Projekterstellung und -verwaltung in der Entwicklungsumgebung PlanAhead

Handout zum 0. Praktikumstermin

Hardwarepraktikum SoSe 17

Diese Anleitung soll die Grundlagen zur Arbeit mit der FPGA-Entwicklungsumgebung PlanAhead der Firma Xilinx sowie des Simulationswerkzeugs ModelSIM von Mentor vermitteln. Sie umfasst das Anlegen und Verwalten von Projekten, Erstellung und Test einfacher HDL-Beschreibungen, die Synthese in eine FPGA-geeignete Darstellungsform, die Simulation des eigenen Designs sowie das Übertragen dieser Konfiguration in den FPGA.

1 Arbeitsumgebung

Die Arbeit erfolgt an Arbeitsplätzen mit SunRay-Terminals auf den Servern des IRB mit Ubuntu Umgebung.

Im Praktikum werden die Werkzeuge ModelSIM von Mentor Graphics zur Simulation bzw. Xilinx PlanAhead / XST zur Logiksynthese verwendet.

Das Einspielen der Konfiguration auf die FPGAs unmittelbar an den SunRays ist bislang nicht möglich. Die Konfigurationsdateien müssen daher per Email (`hwpr@aes.tu-berlin.de`) auf den Rechner des Betreuers übermittelt werden. Als Minimalprojekt wird ein einfaches Oder-Gatter formuliert und getestet.

Falls Ihre sunRay noch nicht mit *IRB-Ubuntu* verbunden ist wählen Sie in der Session-Box IRB-Ubuntu aus.

Um die Werkzeuge nutzen zu können, müssen ein paar Umgebungsvariablen[1] gesetzt werden. Dies erfolgt per Script. Öffnen Sie ein Terminal und geben Sie folgenden Befehl ein:

```
source /afs/tu-berlin.de/units/Fak_IV/aes/scripts/hwptienv
```

[1] Pfade der Werkzeuge (PATH) sowie License-Server (LM_LICENSE_FILE)

Danach sollten alle Werkzeuge die im Rahmen des Praktikums genutzt werden aus diesem Terminal aufrufbar sein.
Sie können diese Zeile auch in ihre `.bashrc` eintragen, beachten Sie jedoch das dies Einfluss auf andere Werkzeuge wie zum Beispiel den gcc hat!

2 Anlegen eines neuen Projekts

Legen Sie ein Verzeichnis für PlanAhead-Projekte in Ihrem Homeverzeichnis an und vergeben Sie einen aussagekräftigen Namen ohne Leerzeichen[2] (beispielsweise `PlanAhead_WORK`). Legen Sie darin einen zweiten Ordner an, der das heutige Projekt enthalten wird. Vergeben Sie einen Namen, z.B. `Tutorial`.

Starten Sie die Entwicklungsumgebung PlanAhead durch Eingabe von `planAhead&` in einem Terminal. Legen Sie unter *Create New Project* ein neues Projekt an und geben Sie den Speicherort des Projektverzeichnisses an. Achten Sie unbedingt darauf, dass weder Projektname noch Projektpfad Leerzeichen enthalten, so wie es in Abbildung 1 dargestellt ist.

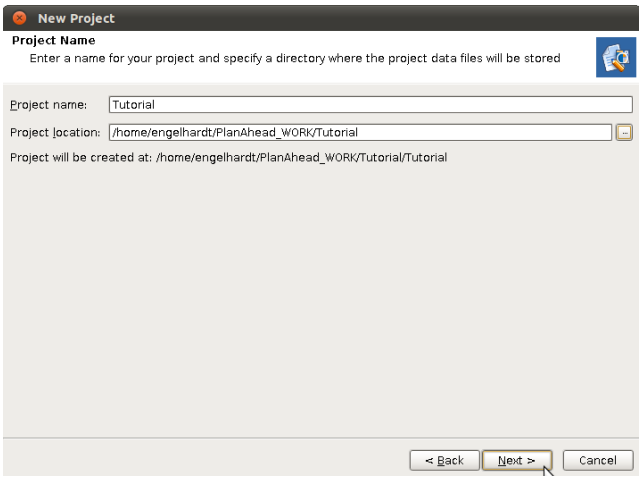

Abbildung 1: Projekterstellung - Name und Pfad

Als nächstes, wählen Sie als *Design Source* den Typ *Specify RTL Sources*. Die nächsten drei Fenster bestätigen Sie ohne Änderungen. Sie könnten hier neue oder bereits existierende Dateien zum Projekt hinzufügen.

