Tutorial Einstieg in den miniSpartan6+ LX9 von Scarab Hardware

Inhalt

Abbildungsverzeichnis

Schritt 1: Entwicklungsumgebung ISE Webpack von Xilinx herunterladen

Zuerst muss die Entwicklungsumgebung heruntergeladen werden. Hier in der kostenfreien Variante ‚ISE Webpack':

https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/design-tools.html

https://www.xilinx.com/products/design-tools/hardware-zone.html

Achtung: Jede Software von Xilinx benötigt eine Registrierung und Lizensierung:

- Eine Registrierung wird vor dem ersten Download benötigt. Login- und Personendaten müssen vor jedem Download erneut eingegeben werden.
- Eine persönliche Lizenz kann vor der Installation im .lic Format von Xilinx heruntergeladen oder aus dem installierten Produkt heraus (Manage Xilinx Licenses) beantragt werden.

Die Entwicklung der ISE wurde 2013 eingestellt, was aber aufgrund des enorm hohen Entwicklungsstands der IDE keinerlei Nachteile mit sich bringt. Für FPGA's ab der Generation 7 existiert ein Update namens ‚Vivando'.

Schritt 2: Nach dem Download

Die ISE Entwicklungsumgebung ist sehr umfangreich und benötigt ca. 17GB (!) Festplattenspeicherplatz.

Schritt 3: Ein Neues Projekt anlegen

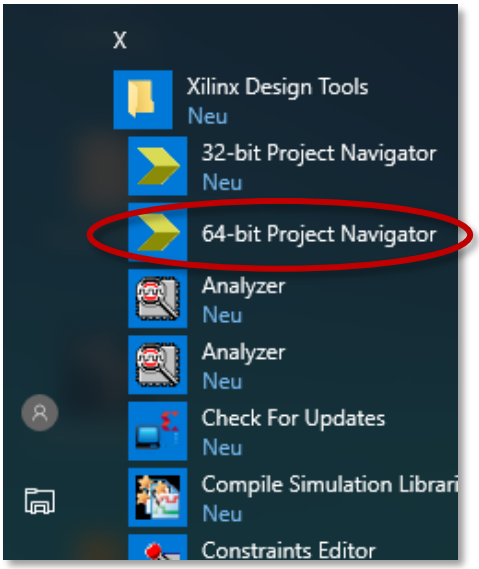


Abbildung 1: Projekt Navigator

Nach der Installation findet man einen Ordner mit 16 Einzelprogrammen. Man startet den ISE Project Navigator, und nach dem Start wird ein neues Projekt angelegt.

Dabei sind zwei Punkte zu beachten: die Auswahl des richtigen FPGA-Chips und des Einstiegspunktes. Die gängigsten Einstiege sind entweder über „Schematic" für den Schaltungsentwurf oder über „HDL" für VHDL-Scripting. Im Beispiel wurde Schematic ausgewählt.

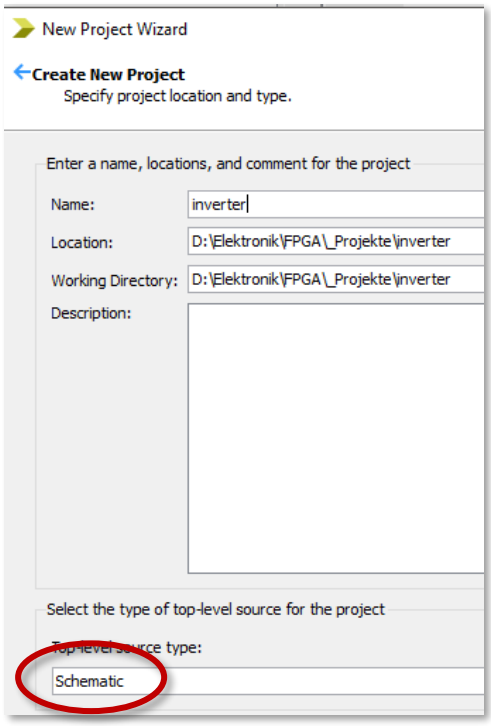


Abbildung 2: Neues Projekt Schematic

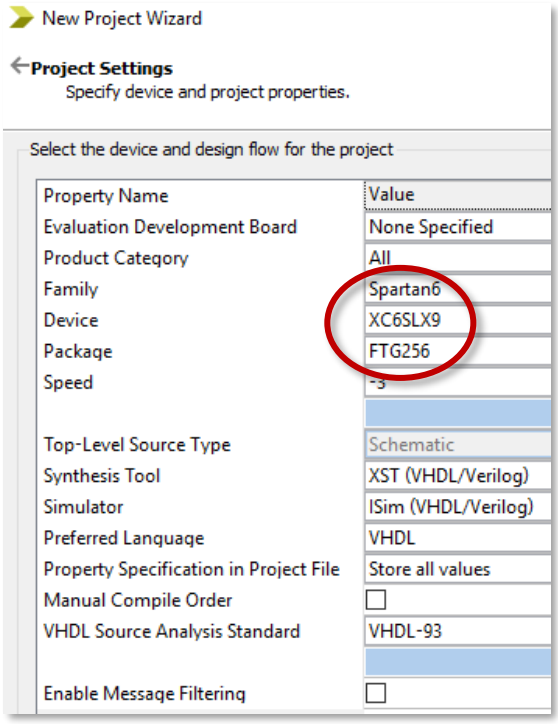


Abbildung 3: Auswahl Chip

Schritt 4: Schemaentwurf

Alle nun folgenden Dateien werden als „Source“ angelegt.

