Einführung in Xilinx Vivado

EuR-III VHDL

Inhaltsverzeichnis

1. Einleitung

1.1. Zweck des Dokuments

Diese Einführung soll Studenten und anderen interessierten Personen helfen, möglichst schnell und effizient die Vivado HLx Umgebung von Xilinx für FPGA Entwicklung zu nutzen.

1.2. Gültigkeit des Dokuments

Die Ausführungen gelten sowohl für die kostenlose Webedition Ausführung, wie auch für die Lizenzierte Vollversion. Dort wo Unterschiede bestehen, wird darauf explizit hingewiesen.

1.3. Begriffsbestimmungen und Abkürzungen

FPGA	Field Programmable Gate Array, ein programmierbarer Baustein.
Xilinx	Xilinx Inc. ist der weltgrösste Entwickler und Hersteller von FPGA
Vivado	Die aktuelle Software von Xilinx für CPLD und FPGA Entwicklungen
VHDL	„Very High-Speed Hardware Description Language“

1.4. Zusammenhang mit anderen Dokumenten

Dieses Dokument ist dient als Einführung für die Kurse

- InI-1 Rechnerarchitektur,
- EuR-3 VHDL Selbststudium, und
- EuR-4 Digitale Systeme

Folgende begleitende Dokumente sind geplant oder bereits in Arbeit:

«VHDL Skript»

- «Einführung in Xilinx Vivado»
- «Smart Testbench Design»
- «VHDL Design Guidelines»

Weitere unterstützende Literatur:

- «The Zynq Book»
- «ZYBO Reference Manual»

2. Installation und Lizenzierung

Die Installation und Lizenzierung werden in „Vivado Installation und Lizenzierung.pdf“ getrennt behandelt.

3. Projekt Start mit Vivado

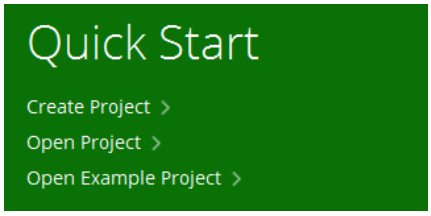
3.1. Starten der Vivado Software

Es gibt verschiedene Möglichkeiten, Vivado zu starten:

- Windows ***Start → Programs → Elektronik → Vivado 2019.1.3***
- Suche nach "Vivado" in der Startleiste von Windows
- Doppelklick auf ein bestehendes .XPR File (Xilinx Project File) eines Projektes

.

3.2. Erzeugung eines neuen Projektes



Nach dem Start von Vivado (ausser durch ein .XPR File) erscheint der Startbildschirm mit der Auswahl «Create Project».

Man kann dann einfach auf diese Auswahl klicken, oder aber
Aus dem Menu heraus ein neues Projekt starten mit «File» - «New Project…».

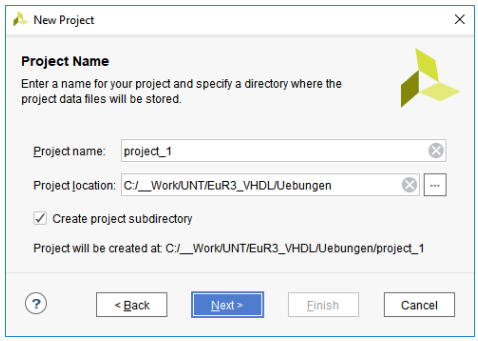
Im neuen Fenster erscheint dann zuerst einfach die Begrüssung durch den «Wizard» …
… hier kann man einfach «Next» klicken.

Schritt 1: Verzeichnis, Name und Top-Level Entity

Projektname: Wählen Sie ausdrucksstarke Namen, die klar sagen, was der Inhalt ist. Nicht einfach nur «Project_1» …

Location: Es beschleunigt die Synthese enorm, wenn das Projekt lokal auf dem D: Laufwerk ist, und nicht immer Netz-Werk Zugriff benötigt.

Create Subdirectory: Unbedingt anwählen!



Achtung: Verzeichnis-Pfad und Projekt-Namen dürfen keine Leerzeichen enthalten!

