

JTAG programming device for FPGA configuration EEPROMs

Miroslav Vadkerti

19.12.2006

1 Abstract

This technical report describes a TAP controller device developed for programming JTAG compliant devices. As the programming data is read from Xilinx SVF¹ file generated by Xilinx iMPACT software, there was implemented a parser for this file format using lex and yacc². The solution is used for programming configuration EEPROMs of FPGAs on COMBO6X cards.

Keywords: JTAG, Test Access Port, Xilinx SVF

2 Introduction

JTAG³ (acronym for Joint Test Action Group) is the usual name used for Standard Test Access Port and Boundary-Scan Architecture (IEEE 1149.1). Boundary-Scan architecture enables engineers to perform extensive debugging and diagnostics of a system through a small number of dedicated test pins. Signals are scanned into and out of the I/O cells of a device serially to control its inputs and test the outputs under various conditions. The Boundary-Scan control signals, collectively referred to as the Test Access Port (TAP), define a serial protocol for scan-based devices.

There are five pins (first four are compulsory, last is optional) in TAP:

TCK/clock: Synchronizes the internal state machine operations.

TMS/mode select: Is sampled at the rising edge of TCK to determine the next state.

¹<http://www.xilinx.com/bvdocs/appnotes/xapp503.pdf>

²<http://dinosaur.compilertools.net/>

³<http://en.wikipedia.org/wiki/JTAG>

TDI/data in: Is sampled at the rising edge of TCK and shifted into the device's test or programming logic when the internal state machine is in the correct state.

TDO/data out: Represents the data shifted out of the device's test or programming logic and is valid on the falling edge of TCK when the internal state machine is in the correct state.

TRST/reset: This signal is optional. When driven low, resets the internal state machine.

JTAG is designed so that multiple chips on a board can have their JTAG lines (TAPs) daisy-chained together, and a TAP controller device need only connect to a single TAP connector to have access to all chips placed on a circuit board. The operation of JTAG is controlled by the TAP controller⁴. TAP Controller is a state-machine whose state transitions are controlled by the TMS signal.

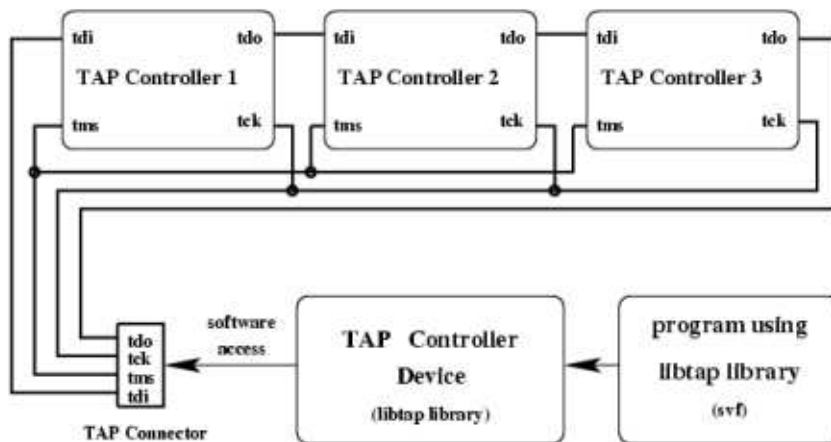


Figure 1: TAP controllers (TAP Controller 1, 2 and 3) in JTAG compliant devices connected to an off-board TAP control device (libtap library) through a Test Access Port (TAP) connector.

3 Test access port connectors

TAP connector is a hardware port accessible from software. This port is linked using a cable to JTAG lines of programmed chips on COMOBO6X board. In our solution we use these two TAP connectors for programming:

- PCI TAP connector - accessible with libcombo as PCI bridge register

⁴<http://www.xjtag.com/jtag-technical-guide.php>

- Parallel port TAP connector - accessible as standard parallel port

These TAP connectors make it possible to program the chips remotely (without the need of physical manipulation with the machine) because they are permanently attached to the JTAG lines and so are accessible to the developed software anytime.