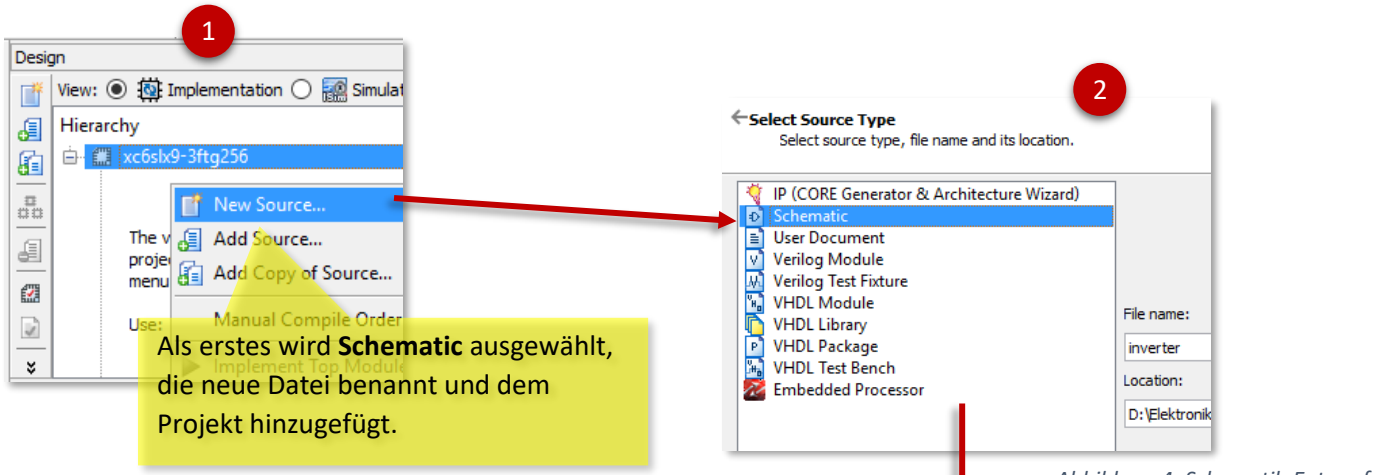
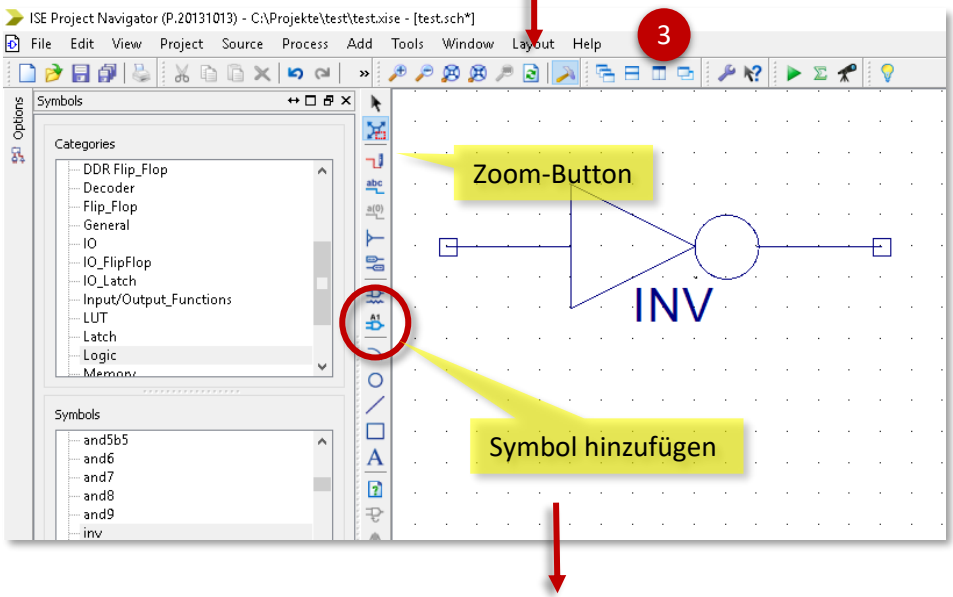


Abbildung 4: Schematik-Entwurf

In der erstellten leeren Datei kann man anfangen, **ein Schema zu bauen.**

Druck auf „Esc“ hebt die aktuelle Auswahl auf.



Die Ein- und Ausgänge müssen mit Markern (I/O Marker) versehen werden:

Die Standardauswahl sieht die automatische Erkennung von Ein- oder Ausgängen und die automatische Benamung vor.