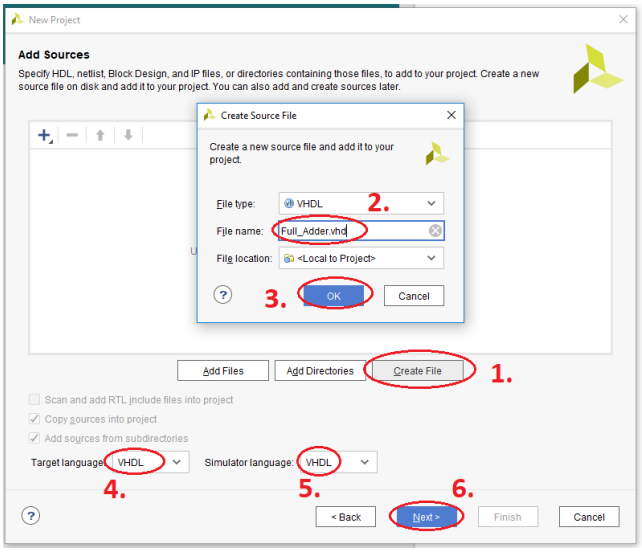
Dann «Next» klicken …

Schritt 2: Projekt Typ

Im Rahmen der NTB Aktivitäten werden wir immer «RTL Project» verwenden.
Dann klicken Sie «Next» …

Schritt 3: Add Sources

Beim nächsten Fenster «Add Sources» können Sie Design-Dateien definieren. Sie können auch später jederzeit zusätzliche Dateien zum Projekt einführen. Aber wenn Sie schon wissen, welche Dateien sie benötigen werden, ist es hier am einfachsten.



Klicken Sie auf «Create File», und geben Sie den neuen Namen mit Endung ein. Sie können hier beliebig viele Dateien hinzufügen.

Kontrollieren Sie, dass am unteren Rand die «Target Language» und auch «Simulator Language» beide auf «VHDL» stehen. Dann klicken Sie «Next».

Schritt 4: Add Constraints

Hier geht es um die «Constraints» beziehungsweise Randbedingungen für das Projekt. Dies ist dann speziell wichtig, wenn das Design über Pins des FPGA mit der Aussenwelt verbunden sein soll.

Zurzeit benötigen wir das jedoch nicht, und können diesen Schritt überspringen.

Schritt 5: Auswahl des Bauteils

Auch wenn wir unser Design noch nicht auf ein FPGA bringen wollen, müssen wir hier trotzdem bereits den richtigen Chip auswählen.

Am Einfachsten wählen Sie bei «Family» die «Zynq-7000» Serie an, und bei Package das Format «clg400» (0.8mm Pitch, RoHS (= bleifrei) kompatibles Gehäuse mit 400 Pins).

Mit dieser Einstellung erscheint dann in der «Part» Auswahl unser Chip «Xc7z010iclkg400-1L» zuoberst in der Liste und kann dann leicht angewählt werden.

«xc» steht dabei für Xilinx Commercial,

«7z» ist die Virtex 7000 «Zynq» Generation

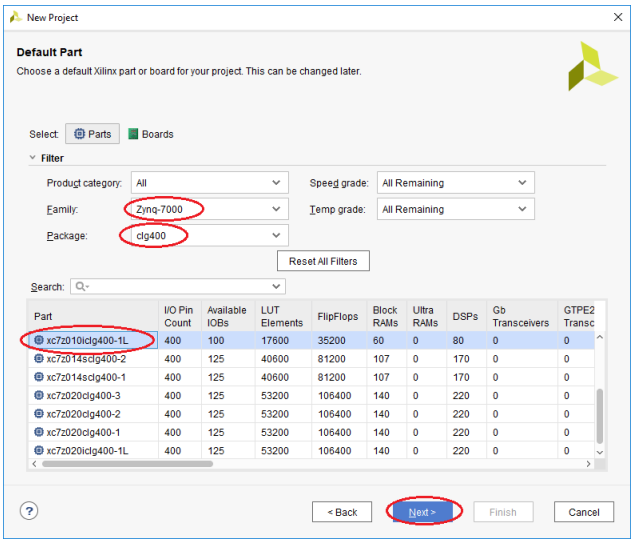
«010» ist eine Angabe für die Grösse (Zahl Logik-Elemente im FPGA)

«i» steht für die Temperaturfestigkeit (Industrial, -40° bis +100°C)

«clg» steht für das Gehäuse, mit 0.8mm Pitch und intern «wire-bond» und somit noch nicht «flip-chip».

