Address	Middle Address	Low Address				
		Unused	Byte Address in Buffer					
Pin	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	...	Byte n+6
MOSI	**Buffer 1 Read** 0xD4 **Buffer 2 Read**(1) 0xD6	0x00	**Default Addressing:** See Table 2-2, page 19 **Power-of-2 Addressing:** See Table A-3, page 89		XX	XX	.. .	XX
MISO	High					Data Byte +0	.. .	Data Byte +*n*

Notes:

1. The Buffer 2 Read command is not available in the XC3S50AN because it has only one SRAM page buffer.
2. The Buffer Read command (High Frequency) is not supported in simulation.

The slower version that operates up to 33 MHz, shown in Table 3-7, is best for reading single, randomly-accessed bytes within the SRAM page buffer. This version has lower initial latency for single-byte transfers.

Table 3-7: **Buffer Read Command Summary (Low Frequency, up to 33 MHz)**

Pin	Command	24-bit Starting Byte Address			Page Buffer Data Bytes (most-significant bit first)		
		High Address	Middle Address	Low Address			
		Unused	Byte Address in Buffer				
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	...	Byte 6
MOSI	**Buffer 1 Read** 0xD1 **Buffer 2 Read**(1) 0xD3	0x00	**Default Addressing:** See Table 2-2, page 19 **Power-of-2 Addressing:** See Table A-3, page 89		XX	...	XX
MISO	High				Data Byte +0	...	Data Byte +*n*

Notes:

1. The Buffer 2 Read command is not available in the XC3S50AN because it has only one SRAM page buffer.
2. The Buffer Read command (Low Frequency) is not supported in simulation.

It is possible for the FPGA application to read from one SRAM page buffer while the other buffer is actively transferring data to the main memory from a previous programming operation, as shown in Figure 3-6.

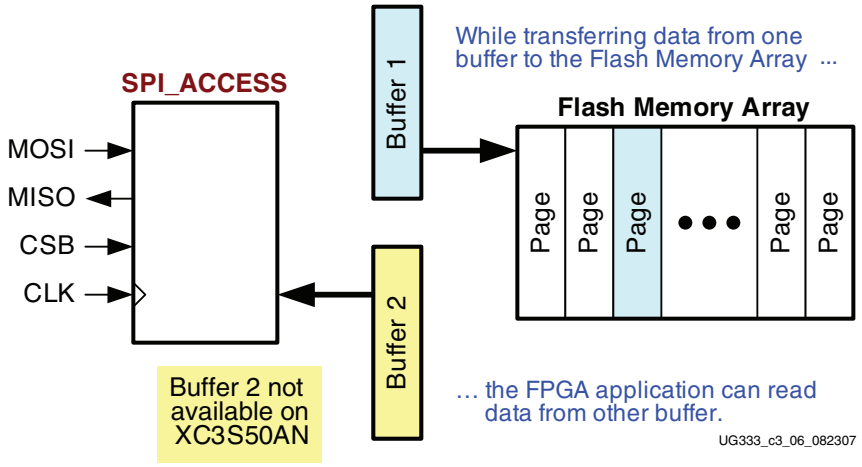

Figure 3-6: **SRAM Page Buffers Support Read-while-Write Operations**

To issue a Buffer Read command, the FPGA application must perform the following actions.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the appropriate Buffer Read command code, shown in either Table 3-6 or Table 3-7, most-significant bit first.
- Similarly, serially clock in a 24-bit starting byte address. The page address bits are ignored.
  - The starting byte location can be anywhere in the ISF memory array, located on any page, as shown in Figure 3-5.

  - If using the default address scheme, see Table 2-2, page 19.
  - If using power-of-2 addressing, see Table A-3, page 89.
- The slower version of the command (Table 3-7) does not require "don't care" bits while the faster version (Table 3-6) of the command does require eight "don't care" bits. At this point, the data supplied on the MOSI input does not matter.
- On the next falling CLK edge, the requested data serially appears on the MISO output port.
  - Data is clocked out serially, most-significant bit first.
  - While CSB is Low, new data appears on the MISO output on every subsequent falling CLK edge. The ISF memory automatically increments the implied address counter through contiguous memory locations, as highlighted in Figure 3-5, regardless of the address mode.
- To end the data transfer, drive CSB High on the falling edge of CLK.

The CSB signal must remain Low throughout the entire data transfer. If the transaction reaches the end of a buffer, the ISF memory continues reading back at the beginning of the buffer.

Chapter 4

Write and Program Commands

The Spartan®-3AN FPGA application programs data into the In-System Flash (ISF) memory either directly to the main Flash memory array or through one of the SRAM page buffers by issuing the appropriate command code. Table 4-1 summarizes and compares the various supported write and program commands. One form is best for writing large blocks of contiguous data while the other form is better for writing single bytes to randomly-addressed locations. Typically, the commands with faster data transfer also have a longer initial latency and require one or more "don't care" bytes after sending the appropriate write command and 24-bit address.

Table 4-1: **Summary of Write and Program Commands**

Command	Best Application	Hex Command Code	Maximum CLK Frequency	Erase Page	Write To	Minimum Write Size	Maximum Write Size	Affects SRAM Page Buffers	Simulation Support
Buffer Write	Write data to SRAM page buffer; when complete, transfer to ISF memory using Buffer to Page Program command	Buffer 1 (0x84) Buffer 2(1) (0x87)	50	No	SRAM page buffer	1 byte	One page	Yes	Yes
Buffer to Page Program with Built-in Erase	First erases selected memory page and programs page with data from designated buffer	Buffer 1 (0x83) Buffer 2(1) (0x86)	50	Yes	ISF page	One page	One page	No	Yes
Buffer to Page Program without Built-in Erase	Program a previously erased page with data from designated buffer	Buffer 1 (0x88) Buffer 2(1) (0x89)	50	No	ISF page	One page	One page	No	Yes
Page Program Through Buffer	Combines **Buffer Write** with **Buffer to Page Program with Built-in Erase** command	Buffer 1 (0x82) Buffer 2(1) (0x85)	50	Yes	Selected buffer first, ISF page later	1 byte	One page	Yes	Yes
Page to Buffer Compare (Program Verify)	Verify that the ISF memory array was programmed correctly	Buffer 1 (0x60) Buffer 2(1) (0x61)	50	No	N/A	N/A	N/A	No	Yes

Table 4-1: **Summary of Write and Program Commands *(Continued)***

Command	Best Application	Hex Command Code	Maximum CLK Frequency	Erase Page	Write To	Minimum Write Size	Maximum Write Size	Affects SRAM Page Buffers	Simulation Support
Auto Page Rewrite	Refresh page contents after 10,000 random program operations to a sector	Buffer 1 (0x58) Buffer 2[1] (0x59)	50	Yes	Buffer, then page	N/A	N/A	Yes	No

Notes:

1. The Buffer 2 commands are not available in the XC3S50AN because it has only one SRAM page buffer.

Buffer Write

The FPGA application can write data directly to one of the SRAM page buffers. Along with the Buffer Read command, the FPGA application can randomly read or write the buffer data without affecting the ISF memory array. Data is not written into nonvolatile Flash memory using this command. Once the data within the buffer is finalized, the FPGA application can subsequently copy contents of the buffer into the ISF memory array using one of the following commands.

- Buffer to Page Program with Built-in Erase
- Buffer to Page Program without Built-in Erase

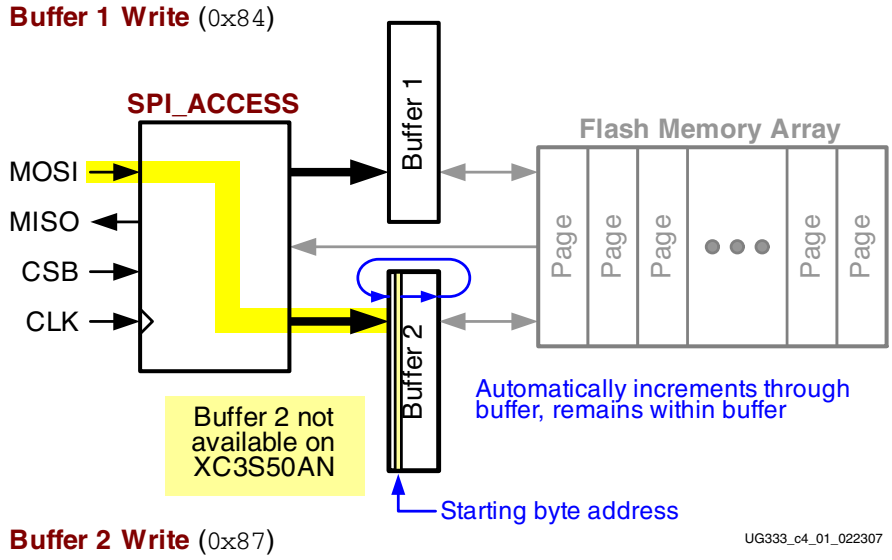

Figure 4-1: **Buffer Write Command**

To issue a Buffer Write command, the FPGA application must perform the following actions.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the appropriate Buffer Write command code, shown in Table 4-2, most-significant bit first.

Table 4-2: **Buffer Write Command Summary**

Pin	Command	24-bit Starting Byte Address			Page Buffer Data			
		High Address	Middle Address	Low Address				
		Unused	Byte Address in Buffer					
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6		Byte n+4
MOSI	**Buffer 1 Write** 0x84 **Buffer 2 Write**(1) 0x87	0x00	**Default Addressing:** See Table 2-2, page 19 **Power-of-2 Addressing:** See Table A-3, page 89		Data +0	Data +1	...	Data +*n*

Notes:

1. The Buffer 2 Write command is not available in the XC3S50AN because it has only one SRAM page buffer.
2. The Buffer Write command is supported in simulation.

- Similarly, serially clock in a 24-bit starting byte address. Any page address information is ignored.
  - The starting byte location can be anywhere within the selected SRAM page buffer, as shown in Figure 4-1.
  - If using the default address scheme, see Table 2-2, page 19.
  - If using power-of-2 addressing, see Table A-3, page 89.
- On the next falling CLK edge, serially supply the write data on the MOSI port.
  - Data is clocked in serially, most-significant bit first.
  - While CSB is Low, present new data on the MOSI pin on every subsequent falling CLK edge. The ISF memory automatically increments the implied address counter through the SRAM page buffer, as highlighted in Figure 4-1.
    - The first data byte written is stored in Byte Address + 0
    - The second data byte written is stored in Byte Address + 1, and so on.
  - If the transaction reaches the end of a buffer, the ISF memory continues writing back at the beginning of the buffer.
- To end the data transfer, drive CSB High on the falling edge of CLK.

The CSB signal must remain Low throughout the entire data transfer.

Buffer to Page Program with Built-in Erase
Buffer to Page Program without Built-in Erase

The ISF memory offers two similar commands that copy the contents of an SRAM page buffer to a selected memory page. One version erases the selected memory page before programming the page, while the second version programs a previously-erased page with data from an SRAM page buffer.

- The **Buffer to Page Program with Built-in Erase** command, shown in Figure 4-2, first erases the selected memory page and then programs the page with the data stored in the designated SRAM page buffer.

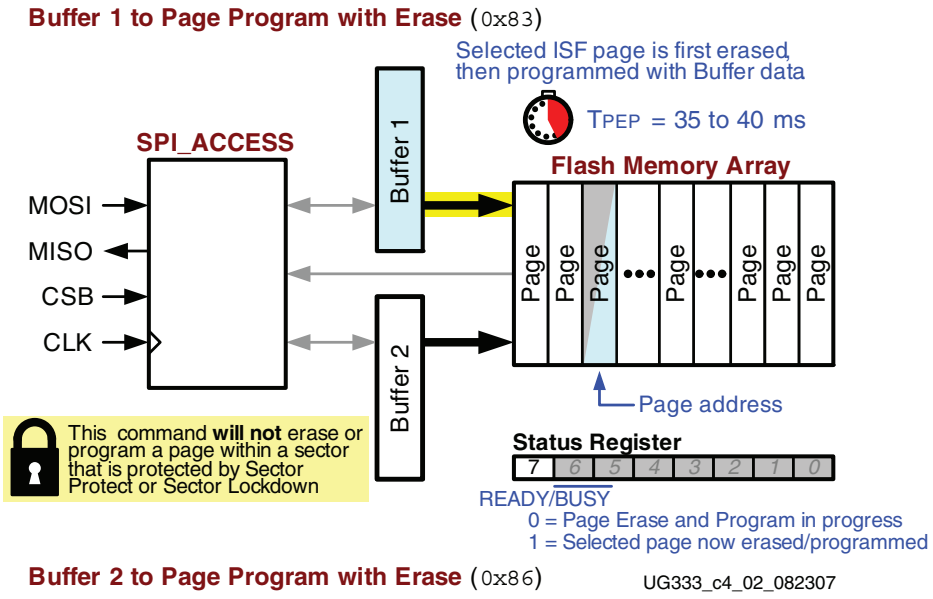

Figure 4-2: **Buffer to Page Program with Built-in Erase**

- The **Buffer to Page Program without Built-in Erase** command, shown in Figure 4-3, simply programs an erased page with the data stored in the designated SRAM page buffer.

