

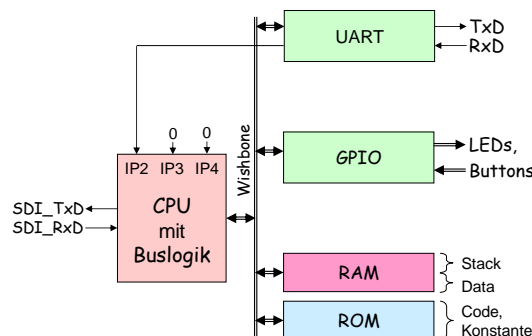
Praktikum Digitale Komponenten

Übung 3: Beispielrechner-System

Das in der Vorlesung vorgestellte Beispielrechner-System soll von Ihnen mit Ihrem Seriellen Sender vervollständigt werden. Dann sollen Hardware und Software übersetzt werden. Das vollständige System wird anschließend auf das FPGA-Board geladen und in seiner Funktion überprüft.

Am System sollen danach die Hard- und Softwareentwicklungspfade im Detail nachvollzogen werden.

Die Einrichtung der Entwicklungsumgebung wird in einem separaten Dokument beschrieben, dieses muss zunächst durchgearbeitet werden.



Aufgabe 1a: UART vervollständigen (ModelSim/QuartaSim)

Kopieren Sie die vorgegebenen Dateien des Versuchs in einen Arbeitsordner (Vorschlag: c:\praktikum). Die Ordnerstruktur muss dabei vollständig erhalten bleiben. Achten Sie darauf, dass im Pfad keine Leerzeichen und keine Sonderzeichen (zum Beispiel Umlaute) enthalten sind.

Ergänzen Sie die UART-Komponente durch den von Ihnen in der Übung 2 erstellten Seriellen Sender. Kopieren Sie dazu Ihre Datei „Serieller_Sender.vhd“ in den Ordner „Hardware\UART“. Öffnen Sie die Datei UART.vhd und instanziiieren Sie an der mit „TODO“ gekennzeichneten Stelle ihre Komponente. Verbinden Sie die Ports der Komponente mit den Signalen der UART-Komponente.

Starten Sie den VHDL-Simulator (ModelSim oder QuestaSim). Ändern Sie dessen Arbeitsordner zu „Hardware\UART“.

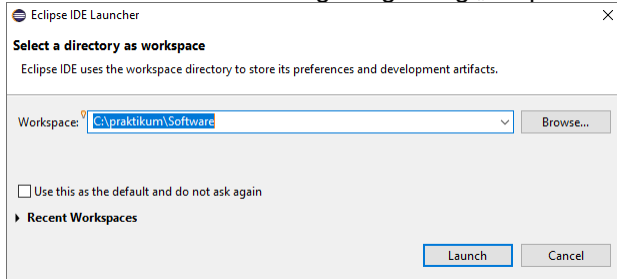
Legen Sie mit dem Kommando „vlib work“ eine Bibliothek an.

Verifizieren Sie die Einbindung im VHDL-Simulator mit der Testbench „UART_tb.vhd“ und dem Script „test_UART.do“.

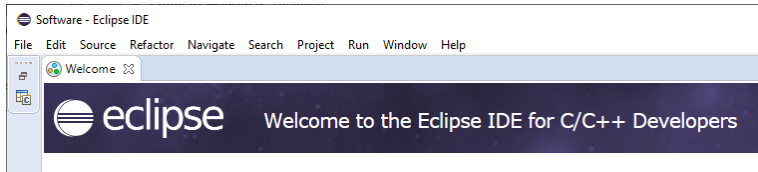
Bei erfolgreicher Ausführung gibt die Testbench die Meldung „Test fertig“ im Transcript-Fenster aus.

Aufgabe 1b: Software für den Beispielrechner übersetzen (Eclipse)

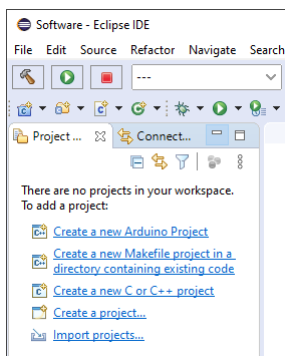
Starten Sie die Entwicklungsumgebung „Eclipse“.



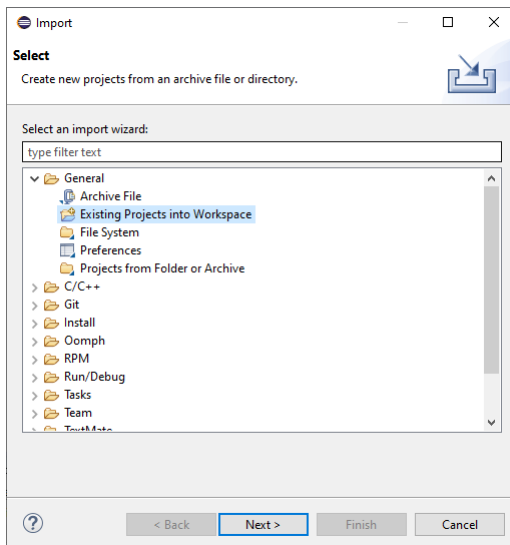
Wählen Sie den Ordner „Software“ als Eclipse-Workspace aus.



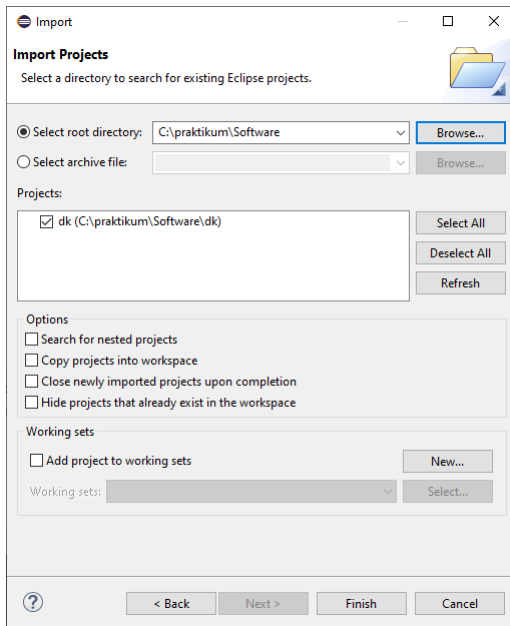
Schließen Sie den Tab „Welcome“ durch Klick auf die Schaltfläche ‚X‘.



Klicken Sie auf „Import Projects...“.



Wählen Sie „General“ → „Existing Projects into Workspace“ und klicken Sie auf „Next“.

