Dokumentation des BitConnector v5X

Michael Krause

4. Juni 2021

1 Einführung

Die noch immer andauernden Bemühungen der mikrokosmischen Minimierung von integriertern Schaltkreisen hat Integrationsdichten hervorgebracht, die traditionelle Schaltkreise wie einzelne Register, Zähler, ALUs oder Multiplexer vom Markt verschwinden ließen. Aus der Sicht des Endkonsumentenmarktes mag dieser Schritt folgerichtig sein, jedoch gibt es jenseits dessen noch immer das Bedürfnis auf der Register-Transfer Ebene oder darunter, Logikschaltkreise zu modellieren und zu realisieren. Dieses Bedürfnis kann beispielsweise der akademischen Lehre, dem Wunsch nach Parallelisierung oder der Vorgabe zeitkritischer Anwendungen entspringen. Obwohl es diesbezüglich eine Vielzahl von FPGAs und CPLD-Typen zur Abhilfe gibt, gestaltet sich der prototypische Aufbau auf einem Steckbrett mit Hilfe dieser Chips als nahezu unmöglich. Derartige Chips besitzen meist eine Vielzahl an Pins, die nicht im Dual in-line package (DIP) vorliegen. Der BitConnector versucht dieses Problem zu lösen.

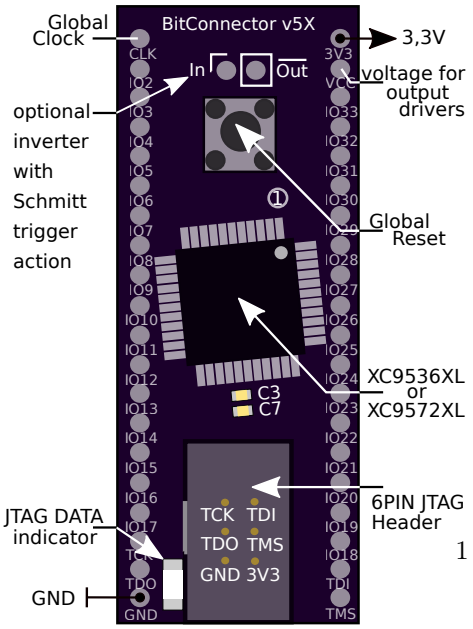

Abbildung 1: BitConnector v5X

Der BitConnector ist ein **C**omplex **P**rogrammable **L**ogic **D**evice, kurz **CPLD** Board, das für den hardwarebasierten, digitalen Prototypenentwurf auf einem Steckboard optimiert wurde und zudem gesockelt auf andere Platinen aufgesetzt werden kann. Das Board ist so entworfen, dass es einen klassischen DIP Chip mit 2,54 mm Rastermaß und 2 * 20 Pins nachbildet. Konfigurationen, die etwa in einer HDL vorliegen, werden mittels JTAG übertragen und dauerhaft auf dem CPLD gespeichert.

2 Eckdaten

BitConnector Board-Spezifikation

- *Anzahl der frei beschaltbaren Ein/Ausgänge:* 33
- *Gesamtanzahl der Pins:* 2 Reihen * 20 Pins (2,54mm)
- *kompatible CPLD-Varianten*: XC95**72**XL sowie XC95**36**XL
- Max. Produktterme: 1600 bei 72XL/800 bei 36XL
- *IC-Package*: VQ44

Absolute, maximale Belastbarkeit:

- *Versorgungsspannung bezogen auf GND:* -0,5V bis 4V
- *Eingangsspannung bezogen auf GND:* -0,5V bis 5,5V
- *Max. Stromfluss bei max. 70 °C Chiptemp.*: $\approx 1,6A$ (siehe Kapitel 4.4)

Es wird empfohlen, Spannungen über und unter 3,3V($\pm 0,3V$) zu vermeiden!