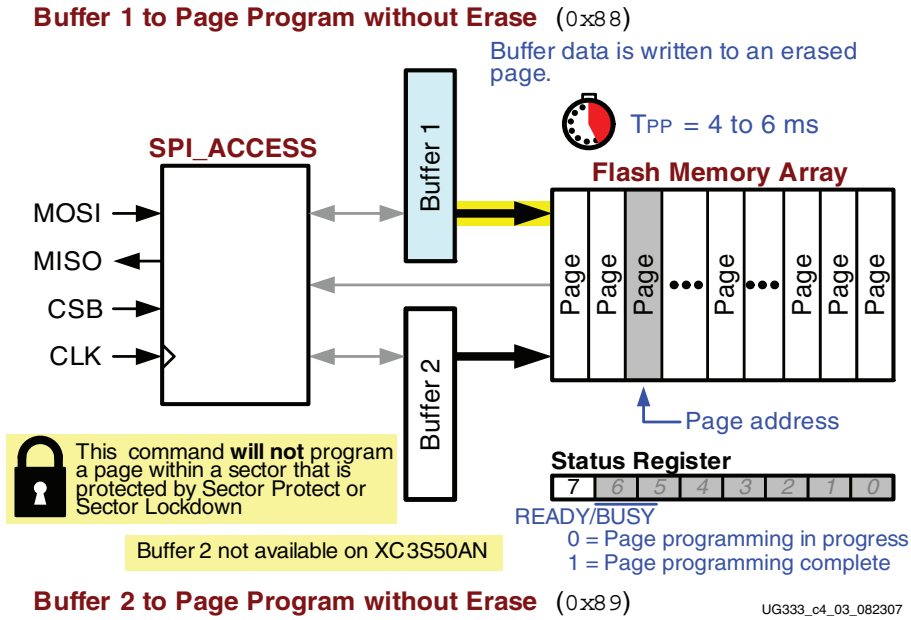

Figure 4-3: **Buffer to Page Program without Built-in Erase**

To issue a Buffer to Page Program command, with or without built-in erase, the FPGA application must perform the following actions.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the appropriate command code, shown in Table 4-3, most-significant bit first.

Table 4-3: **Buffer to Page Program Command Summary**

Pin	Command	24-bit Page Address		
		High Address	Middle Address	Low Address
		Page Address in Buffer		Byte Address Unused
	Byte 1	Byte 2	Byte 3	Byte 4
MOSI	**Buffer 1 to Page Program with Erase** 0x83 **Buffer 2 to Page Program with Erase**(1) 0x86	**Default Addressing:** See Table 5-3, page 53 **Power-of-2 Addressing:** See Table A-4, page 89		Don't Care XX
	Buffer 1 to Page Program without Erase 0x88 **Buffer 2 to Page Program without Erase**(1) 0x89			

Notes:

1. The Buffer 2 commands are not available in the XC3S50AN because it has only one SRAM page buffer.
2. The Buffer to Page Program command is supported in simulation.

- Similarly, serially clock in a 24-bit page address.
  - If using the default address scheme, see Table 2-2, page 19 or Table 5-3, page 53.
  - If using power-of-2 addressing, see Table A-4, page 89.
- To end the data transfer, drive CSB High on the falling edge of CLK.

The CSB signal must remain Low throughout the entire data transfer.

A Low-to-High transition on the CSB input completes the command.

- If using the command ***with*** page erase…
  - the ISF memory first erases the selected memory page
  - then programs the page with the data stored in the designated SRAM page buffer.
  - The operation is internally self-timed and completes in the Page Erase and Programming time, T_{PEP}, shown in Table 4-4 and specified in the Spartan-3AN FPGA data sheet.

Table 4-4: **Page Erase and Programming Time, T_{PEP}**

Symbol	Description	FPGA	Typ	Max	Units
T_{PEP}	Page Erase and Programming Time	XC3S50AN XC3S200AN XC3S400AN XC3S700AN	14	35	ms
		XC3S1400AN	17	40	ms

- If using the command ***without*** page erase…
  - The ISF memory programs the page with the data stored in the designated SRAM page buffer.
  - This version of the command does not erase the selected page.
  - The operation is internally self-timed and completes in the Page Programming time, T_{PP}, shown in Table 4-5 and specified in the Spartan-3AN FPGA data sheet.

Table 4-5: **Page Programming Time, T_{PP}**

Symbol	Description	FPGA	Typ	Max	Units
T_{PP}	Page Programming Time	XC3S50AN XC3S200AN XC3S400AN	2	4	ms
		XC3S700AN XC3S1400AN	3	6	ms

During the command execution time, the READY/BUSY bit (bit 7) of the Status Register indicates whether the Page Programming or Page Erase and Programming operation is in progress or whether it has completed.

Page Program Through Buffer

This operation combines the Buffer Write (Figure 4-1) and Buffer to Page Program with Built-in Erase (Figure 4-2) operations. The FPGA application must specify a full 24-bit address, including both the Page Address and the starting Byte Address within the SRAM page buffer.

To issue a Page Program Through Buffer command, the FPGA application must perform the following actions.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the appropriate Page Program Through Buffer command code, shown in Table 4-2, most-significant bit first.

Table 4-6: **Page Program Through Buffer Command Summary**

		24-bit Page and Starting Byte Address			Page Buffer Data			
	Command	High Address	Middle Address	Low Address				
Pin	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	...	Byte n+4
MOSI	**Page Program Through Buffer 1** 0x82 **Page Program Through Buffer 2**(1) 0x85	**Default Addressing:** See Table 2-2, page 19 **Power-of-2 Addressing:** See Table A-3, page 89			Data +0	Data +1	...	Data +*n*

Notes:

1. The Buffer 2 command is not available in the XC3S50AN because it has only one SRAM page buffer.
2. The Page Program Through Buffer command is supported in simulation.

- Similarly, serially clock in a 24-bit starting page and byte address.
  - The starting byte location can be anywhere within the selected SRAM page buffer, as shown in Figure 4-1.
  - The page address must also be specified.
  - If using the default address scheme, see Table 2-2, page 19.
  - If using power-of-2 addressing, see Table A-3, page 89.
- On the next falling CLK edge, serially supply the write data on the MOSI port.
  - Data is clocked in serially, most-significant bit first.
  - While CSB is Low, present new data on the MOSI pin on every subsequent falling CLK edge. The ISF memory automatically increments the implied address counter through the SRAM page buffer, as highlighted in Figure 4-1.
    - The first data byte written is stored in Byte Address + 0
    - The second data byte written is stored in Byte Address + 1, and so on.
  - If the transaction reaches the end of a buffer, the ISF memory continues writing back at the beginning of the buffer.
  - While it is possible to write less than a full page of data, be sure that the SRAM page buffer contains valid data. The data from any unwritten buffer locations will write the previous contents to the ISF memory page.
- To end the data transfer, drive CSB High on the falling edge of CLK.

The CSB signal must remain Low throughout the entire data transfer. A Low-to-High transition on the CSB input completes the command.

The ISF memory first erases the selected memory page, then programs the page with the data stored in the designated SRAM page buffer. The operation is internally self-timed and completes in the Page Erase and Programming time, T_{PEP}, shown in Table 4-4 and specified in the Spartan-3AN FPGA data sheet.

During the command execution time, the READY/BUSY bit (bit 7) of the Status Register indicates whether the Page Erase and Programming operation is in progress or whether it has completed.

Page to Buffer Compare (Program Verify)

The Page to Buffer Compare command is not actually a programming command, but is primarily used to verify correct programming of nonvolatile data in an ISF memory page. As shown in Figure 4-4, this command compares the contents of the addressed memory page against the contents of the designated SRAM page buffer. If one or more bits differ between the page and buffer, then the Compare bit (bit 6 in the Status Register) is '1'.

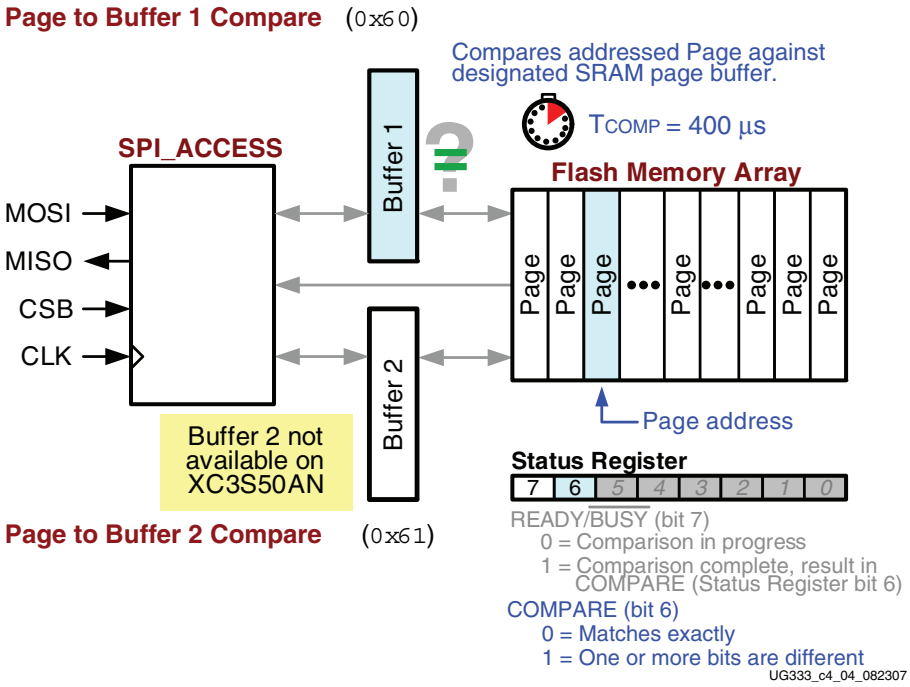

Figure 4-4: **Page to Buffer Compare Command**

To issue a Page to Buffer Compare command, the FPGA application must perform the following actions.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the appropriate command code, shown in Table 4-2, most-significant bit first.

Table 4-7: **Page to Buffer Compare Command Summary**

Pin	Command	24-bit Page Address		
		High Address	Middle Address	Low Address
	Byte 1	Byte 2	Byte 3	Byte 4
MOSI	**Page to Buffer 1 Compare** 0x60 **Page to Buffer 2 Compare**(1) 0x61	**Default Addressing:** See Table 5-3, page 53 **Power-of-2 Addressing:** See Table A-4, page 89		Don't Care XX

Notes:

1. The Buffer 2 command is not available in the XC3S50AN because it has only one SRAM page buffer.
2. The Page to Buffer Compare command is supported in simulation.

- Similarly, serially clock in a 24-bit Page Address.
  - If using the default address scheme, see Table 2-2, page 19.
  - If using power-of-2 addressing, see Table A-4, page 89.
- To end the command, drive CSB High on the falling edge of CLK.

The CSB signal must remain Low throughout the entire data transfer. A Low-to-High transition starts the comparison between the addressed ISF memory page and the designated SRAM page buffer. The operation is internally self-timed and completes in a Page to Buffer Compare time, shown in Table 4-8 and specified in the Spartan-3AN FPGA data sheet.

Table 4-8: **Page to Buffer Compare Time, T_{COMP}**

Symbol	Description	FPGA	Typ	Max	Units
T_{COMP}	Page to Buffer Compare Time	All	—	400	µs

During the command execution time, the READY/BUSY bit (bit 7) of the Status Register indicates whether the Page to Buffer Compare operation is in progress or whether it has completed.

When the operation completes, the Status Register bit 6, Compare, reflects the result of the comparison.

Pre-initializing SRAM Page Buffer Contents

The SRAM page buffers are not automatically pre-initialized to a known value before executing a command. Consequently, never assume the contents of a buffer location. Instead, the FPGA application must define each location.

The fastest way to pre-initialize a buffer is using the Buffer Write command. Ideally, define any blank locations to `0xFF`, which is the erased state of an ISF memory location.

Another easy method to pre-initialize a page buffer is to use an Page to Buffer Transfer command, pointed at a page that is known to be erased. This method copies the blank page, which contains all 0xFF, into the page buffer. Although this method takes more time to complete, about 400 µs, it has less overhead for the FPGA application.

