Chip-Design: Simulation G1/G2

Thomas Müller-Wipperfürth © 2015

7. Übung

Altera Quartus II, Terasic DE1-Soc

Wie wird VHDL-Code zu Hardware?

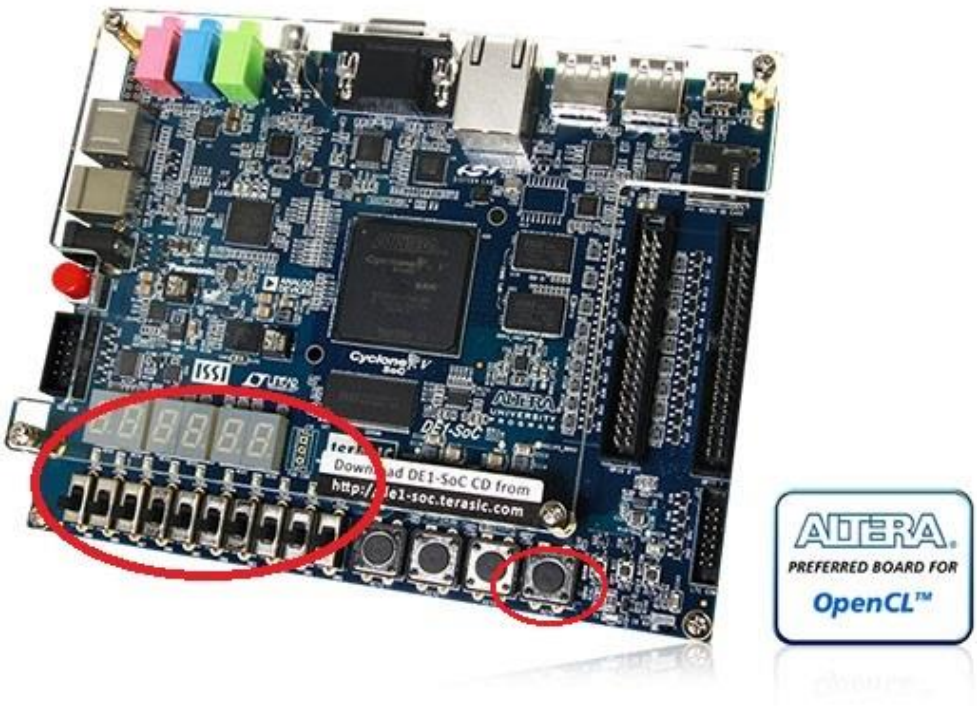
In dieser Übung wird der VHDL-basierte Entwurfsablauf von der Simulation mit QuestaSim über die Synthese und Konfiguration des FPGAs mit der Design-Software Altera Quartus II bis zur Demonstration der Funktionalität auf dem Entwicklungs-Board DE1-SoC vorgestellt. Als Beispiel dient das Projekt `HelloDE1`.

Das Projekt HelloDE1

`HelloDE1` verwendet die Schalter, die LEDs und die Sieben-Segment-Anzeigen des DE1-SoC: Mit acht Schaltern (SW0-SW7) können zwei Hexadezimalzahlen eingestellt werden, die auf zwei Sieben-Segment-Anzeigen (HEX0, HEX1) dargestellt werden. Die LEDs (LEDR0-LEDR7) zeigen zusätzlich an, welche Schalter auf "on" stehen. Die linke LED (LEDR9) blinkt (*Heartbeat*), sobald das FPGA mit `HelloDE1` konfiguriert wurde.

Altera DE1-SoC Board

Das DE1-SoC Board von Terasic dient als Hardwareplattform zur Realisierung einfacher FPGA-Entwürfe bis zu Hardware/Software-Codesign-Systemen aus FPGA und ARM9-Prozessoren unter dem Betriebssystem Linux.
Die ausführliche Dokumentation (CD-ROM zur Version E - rev. E) zu diesem Board findet sich unter: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=165&No=836&PartNo=4.
Die Beispiele dieser CD-ROM sind allerdings mit der 32-Bit Version der Quartus Software (13.1) erstellt worden. Das User-Manual und der Schaltplan des Boards sind im Verzeichnis *literature* dieser Übung enthalten.



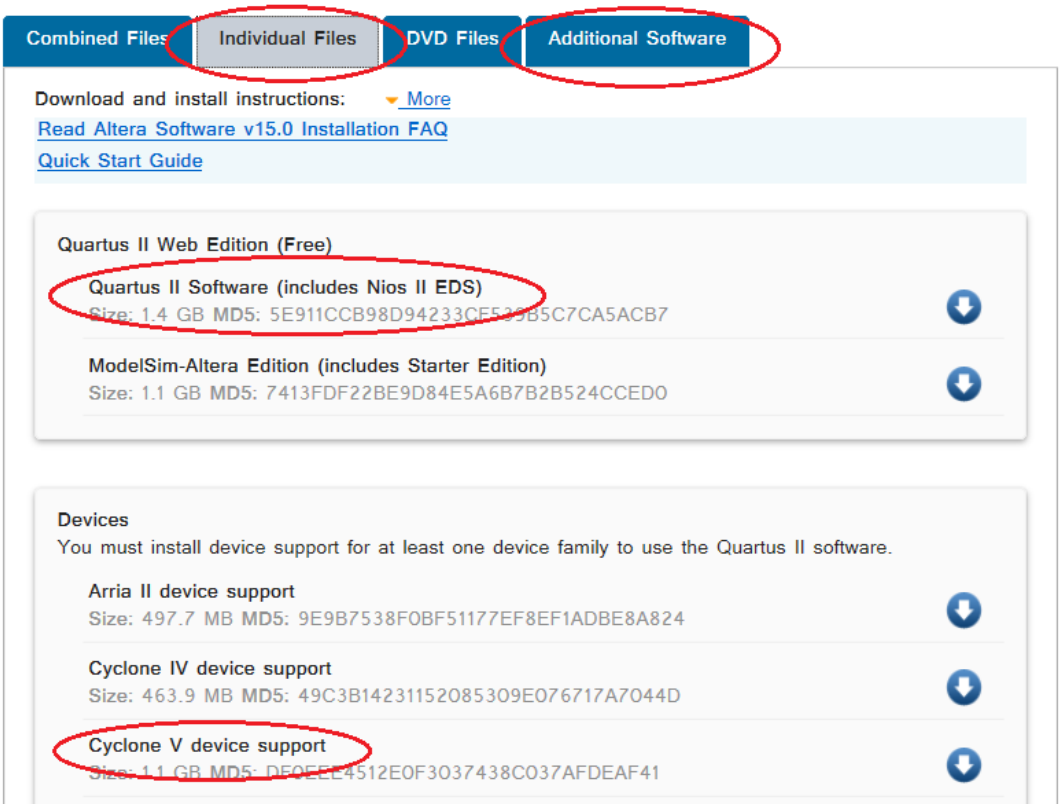
A Vorbereitung

A1 Installation der Altera Design-Software auf einem eigenen Rechner

In den HSD/ESD-Labors ist die lizensierte Version der Design-Software Altera Quartus II installiert. Für die Ausarbeitung der weiteren CHD-Übungen steht die freie Version (Quartus II Webedition) zur Verfügung. Für diese Web-Edition ist kein Lizenz-Server notwendig. Allerdings gibt es einige Einschränkungen, die für die CHD-Übungsbeispiele jedoch keine Relevanz haben.

Für die Quartus-Installation auf einem eigenen Windows-Rechner sind folgende Schritte notwendig:

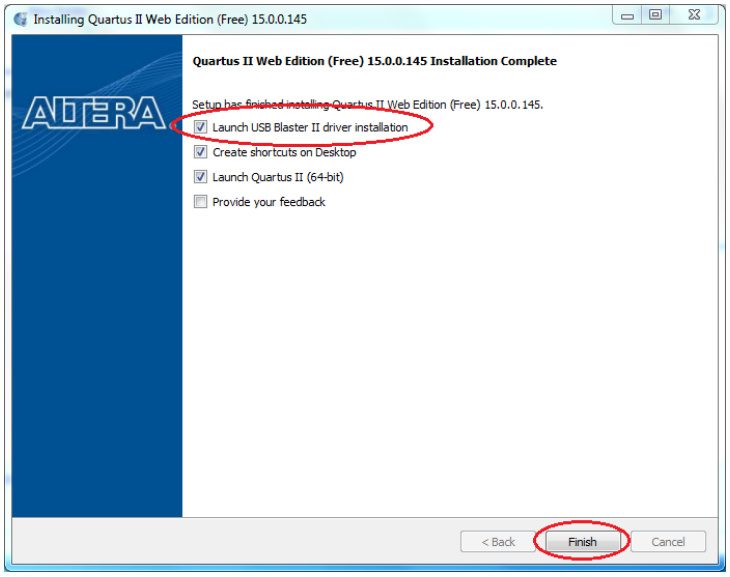
1. **Download der Altera Design-Software** von http://dl.altera.com/?edition=web.
   Die Software (**Altera Quartus II Web Edition**) und die Device-Unterstützung für die **Cyclone V** FPGAs sind nach Wahl des Reiters *Individual Files* sichtbar. Empfehlenswert ist auch der Download der Add-On Software **Quartus II Help**, die im Reiter *Additional Software* verfügbar ist. Diese **drei** Dateien sind in dasselbe temporäre Verzeichnis zu legen.

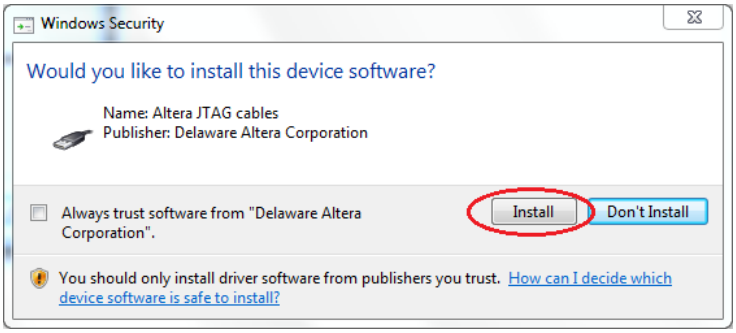


2. **Installation der Software** durch Start von `QuartusSetupWeb-xx.xx.exe`.

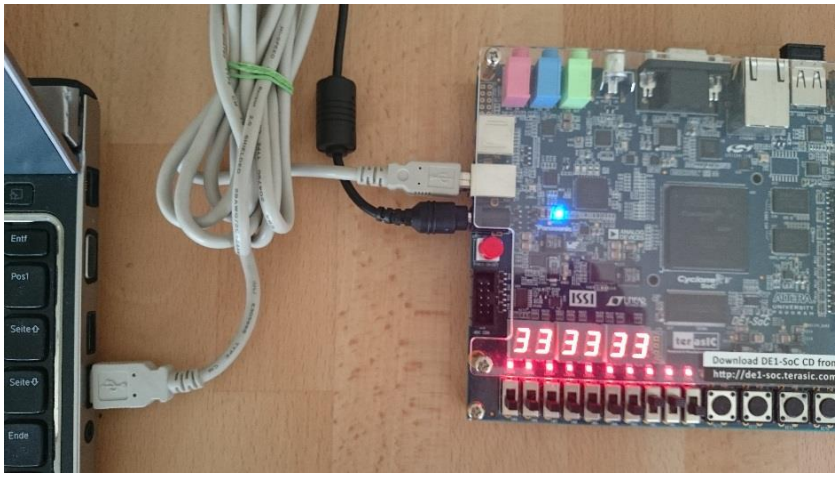
Wichtig: Nicht in einem Pfad mit Leerzeichen,
wie beispielsweise "`C:/Program Files/`" installieren!