3 Pinbelegung BitConnector v5X

Nr	DIP-links	CPLD-Pin	Bemerkung	DIP-rechts	CPLD-Pin	Bemerkung
1	CLK	1	GCK3	3V3	15,35	V_{ccINT}
2	IO2	2		VCC	26	V_{ccIO}
3	IO3	3		IO33	44	GCK2
4	IO4	5		IO32	43	GCK1
5	IO5	6		IO31	42	
6	IO6	7		IO30	41	
7	IO7	8		IO29	40	
8	IO8	12		IO28	39	
9	IO9	13		IO27	38	
10	IO10	14		IO26	37	
11	IO11	16		IO25	36	GTS1
12	IO12	18		IO24	34	GTS2
13	IO13	19		IO23	32	
14	IO14	20		IO22	31	
15	IO15	21		IO21	30	
16	IO16	22		IO20	29	
17	IO17	23		IO19	28	
18	TCK	11	JTAG	IO18	27	
19	TDO	24	JTAG	TDI	9	JTAG
20	GND	17,25,4		TMS	10	JTAG

Pin 33 ist an dem Taster SW1 fest verdrahtet, zudem ist Pin 33 auch GRS (GSR=Global Reset, GTS=Global Tri-State, GCK=Global Clock).

4 Strom und Spannungsversorgung

4.1 Spannungsversorgnung VCC_{INT}

Wie bei traditionellen ICs auch, befindet sich die Betriebsspannung (VCC_{INT}) des BitConnectors oben rechts in Bezug zu GND (unten links).

Die Betriebsspannung sollte 3.6V nicht übersteigen und 3V nicht unterschreiten. [1, S. 3]

Es wurde bewusst auf einen Spannungsregler verzichtet, um den zu treibenden Ausgangsstrom nicht zu begrenzen. Diese Entscheidung schafft zwar Flexibilität, verlagert aber die Verantwortung auf eine korrekte Spannungsversorgung auf die externe Peripherie.
An dieser Stelle ist noch erwähnenswert, dass Sie nach [2, S. 17] einen „ stromlosen“ I/O-Pin mit einer maximalen Spannung von VCC_{INT}+4V treiben dürfen.

4.2 Definition der I/O Spannung über VCC_{IO}

Neben der Versorgungsspannung VCC_{INT} existiert direkt darunterliegend der Pin VCC_{IO}(siehe Kapitel 3), welcher die Höhe der Pin-Ausgangsspannung definiert. Sie können die Spannung der Ausgänge bei einem h-Pegel auf 3,3V oder auf 2,5V setzen. Für eine Ausgangsspannung von 3,3 Volt, sollte VCC_{IO} zwischen 3V und 3,6V liegen. Wird dagegen eine Ausgangsspannung von 2,5V forciert, sollte VCC_{IO} zwischen 2,3V und 2,7V liegen [1, S. 3]

4.3 Ausgangskennlinie eines XC9536XL CPLDs

Sobald Sie einen Pin des BitConnectors mit einem Stromfluss belasten, in dem Sie beispielsweise eine LED ansteuern, sackt die Ausgangsspannung des entsprechenden I/O-Pins ab. Dieser Effekt geht auf den verwendeten CPLD zurück. Um nun eine Vorstellung zu erhalten, wie sich der treibende Strom zur Spannung verhält, hat der Autor einige stichpunktartig Messungen durchgeführt. Diese Messwerte sind in Abbildung 2 mittels Bezierinterpolation grafisch dargestellt. Die Messreihe wurde über ein Fluke 87 III , strom-/spannungsrichtig aufgenommen, VCC_{INT} und VCC_{IO} betrugen 3,3V, IO33 (CPLD Pin 44) war der (einzige) belastete Ausgang. Obwohl es sich bei der Messung um einen XC9536XL CPLD handelt, ist es naheliegend, dass der XC9572XL eine sehr ähnliche Charakteristik zeigt. Verwunderlich ist, dass die Messreihe teilweise signifikant von der Xilinx-Spezifikation [3, S. 6] abweicht. Dennoch wurde in Kapitel 4.4 auf die von Xilinx veröffentlichte Spezifikation zurück gegriffen.

