Altera Virtual JTAG

(Test Access Port)

Author: Nathan Yawn
nathan.yawn@opencores.org

Rev. 1.0

July 18, 2008

History

Rev.	Date	Author	Description
1.0	18/07/2008	Nathan Yawn	First Draft

Contents

1

Introduction

The Altera Virtual JTAG core is used for development purposes (hardware and software debugging). It uses the Altera sld_virtual_jtag megafunction to allow debugging of an OpenRisc-based System-on-Chip (SoC) using the same JTAG interface IO pins used to upload the bitstream to the FPGA. This core can only be used in Altera FPGAs which support the sld_virtual_jtag megafunction. Before continuing with this document, it is recommended that you familiarize yourself with the IEEE 1149.1 specification (JTAG and Boundary Scan), and also that you read the Altera document "sld_virtual_jtag Megafunction User Guide."

The altera_virtual_jtag core is designed to replace the "jtag" Test Access Port (TAP) core in an Altera-based system. It is designed to provide an interface between the FPGA's JTAG pins and the SoC debug core. In particular, the altera_virtual_jtag core is designed to interface with the Advanced Debug Interface (adv_dbg_if core). Other debug cores may require modification in order to work with this one. Note that while the virtual JTAG core provides only a single device enable output (for the debug core), it is a fully functional JTAG TAP, and other Data Register chains could easily be added.

2

Operation

This section describes the operation of the altera_virtual_jtag core. The altera_virtual_jtag core relies heavily on the Altera sld_virtual_jtag megafunction for its operation. In fact, the core is primarily a thin wrapper and instruction decoder for the megafunction.

2.1. Altera sld_virtual_jtag Megafunction

This megafunction is designed to allow a user-defined JTAG scan chain to be accessed through the same pins used to upload a configuration bitstream to the FPGA. It consists of two components. The "hard TAP" is the chip's primary JTAG TAP, with external TCK, TMS, TDI, and TDO pins. The hard TAP is implemented in VLSI as part of the FPGA chip, and cannot be changed by configuring the FPGA. The "virtual TAP" is implemented using FPGA LUT and register resources. It has no external access pins; the virtual TAP IR and DR are accessed through the hard TAP. This allows a virtual TAP to be created with any length of IR, and any number of DRs.

In order to write a value to the virtual TAP IR, a user must write a VIR instruction to the hard TAP IR. This makes the virtual IR active as a Data Register of the hard TAP. A value can then be shifted into the virtual IR using a data shift. Once the virtual IR has been set, the virtual DR can be made active by placing a VDR instruction in the hard TAP's IR.

2.2. Instructions

The altera_virtual_jtag instantiates an sld_virtual_jtag with a virtual IR 4 bits long, the same length used by the "jtag" core. Because the hard TAP in the FPGA provides all of the standard JTAG instructions (BYPASS, MBIST, IDCODE, etc.), the virtual TAP does not support them, in order to save logic. The virtual TAP supports only a single instruction, DEBUG (IR=1000b), which is used to communicate with the SoC debug interface (see the documentation for the adv_dbg_if for details). The value for this instruction was also selected to match the value used by the "jtag" core.

The debug interface core is connected as the Data Register (DR) of the virtual JTAG when the DEBUG instruction is active in the virtual IR. Data from the altera_virtual_jtag core to the debug core is assumed to be latched on the rising edge of the TCK ouput clock. Data from the debug core to the virtual jtag core is latched on the falling edge of TCK.

2.3. Communicating with the sld_virtual_jtag

Altera does not publish details on communication with the sld_virtual_jtag megafunction. However, using Altera's command-line JTAG communication tool, it is possible to determine the procedure necessary to drive the sld_virtual_jtag. The following conventions should be used:

- The VIR instruction is 0x0E for all currently known Altera FPGAs. Placing this (10-bit) value in the hard TAP's IR will make the virtual IR active.
- One bit is added to the length of the virtual IR specified in the sld_virtual_jtag instantiation. Thus, the virtual IR length of the altera_virtual_jtag core is actually 5. A '1' must be placed into this extra bit, in the MSB position. To make the virtual DEBUG instruction active, the 5-bit value “11000” must be shifted into the virtual IR.
- The VDR instruction is 0x0C for all currently known Altera FPGAs. Placing this (10-bit) value in the hard TAP's IR will make the virtual DR active. The virtual DR is directly connected as the hard DR, no bits are added.

Note that these conventions have only been tested for the altera_virtual_jtag core, and may not work for other uses of the sld_virtual_jtag megafunction. In particular, the added '1' bit required in the virtual IR may change if more than one virtual JTAG device is used in the system (e.g. SignalProbe, SignalTap, or a second sld_virtual_jtag).

3

IO Ports

This section describes the top-level ports of the altera_virtual_jtag core. Because all of the JTAG signals come through the sld_virtual_jtag megafunction, the only ports from the core are used to interface to the debug core.

