6 Aufgabenstellungen L-1

Aufgabe 1 BCD zu Siebensegment Konverter 3 Pkt.

In dieser Aufgabe soll der grundlegende Umgang und die Arbeit mit FPGAs geübt werden. Es werden eigene Verhaltens- und Strukturmodelle entwickelt, simuliert, synthetisiert und die entstehenden Netzlisten betrachtet.

Die Siebensegmentanzeige eignet sich auch zur Darstellung von Sedezimal-Code oder BCD-Codes.

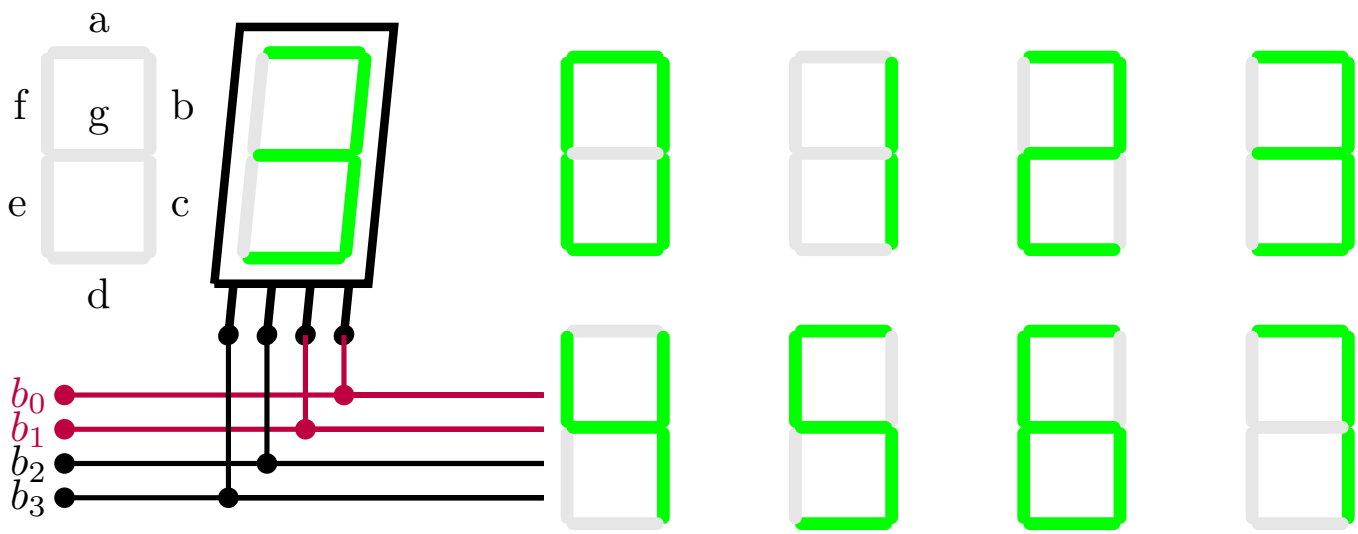

Abbildung 5: Siebensegmentanzeige für BCD-Code.

Abbildung 5 zeigt ein 7-Segment-Modul, welches über ein 4-Bit Datenwort mit den Bits b_0, b_1, b_2 und b_3 angesteuert werden kann. Die Wahrheitstabelle auf der folgenden Seite zeigt die Codierung für eine BCD-Darstellung.

Das Signal LTN steuert die Funktion des Decoders. Ist $LNT = 0_2$, so ist der Decoder inaktiv. Das Signal BLN erlaubt einen Funktionstest der Segmente. Mit$BLN = 1_2$ werden alle Segmente aktiv geschaltet, ohne das die Steuerbits $b_3, \ldots, b_0$ aktiv sind.

Nutzen Sie die Schalter $SW_{0\ldots3}$ und verbinden Sie diese mit den Steuerbits $b_{0\ldots3}$. Zur Darstellung nutzen Sie das Siebensegmentelement mit der Bezeichnung $HEX0$. Die Segmente dieses Displays sind mit $HEX0_0, HEX0_1, \ldots, HEX0_6$ bezeichnet.

Es soll eine rein kombinatorisches BCD zu Siebensegement Konvertermodul gebaut werden. Dieses soll der in Tabelle 1 dargestellten Wahrheitstabelle entsprechen.

▶ Fragestellungen zur Lösung

1. Was ist der Unterschied und die Vor- und Nachteile einer Implementierung **mit** / **ohne** Prozess?
2. Schreiben Sie einen VHDL-Verhaltensmodell für den **BCD zu Siebensegmet Konverter**. Verwenden Sie hierfür bitte die Ihnen zur verfügung gestellte Vorlage.
3. Schreiben Sie eine **Testbench** und überprüfen Sie die Funktion Ihres Konverters mit **ModelSim**.
4. Legen Sie ein neues **Quartus** Projekt an und **synthetisieren** Sie Ihr Design.
5. Betrachten und **verstehen** Sie die aus Ihrem Code generierte **Netzliste** mittels des RTL-Viewers (Tools → Netlist Viewers → RTL Viewer).