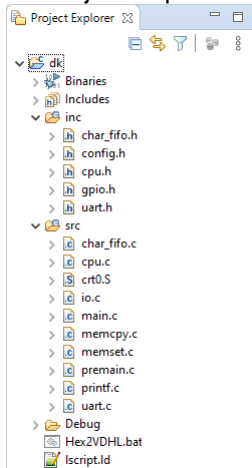


Klicken Sie erst auf „Browse“ und im Auswahlfenster auf „Ordner auswählen“. Das Projekt „dk“ sollte nun unter „Projects“ angezeigt und mit einem Haken ausgewählt sein. Klicken Sie auf „Finish“.

Das Software-Projekt wird automatisch übersetzt. Bei Erfolg sehen Sie in der Konsole:

11:02:20 Build Finished. 0 errors, 0 warnings. (took 2s.631ms)

Im Project Explorer können Sie die zum Projekt gehörenden Dateien:



Das Hauptprogramm finden Sie in der Datei „src\main.c“. Nehmen Sie sich ein wenig Zeit, um die Funktion des Programms zu verstehen. Die mit TODO gekennzeichneten Stellen füllen Sie später in Aufgabe 2.

Schauen Sie sich die Datei „Debug\dk_diss.txt“ an. Diese enthält das von C nach Maschinensprache übersetzte Programm. Sie können dieser Datei auch die Adressen von Unterprogrammen und von globalen Variablen entnehmen.

Erneuern Sie die VHDL-Speicherbeschreibungen (Doppelklick auf Hex2VHDL.bat). Dadurch werden die Speicherbeschreibungen im Ordner „Hardware“ mit dem übersetzten Programm gefüllt. Schauen Sie sich die dabei erstellte Datei „Hardware\Speicher\bsr2_rom.vhd“ im Editor an. Sie enthält nun das Programm in Maschinensprache.

Aufgabe 1c: Beispielrechner-System simulieren (ModelSim / QuestaSim)

Schauen Sie sich die Datei „Hardware\Beispielrechner_System_V3_testbench.vhd“ an. Versuchen Sie, deren Funktion zu verstehen.

Falls der VHDL-Simulator noch aktiv ist: Beenden Sie die Simulation mit „quit –sim“, falls nicht: Starten Sie den VHDL-Simulator.

Setzen Sie dessen Arbeitsordner auf „Hardware“.

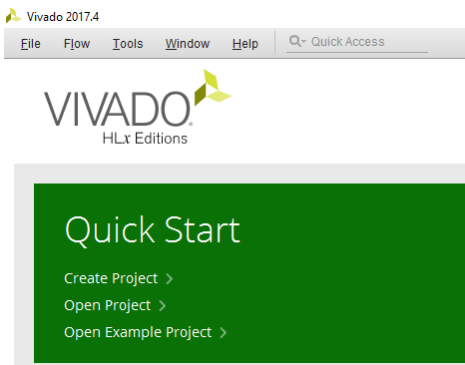
Legen Sie mit dem Kommando „vlib work“ eine Bibliothek im Arbeitsordner an.

Übersetzen und simulieren Sie das System mittels der mitgelieferten Kommandodatei „test_Beispielrechner_System_V3.do“.

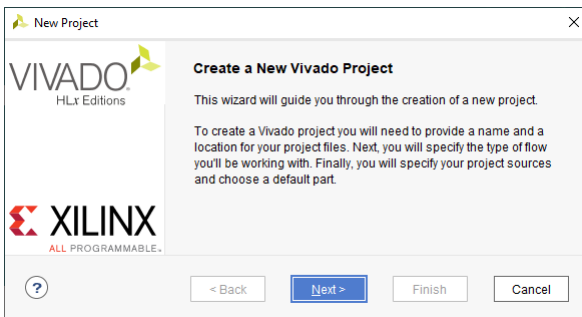
Der Test für Aufgabe 1 (,a‘ senden, ,A‘ empfangen) sollte erfolgreich sein, der Test für Aufgabe 2 (BTN setzen, LED ablesen) schlägt noch fehl („Falscher Wert auf LED“).

Aufgabe 1d: Hardware synthetisieren (Vivado)

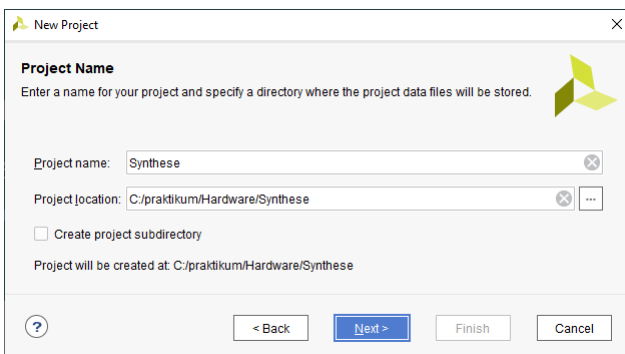
Starten Sie die Synthese-Software „Vivado“. Es erscheint das Fenster „Quick Start“:



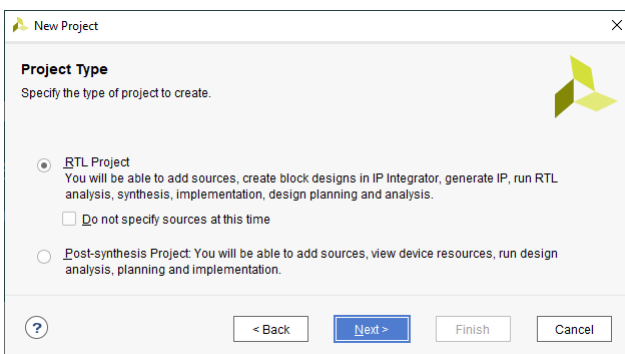
Klicken Sie zum Anlegen eines Projekts auf „Create Project >“.



Wählen Sie „Next >“



Tragen Sie als Projektnamen „Synthese“ ein und wählen Sie als Speicherort den Ordner „Hardware/Synthese“ aus im Arbeitsordner. Klicken Sie auf „Next >“.



Lassen Sie die Option „RTL Project“ aktiv und den Haken bei „Do not specify sources at this time“ inaktiv. Klicken Sie auf „Next >“.

