XAPP1232 (v1.0) March 3, 2016

Bitstream Identification with USR_ACCESS using the Vivado Design Suite

Author: Kyle Wilkinson

Summary

This application note describes a solution for bitstream identification that is accessible to the FPGA user design. Instructions are provided for Vivado® Design Suite write_bitstream and for access to the identification information using Vivado tools.

Introduction

Tracking bitstreams is useful for many applications, for example, identifying versions of a particular design or having serial numbers to more complex situations such as tracking design optimization implementation runs. It is possible to embed any form of static code into the design, allowing limitless possibilities for the size or format of the version data. However, this results in the need to recompile parts if not all of the entire design, which is tedious at best and has its limitations. The USR_ACCESS register, present in all 7 series and UltraScale™ FPGAs, provides the ability to embed version information into a 32-bit fabric-accessible register at the bitstream generation phase, allowing you the best balance of flexibility with minimal impact to the design and implementation time.

USR_ACCESS Primitive

The FPGA configuration logic includes a readable and writable 32-bit register called USR_ACCESS, that is also directly accessible from the user design. The primitive names for the USR_ACCESS registers are USR_ACCESSE2 in both 7 series and UltraScale devices (see Figure 1). All further references to the primitives in this document are simplified as USR_ACCESS. This component provides direct FPGA logic access to the 32-bit value stored by the FPGA bitstream. For purposes of this document, the USR_ACCESS is considered a static value, although it can be changed through the configuration interface, JTAG, or the internal configuration access port (ICAP). The DATAVALID output port toggles when the USR_ACCESS register is updated from the configuration interface. CFGCLK reflects the configuration clock. This dynamic writing functionality is outside the scope of this application note. More information on writing to this

register dynamically can be found by searching for the register AXSS in the respective FPGA family's configuration user guide:

- *UltraScale Architecture Configuration User Guide* (UG570) [Ref 1]
- *7 Series FPGAs Configuration User Guide* (UG470) [Ref 2].

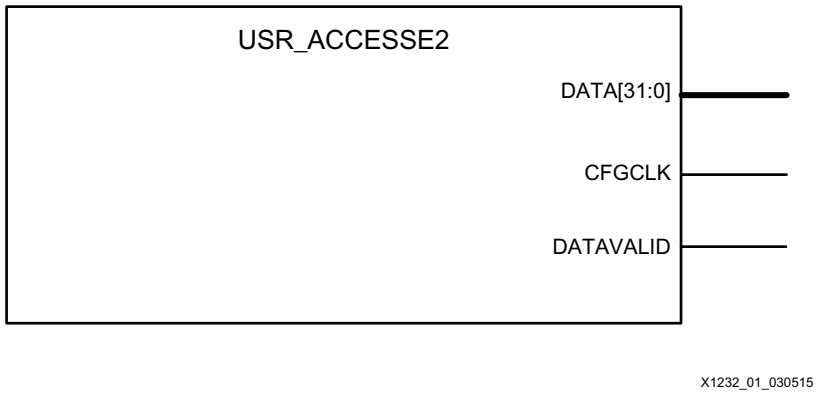

Figure 1: **Schematic of USR_ACCESS Component**

Timestamp

The USR_ACCESS register can be configured with a 32-bit user-specified value or automatically loaded by the bitstream generation command (write_bitstream) with a timestamp. The user-specified value can be used for revision, design tracking, or serial number type applications. The timestamp feature is useful when several implementation runs have been performed, thereby changing design optimization values, but the source design itself is unchanged. The timestamp value can then be compared to the timestamp for the bitstream file to correlate the design in the device to one of the many possible sources. The timestamp feature is not easily implemented by changing the source code. Implementing the USR_ACCESS method provides a more accurate timestamp.

Write_Bitstream Property for USR_ACCESS

The command usage feature is implemented using the following Tcl command during the Generate Bitstream process (write_bitstream).

```
set_property BITSTREAM.CONFIG.USR_ACCESS NONE|0x<8-digit hex>|TIMESTAMP
[current_design]
```

When no value or NONE is entered for this option, the behavior is to do nothing to this register, which defaults to all 0s:

```
NONE - DEFAULT
```

When an 8-character hexadecimal value is detected, this value is entered into the USR_ACCESS register:

```
0xXXXXXXXX
```

When the keyword TIMESTAMP is entered as the value:

```
TIMESTAMP
```

write_bitstream inserts the current timestamp into the 32-bit USR_ACCESS register in this format:

```
ddddd_MMMM_yyyyyy_hhhhh_mmmmmm_ssssss
(bit 31) ..... (bit 0)
```

