Fakultät für Physik
Prof. Dr. M. Weber, Dr. K. Rabbertz
L. Ardila Perez, P. Pfistner, D. Savoiu, B. Siebenborn

20. Dezember 2017

Einführung in die Quartus II/Quartus Prime Lite Software und die Altera-Boards

Inhaltsverzeichnis

Vorbemerkungen

Bei den drei Übungen, die sich mit HDL (**H**ardware **D**escription **L**anguage) beschäftigen, bei uns in der Variante des heute meist üblichen **V**ery High Speed Integrated Circuit HDL (VHDL), ist es wichtiger denn je, zu Hause vorzuarbeiten. Ohne sorgfältige Vorbereitung wird es nicht möglich sein, in dem gegebenen Zeitrahmen auch nur die Pflichtaufgaben durchzuführen. Sie sollten sich zumindest mit der Entwicklungsumgebung vertraut machen und sich genau überlegen, wie Sie die folgenden Aufgaben programmieren wollen. Unterschätzen Sie nicht das Problem, dass sämtliche Befehle parallel bearbeitet werden! Die bekannte sequentielle Programmierlogik bringt einen häufig nicht weiter, sondern irritiert zusätzlich. Am besten schreiben Sie den Quelltext für Ihre Lösungen bereits zu Hause und kompilieren ihn. Während der Übungen werden wir dann noch genug damit zu tun haben, verbliebene Probleme zu lösen. Diese kurze Einführung ist zum schnellen Nachschlagen da und reicht auf keinen Fall alleine zur Vorbereitung aus!

1 Software

1.1 Installation

1. Laden Sie die Quartus II Web Edition (der neuere Name ist Quartus Prime Lite) Software für Windows oder Linux von Altera herunter (Achtung! Das können einige GB sein!). Die gegenwärtige offizielle Download-Seite von Altera lautet:
   `https://www.altera.com/downloads/download-center.html`
   Allerdings wird für die DE1-Boards unbedingt Cyclone II Unterstützung benötigt, die bei den letzten Versionen leider herausgefallen ist. Man kann nun entweder unten auf dieser Seite den „Select by Device“ Software Selector bemühen, oder es gleich über die Seite
   `http://dl.altera.com/devices/` versuchen und aus der Versions-Device-Matrix eine passende Version auswählen. Derzeit trägt die letzte passende Version die Nummer 13.0sp1 und sollte via
   `http://dl.altera.com/13.0sp1/?edition=web`
   herunterladbar sein.

   Ursprünglich musste man sich mit Namen und Adresse registrieren, wobei die Benutzung der Web-Edition allerdings frei ist; es wird keine Lizenz benötigt.

2. Installieren Sie zuerst die heruntergeladene exe Datei (Windows). Bei „Setup Type“ können Sie „Custom“ wählen und durch **Abwahl** der Unterstützung von ARRIA GX, ARRIA II GX, Cyclone III und Cyclone IV noch etwas Festplattenplatz sparen. Die Komponenten Cyclone und Cyclone

II Support werden jedoch unbedingt benötigt. Für die älteren Boards, falls Sie eines ausgeliehen haben, sollten auch die MAX und FLEX Chips unterstützt werden.

3. Eventuell erscheint eine Windows-Meldung über eine fehlgeschlagene „Microsoft Kompatibiltäts-Prüfung“ - Diese Meldung können Sie ignorieren und die Installation fortsetzen.

4. Beim ersten Start müssen Sie eine Benutzeroberfläche wählen. Verwenden Sie bitte den Quartus II Stil. Anschließend werden Sie u.U. zwar nach einer Lizenz gefragt, die Quartus II Web Edition läuft jedoch auch ohne Lizenz. Der angebotene Funktionsumfang dieser Version reicht für unsere Anforderungen vollkommen aus.

5. Anschließend können Sie mit Ihrer Quartus II Version arbeiten.

1.2 Benutzung der Software

Die Benutzung der Software soll anhand eines einfachen Beispiels schrittweise erläutert werden. Die Beispielaufgabe besteht aus einem einfachen ODER-Gatter. Wenn also mindestens einer von zwei Tastschaltern gedrückt wird, soll eine LED leuchten. Sie finden im folgenden die Instruktionen für VHDL.

