



Ausgabe: 08. Februar 2015

Abgaben	{		Theorie	entfällt
			Praxis	entfällt
			Rücksprache	entfällt

Aufgabe 1: Synthese mithilfe von Vivado

Zum Abschluss des Praktikums soll für einige ausgewählte Komponenten eine Synthese durchgeführt werden.

Dazu wird aus Ihrer Hardwarebeschreibung eine Netzliste generiert, welche Komponenten aus einer Bibliothek so verbindet, dass die resultierende Schaltung Ihrem Design entspricht.

Hinweis: Leider lässt sich nicht der gesamte Sprachumfang von VHDL synthetisieren. Daher ist es möglich, dass Ihr Design nicht synthetisierbar ist. Sie sollten entsprechende Fehlermeldungen und den Synthese-Report gründlich lesen.

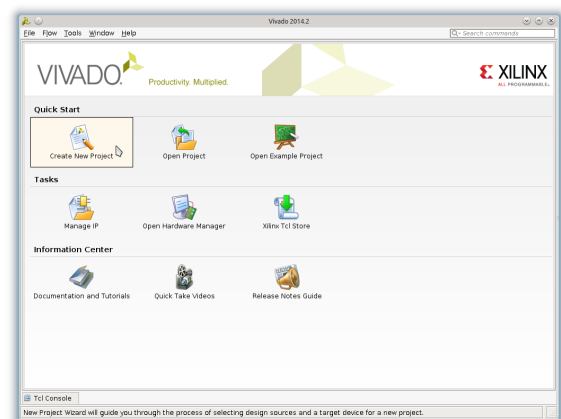
Verwendet wird Programm *Vivado* [1] der Firma *Xilinx*.

Vivado sucht in den Beschreibungen primär nach Funktionalitäten. Funktionalitäten sind zB. Speicherimplementierungen, FSM's, Addierer, etc. Die darum befindliche Logik wird in Wertetabellen abgebildet und optimiert.

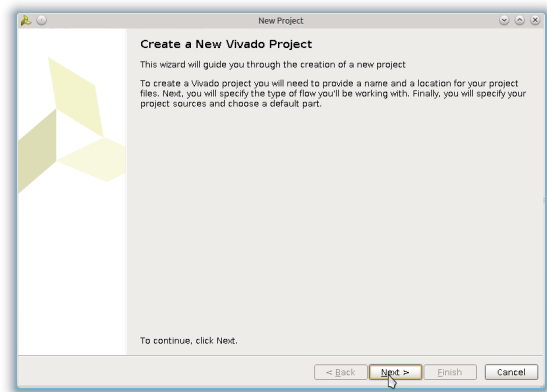
Im Folgenden soll mit Hilfe des Programms *Vivado* das Steuerwerk unseres Prozessors synthetisiert werden. Gehen Sie dabei so vor wie im folgenden beschrieben.

Starten Sie *Vivado*, indem Sie im Terminal erst das Kommando `techgi2pr` ausführen und anschließend `vivado &` eingeben und dies mit der ENTER-Taste bestätigen. Das Starten von *Vivado* kann einige Zeit dauern.

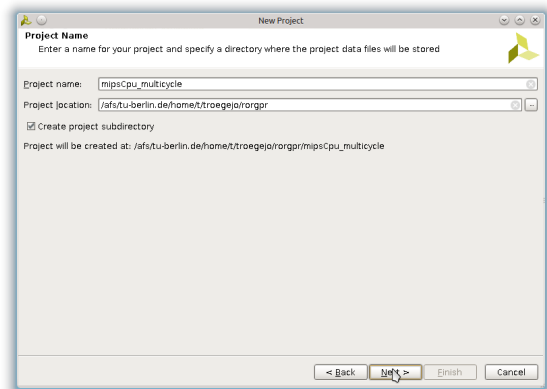
Erstellen Sie ein neues Projekt indem Sie auf **Create New Project** klicken.



