Entwickeln eines Logik Designs für einen CPLD Baustein in VHDL und Download mit der Xilinx Entwicklungsumgebung

Als erstes ist ein Verzeichnis zu erstellen, in dem die Projektdateien abgelegt werden sollen. ZB. C:\xiprojects\praktikum
Weiterhin sollten sie sich, bevor sie fortfahren, überlegen, welche Ein- und Ausgänge der zu entwerfende Baustein benötigt. Notieren sie sich auf Papier, welche Ein und Ausgänge sie benötigen werden und geben sie diesen bereits sinnvolle Namen! Die Zuordnung zu den Pins des Bausteins kann auch später erfolgen.

Projektvorbereitung:

Starten sie den **Project Navigator** von der Schreibtischoberfläche oder dem Startmenu.
Das Program startet normalerweise mit dem letzten bearbeiteten Projekt. Zum Starten eines neuen Projekt das möglicherweise geöffnete Projekt schließen (Menuleiste -> File -> Close Project) und dann **(-> File -> New Project)**.
Es öffnet sich eine Dialogbox. Es ist der Name des neuen Projekts und der Ort wo dieses mit allen seinen Dateien abgespeichert werden soll, anzugeben. Hierfür den vorher erstellten Ordner im Eingabefeld Project Location anwählen. Für den Typ des Projekt den vorgegeben Typ HDL für das Top Level Modul beibehalten. Bestätigen mit „Weiter“.

Beim nächsten Dialogfeld wird der zu verwendende Bausteintyp angegeben. Für den Laborversuch ist hier die Device Family **XC9500XL** (aus dem Dropdownmenu auszuwählen, das Device **xc9572xl** und das Package **VQ44** mit dem Speed Grade **-5**. (falls nicht bereits eingetragen). An den anderen Einstellungen braucht nichts geändert werden, VHDL ist bereits Standardmäßig vorgegeben ist.

Nach dem Klick auf „Weiter“ ist auf den „New Source“ Button zu klicken und man hat in dem nun erscheinenden Dialog die Wahl für einen bestimmten Quelltexttyp, wählen sie hier **VHDL Module** und geben dem Source Modul einen Namen, zB „test1“. Achten sie darauf, das die Checkbox „Add to Project“ angeklickt ist (sollte Defaultmäßig der Fall sein) und bestätigen sie mit „Weiter“.

Eine weitere Dialogbox erscheint, in der nach den Ein und Ausgangsanschlüssen gefragt wird. Klicken sie der Reihe nach in das Feld „Port Name“ und geben dort die entsprechenden Bezeichnungen ein, im Feld „Direction“ wählen sie aus dem Dropdownmenu ob Input, Output oder InOut definiert werden soll. Bei einem einzelnen Anschluß brauchen sie in MSB und LSB keine Eingabe zu machen, falls es sich um einen Bus handeln soll, definieren sie das MSB (zB 7), LSB wird dann automatisch schon auf 0 gesetzt. Bestätigen sie auch diesen Dialog mit „Weiter“.
Es folgt noch eine Zusammenfassung und mit „Fertig stellen“, „Weiter“, und nochmal „Weiter“ und einem letzten „Fertig stellen“ wird die VHDL Datei „test1.vhd“ bereits weitgehend vom System erstellt und im Editor geöffnet.

Weiteres Vorgehen

Im **Modul View** Fenster links oben (Abb. 1) ist die gerade erstellte Datei ausgewählt, im darunter ausgewählten **Process View** Fenster (Abb.3) sind alle Aktionen, die im Moment für die ausgewählte Datei möglich sind aufgeführt. Ganz unten befindet sich noch ein Fenster, in dem alle Aktionen durch (Fehler)Meldungen begleitet werden. (Abb.2)