New Project

Add Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

Index	Name	Library	HDL Source For	Location
16	Beispielrechner_System.vhd	xil_defaultlib	Synthesis & Simulation	C:\praktikum\Hardware
1	GPIO.vhd	xil_defaultlib	Synthesis & Simulation	C:\praktikum\Hardware\GPIO
2	DF_Serial_in_v1_0.vhd	xil_defaultlib	Synthesis & Simulation	C:\praktikum\Hardware\Prozessor
3	DF_Serial_out_v1_0.vhd	xil_defaultlib	Synthesis & Simulation	C:\praktikum\Hardware\Prozessor
4	DF_Wishbone_Interface.vhd	xil_defaultlib	Synthesis & Simulation	C:\praktikum\Hardware\Prozessor
5	Serial_Wishbone_Interface.vhd	xil_defaultlib	Synthesis & Simulation	C:\praktikum\Hardware\Prozessor
6	bsr2_processor.vhd	xil_defaultlib	Synthesis & Simulation	C:\praktikum\Hardware\Prozessor
7	bsr2_processor_core.vhd	xil_defaultlib	Synthesis & Simulation	C:\praktikum\Hardware\Prozessor
8	div.vhd	xil_defaultlib	Synthesis & Simulation	C:\praktikum\Hardware\Prozessor
9	mult.vhd	xil_defaultlib	Synthesis & Simulation	C:\praktikum\Hardware\Prozessor
10	wb_arbiter.vhd	xil_defaultlib	Synthesis & Simulation	C:\praktikum\Hardware\Prozessor
11	bsr2_ram.vhd	xil_defaultlib	Synthesis & Simulation	C:\praktikum\Hardware\Speicher
12	bsr2_rom.vhd	xil_defaultlib	Synthesis & Simulation	C:\praktikum\Hardware\Speicher
13	Serieller_Empfänger.vhd	xil_defaultlib	Synthesis & Simulation	C:\praktikum\Hardware\UART
14	Serieller_Sender.vhd	xil_defaultlib	Synthesis & Simulation	C:\praktikum\Hardware\UART
15	UART.vhd	xil_defaultlib	Synthesis & Simulation	C:\praktikum\Hardware\UART

☐ Scan and add RTL include files into project
☐ Copy sources into project
☒ Add sources from subdirectories

Target language: VHDL Simulator language: Mixed

Fügen Sie mit „Add Files“ alle für die Synthese benötigten VHDL-Dateien aus dem Ordner „Hardware“ (und seinen Unterordnern) hinzu. Klicken Sie anschließend auf „Next >“.

New Project

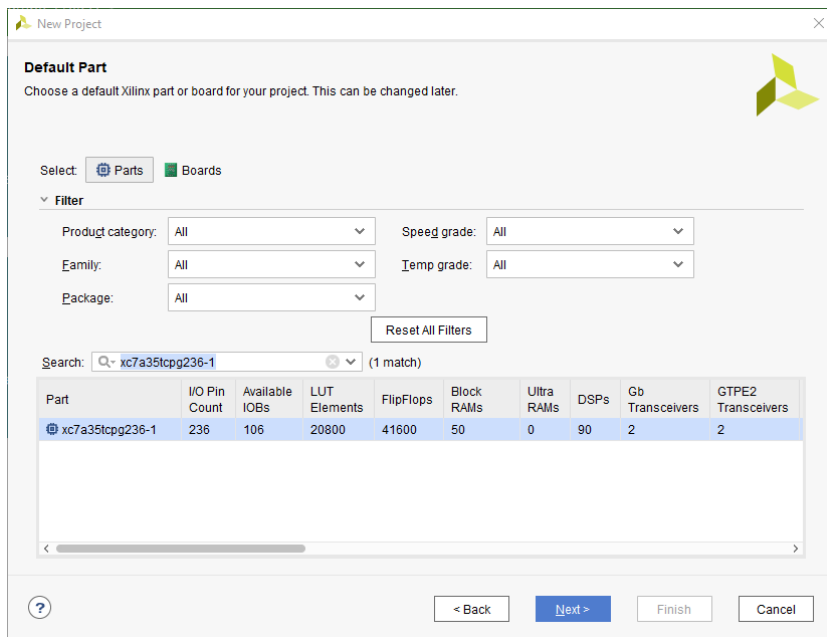
Add Constraints (optional)

Specify or create constraint files for physical and timing constraints.

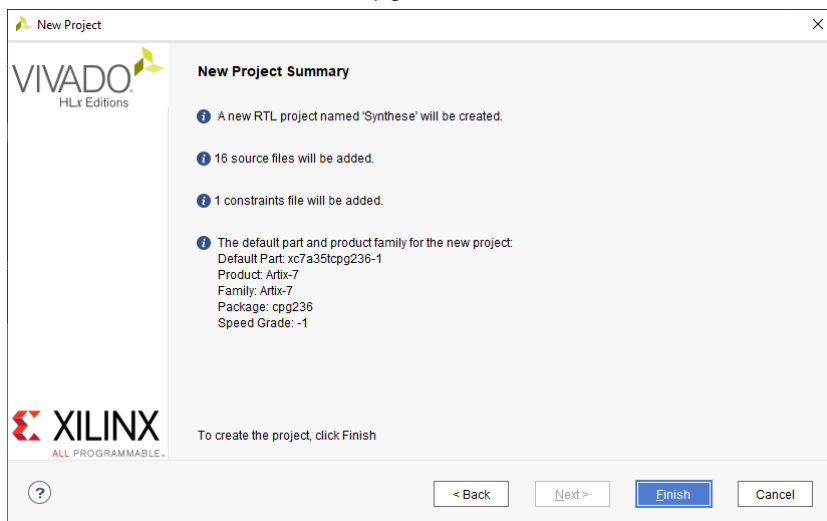
Constraint File	Location
Beispielrechner_System.xdc	C:\praktikum\Hardware

☐ Copy constraints files into project

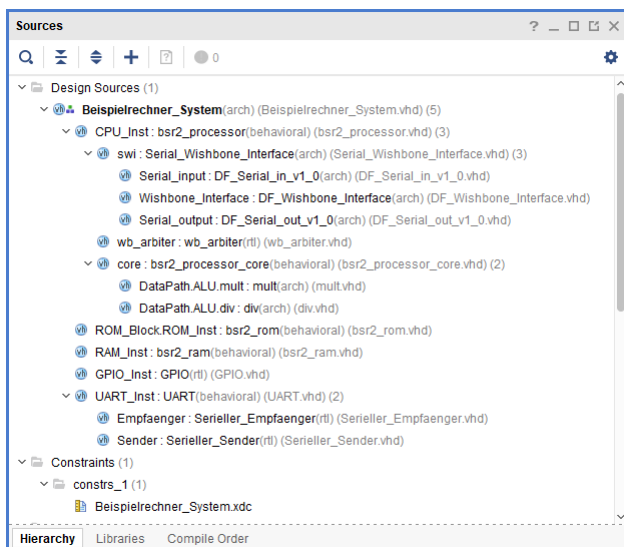
Fügen Sie mit „Add Files“ die Constraints-Datei „Hardware\Beispielrechner_System.xdc“ hinzu. Klicken Sie auf „Next >“.