1. Wählen Sie **File** → **New Project Wizard**. Dann müssen Sie ein Arbeitsverzeichnis wählen. Sie sollten am besten für jedes Projekt, also jede Teilaufgabe ein eigenes, neues Verzeichnis erstellen. Geben Sie als Projektnamen und als Namen für die Top-Level-Entity „orgate“ ein.
   → **Finish**

2. Nun müssen Sie die Hauptprogrammdatei erstellen: **File** → **New** → **VHDL File** → **OK**

3. Diese .vhd Datei speichern Sie nun unter dem Namen orgate.vhd: **File** → **Save As**.

4. Schreiben Sie in die Datei den folgenden Programmcode (VHDL unterscheidet nicht zwischen Groß- und Kleinschreibung):

   VHDL:

```
1   library ieee; use ieee.std_logic_1164.all; use
2   ieee.numeric_std.all;

4   entity orgate is port( key: in std_logic_vector( 1 downto 0 );
5   ledr: out std_logic ); end;

7   architecture behave of orgate is begin

9   ledr <= key(1) or key(0);

11  end;
```

5. Klicken Sie **Assignments** → **Device** und wählen Sie den verwendeten Chip aus (s. Abschnitt 2.1). Dazu muss die entsprechende Chip-Familie (z.B. Cyclone II) und der genaue Typ (z.B. EP2C20F484C7) angegeben werden.

6. Starten Sie den Compiler (**Processing** → **Start Compilation**).

7. Starten Sie den Pin Planner (**Assignments** → **Pin Planner**). Nun werden sämtliche Anschlüsse des Chips dargestellt.

8. Suchen Sie sich zwei Pins für die Nodes Key[0] und Key[1], s. Abschnitt 2.2.4 und 2.3.4 (z.B. Pins R21 und R22 beim DE1 Board, Pins 80 und 81 beim UP1/UP2-Board bzw. Pin 48 und 49 beim UP3 Board), und einen Pin für die LED, s. Abschnitt 2.2.2 und 2.3.2 (z.B. R20 beim DE1 Board), und weisen Sie diese Pins ihren Nodes zu. Schliessen Sie den Pin Planner wieder.

9. Kompilieren Sie das Programm mit der aktuellen Pinbelegung erneut (**Processing** → **Start Compilation**).

10. Das Design kann nun simuliert werden, was relativ aufwändig ist und einen nicht immer weiter bringt. Es soll aber dennoch hier gezeigt werden. Starten Sie dazu das Simulator Tool (**Processing** → **Simulator Tool**)

11. Um mit dem Waveform Editor ein Eingangssignal zu definieren klicken Sie auf **Open**. Klicken Sie in die linke Spalte mit der rechten Maustaste → **Insert** → **Insert Node or Bus** → **Node Finder** → **List** → >> → **OK** → OK.

12. Jetzt sollten alle verfügbaren Inputs und Outputs im Waveform Editor eingetragen sein. Sie können nun in der linken Spalte für die beiden Eingänge KEY[0] und KEY[1] ein Signal wählen, z.B. rechte Maustaste → **Value** → **Clock** um ein periodisches Ein- und Ausschalten zu simulieren.

13. Schließen Sie das Fenster, speichern Sie die eingegebene Waveform mit Hilfe des sich öffnenden Dialogfensters und klicken sie **Start** im Simulator Tool.

14. Wenn Sie jetzt wieder auf **Open** klicken, sollten Sie die simulierte Ausgabe inklusive Verschiebungen durch Laufzeiten im Inneren des Chips sehen.

15. Um das Programm (mit oder ohne vorherige Simulation) auf den Chip zu übertragen, starten Sie den Programmer (**Tools** → **Programmer**). Unter **Hardware Setup** wählen Sie den verwendeten Byteblaster, s. Abschnitt 2.1, und machen ein Häkchen bei **Program/Configure** → **Start**.

