Praktikum zur Vorlesung

Prozessorarchitektur

SS 2018

Übungsblatt 2

Entwicklung und Test sequentieller Logik

Abgabefrist: Mittwoch 25.04.2017, 10:00 Uhr

1.1. Einführung

In dieser Übung werden Sie ein VHDL-Modul entwerfen, welches taktgesteuerte sequentielle Logik implementiert. Es soll ein 8 Bit Aufwärtszähler in VHDL implementiert, eine Testbench-Datei für diese Simulation geschrieben und das entwickelte Modul auf der Ziel-FPGA-Plattform getestet werden. Der Entwurfsablauf ist der Gleiche wie in der vorangegangenen Übung.

1.2. Projektbeschreibung

Der Zähler ist wie folgt spezifiziert.

- Das Modul hat die folgenden Eingänge:
  data[7:0] 8-Bit Vektor zur Initialisierung der Zählerregister
  load 1-Bit Steuersignal zu Steuerung der Initialisierung
  en 1-Bit Steuersignal, high active, schaltet den Zähler ein
  clk 1-Bit Taktsignal
  rst_n 1-bit asynchrones Reset (low active)

- Das Modul hat den folgende Ausgang:
  q[7:0] 8-Bit Zählerausgangsport

- Das Zählermodul ist ein 8-Bit Aufwärtszähler. Wenn das Signal ***load*** ‚0' ist, wird der Zähler mit dem Wert der Daten ***data*** initialisiert. Das Signal ***load*** muss high sein, um die Zählerfunktion wieder zu aktivieren. Der Ladevorgang ist nicht vom Signal ***en*** abhängig. Das Signal ***en*** arbeitet eher als ein Haltesignal, das heißt, wenn es ‚1' ist zählt der Zähler weiter, bei ‚0' hält der Zähler an. Das Reset-Signal ***rst_n*** initialisiert den Zähler auf den Wert Null.

1.3. Erstellen der VHDL-Quellen des Zählers

1. Beginnen Sie mit der Erstellung eines neuen Projekts mit dem Namen ***lab2***. Verwenden Sie dieselben Projekteinstellungen wie in Übungsblatt 1.

2. Erstellen Sie eine neue VHDL-Quelle mit dem Namen ***counter.vhd*** und fügen Sie diese zum Projekt hinzu. Im "*Create File*" Wizard, geben Sie die Modul-Ports nach der Entwurfsspezifikation ein.

3. Öffnen Sie die generierte VHDL-Datei und binden Sie folgende Bibliotheken mit ein:

```
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
```

4. Bearbeiten Sie ***counter.vhd*** um die Zählerfunktionalität zu beschreiben. Das Zähler-Register soll als ein 8 Bit Signal beschrieben werden. Speichern Sie die Datei und führen Sie die RTL-Analyse durch. Prüfen Sie das generierte RTL-Diagramm. Es sollte ähnlich zu dem in Abbildung 1.1 dargestellt werden.

