5 OpenOCD

5.1 Einleitung

OpenOCD[2] bildet den Software Teil von einem Debugger. Zusammen mit einem Hardware Adapter bildet OpenOCD einen vollständigen Debugger und kann als Ersatz für einen teuren Debugger wie beispielsweise dem BDI 3000 von Abatron verwendet werden.

Der Adapter bildet dabei das elektrische Interface zum Prozessor und muss auch auf den Prozessor abgestimmt sein. Relevant sind dabei unter anderem der Transport Layer (JTAG/SWD) das elektrische Potential und natürlich auch der Physikalischer Stecker. In den vielen Fällen basieren solche Adapter, wenn sie zusammen mit OpenOCD verwendet werden, auf dem FT2232 Chip von FTDI. Solch ein generischer Adapter ist in der Abbildung 5.1 zu sehen.

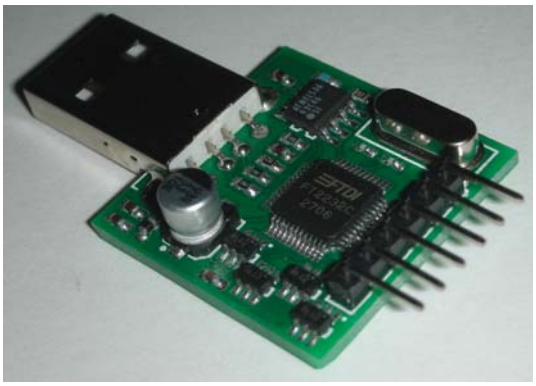

Abbildung 5.1: Generischer JTAG Adapter mit einem FTDI FT2232[3]

Bei Experimentierboards ist der FT2232 oft auch direkt auf das Board aufgelötet. So kann eine einfache USB Verbindung genutzt werden, um den Prozessor zu debuggen. Beim Zybo wurde ebenfalls dieser Ansatz verfolgt. Aus diesem Grund reicht ein einfaches USB Kabel um den Prozessor des Zybos auf einer Hardware-Ebene debuggen zu können.

5.2 Installation

5.2.1 Installation - OpenOCD

OpenOCD kann direkt vom Sourcecode kompiliert werden[1] oder auch als vorkompiliertes Binary herunter geladen werden. Für diese Arbeit wurde das vorkompilierte Windows Binary für ARM Cores Version 0.10.0 von folgender URL verwendet:
http://www.freddiechopin.info/en/download/category/4-openocd?download=154%3Aopenocd-0.10.0

Das eigentliche Binary befindet sich im Ordner:
`/openocd-0.10.0/bin-x64/`

Das Open OCD User Manual[4] befindet sich im Ordner:
`/openocd-0.10.0/`

[1] http://sourceforge.net/p/openocd/code/

5.2.2 Installation - USB Driver WinUSB

Damit OpenOCD mit dem FT2232 Chip kommunizieren kann, werden die richtigen USB Treiber benötigt. Die Installation der Treiber ist am einfachsten mit den *USB Driver Tool*, welches man unter folgender Adresse findet:
http://visualgdb.com/UsbDriverTool/

Das Zybo muss per USB mit dem PC verbunden sein, damit der Treiber installiert werden kann. Wenn der Jumper '*J15*' auf USB gesetzt ist, dann wird keine zusätzliche Stromversorgung für das Zybo benötigt.

Öffnet man das *USB Driver Tool* werden alle USB Devices aufgelistet. Das Device mit der *Vendor ID=0403, Device ID=6010* und *Interface 0* ist das JTAG Interface vom FT2232. Mit einem Rechtsklick darauf kann man den *Install WinUSB* Treiber auswählen und installieren. Abbildung 5.2 zeigt die Liste mit allen USB Devices und das Kontextmenü für die Installation des richtigen Treibers. Nachdem das Zybo einmal aus- und wieder einschaltet wird, ist der Treiber einsatzbereit.

Das Device mit der *Vendor ID=0403, Device ID=6010* und *Interface **1*** ist die UART Verbindung zum Prozessor. Dieser Treiber darf **nicht** ersetzt werden.