Wählen Sie das Part „xc7a35tcpg236-1“ aus und klicken Sie auf „Next >“.

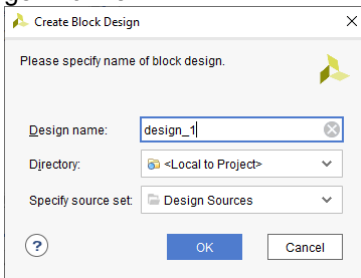


Sie bekommen noch einmal eine Übersicht des neuen Projekts angezeigt. Bestätigen Sie die Erzeugung des neuen Projekts mit „Finish“.

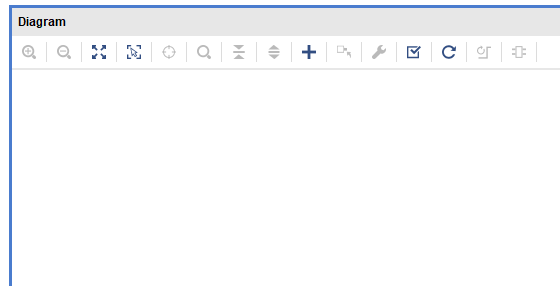


Werfen Sie nun einen Blick auf den Project Manager. Mit der Schaltfläche  (Expand All) können Sie die gesamte Hierarchie ausklappen. Sie sehen dort, ob alle benötigten Dateien im Projekt enthalten sind. Würde noch eine Datei fehlen, wäre Sie mit dem Symbol  markiert. Diese könnten Sie mit „Add Sources“ ergänzen.

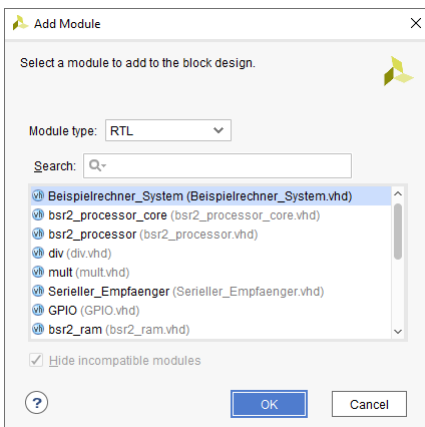
Unser System soll mit einem Systemtakt von 50 MHz betrieben werden. Auf dem Board ist jedoch ein Taktgeber mit 12 MHz vorhanden. Der gewünschte Systemtakt kann aber aus dem Eingangstakt erzeugt werden. Am bequemsten ist die Konfiguration in einem Block Design möglich, welches Sie mit „Create Block Design“ erzeugen können:



Klicken Sie auf „OK“.



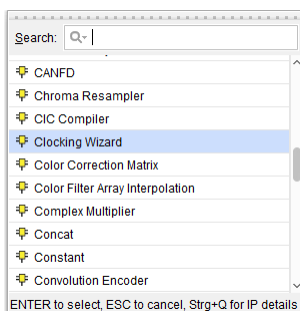
Nun wird das Beispielrechner_System als Modul zum Block Design hinzugefügt. Klicken Sie dazu mit rechts in die freie Fläche des Diagramm-Fensters und wählen Sie im Kontextmenü „Add Module“ aus.



Markieren Sie „Beispielrechner_System“ und klicken Sie auf „OK“. Eine Instanz des Beispielrechner-Systems wird im Diagramm angezeigt:



Fügen Sie einen „Clocking Wizard“ hinzu. Klicken Sie dazu mit rechts in die freie Fläche des Diagramm-Fensters und wählen Sie im Kontextmenü „Add IP“ aus.



Wählen Sie einen „Clocking Wizard“ aus und drücken Sie die Enter-Taste (oder Doppelklick).

Eine Instanz des Clocking Wizards wird im Diagramm angezeigt:



Durch Doppelklick auf den Clocking Wizard kann dieser konfiguriert werden.

Re-customize IP

Clocking Wizard (5.4)

Documentation IP Location

IP Symbol Resource

Show disabled ports

Component Name: clk_wiz_0

Clocking Options Output Clocks MMCM Settings Summary

The phase is calculated relative to the active input clock.

Output Clock	Port Name	Output Freq (MHz)		Phase (degrees)	
		Requested	Actual	Requested	Actual
<input checked="" type="checkbox"/> clk_out1	clk_out1	50	50.000	0.000	0.000
<input type="checkbox"/> clk_out2	clk_out2	100.000	N/A	0.000	N/A
<input type="checkbox"/> clk_out3	clk_out3	100.000	N/A	0.000	N/A
<input type="checkbox"/> clk_out4	clk_out4	100.000	N/A	0.000	N/A
<input type="checkbox"/> clk_out5	clk_out5	100.000	N/A	0.000	N/A
<input type="checkbox"/> clk_out6	clk_out6	100.000	N/A	0.000	N/A
<input type="checkbox"/> clk_out7	clk_out7	100.000	N/A	0.000	N/A

☐ USE CLOCK SEQUENCING

Clocking Feedback

Output Clock	Sequence Number
clk_out1	1
clk_out2	1
clk_out3	1
clk_out4	1
clk_out5	1
clk_out6	1
clk_out7	1

Source: ☒ Automatic Control On-Chip
☐ Automatic Control Off-Chip
☐ User-Controlled On-Chip
☐ User-Controlled Off-Chip

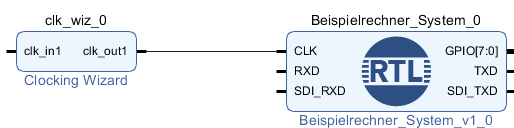
Enable Optional Inputs / Outputs for MMCM/PLL

Reset Type: ☒ Active High ☐ Active Low

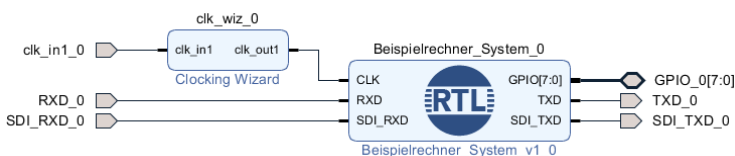
reset power_down input_clk_stopped locked clkfbstopped

OK Cancel

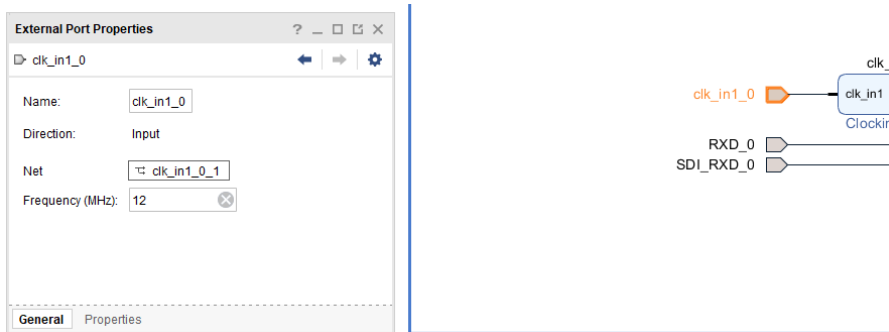
Im Tab „Output Clocks“ wird unter „Requested“ eine 50 eingetragen. Die Haken bei „reset“ und „locked“ werden entfernt. Klicken Sie anschließend auf „OK“.