	Eingänge						Segmente							
$(i)_{10}$	LTN	BLN	b_3	b_2	b_1	b_0	a	b	c	d	e	f	g	a f g b e c d
0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	0	0	0	0	1	1	0	0	1	1	1	1	1
2	1	0	0	0	1	0	0	0	1	0	0	1	0	2
3	1	0	0	0	1	1	0	0	0	0	1	1	0	3
4	1	0	0	1	0	0	1	0	0	1	1	0	0	4
5	1	0	0	1	0	1	0	1	0	0	1	0	0	5
6	1	0	0	1	1	0	0	1	0	0	0	0	0	6
7	1	0	0	1	1	1	0	0	0	1	1	1	1	7
8	1	0	1	0	0	0	0	0	0	0	0	0	0	8
9	1	0	1	0	0	1	0	0	0	0	1	0	0	9
10	1	0	1	0	1	0	1	1	1	1	1	1	1	
11	1	0	1	0	1	1	1	1	1	1	1	1	1	
12	1	0	1	1	0	0	1	1	1	1	1	1	1	
13	1	0	1	1	0	1	1	1	1	1	1	1	1	
14	1	0	1	1	1	0	1	1	1	1	1	1	1	
15	1	0	1	1	1	1	1	1	1	1	1	1	1	
16	0	1	x	x	x	x	0	0	0	0	0	0	0	8
17	0	0	x	x	x	x	1	1	1	1	1	1	1	

Tabelle 1: Codierung des BCD zu Siebensegmet Konverters

Aufgabe 2 Shift-Add-3 IP-Core 4 Pkt.

Im digitalen Hardwaredesign ist es, ähnlich wie in der Softwareentwicklung, gängig, nach dem Prinzip *use and reuse* zu arbeiten. In der Softwareentwicklung wird dieses Prinzip unter Anderem durch Bibliotheken, die Teil der Programmiersprache sind oder durch dritte zur Verfügung gestellt werden verwirklicht.
In der Hardwareentwicklung spricht man hierbei von *Intellectual Property (IP)-Blöcken)*. IP-Blöcke sind Hardwarebeschreibungen, welche eine bestimmte Funktionalität übernehmen und in das eigene Design eingebunden werden können.

In dieser Aufgabe wird ein IP-Block zur Verfügung gestellt, welcher den *shift-add-3* Algorithmus zur Konvertierung von der Binär-Darstellung in die BCD-Darstellung implementiert.
Ziel dieser Aufgabe dass Sie die Funktionalität des Ihnen zur Verfügung gestellten IP-Cores verifizieren.

► **Gehen Sie dazu wie folgt vor:**

- Analysieren Sie den IP-Core und verstehen Sie die Funktionalität.
- Schreiben Sie eine **Testbench** und überprüfen Sie die Funktionalität des IP-Cores mit **ModelSim**.
- Legen Sie ein neues **Quartus** Projekt an und **synthetisieren** Sie den IP-Core.
- Betrachten und **verstehen** Sie die aus dem IP-Core generierte **Netzliste** mittels des RTL-Viewers (Tools → Netlist Viewers → RTL Viewer).
- Betrachten Sie die ***State Machine*** (im RTL-Viewer in gelb dargestellt) per Doppelklick im *State Machine Viewer*:
  - Was bedeuten die **Transitions** in der ***State Table***? (am unteren Bildschirmrand dargestellt)
  - Was könnte das **Encoding** sein? (auswählbar am unteren Bildschirmrand)
- Betrachten Sie die **Register**:
  - Warum ist der SCLR Eingang der Register auf den logischen Wert '0' gezogen?
- Betrachten Sie den **Reset-Pfad**:
  - Warum ist das Reset-Signal an keinem der Register sonder nur an Multiplexern angeschlossen?

Aufgabe 3 Binär zu Siebensegment Konverter 3 Pkt.

Ziel dieser Aufgabe ist es, durch Verschalten der in den Aufgaben 1 und 2 bearbeiteten Hardwarebeschreibungen, einen binär zu Siebensegment-Konverter zu erhalten, dessen Funktionalität auf dem DE2-115 Entwicklungsboard demonstriert werden soll.

► **Gehen Sie dazu wie folgt vor:**

- Legen Sie ein neues **Quartus** Projekt an. Achten Sie darauf, den richtigen FPGA-Baustein auszuwählen.
- Fügen sie die Dateien aus den Aufgabenteilen eins und zwei Ihrem Projekt hinzu.
- Legen Sie eine neue Datei an und schreiben Sie eine **Strukturbeschreibung** in der Sie **so effizient wie möglich** einen binär zu Siebensegment-Konverter mit 3 Ziffern/Digits unter Verwendung der Module aus Aufgabe eins und zwei implementieren.
- Bringen Sie Ihr Design mit selbst gewählten, geeigneten In - und Outputs auf das Entwicklungsboard:
  - Als Takteingang kann entweder ein *push button*, oder die auf dem Entwicklungsboard verbaute 50 MHz Clock dienen.