XAPP483 (v2.0) April 11, 2007

Multiple-Boot with Platform Flash PROMs

Author: Jameel Hussein

Summary

Some applications take advantage of the ability to change the configuration of a Xilinx FPGA at each boot-up, changing the FPGA's functionality as required. The ability to change the FPGA configuration is made easy with the Xilinx Platform Flash XCFxxP PROM's Design Revisioning feature, which allows the user to store multiple configurations as different revisions in a single PROM. With the addition of a small amount of logic internal to the FPGA, the user can dynamically switch between up to four different revisions on the PROM. Multiple-Boot, or the ability to dynamically reconfigure from multiple Design Revisions, is similar to the MultiBoot option offered with Spartan™-3E FPGAs when used with third-party parallel flash PROMs.

This application note will further describe how Platform Flash PROMs provide additional options for enhancing safety in the event of failed configuration, as well as reducing pin count and board space. Moreover, Platform Flash PROMs provide the user with additional advantages: iMPACT programming support, a single-vendor solution, low-cost board design, plus faster configuration loading.

A reference design is also detailed in this application note including VHDL source code, targeted towards Spartan-3E FPGAs.

Introduction

When combined with a small amount of logic internal to the FPGA, Platform Flash PROMs easily support applications requiring the ability to dynamically select from multiple FPGA configurations or revisions (referred to as Multiple-Boot). Multiple-Boot is achieved by utilizing the Xilinx Platform Flash feature, Design Revisioning, along with a small amount of logic internal to the FPGA. An example of an application requiring Multiple-Boot capabilities is when the FPGA needs to support both diagnostic as well as general functionality (Figure 1). In this case, the FPGA boots up using a diagnostics application to perform board-level tests. If the tests are successful, then the FPGA triggers a reconfiguration from a second bitstream containing the general functionality configuration image needed for normal operation. The general FPGA application could be designed to trigger a reconfiguration to reload the diagnostics application at any time as needed.

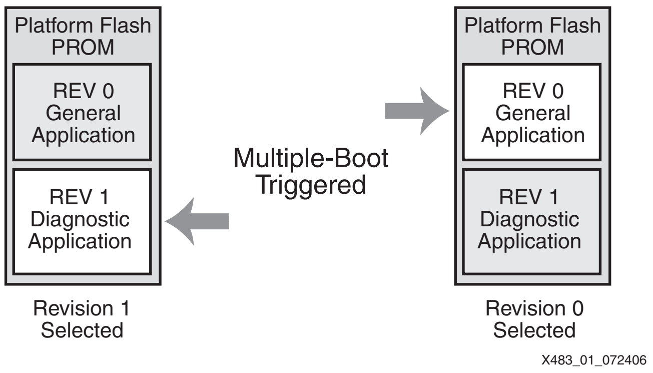


Figure 1: **Example Application Requiring Multiple-Boot Support**

In the example shown in Figure 1, page 1, the default configuration is stored in the PROM as Revision 1. This default configuration is loaded at system reset. When Multiple-Boot is triggered, the FPGA automatically reconfigures itself using the configuration image stored as Revision 0 in the PROM.

Note: The Revision Select[1:0] inputs have an internal 50 KΩ resistive pull-up to V_{CCO} to provide a logic 1 to the device if the pins are not driven.

The default configuration stored as Revision 1 could contain a "golden" or fail-safe configuration image, used to communicate with the outside world to check for a newer configuration image. If a newer configuration image exists and verifies as good, then the golden configuration could trigger a reconfiguration to load the new image.

Note: The user can choose to reconfigure the FPGA using the configuration image stored at any of up to four different revision locations.

Design Revisioning

Design Revisioning allows the user to store up to four unique configuration images on a single PROM or across multiple cascaded PROMs (Figure 2). When combined with dynamic reconfiguration, the design revision capabilities of Platform Flash PROMs allow for the creation of Multiple-Boot applications.

With Design Revisioning, each configuration image is stored at a specific revision location (0 to 3) and is supported for the 16/32 Mbit XCFxxP Platform Flash PROMs in both serial and parallel output modes. The PROM programming files along with the revision information files (`.cfi`) are created using the iMPACT software. This `.cfi` file is later required to enable Design Revision programming (refer to *Xilinx ISE 8 Software Manuals* for more details).