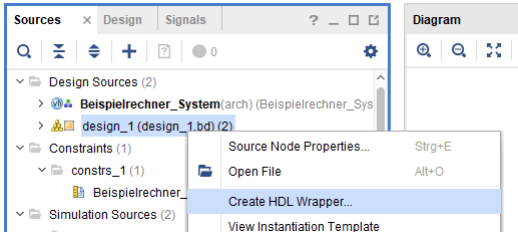
Verbinden Sie mit der Maus die Pins „clk_out1“ des Clocking Wizard und „CLK“ des Beispielsystems miteinander.



Alle anderen Pins werden als externe Ports konfiguriert. Dies geschieht durch Anklicken der am Pin angezeigten Linie mit der rechten Maustaste und Auswahl von „Make External“.

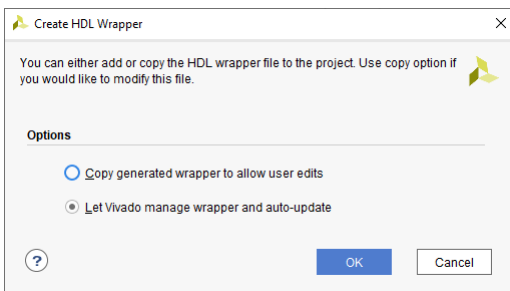


Klicken Sie den Port „clk_in1_0“ an und tragen Sie im Fenster „External Port Properties“ als Frequenz eine „12“ ein (entsprechend dem Eingangstakt).

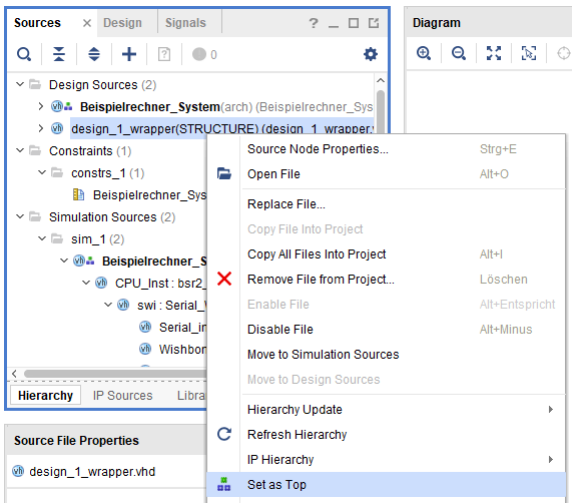


Wechseln Sie im Project-Manager zum Tab „Sources“, falls dieser nicht geöffnet ist.

Klicken Sie mit der rechten Maustaste auf den Namen der Komponente „design_1(design_1.bd)“ und wählen Sie im Kontextmenü „Create HDL Wrapper“ aus.

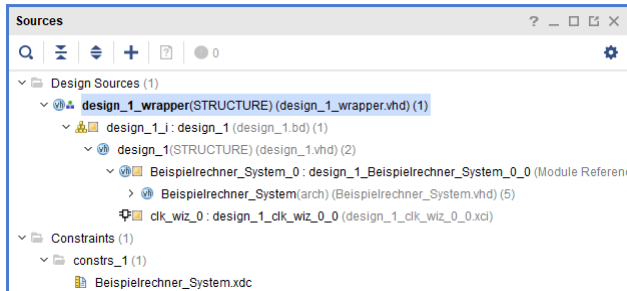


Klicken Sie auf „OK“. Eine neue Komponente „design_1_wrapper“ wird erzeugt, welche als Hülle für das vorhin erstellte Block Design dient. Nun muss diese Hülle als oberste Ebene für die Synthese eingestellt werden.

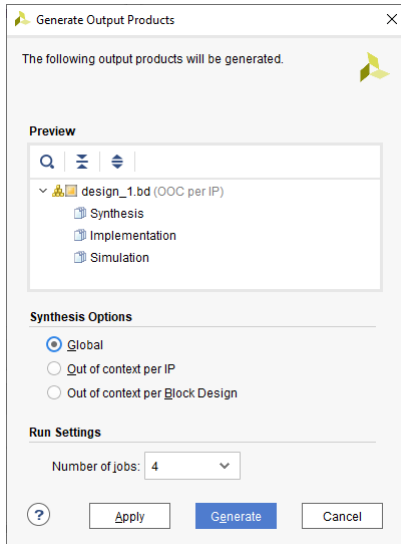


Klicken Sie dazu mit der rechten Maustaste im Project Manager auf den Namen der erzeugten Komponente „design_1_wrapper(STRUCTURE)“ und wählen Sie im Kontextmenü „Set as Top“ aus.

Im Project Manager ergibt sich nun die folgende Hierarchie:



Klicken Sie in der Kommandoleiste links auf „Generate Block Design“.



Ändern Sie bei „Synthesis Options“ die Einstellung auf „Global“ und klicken Sie auf „Generate“. Sie erhalten eine Erfolgsmeldung. Bestätigen Sie diese mit „OK“.

Klicken Sie nun in der Kommandoleiste links auf „Generate Bitstream“. Bestätigen Sie die Nachfragen mit „Yes“ und „OK“. Nun müssen Sie ein paar Minuten warten, bis oben rechts die folgende Meldung erscheint:

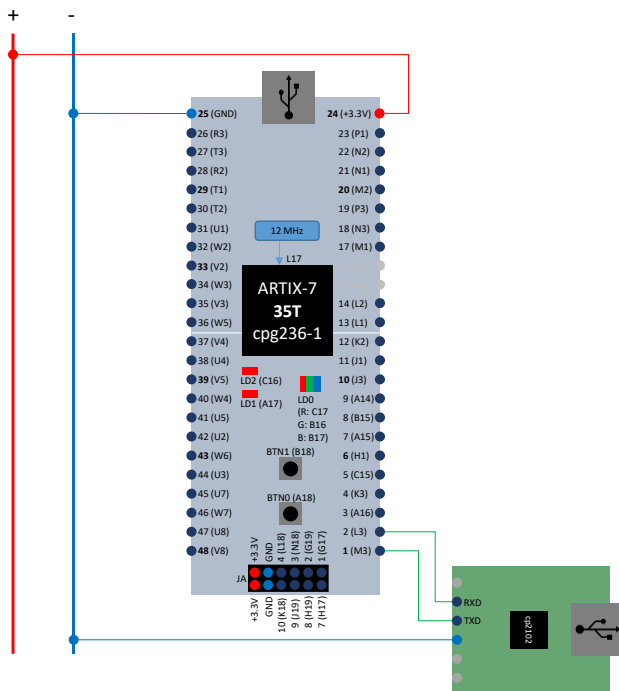
write_bitstream Complete ✓

Diese kritische Warnung (und die normalen Warnungen) können Sie ignorieren:

- ▼ Implementation (1 critical warning)
 - ▼ Write Bitstream (1 critical warning)
 - ! [filemgmt 56-176] Module references are not supported in manual compile order mode and will be ignored.