Im nächsten Fenster, siehe Abbildung 2, wird der FPGA-Typ des Testboards ausgewählt. Diese Optionen sind auch später noch veränderbar.

Geben Sie folgende Vorgaben ein:

Family:	Spartan 3E
Sub-Family:	Spartan 3E
Package:	FG320
Speed Grade:	-4

[2]Leerzeichen oder sonstige nicht-alphanumerische Zeichen außer '_' im Pfad führen zur Arbeitsverweigerung von sämtlichen Xilinx-Tools. Zusätzlich müssen Projektnamen mit einem Buchstaben anfangen.

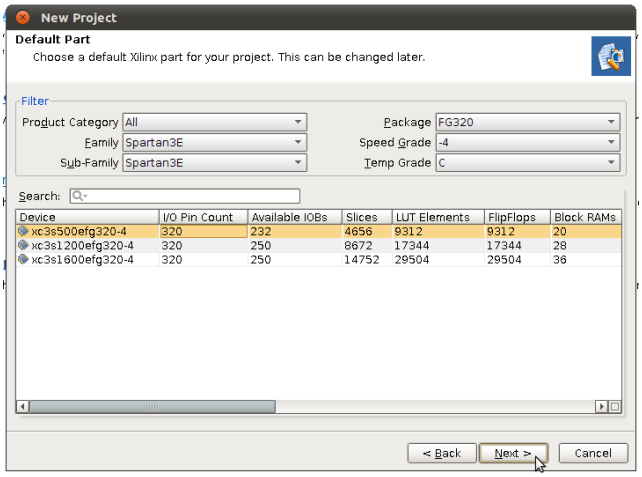

Abbildung 2: Projekterstellung - FPGA Auswahl

Wählen Sie dann im unteren Teil des Fensters das Board mit dem Bezeichner `xc3s500efg320-4` aus.

Abschließend erhalten Sie eine Zusammenfassung der gewählten Einstellungen, bestätigen Sie diese durch einen Klick auf *Finish*. Sie sehen nun ein Fenster, ähnlich dem aus Abbildung 3, mit einer Übersicht über den Projektstatus, ein Fenster für die zum aktuellen Projekt gehörenden Quellen (*Sources*), in der Seitenleiste die auf das Projekt und seine Quellen anwendbaren Aktionen (*Flow Navigator*) und einen Bereich für die Textausgaben der Werkzeuge (*TCL Console*, *Messages*, etc).