Zu lange Pfadnamen können ebenfalls Probleme machen. Nach der ersten Installation der Design-Software Quartus II auf einem Windows-PC müssen auch die **Treiber für die USB-Verbindung** zum DE1-Soc Board installiert werden:



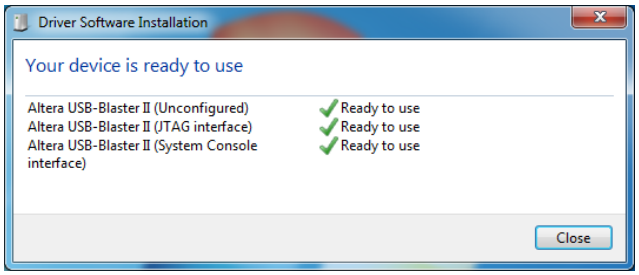


Nach der erfolgreichen Installation kann das DE1-SoC mit dem Rechner verbunden und eingeschaltet werden:

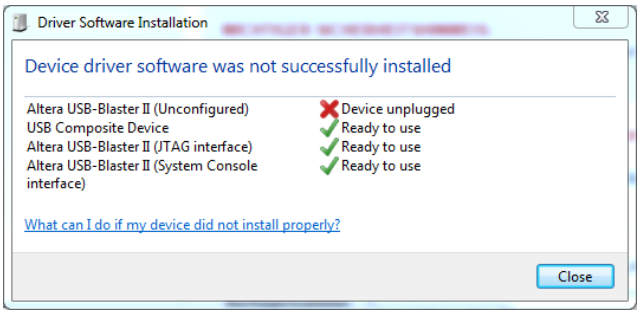


Windows startet automatisch die Installation der Altera USB-Blaster II Device-Treiber. Nach einigen Minuten (!) – bitte nicht ungeduldig werden – sollten die Treiber erfolgreich installiert sein:

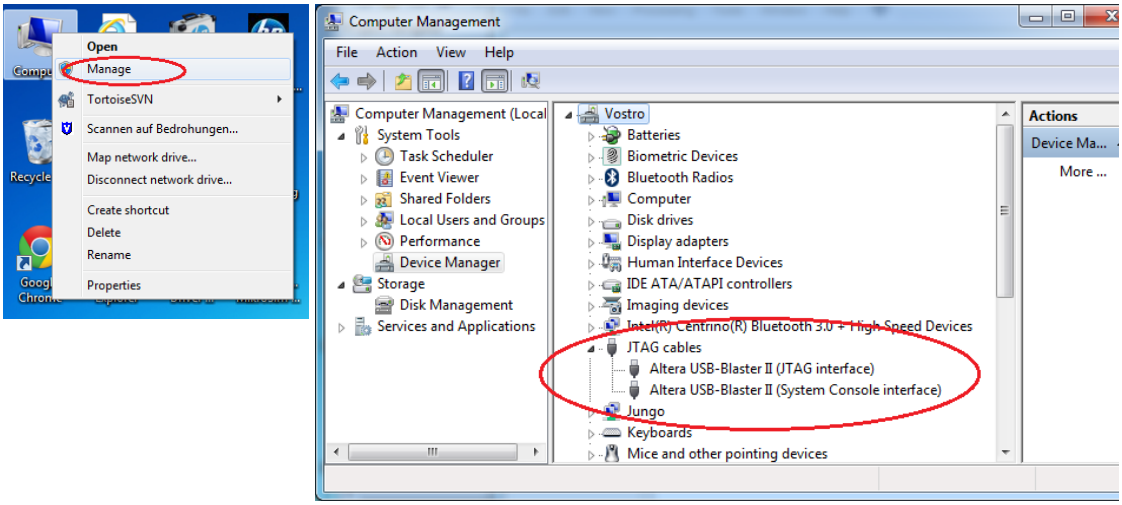
So:



oder auch so:



Anschließend werden die installierten Treiber – bei eingeschaltetem und verbundenem Board – im *Device Manager* des *Computer Management* (erreichbar z.B. mit rechter Maustaste auf Computer/Arbeitsplatz) ersichtlich.



Die Installation auf Linux-Rechnern ist ebenfalls unproblematisch. Näheres dazu im Download-Bereich von Altera.

A2 Einrichten der Verzeichnis-Struktur

In der Übung CHD3-Synthese wird eine neue Verzeichnis-Struktur verwendet. Auch die ZIP-Datei dieser Übung ist nach dieser Struktur organisiert. Eine detaillierte Beschreibung der einzelnen Verzeichnisse wird in CHD3 vorgestellt.

```
/CHD-Root-Directory
    /literature
        /Schematic
            DE1-SoC.pdf
        DE1-SoC_User_manualv.1.2.2_revE.pdf
    /VhdlDesigns
        /grpPackages
            /pkgGlobal
                /src
                    Global-p.vhd
        /grpChd2
            /doc
                CHD2U07.pdf
            /unitHex2SevenSegment
                /src
                    Hex2SevenSegment-e.vhd
                    Hex2SevenSegment-Rtl-a.vhd
                    tbHex2SevenSegment-ea.vhd
                /simQuestasim
                    ModelSim-Shortcut
            /unitHelloDE1
                /simQuestasim
                    ModelSim-Shortcut
                    comp_sim.do
                    wave.do
                /src
                    HelloDE1-e.vhd
                    HelloDE1-Rtl-a.vhd
                    tbHelloDE1-Bhv-a.vhd
                    tbHelloDE1-e.vhd
            /unitTbdHelloDE1
                /simQuestasim
                    ModelSim-Shortcut
                /src
                    TbdHelloDE1-e.vhd
                    TbdHelloDE1-Struct-a.vhd
                /synlayQuartus
                    Quartus-Shortcut
                    TbdHelloDE1.sdc
                    TbdHelloDE1Pins.qsf
```

Shortcuts für Simulation- und Synthese-Software

Im Arbeitsverzeichnis `/VhdlDesigns/grpChd2/unitTbdHelloDE1/synLayQuartus` befindet sich ein Shortcut auf `Quartus.exe`. Das Feld *Start in:* des Shortcuts wurde gelöscht, damit die temporären Dateien im Verzeichnis des Shortcuts erzeugt werden.

In der Übung zu CHD2 wurden bis jetzt die Shortcuts auf eine bestimmte Version von Questasim angelegt. Jetzt werden Shortcuts verwendet, die – ohne Veränderung – immer auf die jeweils aktuelle Installation eines Tools verweisen können. Für Quartus II wird die verwendete Version durch eine Windows-Variable bestimmt. Dazu muss die Umgebungs-Variable `MY_QUARTUS_PATH` gesetzt werden. Dies kann in einem Windows-Kommandofenster (`cmd`) durch den Befehl:

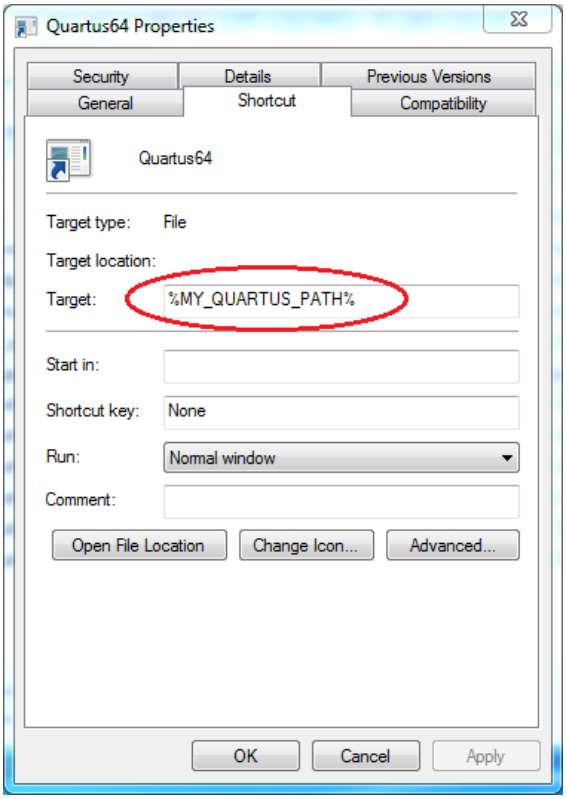
```
setx MY_QUARTUS_PATH "C:\Altera\15.0\quartus\bin64\quartus.exe"
```

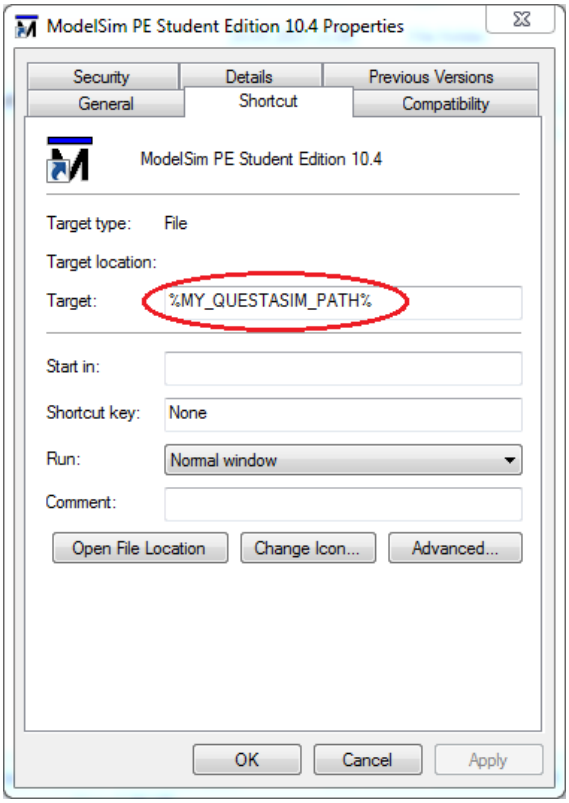
erreicht werden. Der Wert der Variablen ist natürlich an das jeweilige Installationsverzeichnis anzupassen.

Für die Verwendung des Questasim-Shortcuts in `/grpChd2/unitTbdHelloDE1/simQuestasim` muss die Umgebungs-Variable `MY_QUESTASIM_PATH` gesetzt werden:

```
setx MY_QUESTASIM_PATH "C:\Modeltech_pe_edu_10.4\win32pe_edu\modelsim.exe"
```

In den Shortcuts ist der Pfad in *Target:* durch die entsprechende Variable zu ersetzen. Da der Wert der Variablen verwendet soll, sind die Variablen durch "%" zu begrenzen.





B Simulation der RTL-Beschreibungen

Themen in CHD2 sind die Sprache VHDL, Prozesskommunikation mittels *signals*, *transport* und *inertial delay* und der VHDL-Simulationszyklus. In den Übungsbeispielen werden Struktur- und Verhaltensbeschreibungen von Systemen modelliert.

