XAPP719 (v1.1) March 13, 2006

# PowerPC Cache Configuration Using the USR_ACCESS_VIRTEX4 Register

Author: Nick Camilleri and Peter Ryser

## Summary

The Virtex™-4 user access register (USR_ACCESS_VIRTEX4) is a 32-bit register that provides direct access to bitstream data by the FPGA fabric. It is useful for loading PowerPC™ 405 (PPC405) processor caches and/or other data into the FPGA after the FPGA has been configured, thus achieving partial reconfiguration. The USR_ACCESS_VIRTEX4 register is programmed through the bitstream with a command that writes a series of 32-bit words.

This application note shows the steps for configuring the processor caches, and uses the example Verilog design provided in http://www.xilinx.com/bvdocs/appnotes/xapp719.zip. The reference design provides a simple user interface for any user design containing a PowerPC processor.

## Introduction

PPC405 processor caches are not directly programmable because FPGA configuration bitstreams do not contain the data that is stored in these caches. While it is possible to accomplish post-configuration cache-loading with special user logic, this logic occupies some of the FPGA fabric. Where conservation of logic resources is desired, the USR_ACCESS_VIRTEX4 primitive in Virtex-4 FX FPGAs provides an alternative solution. It facilitates the loading of the processor caches, thus providing a means for users to access bitstream data through an external storage device, such as a PROM or CompactFlash card, without disturbing the original configuration. Figure 1 shows the relative location of the USR_ACCESS_VIRTEX4 register within the user access site (USR_ACCESS_SITE).

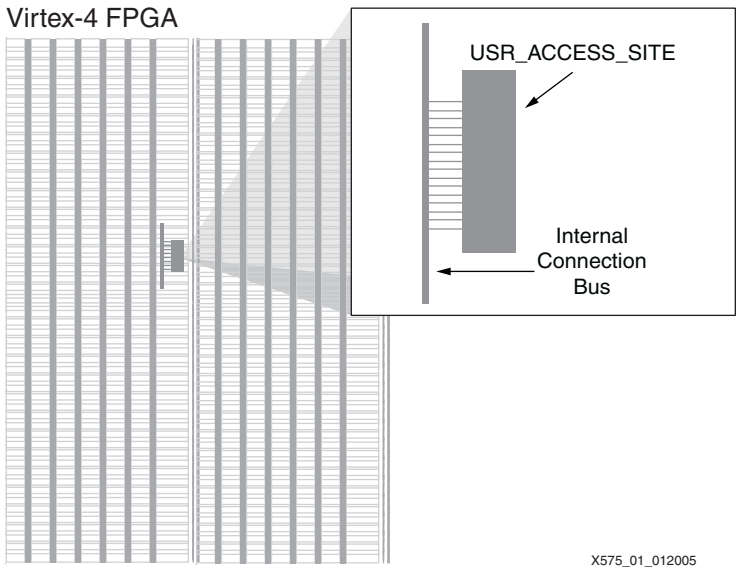



*Figure 1:* **Relative Location of USR_ACCESS_SITE**

## Required Hardware/Tools

- Xilinx ML403 development board
- Xilinx ISE 7.1i (Service Pack 4) or 8.1i (Service Pack 2)
- Xilinx Platform Studio 7.1i or 8.1i

# Instantiation

Figure 2 is a block diagram that shows how to connect the relevant modules to create a simple loadable-cache design that operates in master-serial mode.

