FH-OÖ/Hagenberg/HSD
Chip-Design 3 — CHD3
Markus Pfaff © 2023
Übung 2
.qsf, Know-what, DE1-SoC-Schaltung

MUX ist als Name für Multiplexer durch ein Bibliothekselement von IntelFPGA belegt. Leider bekommt man hierzu keine Fehlermeldung. Dieser Name sollte also vermieden werden. Besser wäre z.B. Mpx.

1. Aufgabe *Pimp your .qsf*

Im Verzeichnis `...\unitHelloOnPcbDe1Soc\quartus` befindet sich das Quartus Settings File `HelloOnPcbDe1Soc.qsf`, welches genauer untersucht werden soll. Öffnen Sie die Datei in ihrem Texteditor.

- Quartus hat einen eigenen Menüpunkt zur Restrukturierung dieser Datei: Project—Organize Quartus II Settings File. Führen Sie diese durch. Wie hat sich die Datei verändert?
- Am Anfang der Datei ist ein Hinweis von Altera eingetragen, der das Editieren der Datei betrifft. Was besagt dieser?
- Starten Sie den Compile-Prozess mit der veränderten Datei erneut. Quartus liest die Datei dabei neu ein. Nun öffnen Sie den *Assignment Editor* in Quartus.
- Dort sind für alle Pins des Designs zwei Einstellungen vorhanden, die nun wie in der QSF-Datei lesbarer sortiert sind. Die Einstellung *Location* ist spezifisch für jeden Pin, da damit der richtige Gehäuse-Pin dem jeweiligen *Port* der *entity* `HelloOnPcbDe1Soc` zugeordnet wird. Beachten Sie, dass dies im Falle von Vektoren für deren Elemente im Einzelnen geschehen muss und nicht pauschal für den ganzen Vektor erledigt werden kann. Erläutern Sie kurz, warum das so ist.
- Die Einstellung für den Wert *I/O Standard* ist dagegen für alle Pins identisch. Dies kann man Quartus mit Hilfe sogenannter *Wildcards* (engl. für die Poker-Spielkarte Joker) für alle Pins in einer einzigen Zeile elegant mitteilen. Hierzu sollen zunächst alle Zeilen mit dem *Assignment I/O Standard* außer einer einzigen gelöscht werden: Markieren, rechte Maustaste, *Delete*. In der verbliebenen einzigen Zeile für *I/O Standard* in Spalte *To* doppelklicken und * eintragen. Dieses Zeichen steht für eine beliebige Anzahl beliebiger Zeichen, damit also jeden beliebigen Namen. Das Zeichen ? ist ebenfalls eine *Wildcard* und steht für *ein* beliebiges Zeichen.
- Nun im Fitter-Report unter *Resource Section / I/O Standards Section / I/O Assignment Warnings* nachschauen. Hierzu gibt es auch eine entsprechende *Warning* von Quartus.
- Nun sollen die noch beanstandeten Einstellungen *drive strength und slew rate* (im *Assignment Editor* heißen sie: *Current Strength* und *Slew Rate*) für die richtigen Signale

hinzugefügt werden, wobei die Einstellungen *Minimum Current* respektive 0 verwendet werden können. Hierzu in der untersten Zeile im *Assignment Editor* (<<*new*>>) auf den *Assignment Name* doppelklicken und das gewünschte *Assignment* auswählen.

Beachten Sie, dass für jedes *Assignment* der Auswahl angegeben ist, ob *Wildcards* möglich sind. Nun in der neu angelegten Zeile den Eintrag für die Spalte *Value* doppelklicken und auswählen. Nun muss das Assignment noch vervollständigt werden, indem angegeben wird, worauf es überhaupt angewendet werden soll. Da es *Assignments* gibt, welche für Verbindungen zweier Punkte im FPGA gedacht sind, gibt es die beiden Spalten *From* und *To*. Für *Assignments*, welche sich auf einen einzigen Punkt innerhalb des FPGAs beziehen, ist die Spalte *From* ausgegraut.

Mit einem Doppelklick aktivieren Sie das Feld in der Spalte *To* nun für die Eingabe. Es erscheint ein Eingabefeld, ein Icon und ein Auswahlpfeil. Klicken Sie den Icon. Es öffnet sich das Fenster des *Node Finders*, der schon in der vorigen Übung kurz zu sehen war.

- Mittels *Node Finder* werden alle Ausgänge (nur um diese geht es) gesucht. Der hierzu nötige Filter-Dialog zur Auswahl nach *Node*-Typ ist nur bei aktiviertem Pfeil-*Button* sichtbar.

