FH-OÖ/Hagenberg/HSD
Chip-Design 3 — CHD3
Markus Pfaff © 2024
Übung 1
Zieltechnologie, Quartus, DE1-SoC

OBERÖSTERREICH

Organisatorisches

- Modus und Notengebung

1. Aufgabe *Hello again!*

a □ Im Rahmen dieser Aufgabe soll das Beispiel-Design, welches in der letzten Übung in CHD2 Thema war, mithilfe des Design-Frameworks Quartus II von Altera genauer unter die Lupe genommen werden. Bitte vollziehen Sie die Schritte simultan nach, welche der Übungsleiter am Projektor zeigt.
Ausgangspunkt ist das – vorerst noch – unveränderte Design aus CHD2. Dieses kopieren Sie bitte in ein Verzeichnis namens designsToChangeChd2.

- Ordnerstruktur
  - Designlibrary oder Projektorientierung?
  - Professionelle Nutzung in einem Projekt? Lösung: Verlinken auf solche Sammlungen aus Projekt heraus mittels Versionsverwaltungswerkzeugen (Git)
  - Jede *unit* wird einzeln geprüft (Simulation in *testbench* und Synthese in einem *testbed*, also einem PCB-Adapter)
  - Verschiedene Simulations- und Synthesewerkzeuge und auch verschiedene Zieltechnologien nebeneinander (z.B. Verzeichnisse `quartus` und `vivado`)
  - Jede *unit* kann beliebige andere *units* verwenden
  - Verzeichnis `doc`: Selbst erstellte Dokumentation
  - Verzeichnis `literature`: Fertig erhaltene Dokumente (Datenblätter, Übungsangaben, Lehrbücher,...)
- Tool-Shortcuts
  - Privat-PC/Laptop: Quartus Prime Lite (mit kostenloser Questasim Intel Edition) in Verzeichnis *ohne Leerzeichen* installieren.
  - Shortcuts von Quartus und Questasim (aus der Quartus-Installation)
    * in die Windows-Kacheln übernehmen.
    * In den Windows-Kacheln mit rechtem Mausklick *More – Open File Location* auswählen und dort für den jeweiligen *shortcut* den Eigenschaftsdialog öffnen.

    * Leeres Arbeitsverzeichnis im *shortcut*: Werkzeug startet in dem Verzeichnis in dem sich der *toolshortcut* befindet. Dort werden auch meist die erzeugten Dateien abgelegt und benötigte Dateien gesucht. Dies ist eine Windows-Konvention, die von vielen EDA-Tool-Herstellern eingehalten wird.
    * Zieleinstellung der *toolshortcuts*

      Questasim: `%QUARTUS_ROOTDIR%\..\questa_fse\win64\questasim.exe`
      Quartus: `%QUARTUS_ROOTDIR%\bin64\quartus.exe`

      So wird auf die von Quartus bei dessen Installation automatisch angelegte Environment-Variable `QUARTUS_ROOTDIR` zurückgegriffen. Diese enthält den Pfad zum Installationsverzeichnis von Quartus, z.B. `C:\Programme\intelFpga\20.1.1std\quartus`. Jede Neuinstallation von Quartus aktualisiert automatisch diese Variable.
    * Nun kann man die *shortcuts* einfach aus den Windows-Kacheln in das Verzeichnis des gerade bearbeiteten Projektes ziehen. Diese sind auch nach der nächsten Quartus-Neuinstallation noch immer funktionsfähig.
  – In Projekten wird meist mit Build-Werkzeugen gearbeitet (vunit, make, scons,...)

- Quartus mit Shortcut aus dem *testbed* `HelloOnPcbDe1Soc` starten, Projekt öffnen (alternativ: Doppelklick auf Projektdatei mit Endung .qpf)

- Menüeinträge View—Utility Windows
  – Project Navigator – Unterschiedliche Sichten auf das Design
  – Node Finder – Automatisch zugehörige Datei im Editor aufrufen. Editor einstellen: Tools—Options—Preferred Editor
  – TCL Console – Befehl pwd
  – Messages, Status, Tasks – Button Start Compilation
  – Status-Window: Unnötig!