3.1. Debug Ports

Port	Width	Direction	Description
tck_o	1	output	JTAG clock signal
test_logic_reset_o	1	output	TAP controller state “Test Logic Reset”, acts as reset signal to sub-modules (debug unit etc.)
run_test_idle_o	1	output	TAP controller state “Run Test / Idle”
shift_dr_o	1	output	TAP controller state “Shift DR”
pause_dr_o	1	output	TAP controller state “Pause DR”
capture_dr_o	1	output	TAP controller state “Capture DR”
update_dr_o	1	output	TAP controller state “Update DR”
debug_select_o	1	output	Debug select, true when DEBUG instruction active in the virtual IR
tdi_o	1	output	TDI signal, connects to all TDI signals of sub-modules (i.e. debug module)
debug_tdo_i	1	input	TDO signal from debug module

Table 1: Debug Ports

4

Registers

This section specifies all registers in the Altera Virtual JTAG core.

4.1. Register List

Name	Width	Access	Description
Virtual IR	5	R/W	Virtual Instruction Register

Table 2: Register List

4.2. Virtual IR (Instruction Register)

Bit #	Access	Description
4	R/W	Must be written as '1', required by sld_virtual_jtag
3:0	R/W	Data Register to make active. 1000 = DEBUG

Table 3: IR Register

This register is always read as 10101b.

5

Integrated System

The Altera Virtual JTAG core is just one part of the complete debugging system. To be useful, the system-on-chip must also include a compatible debug core (i.e. the adv_dbg_if core), a WishBone bus, and an OR1200 CPU (or a CPU with a compatible debug interface). Externally, the debugging system must include a JTAG cable, GDB (the GNU Debugger program), a GDB-to-JTAG bridge program (i.e. adv_jtag_bridge), and an optional graphical front-end to GDB, such as DDD or Eclipse. A block diagram of this system is shown in Figure 1.

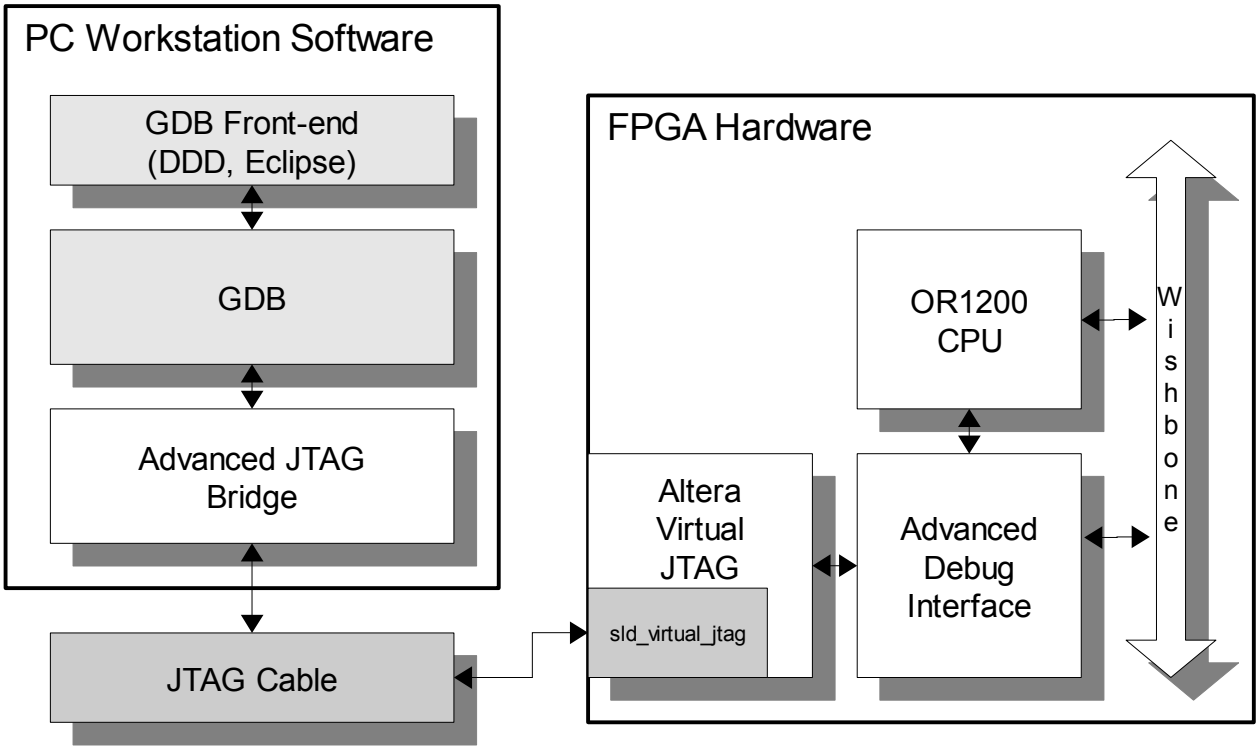

Figure 1: Complete Debug System Block Diagram