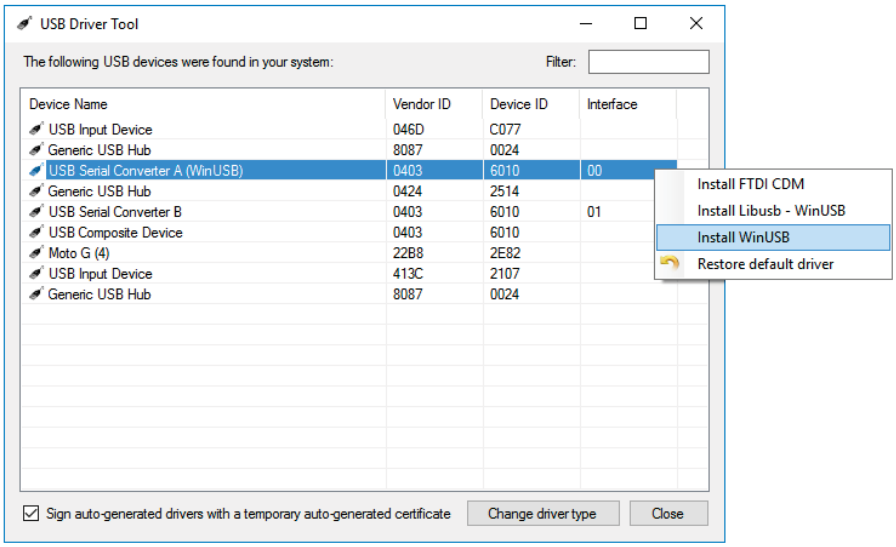

Abbildung 5.2: Installation des *WinUSB* Treibers mit dem *USB Driver Tool*

5.3 OpenOCD CLI - Command Line Interface

Das CLI[2] ist eine einfache Methode um mit dem Debugger zu kommunizieren. Über den Port 4444 kann, z.B. mit Putty, auf dem *Localhost* eine Telnet-Verbindung aufgebaut werden, sobald OpenOCD gestartet wurde. Der Befehl `''help''` listet alle zulässigen Befehle auf.

In den folgenden Kapitel wird folgende Notation verwendet, um ein Befehl zu beschreiben, der das CLI verwendet:
(`CLI: Befehl`)

5.4 OpenOCD Konfiguration - Einleitung

OpenOCD unterstützt eine Vielzahl von Adaptern und Targets (Prozessoren). Beim Start muss die Software für die verwendete Hardware konfiguriert werden. Die Konfiguration erfolgt mit Konfigurationsscripts (*.cfg) in der Scriptsprache *Jim-Tcl*. *Jim-Tcl* ist eine abgespeckte Version von *Tcl*[3].

[2] Command Line Interface
[3] http://www.tcl.tk

Normalerweise werden die Scripts in die drei Gruppen *interface, board* und *target* aufgeteilt. So kann einfach ein Script ausgewechselt werden, wenn man den gleichen Adapter aber einen anderen Prozessor verwenden will. Im Pfad `openocd-0.10.0/scripts` befinden sich eine Sammlung von Konfigurationsscripts für Standardhardware.

Mit folgendem Befehl kann OpenOCD mit Konfigurationsscripten nach Wahl gestartet werden:
`openocd -f zybo-ftdi.cfg -f zybo.cfg`

5.5 OpenOCD Konfiguration - Interface

Die Interface Konfiguration beschreibt hauptsächlich den verwendeten Adapter. Da beim Zybo kein Adapter verwendet wird, sondern der aufgelötete FT2232, wird mit diesem Script der FTDI Chip und dessen Anbindung an den Zynq konfiguriert.

Da ein FTDI-Chip als Interface verwendet wird, sollte ein passender Script unter *openocd-0.10.0/scripts/interface/ftdi/* zu finden sein. Keiner der Scripts passt von Namen her auf *Zynq, Zybo* oder *FT2232*. Eine Google Suche nach einem passenden Script war erfolgreicher. Ein Github User mit dem Namen *emard* hat folgenden Script in einem von seinen Repositories[4] gespeichert:

zybo-ftdi.ocd:

```
1  #
2  # ZYBO ft2232hq usbserial jtag
3  #
4
5  interface ftdi
6  ftdi_device_desc "Digilent Adept USB Device"
7  ftdi_vid_pid 0x0403 0x6010
8
9  ftdi_layout_init 0x3088 0x1f8b
10 #ftdi_layout_signal nTRST -data 0x1000 -oe 0x1000
11 # 0x2000 is reset
12 ftdi_layout_signal nSRST -data 0x3000 -oe 0x1000
13 # green MIO7 LED
14 ftdi_layout_signal LED -data 0x0010
15 #ftdi_layout_signal LED -data 0x1000
16
17 reset_config srst_pulls_trst
```

Zeile 5 bis 7 konfigurieren das Interface als ein standard-FTDI Interface. Von OpenOCD werden neben dem FT2232 auch noch andere Chips unterstützt. Zeile 7 definiert die *Vendor* und *Device-ID* des USB Devices.