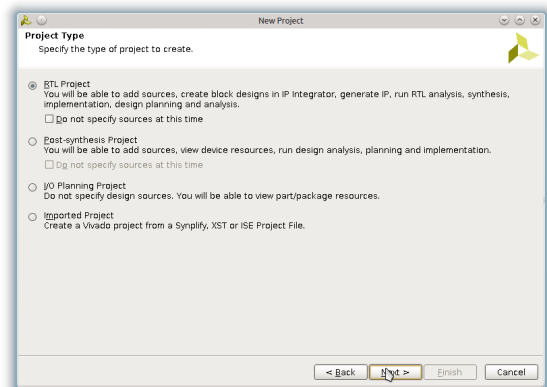
Es sollte sich nun ein Dialog öffnen. Bestätigen Sie mit **Next**.



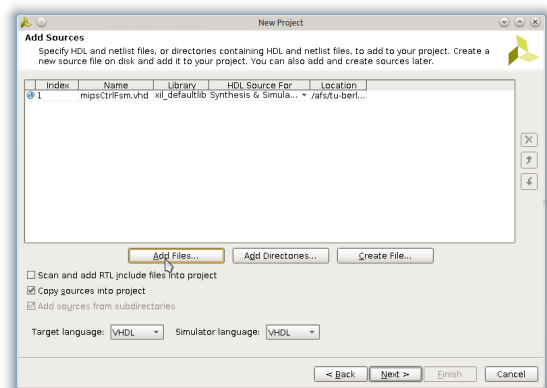
Geben Sie dem Projekt einen geeigneten Namen und geben Sie ein Verzeichnis zur Speicherung der Projektdaten an.



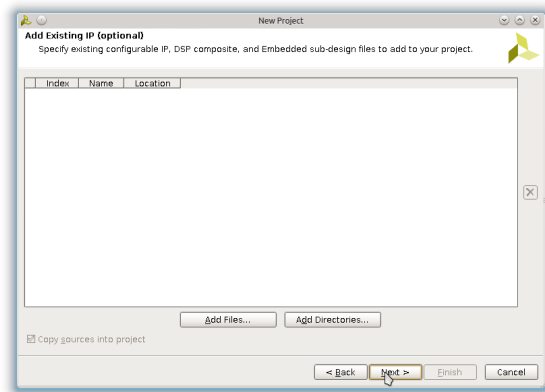
Als nächstes werden Sie gefragt, welche Art von Dateien Sie sofort hinzufügen wollen. Belassen Sie die Auswahl bei **RTL Sources**.



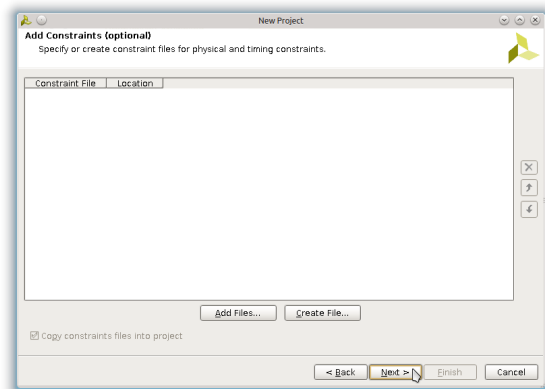
Fügen Sie nun Ihre bereits vorhandene VHDL-Beschreibung `mipsCtrlFsm.vhd` dem Projekt hinzu. Außerdem müssen die beiden Dateien `proc_config.vhd` und `mipsISA.vhd` aus den Vorgaben dem Projekt hinzugefügt werden.



Überspringen Sie die IP (Intellectual Property) Auswahl, da wir keine Designs aus 3. Hand nutzen.

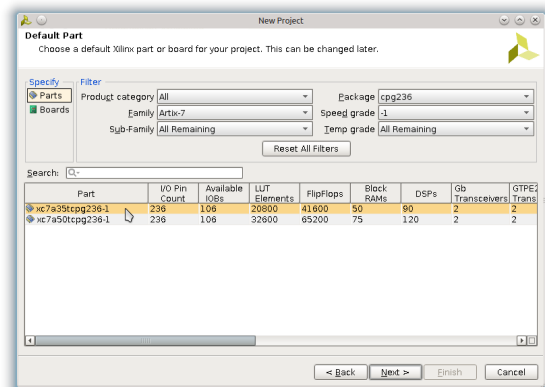


Überspringen Sie die Constraints Auswahl, da wir (noch) keine Hardware Spezifikationen für unser Design brauchen.