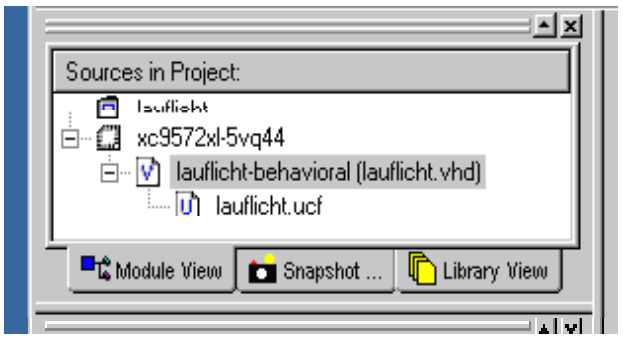

Abb1. : Module View Fenster mit hierarchischer Darstellung der Projektmodule

Abb2.: Message Fenster mit Systemmeldungen und Error Tab

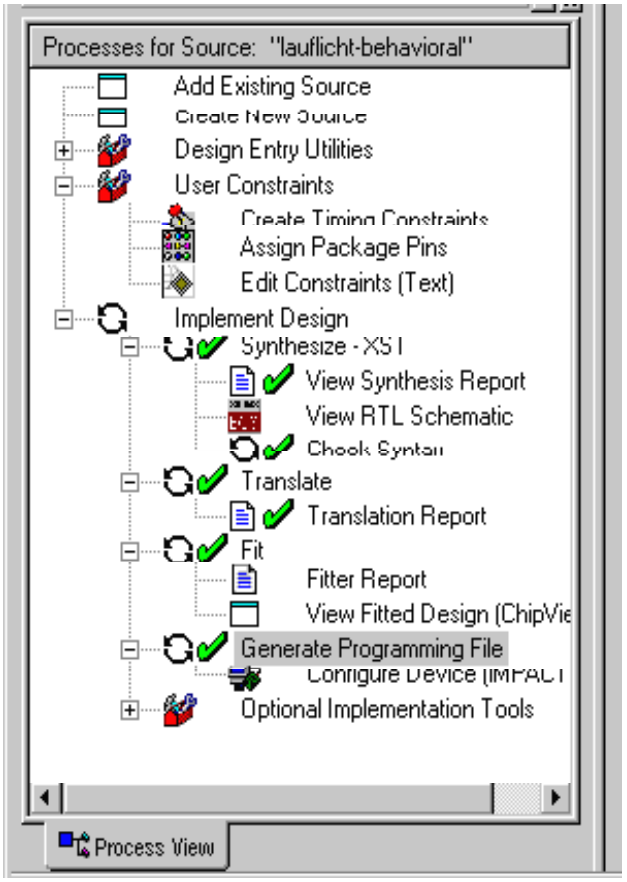

Abb3.: Process View Fenster mit den moglichen Prozessaktivitaten nach fehlerfreier Erzeugung des zum Download notwendigen Jedec Files. Es muss nur noch Configure Device (iMPACT) gestartet werden.

Die prinzipielle weitere Vorgehensweise besteht nun darin, in der VHDL Datei den Block zwischen *architecture Behavioral* .... und *end Behavioral* mit der Beschreibung des gewünschten Hardwareverhaltens in VHDL im Editorfenster zu ergänzen. Siehe VHDL Beispiel
„lauflicht .vhd“ im Anhang.

Im **Process View** Fenster kann jederzeit Zwischendurch durch Doppelklick auf **Check Syntax** im - Implement Design - Synthesize XST Feld eine Syntaxüberprüfung erfolgen, bei Syntaxfehlern werden diese im Message Fenster angezeigt, (Errortab anklicken!) und durch Klick auf die Fehlerzeile im Messagefenster wird im Editorfenster die entsprechende Zeile angewählt und kann dann korrigiert werden. Wenn die Syntax korrekt ist, werden im **Process View** Fenster die erfolgreich durchgeführten Aktionen durch ein grünes Häckchen gekennzeichnet.