Die Marker werden an dem jeweiligen Anschluss angesetzt und abgelegt. Anschließend müssen die Anschlüsse umbenannt werden, z.B. mit „in_“ oder „out_“.

Die Marker , sind hier ***exemplarisch*** benamt mit „in_D1“ und „out_B1“ und symbolisieren die gewünschten Ports
(Port: Anschluss auf dem Board. Pin: Anschluss auf dem Chip).

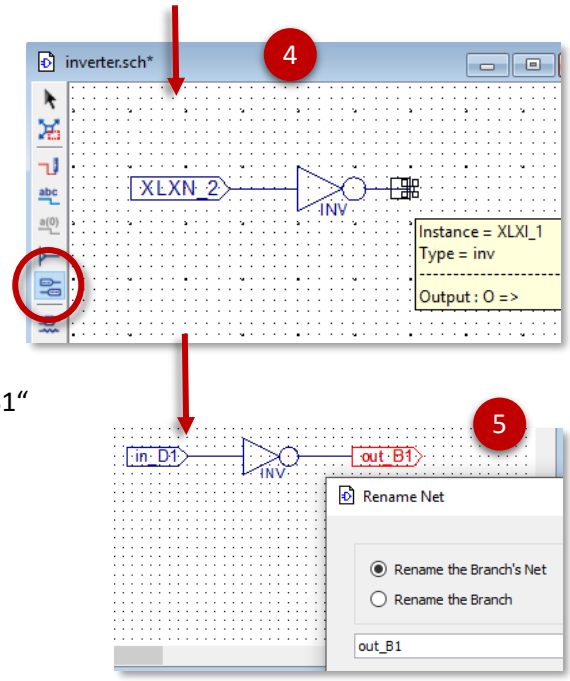


Abbildung 5: Marker für Pins

Schritt 4: Simulation mit ISim

Das fertige Schema kann vor der Anwendung auf der Hardware im ISE simuliert werden. Dafür benötigt man zuerst die sogenannte „VHDL Test Bench“-Datei.

Diese wird ebenfalls über ‚New Source‘ erzeugt.

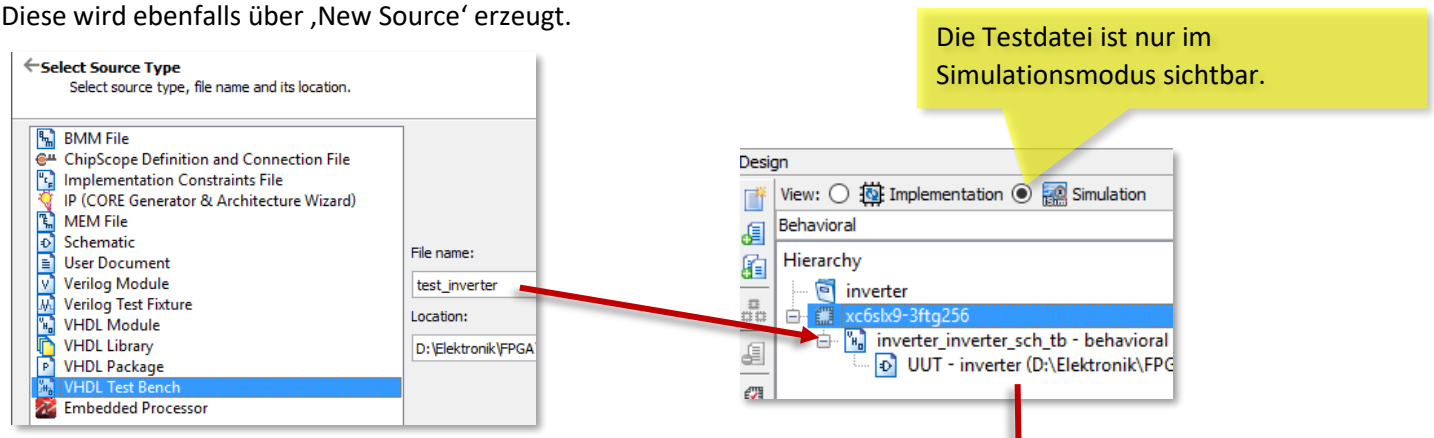


Abbildung 6: Erstellung/ Anpassung Testbench-Datei

Die neu erstellte Datei beinhaltet schon das Grundgerüst, was sich aus der zugrundeliegenden Schemadatei ableitet. Angepasst wird nur der „PROCESS“-Bereich. Hier ordnet man einen Eingangswert den oben definierten Signalen zu oder gleich mehrere Sätze an Eingangswerten. Mit diesen Eingangswerten wird das Schema abgearbeitet. Die Simulationszustände werden durch kurze Wartezeiten im Nanosekundenbereich voneinander getrennt.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY inverter_inverter_sch_tb IS
END inverter_inverter_sch_tb;
ARCHITECTURE behavioral OF inverter_inverter_sch_tb IS

   COMPONENT inverter
   PORT( in_D1 :  IN STD_LOGIC;
         out_B1 :  OUT   STD_LOGIC);
   END COMPONENT;

   SIGNAL in_D1   :  STD_LOGIC;
   SIGNAL out_B1  :  STD_LOGIC;

BEGIN

   UUT: inverter PORT MAP(
      in_D1 => in_D1,
      out_B1 => out_B1
   );

-- *** Test Bench - User Defined Section ***
   tb : PROCESS
   BEGIN
      WAIT for 100ns;
      in_D1 <= '0';
      WAIT for 100ns;
      in_D1 <= '1';
      WAIT for 100ns;
   END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```

Wenn man im Design-Menü die Testbench-Datei auswählt, erscheint unten im Process-Bereich der Link zum ISim Simulator.