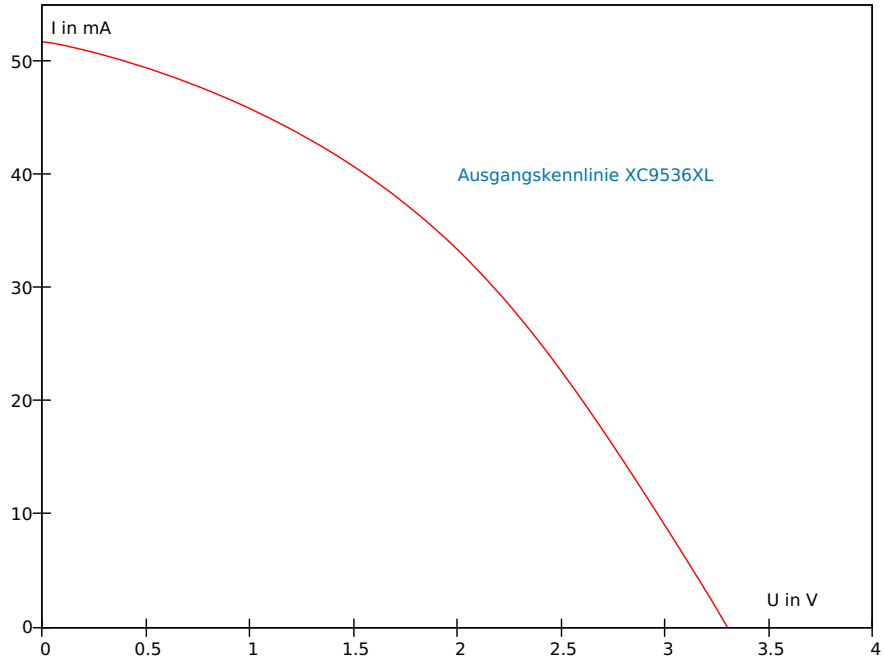

Abbildung 2: Ausgangskennlinie eines XC9536XL

4.4 Maximaler Stromfluss I_{max}

Wer nach einem konkreten Wert bezüglich der maximal zulässigen Stromaufnahme im Datenblatt [1] des XC95**72**XL oder des XC95**36**XL sucht, wird keine derartige Angabe finden. Zwar wird eine Abschätzung der zu erwartenden Stromaufnahme über eine Formel angegeben, welche die Frequenz, Anzahl der verwendeter Macrozellen u.s.w. mit einbezieht, jedoch bleibt ein maximal zulässiger Stromfluss I_{max} unerwähnt. Der Hauptgrund dafür dürfte wohl an der variablen Umgebungstemperatur und in der individuell eingesetzten Kühlung des ICs liegen. Eine plausible Annahme für derartige Werte und die damit einhergehende maximale Stromaufnahme wäre von Xilinx an dieser Stelle jedoch angebracht gewesen und sei es nur, um eine erste Abschätzung zu bekommen.

Um eine solche Abschätzung der maximalen Stromaufnahme I_{max} soll es nachfolgend gehen. Die Berechnung ist ein wenig müßig und führt über verschiedene Dokumentationen. Zuerst wird die zu I_{max} zugehörige Verlustleistung P_d über folgende Ungleichung berechnet.[2, S. 16]:

$$T_{jmax} < (\Theta_{JC} * P_d) + T_a$$

kurze Variablenbeschreibung:

- T_{jmax}: Maximaltemperatur in °C bezogen auf das IC Package (70 °C für Kunststoff) [4, S. 54]
- Θ_{JC}: Wärmeflusswiderstand in °C/$[W]$ zwischen der Chipoberfläche und der Oberfläche des Gehäuses [4, S. 53] (8.2 °C/W für 72/36 in VQ44)
- P_d: tatsächliche Leistung in [W]att, welche sich in Hitze äußert [2, S. 16]
- T_a: Umgebungstemperatur in °C (50 °C)

Um Θ_{JC} nicht aufwendig durch eine Messung bestimmen zu müssen, bietet Xilinx eine Datenbank an [5]. Für den XC95**72**XL/XC95**36**XL liegt dieser Wert bei 8.2 °C/W im VQ44 Package.