3.1 Important connectors and jumpers on COMBO6X card

To ensure correct functionality of the TAP connectors, jumper JP1 has to be short and jumpers JP0, JP2, JP3 have to be open on COMBO6X board.

Connectors JP4 and JP7 are in parallel connection. They share the TAP signals TDO, TDI, TMS, TCK and TMSP.

connectors JP4, JP5, JP7
 jumpers JP0, JP1, JP2, JP3

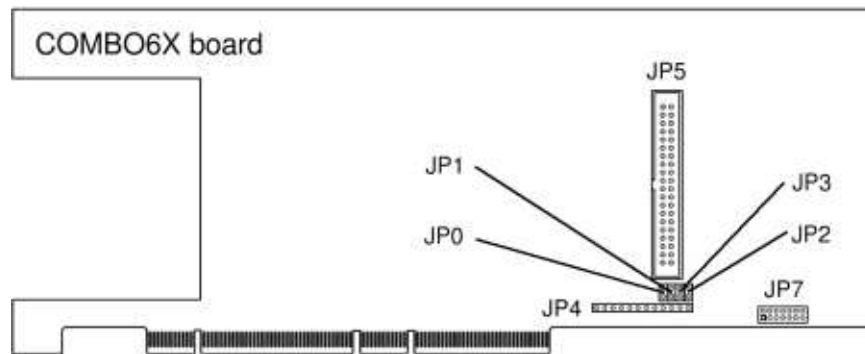


Figure 2: Important connectors and jumpers on COMBO6X board for JTAG programming device.

3.2 PCI TAP connector

This TAP connector is located directly on COMBO6X board. It consists of pins IO29, IO33, IO36, and IO40 of connector J5. These pins are accessible as bits of two PCI bridge registers. PCI bridge registers are accessible with libcombo library in PCI bridge space. To use this connector a cable connecting connector JP5 and connector JP7 must be present on COMBO6X board as described in Figure 3. The advantage of this connector is that it is easily accessible using libcombo library and it only requires a small (cheap) cable connection just on board of the COMBO6X board.

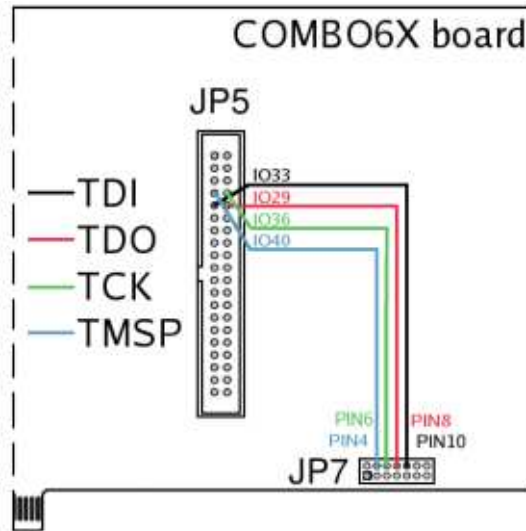


Figure 3: JTAG interface (JP7) connected to FPGA PCI bridge space (JP5).

3.3 Parallel port TAP connector

This TAP connector is accessible as a standard parallel port. It is connected to JTAG lines in connector JP4 on COMBO6X board. For connection the Xilinx DLC5 Type3 parallel cable is used. The main disadvantage of this connector is that the cable must be wired from COMBO6X board to the outer parallel port of the machine. Also the cable is larger and its production is more expensive.

Note:: For construction details on Xilinx DCL5 Type3 cable see the OpenWrt wiki⁵. The maximum length of the cable should be 3 meters.

In practice we use the PCI TAP connector because it has more advantages over the parallel port TAP connector.