EEPROM-Like, Byte-Level Write Operations

The Spartan-3AN FPGA In-System Flash (ISF) memory includes small page size, SRAM page buffers, combined with flexible memory read and write commands. Consequently, the ISF memory can perform small, byte-level write operations, much like EEPROM memories.

To update one or more bytes without affecting other Flash data, perform these steps.

- Copy the contents of an entire ISF memory page to an SRAM page buffer using the Page to Buffer Transfer command.
- Update one or more locations in the SRAM page buffer.
  - If updating one byte or contiguous bytes, use a Page Program Through Buffer command, which automatically erases and programs the addressed page at the completion of the command.
  - If updating multiple, non-contiguous bytes, update selected locations in the page buffer using multiple Buffer Write commands. When the buffer updates are complete, issue a Buffer to Page Program with Built-in Erase command.
- If randomly updating data, each page within a sector must be updated/rewritten at least once every 10,000 cumulative page erase or page programming operations in that sector. Instead of tracking the number of operations, optionally issue a Auto Page Rewrite command after updating each page.

Sequential vs. Random Page Programming, Cumulative Operations

In most applications, the ISF memory stores nonvolatile data such as FPGA bitstreams and MicroBlaze™ processor code. In these applications, entire memory sectors are first erased, then new programming data is written to each page sequentially. In these applications, where data is programmed sequentially, each page supports up to 100,000 program/erase cycles.

In other applications, the FPGA application may store or update nonvolatile data in randomly-addressed page locations. One such example is described in "EEPROM-Like, Byte-Level Write Operations". While each page still supports a maximum of 100,000 program/erase cycles, each page within a randomly-addressed sector must be refreshed every 10,000 cumulative page erase/program operations with that sector. Refreshing the page contents avoids any potential charge buildup due to the random page programming operations.

The ISF memory has a special command, Auto Page Rewrite, specifically for this purpose.

Auto Page Rewrite

This command is only needed if multiple bytes within a page or multiple pages of data are modified in a random fashion within a sector. This command combines two other operations, a Page to Buffer Transfer command followed by a Buffer to Page Program with Built-in Erase command, as shown in Figure 4-5.

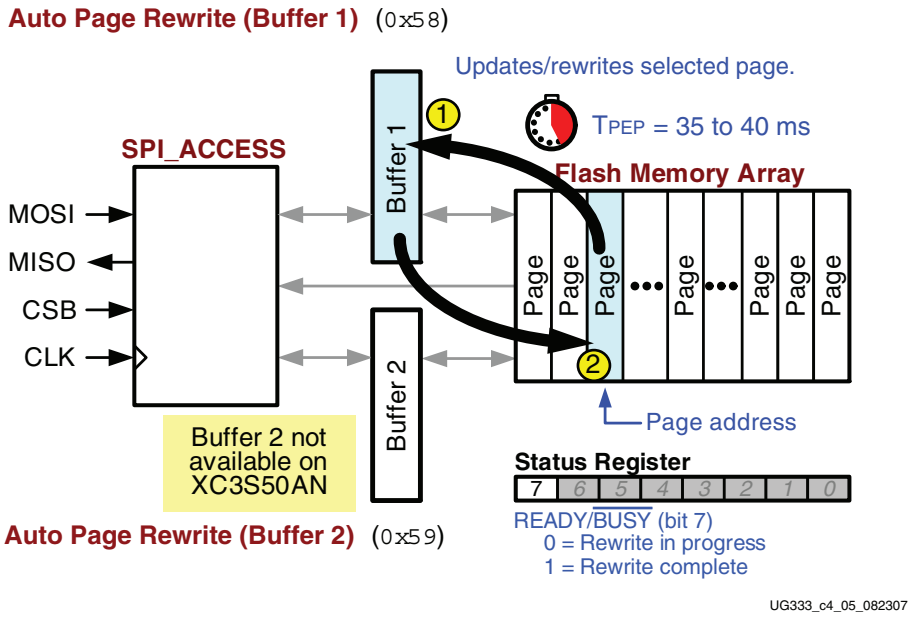

Figure 4-5: **Auto Page Rewrite Command**

1. A page of data is first copied from the addressed ISF memory page to either Buffer 1 or Buffer 2.
2. Then, the same data from Buffer 1 or Buffer 2 is programmed back to the same ISF memory page.

To issue an Auto Page Rewrite command, the FPGA application must perform the following actions.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the appropriate Auto Page Rewrite command code, shown in Table 4-2, most-significant bit first.

Table 4-9: **Auto Page Rewrite Through Buffer Command Summary**

Pin	Command	24-bit Page Address		
		High Address	Middle Address	Low Address
	Byte 1	Byte 2	Byte 3	Byte 4
MOSI	**Auto Page Rewrite (Buffer 1)** 0x58 **Auto Page Rewrite (Buffer 2)**(1) 0x59	**Default Addressing:** See Table 5-3, page 53 **Power-of-2 Addressing:** See Table A-4, page 89		Don't Care XX

Notes:

1. The Buffer 2 command is not available in the XC3S50AN because it has only one SRAM page buffer.
2. The Auto Page Rewrite Through Buffer command is not supported in simulation.

- Similarly, serially clock in a 24-bit Page Address.
  - If using the default address scheme, see Table 2-2, page 19.
  - If using power-of-2 addressing, see Table A-4, page 89.
- To end the data command, drive CSB High on the falling edge of CLK.

A Low-to-High transition on the CSB input completes the command.

The addressed ISF memory page is then copied into the designated buffer, the memory page is erased, then the buffer contents are copied back to the addressed page. The operation is internally self-timed and completes in the Page Erase and Programming time, T_{PEP}, shown in Table 4-4 and specified in the Spartan-3AN FPGA data sheet.

During the command execution time, the READY/BUSY bit (bit 7) of the Status Register indicates whether the Auto Page Rewrite operation is in progress or whether it has completed.

XILINX®

Chapter 5

Erase Commands

The Spartan®-3AN FPGA In-System Flash (ISF) memory supports multiple levels of erase operations for maximum application flexibility.

- The Page Erase command erases the smallest possible unit, a single ISF memory page.
- The Block Erase command erases a group of eight pages.
- The Sector Erase command erases all the pages within an ISF memory sector.

Sector Protect and Sector Lockdown Prevent Erase Operations

The Sector Protection and Sector Lockdown features prevent accidental or unauthorized erase operations. A page, block, or sector erase command will not occur if the operation targets a location within a sector that is actively protected via the Sector Protection or Sector Lockdown features.

Erased State

Erase commands erase the selected page, block, or sector and return the contents to a logical '1', or `0xFF` on a byte basis. Erase commands do not affect the contents of any page, block or sector in a protected or locked down sector.

Page Erase

The Page Erase command erases any individual page in the ISF memory array, as shown in Figure 5-2. Typically, the Page Erase command is used to prepare a page for a subsequent Buffer to Page Program without Built-in Erase command.

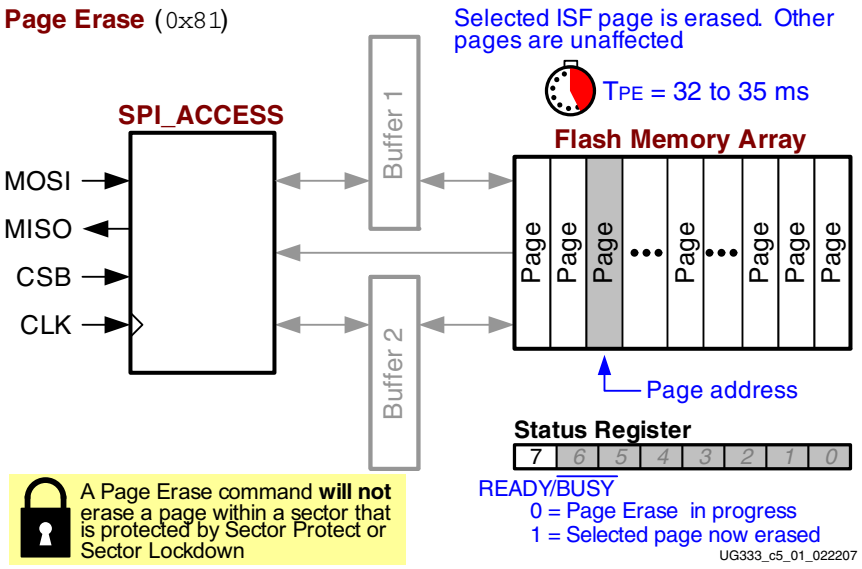

Figure 5-1: **Page Erase Command**

To perform a Page Erase command, the FPGA application must perform the following actions.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the appropriate Page Erase command code, 0x81, most-significant bit first, as shown in Table 5-1.

Table 5-1: **Page Erase (0x81) Command**

		24-bit Address		
		High Address	Middle Address	Low Address
	Command	Page Address		Byte Address Unused
Pin	Byte 1	Byte 2	Byte 3	Byte 4
MOSI	0x81	**Default Addressing:** See Table 5-3 **Optional Power-of-2 Addressing:** See Table A-4, page 89		Don't Care XX

Notes:

1. The Page Erase command is supported in simulation.

- Similarly, serially clock in a 24-bit page address.
  - Only the page address is required. Any byte address values are ignored.
  - The number of pages and the alignment of the page address within the 24-bit address field varies by Spartan-3AN FPGA type, as shown in Table 5-2.

- If using the default address scheme, see Table 5-3.
- If using power-of-2 addressing, see Table A-4, page 89.

- Drive CSB High on the falling edge of CLK to end the command.

Table 5-2: **Page Addressing Summary**

FPGA	Total Pages	Addressing Mode			
		Default		Power-of-2	
		Page Size (Bytes)	Alignment	Page Size (Bytes)	Alignment
XC3S50AN	512	264	Every 512 bytes	256	Every 256 bytes
XC3S200AN XC3S400AN	2,048				
XC3S700AN	4,096				
XC3S1400AN	4,096	528	Every 1K bytes	512	Every 512 bytes

Table 5-3: **Page Address, Default Addressing Mode**

FPGA	High Address								Middle Address								Low Address							
	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
3S50AN	0	0	0	0	0	0	**Page Address** (512 pages)										Don't Care bits							
							PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	X	X	X	X	X	X	X	X	X
3S200AN 3S400AN	0	0	0	0	**Page Address** (2,048 pages)												Don't Care bits							
					PA10	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	X	X	X	X	X	X	X	X	X
3S700AN	0	0	0	**Page Address** (4,096 pages)													Don't Care bits							
				PA11	PA10	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	X	X	X	X	X	X	X	X	X
3S1400AN	0	0	**Page Address** (4,096 pages)														Don't Care bits							
			PA11	PA10	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	X	X	X	X	X	X	X	X	X	X
Bit Location	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

A Low-to-High transition on the CSB input completes the command and the Flash then erases the selected page. The erase operation is internally self-timed and completes in the time shown in Table 5-4 and specified in the Spartan-3AN FPGA data sheet. During this time, the READY/BUSY bit (bit 7) of the Status Register indicates whether the Page Erase operation is in progress or whether it has completed.

Table 5-4: **Page Erase Time**

Symbol	Description	FPGA	Typ	Max	Units
T_{PE}	Page Erase Time	XC3S50AN XC3S200AN XC3S400AN	13	32	ms
		XC3S700AN XC3S1400AN	15	35	ms

While the Page Erase operation is in progress, the FPGA can access any other portion of the ISF memory, including any of the following commands.

- Read from or write to an SRAM page buffer, which are not used during an erase operation.
  - Buffer Read
  - Buffer Write
- Status Register Read
- Information Read

Block Erase

The Block Erase command erases eight pages at one time, as shown in Figure 5-2, which proves useful when writing large amounts of data into the ISF memory. The Block Erase command avoids using multiple Page Erase commands, resulting in a faster erase time for between three to eight ISF pages.