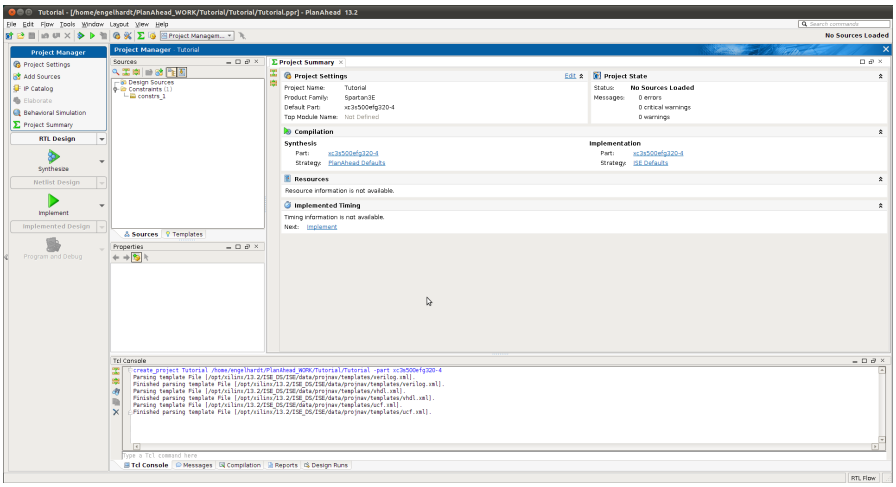

Abbildung 3: PlanAhead nach der Projekterstellung

3 Anlegen und Einbinden einer VHDL-Datei

PlanAhead kann Projektquellen an zwei Orten speichern: lokal im Projektverzeichnis oder in einem getrennt verwalteten Ordner. Da wir im Rahmen des Praktikums zusätzlich mit ModelSIM arbeiten möchten ist es ratsam die Quelldateien in einem separaten Ordner anzulegen. Falls Sie später nur

mit den in PlanAhead integrierten Xilinx-Tools arbeiten sollten, können Sie die Option *Local to Project* wählen und PlanAhead die Quellen verwalten lassen. Wenn Sie jedoch weitere Tools wie z.B. ModelSim zur Simulation verwenden möchten, empfiehlt es sich, die Quellen in einen eigenen Ordner außerhalb des Projektverzeichnisses zu speichern, um einfacher darauf zugreifen zu können und Konflikte zu vermeiden.

Für das Hardwarepraktikum wird daher für jedes Projekt folgende Verzeichnisstruktur empfohlen:

- ▽ PlanAhead_WORK
  - ▽ Tutorial
    - ▷ Tutorial - Xilinx Projektverzeichnis
    - ▷ src - Quellenverzeichnis
    - ▷ work - ModelSim Projektverzeichnis

Über den *Add Sources* Button in der Seitenleiste können Sie nun neue oder bereits existierende Dateien zu Ihrem Projekt hinzufügen. Für synthetisierbare VHDL-Dateien wählen Sie die Option *Add or Create Design Sources* im ersten Dialogfenster. Für Test Benches, die nicht synthetisiert werden, ist hier später *Add or Create Simulation Sources* zu wählen.

Eine leere Datei im Quellenverzeichnis anzulegen, die entity von Hand oder mit Unterstützung eines geeigneten Editors zu schreiben und die Quelle mit *Add Sources* dem Projekt hinzuzufügen dürfte langfristig der schnellste und unkomplizierteste Weg der Projektverwaltung sein.

Wichtig: Achten Sie dabei immer darauf, das Häkchen *Copy Sources into Projekt* zu entfernen, ansonsten arbeitet PlanAhead mit einer Kopie und nimmt Änderungen an der Originaldatei nicht mehr wahr.

Legen Sie in Ihrem Quellenordner die leere VHDL-Datei `or_gate.vhd` an. Öffnen Sie die Datei in einem Editor Ihrer Wahl und übernehmen Sie die Entitydeklaration[3] aus Listing 1.

```
library ieee;
use ieee.std_logic_1164.all;

entity or_gate is
  port(   INPUT_1:  in std_logic;
      INPUT_2:  in std_logic;
      OUTPUT : out std_logic );
end entity or_gate;

architecture structure of or_gate is
begin
end architecture structure;
```

Listing 1: Entity `or_gate`

[3]Der architecture-Teil gehört natürlich nicht zur entity-Deklaration. PlanAhead bindet die VHDL-Datei aber nur mit gültiger architecture-Deklaration als Quelle ins Projekt ein.

Speichern Sie die Datei und fügen Sie sie anschließend in PlanAhead per *Add Source* → *Add or Create Design Sources* → *Add Files...* dem Projekt hinzu.

Die Komponente taucht jetzt im Sources-Fenster unterhalb der default-Library `work` auf, wie Abbildung 4 verdeutlicht. Ein Doppelklick auf die neue Datei im *Sources*-Fenster öffnet den Texteditor.