4 TAP library (libtap)

Libtap library consists of 3 main interfaces and some support routines for controlling TAP controller. The main interfaces are JTAG, pinbus and membus interface with modular structure. Modules written for these interfaces make it possible to communicate with various TAP controller types via different TAP connectors. The communication with a TAP connector is specified by pinbus and membus modules. The JTAG module specifies the TAP controller communication protocol.

⁵http://wiki.openwrt.org/OpenWrtDocs/Customizing/Hardware/JTAG_Cable

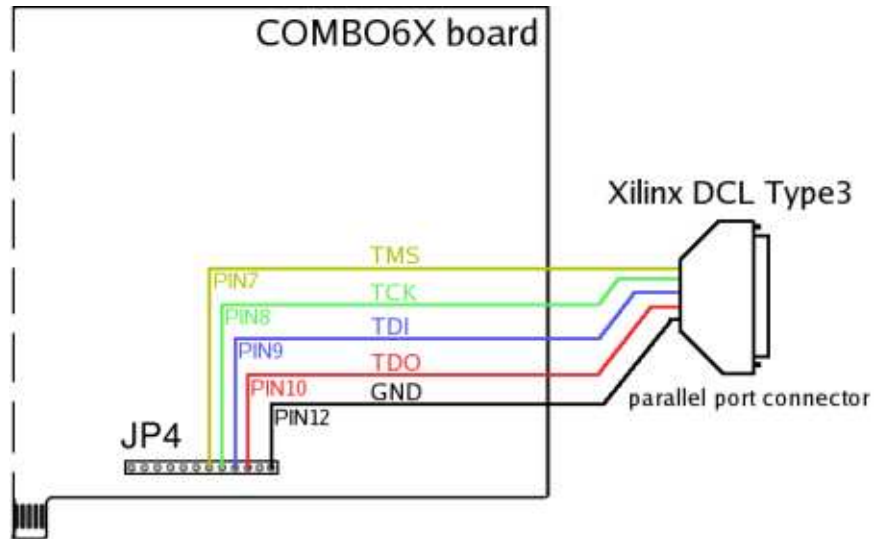


Figure 4: JTAG interface (JP7) connected to FPGA's PCI bridge space (JP5).

4.1 Library structure

4.1.1 Public interface

TAP interface (tap.h): contains the configuration interface for TAP devices

JTAG interface (jtag.h): JTAG (TAP) protocol interface

bitstring data type (bitstring.h): used for storing I/O data

misc utilities (util.h): support routines (verbose, error messages, ..)

4.1.2 Private interface

pinbus interface (pinbus.h): interface for modules abstracting pin operations on a generic memory bus

membus interface (membus.h): interface for modules abstracting memory operations (read, write) on TAP connectors (HW devices)

queue data type (queue.h): macros for working with queue data type (used across the library)

hash table (hash.h): associative table for used for storing and searching in configuration data

4.1.3 Interface modules

COMBO6X membus module (comboio.h): membus module accessing TAP connector in COMBO6X PCI bridge space

parallel port membus module (pcio.h): membus module accessing TAP connector attached to PC's parallel port

COMBO6X pinbus module (pcpci_membus.h): pinbus module working over COMBO6X membus module to access specific bits (TAP pins) in TAP connector

parallel port pinbus module (pclpt_membus.h): pinbus module working over parallel port membus module to access pins in TAP connector

JTAG module (jtag_pinbus.c) JTAG module working over pinbus module

4.2 Library usage

The typical usage of the library is shown in Figure 5. This figure also describes communication between main parts of the library. TAP connector is an access point to TAP signals from JTAG compliant device. Program using the library first needs to configure the TAP modules by calling the function `tap_setup(devname)`. The `devname` parameter specifies the name of the device with TAP connector attached. According to the device name the configuration file is chosen, which specifies the modules which should be used. The configuration file is also used to specify the TAP pins layout. For an example of the configuration file see Appendix A.