Wichtig: Mit einem Doppelklick führt man nacheinander zuerst den Syntaxcheck und dann die Simulation selbst aus. Ansonsten kann es vorkommen, dass die Mapping-Daten verloren gehen und erneut erstellt werden müssen.

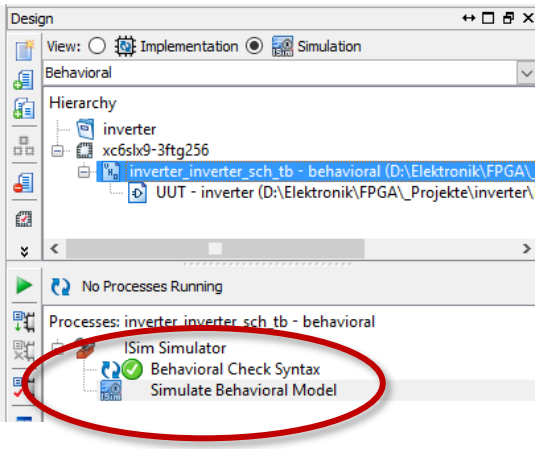


Abbildung 7: Testbench-Simulation

Die Ergebnisse der Simulation werden im ISim-Fenster angezeigt. Zu den in der Testdatei definierten Eingangswerten (hier ‚in_d1') sieht man die Werte des Ausgangssignals (hier ‚out_b1'). Die Prüfung, ob das Schema korrekt funktioniert müssen vom Beobachter selbst gemacht werden.

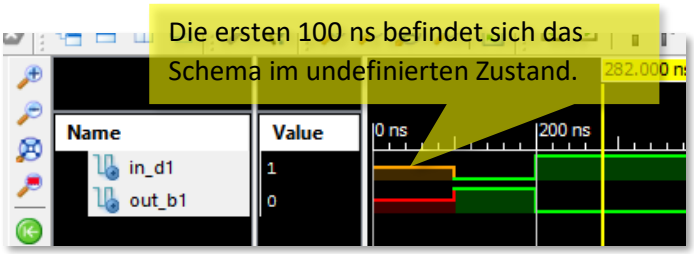


Abbildung 8: Ausgabe-Simulation

Schritt 5: Das Schema kompilieren

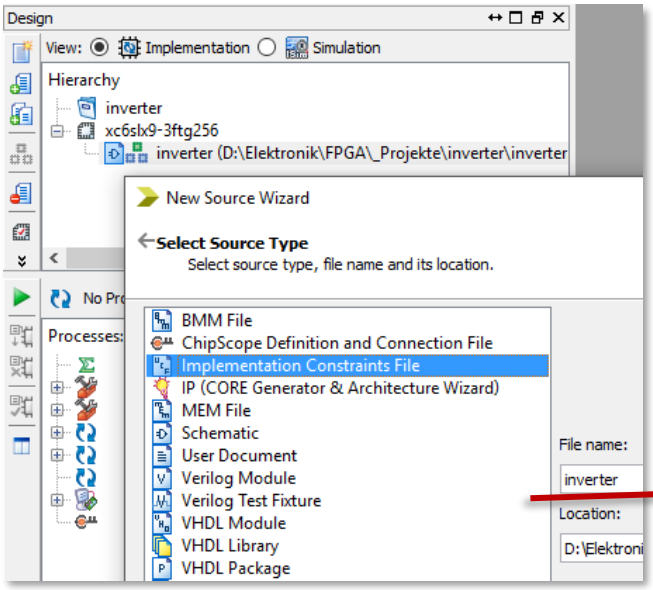


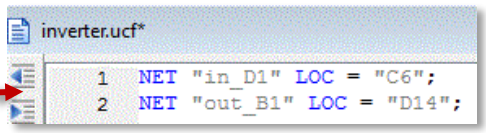
Abbildung 9: Erstellung der Mapping-Datei

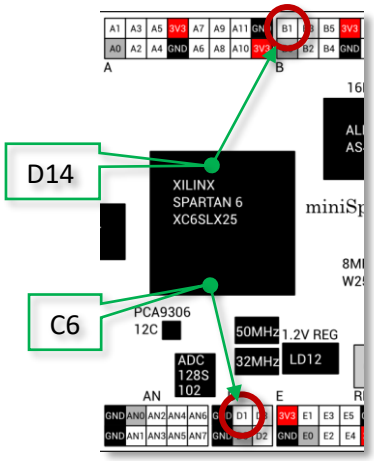
Nach der erfolgreichen Simulation kann das im Schritt 3 entworfene Schema auf den FPGA Chip übertragen werden. Zuvor muss das Schema kompiliert werden.