Zudem schlägt Xilinx für T_{jmax} einen Wert von 70 °C für Kunststoffgehäuse vor. Dieser Wert ist ebenfalls für beide möglichen Chipvarianten angebracht.

Die Umgebungstemperatur des Chips wird vorsichtig mit 50 °C veranschlagt.

Ausgehend von diesen Annahmen berechnet sich die Verlustleistung nach [2, S. 16] wie folgt:

$$\begin{gathered} 70°\mathrm{C} = (8.2°\mathrm{C}/W * P_d) + 50°\mathrm{C} \Rightarrow \\ 20°\mathrm{C} = (8.2°\mathrm{C}/W * P_d) \Rightarrow \\ 2,439W = P_d \end{gathered}$$

Weiter ist notwendig, die Verlustleistung in $P_d = P_{ddesign} + P_{dIO}$ aufzuschlüsseln. $P_{ddesign}$ repräsentiert hierbei die Leistung ohne eine Last, also die Leistung die der Chip selbst für sich beansprucht, ohne einen Ausgang zu treiben. Dieser Wert sollte gemessen werden und lautet für den BitConnector rund 40mW. Wird dieser Wert aus Sicherheitsgründen verdoppelt, ist von rund 80mW auszugehen.

Somit gilt:

$$2,439W = 80mW + P_{dIO}$$

Um schlussendlich auf den gesuchten Wert I_{max} zu kommen, wird folgende Formel nach [2, S. 16] herangezogen:

$$I_{max} * (VCC - 1,85V) = P_d - P_{ddesign}$$

Sicher fragen Sie sich, woher die 1,85V herstammen. Dieser Wert wurde über die Kennlinienfelder in [3, S. 6] abgelesen und sagt aus, dass bei einem Stromfluss von 30mA, die Spannung des entsprechenden Ausgangs auf 1,85V absinkt.
Daraus folgt:

$$I_{max} * (3,3V - 1,85V) = 2,439W - 0.08W \rightarrow$$
$$\underline{\underline{I_{max} \approx 1,63A}}$$

Der Wert I_{max} sagt aus, dass bei einer I/O-Spannung von 1,85V und einer Stromstärke von 30mA pro Ausgang, eine Gesamtstromstärke von 1,63A erreicht wird. Hierbei liegt die Chiptemperatur $\leq$ der als kritisch angenommenen 70°C. Auffällig ist, dass die 1,63A in dieser Konstellation im Bitconnector nie erreicht werden können, da dieser nur 34 mögliche Ausgänge bietet. Mit den getroffenen Annahmen ist jedoch die Versorgung von $\frac{1,63A}{0,03} \approx 54$ Ausgänge möglich.

Bitte verlassen Sie sich nicht auf die angegebene Berechnung und überprüfen Sie auch die Quellen. Sollten Sie einen Fehler finden, wäre ich Ihnen sehr dankbar, wenn Sie mich diesbezüglich kontaktieren.

5 Hardware Entprellung

Gelegentlich ist es sehr wichtig, eine getaktete, sequenzielle Schaltung schrittweise, manuell über einen Taster zu durchlaufen. Leider erzeugen so ziemlich alle handelsüblichen Taster ein Prellen. Ein solches Prellen führt zu einer undefinierten Abfolge von Signalimpulsen. Da Sie bei CPLDs prinzipiell mit Ihren Flip-Flop-Ressourcen sehr haushalten müssen, wurde für den BitConnector eine Hardwareentprellung vorgesehen. Sofern Ihre Ausführung mit dem 74LVC2G14 IC bestückt ist, ist der Taster SW1 auf dem Board automatisch entprellt. Sollte dies nicht der Fall sein, so können Sie auf Grundlage der nachfolgenden Erläuterungen, eine eigene Hardwareentprellung nachschalten.