Aufgabe 1e: Hardware aufbauen, FPGA programmieren

Trennen Sie vor jeder Änderung der Schaltung die USB-Verbindungen zu ihrem Entwicklungsrechner!



Verbinden Sie den USB-UART-Adapter (grüne Platine) mit Jumper-Kabeln auf dem Breadboard mit dem FPGA-Board (blaue Platine).

Verbinden Sie anschließend den USB-UART-Adapter (grüne Platine) per Mini-USB-Kabel mit dem Entwicklungsrechner. Ermitteln Sie im Windows-Geräte-Manager den Namen des virtuellen COM-Ports für die serielle Schnittstelle des Systems (hier wäre das „COM6“):

- ▼ **Anschlüsse (COM & LPT)**
- Silicon Labs CP210x USB to UART Bridge (COM6)**

Verbinden Sie das FPGA-Board (blaue Platine) per Micro-USB-Kabel mit dem Entwicklungsrechner. Ermitteln Sie im Windows-Geräte-Manager den Namen des virtuellen COM-Ports für die Debug-Schnittstelle (hier wäre das „COM7“):

- ▼ **Anschlüsse (COM & LPT)**
- Silicon Labs CP210x USB to UART Bridge (COM6)**
- USB Serial Port (COM7)**

Öffnen Sie die Datei „Tools\program_fpga.bat“ im Editor. In dieser kann der Pfad zur Vivado-Installation angepasst werden:

```
5 rem HIER MUSS DER PFAD ZUR VIVADO-INSTALLATION ANGEPASST WERDEN
6 set XILINX=D:\Xilinx\Vivado\2017.4
```

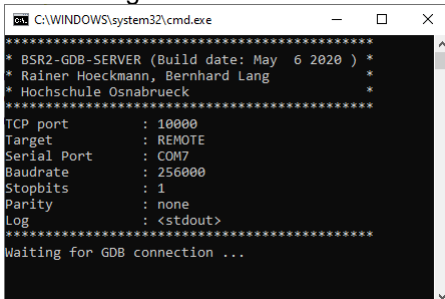
Führen Sie das Batch-File durch Doppelklick aus. Dadurch wird das FPGA mit der von Vivado erzeugten Bitstream-Datei programmiert.

Aufgabe 1e: Software ausführen


Öffnen Sie die Datei „Tools\run_remote.bat“ im Editor. In dieser kann der Name des virtuellen COM-Ports für die Debug-Schnittstelle angepasst werden:

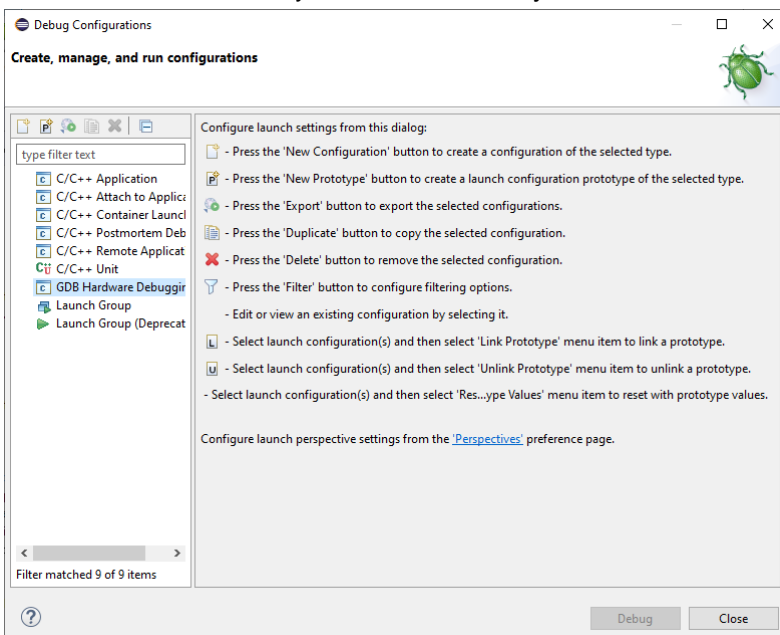
```
3 rem HIER MUSS DER NAME DES COM-PORTS angepasst werden.
4 set COM_PORT=COM7
```


Führen Sie das Batch-File durch Doppelklick aus. Eine eventuell auftretende Meldung der Windows-Firewall können Sie ignorieren und das Fenster durch „Abbrechen“ schließen. Der Debug-Server wartet nun auf eine Verbindungsaufnahme durch den Debugger:

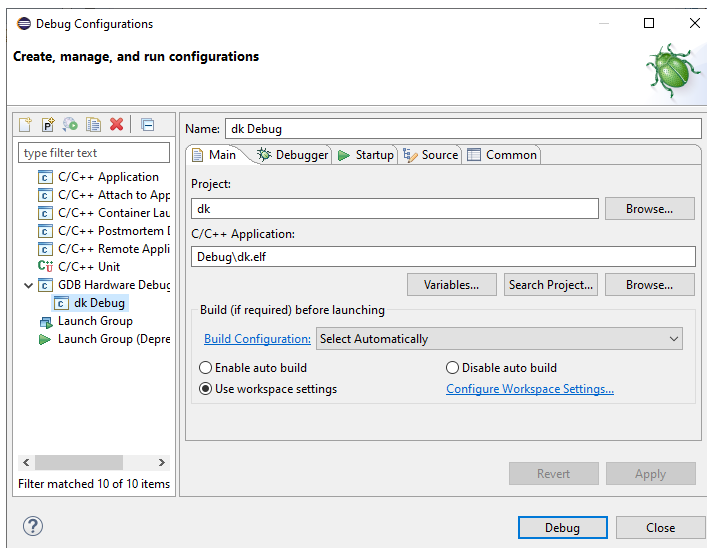


```
*****
* BSR2-GDB-SERVER (Build date: May  6 2020 ) *
* Rainer Hoeckmann, Bernhard Lang           *
* Hochschule Osnabrueck                     *
*****
TCP port      : 10000
Target        : REMOTE
Serial Port   : COM7
Baudrate      : 256000
Stopbits      : 1
Parity        : none
Log           : <stdout>
*****
Waiting for GDB connection ...
```