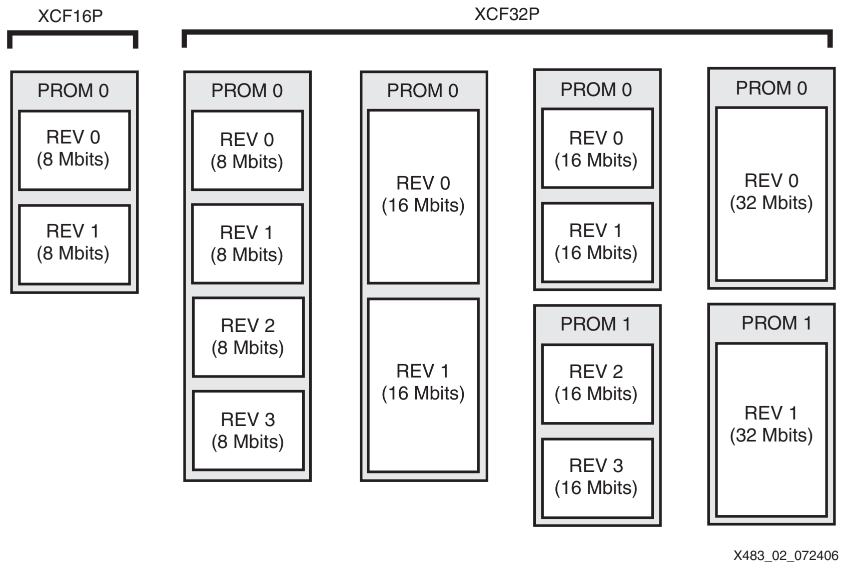


Figure 2: **Example Storage Options Available with XCFxxP Platform Flash PROMs**

After programming the Platform Flash PROM with a set of configuration images, a configuration image stored at a given revision location is selected using the external REV_SEL[1:0] pins or using the internal programmable Design Revision control bits. The $\overline{\text{EN\_EXT\_SEL}}$ pin determines if the external pins or internal bits are used to select the Design Revision. When $\overline{\text{EN\_EXT\_SEL}}$ is Low, Design Revision selection is controlled by the external revision select pins, REV_SEL[1:0]. When $\overline{\text{EN\_EXT\_SEL}}$ is High, Design Revision selection is controlled by the internal programmable revision select control bits.

Note: For the reference design, $\overline{\text{EN\_EXT\_SEL}}$ must be set Low.

During power-up, the Design Revision selection inputs (pins or control bits) are sampled by logic internal to the configuration PROM. After power-up is complete, when $\overline{\text{CE}}$ is asserted Low (enabling the PROM inputs), the Design Revision selection inputs are sampled again after the rising edge of the CF pulse. In the reference design, CE is tied to ground, and the user will control CF using an I/O on the FPGA, see "Reference Design," page 5. The data from the selected Design Revision is then presented on the FPGA configuration interface. The interface can be either 8-bit SelectMAP (parallel) or serial. Refer to DS123, *Platform Flash In-System Programmable Configuration PROMs*, for more details.

Typical Application

To take advantage of Multiple-Boot capabilities with Platform Flash PROMs, a few modifications to the standard PROM interface are needed, as well as a small amount of control logic embedded in the FPGA (Figure 3). Most of the standard connections to the configuration PROM remain the same with a few exceptions:

- CE must be tied LOW to ensure that the PROM is always enabled. Normally, CE is connected to DONE to disable the PROM after configuration is complete.
- CF is driven by an output from the Multiple-Boot control logic inside the FPGA instead of the normal connection to PROG_B.
- The PROG_B pin of the FPGA is driven by an output from the Multiple-Boot control logic.

  Note: This connection requires the use of one user I/O of the FPGA.
- REV_SEL and OE/RESET are driven by outputs from the Multiple-Boot control logic.

