Gisselquist
Technology, LLC

# ICAPE2 ACCESS VIA WISHBONE SPECIFICATION

Dan Gisselquist, Ph.D.
dgisselq (at) opencores.org

April 22, 2016

# Revision History

| Rev. | Date | Author | Description |
|---|---|---|---|
| 0.2 | 4/22/2016 | Gisselquist | Minor Updates |
| 0.1 | 5/25/2015 | Gisselquist | First Draft |

# Contents

# Tables

# Preface

My thanks to those helpers on the Xilinx Forum who helped me through the final step in getting this working.

Dan Gisselquist, Ph.D.

# 1.

# Introduction

This core makes the ICAPE2 FPGA configuration registers available to be read or written from a wishbone bus. As Xilinx's documentation of this capability leaves a bit to be desired, I have put this file together to help document what works.

The interface itself is very valuable for a couple of purposes—from my humble and personal perspective. The first is the user configurable watchdog timer which can be used to automatically reset an FPGA after it locks up. The second is the warm boot start capability, which makes it possible to create a fall back configuration image and test it without compromising the ability of the FPGA to be started in a known good image. The third valuable capability is that of commanding a reconfiguration. All of these capabilities are available through this interface. Further details are available from Xilinx's "7-Series FPGAs Configuration" User Guide[1].

This introduction is the first chapter. Beyond this introduction, most of the capabilities are documented elsewhere. Hence, the register chapter will be omitted and the reader will be gently pointed to the User's Guide. This leaves the Wishbone chapter and the I/O Port's chapter which follow.

As always, write me if you have any questions or problems.

[1]For the Spartan, further details are available in the "Spartan–6 FPGA Coniguration" User Guide

# 2.

# Architecture

If I understand correctly, every one of Xilinx's 7–Series FPGA's contains two ICAPE2 interface modules. These modules allow user logic to communicate with the configuration interface of the chip. This interface, however, isn't well documented. According to the User's Guide, it matches the SelectMAP interface, yet in practice ... it doesn't. It may come close, but the timing and interface requirements of the SelectMAP aren't really the same as the ICAPE2 port.

This core encapsulates the difficulty of matching that interface. Register addresses match those in the User's Guide, as do register definitions.

# 3.

# Operation

Realistically, this is better documented by Xilinx than anything you will find here. Still, two examples might be worthwhile.

First, consider the warm boot reload operation. To do this, write the address in configuration memory of an FPGA image to the warm boot start address (WBSTAR). In this case, that is address 5'h10 within this interface. A second write to the configuration command address (CMD), 5'h4 in this interface, will issue the IPROG command to the FPGA and cause it to configure itself from the address you just gave it.

To show this again in C code, consider Fig. 3.1. There, wasn't that simple?

We could do it for the Spartan–6 as well, as shown in Fig. 3.2.

Now I can, from the comfort of my home, reconfigure an FPGA in my office without needing to press the power button or connect to a JTAG cable. Not bad, no?

```
warmboot(uint32 address) {
    uint32_t *icape = (volatile uint32_t *)0x<ICAPE port address>;
    icape[16] = address;
    icape[4] = 15;
    // FPGA is now reconfiguring itself from the new address
    // If executed on an FPGA, this routine will never return.
}
```

Figure 3.1: Series–7 ICAPE2 Usage

```
warmboot(uint32 address) {
    uint32_t *icape6 = (volatile uint32_t *)0x<ICAPE port address>;
    icape6[13] = (address<<2)&0x0ffff;
    icape6[14] = ((address>>14)&0x0ff)|((0x03)<<8);
    icape6[4] = 14;
    // The Spartan–6 is now reconfiguring itself from the new address.
    // If executed from a softcore internal to a Spartan–6, this routine
    // will never return.
}
```

Figure 3.2: Spartan–6 ICAPE Usage

# 4.

# Wishbone Datasheet

Tbl. 4.1 is required by the wishbone specification, and so it is included here. The big thing to notice

| Description | Specification |
|---|---|
| Revision level of wishbone | WB B4 spec |
| Type of interface | Slave, Read/Write |
| Port size | 32–bit |
| Port granularity | 32–bit |
| Maximum Operand Size | 32–bit |
| Data transfer ordering | (Irrelevant) |
| Clock constraints | See the Datasheet for your part |
| Signal Names | Signal Name / Wishbone Equivalent:<br>i_clk — CLK_I<br>i_wb_cyc — CYC_I<br>i_wb_stb — STB_I<br>i_wb_we — WE_I<br>i_wb_addr — ADR_I<br>i_wb_data — DAT_I<br>o_wb_ack — ACK_O<br>o_wb_stall — STALL_O<br>o_wb_data — DAT_O |

Table 4.1: Wishbone Datasheet

is that this ICAPE2 interface acts as a wishbone slave, and that all accesses to the ICAPE2 registers become 32–bit reads and writes to this interface. Bit ordering is the normal ordering where bit 31 is the most significant bit and so forth. (Bit reversal is accomplished internally to match Xilinx's definition.) The `o_stall` and `o_ack` lines are necessarily used to deal with the fact that operations to the device take many clocks to complete (14 for writes, 21 for reads), so be prepared to wait a couple of clocks for your access to complete. Further, the `o_ack` line will go high while the bus is stalled in many cases, indicating that the operation is complete but that the core is not yet ready to handle a subsequent request.

# 5.

# I/O Ports

This core offers no I/O ports beyond those of the wishbone discussed in Chapt. 4. The I/O ports associated with the ICAPE2 interface are captured internally, and not brought to the output of this core.