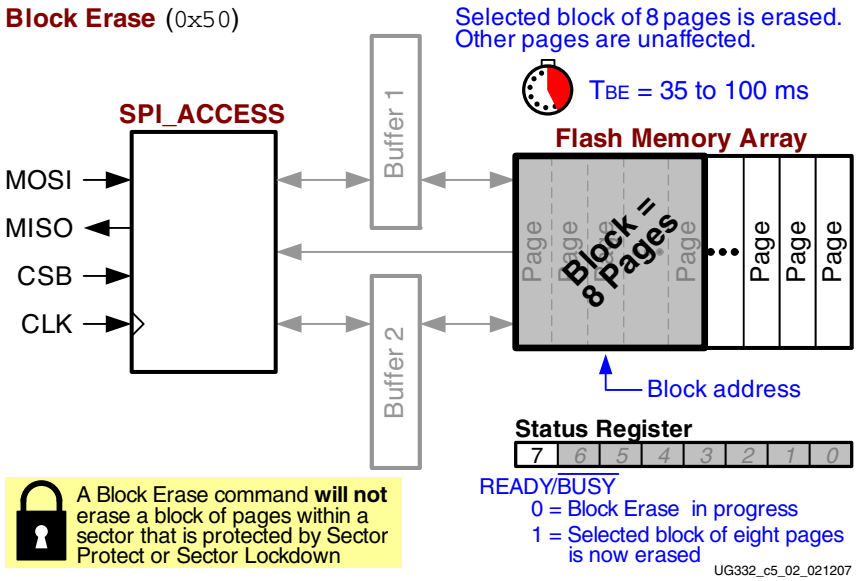

Figure 5-2: **Block Erase Command**

To perform a Block Erase command, the FPGA application must perform the following actions.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the appropriate Block Erase command code, 0x50, most-significant bit first, as shown in Table 5-5.

Table 5-5: **Block Erase (0x50) Command**

Pin	Command	24-bit Address		
		High Address	Middle Address	Low Address
	Command	Block Address		Byte Address Unused
Pin	Byte 1	Byte 2	Byte 3	Byte 4
MOSI	0x50	**Default Addressing:** See Table 5-7 **Optional Power-of-2 Addressing:** See Table A-5, page 90		Don't Care XX

Notes:

1. Note: The Block Erase command is not supported in simulation.

- Similarly, serially clock in a 24-bit block address.
  - Only the block address is required.
    - The block address is always aligned to every 8 pages. It does not span across just any block of eight contiguous pages.
  - The number of blocks and the alignment of the block address within the 24-bit address field varies by Spartan-3AN FPGA type, as shown in Table 5-6.
  - If using the default address scheme, see Table 5-7.
  - If using power-of-2 addressing, see Table A-5, page 90.
- Drive CSB High on the falling edge of CLK to end the command.

Table 5-6: **Block Addressing Summary**

FPGA	Total Blocks	Addressing Mode			
		Default		Power-of-2	
		Block Size (Bytes)	Alignment	Block Size (Bytes)	Alignment
XC3S50AN	64	2,112 (2K+64)	Every 8 pages Every 4K bytes	2,048 (2K)	Every 8 pages Every 2K bytes
XC3S200AN XC3S400AN	256				
XC3S700AN	512				
XC3S1400AN	512	4,224 (4K+128)	Every 8 pages Every 8K bytes	4,096 (4K)	Every 8 pages Every 4K bytes

Table 5-7: **Block Addressing, Default Addressing Mode**

FPGA	High Address								Middle Address								Low Address							
	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
3S50AN	0	0	0	0	0	0	Block Address						Don't Care bits											
							BL5	BL4	BL3	BL2	BL1	BL0	X	X	X	X	X	X	X	X	X	X	X	X
3S200AN 3S400AN	0	0	0	0	Block Address								Don't Care bits											
					BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0	X	X	X	X	X	X	X	X	X	X	X	X
3S700AN	0	0	0	Block Address									Don't Care bits											
				BL8	BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0	X	X	X	X	X	X	X	X	X	X	X	X
3S1400AN	0	0	Block Address										Don't Care bits											
			BL8	BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0		X	X	X	X	X	X	X	X	X	X	X	X

A Low-to-High transition on the CSB input completes the command and the Flash then erases the selected block of 8 pages. The erase operation is internally self-timed and completes in the time shown in Table 5-8 and specified in the Spartan-3AN FPGA data sheet. During this time, the READY/BUSY bit (bit 7) of the Status Register indicates whether the Block Erase operation is in progress or whether it has completed.

Table 5-8: **Block Erase Time**

Symbol	Description	FPGA	Typ	Max	Units
T_{BE}	Block Erase Time	XC3S50AN	15	35	ms
		XC3S200AN XC3S400AN	30	75	ms
		XC3S700AN XC3S1400AN	45	100	ms

While the Block Erase operation is in progress, the FPGA can access any other portion of the ISF memory, including any of the following commands.

- Read from or write to an SRAM page buffer, which are not used during an erase operation.
  - Buffer Read
  - Buffer Write
- Status Register Read
- Information Read

Sector Erase

The Sector Erase command erases any unprotected, unlocked sector in the main memory, as shown in Figure 5-3. Only one sector can be erased at one time.

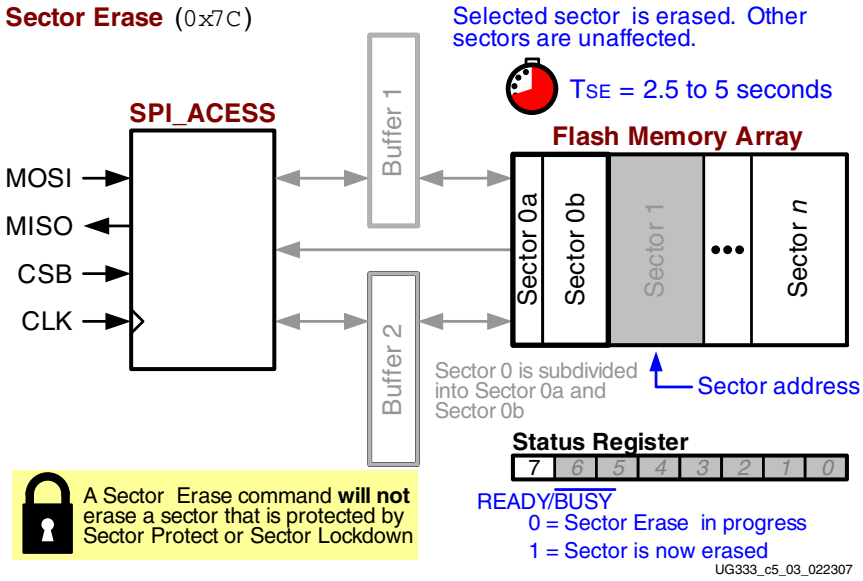

Figure 5-3: **Sector Erase Command**

To perform a Sector Erase command, the FPGA application must perform the following actions.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the appropriate Block Erase command code, 0x7C, most-significant bit first, as shown in Table 5-9.

Table 5-9: **Sector Erase (0x7C) Command**

Command	24-bit Address		
	High Address	Middle Address	Low Address
	Block Address		Byte Address Unused
0x7C	**Default Addressing:** See Table 5-10 **Optional Power-of-2 Addressing:** See Table A-6, page 90 **Full Address Map**, including bitstream locations: See Table 5-11 and Table 5-12.		Don't Care XX

Notes:

1. : The Sector Erase command is supported in simulation.

- Similarly, serially clock in a 24-bit sector address.
  - Sector 0 is different than all other sectors.
    - Sector 0 is subdivided into two subsectors called Sector 0a and Sector 0b.
    - Sector 0a and Sector 0b are each erased individually and require a unique address with additional addressing bits, different than all the other sectors.

- The page address and byte address bits, which follow the Sector Address, specify any valid address location within the sector which is to be erased.
- If using the default address scheme, see Table 5-10.
- If using power-of-2 addressing, see Table A-6, page 90.
- When erasing a sector, be aware of how the FPGA bitstream is stored within the ISF memory. There is always one bitstream stored starting at Sector 0, as indicated in blue in Table 5-11 and Table 5-12, page 60. If using MultiBoot configuration, a second FPGA bitstream is stored starting on the next sector boundary, following the first bitstream. The optional MultiBoot bitstream is shown in yellow in Table 5-11 and Table 5-12, page 60.

- Drive CSB High on the falling edge of CLK to end the command.

Sector Addressing

The addressing to access a sector depends on the address mode.

- "Default Addressing Mode," page 58
- "Optional Power-of-2 Addressing Mode," page 87

Furthermore, Table 5-11 and Table 5-12, page 60 summarize the sector addresses for each Spartan-3AN FPGA and also indicate where the FPGA configuration bitstream is stored.

Also see "Memory Allocation Tables," page 20.

Default Addressing Mode

Table 5-10 shows the sector addressing for default addressing mode. Sector 0 is subdivided into two subsectors, designated as Sector 0a and Sector 0b. These subsectors require additional address bits.

Table 5-10: **Sector Addressing, Default Addressing Mode**

FPGA / Sector		High Address								Middle Address								Low Address							
		23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XC3S50AN	**Sector**	0	0	0	0	0	0	**Sector**		Don't Care bits															
	Sector 0a							**0**	**0**	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X
	Sector 0b												1												
	Sectors 1– 3							SA1	SA0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
XC3S200AN XC3S400AN	**Sector**	0	0	0	0	**Sector**				Don't Care bits															
	Sector 0a					**0**	**0**	**0**	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X
	Sector 0b												1												
	Sectors 1– 7					SA2	SA1	SA0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
XC3S700AN	**Sector**	0	0	0	**Sector**					Don't Care bits															
	Sector 0a				**0**	**0**	**0**	**0**	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X
	Sector 0b												1	X	X	X	X	X	X	X	X	X	X	X	X
	Sectors 1– 15				SA3	SA2	SA1	SA0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
XC3S1400AN	**Sector**	0	0	**Sector**						Don't Care bits															
	Sector 0a			**0**	**0**	**0**	**0**	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X
	Sector 0b											1	X	X	X	X	X	X	X	X	X	X	X	X	X
	Sectors 1– 15			SA3	SA2	SA1	SA0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 5-11: **XC3S50AN, XC3S200AN, XC3S400AN Sector Boundaries**

Spartan-3AN FPGA			XC3S50AN			XC3S200AN		XC3S400AN	
Pages per Sector			128			256			
Addressing			Default	Power-of-2		Default	Power-of-2	Default	Power-of-2
Sector / Size		Page	264	256	Page	264	256	264	256
0	0a	0	0x00_0000	0x00_0000	0	0x00_0000	0x00_0000	0x00_0000	0x00_0000
	0b	8	0x00_1000	0x00_0800	8	0x00_1000	0x00_0800	0x00_1000	0x00_0800
1		128	0x01_0000	0x00_8000	256	0x02_0000	0x01_0000	0x02_0000	0x01_0000
2		256	0x02_0000	0x01_0000	512	0x04_0000	0x02_0000	0x04_0000	0x02_0000
3		384	0x03_0000	0x1_8000	768	0x06_0000	0x03_0000	0x06_0000	0x03_0000
4		—	—	—	1,024	0x08_0000	0x04_0000	0x08_0000	0x04_0000
5		—	—	—	1,280	0x0A_0000	0x05_0000	0x0A_0000	0x05_0000
6		—	—	—	1,536	0x0C_0000	0x06_0000	0x0C_0000	0x06_0000
7		—	—	—	1,792	0x0E_0000	0x07_0000	0x0E_0000	0x07_0000

Table 5-12: **XC3S700AN, XC3S1400AN Sector Boundaries**

Spartan-3AN FPGA			XC3S700AN		XC3S1400AN	
Pages per Sector			256			
Addressing			Default	Power-of-2	Default	Power-of-2
Sector / Size		Page	264	256	528	512
0	0a	0	0x00_0000	0x00_0000	0x00_0000	0x00_0000
	0b	8	0x00_1000	0x00_0800	0x00_2000	0x00_1000
1		256	0x02_0000	0x01_0000	0x04_0000	0x02_0000
2		512	0x04_0000	0x02_0000	0x08_0000	0x04_0000
3		768	0x06_0000	0x03_0000	0x0C_0000	0x06_0000
4		1,024	0x08_0000	0x04_0000	0x10_0000	0x08_0000
5		1,280	0x0A_0000	0x05_0000	0x14_0000	0x0A_0000
6		1,536	0x0C_0000	0x06_0000	0x18_0000	0x0C_0000
7		1,792	0x0E_0000	0x07_0000	0x1C_0000	0x0E_0000
8		2,048	0x10_0000	0x08_0000	0x20_0000	0x10_0000
9		2,304	0x12_0000	0x09_0000	0x24_0000	0x12_0000
10		2,560	0x14_0000	0x0A_0000	0x28_0000	0x14_0000
11		2,816	0x16_0000	0x0B_0000	0x2C_0000	0x16_0000
12		3,072	0x18_0000	0x0C_0000	0x30_0000	0x18_0000
13		3,328	0x1A_0000	0x0D_0000	0x34_0000	0x1A_0000
14		3,584	0x1C_0000	0x0E_0000	0x38_0000	0x1C_0000
15		3,840	0x1E_0000	0x0F_0000	0x3C_0000	0x1E_0000

Default Bitstream **Optional MultiBoot Bitstream**

Operation Timing

A Low-to-High transition on the CSB input completes the command and the Flash then erases the selected block of 8 pages. The erase operation is internally self-timed and completes in the time shown in Table 5-13 and specified in the Spartan-3AN FPGA data sheet. During this time, the READY/BUSY bit (bit 7) of the Status Register indicates whether the Sector Erase operation is in progress or whether it has completed.