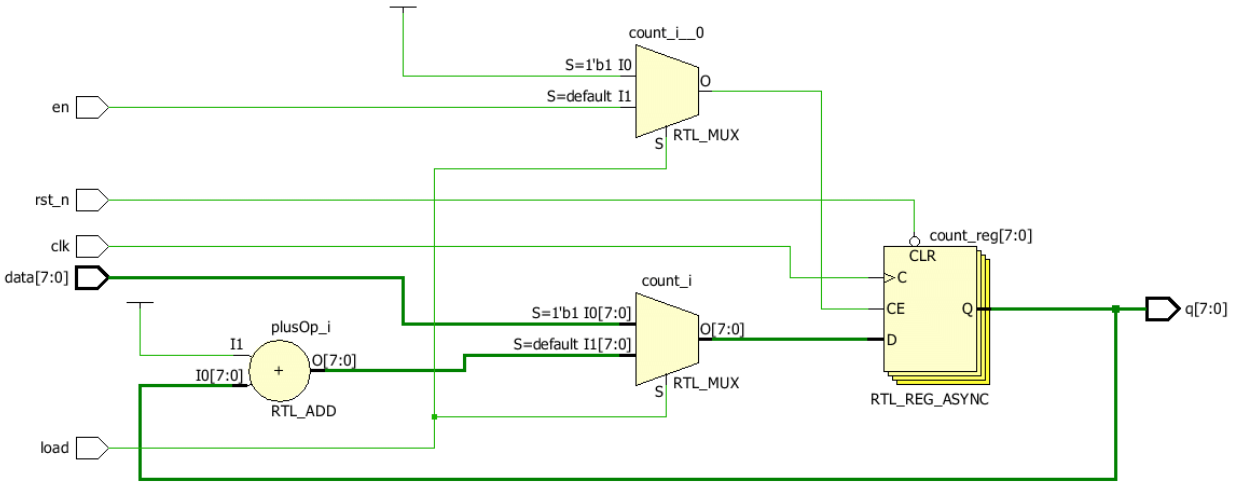

Abbildung 1.1. RTL-Diagramm des Zähler-Moduls.

1.4. Testbench erstellen und Zähler simulieren

Um die Zählerfunktionalität zu überprüfen werden Sie eine VHDL-Testbench erstellen. Diese hat folgende Aufgaben:

- Die Simulation beginnt bei der Zeit 0 us. Das Reset-Signal ist aktiv und der Takt läuft mit 1 MHz. Das Enable-Signal ist inaktiv und der Binärwert des Ladesignals ist “10000000”.
- Nach 10 us ist das Reset-Signal zurückgesetzt und die Simulation läuft weitere 10 us.
- Das Enable-Signal wird gesetzt und die Simulation läuft weitere 10 us.
- Das Enable-Signal wird zurückgesetzt und die Simulation läuft weitere 10 us.
- Das Load-Signal wird für einen Takt gesetzt.
- Das Load-Signal wird zurückgesetzt und die Simulation läuft weitere 10 us.
- Das Enable-Signal wird gesetzt und die Simulation läuft weitere 130 us.
- Das Reset-Signal wird gesetzt und die Simulation läuft weitere 10 us.
- Die Simulation wird beendet.

1. Im ersten Schritt erstellen Sie eine VHDL-Simulations-Vorlage. Klicken Sie auf *Flow Navigator* > *Project Manager* > *Add Sources* und wählen Sie *Add or create simulation sources* aus. Klicken Sie auf *Next*. Klicken Sie auf *Create Data* und wählen Sie *VHDL* als *Data Type*. Benennen Sie die Testbench-Datei mit ***counter_tb.vhd***. Klicken Sie auf *Finish*. Ändern Sie den Namen der Architektur in ***tb*** und klicken Sie auf *Ok*. Öffnen Sie die generierte Datei ***counter_tb.vhd***. Beachten Sie, dass die Portdeklaration im *Architecture*-Teil einer VHDL-Testbench nicht erforderlich ist.

2. Nun sollten Sie die generierte Testbench bearbeiten. Eine VHDL-Testbench enthält in der Regel die Komponentendeklaration einschließlich der Angabe des zu testenden Designs (DUT), die Deklaration der Signale und Konstanten, die Taktdeklaration, einen Stimuli-Prozess und die Instanziierung der Komponenten. Bei der Bearbeitung der Testbench sollten Sie die folgenden Richtlinien beachten:

    - Fügen Sie Komponentendeklaration in den Architekturteil der Testbench ein. Sie können einfach den *Entity*-Teil von ***counter.vhd*** kopieren und in die Testbench einfügen, wobei Sie das reservierte Wort *entity* durch das reservierte Wort *component* ersetzen.