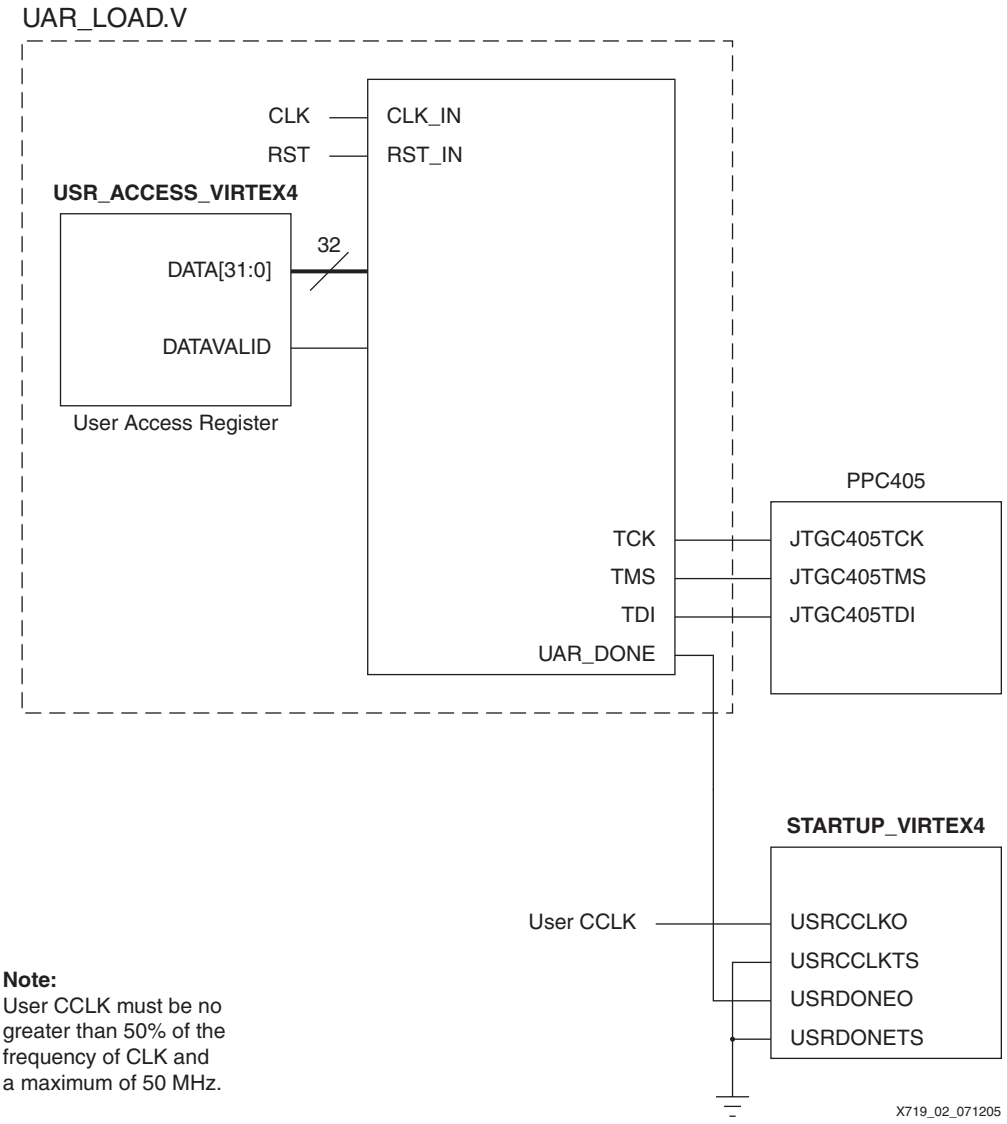



*Figure 2:* **Simple Cache-Loading Block Diagram**

As the cache data is loaded into the FPGA through the USR_ACCESS_VIRTEX4 register, a special core in the user design, uar_load.v, intercepts this data and converts it into the JTAG signals TCK, TMS, and TDI. These signals drive JTAG inputs on the PPC405 in the main user design. Using this method, the cache can be loaded with data contained in the bitstream.

An optional output signal, UAR_DONE, is generated within the uar_load.v module. It goes High after the cache loading has been completed.

The STARTUP_VIRTEX4 module is used to drive the CCLK and DONE pins externally through the FPGA fabric. By driving the USRCCLKO and USRDONEO inputs, the user can control

CCLK and DONE after configuration. This may be necessary, for example, when using some Xilinx development platforms, because the DONE pin is connected to the PROM reset. This means that if DONE goes High after the part is configured (but before the cache is loaded), the cache data in the PROM will never be seen. To avoid this, the DONE signal must be held Low until the cache is loaded by connecting the STARTUP_VIRTEX4 signals. The 3-state output enables for each of these signals (USRCCLKTS and USRDONETS, respectively) should be driven Low to activate the signals.