Table 5-13: **Sector Erase Time**

Symbol	Description	FPGA	Typ	Max	Units
T_{SE}	Sector Erase Time	XC3S50AN	0.8	2.5	seconds
		XC3S200AN XC3S400AN XC3S700AN XC3S1400AN	1.6	5	seconds

While the Sector Erase operation is in progress, the FPGA can access any other portion of the ISF memory, including any of the following commands.

- Read from or write to an SRAM page buffer, which are not used during an erase operation.
  - Buffer Read
  - Buffer Write
- Status Register Read
- Information Read

Chapter 6

Status and Information Commands

The Spartan®-3AN FPGA In-System Flash (ISF) memory provides status and information using two different commands.

- The Status Register provides status about the current state and configuration of the ISF memory.
- The Information Read provide JEDEC-compatible device information.

Status Register

Table 6-1: **Status Register Contents**

Bit	7	6	5	4	3	2	1	0
Name	READY/ $\overline{\text{BUSY}}$	COMPARE	ISF Memory Size				SECTOR PROTECT	PAGE SIZE
Description	**0** = Busy **1** = Ready	**0** = Matches **1** = Different	`0011` = 1 Mbit: XC3S50AN `0111` = 4 Mbit: XC3S200AN or XC3S400AN `1001` = 8 Mbit: XC3S700AN `1011` = 16 Mbit: XC3S1400AN				**0** = Open **1** = Protected	**0** = Extended (Default) **1** = Power-of-2

As shown in Table 6-1, the Status Register describes…

- whether the ISF memory is ready to receive additional commands or is busy completing the current command
- the result of a Page to Buffer Compare (Program Verify) command
- the size of the ISF memory array
- whether Sector Protection is presently enabled or not
- the current addressing mode, and consequently the size of each memory page

READY/$\overline{\text{BUSY}}$

READY/BUSY status of the In-System Flash (ISF) memory is indicated by bit 7 in the Status Register, as defined in Table 6-2.

Table 6-2: **READY/$\overline{\text{BUSY}}$ (Status Register Bit 7)**

READY/$\overline{\text{BUSY}}$	Description
0	**BUSY:** The ISF memory is busy completing a command that accesses the internal array. Even while the ISF memory is busy, the FPGA application can issue select commands, as described below.
1	**READY:** The ISF memory is ready to accept any command. Any previous commands have completed.

There are several commands that cause a busy condition.

- Page to Buffer Transfer
- Page to Buffer Compare (Program Verify)
- Buffer to Page Program with Built-in Erase
- Buffer to Page Program without Built-in Erase
- Page Program Through Buffer
- Page Erase
- Block Erase
- Sector Erase
- Information Read
- Auto Page Rewrite

Even while the ISF memory is busy, the FPGA application can potentially issue the following commands.

- Read from or write to an SRAM page buffer that is not involved in the present operation.
  - Buffer Read
  - Buffer Write
- Status Register Read
- Information Read

Compare

The result of the most recent Page to Buffer Compare command is indicated by bit 6 in the Status Register, as defined in Table 6-3.

Table 6-3: **COMPARE (Status Register Bit 6)**

COMPARE	Description
0	The data stored in the ISF memory page specified by the most-recent Page to Buffer Compare (Program Verify) command matches the data stored in the specified SRAM page buffer.
1	One or more bits differ between the data stored in the ISF memory page and the SRAM page buffer specified by the most-recent Page to Buffer Compare (Program Verify) command.

ISF Memory Size

The Status Register bits 5:2 indicate the size of the ISF memory array, as defined in Table 6-4. This memory density code is different from the code used in the JEDEC device ID information, shown in Table 6-11, page 69.

Table 6-4: **ISF Memory Size (Status Register Bits 5, 4, 3, and 2)**

Status Register Bits				Memory Size	Associated FPGA(s)
5	4	3	2		
0	0	1	1	1 Mbit	XC3S50AN
0	1	1	1	4 Mbit	XC3S200AN XC3S400AN
1	0	0	1	8 Mbit	XC3S700AN
1	0	1	1	16 Mbit	XC3S1400AN

Sector Protect

Bit 1 in the Status Register indicates whether Sector Protection is presently enabled or disabled, as defined in Table 6-5. When enabled, the designated sectors specified in the Sector Protection Register are protected against all programming and erase operations.

Table 6-5: **SECTOR PROTECT (Status Register Bit 1)**

SECTOR PROTECT	Description
0	All memory locations open for program and erase commands.
1	All programming and erase commands are prevented to ISF memory locations within the Sectors defined in the Sector Protection Register.

The Sector Protect bit does not indicate whether any sectors are locked down. To determine if any sectors are locked down, issue a Sector Lockdown Register Read command.

Page Size

Bit 0 in the Status Register indicates the size of each ISF memory page and the addressing mode to access data within the ISF memory array, as defined in Table 6-6.

Table 6-6: **PAGE SIZE (Status Register Bit 0)**

PAGE SIZE	Addressing Mode	FPGA	Page Size
0	Default Addressing Mode	XC3S50AN XC3S200AN XC3S400AN XC3S700AN	264
		XC3S1400AN	528
1	Optional Power-of-2 Addressing Mode	XC3S50AN XC3S200AN XC3S400AN XC3S700AN	256
		XC3S1400AN	512

Status Register Read

The FPGA application can read the Status Register at any time, including during any internally self-timed program or erase operation. To read the Status Register, the FPGA application must perform the following operations using the SPI_ACCESS design primitive.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in Status Register Read command code, `0xD7`, most-significant bit first.

Table 6-7: **Status Register Read (`0xD7`) Command**

Pin	Command	Read Status Byte		
	Byte 1	Byte 2	...	Byte *n*
MOSI	`0xD7`	XX	...	XX
MISO	High	Status Byte	...	Most-recent Status Byte

Notes:

1. The Status Register Read command is supported in simulation.

- On the clock cycle following the last bit of the command code, the ISF memory presents the Status Register byte value on the SPI_ACCESS MISO pin.
  - Clock out the eight status bits, most-significant bit first.
  - While CSB is Low, the continuously updated Status Register value is repeated every eight CLK clock cycles. For example, this capability allows the FPGA application to continuously monitor the READY/BUSY bit of the Status Register until it returns to '1'.
- To end the data transfer, drive CSB High on the falling edge of CLK. The CSB control can be deasserted at any time as the command does not require the FPGA application to read the entire Status Register value.

See also Figure 1-3, page 12 for an example waveform for this command.

Information Read

The ISF memory supports JEDEC standards to enable systems and software to electronically query and identify the device while it is in system. The ISF Information Read command complies with the JEDEC method, "*Manufacturer and Device ID Read Methodology for SPI Compatible Serial Interface Memory Devices.*" This method identifies various attributes about the Flash memory, including the following information.

- Memory manufacturer
- Vendor-specific device family identifier
- The vendor-specific device identifier for the specified family
- The number of bits stored per memory cell
- The product version
- The number of additional Extended Device Information bytes.

To read the ISF Information, the FPGA application must perform the following operations using the SPI_ACCESS design primitive.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the Information Read command code, `0x9F`, most-significant bit first.

Table 6-8: **Information Read Command**

		JEDEC Manufacturer and Device Identifier			
	Command	Manufacturer ID	Device ID		Extended Info
Pin	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
MOSI	`0x9F`	XX	XX	XX	XX
MISO	High	`0x1F`	Family Code/Memory Density Code	`0x00`	`0x00`

Notes:

1. The information read command is supported in simulation.

- On the clock cycle following the last bit of the command code, the ISF memory presents the first byte of the ISF Information value on the SPI_ACCESS MISO output pin.
  - Clock out each information byte, most-significant bit first.
  - The first byte contains the Manufacturer ID, as shown in Table 6-10.
  - The next two bytes contain the Device ID, as shown in Table 6-11 and Table 6-12.
  - The next bytes specify the Extended Device Information String Length, which is `0x00` indicating that no additional information follows. As indicated in the JEDEC standard, reading the Extended Device Information String Length and any subsequent data is optional.
- To end the data transfer, drive CSB High on the falling edge of CLK. The CSB control can be deasserted at any time and does not require the FPGA application to read the entire ISF Information value.

The Spartan-3AN In-System Flash memory is architecturally similar to the Atmel DataFlash SPI Flash memories, as shown in Table 6-9. Consequently, the ISF Information relates to the Atmel DataFlash product family.

Table 6-9: **Spartan-3AN ISF Memory and Similar Atmel DataFlash Memory**

Spartan-3AN FPGA	Similar Atmel DataFlash Memory
XC3S50AN	AT45DB011D
XC3S200AN	AT45DB041D
XC3S400AN	
XC3S700AN	AT45DB081D
XC3S1400AN	AT45DB161D

Manufacturer Identifier

The first byte of the ISF Information, shown in Table 6-10, provides the JEDEC-assigned manufacturer code. This byte is identical for all Spartan-3AN FPGAs. The JEDEC identifier for the ISF memory is `0x1F`. This value is different than the JEDEC code for Xilinx, which is `0x49`. The Spartan-3AN FPGA product identifier is available via JTAG.

Table 6-10: **First Byte: Manufacturer Identifier**

FPGA	JEDEC Assigned Code (0x1F)							
	7	6	5	4	3	2	1	0
All	0	0	0	1	1	1	1	1

Family Code/Memory Density Code

The second byte includes a portion of the device identifier, as shown in Table 6-11.

- The three-bit **Family Code** is set to binary 001.
- The five-bit **Memory Density Code** indicates the size of ISF memory array and differs between various FPGA densities, as shown in Table 6-11. This value differs from the Status Register density code, shown in Table 6-4, page 65.

Table 6-11: **Second Byte: Device Identifier (Part 1 of 2)**

FPGA	Family Code			Memory Density Code				
	7	6	5	4	3	2	1	0
XC3S50AN (1 Mbit)	0	0	1	0	0	0	1	0
XC3S200AN (4 Mbit)				0	0	1	0	0
XC3S400AN (4 Mbit)								
XC3S700AN (8 Mbit)				0	0	1	0	1
XC3S1400AN (16 Mbit)				0	0	1	1	0

Memory Type/Product Version Code

The third byte includes the remainder of the device identifier, as shown in Table 6-12.

- The three-bit **Multi-Level Cell (MLC) Code** is binary 000, indicating that one bit is stored per memory cell.
- The five-bit **Product Version Code** indicates the ISF revision. This value is currently binary 00000, but can change in future revisions.

Table 6-12: **Third Byte: Device Identifier (Part 2 of 2)**

FPGA	Multi-Level Cell Code			Product Version Code				
	7	6	5	4	3	2	1	0
All	0	0	0	0	0	0	0	0

Extended Device Information Field

No additional ISF Information is included. Consequently, the length of the Extended Device Information field is set to 0x00, as shown in Table 6-13.

Table 6-13: **Fourth Byte: Extended Device Information String Length**

FPGA	Byte Count							
	7	6	5	4	3	2	1	0
All	0	0	0	0	0	0	0	0

Chapter 7

Power Management

The Spartan®-3AN FPGA In-System Flash (ISF) memory consumes power on the 3.3V V_{CCAUX} power rail to the Spartan-3AN FPGA. Note that write commands are not allowed until 20 ms after VCCAUX has reached at least 2.5V. In general the ISF adds no significant current to the FPGA current. Standby power consumption is approximately 100 µA, and is included in the quiescent I_{CCAUX} numbers in the Spartan-3AN FPGA data sheet. Standby mode applies whenever the ISF memory is de-selected (CSB pin is High for 35 µs or longer). Table 7-1 outlines the ISF dynamic current consumption requirements.

Table 7-1: **Power-Related ISF Memory Guidelines (V_{CCAUX}=3.6V)**

Description	Condition	Typ	Max	Units
Active current during read operation from In-System Flash memory array	F_{CLK} = 20 MHz	7	10	mA
	F_{CLK} = 33 MHz	8	12	mA
	F_{CLK} = 50 MHz	11	15	mA
Active current during read operation from SRAM page buffer	F_{CLK} = 20 MHz	–	20	mA
Active current during an In-System Flash program or erase operation	–	12	17	mA

Notes:

1. Approximate values added to I_{CCAUX} in addition to FPGA requirements.

Active Mode

The ISF memory is in the Active mode whenever the FPGA application drives the SPI_ACCESS CSB pin Low. The amount of power consumed depends on the operation and the CLK frequency, as shown in Table 7-1.