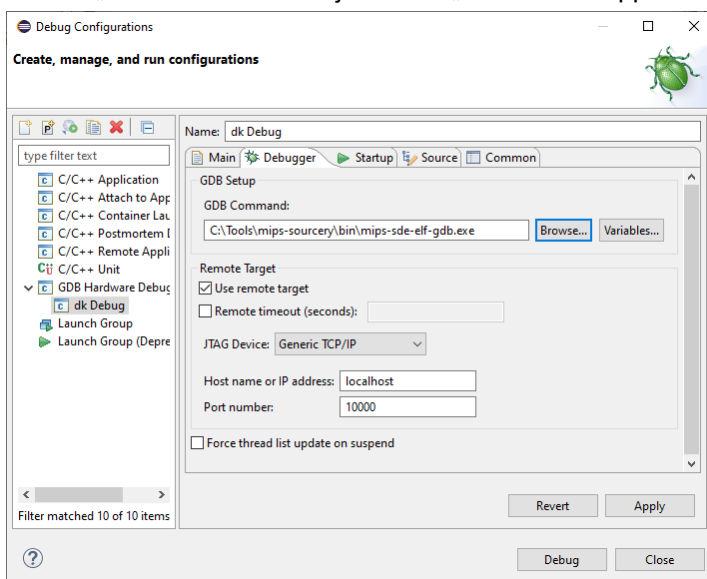
Öffnen Sie nun wieder Eclipse. Zum erstmaligen Ausführen des Programms muss eine Debug-Configuration erstellt werden. Markieren Sie das Projekt dk im Project Explorer und klicken Sie auf den kleinen schwarzen Pfeil neben dem Käfer-Symbol  in der Symbolleiste. Wählen Sie die Option „Debug Configurations...“.



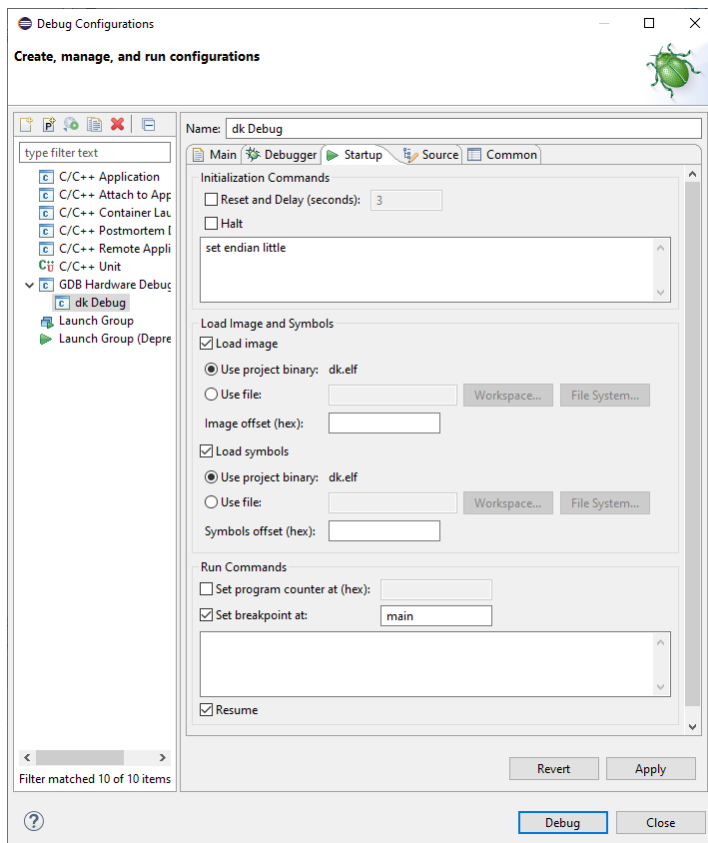
Markieren Sie die Option „GDB Hardware Debugging“ und klicken Sie dann auf das Symbol zum Anlegen einer neuen Konfiguration .



Im Tab „Main“ sollte der Projektname „dk“ und die Application „Debug\dk.elf“ bereits eingetragen sein.

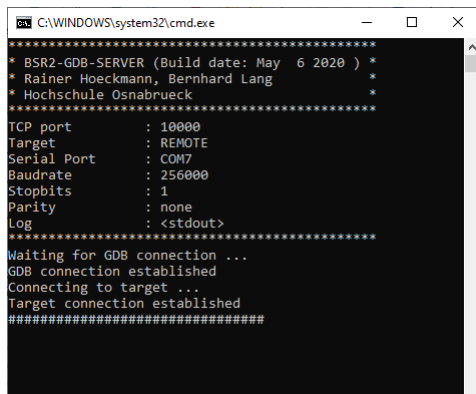


Im Tab „Debugger“ muss der Pfad zum Debugger-Programm in der Toolchain eingetragen (oder über Browse ausgewählt) werden.

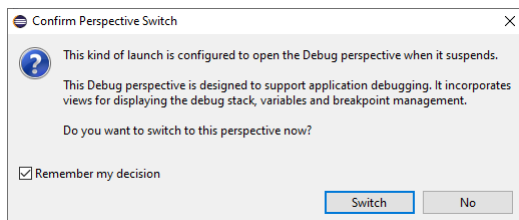


Im Tab „Startup“ wird in das Feld unter „Initialization Commands“ folgendes eingetragen: „set endian little“. Der Haken bei „Set breakpoint at“ wird gesetzt und in das Feld dahinter „main“ eingetragen. Der Haken bei „Resume“ wird gesetzt. Diese Einstellungen sorgen dafür, dass das Programm bis zum ersten Befehl in der Funktion „main“ ausgeführt (Resume) wird und dort an einem Haltepunkt (Breakpoint) stehen bleibt.

Klicken Sie nun auf Debug.