    - Deklarieren Sie die Signale die mit den DUT-Anschlüssen verbunden sind. Sie können die gleichen Namen wie für die Portdeklaration der Komponente nutzen. Initialisieren Sie alle Signale die mit den Eingangsports der DUT Komponenten verbunden sind mit den folgenden Werten: ***data = “01000001“***, ***load = ‘0‘***, ***en = ‘0‘***, ***clk = ‘1‘***, ***rst = ‘0‘***.

    - Definieren Sie eine Konstante mit dem Namen ***period*** vom Typ ***time***, wobei die Konstante einen Wert von 1 us hat.

- Fügen Sie die Taktdefinition nach dem reservierten Wort *begin* in die Testbench ein:

```
clk <= not clk after period/2;
```

- Im nächsten Schritt wird das Stimuli-Verfahren gemäß der beschriebenen Testablaufspezifikation definiert. Sie fangen mit der Festlegung der Anfangswerte für Eingangssignale zum Zeitpunkt 0 an, gefolgt von den Signalzuweisungen wie sie in der Spezifikation oben beschrieben sind. Der Beginn des Stimuli-Prozesses ist vorgegeben:

```
stim_proc: process
begin
  data    <= "01000001";
  load    <= '0';
  en      <= '0';
  rst_n   <= '0';
  wait for 10*period;
  rst_n   <= '1';
  wait for 10*period;
  …
```

Bevor der Stimuli-Prozess beendet wird, sollten Sie die folgenden Zeilen in den Prozess-Body einfügen um die Simulation zu beenden:

```
…
assert false
severity failure;
```

- Schließlich müssen die DUT-Komponenten instanziiert werden. Verwenden sie ***dut*** als Instanzname. Speichern Sie die Datei und suchen Sie nach möglichen Syntaxfehlern.

- Die Simulation kann jetzt gestartet werden. Ändern Sie die Simulationslaufzeit im Feld *Simulation Settings* auf 0 ns und klicken Sie auf *Run behavioral simulation*. Docken Sie das Waveform-Fenster ab. Jetzt drücken sie auf *F3*, um die vollständige Simulation laufen zu lassen. Sie können die Waveforms vergrößern und verkleinern, um die Werte bestimmter Signale während der Simulationszeit zu beobachten.

- Starten Sie jetzt die Simulation durch Drücken von *Shift + Ctrl + F5* neu. Klicken Sie auf der Registerkarte die Datei ***counter_tb.vhd*** an, um den Code der Testbench zu sehen. Sie werden rote Kreise neben den Codezeilen in dem Architekturabschnitt der Testbench feststellen. Durch einen Klick auf einen roten Kreis können Sie einen *Breakpoint* in der Simulation setzten. Wir möchten in dieser Übung die Breakpoints nach jeder Änderung der Stimuli-Signale aktivieren. Klicken Sie dazu auf jeden roten Kreis neben einer Codezeile die mit *wait for* beginnt. In dem Waveform-Fenster klicken Sie mit der rechten Maustaste auf das Signal ***q*** und wählen *Radix* >

Hexadecimal. Drücken Sie anschließend die *F3-Taste*. Die Simulation wird dadurch bis zum ersten festgelegten Breakpoint in der Testbench ausgeführt. Drücken Sie kontinuierlich auf die F3-Taste und beobachten Sie die Änderungen der Signale im Waveform-Fenster. Prüfen Sie ob sich Ihr Modul korrekt nach Spezifikation verhält.

1.5. Testen auf dem FPGA

Jetzt sind Sie bereit das Modul auf dem Nexsys-4 FPGA-Board zu testen. Überwachen Sie die Ausgabe des Zählers über die vorhandenen LCD-Displays (Siebensegmentanzeigen) auf dem Board. Dafür benötigen Sie einen *Display Controller* der die Ausgabe des Zählers codiert und die Steuersignale für die LCD-Displays liefert. Darüber hinaus benötigen Sie einen Taktteiler, der den On-Board-generierten 100 MHz Takt verlangsamt. Mit geringerer Taktfrequenz sind Sie in der Lage, die Signaländerungen auf den Displays wahrzunehmen. Die VHDL-Quellen des Display-Controllers und des Taktteilers werden Ihnen zur Verfügung gestellt. Ihre Aufgabe ist es, ein Top-Modul in VHDL zu erstellen, in dem das entwickelte Zählermodul mit dem mitgelieferten Display-Controller und Taktteiler verbunden wird.