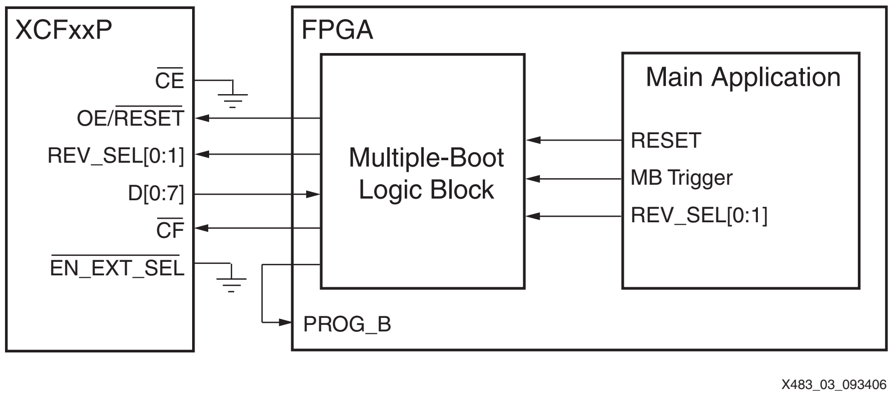


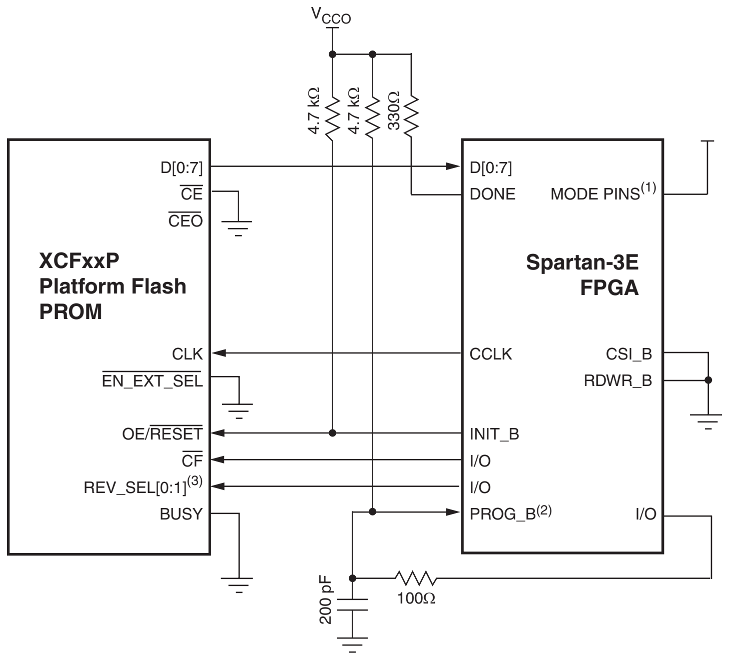
Figure 3: **Typical Application Block Diagram**

The control logic monitors the data pin D[0:7] and RESET, which resets the control logic state machine and Multiple-Boot trigger (MB Trigger in Figure 3). The timing between each state is critical; there are several setup and hold times that need to be closely observed for the reconfiguration to occur successfully. A fully verified control logic design, compliant to these timing constraints is provided (see "Reference Design," page 5).

Hardware Interface

To enable Multiple-Boot functionality, an output on the FPGA controls the sampling of the revision select pins on the PROM. This signal connects to the CF input on the PROM. When the PROM sees a rising edge on CF, it samples the revision select pins. The revision select pins, REV_SEL[1:0], are driven by other output(s) from the FPGA to control which revision provides configuration data to the FPGA. The revision select pins must be set at least 300 ns before the sampling is triggered, (refer to *Platform Flash In-System Programmable Configuration PROMs*). After CF is taken High, the Multiple-Boot logic block inside the FPGA drives an output connected to PROG_B Low for 300 ns to trigger reconfiguration. After configuration is complete, the FPGA functions according to the configuration read from the selected revision.

The configuration signals used by the FPGA to interface with the Platform Flash PROM for SelectMAP mode are described in Figure 4, page 4 and Table 1, page 4.



Notes:
1. For Mode pin connections, refer to the appropriate FPGA data sheet. For Spartan-3E and newer FPGA families.
2. Open-drain driver recommended.
3. REV_SEL[0:1] have an internal 50 KΩ resistive pull-up to V_{CCO}.
4. RC circuit added to ensure sufficient pulse of PROG_B.