- Tasks-Window: Flow: Compilation einstellen
  – Wofür sind die Werkzeuge zuständig? Ergebnis jedes Schrittes?
  – Flow Log im Compilation Report Window
  – Analysis & Synthesis (eigener Button)
    * View Report
      · Summary – Warum keine ALM-Anzahl?
      · Ressource Usage Summary
    * Netlist Viewers
    * I/O Assignment Analysis—Pin Planner
  – Fitter

        * View Report
            · Summary – Anzahl an ALM ist bekannt
            · Netlist Optimizations
            · Ressource Section
        * Chip Planner
        * Technology Map Viewer – Locate Node in:
        * Ressource Property Viewer – Prinzipiell auch Design auf unterster Ebene möglich
    - Assembler
        * Edit Settings: Assembler abschalten
    - Timing Analysis (TimeQuest)
        * Wozu ist der Timing Analyzer gut? Funktionierendes Design = Funktion + Timing
        * Taktorientierte Analyse ohne Berücksichtigung der logischen Funktion (daher: statische Analyse)
        * Festlegung des Timings: .sdc-Datei (im Editor anschauen)
        * Multicorner Timing Analysis
        * Unconstrained Paths
    - EDA Netlist Writer
        * Simulation—Generated Files
        * Datei im Editor anschauen
    - Programmer
- Tasks-Window: Flow Full Design
    - Start Project
    - Create Design
        * Qsys: Systementwurf mit IP-Blöcken auf Basis standardisierter On-Chip-Busse (Altera Avalon, ARM AXI)
    - Assign Constraints
        * Pin Planner
        * Assignment Editor
    - Verify Design
        * Simulate Design
        * On-chip Debugging
        * Power Analyzer PowerPlay
        * SSN Analyzer

2. Aufgabe *Hello somewhat different*

a □ Verändern Sie das Design folgendermaßen:

- Die beiden Hex-Zahlen sollen auf den beiden linken Hex-Anzeigepaaren gleichermaßen angezeigt werden. Auf dem rechten Paar soll nichts angezeigt werden.
- Die blinkende LED soll LEDR8 sein.
- Mit SW9 soll das Blinken ausgeblendet werden (an der LED soll der Pegel für eine dunkle LED anliegen).
- Bauen Sie noch eine eigene Veränderung nach Belieben ein.

b □ Das Design soll auf dem Board funktionsfähig sein und demonstriert werden können.

3. Aufgabe *Schaltplan: Tasten, Schalter und LEDs*

Es geht nun um die Stellen im Schaltplan des DE1-SoC-Boards, welche im Design eine Rolle spielen (LEDs, Tasten,...). Bei den nachfolgenden Fragen kann auch das *User Manual* des DE1-SoC hilfreich sein.

a □
- Wie kann man im Schaltplan erkennen, ob LEDs und LEDs in 7-Segment-Anzeigen bei anliegendem 1-Pegel leuchten oder dunkel bleiben? Braucht man weitere Unterlagen wie Datenblätter der LEDs und das *User Manual* des DE1-SoC?

b □
- Berechnen Sie die Ströme durch die Einzel-LEDs und die LEDs der 7-Segment-Anzeigen für den Fall dass diese leuchten. Die Durchlassspannung einer roten LED nehmen Sie dazu mit 1,6 V an. Wie hoch ist der Leistungsumsatz jeweils?

c □ d □
- Die Serienwiderstände zwischen FPGA-Pin und jedem Schiebeschalter (SW) dienen der Strombegrenzung im Fehlerfall. Wie kommt es möglicherweise zu einem solchen Fehlerfall und wie hoch ist dann der Strom? Ist dieser Strom für das FPGA zulässig? Wie hoch ist der Strom für den Normalfall (kein Fehler)?

e □
- Die Schaltung, welche die Tasten KEY0 bis 3 mit dem FPGA verbindet, sorgt dafür, dass die Tasten entprellt werden. Erläutern Sie, wie die Schaltung funktioniert (Datenblatt des Bausteins 74HC245 liegt der Board-Dokumentation von terasic bei).