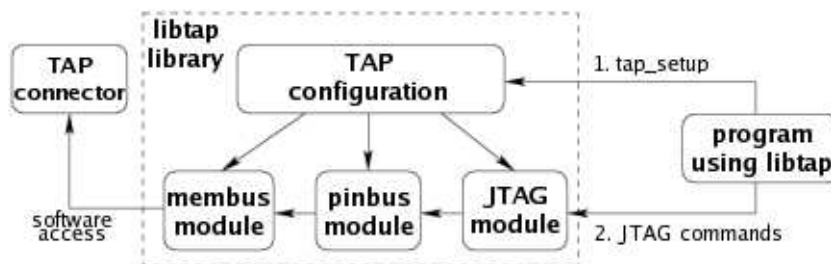


Figure 5: Libtap usage

After successful `tap_setup` the program should attach the last module in chain, using the function `tap_query_attribute` and reset the TAP controller to known state with `jtag_reset` function. After these steps the JTAG commands defined in `jtag.h` can be used to control the TAP controller in the JTAG compliant device.

4.3 Adding new TAP connector driver

Thanks to the modular interface of the library adding a new TAP connector driver to library requires in most cases only writing new modules for membus and pinbus interfaces. These modules specify access to the new TAP connector. The attachment of the new modules must be specified in attconf.c file. For the new TAP connector also a new configuration file is needed, which specifies the relation between the different modules and it's used for the definition of pin layout of the TAP signals.

5 SVF file parser and interpreter

Developed CLI tool svf is a Xilinx SVF file parser and interpreter. It was built using the lex and yacc tools and it uses the libtap library for interpreting the parsed SVF file. SVF file is used to record JTAG operations by describing the information that needs to be shifted to the device chain of the programmed device. SVF file is written as ASCII text and therefore can be easily parsed.

JTAG operations shift data into and out of JTAG instruction and Data registers. The TAP controller provides direct access to all these registers. JTAG registers are divided into two classes: instruction register - IR (only one) and data registers - DR (many). Access to the IR is provided through the Shift-IR state and access to the DR is provided through the Shift-DR state.

To provide the desired operation (e.g. shift) the TAP controller of the programmed device must be moved to the corresponding state. The data is shifted into the device starting from LSB first.

5.1 Basic SVF Commands

Shifts to the Instruction and Data registers are specified in SVF by two instructions:

Scan Instruction Register (SIR)

synopsis:

```
SIR length TDI (tdi) TDO (tdo) SMASK (smask);
```

parameters:

```
length - specifies the number of bits to be shifted
         into the Shift-IR state.
```

```
TDI - specifies the scan pattern to be applied to the Shift-IR state.
```

```
SMASK - specifies don't care bits in the scan pattern.
```

Scan Data Register (SDR)

synopsis:

```
SDR length TDI (tdi) SMASK (smask) [TDO (tdo) MASK (mask)];
```

parameters:

length - specifies the number of bits to be shifted
into the Shift-IR state.

TDI - specifies the scan pattern to be applied to the Shift-IR state.

SMASK - specifies don't care bits in the scan pattern.

TDO - specifies the expected pattern on TDO while shifting through
the Shift-DR state.

MASK - specifies don't care bits in the expected TDO pattern.

The third SVF instruction of importance to Xilinx users is the RUNTEST instruction. The RUNTEST instruction specifies an amount of time for the TAP Controller to wait in the Run-Test-Idle state. This wait time is a required part of the programming algorithm for certain Xilinx devices.

RUNTEST

synopsis:

```
RUNTEST run_count TCK;
```

parameters:

TCK - specifies the amount of time to wait in Run-Test-Idle state

All these three SVF instructions can be used to perform nearly any JTAG operation on any JTAG compliant device. For detailed specification of the SVF format and commands please see the SVF⁶ specification.

The programming frequency of the tool is fixed (about 160kHz on a 2GHz Intel Pentium 4 processor) and is given by the maximum speed of the program execution (parsing and interpreting). It is way below the maximum programming frequency that is used by some USB JTAG programmers (about 5MHz) so no delay is used between the executed JTAG commands.

