0. Übung

Fachgebiet Architektur eingebetteter Systeme
Hardwarepraktikum (TI)

AES Embedded Systems Architecture
SoSe 17

Ausgabe: 19.04.2017

Theorie Präsentation: entfällt
Praxis Abgabe: entfällt

Arbeitsziel

Rekapitulieren Sie die Hardwarebeschreibung mit VHDL und erlernen Sie den Umgang mit der Entwicklungsumgebung *Xilinx PlanAhead* sowie dem Evaluation Board anhand einfacher synchroner und asynchroner Beispielkomponenten.

Arbeitsmaterialien

- Projekttutorial
- Spartan-3E Starter Kit User Guide[5]
- PlanAhead User Guide[1]

Aufgabenbeschreibung

Sie erlernen den Umgang mit der Entwicklungsumgebung *PlanAhead* anhand eines vollständig vorgegebenen Beispiels und frischen anschließend ihre VHDL-Kenntnisse durch einfache Beispielaufgaben auf. Eine Bewertung findet nicht statt.

Aufgabe 1 (Tut) Beispiel: ODER-Gatter

Beginnen Sie mit der im Projekttutorial[1] beschriebenen Beispielaufgabe: Anlegen eines neuen Projekts, Erstellen und Einbinden eines VHDL-Moduls, Simulation mittels einer Testbench-Waveform und Abbilden der VHDL-Beschreibung auf den FPGA. Testen Sie die Implementierung auf dem FPGA.

Aufgabe 2 (Tut) Beispiel: Zähler

Erstellen Sie in PlanAhead ein neues Projekt für den Zähler. Legen Sie eine Datei `counter.vhd` im Quellenverzeichnis an und binden Sie sie in das Projekt ein. Eine Beschreibung der Schnittstelle des Zählers finden Sie in Tabelle 1.

`EXT_RST`	`in`	Synchrones, highaktives Resetsignal des Zählers. Für `EXT_RST = '1'` während einer steigenden Flanke an `EXT_CLK` wird der Zähler auf `'0'` gesetzt.
`EXT_CLK`	`in`	Taktsignal des Zählers. Zu jeder steigenden Flanke des Taktsignals wird der Zähler inkrementiert.
`EXT_LED(7:0)`	`out`	Zeigt den aktuellen Wert des Zählers an.

Tabelle 1: Beschreibung der Schnittstelle des Zählers

Benennen Sie die Entity mit `counter` und binden Sie folgende Pakete ein, wie es in Listing 1 dargestellt ist.

[1] Das Projekttutorial finden Sie als Handout auf der ISIS Seite

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

Listing 1: Benötigte Pakete für den Zähler

Ergänzen Sie eine Architecture `fast` und implementieren Sie folgendes Verhalten:

Der Zähler enthält ein synchrones 8-Bit-Register, das mit jeder steigenden Taktflanke von `EXT_CLK` seinen Inhalt inkrementiert. Überläufe müssen nicht abgefangen werden. `EXT_RST` ist ein synchrones Resetsignal, welches das Zählregister wieder auf Null setzt.

Die Realisierung von Registern mit synchronem Resetsignal entnehmen Sie bei Bedarf dem *XST User Guide* [6], Kapitel 3, *HDL Coding Techniques*. Der Port `EXT_LED` entspricht dem aktuellen Inhalt des Zählregisters. Erklären Sie ihre neue Komponente zum Topmodul.

Testen Sie die Funktionsweise ihres Zählers durch eine Test Bench. Erstellen Sie die Testbench über den *Add Sources* Dialog als *Simulation-only source*. Achten Sie darauf, als Zielverzeichnis ihr Quellenverzeichnis anzugeben. Definieren Sie eine `entity` ohne Ports mit dem Namen `counter_tw`. Instanziieren Sie ihren Zähler innerhalb dieser und schließen Sie die Ein- und Ausgangssignale an interne Signale an. Definieren Sie die Input-Signale `EXT_CLK` und `EXT_RST` mithilfe des `after` Konstrukts, wie im Tutorial beschrieben. Legen Sie einen 50Mhz Takt an `EXT_CLK` und setzen Sie `EXT_RST` für zwei Takte auf `'1'` und dann wieder auf `'0'`.