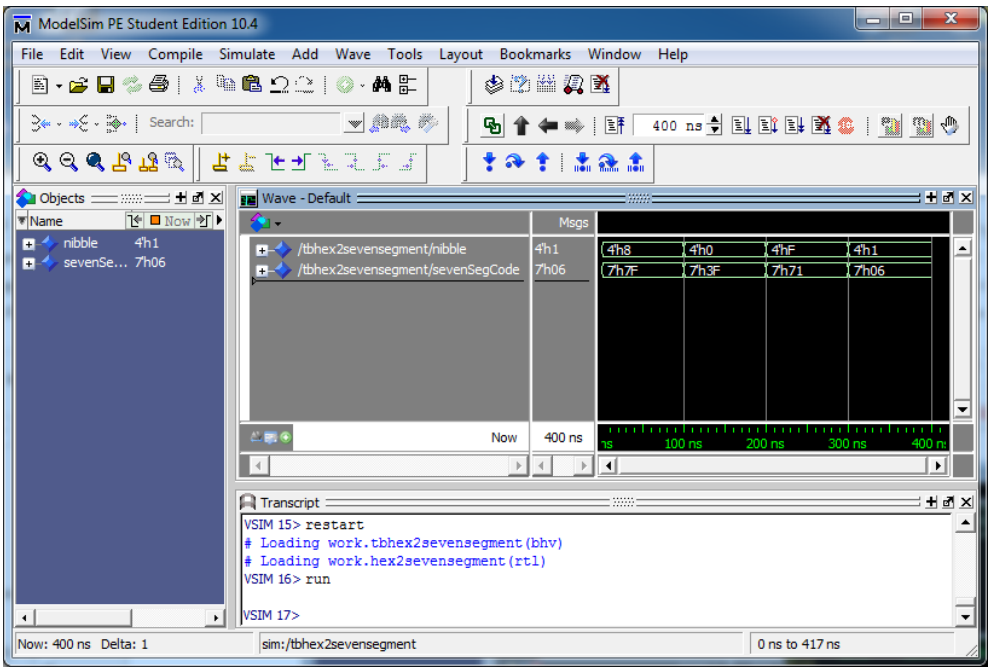
Die Vorlesung und Übung CHD3 hingegen behandelt synthesefähigen VHDL-Code. Um VHDL-Beschreibungen in einem Synthese-Schritt auf Hardware abzubilden, unterliegt der VHDL-Code gewissen Einschränkungen. Es ist z.B. nur eine Teilemenge der Sprachumfangs erlaubt. Beispielsweise ist ein `wait for` Statement nicht synthesefähig. Grob gesprochen wird in synthesefähigem VHDL-Code festgelegt, welche Werte welchen Registern zu bestimmten Zeitpunkten zugewiesen werden. Man spricht deswegen von RTL-Modellen (Modelle auf *Register Transfer Level*). Das Projekt `HelloDE1` liegt in synthesefähigem VHDL-Code in der entpackten Verzeichnisstruktur in den Verzeichnissen `unitHex2SevenSegment`, `unitHelloDE1` und `unitTbdHelloDE1`. In jedem dieser Verzeichnisse sind die VHDL-Dateien zu den gleichnamigen Units enthalten:

unitHex2SevenSegment

Im Verzeichnis `unitHex2SevenSegment/src` befinden sich *entity* und *architecture* eines Decodieres, der eine 4-Bit Dualzahl in einen 7-Bit Code zur Ansteuerung einer Siebensegment-Anzeige umsetzt.

Der Decoder `Hex2SevenSegment` ist in den Dateien
`Hex2SevenSegment-e.vhd` und
`Hex2SevenSegment-Rtl-a.vhd` beschrieben.

Vor der Weiterverwendung der *entity/architecture* muss die Funktionalität durch Simulation in einer Testbench überprüft werden, die hier in der Datei `tbHex2SevenSegment-ea.vhd` definiert ist. Das nachstehende Bild zeigt die Simulation dieser Testbench.



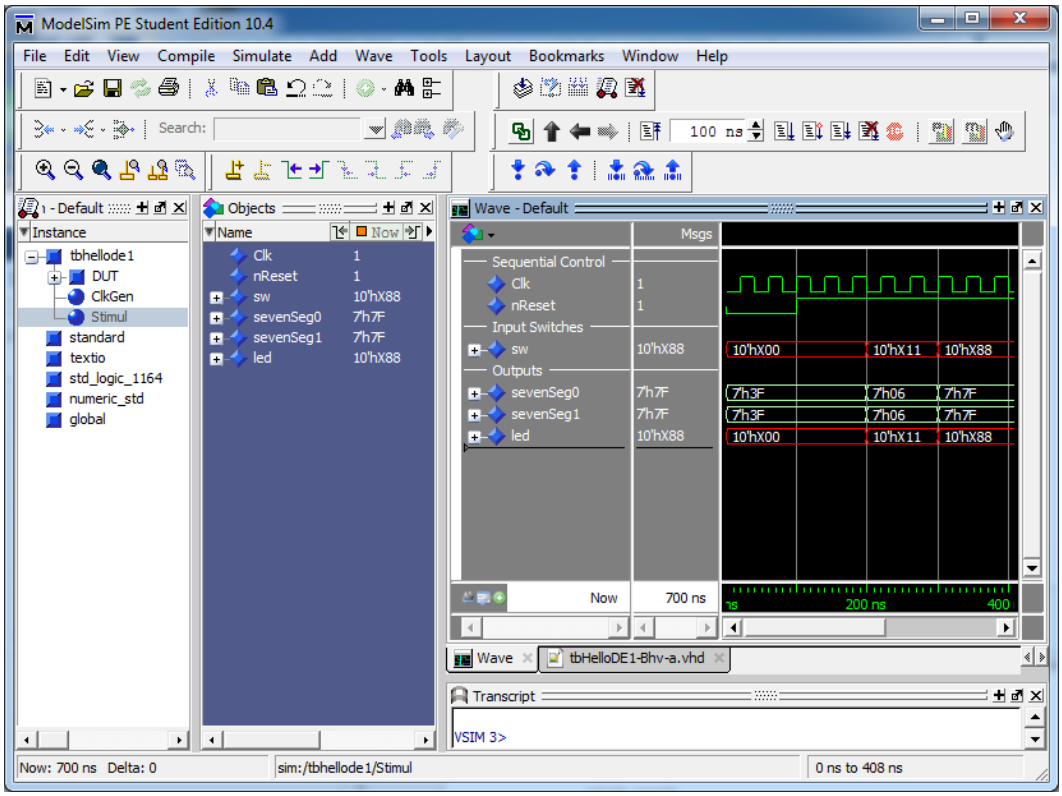
unitHelloDE1

Die folgenden Dateien beschreiben die Unit und die zugehörigen Testbench:

```
HelloDE1-e.vhd
HelloDE1-Rtl-a.vhd
tbHelloDE1-Bhv-a.vhd
tbHelloDE1-e.vhd
```

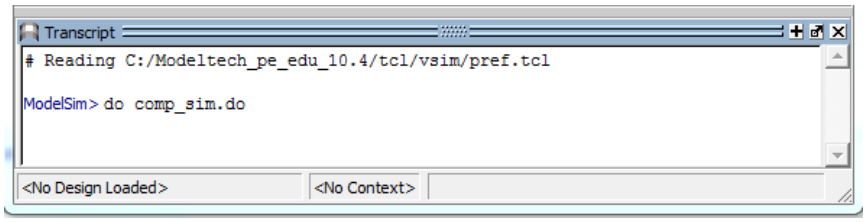
Die *architecture* `RTL` von `HelloDE1` verwendet zwei Instanzen von `Hex2SevenSegment` zur Konvertierung von acht Schalterstellungen (`iSW(0)` bis `iSW(7)`) in Code für zwei Siebensegment-Anzeigen (`o7SegCode0` und `o7SegCode1`). Zusätzlich wird in einem separaten Prozess mit Hilfe eines Counters das Taktsignal `iClock` geteilt, um die `oLed(9)` blinken zu lassen.

Die Dateien `tbHelloDE1-Bhv-a.vhd` und `tbHelloDE1-e.vhd` beschreiben eine Testbench zur Validierung von `HelloDE1`.



DO-Files

Modelsim und Questasim erlauben die Ausführung von Skript-Dateien, sogenannten DO-Files. Für die Unit `HelloDE1` sind beispielhaft zwei DO-Files definiert, die neben dem Compile-Schritt auch die Waveform für die Simulation einrichten. Der Aufruf erfolgt im Transcript-Fenster:



comp_sim.do

```
vlib work
vcom -work work ../../../grpPackages/pkgGlobal/src/Global-p.vhd
vcom -work work ../../../grpChd2/unitHex2SevenSegment/src/Hex2SevenSegment-e.vhd
vcom -work work ../../../grpChd2/unitHex2SevenSegment/src/Hex2SevenSegment-Rtl-a.vhd
vcom -work work ../../../grpChd2/unitHelloDE1/src/HelloDE1-e.vhd
vcom -work work ../../../grpChd2/unitHelloDE1/src/HelloDE1-Rtl-a.vhd
vcom -work work ../../../grpChd2/unitHelloDE1/src/tbHelloDE1-e.vhd
vcom -work work ../../../grpChd2/unitHelloDE1/src/tbHelloDE1-Bhv-a.vhd
vsim -voptargs=+acc work.tbHelloDE1(Bhv)
do wave.do
run 1000 ms
```

wave.do

```
onerror {resume}
quietly WaveActivateNextPane {} 0
add wave -divider "Sequential Control"
add wave -noupdate -format Logic /tbhellode1/Clk
add wave -noupdate -format Logic /tbhellode1/nReset
add wave -divider "Input Switches"
add wave -noupdate -format Logic /tbhellode1/SW
add wave -divider Outputs
add wave -noupdate -format Logic /tbhellode1/sevenSeg0
add wave -noupdate -format Logic /tbhellode1/sevenSeg1
add wave -noupdate -format Logic /tbhellode1/led
TreeUpdate [SetDefaultTree]
WaveRestoreCursors {{Cursor 1} {100 ns} 0}
configure wave -namecolwidth 132
configure wave -valuecolwidth 54
…
configure wave -timeline 0
configure wave -timelineunits ns
update
WaveRestoreZoom {0 ns} {1000 ns}
```

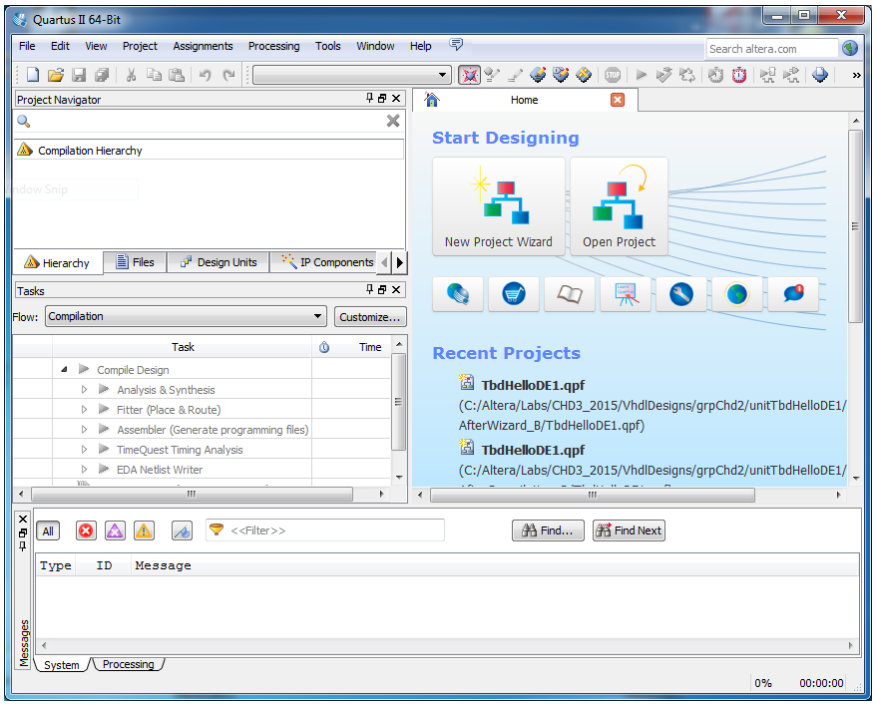
unitTbdHelloDE1

Die *entity* von `TbdHelloDE1` (Tbd: Testbed) ist die *topl-level entity* des Projekts. Die *entity* und *architecture* von `HelloDE1` sind unabhängig von einem physischen FPGA und einem zugehörigen Board. Das Testbed `TbdHelloDE1` erledigt die spezifischen Anpassungen an das DE1-SoC Board. Es werden z.B. die Input- und Output-Ports der *entity* `HelloDE1` den Leitungsnamen des Boards zugeordnet. Die Details dazu werden später im Kapitel *C2 Pin Assignment* näher beschrieben.