16. Das Programm wird sofort nach der Übertragung auf dem Board gestartet.

17. Wir stellen fest: Das Programm arbeitet nicht wie ein ODER Gatter! Der Grund: Die Tastschalter ergeben beim Drücken den logischen Zustand GND, s. Abschnitt 2.2.4. Man hätte also die Eingänge noch invertieren müssen:
VHDL:

```
15      key(0);
```

Bei den UP-Boards sind außerdem die LEDs active low, leuchten also bei logischem low pegel. Hier müsste das Kommando also
VHDL:

```
15      ledr <= not( not key(1) or not key(0) );
```

heißen.

Nun wollen wir noch die Benutzung von Subdesigns üben. Dazu benutzen wir folgendes Beispiel: Eine LED soll leuchten, wenn mindestens einer von drei Tastschaltern gedrückt wird. Dazu soll (unsinnigerweise) das obige Subdesign verwendet werden.

1. Beginnen Sie ein neues Projekt mit Namen orgate2 und speichern Sie es in ein neues Verzeichnis.

2. Schreiben Sie in die neu angelegte orgate2.vhd:
VHDL:

```
1       library ieee; use ieee.std_logic_1164.all; use
2       ieee.numeric_std.all;

4       entity orgate2 is port( key: in std_logic_vector( 2 downto 0 );
5       ledr: out std_logic ); end;

7       architecture behave of orgate2 is

9       component orgate port ( key: in std_logic_vector( 1 downto 0 );
10      ledr: out std_logic ); end component; signal or1: std_logic;

12      begin

14      c0: orgate port map( key => key( 1 downto 0 ). ledr => or1 ); c1:
15      orgate port map( key => key(2) & or1. ledr => ledr );

17      end;
```

3. Kopieren Sie die Datei orgate.vhd von vorhin in das orgate2 Verzeichnis und öffnen Sie sie in Quartus.

4. Klicken Sie **File → Create/Update → Create VHDL Include File**
5. Klicken Sie **Project → Add current File to Project**
6. Das Subdesign ist fertig eingebunden. Sie können das Projekt jetzt kompilieren, Pins zuweisen, kompilieren und auf das Board programmieren.

Um sich weitergehend über die Sprache VHDL zu informieren, betrachten Sie folgende Links zu Tutorials und Bücher über VHDL, welche im Internet oder als E-Book auf der Webseite der Bibliothek zu finden sind. Empfehlenswert ist die VHDL Einführung von der Technischen Universität Hamburg-Harburg.

- Kurze Einführung in VHDL von der Technischen Universität Hamburg-Harburg, wichtig sind Kapitel 1-8,12-13,15
  `http://www.tuhh.de/t3resources/ict/dateien/Lehre/Hardware-Praktikum/VHDL_Einfuehrung.pdf`
- Auch eine kurze Einführung, diesmal von der Interstaatlichen Hochschule für Technik Buchs. Alles ist mehr oder weniger relevant für die Übungen. Übersehen Sie nicht das Kapitel 12 über Hierarchie fast am Ende.
  `http://wiki.ntb.ch/infoportal/_media/software:vhdl:vhdlscript.pdf`
- Im folgenden Buch ist das Kaptitel 4 eine Einführung in VHDL. In den anderen Kapiteln werden aber digitale Schaltungen meist mit VHDL Beispiel beschrieben. So sind in Kapitel 8.6. Zähler beschrieben und in Kapitel 9.4 Zustandsmaschinen (Finite-State-Maschine / FSM)
  Mohammed Ferdjallah: Introduction to digital systems : modeling, synthesis, and simulation using VHDL
- Noch einmal eine etwas ausführlichere Einführung in VHDL, ab Kapitel 3
  William Kafig: VHDL 101 : everything you need to know to get started

Außerdem sei auf die entsprechenden Kapitel im Vorlesungsskript und auf die Quartus II Hilfe verwiesen.