X483_04_032407

Figure 4: **SelectMAP Mode Configured from Platform Flash XCFxxP PROM Using Multiple-Boot**

Table 1: **FPGA to Platform Flash XCFxxP PROM Connections for Multiple-Boot**

FPGA Pin	Dir.	PROM Pin	Functionality	
			FPGA	PROM
N/A		$\overline{\text{CE}}$	Not connected.	Chip Enable tied to GND to enable PROM after configuration.
N/A		$\overline{\text{CEO}}$	Not connected.	Chip Enable Output connected to the CE input of the next PROM in the chain.
DONE			FPGA Configuration Done. Recommend 330Ω pull-up.	
INIT_B	→	OE/$\overline{\text{RESET}}$	Initialization Indicator. Recommended 4.7 kΩ pull-up.	Resets PROM and enables PROM to output data.
User I/O	→	REV_SEL[0]	Used to choose revision 0 or 1.	Revision selection pin 0.
User I/O	→	REV_SEL[1]	Optionally used to enable selection of revision 2 or 3.	Revision selection pin 1. If using more than two revisions, then drive from the reference design via an FPGA user I/O. Otherwise, tie to GND to save FPGA user I/O for other purposes.
D[0:7]/DIN	←	D[0:7]/D0	Configuration Data Input.	Configuration Data Output.
User I/O	→	$\overline{\text{CF}}$	Used to control when PROM samples REV_SEL pins.	On rising edge the REV_SEL pins are sampled.

Table 1: **FPGA to Platform Flash XCFxxP PROM Connections for Multiple-Boot *(Continued)***

FPGA Pin	Dir.	PROM Pin	Functionality	
			FPGA	PROM
PROG_B		–	Hold Low for 300 ns to trigger reconfiguration.	Not connected.
CCLK	→	CLK	Configuration Clock.	Configuration Clock.
CSI_B	←	–	Chip select – must be Low during configuration.	Not connected.
RDWR_B	←	–	Read/Write control – must be low during configuration.	Not connected.
M[2:0]	←	–	Refer to appropriate FPGA data sheet for settings.	Not connected.
BUSY		BUSY	Busy indication not used.	Do not connect to FPGA BUSY. Must be tied Low.
N/A		$\overline{\text{EN\_EXT\_SEL}}$	Not connected.	Set Low to allow revision selection by REV_SEL pins.

Reference Design

The reference design described in this application note implements the control logic necessary to load Multiple-Boot configuration data from revisions stored in Platform Flash PROMs. The control logic, to be implemented internal to the FPGA, consists of a state machine designed to select the Design Revisions and to dynamically reconfigure the FPGA. The Multiple-Boot control logic is designed as a separate module to facilitate integration into the design. To use the module, the user must make the appropriate connections and insert logic in the main application to set and hold the REV_SEL pins prior to triggering the dynamic reconfiguration process.

The reference design described in this application note can be downloaded from the following link:

http://www.xilinx.com/xlnx/xweb/xil_publications_display.jsp?category=Application+Notes/Device+Configuration+and+Programming/FPGA+Configuration&show=xapp483.pdf

Control Logic State Machine

The control logic state machine consists of seven states:

State 0

The FPGA will enter this state upon power-up after an initialization state (not shown) to setup/reset any variables or signals used in the design. The machine waits in state 0 for a dynamic reconfiguration trigger from the main FPGA design to enter State 1.

State 1

The control logic drives a Low signal to the $\overline{\text{CF}}$ pin on the PROM and passes the revision set by the main application to the PROM's REV_SEL[0:1] pins. The machine enters State 5 after the specified set-up time for the REV_SEL (300 ns).

Note: For designs utilizing the XCF16P PROM or requiring only two revision locations, the Rev_sel[1] input of the Multiple-Boot control logic block must be grounded, and REV_SEL[1] of the PROM must be tied to ground to ensure proper FPGA configuration.

State 2

The control logic drives a HIGH signal to $\overline{\text{CF}}$ and OE/$\overline{\text{RESET}}$ on the PROM. The resulting rising edge on $\overline{\text{CF}}$ signals the PROM to sample the REV_SEL pins. Taking OE/$\overline{\text{RESET}}$ High signals the PROM to place the first bit or byte (only) of that revision onto the data bus. After 300 ns, the machine transition State 3.

> ***Note:*** When $\overline{\text{CF}}$ is set HIGH, it takes 300 ns for the data to become valid on D0/D[0:7]. In addition, the control logic drives OE/$\overline{\text{RESET}}$ Low during normal operation to eliminate multiple sampling of the REV_SEL pins.

State 3

In state 3, the control logic checks the data on the data bus (D[0:7]) for the user mark in the PROM file (see "Adding the User Mark," page 7). If the mark is present, then the control logic assumes the bitstream is valid and continues to State 4, if it is not present, then the machine enters an idle state (State 6).