Laut Datenblatt des DE1-SoC Boards sind die Siebensegment-Anzeigen mit negativer Logik (*low active*) anzusteuern. Die Anpassung der Ausgabewerte von `Hex2SevenSegment` (positive Logik) an das Board erfolgt ebenfalls in der *architecture* von `TbdHelloDE1`.

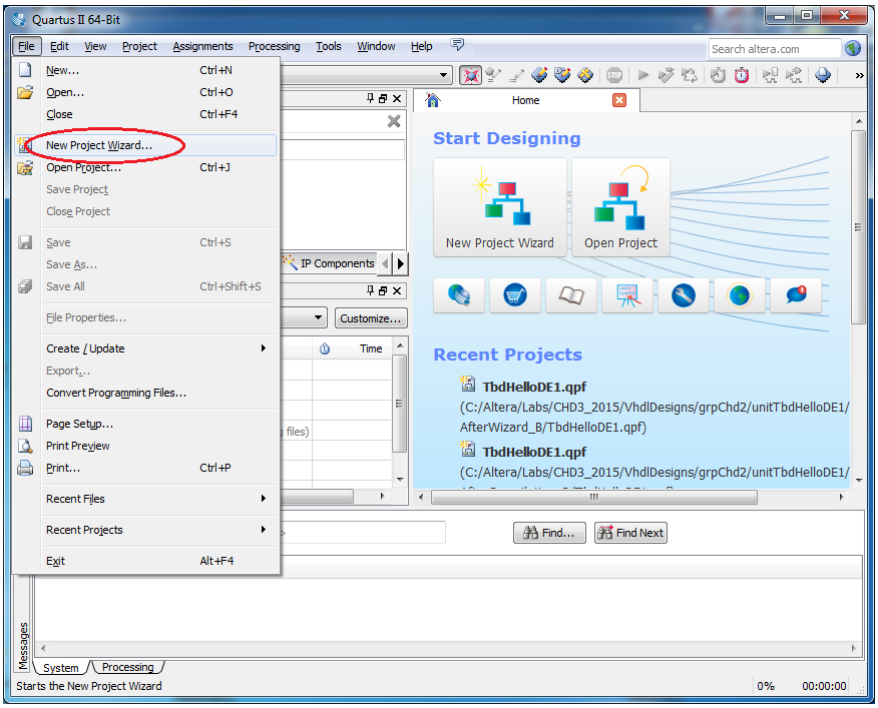
C Synthese von tbHelloDE1 für das DE1-SOC Board

Nach dem Start von Quartus II mit Hilfe des Shortcuts öffnet sich das Hauptfenster.

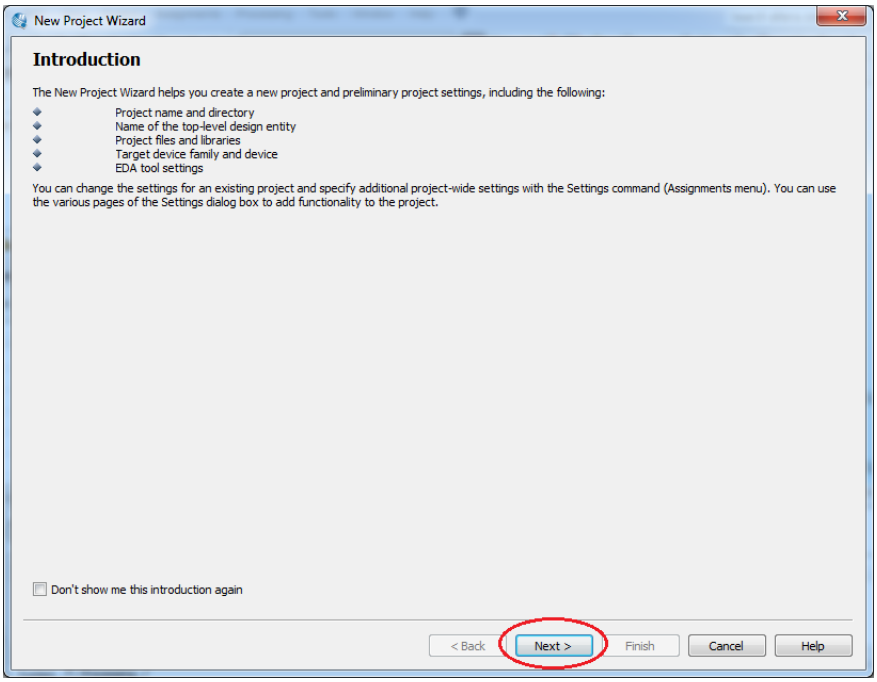


C1 Anlegen eines Projektes

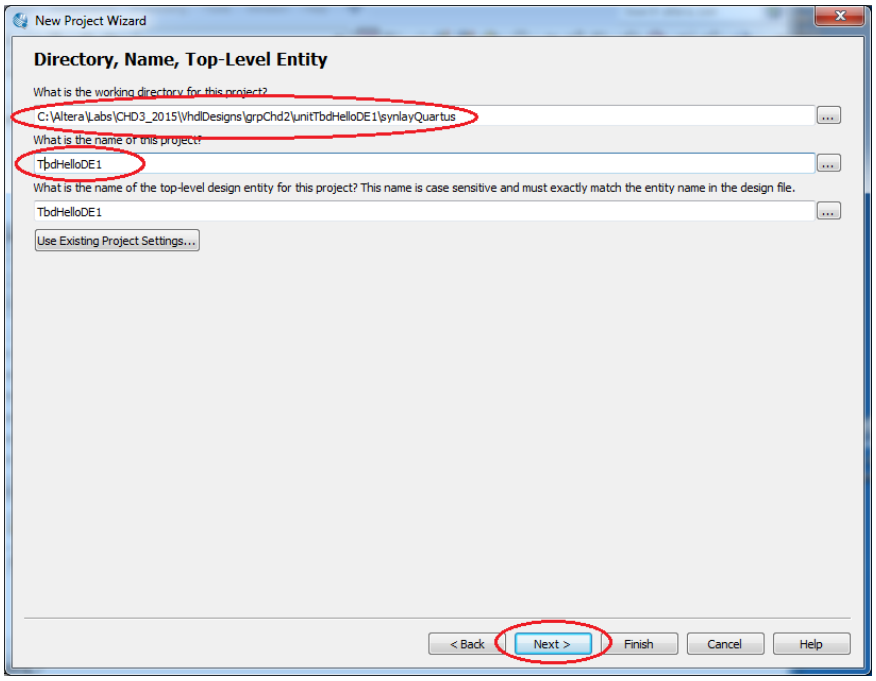
Mit dem *New Project Wizard* wird ein neues Projekt erzeugt und für das Projekt `TbdHelloDE1` konfiguriert:



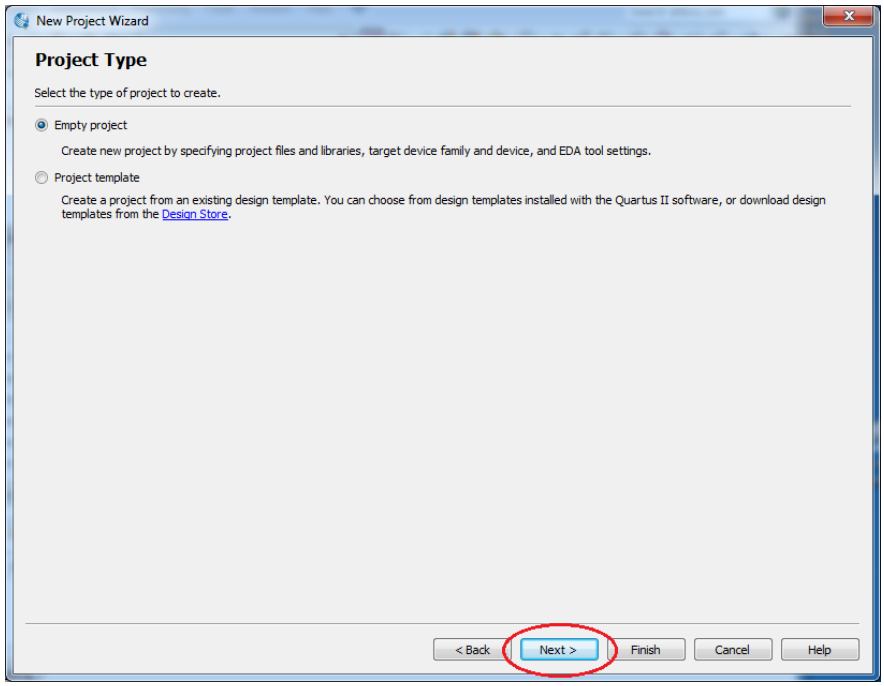
Das erste Fenster des *Wizard* gibt einen Überblick über die einzustellenden Parameter.



Zuerst wird das Arbeitsverzeichnis des Projekts (`../unitTbdHelloDE1/synLayQuartus`) und der Projektname `TbdHelloDE1` angegeben. Quartus nimmt als Namen der *top-level entity* des Projekts den Projektnamen an. Man sollte diesem Vorschlag folgen und das Projekt entsprechend benennen.