2 Hardware

2.1 Vorbemerkungen

Zur Bearbeitung der VHDL-Übungsaufgaben stehen die Altera-Boards DE1 (**D**evelopement and **E**ducation) zur Verfügung, von denen jeweils eines pro Gruppe an Sie ausgeliehen werden kann. Die älteren UP1-, UP2- und UP3-Boards (**U**niversity **P**rogram) mit unterschiedlichen technischen Spezifikationen können Sie, falls gewünscht, ebenfalls zur Vorbereitung mit nach Hause nehmen. Die UP1- und UP2-Boards sind mit einem MAX7000S EPM7128SLC84-7 Chip ausgestattet. Dieser CPLD (**C**omplex **P**rogrammable **L**ogic **D**evice) besitzt nichtflüchtigen Programmspeicher. Das bedeutet, dass auch ohne Spannungsversorgung die Programmierung erhalten bleibt. Die UP3 Boards mit dem Cyclone EP1C6Q240C8 und die DE1 Boards mit dem Cyclone2 EP2C20F484C7 haben dagegen komplexere FPGAs (**F**ield **P**rogrammable **G**ate **A**rray), die ihre Programmierung bei jedem Ausschalten verlieren.

Trotz dieser Unterschiede können aber problemlos alle Übungsaufgaben mit jedem Board durchgeführt werden. Dieselben tdf Design-Dateien funktionieren auf allen Boards. Lediglich auf die unterschiedliche Pinbelegung muss Rücksicht genommen werden; der Compiler erledigt den Rest. Zur Ein- und Ausgabe von Daten stehen Tastschalter, DIP-Schalter, LEDs und Siebensegmentanzeigen zur Verfügung, die zum Teil auf Ansteck-Platinen untergebracht sind.

Um das Programm vom Rechner auf das Board zu übertragen, wird das DE1 Board mit einem USB-Kabel an den Computer angeschlossen. Dazu muss der USB-Treiber im Unterverzeichnis `drivers\usb-blaster` der Quartus II Software installiert werden. Das Board wird dann als ByteBlaster erkannt. Für die normale JTAG-Programmierung (**J**oint **T**est **A**ction **G**roup) muss der Schiebeschalter `SW12` auf `RUN` stehen und im Quartus Programmer auf Start geklickt werden. Diese Programmierung geht aber ohne Spannungsversorgung sofort verloren.

Bei der AS-Programmierung (**A**ctive **S**erial) wird das Programm dagegen in einen Flash Speicher geschrieben, und bei jedem Neustart von einem speziellen Baustein (EPCS4) automatisch in den FPGA geladen. Dazu muss zunächst unter **Assignments → Device → Device&Pin Options → Configuration** das Configuration Device **EPCS4** gewählt und neu kompiliert werden. Dann wird im Programmer

der Mode **Active Serial** gewählt und mit **Add File**... die neu erzeugte Programmdatei (Dateiname .pof statt .sof) benutzt. Der Schalter `SW12` muss dabei auf `PROG` stehen. Die AS Programmierung wird nun bei jedem Neustart des Boards verwendet. Führt man eine normale JTAG Programmierung durch, wird das AS Programm bis zum nächsten Neustart überschrieben.

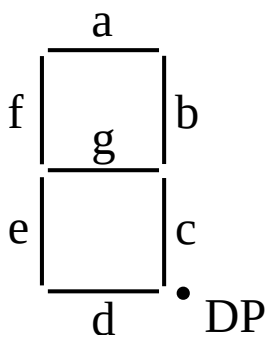
Für Details zur Programmierung der UP-Boards, lesen Sie bitte die Abschnitte 2.3.6 und 2.3.7.
Eine gute Seite um mehr über die Altera-Boards zu erfahren ist:
`http://www.ece.gatech.edu/users/hamblen/`
Technische Details und Beispielprogramme für die UP - beziehungsweise DE-Boards befinden sich auf den Seiten:
`http://www.ece.gatech.edu/users/hamblen/ALTERA/altera.htm`
`http://www.ece.gatech.edu/users/hamblen/UP3`
`http://www.ece.gatech.edu/users/hamblen/DE1`

2.2 Pinbelegung des DE1 Boards

2.2.1 Siebensegmentanzeigen

Das DE1 Board hat vier Siebensegmentanzeigen, die von rechts nach links mit `HEX0` bis `HEX3` bezeichnet sind. Die Dezimalpunkte sind hier nicht angeschlossen. Alle LEDs sind *active low*, d.h. sie leuchten beim logischen Zustand GND.