Standby Mode

The ISF memory initially powers up in Standby mode. The ISF memory also automatically re-enters Standby mode whenever the CSB input on the SPI_ACCESS primitive is High longer than 35 µs. The associated current drops to approximately 100 µA.

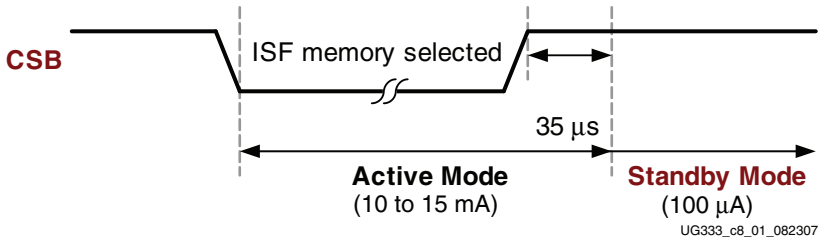

Figure 7-1: **ISF Memory Enters Standby Mode when CSB is High Longer than 35 µs**

Thermal Considerations

The ISF memory has negligible effect on the thermal considerations for the FPGA. The junction temperature of the ISF memory will be the same as the junction temperature calculated for the FPGA.

Chapter 8

Sector-Based Program/Erase Protection

The Spartan®-3AN FPGA In-System Flash (ISF) memory supports various mechanisms to protect stored data against accidental or intentional changes.

- The **Sector Protection** feature provides the ability to selectively write-protect individual ISF memory sectors.
- The **Sector Lockdown** feature permanently locks a selected ISF memory sector, essentially converting the Flash memory into read-only ROM. Once a sector is locked down, it cannot be erased or modified.

Table 8-1 describes how these features affect a specific memory sector and how the features interact together.

Table 8-1: **Sector Protect and Sector Lockdown Affect Memory Sector Program/Erase Functionality**

Sector Protect Feature		Sector Lockdown	Functionality			Description
Global Control	Sector-Specific Control					
Sector Protect Enable/Disable Status(1)	Sector Protect Status(2)	Sector Lockdown Status(3)	READ	PROGRAM	ERASE	
N/A	OPEN	OPEN	◆	◆	◆	All operations are possible. The FPGA application can read, program, and erase all locations in the sector.
DISABLE	PROTECTED	OPEN	◆	◆	◆	
ENABLE	PROTECTED	OPEN	◆			Temporarily Read-Only. The FPGA application can only read locations in the sector. In order to program or erase the sector, the application must either issue the Sector Protection Disable command or issue the Sector Protection Register Erase command followed by a Sector Protection Register Program command where the sector protection control byte is changed to OPEN.
N/A	N/A	LOCKED	◆			Permanently Read-Only. The sector can never again be programmed or erased.

Notes:

1. Issue the Sector Protection Enable command to enable the protection settings defined in the Sector Protection Register. Issue the Sector Protection Disable command to unprotect all sectors.
2. The sector protection setting is defined by the Sector Protection Register. To program a new value, the FPGA application must first issue the Sector Protection Register Erase command followed by a Sector Protection Register Program command.
3. To lock a specific sector, issue the Sector Lockdown Program command.

Sector Protection

Using Sector Protection, the FPGA application protects selected memory sectors against erroneous program and erase cycles. There are five commands associated with the Sector Protection feature, as summarized in Table 8-2.

Table 8-2: **Sector Protection Commands**

Command	Function	Hexadecimal Command Sequence
Sector Protection Register Read	Read the contents of the Sector Protection Register to determine which ISF memory sectors are protected against accidental erase or program operations, assuming that protecting is enabled using the Sector Protection Enable command.	0x32
Sector Protection Register Erase	Erase the Sector Protection Register. Required before programming the Sector Protection Register.	0x3D + 0x2A + 0x7F + 0xCF
Sector Protection Register Program	Program the control bytes within the Sector Protection Register to protect selected sectors against program or erase operations. Erase the Sector Protection Register before programming.	0x3D + 0x2A + 0x7F + 0xFC
Sector Protection Enable	Enables protection for the sectors specified in the Sector Protection Register.	0x3D + 0x2A + 0x7F + 0xA9
Sector Protection Disable	Disables protection for all sectors.	0x3D + 0x2A + 0x7F + 0x9A

Sector Protection Status at Power-Up

At power-up, the Sector Protection feature is disabled, meaning that the FPGA application has full program and erase access to all sectors.

If the application employs the Sector Protection feature, then the FPGA application should issue the Sector Protection Enable command immediately after the FPGA is configured.

Sector Protection Register

The nonvolatile Sector Protection Register specifies which sectors are presently protected or unprotected. The Sector Protection Register contains between four to 16 bytes of data, depending on the specific Spartan-3AN FPGA as shown in Table 8-3. The Sector Protect control location directly corresponds to the associated ISF memory sector. For example, control byte 1 protects or unprotects to Sector 1, etc.

The specification for Sector 0 differs from the other sectors because Sector 0 is actually sub-divided into two smaller sectors of different sizes. The byte-level description for the Sector 0 control appears in Table 8-4.

The FPGA application can modify the Sector Protection Register contents, although the register must first be erased using the Sector Protection Register Erase command.

Sector Protection is not active until the Sector Protection Register is programmed and the FPGA application issues the Sector Protection Enable command.

As shipped from Xilinx, all ISF memory sectors are unprotected (0x00).

Table 8-3: **Sector Protection Register (one control byte per sector)**

Available Sectors						Sector/ Control Byte	Protection Status	
XC3S50AN		XC3S200AN XC3S400AN		XC3S700AN XC3S1400AN			PROTECT	OPEN
4 Sectors	◆	8 Sectors = 8 Bytes	◆	16 Sectors = 16 Bytes	◆	0 (special)	(see Table 8-4)	
	◆		◆		◆	1	0xFF	0x00
	◆		◆		◆	2		
	◆		◆		◆	3		
N/A			◆		◆	4		
N/A			◆		◆	5		
N/A			◆		◆	6		
N/A			◆		◆	7		
N/A		N/A			◆	8		
N/A		N/A			◆	9		
N/A		N/A			◆	10		
N/A		N/A			◆	11		
N/A		N/A			◆	12		
N/A		N/A			◆	13		
N/A		N/A			◆	14		
N/A		N/A			◆	15		

Table 8-4: **Sector 0a and Sector 0b Protection Settings (Byte 0 in Table 8-3)**

Sector Function	Sector 0a		Sector 0b		Don't Care				Hex Value
	7	6	5	4	3	2	1	0	
Sectors 0a and 0b unprotected	0	0	0	0	X	X	X	X	0x00
Protect Sector 0a	1	1	0	0	X	X	X	X	0xC0
Protect Sector 0b	0	0	1	1	X	X	X	X	0x30
Protect both Sector 0a and Sector 0b	1	1	1	1	X	X	X	X	0xF0

Sector Protection Register Erase

The Sector Protection Register must be erased before it can be programmed. Use the Sector Protection Register Erase command sequence to erase the register.

To issue the Sector Protection Register Erase command sequence, the FPGA application must perform the following actions using the SPI_ACCESS design primitive.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the four-byte Sector Protection Register Erase command sequence shown in Table 8-5 on the MOSI pin, most-significant bit of each byte first.

Table 8-5: **Sector Protection Register Erase Command Sequence**

Pin	Four-byte Command Sequence			
	Byte 1	Byte 2	Byte 3	Byte 4
MOSI	0x3D	0x2A	0x7F	0xCF

- After clocking in the last bit of the command sequence, deassert the CSB pin High.

A Low-to-High transition on the CSB input completes the command sequence and the ISF memory then erases the Sector Protection Register. The erase operation is internally self-timed and completes in the Page Erase time, T_{PE}, shown in Table 5-4, page 53 and specified in the Spartan-3AN FPGA data sheet. During this time, the READY/BUSY bit (bit 7) of the Status Register indicates whether the Sector Protection Register Erase operation is in progress or whether it has completed.

If the FPGA V_{CCAUX} power supply is interrupted before the erase cycle completes, then the content of the Sector Protection Register is not guaranteed.

The Sector Protection Register can be erased regardless if sector protection is enabled or disabled. The erased state of each byte in the Sector Protection Register is 0xFF, which protects each sector. Leave sector protection enabled while erasing of the Sector Protection Register, which then prevents accidental programming or erasing of the device. If the FPGA application should issue an erroneous program or erase command immediately after erasing the Sector Protection Register and before the register is reprogrammed, then the erroneous program or erase command is prevented.

Sector Protection Register Program

Once the Sector Protection Register is erased, reprogram the register using the Sector Protection Register Program command sequence.

To issue the Sector Protection Register Program command sequence, the FPGA application must perform the following actions using the SPI_ACCESS design primitive.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the four-byte Sector Protection Register Program command sequence shown in Table 8-6 on the MOSI pin, most-significant bit of each byte first.

Table 8-6: **Sector Protection Register Program Command Sequence**

Pin	Four-byte Command Sequence				Sector Protection Register Value (byte locations corresponds to sector)							
					XC3S700AN, XC3S1400AN (16 bytes)							
					XC3S200AN, XC3S400AN (8 bytes)							
					XC3S50AN (4 bytes)							
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	...	Byte 12	...	Byte 20
MOSI	0x3D	0x2A	0x7F	0xFC	Sector 0	Sector 1	Sector 2	Sector 3	...	Sector 7	...	Sector 15

- After clocking in the last bit of the command sequence, send the Sector Protection Register programming data on the MOSI pin.
  - The number of bytes depends on the Spartan-3AN FPGA, as highlighted in Table 8-6 and Table 8-3, page 75.

- The first data byte corresponds to Sector 0, the second data byte to Sector 1, and so on.
- The XC3S50AN FPGA requires four bytes.
- The XC3S200AN and the XC3S400AN FPGAs each require 8 bytes.
- The XC3S700AN and the XC3S1400AN FPGAs require 16 bytes.
- If the proper number of data bytes is not clocked in before the CSB pin is deasserted, then the protection status of the sectors corresponding to the unwritten bytes is not guaranteed. For example, if the FPGA application only writes two bytes, then the protection status for the subsequent sectors is not guaranteed.
- If the FPGA application writes more than required number of bytes to the Sector Protection Register, then the data wraps back around to the beginning of the register. For example, if the application writes five bytes to the XC3S50AN, then the fifth byte actually overwrites the value for Sector 0.

♦ Sector 0 is subdivided into two smaller sectors, called Sector 0a and Sector 0b. See Table 8-4, page 75 for information on how to protect or unprotect Sector 0 locations.

♦ All sectors, other than Sector 0, use the same command byte, as shown in Table 8-3, page 75.

- Write `0x00` to unprotect a sector.
- Write `0xFF` to protect a sector.
- If a value other than `0x00` or `0xFF` is clocked into the Sector Protection Register, then the protection status of the corresponding sector is not guaranteed.

- After clocking in the last data bit to program the register, drive the CSB pin High on the falling edge of CLK to end the command.

Caution! The Sector Protection Register Program command uses the Buffer 1 SRAM page buffer. Any data stored in Buffer 1 prior to issuing the Sector Protection Register Program command is destroyed.

A Low-to-High transition on the CSB input completes the command sequence and the ISF memory then programs the Sector Protection Register. The erase operation is internally self-timed and completes in the Page Programming time, T_{PP}, or between 4 to 6 ms as shown in Table 4-5, page 44 and specified in the Spartan-3AN FPGA data sheet. During this time, the READY/BUSY bit (bit 7) of the Status Register indicates whether the Sector Protection Register Program operation is in progress or whether it has completed.

If the FPGA V_{CCAUX} power supply is interrupted before the programming operation completes, then the content of the Sector Protection Register is not guaranteed.

The FPGA application can always erase and program the Sector Protection Register regardless of the sector protection status.

Unprotecting Sectors While Sector Protection Enabled

Reprogramming the Sector Protection Register while sector protection is enabled allows the FPGA application to temporarily unprotect an individual sector or sectors rather than disabling the sector protection mechanism completely. However, there are limitations to this technique as described below.

Sector Protection Register Limited to 10,000 Program/Erase Cycles

The Sector Protection Register is limited to 10,000 erase/program cycles. If, as described above, the FPGA application modifies the Sector Protection Register to temporarily unprotect an individual sector or sectors, then the application must limit this practice to 10,000 cycles. Instead, use a combination of the techniques below.

- Temporarily unprotect individual sectors, but no more than 10,000 cycles
- Use the Sector Protection Enable and Sector Protection Disable commands to completely enable or disable the sector protection mechanism.