Dafür benötigt man die Mapping-Datei (Constraints File).

Diese wird als leere Datei angelegt und muss manuell mit den Zuordnungen von Ports am Board zu den Pins vom FPGA gefüllt werden.

Für unser Beispiel besteht die Datei nur aus zwei Zeilen.





Das Kompilieren kann vom letzten Schritt („Generate Programming File“) aus gestartet werden. Alle vorherigen Schritte (Synthese und Implementierung) werden automatisch in der korrekten Reihenfolge ausgeführt.

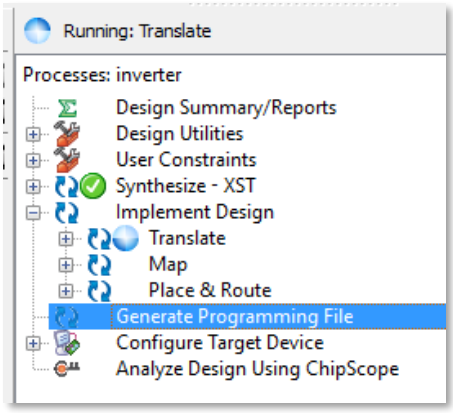


Abbildung 10: Schema kompilieren

Schritt 6: Schema auf den Chip anwenden

Um die kompilierte “.bit“-Datei auf den Chip zu übertragen, muss noch die Kommunikation zwischen PC und Board hergestellt werden. Auf dem miniSpartan6+ Board ist dafür ein FTDI Chip (FT2232HL) vorgesehen.

Für die Nutzung des FTDI-Chips muss zuerst der entsprechende Treiber herunterladen und installiert werden: http://www.ftdichip.com/Drivers/D2XX.htm