Die einzelnen Segmente sind an die folgenden Pins des FPGAs angeschlossen:



Segment	DE 1 Board, Pin Nr.			
	HEX3	HEX2	HEX1	HEX0
a	F4	G5	E1	J2
b	D5	G6	H6	J1
c	D6	C2	H5	H2
d	J4	C1	H4	H1
e	L8	E3	G3	F2
f	F3	E4	D2	F1
g	D4	D3	D1	E2
DP	–	–	–	–

2.2.2 Weitere LEDs

Das DE1-Board besitzt neben den Siebensegmentanzeigen noch 10 rote und 8 grüne LEDs. Diese LEDs sind *active high*, das bedeutet, sie leuchten beim logischen Zustand VCC.

LED Nr.	DE1 Board, Pin Nr.	
	rote LEDs	grüne LEDs
0	R20	U22
1	R19	U21
2	U19	V22
3	Y19	V21
4	T18	W22
5	V19	W21
6	Y18	Y22
7	U18	Y21
8	R18	–
9	R17	–

2.2.3 Ampelschaltung

Für die auf Übungsblatt 10 zu programmierende Ampelschaltung sind auf der Erweiterungsplatine 12 bunte LEDs in Form einer Kreuzung mit Ampel angeordnet. Die Erweiterungsplatine wird mit einem Flachbandkabel an den Steckplatz `GPIO1` angeschlossen. Die LEDs sind *active low*.

2.2.4 Schalter

Zur Eingabe von Daten besitzt das DE1 Board zehn Schiebeschalter (`SW0 - SW9`), vier Tastschalter (`KEY0 - KEY3`) sowie acht DIP-Schalter auf der Erweiterungsplatine (angeschlossen an `GPIO1`). Die Taster sind bereits entprellt und geben im gedrückten Zustand ein logisches GND ab. Die Schiebeschalter und DIP Schalter sind nicht entprellt. Die untere Schalterstellung (bei der DIP Reihe die Stellung `ON`) entspricht dem logischen GND.

Ampel-LEDs	DE1 Board, Pin Nr.
Rot 1	E15
Gelb 1	G15
Grün 1	H13
Rot 2	E14
Gelb 2	H14
Grün 2	H12
Rot 3	F20
Gelb 3	D20
Grün 3	C20
Rot 4	D14
Gelb 4	D16
Grün 4	C18

Schiebeschalter	
SW0	L22
SW1	L21
SW2	M22
SW3	V12
SW4	W12
SW5	U12
SW6	U11
SW7	M2
SW8	M1
SW9	L2

Tastschalter	
KEY0	R22
KEY1	R21
KEY2	T22
KEY3	T21

DIP-Schalter	
DIP1	G17
DIP2	H17
DIP3	J15
DIP4	H18
DIP5	N22
DIP6	N21
DIP7	P15
DIP8	N15

2.2.5 Systemuhr und Frequenzeingang

Um einen Systemtakt zu erhalten, werden die Pins der internen Systemuhr (Clock) benötigt. Das DE1 besitzt drei Taktgeneratoren mit verschiedenen Frequenzen. Es wird empfohlen, den 50 MHz Eingang zu verwenden. Außerdem gibt es einen externen Eingang für den Frequenzgenerator. Der Frequenzgenerator ist dabei so einzustellen, dass er weder negative Spannungen noch Spannungen über 3.3 V ausgibt, da sonst der Hauptchip zerstört wird! **Vor der Benutzung des externen Eingangs ist auf jeden Fall ein Betreuer hinzuzuziehen!**

DE1 Board, Clock inputs			
50 MHz	24 MHz	27 MHz	extern
L1	A12	D12	P18

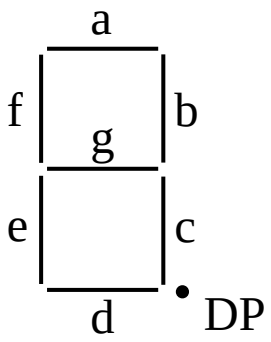
2.3 Pinbelegung der UP Boards

2.3.1 Siebensegmentanzeigen

Bei den UP1- und UP2-Boards befinden sich die Siebensegmentanzeigen fest verdrahtet auf der Hauptplatine, bei den UP3-Boards auf der Erweiterungsplatine. Die LEDs sind *active low* und der Dezimalpunkt kann separat angesteuert werden.