Vorbereitung zum Download

Um eine fehlerfrei synthetisierte VHDL Datei (man spricht hier nicht wie bei normalen Programmiersprachen von Compilieren sondern, da ja Hardware beschrieben wird, vom Synthetisieren) zum Download in einen Baustein vorzubereiten ist nun noch eine konkrete Zuordnung der Input und Outputs zu konkreten Pinanschlüssen des Bausteins notwendig. Hierfür muß dem Projekt eine entsprechende Zuordnungsdatei hinzugefügt werden. Dies geschieht am Einfachsten indem im **Process View** Fenster unter - User Constraints der **Assign Package Pins** Prozess angeklickt wird. (Abb. 4)

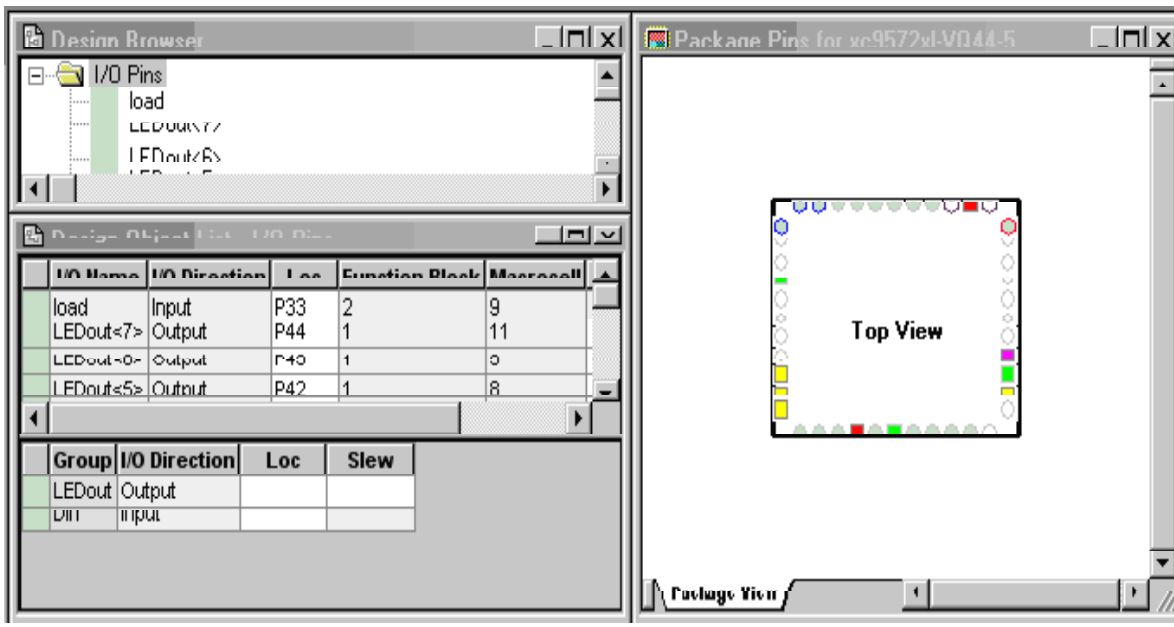

Abb. 4: Assign Package Pins Process unter User Constraints um die Zuordnung der Pinbezeichnungen zu aktuellen Pins des Bausteins zu definieren

Alle notwendigen Prozessschritte werden jetzt nochmal automatisch nacheinander ausgeführt und nur wenn keine Fehler mehr vorhanden sind wird das PACE Programm gestartet, welche die Zuordnungsdatei erstellt (mit .ucf Extension). Hierfür den Dialog beim Start dieses Programms mit „Yes“ bestätigen. Sie sehen jetzt drei Fenster, rechts das Fenster mit dem Blick auf die Anschlüße des gewählten Bausteins und links zwei Fenster mit den im VHDL Entwurf benutzten Pinbezeichnungen.
Sie können jetzt links auf die einzelnen Pinnamen Klicken und per Drag und Drop rechts die Zuordnung zu den gewünschten Pins machen, Pins die nicht verwendet werden können (zB. Spannungsversorgung) können nicht ausgewählt werden. Nach erfolgreicher Zuordnung können sie nach dem Sichern der ucf Datei dieses Hilfsprogramm beenden und sind dann wieder im Projekt Navigator Programm. Im Modul View Fenster ist jetzt diese neue Datei als „test1.ucf“ eingetragen. Nun sind alle Zuordnungen erfolgt, sodas die eigentliche Jedec Datei, die anschließend zum Programmieren in den Baustein geladen werden soll, generiert werden kann.