State 4

The control logic assumes that the data in the revision is valid and proceeds with reconfiguration. The PROM is set up again just as it was in state 2, to get it ready to send configuration data. The machine enters State 5 after the specified set-up time for the REV_SEL (300 ns).

State 5

Once in state 5, the state machine drives a HIGH signal to the $\overline{\text{CF}}$ input of the PROM and a LOW signal to the PROG_B pin on the FPGA. After 300 ns, reconfiguration begins, and the FPGA starts re-initializing configuration memory. Once INIT_B (on the FPGA which is connected to OE/$\overline{\text{RESET}}$ on the PROM) goes HIGH, the FPGA is ready for data. The PROM sends the configuration data stored at the selected revision to FPGA. The FPGA configuration logic signals that the reconfiguration process is complete by taking DONE High. The machine then returns to State 0.

Note: This application note assumes that each revision contains an instantiation of the Multiple-Boot control logic. In reality, after PROG_B goes Low in state 2 (after 300 ns) the configuration memory is erased, including the state machine. Therefore, the state machine does not actually transition from state 2 back to state 0, but rather ceases operation while in state 2. After the FPGA is reconfigured and initialized, the Multiple-Boot control logic instantiation in the new configuration enters state 0.

State 6

This is an idle state entered into if valid data is not stored at the desired revision location. The control logic waits in this state until it receives a RESET signal. Once a RESET is received, all signals and variables are reset, and the machine enters State 0 to await a Multiple-Boot trigger to attempt a reconfiguration.

Figure 5 illustrates a state diagram for the state machine used in the reference design.