5.1 Prinzip der Hardwareentprellung

Da die vom Menschen erzeugten Frequenzen der Tastendrücke im Vergleich zum Prellen des Tasters sehr niedrig sind, wurde ein Tiefpassfilter in Form eines RC-Gliedes dem Taster nachgeschaltet. Dieser Tiefpass filtert die hochfrequenten Anteile des Prellens aus dem Tastendruck heraus. Das Prellen ist verschwunden. Es entsteht allerdings ein neues Problem. Da ein RC-Glied naturgemäß aus einem Widerstand und einem Kondensator besteht, durchläuft dieser Kondensator eine Ladespannung, die dem natürlichen Logarithmus sehr nahe kommt (siehe beispielsweise [6]). Eine digitale Schaltung auf CMOS-Basis benötigt jedoch eine steile Signalflanke, also beispielsweise einen Signalwechsel von 0V auf direkt 3,3V. Andernfalls besteht die Gefahr, dass die Schaltung in einen undefinierten Zustand abrutscht. Auf Transistorebene kann dieser undefinierte Zustand sehr gefährlich sein, da eine Transistorkonstellation, die es theoretisch so nicht geben sollte, plötzlich anfangen kann, den Strom zu leiten. Zudem werden Sie immer noch keinen auswertbaren Tastenimpuls erwarten können. Es ist daher zwingend anzuraten, diese undefinierten Signalzustände in digitalen Schaltungen zu vermeiden. Dies geschieht hier mit Hilfe eines Schmitt-Triggers. Der Schmitt-Trigger soll verkürzt dargestellt, den undefinierten Spannungsbereich auf einen definierten high- oder

low- Pegel zurück bringen. Die charakteristische Hysterese eines Schmitt-Triggers ist dafür verantwortlich, den undefinierten Spannungsbereich zu vermeiden. Sie können sich den Schmitt-Trigger in diesem Fall als den „eigentlichen Schalter“ vorstellen, der am Ausgang immer einen steilen Pegel schaltet, während er am Eingang eine Kondenstorkennlinie „sieht“.

6 Entwicklungsumgebung

Momentan gibt es noch keine vollumfängliche und freie [7] Entwicklungsumgebung für Xilinx CPLDs, weswegen der Autor auf proprietäre Software der Firma Xilinx zurückgreifen muss. Fraglicherweise bietet Xilinx in seiner aktuellen Entwicklungsumgebung „Vivado“ keine Unterstützung für CPLDs an, siehe [8, S. 17]. Dennoch ist es möglich, mit der älteren Design Suite „ISE“, Xilinx CPLDs problemlos zu konfigurieren. Mit einer Registrierung bei Xilinx erhalten Sie eine kostenlose Lizenz, mit derer Sie die ISE-Demo-Version auf eine Webpack-Version [9] freischalten können. Diese Lizenz ist ausreichend, um alle CPLDs der XC9500 Serie ohne Einschränkungen zu konfigurieren.

7 Die JTAG Schnittstelle via 6 Pin IDC-Header

Für die Übertragung der HDL Konfigurationsdateien auf den BitConnector ist ein 6 Pin IDC-Stecker (JTAG Schnittstelle) im 2.54mm Rastermaß vorgesehen. Die 4 aktiven JTAG-Leitungen zur Datenübertragung sind:

1. Pin TCK = Test Clock
2. Pin TDI = Test Data In
3. Pin TDO = Test Data Out
4. Pin TMS = Test Mode Select

Abbildung 3: Draufsicht JTAG Pin-Header (6 way IDC male connector)

Die Abbildung 3 zeigt die Draufsicht dieses Steckers, welcher auf dem Board verlötet ist. Innerhalb dieser Abbildung erkennen Sie eine „1“, die den ersten Pin markiert. Dieser Pin ist zur Orientierung dem JTAG Test-Takt zugeordnet. Um die Anfertigung eines eigenen Programmierkabels zu erleichtern, wird nachfolgend auch das passende Datenkabel illustriert. Ich möchte Sie ermutigen, die Anfertigung eines eigenen Kabels nicht zu fürchten. Die Minimalausrüstung sehen sie in Abbildung 4