«400» ist die Zahl der Pins, also ein Array von 20 x 20 Pins

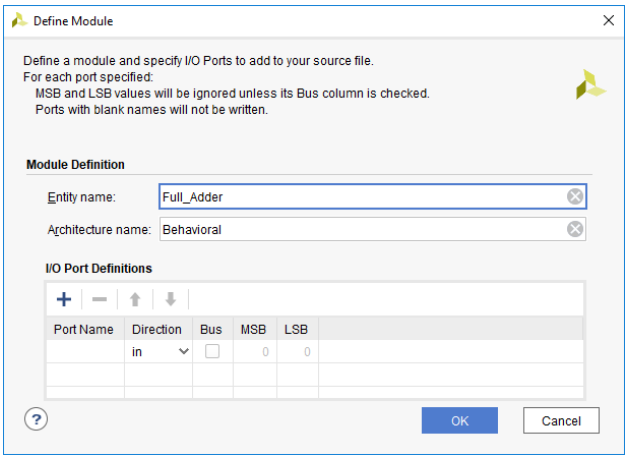
«-1L» ist der «Speed Grade». «1L» ist langsam, aber Low-Power fähig.



Diese Detail Informationen sind zurzeit nicht wichtig … sie können nach der Auswahl des richtigen Bauteils einfach «Next» drücken.

Schritt 6: New Project Summary

Hier werden die gewählten Einstellungen nochmals zusammengefasst.
Klicken Sie einfach «Finish».



Nach dem Finish kommt ein Fenster zur Definition der Module. Hier könnte man interaktiv die Ports für das neue Design-File definieren …

… was jedoch unpraktisch ist.
Überspringen Sie diesen Schritt einfach mit «OK» und dann «Yes».

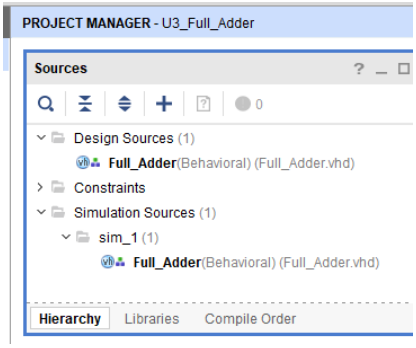
Schritt 7: Editieren der Files

Jetzt kann es losgehen.

Im «PROJECT MANAGER» Fenster sieht man die Files und Einheiten (ENTITY) des Designs.

Design-Elemente erscheinen gleichzeitig immer bei «Design Sources» und auch bei «Simulation Sources».

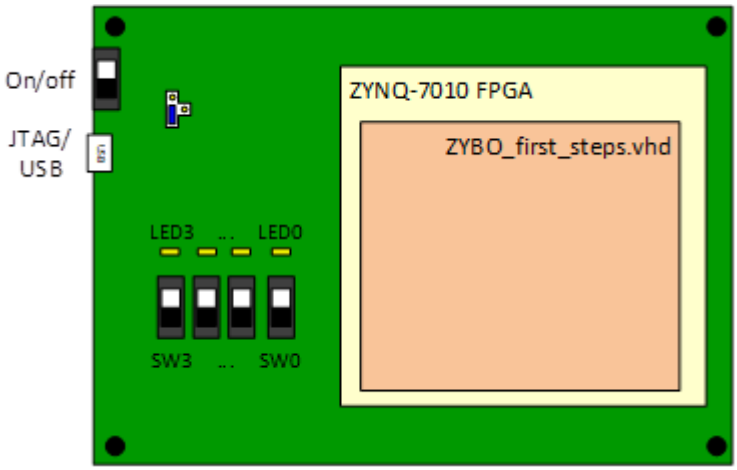
Durch Doppelklick öffnet man die Files zum Bearbeiten.