  Gefunden werden die Vektoren `HEX1`, `HEX2` und `LEDR`. Im *Node Finder* kann man die Pins aus der linken Spalte (*Found Nodes*) zu den *Selected Nodes* hinzufügen. Mit *OK* werden die entsprechenden *Assignments* erzeugt.

- Schauen Sie nun wieder in der QSF-Datei nach. Offenbar gibt es neben *Wildcards* noch eine zweite Möglichkeit, um mehrere *Assignments* in einer Zeile zu setzen. Und zwar?

- Prüfen Sie mit einem erneuten *Compile* in Quartus nach, ob die Warnungen, um die es ging nun nicht mehr vorhanden sind.

- Im *Messages Utility Window* von Quartus sind noch immer einige *Warnings* zu finden:
  - `oLed(8)` wird nicht sinnvoll im VHDL-Code angesteuert.
  - Zwei Eingänge des Vektors `SW` werden im VHDL-Code zwar definiert, aber nie verwendet.

  Gestalten Sie den VHDL-Code (und zugehörige *Assignments*) so um, dass diese *Warnings* beseitigt werden.

- *Warnings* die nicht warnen sondern nur nerven gibt es leider auch, aber sie lassen sich ausblenden. Dass das *Feature LogicLock* in der *Lite Edition* von Quartus Prime nicht vorhanden ist, sondern nur in der *Standard Edition* haben wir inzwischen oft genug mitgeteilt bekommen. Die *Warning* mit der rechten Maustaste anklicken und mit *Suppress* ausblenden hilft in diesem Fall.

- *Warnings*, die eine Bedeutung haben (könnten), sollte man keinesfalls ausblenden! Die *Warning Generated the EDA functional…* ist rätselhaft. Schauen Sie mit der rechten Maustaste und *Help* genauer nach.

  Leider ist auch die Erklärung in der Hilfe nicht um so viel klarer. Es geht hierbei um ein Simulationskonzept, welches mit den Serie-V-FPGAs nicht mehr von Altera unterstützt wird. Daher können Sie nun folgende Einstellung machen, um diese

Warning loszuwerden: *Assignments—Settings...—EDA Tool Settings Simulation—More EDA Netlist Writer Settings...—Generate netlist for functional simulation only—On*. In einer der nächsten Quartus-Versionen wird diese Einstellung vermutlich ohnehin zum Standard werden.

- Nun sollten Sie keinerlei Warnings mehr nach dem *Compile* in Quartus sehen – ein Zustand, der für jedes Design anzustreben ist.

2. Aufgabe *Fakultativ: Swell-Led*

Das Design Swell-LED ist ein Klassiker im Studiengang HSD. Es ist sogar zwei Monate älter als der Studiengang selbst. Mit zwei Tasten kann man eine LED in ihrer Helligkeit einstellen. Das Design ist im Archiv zu dieser Übung enthalten. Im Verzeichnis `doc` der *unit* `SwellLed` findet sich die genaue Spezifikation (Funktionsbeschreibung) des Designs.
Leider fehlen jedoch das *Testbed* und die QSF-Datei. Es ist lediglich eine QSF-Standarddatei von terasic vorhanden, welche *alle* Pins des FPGAs beschreibt und zusätzlich auch berücksichtigt, dass am FPGA SDRAM-Bausteine angeschlossen sind. Der überaus größte Teil dieser Datei ist also für diese Aufgabe irrelevant und kann gelöscht werden. Andere Einstellungen können unter Verwendung von *Wildcards* deutlich eleganter gemacht werden und wieder andere fehlen und sollten per *Assignment Editor* in Quartus später noch hinzugefügt werden.
Erstellen Sie ein Testbed (*entity* und *architecture* welches passende Tasten und LEDs des Boards an das Design anschließt. Berücksichtigen sie dabei Signale mit negativer Logik (z.B. `KEY`).
Erstellen Sie ein Quartus-Projekt, führen Sie den *Compile*-Vorgang durch und spielen Sie den *Bitstream* auf dem Board ein. Nun sollten Sie die LED mittels der entsprechenden Tasten in der Helligkeit einstellen können.

3. Aufgabe *2zu1-Multiplexer*

In dieser Aufgabe geht es um einen 2zu1-Multiplexer (*entity* `Mpx`) mit den Eingängen *iA* und *iB*, welche auf den Ausgang *oY* gemultiplext werden. Der Auswahleingang *iSel* bestimmt, welcher der Eingänge auf den Ausgang *oY* durchgeschaltet wird.

Stellen Sie zunächst eine Wertetabelle des Multiplexers auf.