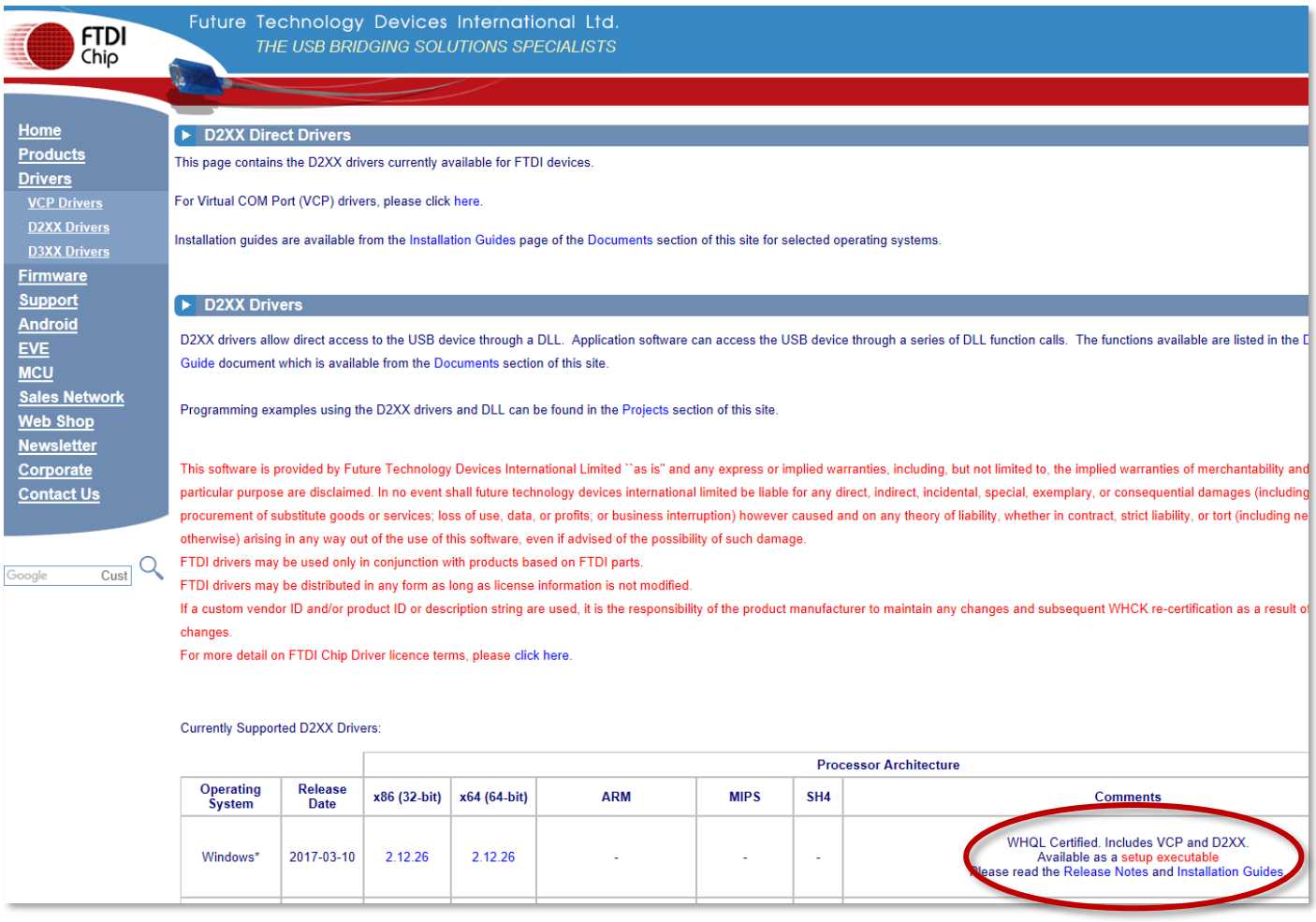


Abbildung 11: FTDI-Treiberdownload

Um die „.bit“-Datei via FTDI auf den FPGA hochzuladen, benötigt man ein zusätzliches Tool. Dies ist „portable“ (muss also nicht installiert werden) und kann hier heruntergeladen werden:

https://sourceforge.net/projects/xc3sprog/

Zuerst wird mit dem Tool die Erkennung der Hardware ausgeführt. (Ist einmalig pro Board und PC auszuführen.)

- „cmd“ als Administrator starten
- *cd* <Verzeichnis von xc3sprog>
- ***xc3sprog.exe -c ftdi bscan_spi_s6lx9_ftg256.bit***

Hinweis 1:
diese „.bit“-Datei ist chip- und boardabhängig und kann z.B. hier heruntergeladen werden:
http://hamsterworks.co.nz/mediawiki/index.php?title=MiniSpartan6%2B_SPI_Programming&oldid=4825

Hinweis 2:
Die Ausführung der „.bit“-Datei (siehe Screenshot) sieht recht fehlerhaft aus. Entscheidend ist aber, dass hinter der Zeile „DNA is …“ keine Fehlermeldungen mehr kommen.

Eine der möglichen Fehlerursachen, z.B. bei der Meldung mit „… JTAG Chain…“, kann der falsche USB-Port sein.

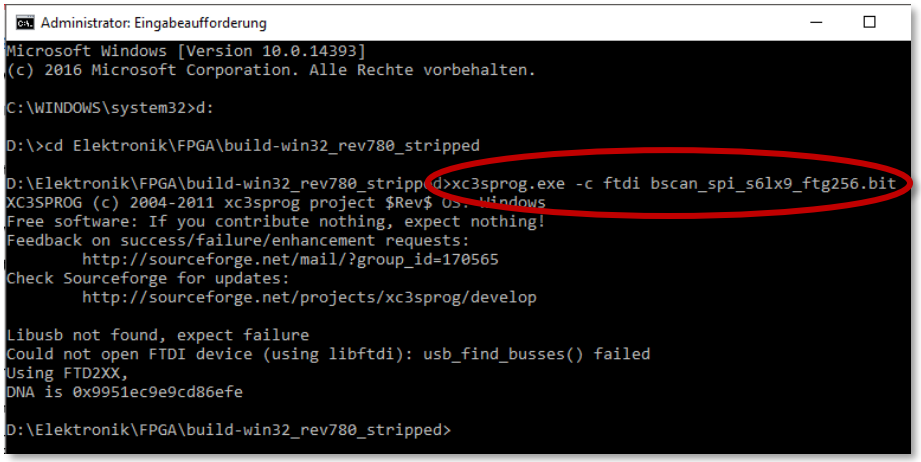


Abbildung 12: Ausführung XC3Prog

Zuletzt überträgt man das entworfene Design auf den FPGA mit der folgenden Befehlszeile:

```
D:\Elektronik\FPGA\build-win32_rev780_stripped>xc3sprog.exe -c ftdi d:\Elektronik\FPGA\_Projekte\inverter\inverter.bit
```

Hier wird das vorher in ISE erstellte und kompiliertes Programm über die FTDI-Schnittstelle auf den FPGA geschrieben und startet auf dem FPGA sofort.

Achtung:
Diese Befehlszeile muss nach jeder Korrektur des Schemas oder jedem Neustart des Boards ausgeführt werden.

Schritt 7: Aufbau der externen Beschaltung

Um den FPGA in Arbeit zu sehen, benötigen wir reale Eingangssignale und die Visualisierung der Ausgangssignale. Dafür wurde die hier dargestellte Schaltung aufgebaut.

Das Eingangssignal erzeugt ein simpler, nicht entprellter Taster:

im geöffneten Zustand liegt der Eingangsport (beispielsweise D1) auf niedrigem Potential (auf ‚0') und im geschlossen – auf hohem Potential (auf ‚1').

Als Aktor am Ausgangsport dient eine LowCurrent-LED mit Vorwiderstand.

Im Initialzustand der Schaltung leuchtet die LED. Bei gedrücktem Taster geht die LED aus:

Das Signal wird logisch invertiert.