Dies geschieht (nach Auswahl von test1.vhd im Modul View Fenster) mit dem Process **Generate Programming File** im Process View Fenster. Ein grünes Häckchen sollte die erfolgreiche Generierung von „test1.jed“ bestätigen.

Nun kann als letzter Schritt der Entwurf in den Baustein geladen werden. Hierfür ist der Prozess **Configure Device (iMPACT)** im Process View Fenster zu starten. (Abb. 5) Ein neues Programm startet, welches erst fragt wie der Baustein konfiguriert werden soll (vorgegeben ist Boundary Scan) mit „Weiter“ bestätigen, mit „Fertigstellen“ wird nun automatisch bei mit dem Printerport verbundenen JTAG Kabel des Boards die Bausteinkonfiguration ermittelt. Je nach Jumperstellung auf dem Board werden Beide Bausteine des Boards oder nur der ausgewahlte Baustein angezeigt und abgefragt, für den Baustein XC9572XL wählen sie die gerade erstellte „lauflicht.jed“ Datei aus.

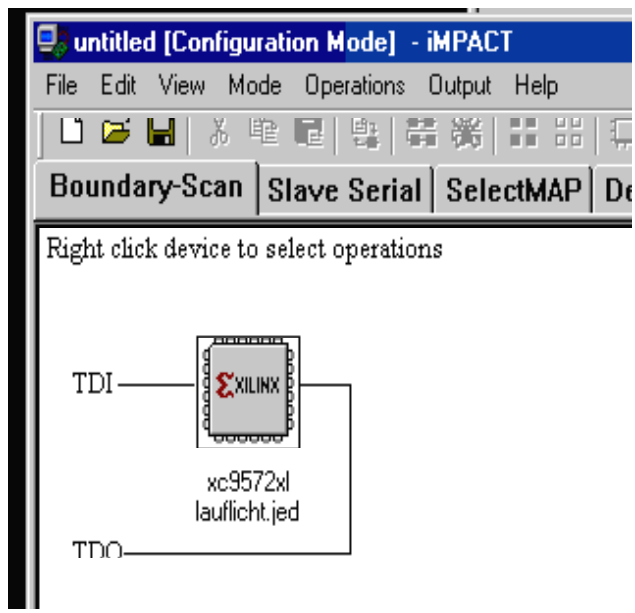

Abb. 5: iMPACT Prozess zum Download des Jedec Files, mit Rechtsklik auf den Baustein wird ein Popup menu geoffnet, in dem die Programmieroption enthalten ist

Mit Rechtsklick auf den Baustein können sie nun mit **Program...** die Jedecdatei in den Baustein downloaden und diesen damit programmieren. Anschließend sollte der Baustein sich dann direkt so verhalten wie gewünscht :-))

Anhang:

VHDL Datei für ein mit einem Schieberegister realisierten Lauflicht

```
-- Lauflicht aus Laborversuch in VHDL  6.4.04 A.Schnell

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity lauflicht is
    Port ( clock : in std_logic;
             Din : in std_logic_vector(7 downto 0);
          LEDout : out std_logic_vector(7 downto 0);
            load : in std_logic);
end lauflicht;

architecture Behavioral of lauflicht is
     signal SR_int: std_logic_vector(7 downto 0);
                                       -- internes Schieberegister
     signal clk_div: unsigned(18 downto 0); --18 stage counter
begin
     process(clk_div(18)) -- Lauflicht Prozess --------------
     begin
       if clk_div(18)'event and clk_div(18) = '1' then
         if (load='1') then
           SR_int <= Din;
                    -- Datenuebernahme mit load Button
         else
           SR_int <= SR_int(6 downto 0) & '0';
                                  -- schiebt null nach
         end if;
       end if;
     end process;

     LEDout <= SR_int;

     process (clock) -- clock prescaler ----------------------
     begin
       if (clock'event) and (clock = '1') then
         clk_div <= clk_div + 1;    -- Counter
       end if;
     end process;
end Behavioral;
```