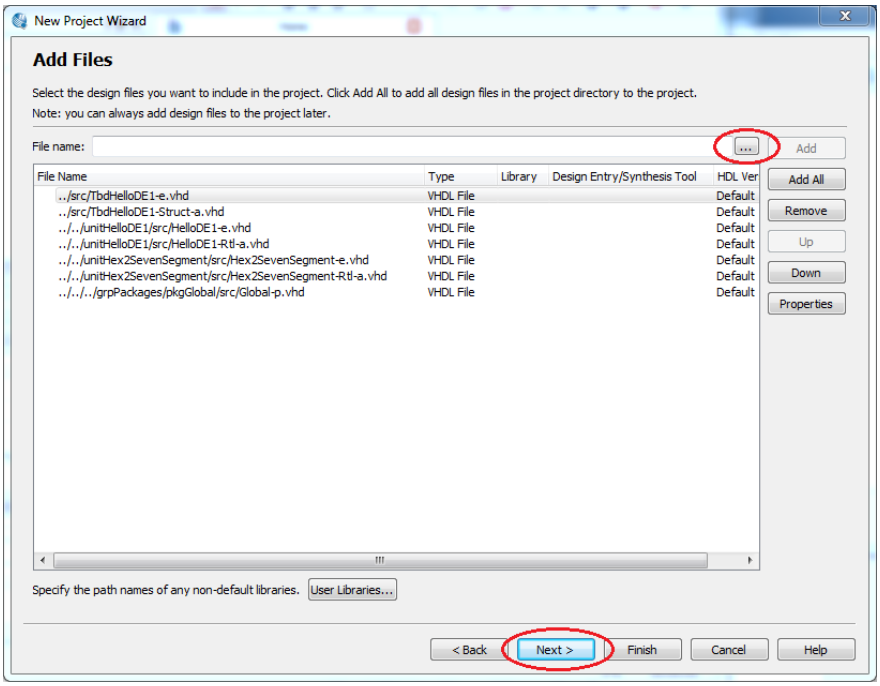


Wir starten ohne Template und wählen ein leeres Projekt.

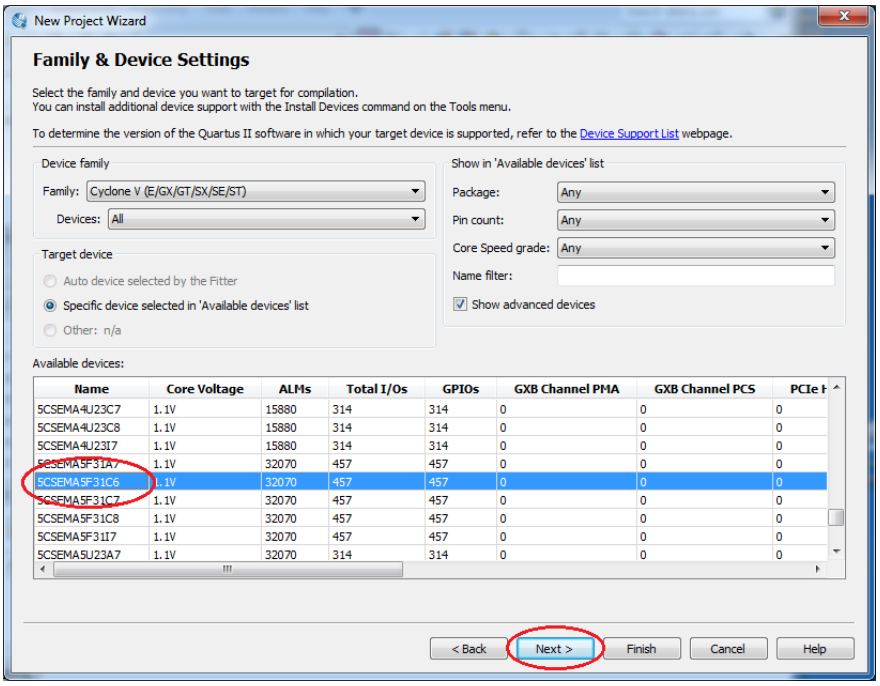


Im nächsten Fenster werden die VHDL-Dateien des Projektes ausgewählt, die für die Synthese relevant sind (Testbenches sind deshalb nicht anzugeben):

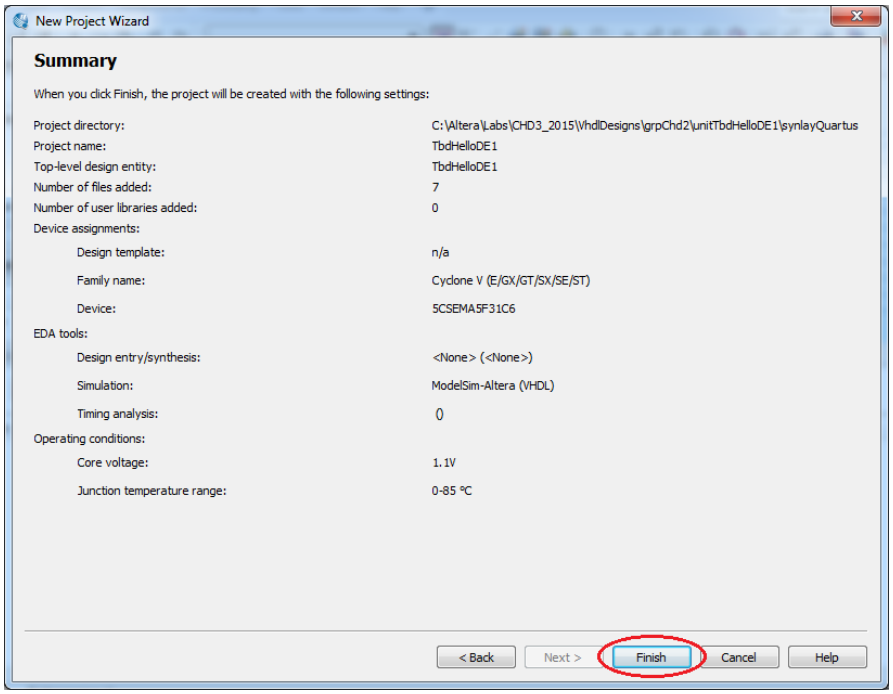
- `Global-p.vhd`
- `Hex2SevenSegment-e.vhd`
- `Hex2SevenSegment-Rtl-a.vhd`
- `HelloDE1-e.vhd`
- `HelloDE1-Rtl-a.vhd`
- `TbdHelloDE1-e.vhd`
- `TbdHelloDE1-Struct-a.vhd`



Jetzt muss die Ziel-Hardware angegeben werden. Wir wählen das FPGA, das sich auf dem DE1-SoC Board befindet (vgl. Aufdruck auf dem Chipgehäuse): `Cyclone V 5CSEMA5F31C6`.



Für das Projekt `TbdHelloDE1` sind im nächsten Fenster keine Eingaben erforderlich: also weiter mit *Next*. Das letzte Fenster gibt eine Zusammenfassung der Eingaben und ermöglicht eine abschließende Kontrolle.



Quartus legt eine Projektdatei `TbdHelloDE1.qpf` (**Q**uartus **P**roject **F**ile) an, die nach einem Neustart von Quartus geladen werden kann. Die Datei `TbdHelloDE1.qws` (**Q**uartus **W**orkspace **F**ile) ist ebenfalls nicht editierbar und enthält Informationen z.B. über Fenster-Layout und User-Einstellungen.

Die Datei `TbdHelloDE1.qsf` (**Q**uartus **S**ettings **F**ile) wird ebenfalls angelegt, ist jedoch lesbar und enthält zum momentanen Zeitpunkt folgende Daten:

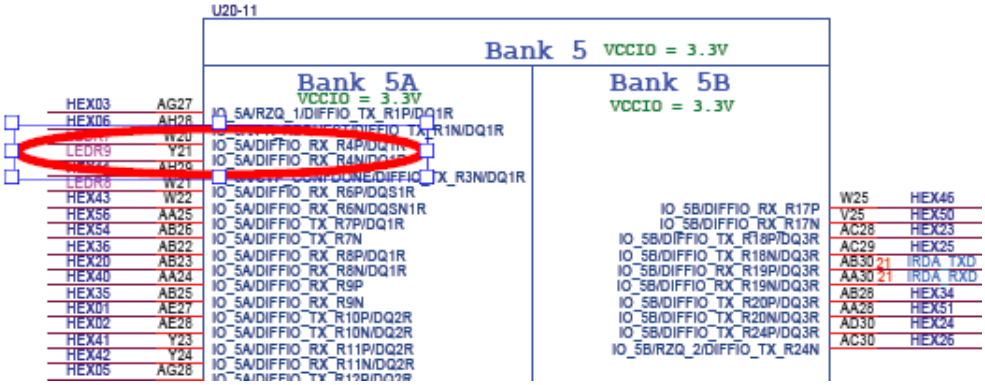
TbdHelloDE1.qsf

```
set_global_assignment -name FAMILY "Cyclone V"
set_global_assignment -name DEVICE 5CSEMA5F31C6
set_global_assignment -name TOP_LEVEL_ENTITY TbdHelloDE1
set_global_assignment -name ORIGINAL_QUARTUS_VERSION 15.0.0
set_global_assignment -name PROJECT_CREATION_TIME_DATE "16:33:55  MAY 19, 2015"
set_global_assignment -name LAST_QUARTUS_VERSION 15.0.0
set_global_assignment -name VHDL_FILE "../src/TbdHelloDE1-Struct-a.vhd"
set_global_assignment -name VHDL_FILE "../src/TbdHelloDE1-e.vhd"
set_global_assignment -name VHDL_FILE "../../unitHelloDE1/src/HelloDE1-Rtl-a.vhd"
set_global_assignment -name VHDL_FILE "../../unitHelloDE1/src/HelloDE1-e.vhd"
set_global_assignment -name VHDL_FILE
      "../../unitHex2SevenSegment/src/Hex2SevenSegment-Rtl-a.vhd"
set_global_assignment -name VHDL_FILE "../../unitHex2SevenSegment/src/Hex2SevenSegment-e.vhd"
set_global_assignment -name VHDL_FILE "../../../grpPackages/pkgGlobal/src/Global-p.vhd"
set_global_assignment -name PROJECT_OUTPUT_DIRECTORY output_files
set_global_assignment -name MIN_CORE_JUNCTION_TEMP 0
set_global_assignment -name MAX_CORE_JUNCTION_TEMP 85
set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 256
set_global_assignment -name EDA_SIMULATION_TOOL "ModelSim-Altera (VHDL)"
set_global_assignment -name EDA_OUTPUT_DATA_FORMAT VHDL -section_id eda_simulation
```

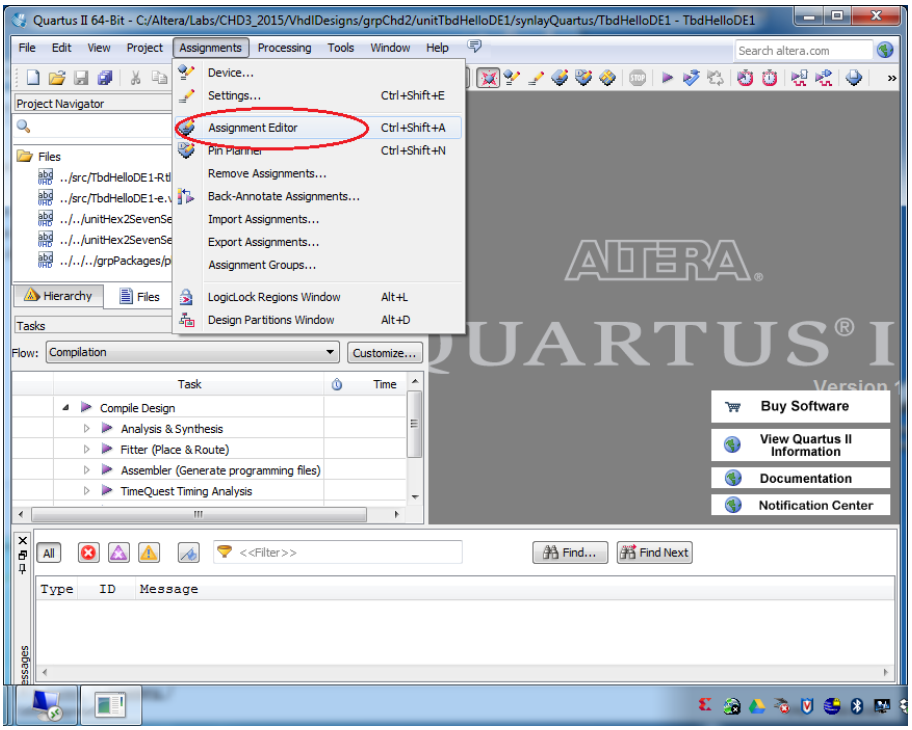
C2 Pin Assignments

Der Output-Port `LEDR(9)` der *entity* `TbdHelloDE1` erzeugt ein Signal, das die LED mit Nummer 9 des DE1-SoC Boards blinken lassen soll. Aber woher weiß Quartus, ob und wie der VHDL-Name `LEDR(9)` mit dieser LED zu verbinden ist?