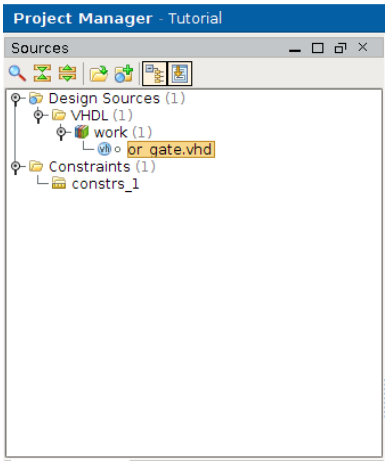

Abbildung 4: Entity und Architecture einer Projektquelle

4 PlanAhead Texteditor anpassen oder anderen Editor einbinden

Die VHDL-Dateien können natürlich unabhängig von der Entwicklungsumgebung in Ihrem bevorzugten Editor bearbeitet werden. Doch auch beim Öffnen von Dateien aus PlanAhead sind Sie nicht an den mitgelieferten Texteditor gebunden. Um Quellen aus der Entwicklungsumgebung heraus mit dem Editor der Wahl öffnen zu können, gehen Sie folgendermaßen vor: Öffnen Sie die PlanAhead-Voreinstellungen durch einen Klick auf *Tools* → *Options*. Wenn Sie in der Kategorie *General* ein wenig herunterscrollen, finden Sie ein Feld *Text Editor*, dass der Anpassung des eingebauten Editors dient. Hier kann ein anderer Standardeditor angegeben werden. Taucht im DropDown-Menü der von ihnen gewünschte Editor nicht auf, wählen Sie *Custom Editor...* und geben den Befehl zur Ausführung Ihres Editors entsprechend den dargestellten Erläuterungen ein.

5 Syntaxcheck und RTL Übersicht

Vervollständigen Sie die architecture von `or_gate.vhd`, z.B. folgendermaßen[4] wie in Listing 2 vorgeschlagen.

```
architecture structure of or_gate is
begin
  OUTPUT <= INPUT_1 or INPUT_2;
```

[4]Der Name der architecture ist im Prinzip beliebig und für die Funktionsbeschreibung des Oder-Gatters existieren zahlreiche Möglichkeiten

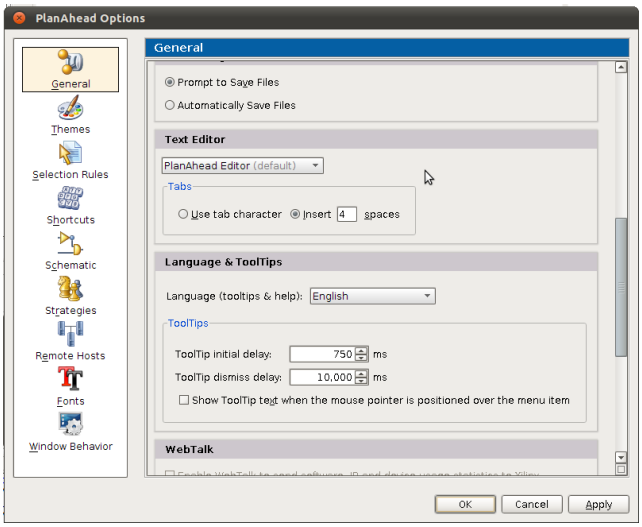

Abbildung 5: Festlegen des Standardtexteditors

```
end architecture structure;
```

Listing 2: Mögliche architecture für das Oder-Gatter

Elaborate ist der erste Schritt zur Überprüfung Ihres VHDL Codes. Dabei wird Ihre Hardwarebeschreibung auf eine Register Transfer Level (RTL) Struktur abgebildet[5]. Die dabei verwendeten Bausteine (Gatter, Register, Multiplexer etc.) sind noch unabhängig von der tatsächlich verwendeten Zieltechnologie. In diesem Fall wird ein Schaltungssymbol angezeigt. Ein Doppelklick auf dieses Symbol öffnet die nächst tiefere Ebene. Wie in Abbildung 6 dargestellt wird ein Oder-Gatter angezeigt, obwohl das FPGA nicht direkt über ein derartiges Gatter verfügt.