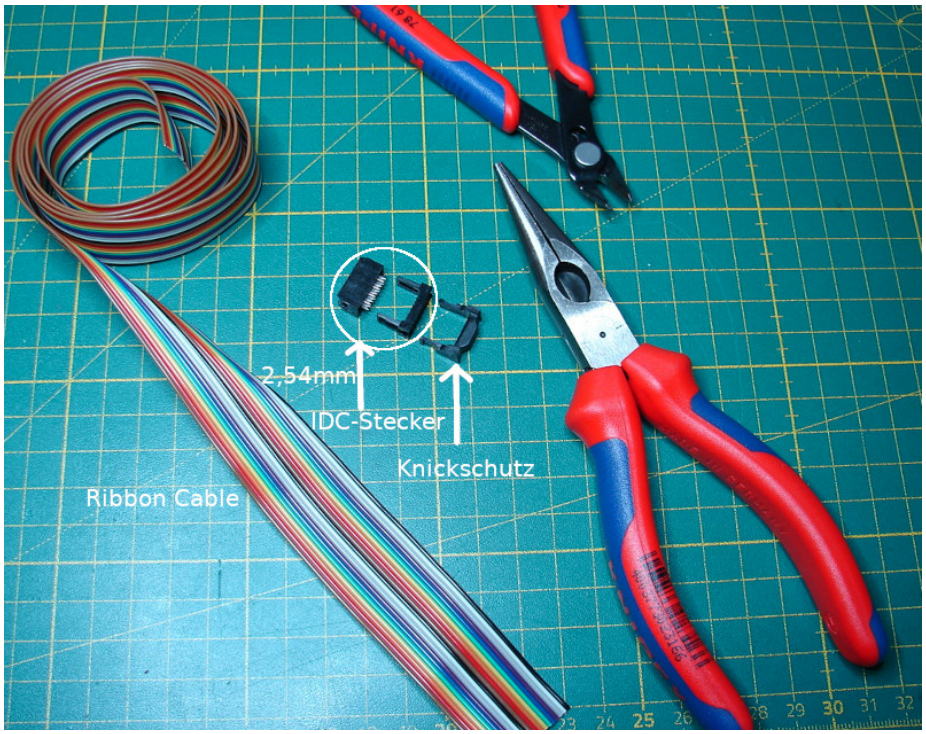

Abbildung 4: Werkzeug und Ribbon Cable + IDC-Stecker zum selber crimpen

Da es mehrere Programmiergerät gibt, existieren somit mehrere Stecker-Möglichkeiten und Adapter. In Abbildung 5 erkennen Sie ein mögliches Datenkabel vom Bit-Connector zum Programmiergerät. Anders als in Abbildung 3, ist der 6 Pin Connector diesmal als Buchse ausgeführt und somit gespiegelt.
Bei dem HW-RIBBON14 female connector hat sich Xilinx für eine eher exotische Buchse entschieden. Die Kette des Datenkabels sieht abstrakt zusammen gefasst folgendermaßen aus:

PC<->USB-Kable<->Programmierer<->Programmierkabel<->BitConnector

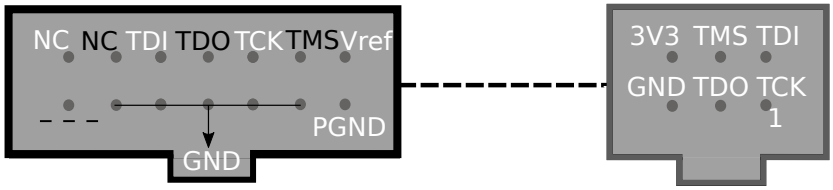

Abbildung 5: Programmierkabel HW-RIBBON14 zu 6 Pin IDC female

Falls Sie kein passendes Kabel zur Hand haben, können Sie die herausgeführten JTAG Pins von den beiden Stiftleisten über ein Steckbrett abgreifen(rote Markierung).