4. Erstes kleines Vivado Projekt in VHDL mit ZYBO

In diesem Kapitel geht es nur darum, eine möglichst einfache Schaltung auf den Zynq-FPGA des ZYBO Board von Digilent zu bringen – um den ganzen Design-Fluss einmal durchzuspielen.

Die VHDL Schaltung ist sehr simple und steuert mit den 4 Schaltern synchron zum 125 MHz Takt die 4 LEDs an.



ACHTUNG: Wenn Sie Vivado selbst installieren, ist es **ganz wichtig**, dass Sie die **Version 2019.1.3** verwenden. Vivado ist sehr heikel, wenn es um die Kompatibilität von Design-Komponenten geht, und nur schon ein Wechsel z.B. von 2017.3 auf 2017.4 verhindert, dass diese problemlos integriert werden können.

Bei dieser ersten Übung spielt das noch keine Rolle, aber bei späteren Übungen kann dies den Erfolg verhindern!

4.1. Neues Projekt «ZYBO_First_Steps»

Erzeugen Sie, wie bereits im vorhergehenden Kapitel erklärt, ein neues Projekt, diesmal jedoch mit dem Namen «ZYBO_First_Steps».

ACHTUNG: Der Datenpfad und Projektname dürfen keine Leerzeichen und auch keine Umlaute enthalten, sonst kann es später bei der Kompilierung und Software-Erstellung Probleme geben!

Gut zu wissen: Für die Übungen in «Digitale Systeme» können Sie die Projektdaten auf dem NTB M:\ Laufwerk (oder Desktop – ist dasselbe) erstellen …
… was aber die Synthese infolge des Netzwerk-Verkehrs doch massiv langsamer machen kann.

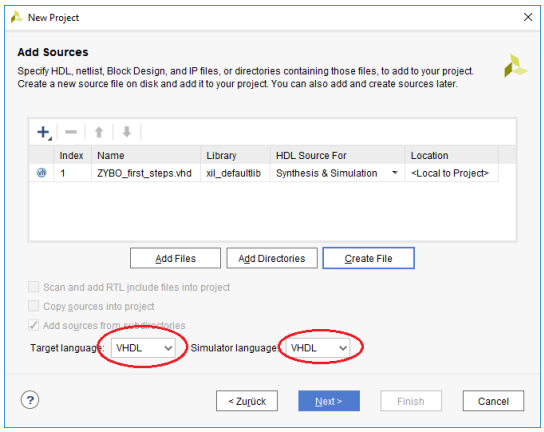
Schneller ist es, auf dem lokalen Laufwerk (an der NTB ist es D:\ ) zu arbeiten, und erst am Ende das ganze Projekt auf M:\ zu kopieren.

Im Fenster «Project Type» wählen Sie dann als «Project Type» das Feld «RTL Project».

Beim nächsten Fenster «Add Sources» erzeugen Sie mit «Create File» ein neues File mit dem Namen «ZYBO_First_Steps.vhd».

Achten Sie darauf, dass das neue File vom Typ «VHDL» ist … und nicht «Verilog»!

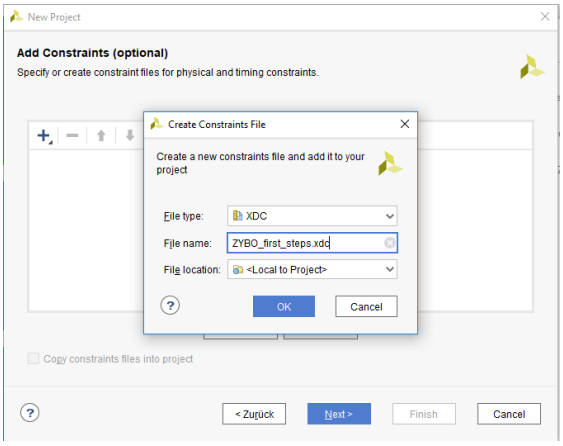
Achten Sie auch auf die beiden Felder am unteren Bildrand für die «Target language» und auch die «Simulator language» … beide müssen ebenfalls auf «VHDL» stehen.



