XAPP497 (v1.0) August 15, 2011

Bitstream Identification with USR_ACCESS

Author: Arthur Yang

Summary

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 the Virtex®-5, Virtex-6, and all 7 series FPGAs, provides the ability to embed version information into a 32-bit fabric-accessible register at the bitstream generation phase, allowing for the best balance of flexibility for the user with minimal impact to the design and implementation time.

JTAG USERCODE Solution

The Virtex-5, Virtex-6, and all 7 series FPGAs support the IEEE Std. 1149.1 JTAG feature USERCODE that is similar to USR_ACCESS. USERCODE is also a 32-bit register that can store a user designated value. However, one limitation to the USERCODE is that it is not accessible via the FPGA logic. USERCODE is intended for quick access via JTAG. USERCODE is also entered during the Generate Programming File process. If there is no need for dynamic timestamp or logic access to this value, USERCODE can be an acceptable solution.

USR_ACCESS Primitive

The primitive names for the USR_ACCESS registers are USR_ACCESS_VIRTEX5, USR_ACCESS_VIRTEX6, and USR_ACCESSE2. 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 by additional writes to this configuration register either through the external configuration interface, JTAG, or the internal configuration access port (ICAP). Dynamic writes utilize the CFGCLK and DATAVALID ports. For a static value, only the 32-bit DATA bus is required. 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.

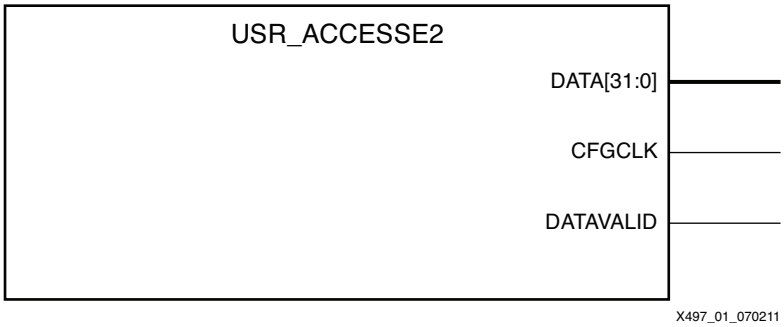

Figure 1: **Schematic of USR_ACCESS Component for 7 Series FPGAs**

Timestamp

USR_ACCESS can be configured with a 32-bit user-specified value or automatically loaded by the bitstream generation program (BitGen) with a timestamp. The user-specified value can be used for revision, design tracking, or serial numbers. 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.

Command Usage

The command usage feature is implemented using a **`-g USR_ACCESS`** switch in the Generate Programming File process (BitGen).

```
-g USR_ACCESS: NONE | xxxxxxxx | TIMESTAMP
```

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

```
NONE - DEFAULT
```

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

```
xxxxxxxx
```

When the keyword TIMESTAMP is entered as the value:

```
TIMESTAMP
```

BitGen 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:

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

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

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

`hhhhh` = 5 bits to represent 23 hours in a day

`mmmmmm` = 6 bits to represent 59 minutes in an hour

`ssssss` = 6 bits to represent 59 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 BitGen, 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.

ISE® Tools Flow

This option is entered in the Generate Programming File process. To access the options:

1. Right click **Generate Programming File**.
2. Select **Process Properties**.
3. Select **Advanced** for Property Display Level.
4. Select **Configuration Options** in the Category panel.
5. Enter the command shown in Figure 2 in the **User Access Register Value** field.
6. Replace the text **aced0123** with **TIMESTAMP** if the timestamp feature is desired.

Note: Not entering this option results in the default NONE that leaves this register with all 0s.

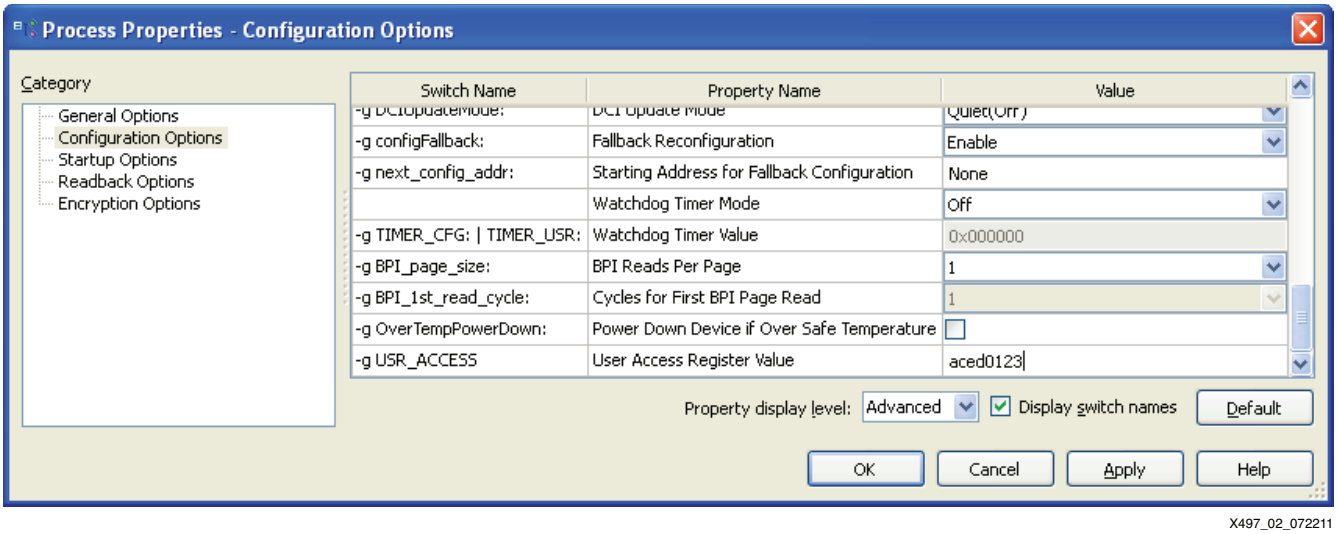

Figure 2: **Configuration Options**

Conclusion

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

Appendix A: Instantiation Templates

The following instantiation templates are for 7 series devices. The port names are identical for Virtex-5 and the Virtex-6 devices. The only difference is the name of the component.

- Virtex-5: USR_ACCESS_VIRTEX5
- Virtex-6: USR_ACCESS_VIRTEX6
- 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
);
```

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
```

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.

The following is an annotated section of a Kintex-7 device bitstream with a timestamp value.

```
00110000000000100010000000000001
00000000000000000000000000000000
00110000000000011010000000000001   ← Type 1, write, address 01101, one word
11101001100101110101110101000011   ← USR_ ACCESS value
00110000000000100110000000000001
00000000000000000000000000000000
```

The above example stores this timestamp value:

```
11101 0011 001011 10101 110101 000011
```

Where:

- 5 bits for day: `11101` = 29th day
- 4 bits for month: `0011` = 3rd month
- 6 bits for year: `001011` = 11th year (2011)
- 5 bits for hour: `10101` = 21
- 6 bits for minute: `110101` = 53
- 6 bits for seconds: `000011` = 3

Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
08/15/11	1.0	Initial Xilinx release.

Notice of Disclaimer

fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: http://www.xilinx.com/warranty.htm#critapps.