1. Die physikalischen Anschlüsse (Pins) des FPGA und deren Verbindungen zu Komponenten des Boards (LEDR9) sind im Schaltplan dokumentiert. Auf Seite 4 findet sich der Pin des FPGAs, der mit der physikalischen LED mit dem Namen *LEDR9* verbunden ist: **`Y21`**.

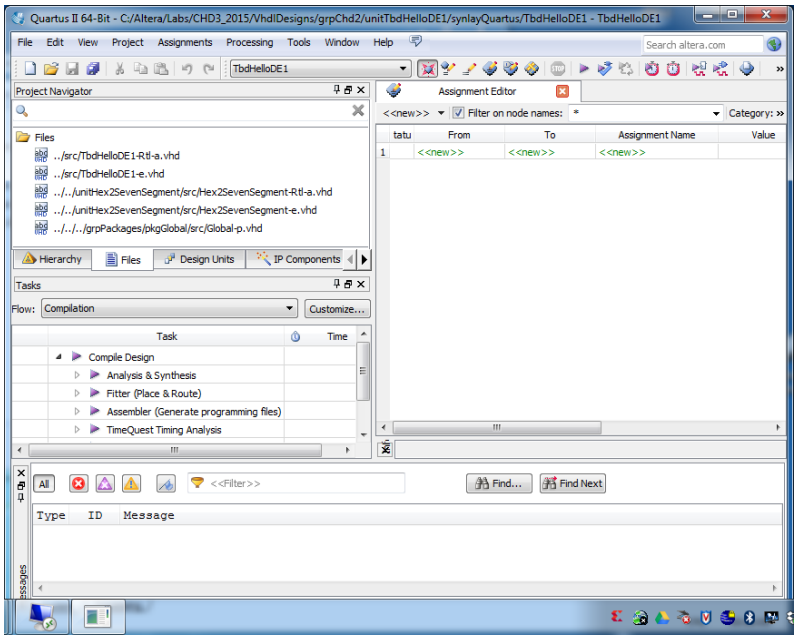


2. Die Zuordnung der FPGA-Pins zu Port-Namen der *top-level entity* `TbdHelloDE1` erfolgt in Quartus mit dem *Assignment Editor*:



Bis jetzt haben wir für das Projekt nur den Typ des FPGA (`5CSEMAF31C6`) angegeben, aber noch keine Angaben über die Verdrahtung des FPGA mit den Komponenten des Boards gemacht.

Deswegen kennt der *Assignment Editor* nach dem Start noch keine Zuordnung der Ports des Designs zu Pins des FPGAs:



Die mühevolle Arbeit, alle Pins im Schaltplan suchen und einzeln im *Assignment Editor* eingeben zu müssen, bleibt uns hier erspart. Die Verdrahtung der Board-Komponenten mit FPGA-Pins ist im *constraint file* `TbdHelloDE1Pins.qsf` – im Quartus-Settings-File-Format – zusammengefasst und kann eingelesen werden.

Diese Pin-Zuweisungen bilden nur einen Ausschnitt der vollständigen QSF-Datei des gesamten Boards. Mit dem *SystemBuilder* (auf der Terasic CD-ROM: `DE1-SoC_v.4.0.2_HWrevE_SystemCD/Tools/SystemBuilder/DE1SoC_SystemBuilder.exe`) können weitere spezifische Versionen dieser Datei erstellt werden.

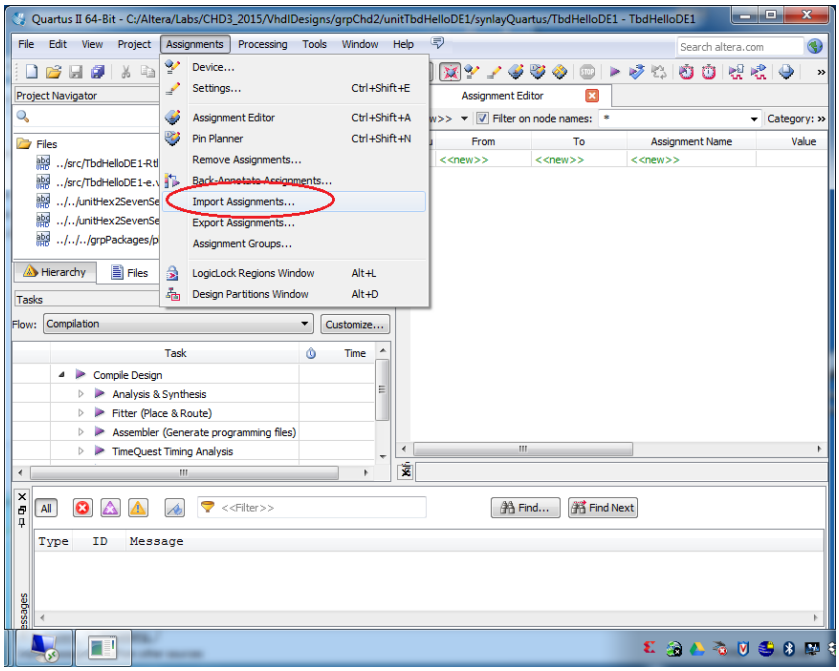
Achtung: Der *SystemBuilder* verknüpft die Pins des FPGA mit den Namen der Komponenten des Boards, wobei nicht z.B. zwischen LEDR9 und LEDR[9] unterschieden wird. Wir haben also drei Versionen: `LEDR9` im Schaltplan, `LEDR[9]` im erzeugten QSF-File und `LEDR(9)` als VHDL-Portname in `TbdHelloDE1`.

> **Die QSF-Datei enthält die Zuordnung der Portnamen der *top-level entity* zu den FPGA-Pins, wobei eckige Klammern für Vektor-Elemente verwendet werden.**

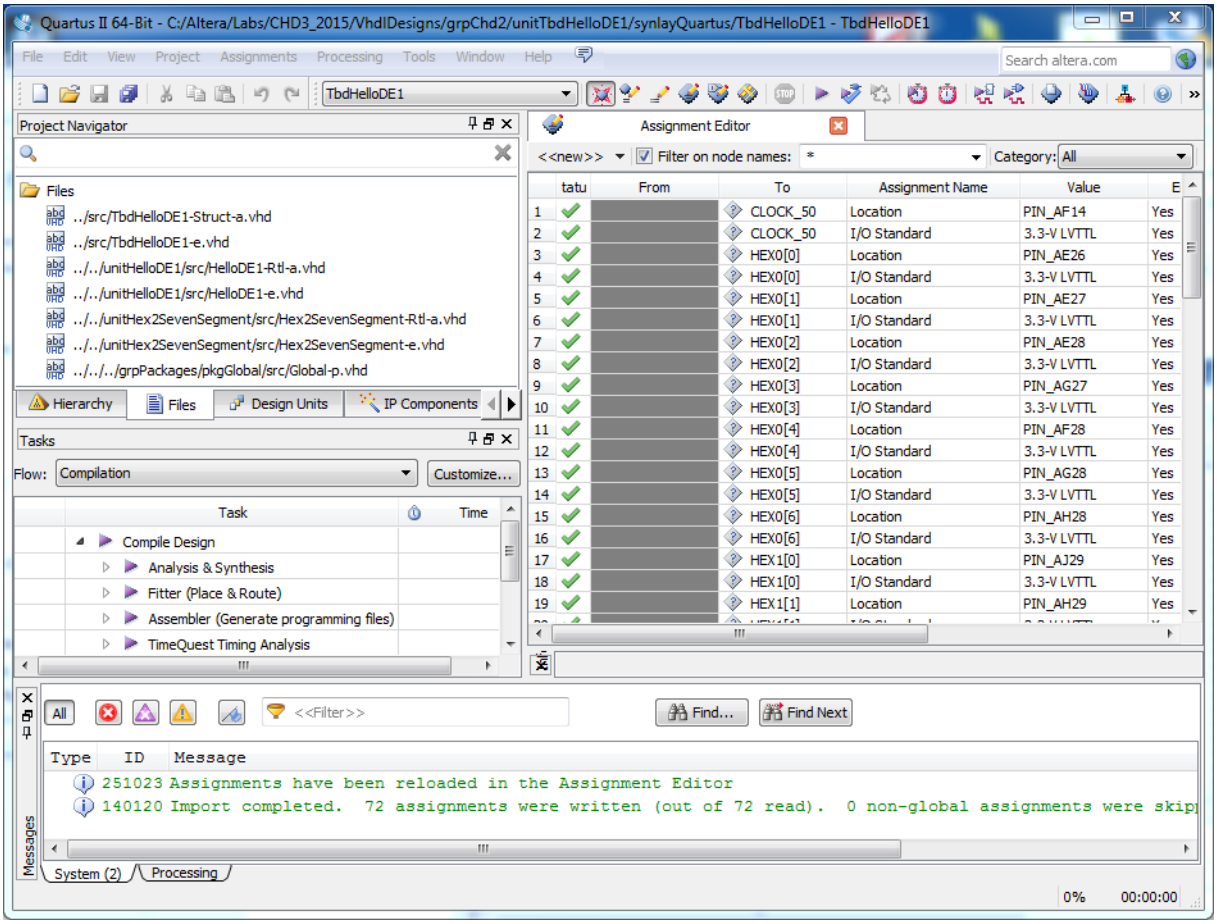
Der *SystemBuider* erzeugt weitere Informationen über die Pin-Verwendung in der QSF-Datei (nähere Informationen dazu wird es in CHD3 geben).

```
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to LEDR[9]
```

Die Datei `TbdHelloDE1Pins.qsf` enthält sämtliche relevante Information über die verwendeten FPGA-Pins. Sie wird in den *Assignment Editor* importiert:

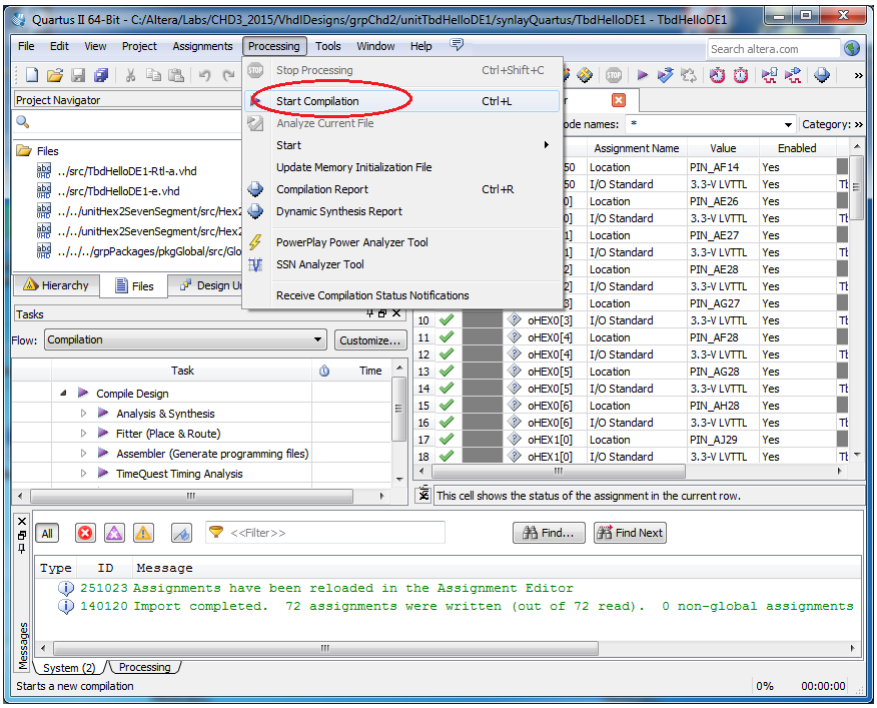


Nach dem Import von `TbdHelloDE1Pins.qsf` können die aufgelisteten Pin-Beschreibungen auf Vollständigkeit überprüft werden. Die Datei `TbdHello1.qsf` enthält jetzt zusätzlich die importierten Pin-Assignments.

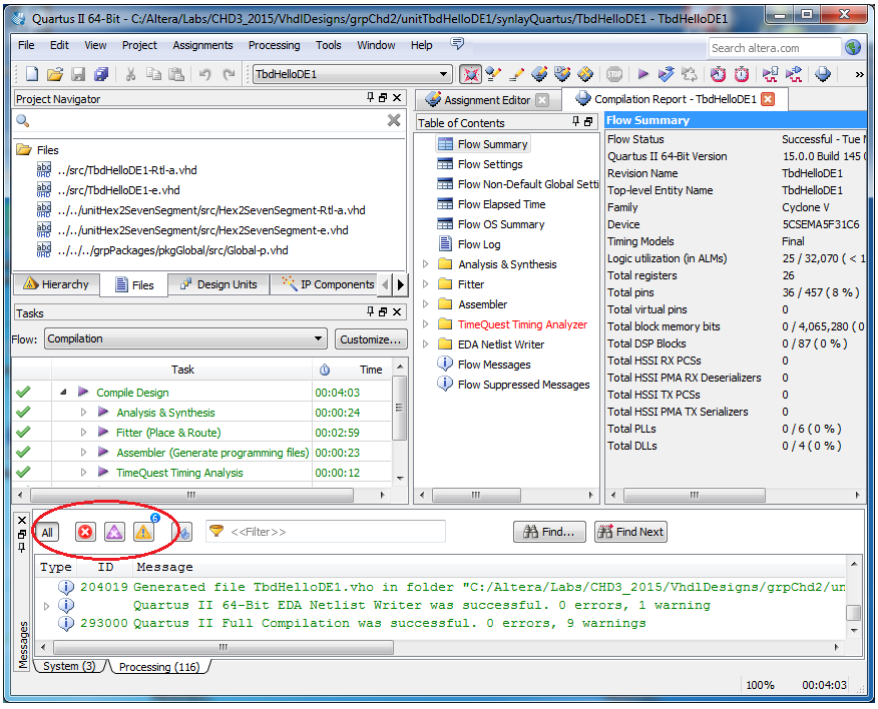


C3 Synthese

Jetzt können wir die Erzeugung einer Konfiguration des FPGAs aus der VHDL-Beschreibung starten. Die Synthese – in Quartus *Compilation* genannt – kann je nach PC-Hardware unterschiedlich lange dauern (hier ca. 4 Minuten).



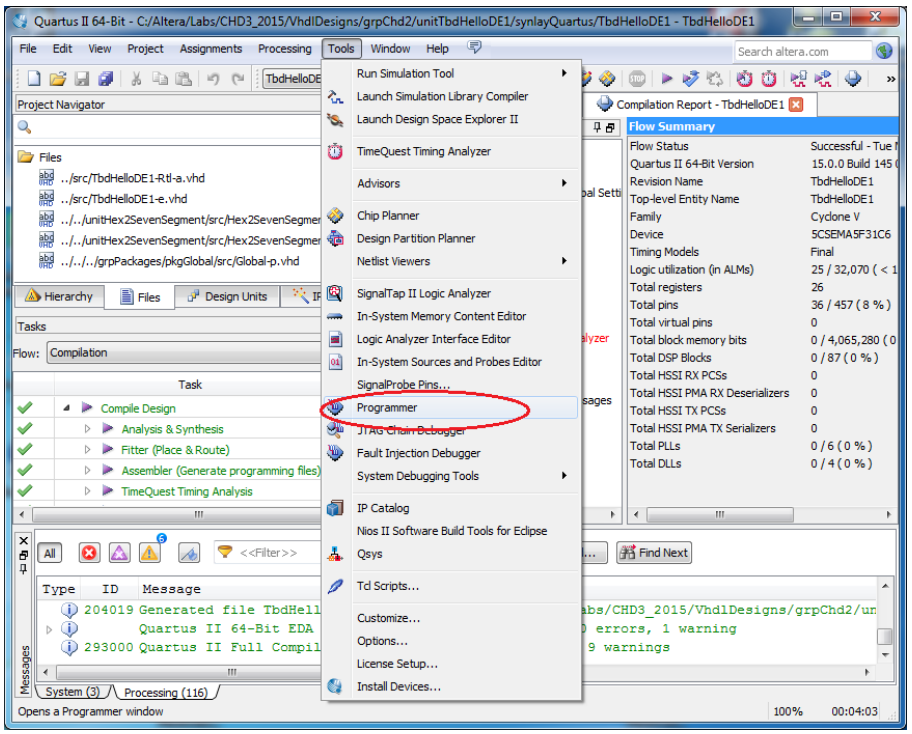
Das Ergebnis der Synthese wird in einer Vielzahl von *Reports* dokumentiert. Falls *Errors* auftreten, liegt kein weiterverwendbares Ergebnis vor. Auf *Critical Warnings* ist unbedingt zu reagieren. In diesem Fall hat die Synthese erfreulicherweise nur sechs Warnings erzeugt, da u.a. die Input-Ports `SW(8)` und `SW(9)` in der *architecture* nicht verwendet werden und `LEDR(8)` nicht gesetzt wird.



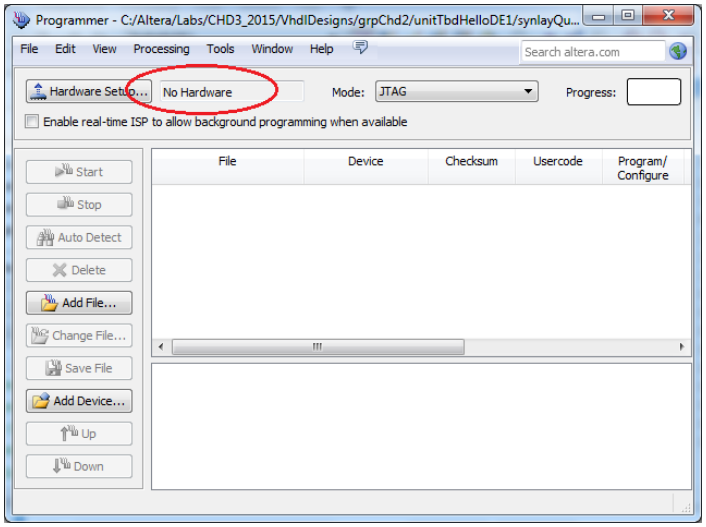
C4 FPGA-Konfiguration

Nach der fehlerfreien Synthese kann die erzeugte Konfigurations-Datei in das FPGA geladen werden. Dazu steht der *Programmer* zur Verfügung. Die Kommunikation mit dem DE1-Soc Board erfolgt mittels des *USB-Blasters II*. Vor dem Start des *Programmers* sollte das Board eingeschaltet sein und der JTAG-Stecker des Boards über ein USB-Kabel mit dem Rechner verbunden sein.

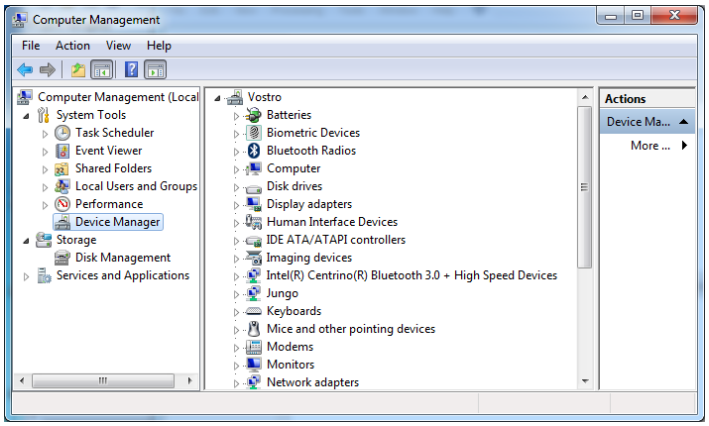
Der *Programmer* wird über das Tools-Menu gestartet:



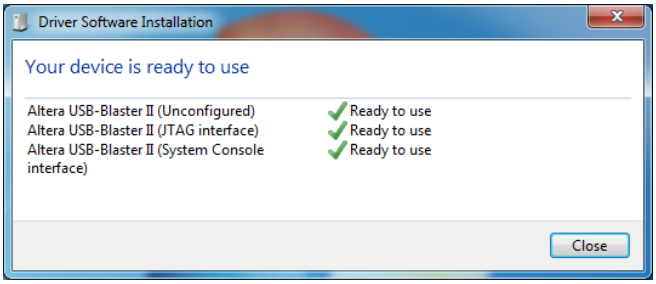
Falls das Board noch nicht eingeschaltet war oder nicht über USB verbunden ist, kann keine Hardware erkannt werden.



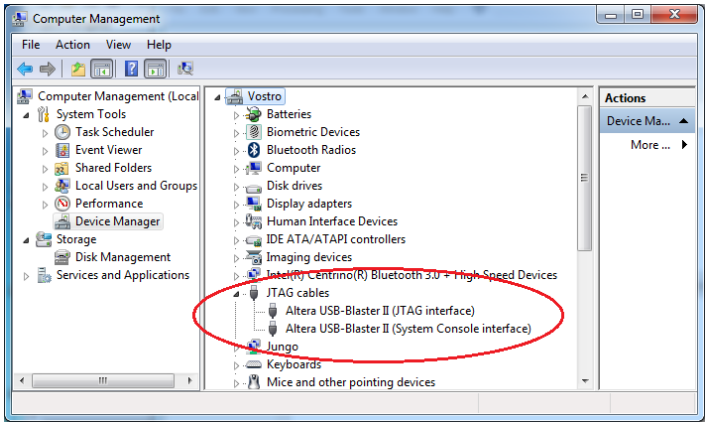
Im Windows Computer Management – Device Manager sind zu diesem Zeitpunkt keine JTAG-Cables sichtbar:



Nach dem ersten Einschalten und nach der Herstellung der USB-Verbindung erfolgt die Installation der USB-Blaster-Treiber automatisch (siehe dazu auch die Anleitung zur Installation von Quartus am Beginn dieser Übung). Danach wird die Verbindung zum Board hergestellt.