5.5.1 Resetverhalten

Liest man aus einer unerlaubten Speicheradresse (`CLI: mdw 0x40000000`), dann hängt sich die Debug-Peripherie des Zynq auf. Nach so einem unerlaubten Speicherzugriff können auch keine erlaubten Speicherstellen mehr gelesen werden. Beim Versuch erscheint die Fehlermeldung:
`Timeout waiting for cortex_a_exec_optcode.`
Mit einem manuellen Powercycle vom Zybo kann die Hardware wieder zurückgesetzt werden.

Mit OpenOCD ist es grundsätzlich möglich, einen Reset automatisch durchzuführen. Dabei wird zwischen einen *System Reset* (SRST) und dem *TAP*[5] *Reset* (TRST) unterschieden. Der SRST führt dabei einen Powercycle vom ganzen System durch, der TRST setzt nur den TAP zurück

Beim obigen Script ist aber das Resetverhalten nicht sauber definiert. Mit dem Befehl ''`CLI: reset halt`'' im CLI sollte ein Reset vom FT2232 durchgeführt werden. Der Befehl führt aber zur Fehlermeldung:
`... zynq.cpu0: how to reset? ...`

[4] https://github.com/f32c/f32c/blob/master/rtl/proj/xilinx/zybo/xram_bram_hdmi_ise/zybo.ocd
[5] Test Access Port

Im OpenOCD User Manual[4] im Kapitel *9 Reset Configuration* ist beschrieben, wie das Resetverhalten konfiguriert werden kann. Mit dem Script-Befehl ''`reset_config srst_only`'' wird der TAP Reset ignoriert. So kann das Problem auf den System Reset begrenzt werden.

Wenn OpenOCD mit der neuen Konfiguration neu gestartet wird, dann scheint der Befehl ''`CLI: reset halt`'' zu funktionieren. Greift man vorher aber wieder auf eine ungültige Speicherstelle zu, dann erscheint beim Reset die Fehlermeldung
`... Timeout waiting for dpm prepare ....`
Der erneute Timeout legt die Vermutung nahe, dass der Zynq nicht ordentlich zurück gesetzt wurde.

Zeile 12 ''`ftdi_layout_signal nSRST -data 0x3000 -oe 0x1000`'' konfiguriert die I/O Pins des FT2232 welche für den System Reset verwendet werden. Auf dem elektrischen Schema vom Zybo**??** könnte man überprüfen, welche I/Os vom FT2232 effektiv für den Reset verwendet werden. Die Seite mit dem Schema für den FT2232, Seite 7, ist aber als einzige Seite im Schema nicht veröffentlicht worden. Die korrekten I/O Pins lassen sich also nicht mit dem Schema ermitteln.

Im OpenOCD User Manual[4] wird der für ''`ftdi_layout_signal nSRST` genauer beschrieben. Der Switch *-data 0x3000* definiert alle relevanten Pins für den SRST und *-oe 0x1000* konfiguriert alle Ausgänge. In einem Versuch wurden diverse Kombinationen für die beiden Switches ausprobiert. Keine Kombination mit nur einem Pin (z.B. *-data 0x2000* mit *-oe 0x2000*) hat funktioniert. Es hat sich herausgestellt, dass die Kombination *-data 0x3000* mit *-oe 0x3000* tatsächlich einen System Reset ermöglicht.

Weil der Debugger direkt nach dem SRST versuch mit dem Zynq zu kommunizieren, tritt folgende Fehlermeldung auf:
`Invalid ACK (7) in DAP response`
`JTAG-DP STICKY ERROR`
Mit dem Kommando ''`adapter_nsrst_delay 40`'' wartet der Debugger nach dem SRST zusätzliche 40 Millisekunden. Diese Wartezeit genügt, damit die Debug Peripherie wieder aufgestartet ist, wenn der Debugger versucht zu kommunizieren.

5.6 OpenOCD Konfiguration - Board

Da beim Zybo der Adapter direkt auf dem Board ist, ist die Bordkonfiguration bereits im Konfigurationsscript für das Interface enthalten.

5.7 OpenOCD Konfiguration - Target

Für das Target, in diesem Fall der Zynq 7000 SOC, ist bereits ein Script unter *openocd-0.10.0/scripts/target/zynq_7000.cfg* enthalten. In diesem Script werden nicht nur beide Kerne des Prozessors definiert, sondern auch ein TAP[6] für das FPGA. Es ist also auch möglich, den FPGA mit dieser Toolchain zu laden.

[6] Test Access Port