***Note:*** User CCLK and USRCCLKTS should not be connected when programming in slave-serial or slave SelectMAP modes.

## Design Flow

The ISE design flow used in this application note is shown in Figure 3. First, the main design containing the PPC405 is compiled, using XST or another synthesis tool. The design is run through **ngdbuild**, using the supplied user constraints file (UCF). Map, place, and route are run to create the routed NCD file. This NCD file, along with an executable BIN file or ELF file, can be converted into a bitstream using the PERL script, genmcs.pl, located in the xapp719.zip file (see "Using the GENMCS PERL Script," page 6 for details). This PERL script reads the NCD file (design.ncd), and a binary file (executable.bin) or ELF file (executable.elf) that contains the C code instructions for the processor to execute. After the binary data has been processed, the script creates an output rawbits file (output.rbt) and an output MCS file (output.mcs) that contain the original design and the cache data concatenated within the bitstream. The MCS file can be used to program a serial PROM, which configures the FPGA. The output format can also be specified as EXO.

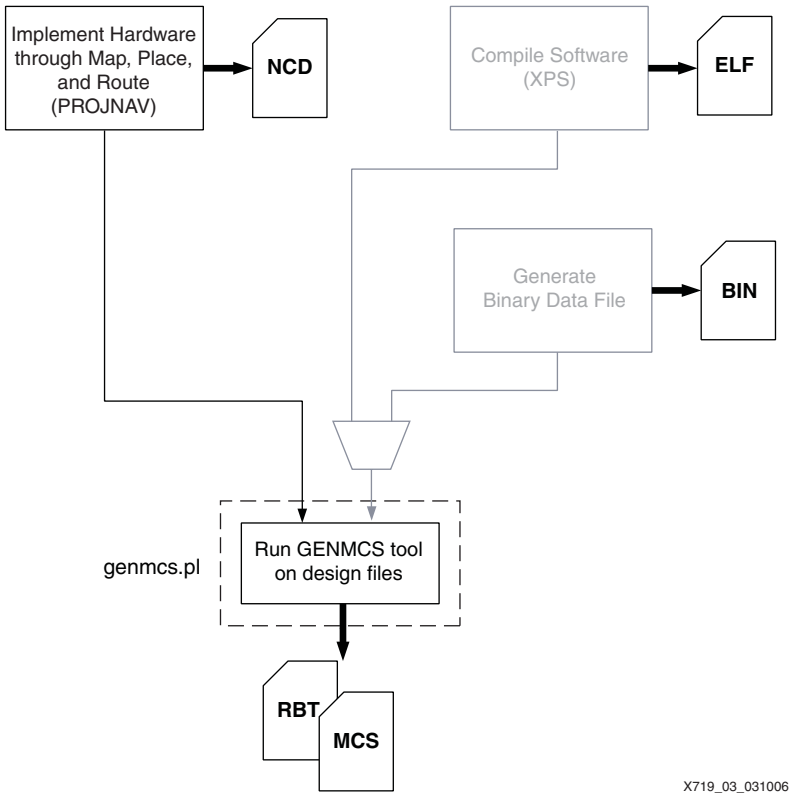



*Figure 3:* **Design Flow**

# Quick Start

The Quick Start is a step-by-step procedure for manually generating the RBT and MCS output files. Table 7, page 7 shows the pre-generated output files that can be used for testing the completed design, if the user wants to bypass the Quick Start.

This application note uses the Simon example application from the UltraController-II design. (See www.xilinx.com/ultracontroller.) The Quick Start includes steps to compile the design and create a bitstream that programs the FPGA in master-serial mode (where the PROM is a slave). If using iMPACT to download the RBT or MCS files, the startup clock setting in the Preferences dialog box must be changed from Automatic Correction to Ignore Setting.

## Generating the PPC405 Design

1. Instantiate the uar_load.v module, and connect the signals as shown in Table 1.

*Table 1:* **UAR_LOAD.V Signal Connectivity**

| Signal | Direction | Connectivity |
|---|---|---|
| clk | Input | Connect to a system clock (ML403 sys_clk_in = 100 MHz) |
| rst | Input | Connect to an active-High system reset, or connect to ground to disable |
| tck | Output | Connect to the JTGC405TCK input of the PPC405 |
| tdi | Output | Connect to the JTGC405TDI input of the PPC405 |
| tms | Output | Connect to the JTGC405TMS input of the PPC405 |
| uar_data_out[31:0] | Output | No connect - extra tap if needed |
| uar_datavalid | Output | No connect - extra tap if needed |
| uar_done | Output | (Optional) Connect to USRDONEO input of STARTUP_VIRTEX4 block |

2. (Optional): Instantiate the STARTUP_VIRTEX4 module.