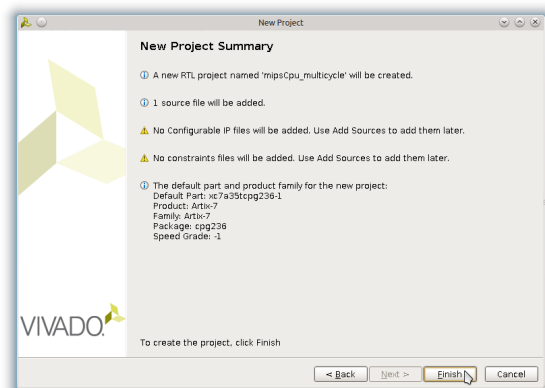


Wählen Sie als Entwicklungsgrundlage folgendes FPGA-Bauteil aus:

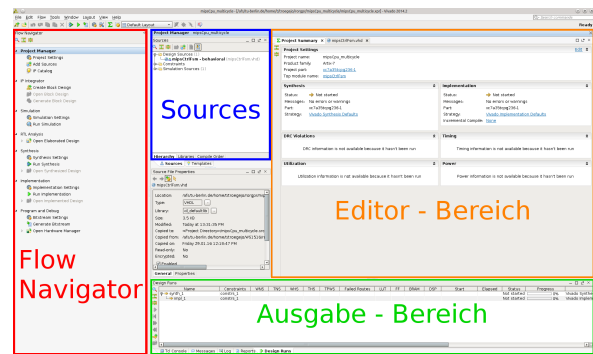
Family	Package	Speed grade	Part
Artix-7	cpg236	-1	xc7a35tcpg236-1



Beenden Sie die Projekterstellung und bestätigen Sie den folgenden Dialog.



Nun sehen Sie das Hauptfenster von Vivado. Links sehen Sie den Flow Navigator der alle wichtigen Funktionen beinhaltet. In dem Sources-Bereich (oben mitte) sehen Sie die Struktur ihres Projektes. Den größten Bereich nimmt der Editor-Bereich ein. In diesem werden ihre Quelldateien zum bearbeitet angezeigt. Sie können eine Quelldatei öffnen indem Sie doppelt auf die Datei im Sources-Bereich klicken. Außerdem werden hier die Projektzusammenfassung und diverse Report-Dateien angezeigt. Zwischen den einzelnen Fenstern des Editor-Bereiches können Sie mit der Tab-Leiste am oberen Ende des Bereiches wechseln. Am unteren Rand sehen Sie den Ausgabe-Bereich. Hier werden die ausgeführten Befehle im Konsolen-Tab, Warnungen im Message-Tab und Report-Dateien im Report-Tab aufgelistet. Die Leiste zum Wechseln der Tabs befindet sich am unteren Rand des Bereichs.



Wo Sie die **RorgPrSimLib** genutzt haben, ändern Sie bitte folgendes in Ihrer Implementierung:

Ersetzen Sie

```
library ROrgPrSimLib;
use ROrgPrSimLib.proc_config.mips_ctrl_state_type;
use ROrgPrSimLib.mipsISA.all;
```

durch

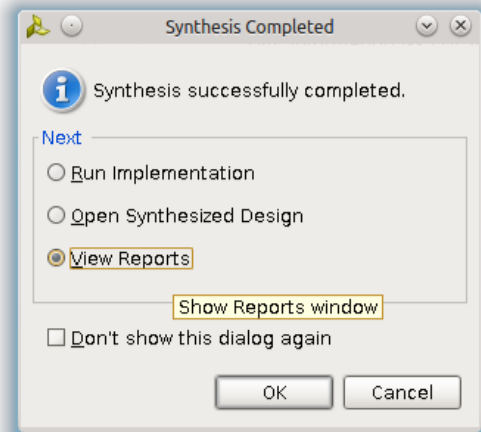
```
library work;
use work.proc_config.all;
use work.mipsISA.all;
```



Erzeugen Sie den Synthese Bericht, indem Sie links im Flow Navigator auf den Button **Run Synthesis** drücken.