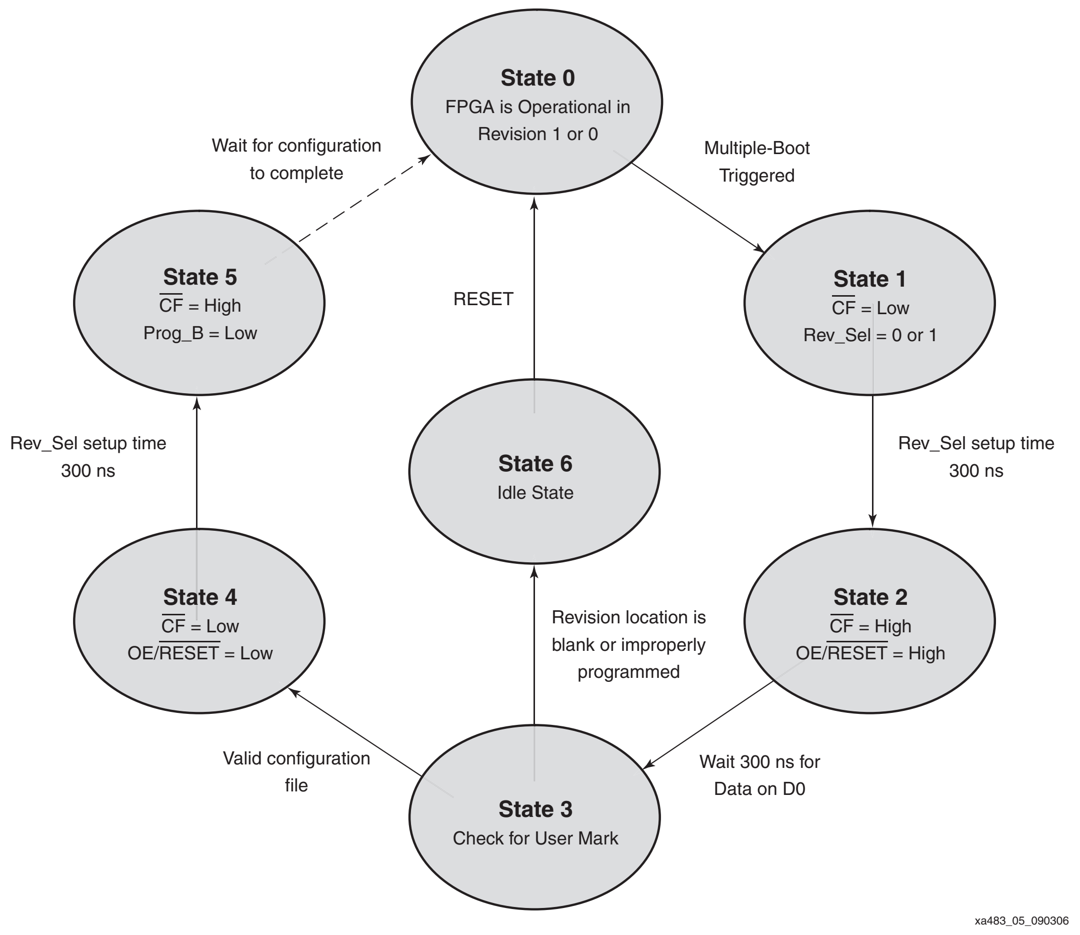


Figure 5: **Reference Design State Diagram**

Added Safety

The Multiple-Boot control logic described in this application note has the added ability to detect a blank or incompletely programmed PROM. This added safety feature exploits the ISC_DONE behavior of the Platform Flash XCFxxP PROM. IEEE Standard 1532 for in-system configuration specifies that the PROM behave like a blank device until the device program has been successfully written to the satisfaction of an external algorithm. In other words, the XCFxxP PROM cannot output its programmed data contents until a revision location has been completely programmed. In all other cases of a blank or incompletely programmed revision location, the XCFxxP PROM outputs only 1s from its data output pin(s) reflecting a blank state of the revision location. The reference design takes advantage of the ISC_DONE behavior to determine whether a bitstream has been completely programmed into the selected revision location before FPGA reconfiguration from that revision location is initiated.

Adding the User Mark

In order to take advantage of this added safety feature, the user must "mark" the configuration files to be used. This user mark is then read in State 3 to verify that a valid configuration file is located at that revision, prior to triggering reconfiguration. To mark the configuration files, the user must change the first byte in the PROM file to be read by the FPGA, from `FF` to `99` in the second line of the file using the XMCSUTIL provided with this application note (Figure 7, page 8). In addition, the CRC for that line must also be adjusted.

The new CRC is calculated by taking the 2's complement of the sum of the 8-bit numbers on each line in the PROM file. The utility recalculates the CRC and makes the necessary adjustments in the file. To invoke the file:

```
XMCSUTIL -d -i filename.mcs -o filename.mcs -23 0x00000 0x99 0x200000 0x99
```

The utility overwrites the existing PROM file (`.mcs`) with the -d option in order to maintain compatibility with the `.cfi` and `.prm` files generated with the `.mcs` file.

The arguments at the end specify which byte to set and what to set it to. For this application, the user should change the first byte in each revision to `0x99` (the state machine looks for this value on the data bus to determine if a valid revision is at that location). The address of the first byte in the revision can be found at the end of the `.prm` file (a file generated with the `.mcs` file).

Figure 6 shows the start address of each revision. Underneath Addr1, the start address is displayed in hexadecimal for each revision in the PROM file, with revision 0 start address of `0x000000` and revision 1 start address of `0x200000`. When passing the address to the utility, use the format shown above: `0x000000` (`0x` followed by the six hexidecimal digit address).

```
Addr1         Addr2          Date File(s)
0000:0000     0017:f32f      Apr 25 14:17:10 2006 C:/Data/Demo/module_schem_0to1.bit
0020:0000     0037:f32f      Mar 23 14:18:17 2006 C:/Data/Demo/module_schem_1to0.bit
```

Figure 6: **Example `.prm` File**

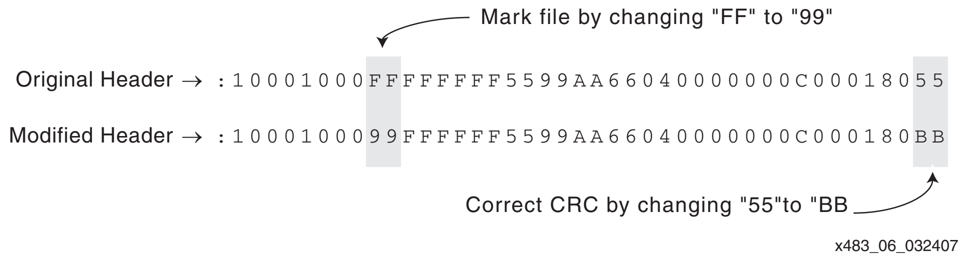

Mark file by changing "FF" to "99"
Original Header → :10001000FFFFFFFF5599AA660400000000C00018055
Modified Header → :1000100099FFFFFF5599AA660400000000C000180BB
Correct CRC by changing "55"to "BB
x483_06_032407