1. Fügen Sie die VHDL Quellen ***display.vhd*** und ***clk_divider.vhd*** zum aktuellen Projekt hinzu. Stellen Sie sicher dass die Option *Copy sources into projects* ausgewählt ist. Öffnen Sie die eingefügten Quelldateien und analysieren sie den Code. Die Datei ***display.vhd*** bietet Funktionen für die Codierung der Siebensegmentanzeigen auf dem Nexsys4-Board. Da nur eine einzige 7-Segmentanzeige zu jedem Zeitpunkt aktiv sein kann, verwendet der Anzeigecontroller einen Multiplexer und einen Zähler, der ein schnelles Umschalten zwischen den verschiedenen 7-Segmentanzeigen ermöglicht. Das schnelle Umschalten ist vom menschlichen Auge nicht wahrnehmbar und erzeugt die Illusion, dass alle ausgewählten 7-Segmentanzeigen gleichzeitig aktiv sind. Das 3-Bit Eingangssignal ***disp_sel*** steuert die Anzahl der aktiven 7-Segmentanzeigen. Zum Beispiel aktiviert ***disp_sel = "000"*** nur die 7-Segmentanzeige AN7, hingegen ***disp_sel = "001"*** AN7 und AN6 aktiviert. Wählen Sie ***disp_sel = "111"***, werden alle 7-Segmentanzeigen von AN7 bis AN0 aktiv. Das Signal ***disp_en*** aktiviert den Display-Controller. Wenn dieser 0 ist, sind die Displays aus. Die 8-Bit-Eingangsvektoren ***reg0*** bis ***reg3*** sind den Displays in folgender Weise zugeordnet:

   reg3(7 downto 4) -> AN7
   reg3(3 downto 0) -> AN6
   reg2(7 downto 4) -> AN5
   reg2(3 downto 0) -> AN4
   reg1(7 downto 4) -> AN3
   reg1(3 downto 0) -> AN2
   reg0(7 downto 4) -> AN1
   reg0(3 downto 0) -> AN0

Der Taktteiler (***clk_divider.vhd***) reduziert die FPGA Taktfrequenz von 100 MHz auf einige Herz, wodurch eine Überwachung und Wahrnehmung der Zählerausgaben erst ermöglicht wird.

2. Erstellen Sie ein VHDL Top-Modul namens ***fpga_counter.vhd***. Verwenden Sie die gleichen Port- und Signalnamen wie in der Abbildung 1.2. Wenn die Dateibearbeitung abgeschlossen ist, klicken Sie mit der rechten Maustaste auf den oberen Modulnamen im Fenster *Sources* und wählen Sie *Set as Top* aus. Starten Sie die *RTL Analyse* und prüfen Sie das Top-Design auf Fehler. Sobald das Design erfolgreich ausgearbeitet wurde, vergleichen Sie das erstellte Block-Diagramm mit dem in Abbildung 1.2.