5.2 List of tested Xilinx PROMs programmed with svf tool

- XC18V04
- XC18V512

⁶<http://www.xilinx.com/bvdocs/appnotes/xapp503.pdf>

5.3 Example of svf tool usage

The svf tool parameters specify the device used to access the TAP connector and the input file. If no device is specified, /dev/combosix/0 device is used as default.

```
% svf -f combo6x_rev1.svf -v
```

Program PROM(s) through TAP connector connected to default device /dev/combosix/0, be verbose.

```
% svf -d /dev/combosix/1 -f combo6x_rev1.svf
```

Program PROM(s) through TAP connector at /dev/combosix/1.

```
# svf -d /dev/parport/1 -f combo6x_rev1.svf -v .. program PROM(s) through TAP:  
connector connected to parallel port 2 (0x278), be verbose
```

6 Conclusions

This technical report introduced a JTAG programming software developed for programming configuration EEPROMs for FPGAs on COMBO6X cards. The developed library libtap, thanks to its modular structure, is easily expandable with new modules and can suite different purposes in the future where JTAG interface is available.

7 Appendix A - COMBO6X PCI bridge space TAP connector configuration file

```
# attach membus module  
comboio0 at root  
  
# attach pinbus to membus module  
pcpci0 at comboio0 base 0x0 size 0x100000  
  
# attach JTAG module to pinbus module  
# JTAG signals are mapped to I00(tdo) and I01(tdi,tck,tms) registers  
# Dx - represents the x-th mapped bit in I00 (for tdo) or I01 (for every other s  
jtag0 at pcpci0  
{
```

```

        tdo      D29
        tdi      D1
        tck      D4
        tms      D8
    }

```

8 Appendix B - Example of an SVF file

This is only a small part of the whole SVF file.

```

// Created using Xilinx iMPACT Software [ISE Foundation - 7.1.04i]
TRST OFF;
ENDIR IDLE;
ENDDR IDLE;
STATE RESET IDLE;
FREQUENCY 1E6 HZ;
TIR 0 ;
HIR 0 ;
TDR 0 ;
HDR 0 ;
TIR 8 TDI (ff) SMASK (ff) ;
HIR 0 ;
HDR 0 ;
TDR 1 TDI (00) SMASK (01) ;
//Loading device with 'idcode' instruction.
SIR 8 TDI (fe) SMASK (ff) ;
SDR 32 TDI (00000000) SMASK (ffffffff) TDO (f5036093) MASK (0ffeffff) ;
//Loading device with 'conld' instruction.
SIR 8 TDI (f0) ;
RUNTEST 110000 TCK;
//Check for Read/Write Protect.
SIR 8 TDI (ff) TDO (01) MASK (03) ;
// Validating chain...
TIR 0 ;
HIR 0 ;
TDR 0 ;
HDR 0 ;
SIR 16 TDI (ffff) SMASK (ffff) TDO (0101) MASK (0303) ;
TIR 0 ;
HIR 0 ;
TDR 0 ;
HDR 0 ;

```

```
TIR 0 ;
HIR 0 ;
TDR 0 ;
HDR 0 ;
STATE RESET;
// Loading devices with 'ispen' or 'bypass' instruction.
SIR 16 TDI (ffe8) ;
SDR 7 TDI (37) SMASK (7f) ;
// Loading device with 'faddr' instruction.
SIR 16 TDI (ffeb) ;
SDR 17 TDI (000001) SMASK (01ffff) ;
RUNTEST 1 TCK;
RUNTEST 1 TCK;
// Loading device with 'ferase' instruction.
SIR 16 TDI (ffec) ;
RUNTEST 1 TCK;
RUNTEST 15000000 TCK;
// Loading devices with 'conld' or 'bypass' instruction.
SIR 16 TDI (fff0) ;
RUNTEST 110000 TCK;
STATE RESET;
```