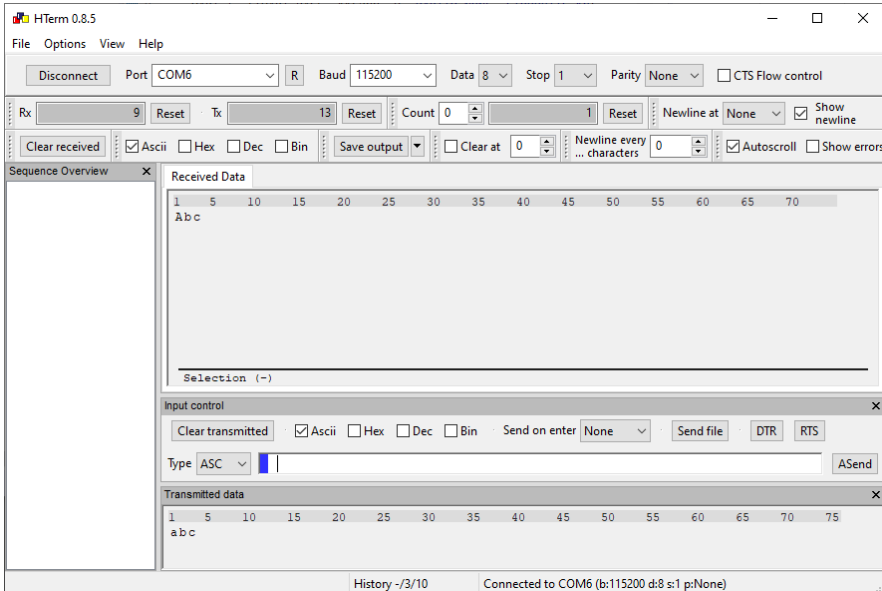
Im Fenster der Debug-Verbindung kann man sehen, dass der Debugger eine Verbindung aufgebaut hat und das Programm zum Beispielrechner-System im FPGA überträgt.



Eclipse fragt, ob beim Debuggen zur Debug-Perspektive gewechselt werden soll. Dies können Sie mit „Switch“ erlauben.

```
main.c
1 #include <gpio.h>
2 #include <uart.h>
3 #include <cpu.h>
4 #include <config.h>
5 #include <stdio.h>
6
7 int main()
8 {
9     UART_Init(UART_BASE, 115200, 8, PARITY_NONE, STOPBITS_10);
10 }
```

Die Ausführung stoppt am Beginn der Funktion „main“. Um das Programm weiter auszuführen, klicken Sie auf das Resume-Symbol (🟢).



Starten Sie auf dem PC ein Terminal-Programm (z.B. HTerm) und stellen Sie dort den COM-Port für die serielle Übertragung und die Übertragungsparameter wie im Programm vorgegeben ein (115200 Baud, 8 Datenbits, 1 Stoppbit, kein Paritätsbit).

Senden Sie mit dem Terminalprogramm Zeichen vom PC zum Beispielrechner-System. (Fast) alle Zeichen werden von diesem unverändert zurückgesendet, nur bei einem kleinen „a“ wird mit einem großen „A“ geantwortet.

Aufgabe 2a: Software ändern, Parallele Ein- und Ausgabe

Ergänzen Sie die Software an den mit „TODO“ gekennzeichneten Stellen, so dass der Zustand der Taster von den GPIO-Pins 0-3 eingelesen und auf den LEDs an den GPIO-Pins 4-7 ausgegeben wird. Nutzen Sie dabei die Funktionen „in32“ und „out32“ sowie die Konstanten GPIO_BASE, GPIO_DIR, GPIO_PINS und GPIO_DATA).

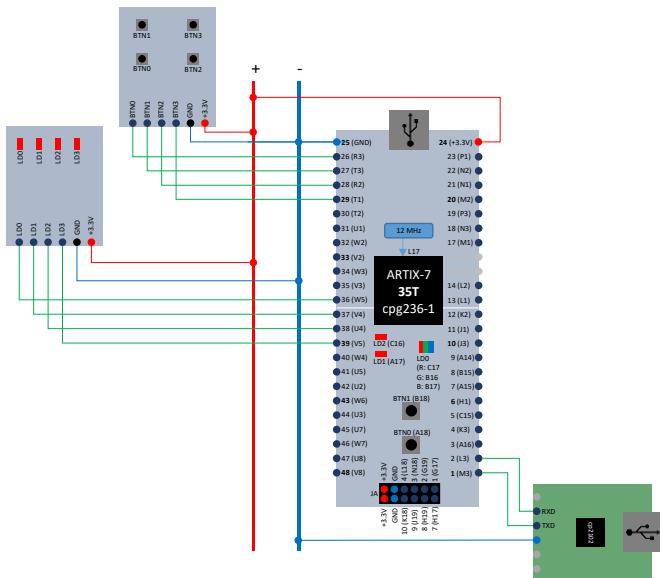
Nach erfolgreicher Übersetzung kann zunächst eine Simulation im VHDL-Simulator erfolgen. Erneuern Sie dazu die VHDL-Speicherbeschreibungen mit der Batch-Datei „Hex2VHDL.bat“ (vorher wieder zum Project Explorer umschalten).

Nun können Sie wieder die Testbench mittels der Kommandodatei „test_Beispielrechner_System_V3.do“ ausführen.

Jetzt sollte (wenn Sie alles richtig gemacht haben) auch der Test für Aufgabe 2 erfolgreich sein.

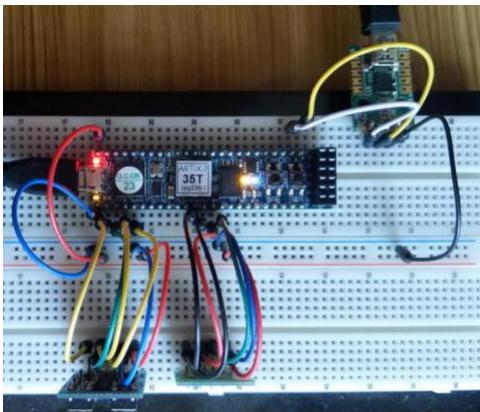
Aufgabe 2b: Geändertes Programm auf dem FPGA testen

Trennen Sie vor jeder Änderung der Schaltung die USB-Verbindungen zu ihrem Entwicklungsrechner!




Verbinden Sie die Module PmodBTN (Taster) und PmodLED (LED) mit Jumper-Kabeln auf dem Breadboard mit dem FPGA-Board (blaue Platine). Stellen Sie anschließend die beiden USB-Verbindungen zum PC wieder her.

Ein fertiger Aufbau sieht ungefähr so aus:



Zum Ausführen des Programms starten Sie zunächst den Debug-Server mit der Batch-Datei „Tools\run_remote.bat“.

Sie müssen keine neue Debug-Konfiguration erstellen, sondern können die letzte verwendete Konfiguration wiederverwenden (Schaltfläche ).

Führen Sie das Programm mit Resume aus und prüfen Sie durch Drücken der Taster, ob die Tasterwerte auf den LEDs ausgegeben werden.