   If desiring user control of external CCLK and DONE signals, instantiate the STARTUP_VIRTEX4 module in the design, which has the ability to control the CCLK and DONE signals externally through the USRCCLKO and USRDONEO inputs. Connect the STARTUP_VITEX4 signals as shown in Table 2. If programming from a PROM, the USRDONEO and USRDONETS outputs are required, as the PROM reset is connected to the DONE pin on the ML403 board.

*Table 2:* **STARTUP_VIRTEX4 Signal Connectivity for Optional User Control**

| Signal | Connectivity |
|---|---|
| USRCCLKO | Connect to a system/configuration clock at least 50% slower than CLK |
| USRCCLKTS | Connect to ground to permanently enable the output (High = high-impedance state) |
| USRDONEO(1) | Connect to the uar_done output of the uar_load.v module |
| USRDONETS(1) | Connect to ground to permanently enable the output (High = high-impedance state) |

**Notes:**

1. Do not connect USRDONEO and USRDONETS if using slave programming modes.

2. Generate desired PPC405 instruction code (out.elf) or binary file (out.bin).
3. Run genmcs.pl on the routed NCD file to create the final MCS and RBT files.

```
% xilperl genmcs.pl design.ncd out.bin output.mcs
```

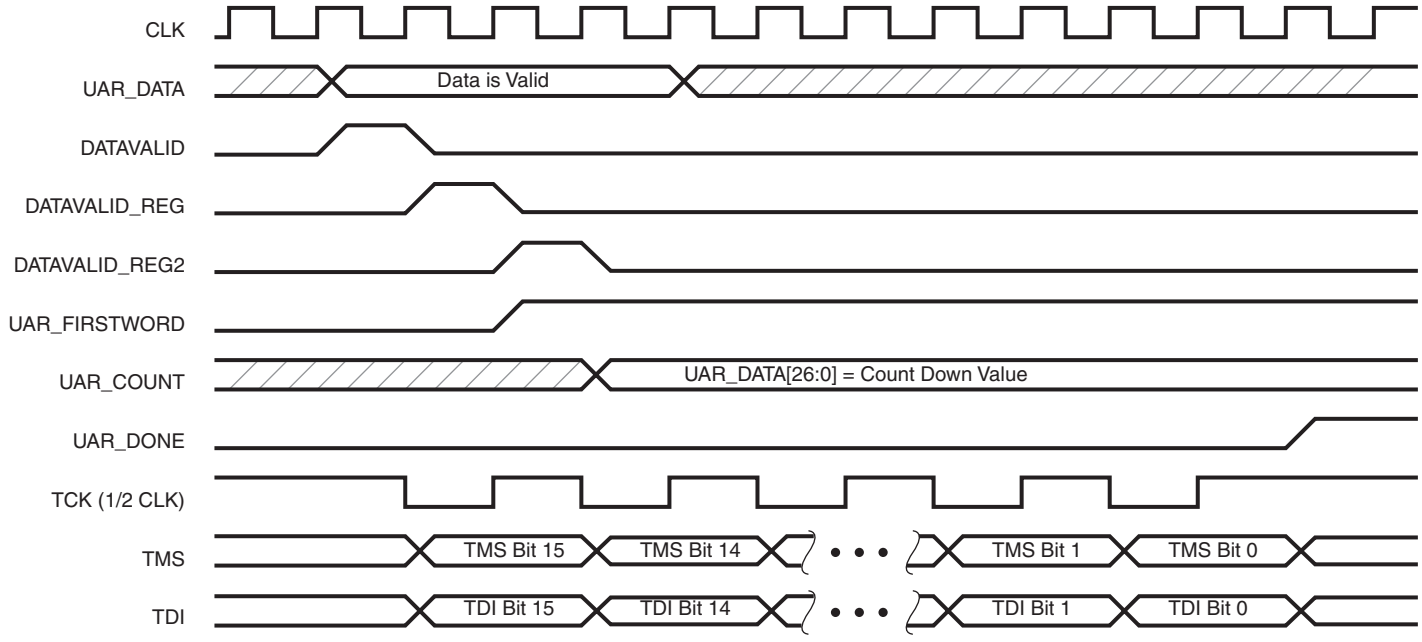
This command generates two files:

- The output.mcs is used if programming a PROM to load the bitstream
- The output.rbt file is used if programming by way of a JTAG interface

## Configuring the FPGA

Using the USRCCLKO input on the STARTUP_VIRTEX4 module allows the cache to be configured externally with a user clock. The cache is configured at a rate of one bit per CLK cycle. If using iMPACT to download the RBT or MCS files, the startup clock setting in the Preferences dialog box must be changed from Automatic Correction to Ignore Setting.

1. Download the bitstream to the PROM.
2. Configure the FPGA using master-serial mode. When the FPGA is operational, internal JTAG TCK, TMS, and TDI signals transmit the data to the cache. Figure 4 shows an example of the signal transmission.




**Notes:**

1) UAR_COUNT is only loaded after the first word written into the USR_ACCESS_VIRTEX4 register.

2) UAR_DONE goes high only when UAR_COUNT is 0, and then the last word is written to the JTAG port.

3) TMS and TDI bit order is different than the USR_ACCESS_VIRTEX4 bit order. Due to byte order and little endian data, TMS data appears in the following order:
UAR_DATA bit 7, 5, 3, 1, 15, 13, 11, 9, 23, 21, 19, 17, 31, 29, 27, 25.
Similarly, TDI appears as UAR_DATA bit 6, 4, 2, 0, 14, 12, 10, 8, 22, 20, 18, 16, 30, 28, 26, 24.

X719_04_081005

*Figure 4:* **Data Transfer Timing Diagram**

## Using the GENMCS PERL Script

A PERL script, genmcs.pl v1.03, is included in the xapp719.zip file and is used for generating RBT and MCS files for loading a bitstream into the PPC405 caches. It uses a companion PERL script called svf2bin.pl, to help convert an ELF file to a BIN file if necessary. Table 3 shows the GENMCS files.

*Table 3:* **GENMCS Files**

| File | Description |
|---|---|
| design.ncd | Input design NCD file containing the physical design description in the target device |
| data.<bin,elf> | Input binary data file containing executable code (ELF input accepted on PC only) |
| output.mcs | Output MCS file for programming the PROM (default = “genmcs.mcs”) |
| output.rbt | Output RBT file for programming via JTAG (default = “genmcs.rbt”) |

Table 4 shows the options for using GENMCS.

*Table 4:* **GENMCS Options**

| Option | Description |
|---|---|
| **-a** | Specifies the architecture type (default = virtex4) |
| **-p** | Specifies the part type in the bitstream (default = 4vfx12ff668) |
| **-f** | Specifies the output PROM format (default = mcs) |
| **-b** | Skip bitgen compilation (when RBT and BIT files are already present) |

The following is an example of how to use the genmcs.pl script:

```
xilperl genmcs.pl [options] <design>[.ncd] <data.[bin,elf]> [output][.mcs]
```

The script processes the design.ncd and the data.bin (or data.elf) files that contain the code for the PPC405. Running the following command generates a default bitstream and MCS file:

```
% xilperl genmcs.pl design.ncd data.bin output.mcs
```

This command generates two files:

- The output.mcs is used if programming a PROM to load the bitstream
- The output.rbt file is used if programming the by way of a JTAG interface

The part type is read from the design but can be overridden by running the **-p** option, for example:

```
% xilperl genmcs.pl -p 4vfx12ff668 design.ncd data.bin output.mcs
```

The output format can be set to anything that PROMGen can create. For example, the following command produces a PROM file in the EXO format instead of MCS:

```
% xilperl genmcs.pl -f exo design.ncd data.bin output.exo
```

## Resource Utilization

Table 5 shows the resource utilization for the reference design implemented in an XC4VFX12 device.