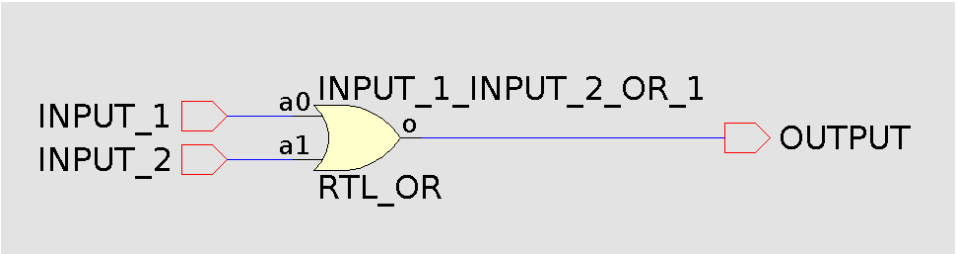

Abbildung 6: RTL Schematic für das Oder-Gatter

Beim ersten Betätigen von *Elaborate* kann PlanAhead Sie nach dem Topmodul fragen. Innerhalb eines Projekts stellt immer eine Komponente das Topmodul dar. Diese Komponente und alle in der Hierarchie darunter liegenden (durch Instantiierung im Topmodul) werden im FPGA realisiert. Andere Komponenten, die nicht in der Hierarchie unterhalb des Topmoduls liegen, werden ignoriert. Geben Sie hier `or_gate` ein. Falls Sie dies später noch einmal ändern möchten, kann eine Designkomponente im Sources-Fenster in der Hierarchy-Ansicht durch auswählen der Komponente und anschließendem (!) Rechtsklick zum Topmodul erklärt werden (Menüpunkt ”Set as Top”).

[5]Falls das Fenster *RTL Schematic* sich nicht automatisch öffnet, wählen Sie in der Toolbar das Layout *Design Analysis*.

Sie sollten sich angewöhnen, gelegentlich einen Blick auf das RTL Schematic zu werfen. Fehler in Ihrem Design fallen hier oft frühzeitig auf. Vergessen Sie beispielsweise, Ihre Modulausgänge zu beschalten, bleibt das Schematic vollständig leer.

6 Simulation mit Test Bench in ModelSim

Der Test von Komponenten wird mit einer Test Bench durchgeführt. Dies ist eine spezielle VHDL Entity, die keine Eingänge oder Ausgänge enthält und nicht synthesefähig ist. In dieser wird die zu testende Komponente instanziiert und Ihre Eingangssignale mithilfe der Simulationskonstrukte `wait` und `after` generiert.

Erstellen Sie eine Datei namens `or_gate_tw.vhd` in Ihrem Quellenverzeichnis. Fügen Sie den Code aus Listing 3 ein. Bitte beachten Sie, dass Copy&Paste zu Syntaxfehlern führen kann, weshalb Abtippen zu bevorzugen ist.

```
library ieee;
use ieee.std_logic_1164.all;

entity or_gate_tw is

end or_gate_tw;

architecture testbench of or_gate_tw is

  signal A : std_logic := '0';
  signal B : std_logic := '0';
  signal C : std_logic;

  component or_gate
    port ( INPUT_1 : in std_logic;
    INPUT_2 : in std_logic;
    OUTPUT : out std_logic );
  end component;

  begin
    A <= '1' after 20 ns, '0' after 60 ns;

    B <= '1' after 40 ns, '0' after 80 ns;

    uut : or_gate
      port map ( INPUT_1 => A,
      INPUT_2 => B,
      OUTPUT => C );



end testbench;
```

Listing 3: Test Bench für das Oder-Gatter

Wechseln Sie in einem Terminal in das Verzeichnis `~/PlanAhead_WORK/Tutorial/` und starten Sie ModelSim mit dem Befehl `vsim&`.

ModelSim benötigt ein Arbeitsverzeichnis namens `work`. Sie erstellen das Arbeitsverzeichnis (falls es noch nicht existiert) einmalig unterhalb des jeweiligen Projektverzeichnisses. Wählen Sie dazu im Menü *File* → *New* → *Library...* – wenn Sie das Arbeitsverzeichnis noch nicht eingerichtet haben sollte hier automatisch `work` in beide Textfelder eingetragen sein, sodass Sie nur noch bestätigen müssen – oder tragen Sie im *Transcript*-Fenster (die „Konsole“ von ModelSim im unteren Bildschirmteil) `vlib work` gefolgt von `vmap work work` ein. Das Arbeitsverzeichnis bleibt zukünftig erhalten.