Nach dem Abschluss der Synthese wird Ihnen eine Meldung angezeigt, welche angibt ob die Synthese erfolgreich war. Sollte die nicht der Fall sein, schließen Sie die Meldung mit dem Button **ok** nachdem Sie die Option *View Messages* aktiviert haben. Nun sollte der *Message*-Tab im Ausgabe-Bereich ausgewählt sein, indem die Probleme bei der Synthese aufgelistet sind.

Sollte die Synthese erfolgreich sein wählen Sie die Option *View Reports*. Nun sollte der *Report*-Tab im Ausgabe-Bereich ausgewählt sein und es wurden zwei Report-Dateien () erstellt. Diese können durch einen Doppelklick auf den Listeneintrag im Editor-Bereich geöffnet werden.

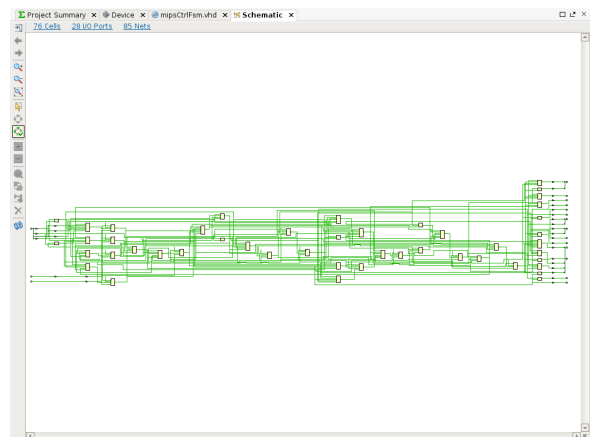


Studieren Sie diese Report-Dateien hinsichtlich der folgenden Fragen.

- Welche Hardwarekomponenten wurden erkannt?
- Welche Ressourcen werden zu Umsetzung Ihres Designs benötigt?


Falls die Synthese erfolgreich war können Sie fortfahren.

Erzeugen Sie das Registertransfer-Schaltbild, indem Sie im Flow Manager das Menü *Open Synthesized Design* öffnen und den Eintrag *Schematic* anklicken.



Im Editor-Bereich sollte nun der interne Aufbau ihres endlichen Automaten dargestellt werden. Die Schaltung die Sie sehen ist relativ komplex. Versuchen Sie den Teil für die Zustandsfortschaltung nachzuvollziehen.

Aufgabe 2: Synthese des restlichen Prozessors

Versuchen Sie weitere Einzel-Synthesen erfolgreich durchzuführen. Fügen Sie dazu die benötigten Dateien dem Projekt hinzu indem Sie am oberen Rand des *Sources*-Bereichs den Button  (*Add Sources*) auswählen und in dem sich öffnenden Fenster im ersten Schritt den Punkt *Add or Create Design Sources* auswählen und im nächsten Schritt die Dateien hinzufügen. Nach dem Schließen des Dialogs werden die hinzugefügten Dateien in der Struktur im *Sources*-Bereich angezeigt. Sollten einige Module fehlen, werden diese auch in der Struktur angezeigt und als fehlend markiert.

Erklären Sie nun jeweils die zu synthetisierende Komponente zum Topmodul (per Rechtsklick im Sources-Bereich auf die Quelldateien und auswählen des Menüpunkts *Set as Top*). Zur Synthese der Module müssen alle Generics einen Default-Wert haben, der für die Synthese verwendet wird. Der Default-Wert wird nach der Definition des Generics in der entity festgelegt. Verwenden Sie zum Setzen der Default-Werte die Konstanten aus der Datei `proc_config.vhd`.

```
entity regFile is
  generic (NUM_REGS : integer := NUM_REGS;
          LOG2_NUM_REGS : integer := LOG2_NUM_REGS;
          REG_WIDTH : integer := REG_WIDTH);
  port (clk : in std_logic;
        ...
```

Sehen Sie sich die Synthese-Reports genau an und beseitigen Sie eventuelle Probleme.