Where:

`ddddd` = 5 bits to represent 31 days in a month

`MMMM` = 4 bits to represent 12 months in a year

`yyyyyy` = 6 bits to represent 0 to 63 (to note year 2000 to 2063)

`hhhhh` = 5 bits to represent 24 hours in a day

`mmmmmm` = 6 bits to represent 60 minutes in an hour

`ssssss` = 6 bits to represent 60 seconds in a minute

When using the TIMESTAMP value, the minute and second values might not correspond directly with the timestamp on the file. This occurs because the timestamp value is determined near the beginning of the bitstream generation process, but the operating system file timestamp is at the end of the file creation process. Therefore, depending on the speed of the machine and the complexity of the operations required for write_bitstream, these values might not match exactly with the file timestamp. Similarly, the same can occur if file generation is started close to the end of the hour or day.

Vivado Tools Flow

There are a couple of different ways to implement the USR_ACCESS feature using the Vivado tools. The recommended way to set the USR_ACCESS register is to add it to the Xilinx Design Constraints (XDC) file before running Synthesis and Implementation, similar to an I/O location (LOC) or IOSTANDARD constraint. Alternatively, this option can also be entered using the GUI, under the **Edit Device Properties** process. To access the options using the Vivado integrated design environment (IDE) or GUI:

1. Open a **Synthesized Design** or **Implemented Design**.
2. Under the **Tools** menu, select **Edit Device Properties.**
3. Select **Configuration**.

4. Enter an 8-digit hex value as shown in Figure 2 in the **User Access** field.
5. Replace the text **abcd0123** with **TIMESTAMP** if the TIMESTAMP feature is desired.

   Note: Not entering this option defaults to NONE, which leaves this register with all 0s.
6. Click **OK**.
7. Click **Generate Bitstream**.
8. Select **Save.** XDC is updated. Rerun **Synthesis and Implementation.**