Jetzt müssen alle gewünschten Quelldateien nach `work` kompiliert werden. Unter der Voraussetzung, dass Ihre Quelldateien im Verzeichnis `src` im Projektverzeichnis liegen, gelingt dies durch die Eingabe von `vcom -work ./work/ ./src/*.vhd`
Falls Sie die grafische Oberfläche bevorzugen, öffnen Sie im Menü *Compile* → *Compile...*, wählen Sie alle Quelldateien aus und klicken Sie auf *Compile*. Schließen Sie das Fenster mit *Done*, wenn die Kompilation abgeschlossen ist.

Im *Workspace*-Fenster von ModelSim finden Sie den Bibliothekseintrag work und darin die Datei `or_gate_tw`, ähnlich wie in Abbildung 7.

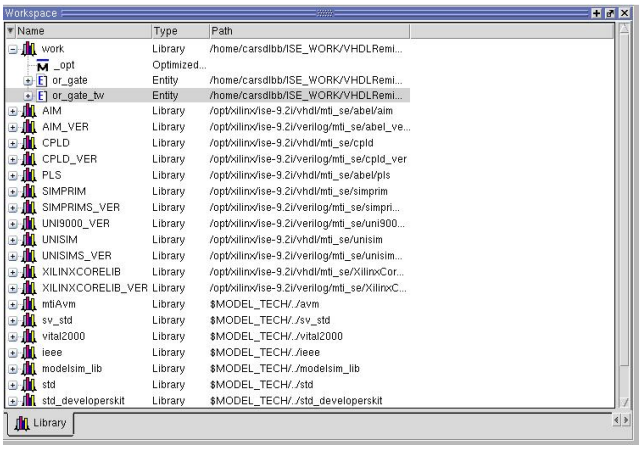

Abbildung 7: ModelSim Workspace

Doppelklicken Sie darauf. Es öffnet sich unter anderem das *Objects*-Fenster von ModelSim. Markieren Sie hier alle Einträge und wählen Sie in deren Kontextmenü *Add* → *To Wave* → *Selected Signals*, wie in Abbildung 8 gezeigt.

Es öffnet sich das Wave-Fenster mit den hinzugefügten Signalen. Klicken Sie auf *Run* oder geben Sie in das *Transcript*-Fenster run 100 ns ein. Im *Wave*-Fenster sehen Sie die von Ihnen erzeugten Stimuli auf `INPUT_1` und `INPUT_2` sowie das daraus erzeugte Ausgangssignal `OUTPUT`, wie in Abbildung 9 gezeigt.

Beenden Sie ModelSim wenn Sie sich von der korrekten Funktion der getesteten Komponente überzeugt haben. Dateien, die einmal per `vcom`[6] im ModelSim -Arbeitsverzeichnis bekannt gemacht wurden, können weiterhin außerhalb oder innerhalb von ModelSim bearbeitet werden. Im Kontextmenü eines Moduls im *Workspace*-Fenster von ModelSim können Sie durch *Recompile* eine veränderte Datei neu kompilieren.

[6]Die Dateien verbleiben an dem Ort, aus dem heraus Sie durch `vcom` erstmalig kompiliert wurden, hier also im Ordner `src` unterhalb des Projektverzeichnises.