Sector Protection Register Read

To read the Sector Protection Register contents, issue the Sector Protection Register Read command sequence. The FPGA application must perform the following actions using the SPI_ACCESS design primitive.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the four-byte Sector Protection Register Read command sequence shown in Table 8-7 on the MOSI pin, most-significant bit of each byte first. The last 3 bytes are dummy bytes.

Table 8-7: **Sector Protection Register Read Command Sequence**

	Four-byte Command Sequence				**Sector Protection Register Value** (byte locations corresponds to sector)							
					XC3S700AN, XC3S1400AN (16 bytes)							
					XC3S200AN, XC3S400AN (8 bytes)							
	Command	**Don't Care**			XC3S50AN (4 bytes)							
Pin	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**	**...**	**Byte 12**	**...**	**Byte 20**
MOSI	0x32	XX	XX	XX	XX	XX	XX	XX	...	XX	...	XX
MISO					Sector 0	Sector 1	Sector 2	Sector 3	...	Sector 7	...	Sector 15

- After clocking in the last bit of the command sequence, read the Sector Protection Register contents on the MISO pin.
  - The number of bytes depends on the Spartan-3AN FPGA, as highlighted in Table 8-7 and Table 8-3, page 75.
    - The first data byte corresponds to Sector 0, the second data byte to Sector 1, and so on.
    - The XC3S50AN FPGA provides four bytes.
    - The XC3S200AN and the XC3S400AN FPGAs each provide 8 bytes.
    - The XC3S700AN and XC3S1400AN FPGAs provide 16 bytes.
    - If the FPGA application reads more than required number of bytes from the Sector Protection Register, any additional data provided on the MISO pin is undefined.
- After clocking the last data bit to read the register, deassert the CSB pin High.

Sector Protection Enable

The Sector Protection Enable command applies the protection level specified in the Sector Protection Register. To issue the Sector Protection Enable command sequence, the FPGA application must perform the following actions using the SPI_ACCESS design primitive.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the four-byte Sector Protection Enable command sequence shown in Table 8-8 on the MOSI pin, most-significant bit of each byte first.

Table 8-8: **Sector Protection Enable Command Sequence**

Pin	Four-byte Command Sequence			
	Byte 1	Byte 2	Byte 3	Byte 4
MOSI	0x3D	0x2A	0x7F	0xA9

- After clocking in the last bit of the command sequence, drive the CSB pin High on the falling edge of CLK. The defined sector protection level is now active.

Sector Protection Disable

The Sector Protection Disable command removes protection from all sectors. To issue the Sector Protection Disable command sequence, the FPGA application must perform the following actions using the SPI_ACCESS design primitive.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the four-byte Sector Protection Disable command sequence shown in Table 8-9 on the MOSI pin, most-significant bit of each byte first.

Table 8-9: **Sector Protection Disable Command Sequence**

Pin	Four-byte Command Sequence			
	Byte 1	Byte 2	Byte 3	Byte 4
MOSI	0x3D	0x2A	0x7F	0x9A

- After clocking in the last bit of the command sequence, drive the CSB pin High on the falling edge of CLK. Sector protection is now removed from all sectors.

Sector Lockdown

The ISF memory provides a mechanism called Sector Lockdown that permanently and irreversibly protects the contents of an individual sector against any program and erase cycles. Sector Lockdown effectively converts the erasable Flash memory sector to a one-time programmable (OTP), read-only memory (ROM).

Sector Lockdown is useful in applications to prevent malicious attempts to alter the FPGA configuration bitstream, program code, or security information. To prevent accidental changes or to have selective protection, see "Sector Protection," page 74.

> ***Caution!*** Once an ISF memory sector is locked down, it can never be erased, programmed, or unlocked. Locking down a sector is irreversible.

There are two commands associated with the Sector Lockdown feature, as shown in Table 8-10.

Table 8-10: **Sector Lockdown Commands**

Command	Function	Hexadecimal Command Sequence
Sector Lockdown Register Read	Read the contents of the Sector Lockdown Register to determine the current ISF memory settings.	0x35
Sector Lockdown Program	Lock down a specific Sector.	0x3D + 0x2A + 0x7F + 0x30

Sector Lockdown Program

To lock down a specific sector, issue a Sector Lockdown Program command, followed by a 24-bit sector address, anywhere within the memory sector to be locked down.

To issue the Sector Lockdown Program command, the FPGA application must perform the following actions using the SPI_ACCESS design primitive.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the four-byte Sector Lockdown Program command sequence shown in Table 8-11 on the MOSI pin, most-significant bit of each byte first.

Table 8-11: **Sector Lockdown Program Command Sequence**

Pin	Four-byte Command Sequence				24-bit Address		
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
MOSI	0x3D	0x2A	0x7F	0x30	Default Addressing: See Table 5-10, page 59. Power-of-2 Addressing: See Table A-6, page 90.		XX

- Immediately following the four-byte command code, serially clock in a 24-bit Sector address.
  - Sector 0 is different than all other sectors.
    - Sector 0 is subdivided into two subsectors called Sector 0a and Sector 0b.
    - Sector 0a and Sector 0b are each erased individually and require a unique address with additional addressing bits, different than all the other sectors.
  - The page address and byte address bits, which follow the Sector Address, specify any valid address location within the sector which is to be erased.
  - If using the default address scheme, see Table 5-10, page 59.
  - If using power-of-2 addressing, see Table A-6, page 90.
- After clocking in the last bit of the 24-bit address, drive the CSB pin High on the falling edge of CLK.

A Low-to-High transition on the CSB input completes the command sequence and the ISF memory then programs the Sector Lockdown Register. The erase operation is internally self-timed and completes in the Page Programming time, T_{PP}, or between 4 to 6 ms as

shown in Table 4-5, page 44 and specified in the Spartan-3AN FPGA data sheet. During this time, the READY/BUSY bit (bit 7) of the Status Register indicates whether the Sector Lockdown Program operation is in progress or whether it has completed.

If the FPGA V_{CCAUX} power supply is interrupted before the programming operation completes, then the content of the Sector Lockdown Register is not guaranteed. If power is interrupted, read the Sector Lockdown Register to determine if it was programmed correctly. If no, reissue the Sector Lockdown Program command, if necessary.

Sector Lockdown Register

The nonvolatile Sector Lockdown Register indicates which sectors are open and which are permanently locked down. The Sector Lockdown Register contains between four to 16 bytes of data, depending on the specific Spartan-3AN FPGA as shown in Table 8-12. The Sector Lockdown status byte location directly corresponds to the associated ISF memory sector. For example, status byte 1 corresponds to the lockdown status of Sector 1, etc.

The specification for Sector 0 differs from the other sectors because Sector 0 is actually sub-divided into two smaller sectors of different sizes. The byte-level description for the Sector 0 status appears in Table 8-13.

The Lockdown Status for a specific sector is updated based on a Sector Lockdown Program command. The Sector Lockdown Register is a read-only location and cannot be directly modified.

Table 8-12: **Sector Lockdown Register (Read-Only)**

<table>
<tr><th colspan="5">Available Sectors</th><th rowspan="2">Sector/
Status Byte</th><th colspan="2">Lockdown Status</th></tr>
<tr><th colspan="2">XC3S50AN</th><th colspan="2">XC3S200AN
XC3S400AN</th><th>XC3S700AN
XC3S1400AN</th><th>LOCKED</th><th>OPEN</th></tr>
<tr><td rowspan="4">4 Sectors</td><td>◆</td><td rowspan="8">8 Sectors = 8 Bytes</td><td>◆</td><td rowspan="16">16 Sectors = 16 Bytes</td><td>◆</td><td>0 (special)</td><td colspan="2">(see Table 8-13)</td></tr>
<tr><td>◆</td><td>◆</td><td>◆</td><td>1</td><td rowspan="16">0xFF</td><td rowspan="16">0x00</td></tr>
<tr><td>◆</td><td>◆</td><td>◆</td><td>2</td></tr>
<tr><td>◆</td><td>◆</td><td>◆</td><td>3</td></tr>
<tr><td colspan="2">N/A</td><td>◆</td><td>◆</td><td>4</td></tr>
<tr><td colspan="2">N/A</td><td>◆</td><td>◆</td><td>5</td></tr>
<tr><td colspan="2">N/A</td><td>◆</td><td>◆</td><td>6</td></tr>
<tr><td colspan="2">N/A</td><td>◆</td><td>◆</td><td>7</td></tr>
<tr><td colspan="2">N/A</td><td colspan="2">N/A</td><td>◆</td><td>8</td></tr>
<tr><td colspan="2">N/A</td><td colspan="2">N/A</td><td>◆</td><td>9</td></tr>
<tr><td colspan="2">N/A</td><td colspan="2">N/A</td><td>◆</td><td>10</td></tr>
<tr><td colspan="2">N/A</td><td colspan="2">N/A</td><td>◆</td><td>11</td></tr>
<tr><td colspan="2">N/A</td><td colspan="2">N/A</td><td>◆</td><td>12</td></tr>
<tr><td colspan="2">N/A</td><td colspan="2">N/A</td><td>◆</td><td>13</td></tr>
<tr><td colspan="2">N/A</td><td colspan="2">N/A</td><td>◆</td><td>14</td></tr>
<tr><td colspan="2">N/A</td><td colspan="2">N/A</td><td>◆</td><td>15</td></tr>
</table>

Table 8-13: **Sector 0a and Sector 0b Lockdown Settings (Byte 0 in Table 8-12)**

Sector Function	Sector 0a		Sector 0b		Unused				Hex Value
	7	6	5	4	3	2	1	0	
Sectors 0a and 0b unlocked	0	0	0	0	0	0	0	0	0x00
Sector 0a locked down	1	1	0	0	0	0	0	0	0xC0
Sector 0b locked down	0	0	1	1	0	0	0	0	0x30
Both Sector 0a and Sector 0b locked down	1	1	1	1	0	0	0	0	0xF0

Sector Lockdown Register Read

The FPGA application can read the Sector Lockdown Register to determine which sectors in the memory array, if any, are permanently locked down.

To read the Sector Lockdown Register contents, issue the Sector Lockdown Register Read command sequence. The FPGA application must perform the following actions using the SPI_ACCESS design primitive.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the four-byte Sector Lockdown Register Read command sequence shown in Table 8-14 on the MOSI pin, most-significant bit of each byte first. The last 3 bytes are dummy bytes.

Table 8-14: **Sector Lockdown Register Read Command Sequence**

	Four-byte Command Sequence				Sector Lockdown Register Value (byte location corresponds to sector)							
					XC3S700AN, XC3S1400AN (16 bytes)							
					XC3S200AN, XC3S400AN (8 bytes)							
	Command	Don't Care			XC3S50AN (4 bytes)							
Pin	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	...	Byte 12	...	Byte 20
MOSI	0x35	XX	XX	XX	XX	XX	XX	XX	...	XX	...	XX
MISO					Sector 0	Sector 1	Sector 2	Sector 3	...	Sector 7	...	Sector 15

- After clocking in the last bit of the command sequence, read the Sector Lockdown Register contents on the MISO pin.
  - The number of bytes depends on the Spartan-3AN FPGA, as highlighted in Table 8-14 and Table 8-3, page 75.
    - The first data byte corresponds to Sector 0, the second data byte to Sector 1, and so on.
    - The XC3S50AN FPGA provides four bytes.
    - The XC3S200AN and the XC3S400AN FPGAs each provide 8 bytes.
    - The XC3S700AN and the XC3S1400AN FPGAs provide 16 bytes.
    - If the FPGA application reads more than required number of bytes from the Sector Lockdown Register, any additional data provided on the MISO pin is undefined.
- After clocking the last data bit to read the register, drive the CSB pin High.

Chapter 9

Security Register

The Spartan®-3AN FPGA In-System Flash (ISF) memory includes a special 128-byte Security Register, shown in Table 9-1.

Security Register

The 128 bytes are further divided into two subfields.

- The **User-Defined Field** (byte locations 0 through 63), can be programmed with any value at any time, but it can only be programmed once. In other words, this field is one-time programmable (OTP). The default delivered state is erased, and all locations are `0xFF`.

  Caution! Once the 64-byte User-Defined Field is programmed, it cannot be erased or changed.

- The **Unique Identifier** (byte locations 64 through 127) is programmed during Xilinx factory manufacturing and contains a unique identifier per each ISF memory. This field is different from the Device DNA feature accessible within Spartan-3A and Spartan-3AN FPGAs. The factory-programmed data cannot be erased or changed.