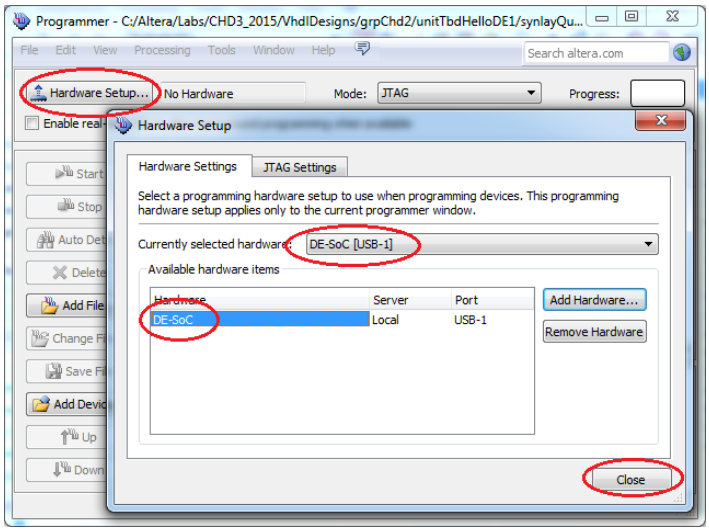


Bei jedem Einschalten des Boards (auch nach der Installation der Treiber), wird eine USB-Verbindung hergestellt und die USB-Blaster-Treiber im Computer Management angezeigt (als JTAG cables).

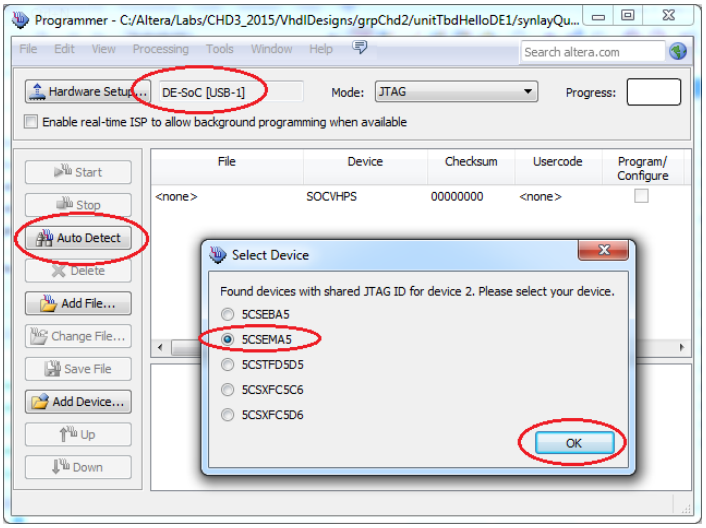


Wenn beim Start des *Programmers* keine Hardware verfügbar war, kann im *Hardware Setup* das Board nachträglich ausgewählt werden – vorausgesetzt, dass die Treiber korrekt installiert, das Board eingeschaltet und das USB-Kabel richtig angeschlossen ist.

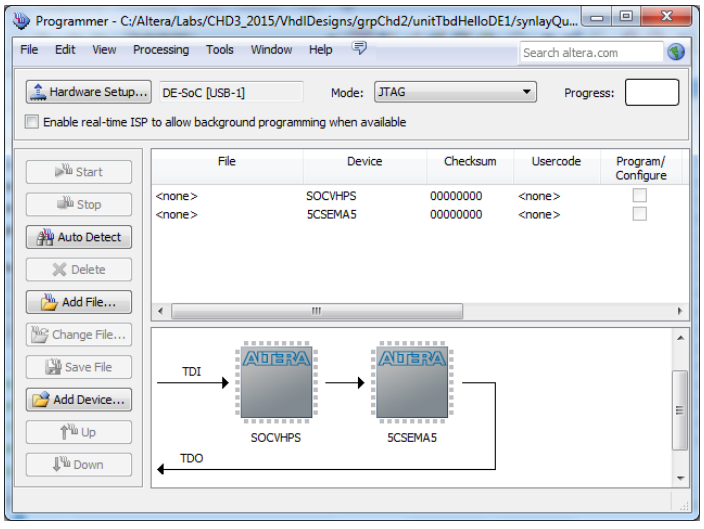
Nach dem Einschalten, Herstellen der USB-Verbindung Auswahl von *Hardware Setup*:



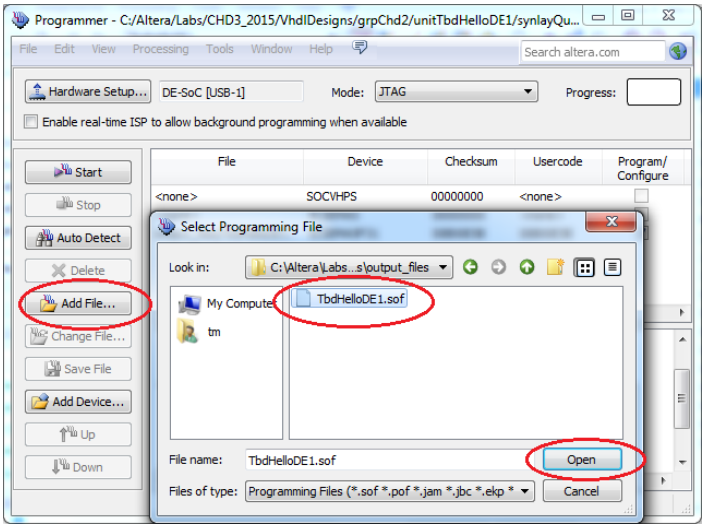
Im Regelfall werden noch keine Devices angezeigt. Mit *Auto Detect* werden verfügbare Devices gesucht. Das HPS (Hard-Processor-System = ARM9-Prozessoren) wird automatisch erkannt, das FPGA muss durch Auswahl von `5CSEMA5` spezifiziert werden.



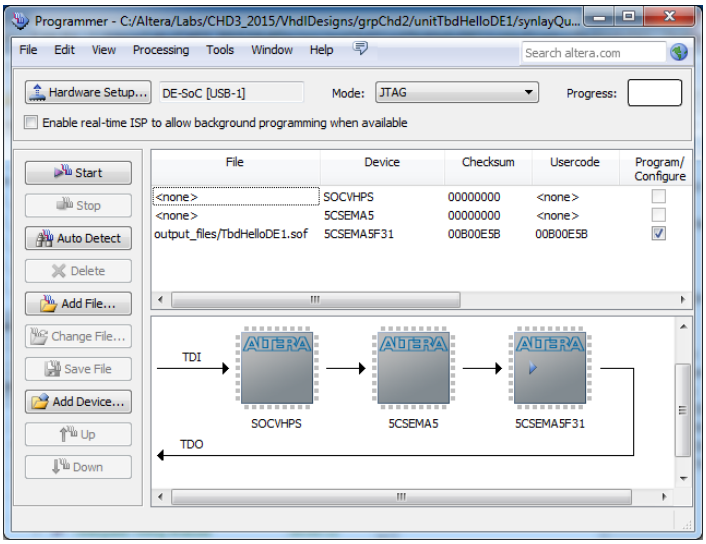
Danach werden zwei Devices in einer JTAG-Chain angezeigt.



Das HPS wird in dieser Übung nicht benutzt. Mit dem Menu *Add File* wird die Output-Datei der Synthese `TbdHelloDE1.sof` ausgewählt (SOF: SRAM Object File). Quartus hat diese Datei im Verzeichnis `/unitTbdHelloDE1/synlayQuartus/output_files/` erzeugt.

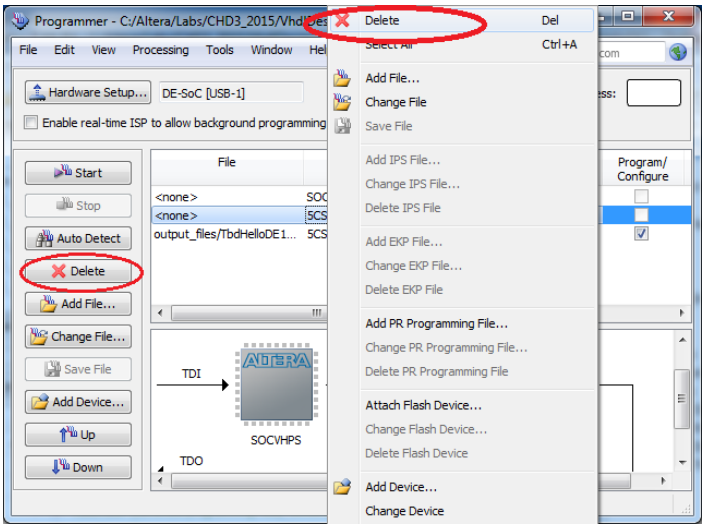


Durch `TbdHelloDE1.sof` wird auch das Ziel-FPGA genau spezifiziert und - leider - als drittes Device in der JTAG-Chain angezeigt.

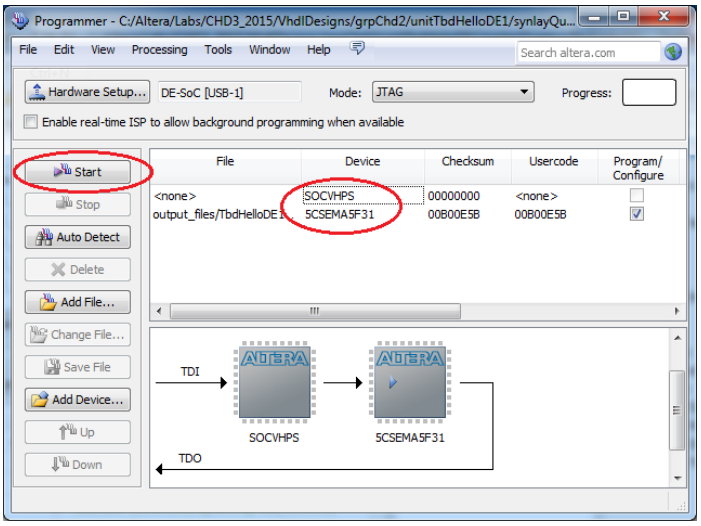


Dieses dritte Device muss vor der Programmierung gelöscht werden. Entweder mit dem Menu der rechten Maustaste (siehe nächstes Bild) oder mit dem Menu-Eintrag *Delete* am rechten Rand.

Löschen des dritten Device aus der JTAG-Chain:



Die Reihenfolge der Devices ist wichtig. Mit *Up* und *Down* kann die Reihenfolge der Devices geändert werden. Nur mit der **korrekten Reihenfolge** (siehe nächstes Bild) kann die Programmierung erfolgreich durchgeführt werden. Gestartet wird die Konfiguration des FPGA mit dem Button *Start*.



Wird das nachstehende Fenster angezeigt und beginnt die LED zu blinken, gratulieren wir zur ersten erfolgreichen FPGA-Konfiguration!