Figure 7: **Marking the Configuration File Header**

Checking for the User Mark

In State 3, the control logic begins checking the data by setting the revision select pins and sending a rising edge to $\overline{\text{CF}}$ on the PROM. Next, OE/$\overline{\text{RESET}}$ on the PROM is driven High via INIT_B. Taking OE/$\overline{\text{RESET}}$ High causes the PROM to load the first bit or byte (depending on configuration mode) of the configuration file stored at that revision location onto the data bus (DIN/D[0:7]). After a 300 ns set-up time has elapsed, the control logic checks the bits at the user mark location. If `99` is present then the PROM file is in fact valid. If anything else is present, then that revision location is either blank or incompletely programmed. The user application must then decide what action to take.

Note: The control logic is also tied to the INIT_B signal of the FPGA. Taking the INIT_B pin of the FPGA High does not affect its functionality.

Enhancements

The reference design has the $\overline{\text{CE}}$ pin tied to ground so the PROM is always enabled, causing the PROM to draw a small amount of current even when it is not in use. To save power, the user can add logic to the state machine to disable the PROM when not in use (via the $\overline{\text{CE}}$ signal). However, if the user drives the $\overline{\text{CE}}$ pin High as a part of the normal application, the PROM enters a low-power stand-by mode. The state machine then must drive the $\overline{\text{CE}}$ pin Low to enable the PROM whenever reconfiguration is required.

More Safety

The reference design for the Multiple-Boot control logic included with this application note only checks for the user mark in the configuration file before triggering reconfiguration. A more sophisticated verification scheme is possible; however, this adds a significant degree of complexity to the design. For example, during State 3, data could be clocked out of the PROM until the control logic verifies that the sync-word is present inside that specific revision. There are other potential obstacles that can be encountered with such a verification scheme. For most applications, checking for the user mark should prove sufficient.

Note: While all of the added safety measures described here greatly reduced the chance of a failed configuration, this solution is not fail-safe. An additional external device is required to implement a truly fail-safe solution. Refer to XAPP693, *A CPLD-Based Configuration and Revision Manager for Xilinx Platform Flash PROMs and FPGAs*, for more details.

Advantages

The main advantages of the Multiple-Boot solution described here are:

- No third device required for reconfiguration. This solution only requires a Platform Flash XCFxxP PROM and a FPGA while most reconfiguration designs require an extra logic device such as a CPLD.
- The ability to dynamically reconfigure the FPGA. As long as the functionality stored at each revision location is not required simultaneously, the user can take advantage of up to four completely different designs for the same FPGA, effectively timesharing the FPGA resources.

In addition, Multiple-Boot gives the user access to the advantages of using Platform Flash PROMs:

- Integrated iMPACT programming support. In-System Programmability makes design changes easy during development and verification.
- Faster configuration. Platform Flash PROMs have been optimized for fast configuration using both the x8 interface and a configuration clock period of 30 ns.
- Lower board cost due to the reduced number of interface lines needed. Since only the data lines need to be used, the number of signals to be routed on the board is minimized compared to parallel NOR flash.
- Single vendor support. Customers can maintain a single vendor solution with Xilinx Platform Flash and Xilinx FPGAs.

For more on the benefits of Platform Flash PROMs visit:

http://www.xilinx.com/products/silicon_solutions/proms/pfp/index.htm

Conclusion

Xilinx Platform Flash XCFxxP PROMs enable users to implement Multiple-Boot capabilities in their designs. Essential to Multiple-Boot support is the Design Revisioning capabilities of Platform Flash PROMs, allowing the storage of multiple FPGA configurations in as little as a single PROM (supported for all Xilinx FPGA families). When the Design Revision capabilities of Platform Flash PROMs are paired with dynamic reconfiguration, the powerful result is Multiple-Boot.

A Multiple-Boot solution with Platform Flash PROMs is easy to implement, reduces board cost, has iMPACT programming support, and allows for rapid FPGA configuration.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
08/02/06	1.0	Initial Xilinx release.
04/11/07	2.0	Added the section "Added Safety" to help users prevent failed configuration.