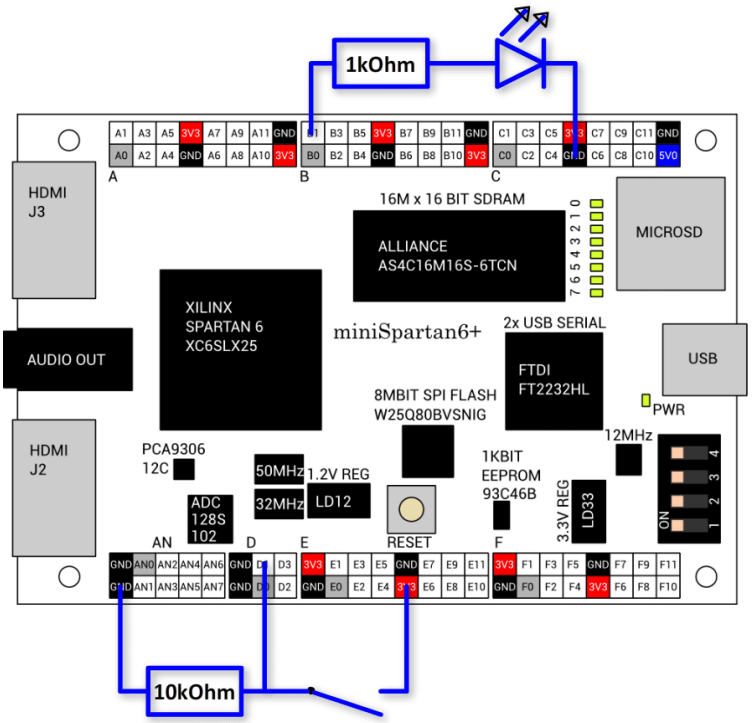


Abbildung 13: externe Beschaltung des Boards

Auf dem Foto sieht man beide Zustände der Schaltung.

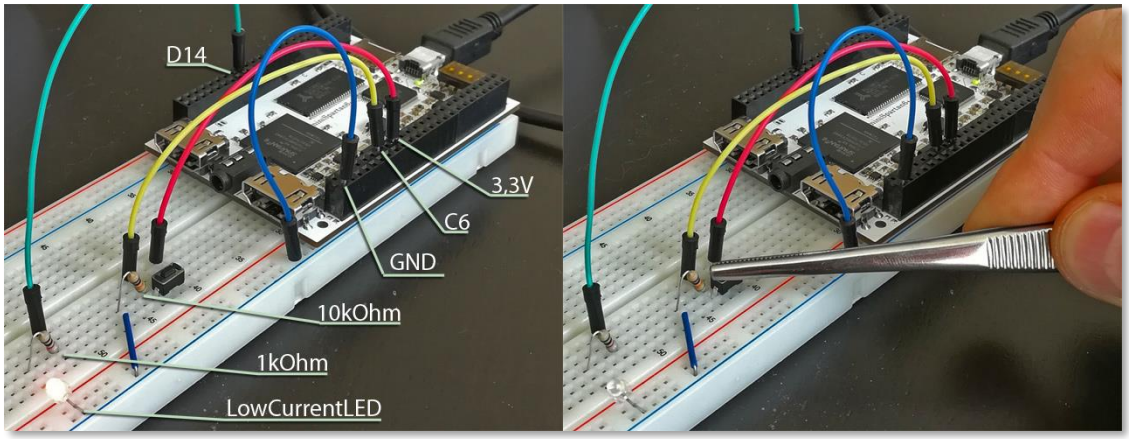


Abbildung 14: Fotodokumentation Board

Anhang: Über PORTs und PINs

Im Umgang mit FPGA begegnet man immer wieder den Pins und Ports.

Merksatz: Ports sind am Board. Pins am Chip.

Zuerst ist es wichtig zu wissen, wie man die Ports anspricht, an denen etwas angeschlossen wird.
Diese Information entnimmt man dieser Abbildung, z.B.:

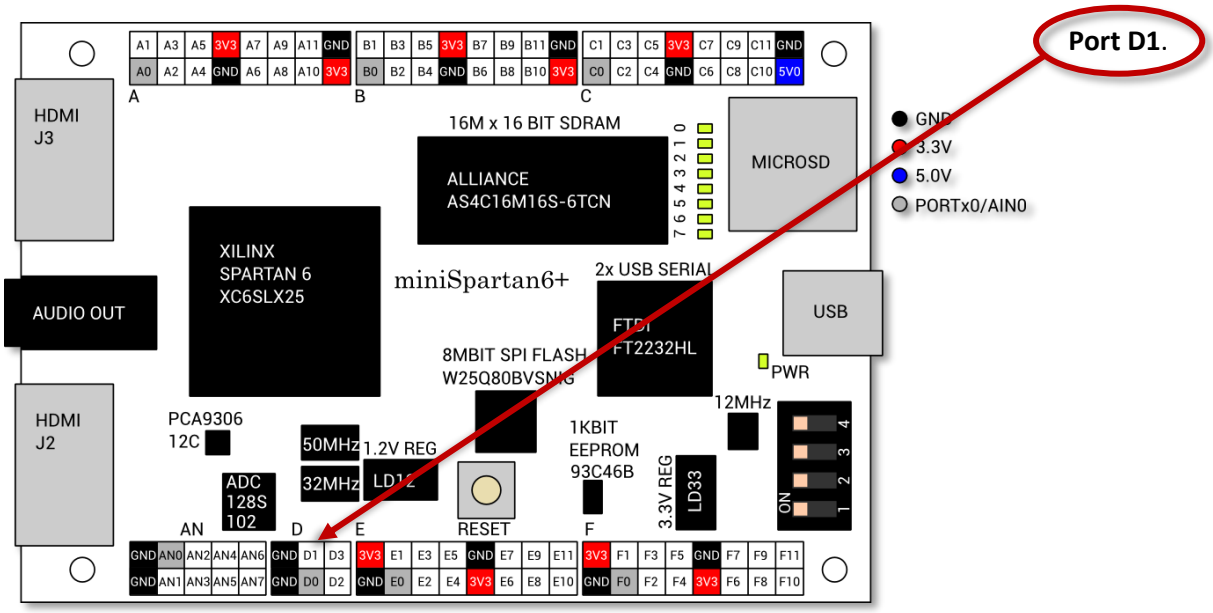


Abbildung 15: Dokumentation der Ports

Diese Ports sind intern den **Pins** des FPGA-Chips zugeordnet. Die Informationen über diese Zuordnung kann ebenfalls der Dokumentation entnommen werden (auf Seite 2 des PDFs):

https://github.com/scarabhardware/miniSpartan6-plus/blob/master/miniSpartan6%2B_Rev_B.pdf

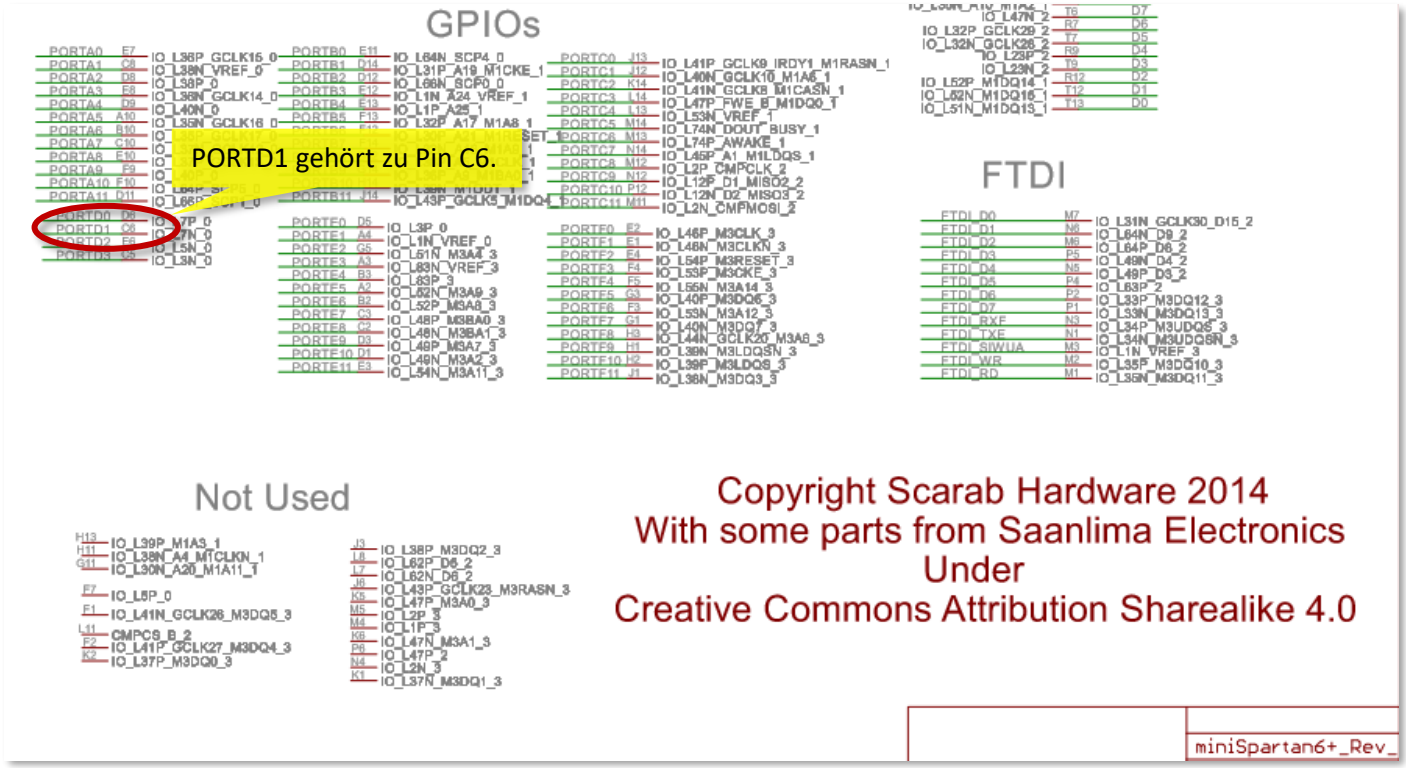


Abbildung 16: Dokumentation Port-Pin-Mapping