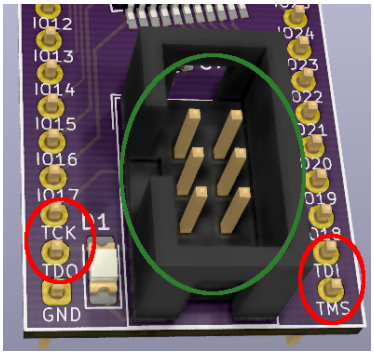

Abbildung 6: Die JTAG Schnittstelle, einmal rot umkreist als Stiftleisten und grün umkreist als IDC Header ausgeführt

Sofern Sie mehrere BitConnectoren auf einer Platine gesockelt haben, bieten sich die Pins der Stiftleisten ebenfalls an, um die CPLDs mit nur einer herausgeführten Schnittstelle sequenziell zu konfigurieren.

7.1 JTAG Programmiergerät

Für die Ansteuerung der JTAG-Schnittstelle wird ein Programmiergerät benötigt. Hier gibt es mehrere Lösungen mit unterschiedlichen Funktionsumfang. Verbreitet sind 4 Versionen des Xilinx Platform Cable USB I/II. Dabei entfallen bereits 3 Versionen auf das Platform Cable I:

- **DLC9G** (neuste Version bezüglich Platform Cable I)
- **DLC9LP** (Vorgänger des DLC9G)
- **DLC9** (älteste Version)

Zu diesen 3 Versionen schreibt Xilinx u.a.: „*The DLC9G and legacy DLC9LP cable models draw less than 100 mA from the host USB port. The legacy DLC9 cable model requires 230 mA to operate in USB 2.0 Hi-Speed mode or 150 mA to operate in USB 2.0/1.1 full-speed mode.*“ [10, S. 2]

Das Xilinx Platform Cable USB II wird momentan unter der Versionsbezeichnung „**DLC10**“ gehandelt. Die hohen Verbreitung dieser Programmiergeräte hat dazu beigetragen, dass es selbige auch als preisgünstige Nachbauten von anderen Herstellern zu einem Zehntel des Preises gibt.

Abbildung 7: oben: XUP USB-JTAG Programming Cable
unten: Platform Cable USB II DLC10

Als Alternative zum „Platform Cable I/II“ existiert eine kleine und günstigere Variante aus dem „Xilinx University Program“ dass die Bezeichnung „XUP USB-JTAG Programming Cable“ trägt. Sie können es nachfolgend in Abbildung 7 erkennen. Allerdings scheint diese Variante mit dem heutigen Preisverfall des Platform Cables I/II wenig attraktiv zu sein. Der Vollständigkeit halber soll an dieser Stelle noch das **DLC7**/Parallel Cable IV erwähnt werden, dass jedoch in der neuen Entwicklungsumgebung von Xilinx/Vivado nicht mehr unterstützt wird [11] und aufgrund der „ausgestorbenen“ parallelen Schnittstelle bei Anwendern ebenfalls nur mehr selten anzutreffen sein dürfte. Wenn Sie keinen Wert auf Zusatzfunktionen wie „SPI PROM Support, ChipScope oder AES-Cryptographic“ legen, sollte die Wahl des Programmers eine untergeordnete Rolle spielen.

7.2 Offene JTAG Hardware-/Softwarelösungen für Xilinx

Es gibt einige Projekte, die sich darum bemühen, alternative und offene JTAG Hardware-/Softwarelösungen für Xlinix Produkte anzubieten. Um ein CPLD/FPGA zu konfigurieren, wird ein Mikrocontroller (JTAG-Programmiergerät) benötigt, der den JTAG Standard „IEEE 1149.1“ und eventuelle herstellerspezifische Abweichungen implementiert. Um mit diesem Mikrocontroller kommunizieren zu können, wird weiter eine PC-Software (JTAG Client) benötigt, welche die HDL synthetisierte Konfiguration (im XSVF Format) an diesen Mikrocontroller überträgt. Der Mikrocontroller nimmt letztlich über das vom PC aus übertragene XSVF Format die Konfiguration des CPLD/FPGA mittels JTAG vor.