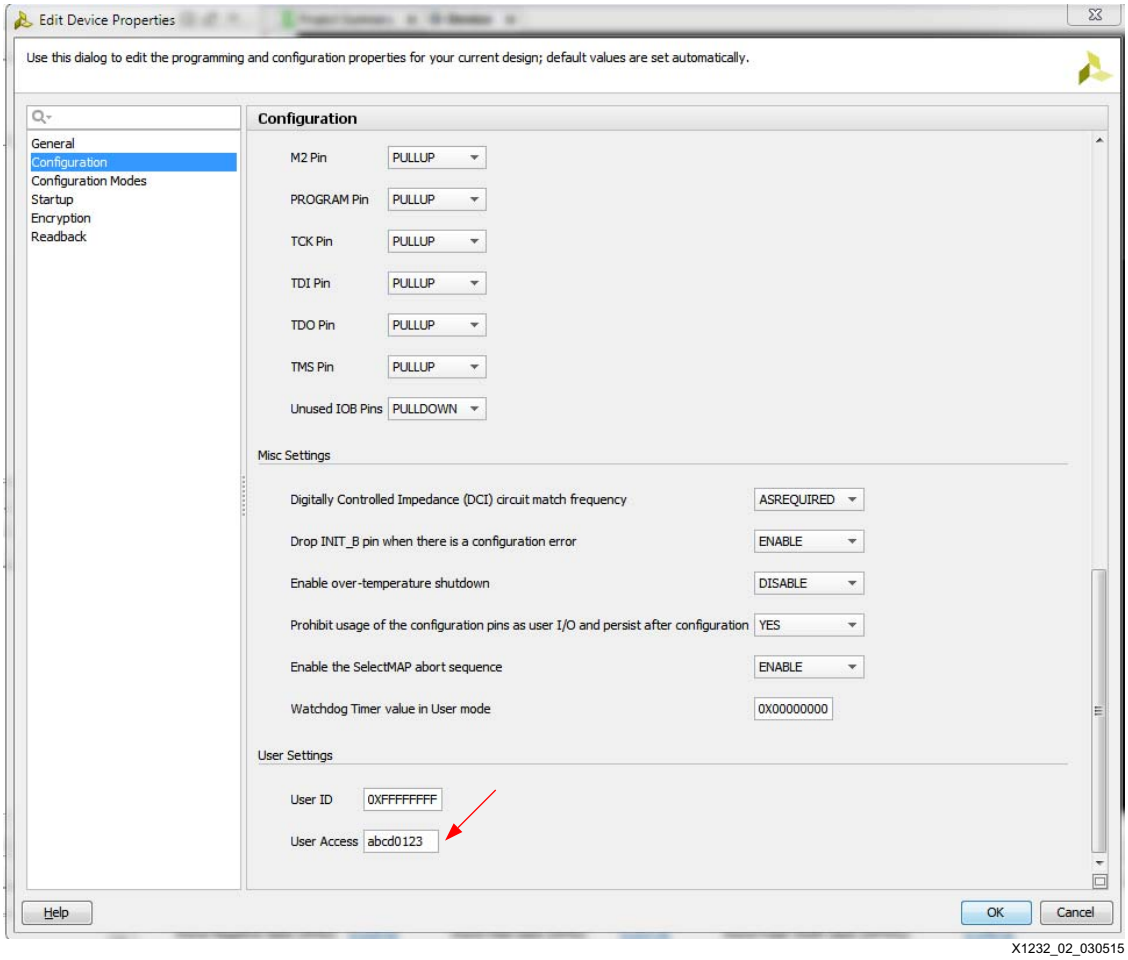

Figure 2: **Edit Device Properties**

To confirm the write_bitstream property is correctly written, open the XDC file and search for the USR_ACCESS property, for example:

```
set_property BITSTREAM.CONFIG.USR_ACCESS abcd0123 [current_design]
##############################################################################
# End
##############################################################################
```

Ability to Read the USR_ACCESS Register with Vivado Tools

The Vivado Hardware Manager feature can read the USR_ACCESS register. The USR_ACCESS register is part of the selected hw_device registers. After connecting to a valid hw_target, the ability to verify the USR_ACCESS register is available. This allows you to confirm the register is indeed written with the expected value or to confirm if the device is configured with the desired BIT file. To verify the USR_ACCESS register is set correctly using the GUI, see Figure 3.

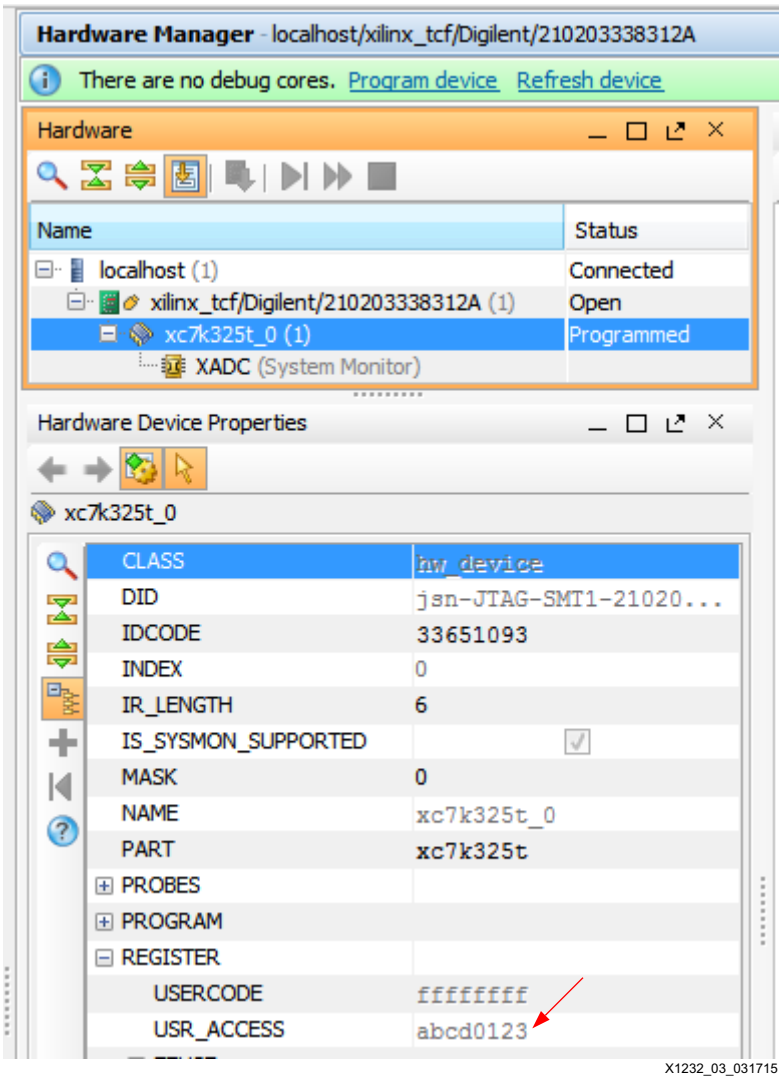

Figure 3: **Hw_target Device Properties**

To verify the USR_ACCESS register is set correctly using Tcl, use the `get_property` of device REGISTER.USR_ACCESS:

```
get_property REGISTER.USR_ACCESS [lindex [get_hw_devices] 0]
```

See Figure 4.