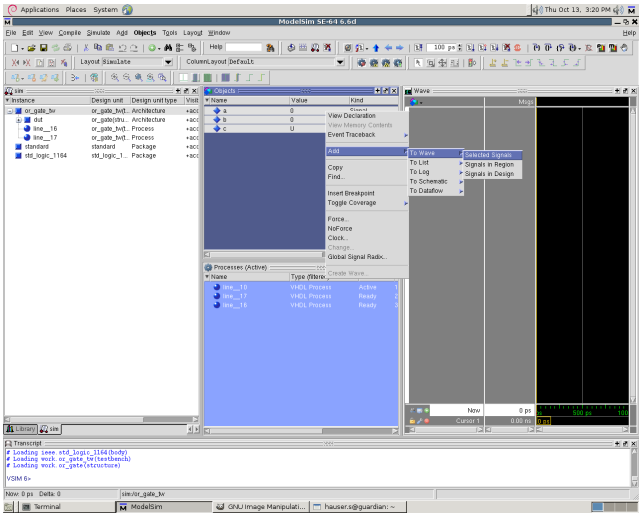

Abbildung 8: Hinzufügen von Signalen zur Waveform

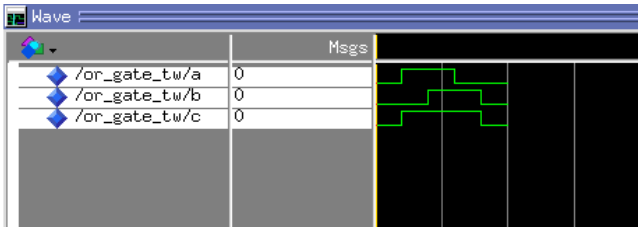

Abbildung 9: Wave-Fenster mit 100ns Simulation

7 Abbildung von Signalen auf FPGA-Pins - Anlegen von Location Constraints

Als nächstes ist es notwendig festzulegen, wie die Ports Ihres Designs den Pins des FPGA zugeordnet werden sollen. Dazu verwenden wir wieder das Werkzeug PlanAhead, öffnen Sie dieses falls Sie es zuvor geschlossen haben. Klicken Sie auf *Add Sources* in der Seitenleiste und wählen Sie diesmal *Add or Create Constraints*. Sie können nun entweder wie vorher die VHDL-Dateien extern eine Datei namens `or_gate.ucf` im Ordner `src` anlegen und diese via *Add Files...* hinzufügen, oder Sie legen die Datei direkt im PlanAhead-Dialog an. Wählen Sie hierzu *Create File...*, geben Sie als Name `or_gate` ein (PlanAhead fügt automatisch `.ucf` an), und wählen Sie als *File Location* den Ordner `src`, wie in Abbildung 10 dargestellt.

Die Datei taucht nun im *Sources*-Fenster auf. Öffnen Sie sie durch Doppelklick.

Wir wollen die Wirkung des Oder-Gatters im FPGA visualisieren und testen. Unsere Experimentierplatinen verfügen beispielsweise über Schiebeschalter und LEDs. Zwei der Schalter werden die Eingänge des Gatters speisen, eine LED den Ausgangswert anzeigen. Die Zuordnung der FPGA-Pins zur Peripherie (inkl. fertig ausformulierter Location Constraints) entnehmen Sie dem *Spartan-3E Starter Kit Board User Guide*[Xilinx(2011)]. Auf Seite 20 finden Sie die UCF Location Constraints der 8 vorhandenen LEDs. Wir werden LED 0 verwenden (ganz rechts unten auf dem Board):

```
NET "LED<0>" LOC = "F12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8;
```

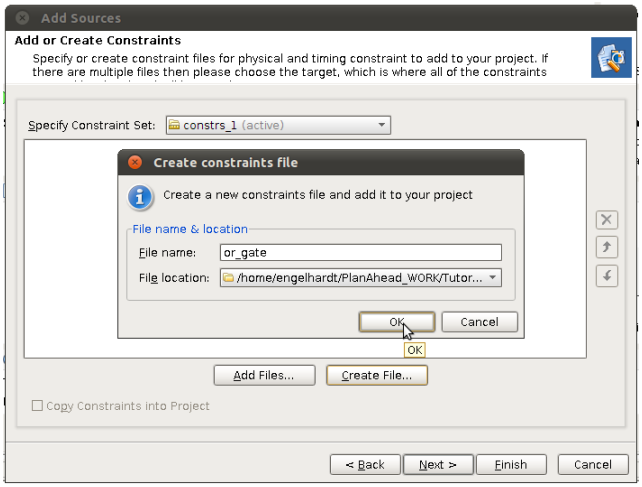

Abbildung 10: Erstellen von Location Constraints

Kopieren Sie diese Constraintzeile vollständig in das UCF und tauschen Sie `LED<0>` gegen unseren tatsächlichen Signalnamen `OUTPUT` aus. Falls ein Port in VHDL mehrere Bit breit ist wird jedem Bit ein eigenes Location Constraint zugeordnet. Der Bezeichner `LED<0>` im obigen Beispielconstraint greift die niederwertigste Stelle aus einem Vektor mit der Bezeichnung LED heraus, gekennzeichnet durch die Klammern < und >.