Table 9-1: **Security Register Contents**

	Security Register (128 bytes)							
	User-Defined Field (64 bytes)				Factory ID (64 bytes)			
Byte	**0**	**1**	**...**	**63**	**64**	**65**	**...**	**127**
Description	64 bytes of user-defined data				Unique identifier for each ISF memory inside each Spartan-3AN FPGA			
Number of Programming Cycles	1 (this field is one-time programmable, OTP)				0 (this field is permanently programmed during Xilinx manufacturing)			

The Security Register serves a variety of potential applications, such as uniquely identifying a particular ISF memory device, for protecting FPGA bitstreams or user data, or for locked key storage. Applications and usage are similar to that for the FPGA Device DNA register. See the *Protecting FPGA Designs* chapter in UG332, Spartan-3 Generation Configuration User Guide.

Security Register Program

Only the User-Defined Field in the Security Register is programmable. This field does not need to be erased before it is programmed.

To issue the Security Register Program command sequence, the FPGA application must perform the following actions using the SPI_ACCESS design primitive.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the four-byte Security Register Program command sequence shown in Table 9-2 on the MOSI pin, most-significant bit of each byte first.

Table 9-2: **Security Register Program Command Sequence**

Pin	Four-byte Command Sequence				Security Register Value (64-byte User-Defined Field)			
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	...	Byte68
MOSI	0x9B	0x00	0x00	0x00	User-Field Data 0	User-Field Data 1	...	User-Field Data 63

Notes:

1. The Security Register Program command is supported in simulation. The simulation model has a default Factory ID of 00. For simulation purposes, a user parameter can set the Factory ID to a different value (see Table 1-3).

- After clocking in the last bit of the command sequence, send the Security Register programming data on the MOSI pin
  - Specify all 64 bytes.
  - The content of any unwritten locations is not guaranteed.
  - If more than 64 bytes are provided, then the additional data wraps around starting again at location 0.
  - Specify each byte location serially, most-significant bit first.
- After clocking in the last data bit to program the register, drive the CSB pin High on the falling edge of CLK.

A Low-to-High transition on the CSB input completes the command sequence. The programming operation is internally self-timed and completes in the Page Programming time, T_{PP}, shown in Table 4-5, page 44 and specified in the Spartan-3AN FPGA data sheet. During this time, the READY/BUSY bit (bit 7) of the Security Register indicates whether the Security Register Program operation is in progress or whether it has completed.

Caution! If the FPGA V_{CCAUX} power supply is interrupted before the erase cycle completes, then the content of the 64-byte User-Defined Field within the Security Register is not guaranteed.

Caution! If less than 64 bytes of data is clocked in before the CSB pin is deasserted, then the values for any unspecified byte locations are not guaranteed. For example, if only the first two bytes are clocked in rather than the complete 64 bytes, then the remaining 62 bytes of the User-Defined Field within the Security Register are not guaranteed.

Caution! The User-Defined Field within the Security Register can only be programmed once. Consequently, it is not possible to program only the first two bytes of the register and then later program the remaining 62 bytes.

Caution! The Security Register Program command uses the SRAM page Buffer 1 for temporary data storage. Consequently, this command overwrites any previous contents of Buffer 1.

Security Register Read

To read the Security Register, the FPGA application must perform the following actions using the SPI_ACCESS design primitive.

- Drive CSB Low while CLK is High or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the four-byte Security Register Read command sequence shown in Table 9-3 on the MOSI pin, most-significant bit of each byte first. The last 3 bytes are dummy bytes.

Table 9-3: **Security Register Read Command Sequence**

Pin	Four-byte Command Sequence				Security Register							
					User-Defined Field				Unique Identifier			
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	...	Byte 68	Byte 69	Byte 70	...	Byte 132
MOSI	0x77	XX	XX	XX	XX	XX	...	XX	XX	XX	...	XX
MISO	High				Data 0	Data 1	...	Data 63	Data 64	Data 65	...	Data 127

Notes:

1. The Security Register Read command is supported in simulation.

- After clocking in the last bit of the command sequence, receive the Security Register data on the MISO pin
  - Receive each byte location serially, most-significant bit first.
  - The FPGA application can read less than the complete 128 bytes. For example, the application can stop after reading first few bytes of the User-Defined Field.
  - Any data beyond the end of the 128-byte Security Register is undefined.
- After clocking in the last data bit to program the register, drive the CSB pin High on the falling edge of CLK.

Appendix A

Optional Power-of-2 Addressing Mode

As initially shipped, the Spartan®-3AN FPGA In-System Flash (ISF) memory uses the Default Addressing Mode, highlighted in Table 2-2, page 19. The Default Addressing Mode provides 3% more total bits, providing additional "extra" bits and bytes within each page for potential file system control and error correction applications.

For some applications, however, a power-of-2 binary addressing method may prove more natural. The ISF memory supports such an addressing mode, as highlighted in Table A-3, page 89, but does require an additional irreversible programming step, described below. Typically, this step is performed on a production tester and not by the FPGA application itself.

> ***Caution!*** The Power-of-2 Addressing Mode is not supported in simulation. Simulation only supports the Default Addressing Mode.
>
> ***Caution!*** Changing over to the power-of-2 addressing mode is irreversible. Once the FPGA's In-System Flash memory (ISF) is programmed for this mode, it cannot be changed back to the Default Addressing Mode.
>
> ***Caution!*** Immediately after Permanently Changing to the Power-of-2 Addressing Mode, the FPGA's power supply must be cycled off and on before the change takes effect.
>
> ***Caution!*** Any data stored while the FPGA ISF memory was in the Default Addressing Mode, such as FPGA configuration bitstreams, is corrupted once the ISF memory is irreversibly changed over to the Power-of-2 addressing mode. The ISF memory must be erased and re-written after changing to the optional power-of-2 addressing mode.

Table A-1 illustrates how the addressing mode affects total memory size.

Table A-1: **Default Addressing Mode vs. Power-of-2 Addressing Mode**

FPGA	Total Bits by Addressing Mode		Page Size by Addressing Mode	
	Default	Power-of-2	Default	Power-of-2
XC3S50AN	1,081,344	1,048,576	264 bytes	256 bytes
XC3S200AN	4,325,376	4,194,304		
XC3S400AN				
XC3S700AN	8,650,752	8,388,608		
XC3S1400AN	17,301,504	16,777,216	528 bytes	512 bytes

How to Determine the Current Addressing Mode

The Page Size bit, bit 0, in the Status Register indicates the current addressing mode, as outlined in Table 6-6, page 66. If required, the ISF memory programmer or the FPGA application can read the Status Register by issuing a Status Register Read command.

Permanently Changing to the Power-of-2 Addressing Mode

To permanently change over from the Default Addressing Mode to the optional power-of-2 addressing mode, the programmer or FPGA application must perform the following steps.

- Drive CSB Low while CLK is High, or on the rising edge of CLK.
- On the falling edge of CLK, serially clock in the Power-of-2 Page Size command sequence, shown in Table A-2, most-significant bit first.

Table A-2: **Power-of-2 Page Size Command Sequence**

Pin	Four-byte Command Sequence			
	Byte 1	Byte 2	Byte 3	Byte 4
MOSI	0x3D	0x2A	0x80	0xA6

- After clocking in the last bit of the command sequence, set CSB High to start the internally self-timed programming cycle.
- Wait at least the Page Programming time, T_{PP}, shown in Table 4-5, page 44 and specified in the Spartan-3AN FPGA data sheet. During this time, the READY/BUSY bit (bit 7) is '0' in the Status Register, indicating that an ISF memory programming operation is in progress.
- After the programming operation completes, the FPGA power supplies must be cycled off, then reapplied before the Power-of-2 addressing mode becomes active.
- Any data stored in the ISF memory prior to setting the Power-of-2 addressing mode will be corrupted. Erase and reprogram the ISF memory with the intended data.

If the FPGA is powered-down before the completion of the program cycle, then the Power-of-2 address setting cannot be guaranteed. Should this occur, however, check that the Page Size bit (bit 0) is set to '1' in the Status Register, indicating that the page size was successfully programmed. If not, re-issue the Power-of-2 Page Size command.

Power-of-2 Addressing Mode

All Spartan-3AN FPGAs are shipped supporting the Default Addressing Mode. The optional Power-of-2 Addressing mode is only available after Permanently Changing to the Power-of-2 Addressing Mode.

Power-of-2 Addressing

Table A-3 summarizes the address mapping for the optional power-of-2 addressing mode.

Table A-3: **Power-of-2 Addressing Mode**

FPGA	High Address								Middle Address								Low Address							
	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Binary Address	0	0	0	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
3S50AN	0	0	0	0	0	0	0	**Sector**	Sector 0a, 0b					X	X	X	X	X	X	X	X	X	X	X
								Block (64 blocks)						X	X	X	X	X	X	X	X	X	X	X
								Page Address (512 pages)									**Byte Address** (256 bytes)							
3S200AN 3S400AN	0	0	0	0	0	**Sector** (8)			Sector 0a, 0b					X	X	X	X	X	X	X	X	X	X	X
						Block (256 blocks)								X	X	X	X	X	X	X	X	X	X	X
						Page Address (2,048 pages)											**Byte Address** (256 bytes)							
3S700AN	0	0	0	0	**Sector** (16)				Sector 0a, 0b					X	X	X	X	X	X	X	X	X	X	X
					Block (512 blocks)									X	X	X	X	X	X	X	X	X	X	X
					Page Address (4,096 pages)												**Byte Address** (256 bytes)							
3S1400AN	0	0	0	**Sector** (16)					Sector 0a, 0b				X	X	X	X	X	X	X	X	X	X	X	X
				Block (512 blocks)									X	X	X	X	X	X	X	X	X	X	X	X
				Page Address (4,096 pages)													**Byte Address** (512 bytes)							
Binary Address	0	0	0	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

Power-of-2 Page Addressing

Table A-4 summarizes how page addresses are specified within the 24-bit address field.

Table A-4: **Page Address, Power-of-2 Addressing Mode**

FPGA	High Address								Middle Address								Low Address							
	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Binary Address	0	0	0	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
3S50AN	0	0	0	0	0	0	0	**Page Address** (512 pages)									X	X	X	X	X	X	X	X
3S200AN	0	0	0	0	0	**Page Address** (2,048 pages)											X	X	X	X	X	X	X	X
3S400AN	0	0	0	0	0	**Page Address** (2,048 pages)											X	X	X	X	X	X	X	X
3S700AN	0	0	0	0	**Page Address** (4,096 pages)												X	X	X	X	X	X	X	X
3S1400AN	0	0	0	**Page Address** (4,096 pages)												X	X	X	X	X	X	X	X	X

Power-of-2 Block Addressing

Table A-5 summarizes how block addresses are mapped into the 24-bit address field.

Table A-5: **Block Addressing, Power-of-2 Addressing Mode**

FPGA	High Address								Middle Address								Low Address							
	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Binary Address	0	0	0	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
XC3S50AN	0	0	0	0	0	0	0	**Block** (64 blocks)						X	X	X	X	X	X	X	X	X	X	X
XC3S200AN	0	0	0	0	0	**Block** (256 blocks)								X	X	X	X	X	X	X	X	X	X	X
XC3S400AN	0	0	0	0	0	**Block** (256 blocks)								X	X	X	X	X	X	X	X	X	X	X
XC3S700AN	0	0	0	0	**Block** (512 blocks)									X	X	X	X	X	X	X	X	X	X	X
XC3S1400AN	0	0	0	**Block** (512 blocks)									X	X	X	X	X	X	X	X	X	X	X	X

Power-of-2 Sector Addressing

Table A-6 shows the sector addressing for the Power-of-2 addressing mode. Sector 0 is subdivided into two subsectors, designated as Sector 0a and Sector 0b. These subsectors require additional address bits.

Table A-6: **Sector Addressing, Power-of-2 Addressing Mode**

FPGA / Sector		High Address								Middle Address								Low Address							
		23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Binary Address		0	0	0	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
XC3S50AN	**Sector**	0	0	0	0	0	0	0	**Sector**	Don't Care bits															
	Sector 0a								**0**	**0**	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X
	Sector 0b													1											
	Sectors 1– 3								A16	A15	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
XC3S200AN XC3S400AN	**Sector**	0	0	0	0	0	**Sector**			Don't Care bits															
	Sector 0a						**0**	**0**	**0**	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X
	Sector 0b													1											
	Sectors 1– 7						A18	A17	A16	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
XC3S700AN	**Sector**	0	0	0	0	**Sector**				Don't Care bits															
	Sector 0a					**0**	**0**	**0**	**0**	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X
	Sector 0b													1	X	X	X	X	X	X	X	X	X	X	X
	Sectors 1– 15					A19	A18	A17	A16	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
XC3S1400AN	**Sector**	0	0	0	**Sector**					Don't Care bits															
	Sector 0a				**0**	**0**	**0**	**0**	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X
	Sector 0b												1	X	X	X	X	X	X	X	X	X	X	X	X
	Sectors 1– 15				SA3	SA2	SA1	SA0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X