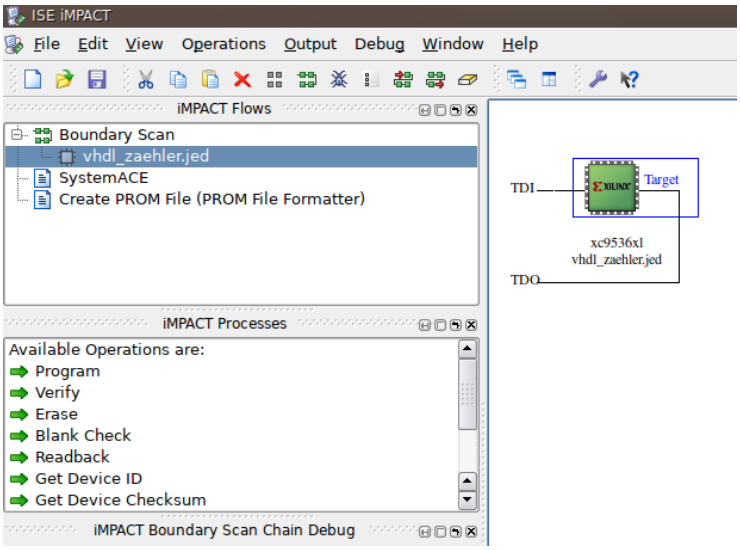

Abbildung 8: Das proprietäre iMPACT überträgt und verifiziert die HDL Konfiguration (über ein JTAG Programmiergerät) auf dem CPLD oder FPGE

Um aber auf die offenen Alternativen hinzuweisen, soll hier die Lib(X)SVF von Clifford Wolf[12] Erwähnung finden. Lib(X)SVF ist eine C-Bibliothek zur Implementierung/Umsetzung von JTAG-Programmiergeräten mittels XSVF/SVF Format. Eine konkrete Implementierung ist beispielsweise das „clujtag“ Projekt, wobei clujtag [13] die Firmware eines einfachen AVR basierten Programmiergeräts bildet und clujtag-client [14] eine kommandozeilenbasierte Alternative zu Xilnix iMPACT darstellt. In welchen Umfang diese und andere Projekte heute tragfähig sind, kann der Autor bisher nicht einschätzen.

Literatur

[1] *XC9572XL High Performance CPLD - Product Specification*. DS057. v2.0. Xilinx. Apr. 2007.

[2] *CPLD I/O User Guide*. UG445. v1.2. Xilinx. Jan. 2014.

[3] *I/V Curves for Xilinx FPGA and CPLD Families*. 1-800-255-7778. v1.1. Xilinx. Mai 2001.

[4] *Device Package User Guide*. UG112. v3.7. Xilinx. Sep. 2012.

[5] *Package Thermal Data Query*. `https://www.xilinx.com/cgi-bin/thermal/thermal.pl`. accessed: 14-04-2021.

[6] *Natürlicher Logarithmus für die Kondensatoraufladung*. `https://et-tutorials.de/5703/naturlicher-logarithmus-fur-die-kondensatoraufladung/`. accessed: 14-04-2021.

[7] *Freie Software. Was ist das?* `https://www.gnu.org/philosophy/free-sw.de.html`. accessed: 14-04-2021.

[8] *Vivado Design Suite UserGuide - Release Notes, Installation, and Licensing*. UG973. v2018.3. Xilinx. Dez. 2018.

[9] *Download ISE WebPACK Design Software*. `https://www.xilinx.com/products/design-tools/ise-design-suite/ise-webpack.html`. accessed: 14-04-2021.

[10] *Platform Cable USB*. DS300. v3.2. Xilinx. Mai 2008.

[11] *Parallel Cable IV/Vivado*. `https://www.xilinx.com/support/answers/54136.html`. accessed: 14-04-2021.

[12] Clifford Wolf. *Lib(X)SVF: A library for implementing SVF and XSVF JTAG players*. `http://www.clifford.at/libxsvf/`. accessed: 14-04-2021.

[13] Alexey Avdyukhin. *Very simple JTAG programmer based on AVR microcontroller with hardware USB*. `https://github.com/ClusterM/clujtag-avr`. accessed: 14-04-2021.

[14] Alexey Avdyukhin. *Client for AVR-based JTAG programmer*. `https://github.com/ClusterM/clujtag-client`. accessed: 14-04-2021.