2.3.2 Weitere LEDs

Bei den UP Boards befinden sich jeweils acht rote LEDs auf den Erweiterungsplatinen. Hier sind die LEDs *active low*!.



Segment	linke Anzeige (HEX1)			rechte Anzeige (HEX0)		
	UP1	UP2	UP3	UP1	UP2	UP3
a	58	58	141	69	69	126
b	60	60	139	70	70	132
c	61	61	143	73	73	137
d	63	63	160	74	74	128
e	64	64	158	76	76	133
f	65	65	138	75	75	127
g	67	67	156	77	77	125
DP	68	68	140	79	79	135

LED Nr.	Pin Nr.		
	UP1	UP2	UP3
0	37	37	197
1	39	40	199
2	44	44	200
3	45	46	201
4	50	49	202
5	52	51	203
6	54	54	204
7	56	56	205

2.3.3 Ampelschaltung

Die LEDs für die Ampelschaltung befinden sich wie beim DE1 Board auf der Erweiterungsplatine. Auch hier sind die LEDs *active low*.

Ampel-LEDs	Pin Nr.		
	UP1	UP2	UP3
Rot 1	20	20	163
Gelb 1	17	17	165
Grün 1	15	15	167
Rot 2	22	22	168
Gelb 2	24	25	180
Grün 2	29	28	175
Rot 3	31	30	177
Gelb 3	33	33	181
Grün 3	35	35	169
Rot 4	12	12	166
Gelb 4	11	11	164
Grün 4	10	10	162

2.3.4 Schalter

Die UP Boards besitzen 16 DIP Schalter in zwei Bänken auf den Erweiterungsplatinen. Die Schalterstellung ON bedeutet hier den logischen Zustand GND. Auf dem UP3 Board sind außerdem Taster und DIP Schalter vorhanden, die in Abschnitt 2.3.7 beschrieben werden. Bei den UP1/2 Boards können zusätzlich Taster und Schalter mit kurzen Kabeln angesteckt werden (s. A. 2.3.6).

2.3.5 Systemuhr und Frequenzeingang

Um einen Systemtakt zu erhalten, werden die Pins der internen Systemuhr (Clock) benötigt, die Frequenzen unterscheiden sich dabei von denen des DE1-Boards. Aufgaben mit Zeitschleifen müssen dementsprechend angepasst werden.

Schalter Nr.	DIP-Bank 1			DIP-Bank 2		
	UP1	UP2	UP3	UP1	UP2	UP3
1	16	16	220	40	39	217
2	18	18	219	41	41	216
3	21	21	218	46	45	215
4	25	24	221	48	48	206
5	27	27	222	49	50	207
6	28	29	223	51	52	208
7	30	31	224	55	55	213
8	34	34	225	57	57	214

UP1		UP2		UP3	
Clockfrequenz	Pin	Clockfrequenz	Pin	Clockfrequenz	Pin
25.175 MHz	83	25.175 MHz	83	48 MHz	29

Außerdem gibt es einen externen Eingang für den Frequenzgenerator. Der Frequenzgenerator ist dabei so einzustellen, dass er weder negative Spannungen noch Spannungen über 5 V ausgibt, da sonst der Hauptchip zerstört wird. **Vor der Benutzung ist auf jeden Fall ein Betreuer hinzuzuziehen!**

External Clock Pin Nr.		
UP1	UP2	UP3
2	2	144

2.3.6 Besonderheiten der UP1/UP2-Boards

Zur Übertragung des Programms auf das Board ist hier ein sogenannter ByteBlaster Adapter nötig. Neben Adaptern für den Parallelport stehen auch einige USB-Blaster zur Verfügung. Diese benötigen den USB-Treiber aus dem Unterverzeichnis `drivers\usbblaster` der Quartus II Installation.