Nach einem «Next» erzeugen Sie im nächsten Fenster mit Namen «Add Constraints» mit «Create File» ein neues File Xilinx Design-Constraints (xdc) File mit dem Namen «ZYBO_First_Steps.xdc».

Dieses wird die Pin-Belegung und weitere Elemente zur Steuerung der Synthese enthalten.

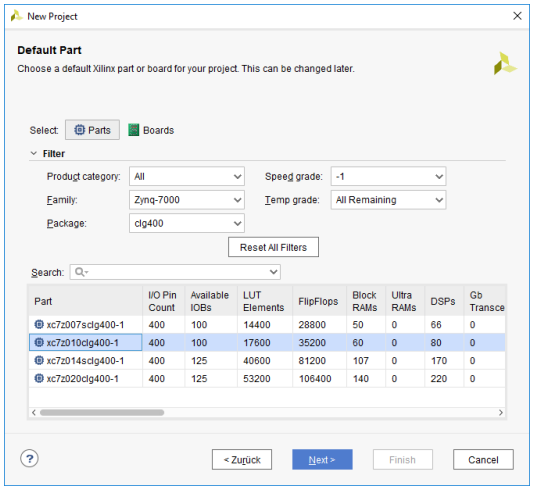
Anschliessend «OK» und «Next» klicken.



Es folgt die Eingabe des Boards oder des verwendeten FPGA Bauteils im Fenster «Default Part».

Auf dem ZYBO Board wird das Xilinx FPGA mit der Bezeichnung «xc7z010clg400-1» verwendet.
Es hilft, wenn man bei «Family» den Zynq-7000, und bei «Package» das CLG400 eingibt. Man kann
auch noch bei «Speed grade» ein -1 setzen, um die Auswahl weiter einzuschränken.

Am Ende wählen Sie «xc7z010clg400-1», oder. «xc7z010clg400-1L» (Low-Power Version).

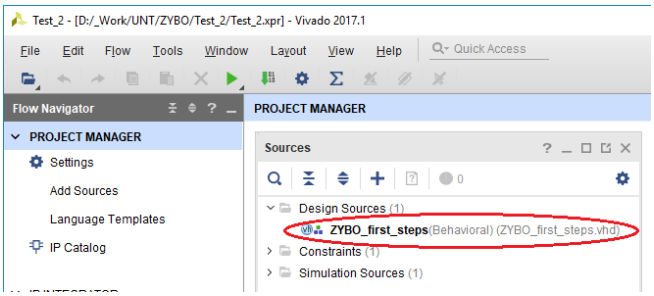


Dann einfach noch «Next» und «Finish».

Die Eingaben bei «Define Module» für irgendwelche «I/O Port Definitions» können Sie mit «OK» und «YES» ruhig überspringen.

4.2. VHDL Design File Schreiben

Öffnen Sie im «Project Manager» unter «Design Sources (1)» das VHDL File «ZYBO_First_Steps.vhd» mit Doppelklick.



Ersetzen Sie darin alle von Vivado vorgegebenen Zeilen mit dem Inhalt wie unten gezeigt. Sie können die paar Zeilen abtippen, oder mit Copy-Paste in Vivado übernehmen.

Wie Sie leicht erkennen können, ist der Inhalt trivial: Bei jeder positiven Flanke von «isl_clock»
werden die 4 Schalter Eingänge «islv4_switch» auf die 4 LED Ausgänge «oslv4_led» kopiert.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY zybo_first_steps IS
    PORT (
        isl_clock     : IN  STD_LOGIC;
        islv4_switch  : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
        oslv4_led     : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
      );
END ENTITY zybo_first_steps;


ARCHITECTURE rtl OF zybo_first_steps IS

BEGIN

    reg_proc : PROCESS (isl_clock)
    BEGIN
        IF rising_edge(isl_clock) THEN
            oslv4_led   <= islv4_switch;
        END IF;
    END PROCESS reg_proc;