Der Multiplexer soll nun auf drei verschiedene Arten realisiert werden. Es werden nur die einfachen logischen Grundfunktionen **and**, **or**, ... verwendet. Die drei Beschreibungsarten sind:

- Mittels aller Minterme des *Karnaugh/Veitch-Diagramms*. Es existiert also ein Term pro Feld, das einen 1er enthält. *Architecture: Minterms*.
- Mittels der wesentlichen Primimplikanden. Es soll also zunächst minimiert werden. Danach wird das Minimierungsergebnis implementiert. *Architecture: PrimeImpl*.
- Ausschliesslich mit Hilfe der Funktion **nand**. *Architecture: NandOnly*.

Alle drei Implementationen sollen innerhalb einer Prüfumgebung *Testbed* namens `TbdMpx` instanziiert werden. Dies ist jedoch *nicht* die *testbench* sondern dient zum Probieren auf

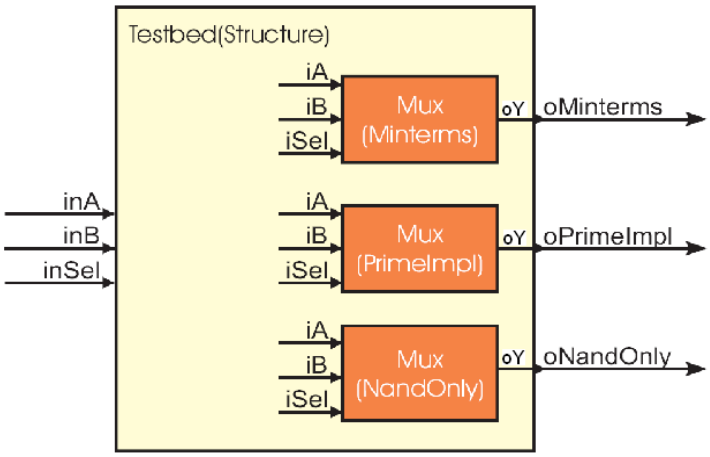

Abbildung 1: Struktur des *testbeds* welches implementiert werden soll.

dem Board. Wie in Abb. 1 dargestellt, werden die Eingänge (z.B. von den Tastern) aller drei *entities* mit den gleichen *signals* beschaltet. Die Ausgänge werden natürlich getrennt betrachtet.

Um Ihre Beschreibungen lesbarer zu machen, können Sie die *constants* für positive und negative Logik verwenden, die im projektweiten *package Global* beschrieben werden. Dieses *package* finden Sie im Archiv des Projektes *SwellLed*. Eine gedrückte Taste hat wegen der negativen Logik den Wert *cnActivated*. Die Schalter dagegen verwenden eine positive Logik. Im *Testbed* werden Ein- und Ausgangssignale gegebenenfalls negiert, so dass die eigentliche Beschreibung immer mit positiver Logik erstellt werden kann.

Alternativ können Sie bei positiver Logik auch die Werte `'0'` und `'1'` und bei negativer Logik die Werte `not('0')` und `not('1')` für den inaktiven, respektive aktiven Wert verwenden.
Nicht alle Signale kennen einen aktiven und einen inaktiven Wert. Ein Taktsignal wechselt beispielsweise einfach zwischen zwei Werten hin und her, von denen keiner den Wert aktiv oder inaktiv hat.
Erstellen Sie für dieses *Testbed* eine *testbench* und weisen Sie durch Simulation nach, dass alle drei Entwürfe fehlerfrei funktionieren.

Um Ihren Entwurf auch auf dem Board ausprobieren zu können, belegen Sie die Eingänge mit den Schaltern (*iA: SW2*, *iB: SW1*, *iSel: SW0*) oder mit den Tastern (*inA: KEY2*, *inB: KEY1*, *inSel: KEY0*) und die drei Ausgänge mit den LEDs *LEDR0*, *LEDR1* und *LEDR2* in der Reihenfolge der obigen Aufzählung. Wenn Sie sicher sind, dass ihr Design und mehr noch die Zuordnung der FPGA-Pins zu den *Ports* aus der VHDL-Beschreibung fehlerfrei sind, probieren Sie den Entwurf auf dem Board aus.

Untersuchen Sie die entstehenden Schaltungen mit dem *RTL-Viewer*.
Vergleichen Sie den Verbrauch an Ressourcen in den *Ressource Usage Summarys* der Schritte *Analysis & Synthesis* und *Fitter*. Gehen Sie dem Unterschied durch Vergleich der Schaltungen mit dem *Technology Map Viewer (Post-Mapping)* und dem *Technology Map Viewer (Post-Fitting)* auf den Grund.