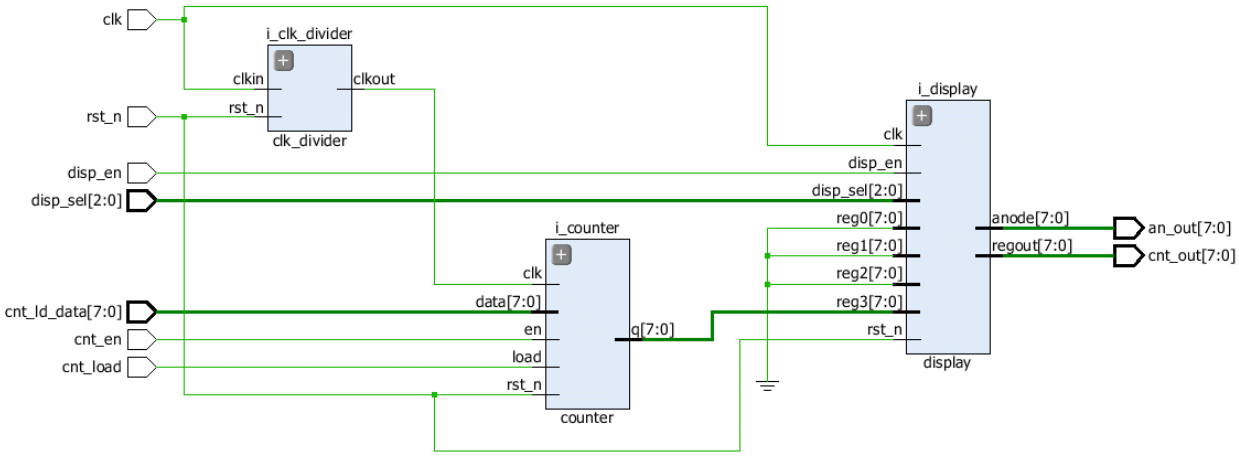

Abbildung 1.2. Block-Diagramm Top-Modul.

3. Erstellen Sie die Port-Mapping-Datei ***fpga_counter.xdc*** um die Top-Modul-Ports mit den FPGA-Pins wie folgt zu verbinden:
   - ***cnt_ld_data[7:0]*** ist verbunden mit den Schaltern SW7-SW0.
   - ***cnt_load*** ist verbunden mit Schalter SW8.
   - ***cnt_en*** ist verbunden mit Schalter SW9.
   - ***clk*** ist verbunden mit Pin E3.
   - ***rst_n*** ist verbunden mit Pin C12.
   - ***disp_en*** ist verbunden mit Schalter SW15.
   - ***disp_sel[2:0]*** ist verbunden mit den Schaltern SW14-SW12.
   - ***an_out[7:0]*** ist verbunden mit den Displayanoden AN7-AN0.
   - ***cnt_out[7:0]*** ist verbunden mit den Display-nodes DP-CA.

4. Beachten Sie bei der Erstellung der XDC-Datei die Hinweise im Nexsys4 FPGA-Board Referenzhandbuch auf Seite 18. Fügen Sie der XDC-Datei folgende Taktdefinition hinzu:

```
# connect clock to pin E3
set_property PACKAGE_PIN E3 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
create_clock -add -name clk -period 10.00 -waveform {0 5}
[get_ports clk]
```

5. Führen Sie die Schritte *Synthesis*, *Implementation* und *Bit-Stream Generation* durch und programmieren Sie den FPGA.

6. Testen Sie nun die Funktion des Zählermoduls auf dem FPGA-Board. Nutzen Sie dazu die entsprechenden Schalter.

   - Verwenden Sie ***disp_en***, um die Anzeige zu aktivieren. Standardmäßig wird nur eine Anzeige an sein. Setzen Sie ***display_sel***, um die zweite Anzeige zu aktivieren. Was ist der angezeigte Zählerwert?
   - Setzten Sie ***cnt_en,*** um den Zähler zu aktivieren. Zählt der Zähler richtig? Setzten Sie ***cnt_en*** zurück, um den Zähler anzuhalten.
   - Laden Sie den Zähler mit dem Hex-Wert x"F0". Setzen sie ***cnt_en***, um den Zähler weiterzählen zu lassen.
   - Halten Sie den Zähler an und setzen Sie *Reset*. Starten Sie den Zähler. Versuchen Sie es mit verschiedenen Steuersignalwerten und überprüfen Sie die Funktionalität laut Spezifikation.