END ARCHITECTURE rtl;
```

Speichern Sie Ihre Änderungen in Vivado mit «Ctrl-S», dem Floppy- Zeichen  oder mit dem Menu «File» - «Save File».

4.3. Pin-Definition im .XDC File

Damit Vivado die Ein- und Ausgänge richtig verbinden kann, müssen diese irgendwo definiert werden. Dies geschieht im XDC-File (Xilinx Design-Constraints) – das sind die «Constraints» oder «Randbedingungen».

Öffnen Sie im «Project Manager» unter «Constraints (1)» das XDC File «ZYBO_first_steps.xdc» mit Doppelklick.

Ersetzen Sie allfällig vorgegebenen Text mit den folgenden Zeilen:

```
##  Clock signal
##===============
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 }
                   [get_ports { isl_clock }];
create_clock -add -name sys_clk_125m -period 8.00 -waveform {0 4}
                   [get_ports { isl_clock }];
set_input_jitter [get_clocks -of_objects [get_ports isl_clock]] 0.1

##  Switches
##===========
set_property -dict { PACKAGE_PIN G15   IOSTANDARD LVCMOS33 }
                 [get_ports { islv4_switch[0] }];
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 }
                 [get_ports { islv4_switch[1] }];
set_property -dict { PACKAGE_PIN W13   IOSTANDARD LVCMOS33 }
                 [get_ports { islv4_switch[2] }];
set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCMOS33 }
                 [get_ports { islv4_switch[3] }];