*Table 5:* **Utilization in an XC4VFX12 Device**

| Resource | Usage | Percent Used |
|---|---|---|
| Slice flip-flops | 110 | 1.00% |
| LUTs (4-input) | 169 | 1.54% |
| Occupied slices | 120 | 2.19% |
| USR_ACCESS_VIRTEX4 | 1 | 100% |
| STARTUP_VIRTEX4 | 1 (optional) | 0-100% |

## Reference Design

The reference design files are located at http://www.xilinx.com/bvdocs/appnotes/xapp719.zip. The example design was developed and tested in a Linux environment on the Xilinx ML403 embedded system development platform. The design can be compiled on a Windows platform, although this is not covered in this application note.

Table 6 lists the files required for a user design.

*Table 6:* **Required Files for a User Design**

| File | Description |
|---|---|
| genmcs.pl | PERL script to create cache-loading bitstreams |
| svf2bin.pl | Companion PERL script that is called by genmcs.pl that converts an SVF file to a BIN file |
| uar_load.v | Design to convert USR_ACCESS data into TCK/TMS/TDI signals which drive the PPC405 |

Table 7 lists the directory and files for the example design.

*Table 7:* **Example Design Directory and Files**

| File | Description |
|---|---|
| debughalt_0_wrapper.ngc<br>ff_0_wrapper.ngc<br>ppc405_0_wrapper.ngc<br>vl_0_wrapper.ngc | NGC files required when compiling the Verilog code using XST in the ISE flow for the Simon application |
| runfile | Sample batch file for Linux/Solaris |
| simon.bin | Software (binary) for Simon design |
| simon.scr | XST script file for design synthesis |
| simon.ucf | UCF file for using with ML403 board |
| simon.v (1) | Top-level Verilog for Simon design |
| simon_routed.mcs | MCS file of routed Simon design (PROM download); includes the cache data |
| simon_routed.ncd (2) | NCD file of routed Simon design |
| simon_routed.rbt (2) | RBT file of routed Simon design (JTAG download); includes the cache data |
| system_v4.v | Simple PPC405 design used by simon.v |
| xstfiles.v | List of XST Verilog files for use with XST script |
| webserver.bin (3) | Binary file for web server design |

*Table 7:* **Example Design Directory and Files *(Continued)***

| File | Description |
|---|---|
| webserver.ncd (3) | Sample web server design |
| webserver.elf (3) | ELF file for web server design |
| webserver.rbt (3) | RBT file for routed web server design; includes the cache data |
| webserver.mcs (3) | MCS file for routed web server design; includes the cache data |

**Notes:**

1. For details, see XAPP575, UltraController II: Minimal Footprint Embedded Processing Engine [Ref 1].
2. The simon_routed MCS and RBT files comprise concatenated bitstreams with the simon.bin at the end of each file. The software generates a routed NCD file, but the MCS and RBT files actually contain data for both the routed NCD and the BIN/ELF.
3. For details, see XAPP807, *Minimal Footprint Tri-Mode Ethernet MAC Processing Engine* [Ref 2].

## Conclusion

Loading the PPC405 processor cache is simplified by using the user access register available in Virtex-4 FPGAs, along with the design files and scripts associated with this application note.

Other potential uses for this methodology include loading data into external memory through the bitstream, pre-loading a special device ID or a timestamp into the FPGA, and downloading sensitive data in bitstream-encrypted mode.

## References


1. XAPP575, *UltraController II: Minimal Footprint Embedded Processing Engine*
2. XAPP807, *Minimal Footprint Tri-Mode Ethernet MAC Processing Engine*
3. XAPP571, *DEBUGHALT Controller for PowerPC Boot and Reset Operations*
4. UG018, *PowerPC™ 405 Processor Block Reference Guide*
5. UG071, *Virtex-4 Configuration Guide*
6. UG082, *ML40x Reference Design User Guide*


## Revision History

The following table shows the revision history for this document.

| Date | Version | Revision |
|---|---|---|
| 10/04/05 | 1.0 | Initial Xilinx release. |
| 03/13/06 | 1.1 | Updated the genmcs.pl script to no longer require a special DLL (available only on PC) to directly convert ELF files to BIN files. The function has been rewritten to work on Windows, Linux, and Solaris platforms. |