Im Wesentlichen können Sie dem Constraint die Informationen entnehmen, dass das Signal `OUTPUT` jetzt den FPGA-Pin F12 treibt (an den eine der LEDs angeschlossen ist) und dass der Signalpegel zwischen 0 und 3,3V (entspricht dem Standard „Low Voltage Transistor Transistor Logic“ = LVTTL) liegt. Im Rahmen des Hardwarepraktikums ist keine nähere Beschäftigung mit der Funktionsweise der Constraints notwendig. Location Constraints aller notwendigen Pins können vollständig dem *Starter Kit Board User Guide*[Xilinx(2011)] entnommen werden.

Suchen Sie anschließend die Constraints der Schiebeschalter SW0 und SW1, kopieren Sie sie in das UCF und weisen Sie ihnen die Signale `INPUT_1` und `INPUT_2` zu. Speichern und schließen Sie das Constraintfile.

8 Synthese und Implementation

Nun können Sie Ihr Design auf die Zielhardware abbilden lassen. Dies geschieht in zwei Schritten: Synthese und Implementation. Der erste Schritt, den Sie durch einen Klick auf *Synthesize* in der Seitenleiste anstoßen können, übersetzt Ihre Hardware-Beschreibung in eine *Netlist*: eine graphförmige Beschreibung, bestehend aus LUTs, Puffern und Verbindungen (*Nets*). Nach der Synthese fragt PlanAhead, wie Sie fortfahren möchten: wählen Sie hier *View Reports*. Daraufhin öffnet sich im unteren Bereich ein Fenster namens *Reports*. Doppelklicken Sie auf *XST Report* (unter *Synthesis*) und lesen Sie sich die Ausgaben durch. Achten Sie insbesondere auf Angaben zu abgeleiteten Elementen, z.B. Einträge der Form `inferred 1 D-type flip-flop(s)` im Abschnitt *HDL Synthesis* und versichern Sie sich, dass diese Ihren Vorstellungen entsprechen. Sehen Sie sich auch die Ressourcen an, die Ihr Projekt benötigt (im Abschnitt *Device utilization summary*, oder grafisch im *Project Summary* unter *Resources*).

In einem Zweiten Schritt, *Implement* in der Seitenleiste, werden die Elemente der Netlist auf die Komponenten im FPGA abgebildet. Schauen Sie sich auch hier die Ausgaben an und versichern Sie sich, dass das implementierte Design alle Timing Constraints einhält.

9 Erzeugen von Bitfiles für die Konfiguration des FPGA

Als letztes muss das implementierte Design noch auf den FPGA übertragen werden. Wählen Sie unter *Program and Debug* in der Seitenleiste *Generate Bitstream...* und bestätigen Sie im sich öffnenden Fenster die Vorgaben. Treten jetzt keine Fehler mehr auf, wird im Verzeichnis `<Projekt>/<Projekt>.runs/impl_1` die Datei `or_gate.bit` erzeugt, mit der das FPGA „programmiert“ werden kann.

PlanAhead enthält das Werkzeug *Impact*, mit dem das Bitfile direkt in den FPGA transferiert werden kann, wenn das Entwicklungsboard per USB an den Arbeitsplatzrechner angeschlossen ist. Für SunRay-Arbeitsplätze verwenden wir jedoch die folgende Lösung: Suchen Sie in Ihrem Projektverzeichnis die Datei `or_gate.bit`. Geben Sie der Datei den Namen `<Gruppennummer>_or_gate.bit` um sie von den Dateien der anderen Gruppen abzugrenzen.

Senden Sie die Datei an `hwpr@aes.tu-berlin.de` und melden Sie sich bei einem der Betreuer um die Datei auf den FPGA zu übertragen.

Literatur

[Xilinx(2011)] XILINX: *Spartan-3E Starter Kit Board User Guide*, 2011. URL `http://www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf`.