##  LEDs
##=======
set_property -dict { PACKAGE_PIN M14   IOSTANDARD LVCMOS33 }
                 [get_ports { oslv4_led[0] }];
set_property -dict { PACKAGE_PIN M15   IOSTANDARD LVCMOS33 }
                 [get_ports { oslv4_led[1] }];
set_property -dict { PACKAGE_PIN G14   IOSTANDARD LVCMOS33 }
                 [get_ports { oslv4_led[2] }];
set_property -dict { PACKAGE_PIN D18   IOSTANDARD LVCMOS33 }
                 [get_ports { oslv4_led[3] }];
```

Sie erkennen jeweils leicht den Zusammenhang zwischen dem «PACKAGE_PIN» dem Standard für die I/O Spannung (Low-Voltage CMOS 3.3V) und den Signalnamen.

Speichern Sie Ihre Änderungen in Vivado mit «Ctrl-S», dem Floppy- Zeichen  oder mit dem Menu «File» - «Save File».

4.4. Kompilierung

Die Umwandlung unserer Design-Eingaben in das FPGA Konfigurations-File verläuft über mehrere Stufen.

Diese können nacheinander einzeln gestartet werden, oder als Ganzes im Ablauf durch Doppelklick des dritten Knopfes «Generate Bitstream».

Teil 1: Run Synthesis
Hier wird das Design analysiert und in ein internes Meta-Format umgewandelt, welches die gewünschten Strukturen und Zusammenhänge effizient abspeichert.

Wenn es im Design Fehler gibt, dann werden Sie hier offensichtlich.

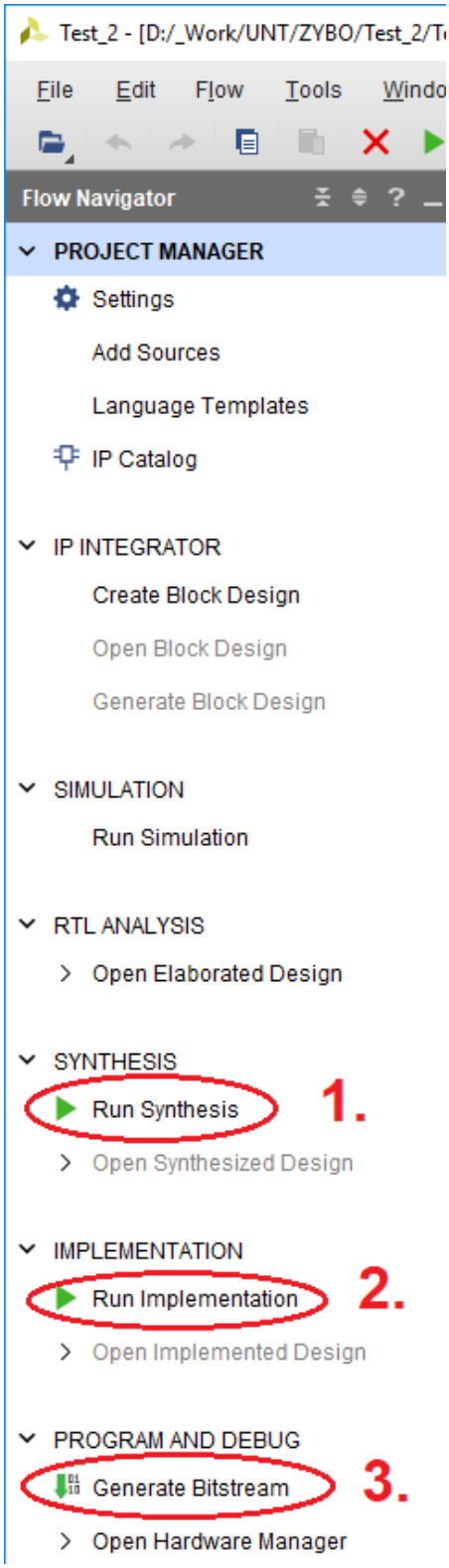
Dieser Schritt (wie auch die folgenden) dauert seine Zeit. Dabei wird rechts oben mit einem grünen rotierenden Ring angezeigt, dass der Prozess im Hintergrund läuft …

Teil 2: Run Implementation
Dabei wird das analysierte Design auf die im FPGA vorhandenen Strukturen und Gatter abgebildet. Dies beinhaltet die logischen Gatter, aber auch Multiplizierer und Speicher.

Wenn es im gewählten FPGA zu wenig Ressourcen gibt, dann wird dies hier erkannt.

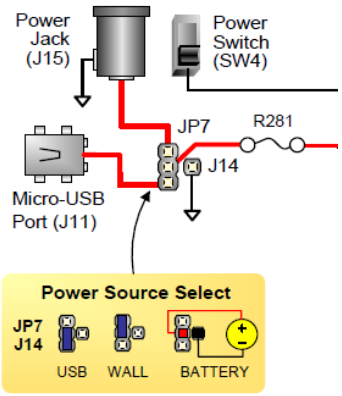
Teil 3: Generate Bitstream
Jetzt muss noch das effektive Konfigurations-File erzeugt werden. In diesem Schritt werden die Schnittstellen nach aussen «angeschlossen», und das «Bitstream» File geschrieben.

Nach jedem Teilschritt kommt ein Pop-Up Menu, wie es weitergehen soll: Ob das Design zur visuellen Kontrolle geöffnet werden soll, oder ob der jeweils nächste Schritt gestartet werden soll.