Laden Sie ihre Testbench in ModelSim und simulieren Sie sie für 200 ns. Zeigen Sie dabei die Signale `EXT_CLK`, `EXT_RST` und `EXT_LED` in der Wave an. Überprüfen Sie, dass ihr Design wie gewünscht funktioniert.

Treten keine Fehler auf, so passen Sie das bisherige UCF an ihr neues Modul an: die bisherigen *Location Constraints* werden auskommentiert (ein # am Zeilenbeginn) und das UCF um die neuen *Location Constraints* ergänzt: `EXT_CLK` wird auf den Takteingang abgebildet, an dem das 50Mhz Taktsignal des Entwicklungsboards anliegt. `EXT_RST` wird auf einen der Pushbuttons ihrer Wahl abgebildet. `EXT_LED` wird auf die 8 LEDs des Boards abgebildet.

Implementieren Sie ihr Design und sehen sie sich die Ausgaben von *Open RTL Design* und *Open Netlist Design* an (womit gleichzeitig die Synthesefähigkeit ihres Codes überprüft wird).

Generieren Sie anschließend das neue Bitfile und spielen Sie es in den FPGA ein.

Sie sollten sehen können, dass die 8 LEDs unterschiedlich hell leuchten, die Arbeitsweise des Zählers wird jedoch nicht nachzuvollziehen sein, weil er zu schnell zählt.

Aufgabe 3 (Tut) Beispiel: Zähler 2

Um die Funktionsweise des Zählers auch auf dem Board nachvollziehen zu können muss die Zählfrequenz reduziert werden.

Kopieren sie Ihre Architecture und nennen Sie die neue Architecture `slow`. Kommentieren Sie `fast` vollständig aus.

Realisieren Sie einen langsam zählenden Zähler mit Hilfe des vorgegebenen VHDL-Moduls **ArmWaitStateGenAsync.vhd**. Kopieren Sie dieses in Ihr Quellenverzeichnis und instantiieren Sie es in der Architektur `slow` des Zählers. Die Schnittstelle des vorgegebenen Moduls finden Sie in Listing 2.

```
entity ArmWaitStateGenAsync is
    port(
        SYS_CLK :       in STD_LOGIC;
        SYS_RST :       in STD_LOGIC;
```

```
        WSG_COUNT_INIT: in STD_LOGIC_VECTOR(31 downto 0);
        WSG_START :     in STD_LOGIC;
        WSG_WAIT :      out STD_LOGIC
    );
end ArmWaitStateGenAsync
```

Listing 2: Entity `ArmWaitStateGenAsync`

Dieser Waitstategenerator enthält einen eigenen Zähler. In jedem Takt, in dem `WSG_START = '1'` ist, wird der interne Zähler mit dem an `WSG_COUNT_INIT` anliegenden Wert initialisiert und das Signal `WSG_WAIT` auf `'1'` gesetzt. [2] Sobald `WSG_START` wieder auf `'0'` zurückkehrt, wird der interne Zähler mit jedem Takt dekrementiert. `WSG_WAIT` behält den Wert `'1'`, solange der interne Zähler nicht Null erreicht hat.

Starten Sie nach jedem Inkrementieren des Zählers den Waitstategenerator. Bis das Wartesignal wieder `'0'` ist, soll Ihr Zähler seinen Wert halten.

Simulieren Sie auch diese Version von `counter` in ModelSim. Verwenden Sie als Initialwert des Zählers (`WSG_COUNT_INIT`) den Wert `X"00000100"` in der Simulation und `X"00100000"` für die Synthese.

Synthetisieren sie Ihr Design erneut und spielen Sie es in den FPGA ein.

Literatur

[1] *PlanAhead User Guide*. `http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_3/PlanAhead_UserGuide.pdf`, Juli 2011.

[2] MENTOR GRAPHICS CORPORATION: *ModelSim SE Reference Manual*, 6.6d Auflage.

[3] MENTOR GRAPHICS CORPORATION: *ModelSim SE Tutorial*, 6.6d Auflage.

[4] MENTOR GRAPHICS CORPORATION: *ModelSim SE User's Manual*, 6.6d Auflage.

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

[6] XILINX: *XST User Guide*. `http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_2/xst.pdf`, Dezember 2010.

[2] Bei Initialwerten größer null wird der Wert bereits bei der Übernahme dekrementiert, da es sich aus der Sicht des übergeordneten Moduls bereits um den ersten Wartezyklus handelt.