Der MAX7000S Chip behält seine JTAG-Programmierung auch ohne Spannungsversorgung. Beim ersten Einschalten wird also sofort das zuletzt geladene Programm gestartet.
Die UP1/UP2-Boards sind mit zwei Tastschaltern (gedrückter Zustand entspricht logisch GND), zusätzlichen 16 LEDs und 16 DIP-Schaltern ausgestattet, die mit Kabeln an freie Pins angesteckt werden können. Freie IO-Pins sind z.B. 4, 6, 8, 9, 36, 80 und 81.

2.3.7 Besonderheiten des UP3-Boards

Auch das UP3 Board benötigt zur Programmierung einen ByteBlaster Adapter (s. 2.3.6). Der Cyclone Chip verliert bei jeder Unterbrechung der Spannungsversorgung seine JTAG-Programmierung. Ähnlich wie bei den DE1 Boards ist eine nichtflüchtige AS-Programmierung über den EPCS1 Chip möglich. Dazu besitzen die UP3-Boards zwei Programmiereingänge (JTAG und AS).
Um eine AS Programmierung durchzuführen, muss zunächst unter **Assignments** → **Device** → **Device&Pin Options** → **Configuration** das Configuration Device **EPCS1** gewählt und neu kompiliert werden. Dann wird im Programmer der Mode Active Serial gewählt und mit **Add File**... die neu erzeugte Programmdatei (Dateiname .pof statt .sof) benutzt. Die AS Programmierung wird nun bei jedem Neustart des Boards verwendet. Führt man eine normale JTAG Programmierung durch, wird das AS Programm bis zum nächsten Neustart überschrieben.

Das UP3-Board besitzt zusätzlich zu den oben aufgezählten Anschlüssen noch folgende fest verdrahtete Elemente links unten auf der Hauptplatine:

- Vier grüne LEDs, *active high*
- Vier DIP-Schalter, Schalterstellung ON ergibt logisch VCC
- Vier Tastschalter, gedrückter Zustand entspricht logisch GND

grüne LED Nr.	Pin Nr.	DIP-Schalter Nr.	Pin Nr.	Tastschalter Nr.	Pin Nr.
1	53	1	58	1	48
2	54	2	59	2	49
3	55	3	60	3	57
4	56	4	61	4	62

2.4 Verwendung von Konfigurationsdateien für die Anschlussbelegung

Alternativ zur manuellen Eingabe im „Pin Planner“, kann die Anschlussbelegung auch mithilfe von Konfigurationsdateien im .csv („comma separated value“) Format vorgenommen werden. Das hat vor allem den Vorteil, dass der Wechsel von einem Boardtyp auf den anderen sehr einfach wird. Dazu wird unter **Assignments** → **Import Assignments** die passende .csv Datei ausgewählt und neu kompiliert. Die Pins können anschließend im „Pin Planner“ kontrolliert oder verändert werden.

In der .tdf Programmdatei müssen dann aber für Inputs und Outputs einheitliche Variablennamen verwendet werden. Die in den .csv Dateien verwendeten Variablennamen sind:

Variable	Bedeutung	Vorhanden auf Board
HEX0[6..0]	7-Segmentanzeige 0, ohne DP	alle
HEX1[6..0]	7-Segmentanzeige 1, ohne DP	alle
HEX2[6..0]	7-Segmentanzeige 2, ohne DP	DE1
HEX3[6..0]	7-Segmentanzeige 3, ohne DP	DE1
LEDR[7..0]	rote LEDs	UP1, UP2, UP3
LEDR[9..0]	rote LEDs	DE1
LEDG[3..0]	grüne LEDs	UP3
LEDG[7..0]	grüne LEDs	DE1
AMPELR[4..1]	Ampel LEDs rot	alle
AMPELY[4..1]	Ampel LEDs gelb	alle
AMPELG[4..1]	Ampel LEDs grün	alle
KEY[3..0]	Taster	DE1, UP3
SW1[7..0]	DIP Schalter Bank1	UP1, UP2, UP3
SW1[9..0]	Schiebeschalter	DE1
SW2[7..0]	DIP Schalter Bank2	alle
CLK	Systemtakt (unterschiedliche Frequenzen!)	alle
EXT	externer BNC Eingang	alle