ACHTUNG: Die Kompilierung ist recht zeitintensiv. Dabei kann es für mehrere Minuten so aussehen, als passiere gar nichts. Achten Sie auf die Ecke oben rechts, ob dort ein grüner Ring rotiert (busy), oder aber ein grüner «Ok»-Haken und das Wort «Complete» steht.

4.5. Transfer des Files auf das FPGA Board



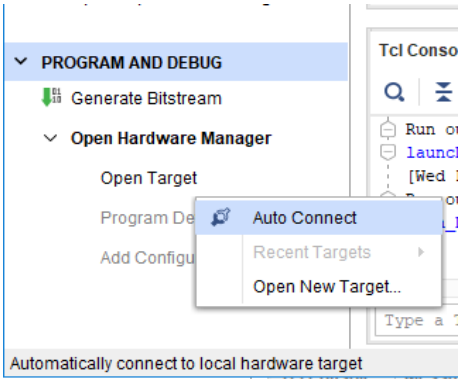
Bereiten Sie die FPGA Hardware vor:

- Kontrollieren Sie, dass der blaue 3-Positionen Jumper «JP7» auf «USB» steht.
- Schliessen Sie über Micro-USB Stecker «PROG UART» das Board an den PC an
- Schalten Sie das Board mit dem Schalter SW4 (beim USB-Stecker) ein

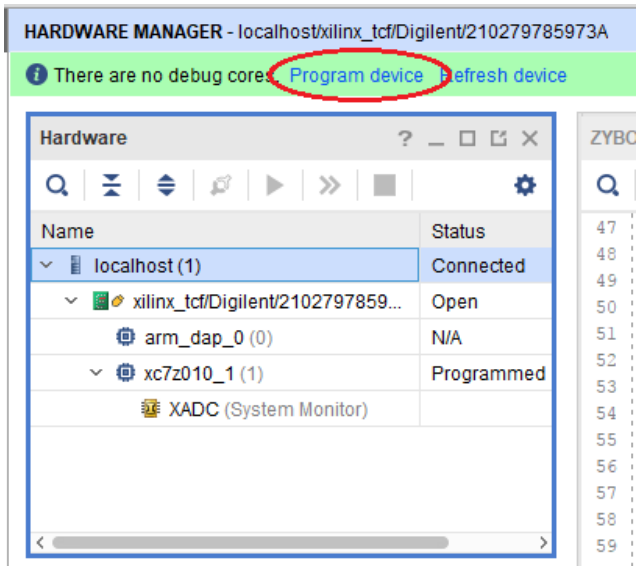
ACHTUNG: Verwenden Sie **<u>nicht</u>** die USB Anschlüsse an den NTB Bildschirmen, sondern nur die am PC selbst … damit es hier sicher kein Problem gibt mit der Verbindung.

Nach der ersten Erzeugung des «Bitstream» kommt ein Pop-Up Fenster mit "Open Hardware Manager".

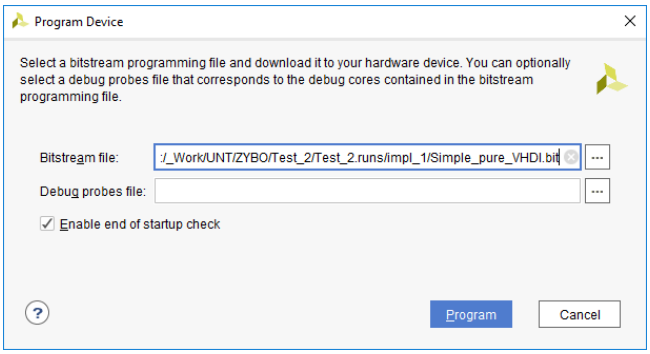
Wenn nicht, dann können Sie den Hardware Manager auch manuell öffnen, indem Sie im linken Menu bei «PROGRAM AND DEBUG» unter «Open Hardware Manager», dann «Open Target» und «Auto Connect» um die Verbindung zur Ziel-Hardware zu erstellen.



Es erscheint neu das Fenster «Hardware Manager» mit dem genauen FPGA Chip, und dem Link «Program device».

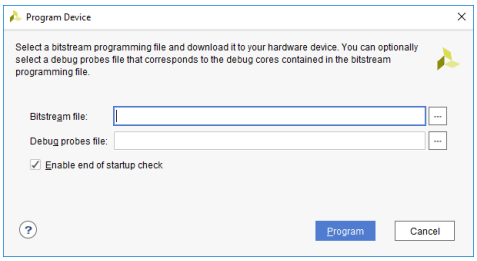


Klicken Sie auf «Program device», und es erscheint ein weiteres Fenster mit dem automatisch gesetzten «Bitstream» File und dem Knopf für «Program».



Achtung:

Es kann vorkommen, dass kein Bitstream File ausgewählt wird.



In diesem Fall muss man es mit dem Knopf rechts [...] selber suchen.
Sie finden das File in Ihrem Projektverzeichnis
« <Projektname>.runs / impl_1 / ZYBO_first_steps.bit »

Kontrollieren Sie, dass das Bitstream-File tatsächlich auf Ihr Projekt-Verzeichnis zeigt, und starten Sie die Programmierung.

Wenn alles erfolgreich war, dann sollten Sie mit jedem der 4 Schaltern je eine LED ein- und ausschalten können.

Herzliche Gratulation zum vielleicht ersten eigenen FPGA Projekt!