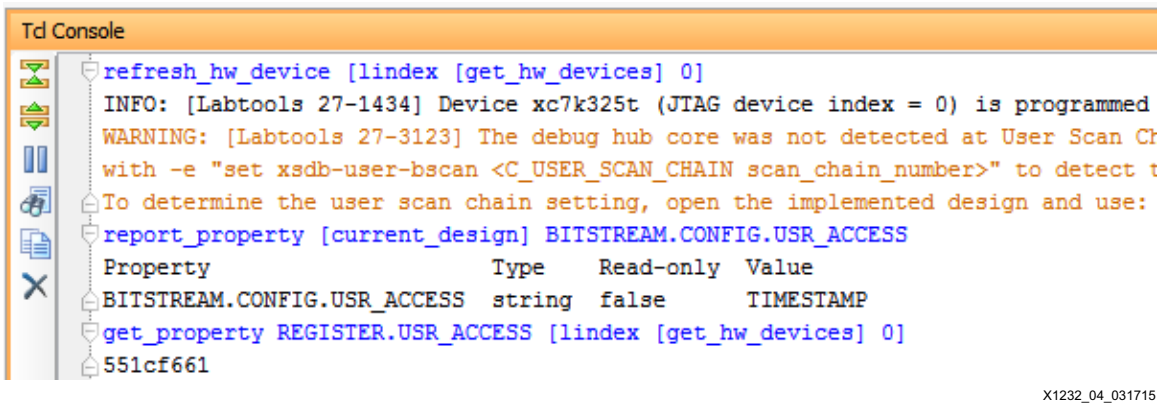

Figure 4: **Tcl get_property**

Conclusion

The USR_ACCESS feature is a method to track bitstreams for revisioning, or to track bitstreams with specific implementation runs without requiring changes to source code or requiring reimplementation.

Appendix A: Instantiation Templates

The following instantiation templates are for 7 series devices. The port names are identical for UltraScale devices.

- Artix®-7: USR_ACCESSE2
- Kintex®-7: USR_ACCESSE2
- Virtex®-7: USR_ACCESSE2

The VHDL instantiation template for 7 series devices is:

```
Library UNISIM;
use UNISIM.vcomponents.all;

USR_ACCESS_7series_inst : USR_ACCESSE2

port map (
```

```
CFGCLK => CFGCLK, -- Not utilized in the static use case in this application note
DATA => DATA, -- 32-bit output Configuration Data output
DATAVALID => DATAVALID -- Not utilized in the static use case in this application note
);
```

Note: Instantiations are also available in the Vivado Language Templates.

The Verilog instantiation template for 7 series devices is:

```
USR_ACCESSE2 USR_ACCESS_7series_inst (
CFGCLK(CFGCLK), // Not utilized in the static use case in this application note
DATA(DATA), // 32-bit output Configuration Data output
DATAVALID(DATAVALID) // Not utilized in the static use case in this application note
);
```

Appendix B: Bitstream Composition

The USR_ACCESS register can be found in the bitstream by searching for the command:

```
Type 1, Write command, address 01101, 1 word

00110000000000011010000000000001 - 0x3001A001
```

The 32-bit value after that command is the USR_ACCESS register value. Details on the syntax to read and write values through the configuration port can be found in the configuration details chapter of the respective configuration user guide [Ref 1] or [Ref 2].

The following is an annotated section of a Kintex-7 device bitstream (in raw bit file [.rbt] format) with a TIMESTAMP value.

```
00110000000000010001000000000001
00000000000000000000000000000000
00110000000000011010000000000001  • Type 1, write, address 01101, one word
01010101000111001111011001100001  • USR_ ACCESS value
00110000000000010011000000000001
00000000000000000000000000000000
```

The above example stores this TIMESTAMP value: REGISTER.USR_ACCESS:0x551cf661

Breaking down the 32-bit USR_ACCESS value results in a TIMESTAMP of 10/10/14 3:25:33 pm.

```
01010 1010 001110 01111 011001 100001
```

Where:

5 bits for day: `01010` = 10th day

4 bits for month: `1010` = 10th month

6 bits for year: `001110` = 14th year (2014)

5 bits for hour: `01111` = 15

6 bits for minute: `011001` = 25

6 bits for seconds: `100001` = 33

References

1. *UltraScale Architecture Configuration User Guide* (UG570)
2. *7 Series FPGAs Configuration User Guide* (UG470)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/03/2016	1.0	Initial Xilinx release.

Please Read: Important Legal Notices