1. Synthetisieren Sie das RegisterFile und analysieren Sie die Syntheseberichte. Wie viele 1-Bit-Register werden verwendet. Wie lässt sich dies erklären?
2. Synthetisieren Sie nacheinander Ihre Alu in der Carry-Ripple- und der Carry-Select-Variante, indem sie jeweils die entsprechende Architektur des 1-Bit-Addierers instanziiieren. Vergleichen Sie den Ressourcenverbrauch beider Implementierungen. Erklären Sie Ihre Funde und beachten Sie dabei die Eigenschaften von FPGAs im Vergleich zu Chips auf Transistorbasis.
3. Versuchen Sie den gesamten MultiCycle-Prozessor zu synthetisieren. Versuchen Sie Probleme zu beseitigen und das Design synthesefähig zu machen. Dazu müssen alle Komponenten von ihnen implementiert vorliegen, sodass keine Komponenten aus der `RorgPrSimlib` verwendet werden. Eine Implementierung des RAMs für den Multicycle-Prozessors finden Sie in den Vorgaben. Die Vorgaben enthalten außerdem eine Datei, welche den initialen Inhalt des Speichers vorgibt. Fügen Sie deshalb sowohl die Datei `clip.mif`, als auch die Datei `flashRAM.vhd` ihrem Projekt hinzu um den RAM des Multicycle-Prozessors zu verwenden. Der Generic `INIT_FILE_NAME` wird auf den string `"clip.mif"` gesetzt. Lassen Sie sich mithilfe der Schematic das gesamte Design grafisch umsetzen und versuchen Sie die Einzelkomponenten darin zu identifizieren. Entnehmen Sie dem Synthesebericht den Ressourcenverbrauch.

Aufgabe 3: Implementierung des Multicycle-Prozessors

Abschließend möchten wir den synthetisierten Multicycle-Prozessor implementieren und auf einem FPGA-Board (*Digilent Basy3*) testen. Um den Prozessor bei der Ausführung der Programmes beobachten zu können werden die LEDs des FPGA-Boards für das Anzeigen der Automaten-Zustände und die 7-Segment-Anzeigen für die Ausgabe des PCs verwendet. Um eine Datei für die Programmierung des FPGAs (*Bitstream*) zu generieren muss zuerst eine Schnittstelle zwischen dem Board und der Prozessor geschaffen werden. Diese Schnittstelle ist in den Vorgaben enthalten. Fügen Sie ihrem Design zuerst die Datei `mispCpu_mc_wrapper.vhd` hinzu und erklären Sie sie zum Top-Modul. Fügen Sie danach die Datei `mispCpu_mc_wrapper.xdc` hinzu, wobei bei dieser Datei im ersten Schritt des Hinzufügen-Dialogs der Punkt *Add or Create Constraints* ausgewählt werden muss.

Nun können Sie das gesamte Projekt synthetisieren. Wenn dies erfolgreich war können Sie die Implementierung durchführen. Verwenden Sie dazu auf den Eintrag **Run Implementation** im *Flow Manager*. Nach dem erfolgreichen Abschluss der Implementierung kann der *Bitstream* generiert werden. Klicken Sie hierzu im *Flow Manager* auf den Punkt **Generate Bitstream**. Diese Funktion generiert, sofern keine Fehler auftreten, eine Datei die zum Programmieren des Boards verwendet werden kann. Nach der fehlerfreien Generierung ist der *Bitstream* in der Datei `$PROJEKTVERZEICHNIS/$PROJEKTNAME$.runs/impl_1/mispCpu_mc_wrapper.bit` gespeichert. Diese Datei kann nun

LED	State
0	INSTR_FETCH
1	INSTR_DECODE
2	BRANCH_COMPL
3	JUMP_COMPL
4	EXECUTION
5	RTYPE_COMPL
6	MEM_ADDR_CALC
7	MEM_WRITE
8	MEM_READ
9	MEM_READ_COMPL

Tabelle 1: Mapping der einzelnen LEDs auf die Zustände des Multicycle-Automaten

auf des FPGA-Board übertragen werden.

Wenn das Board programmiert wurde, leuchtet die LED 0 und die 7-Segment-Anzeige zeigt 0000 an. Sie können nun eine Taktflanke erzeugen indem sie den mittleren Taster auf dem Board drücken und nach kurzer Zeit loslassen. Außerdem können Sie den Prozessor zurücksetzen indem Sie den unteren Taster gedrückt halten und eine Taktflanke erzeugen. Eine Übersicht über die LEDs und die zugehörigen zustände befinden sich in Tabelle 1.

Literatur

[1] Vivado design suite user guide, 2014.