Übung 5 Clock Instantiation

Jan Grapengeter

May 1, 2019

1 Einlesen eines Frequenzsignals

Um einen endlichen Automaten richtig aufbauen zu können, ist ein Taktsignal vonnöten. Bei einer positiven Taktflanke werden dann die Signalzustände überprüft und dementsprechende Ausgabewerte generiert. Auf dem Basys3-board befindet sich ein 100MHz Oszillator. Dieser soll nun benutzt werden, um einen endlichen Automaten aufzubauen. Die Syntax, um eine Sequenz bei nach einer positiven Taktflanke zu triggern, ist:
if(rising_edge(uhr)) then
SEQUENZ
end if;

Aufgabe:
Implementieren Sie den Zustandsautomaten aus der letzten Übung erneut, fragen Sie die zum Umschalten notwendigen Zustände bei steigenden Taktflanken der Uhr ab. Erweitern Sie Ihre Constraints Datei entsprechend.

2 Frequenzteiler

Ein std_logic_vector kann mithilfe einiger Bibliotheken zu einem Frequenzteiler umfunktioniert werden. Legen Sie ein neues Projekt an und binden Sie folgende Bibliotheken in Ihr Projekt ein:
use ieee.std_logic_unsigned.all;
use IEEE.numeric_std.all;
Diese Bibliotheken erlauben Ihnen mathematische Operationen in Ihrem Code zu benutzen. Ein Frequenzteiler lässt sich dann wie folgt realisieren:
if(rising_edge(UHR)) then
TEILER<=TEILER+1;
end if;
Wobei TEILER vom Typ ein std_logic_vector ist.

Aufgaben:
1.: Implementieren Sie einen Frequenzteiler, sodass dieser das Frequenzsignal Ihrer Uhr auf etwas 1Hz herunterteilt. Legen Sie das heruntergeteilte Signal auf einen LED-Pin. Was ist die exakte Frequenz Ihres heruntergeteilten Signals?
2.: Ändern Sie ihren Frequenzteiler nun so, dass dessen Ausgangsfrequenz exakt 1Hz ist. Lassen Sie mit dem Signal wiederum eine LED blinken.
3.: Nutzen Sie Ihr Wissen um Pulsweitenmodulation, um die Helligkeit einer LED zu regulieren.

3 Synchrones und Asynchrones Design

Register/D-FFs lassen sich entweder synchron oder asynchron zurücksetzen. Die Syntax um die Register synchron zurückzusetzen, ist:

```
if(rising_edge(uhr)) then
if(reset='1') then
SEQUENZ_RESET
else
SEQUENZ
end if;
end if;
```

Die Syntax, um die Register asynchron zurückzusetzen, ist:

```
if(reset='1') then
SEQUENZ_RESET
elsif(rising_edge(uhr)) then
SEQUENZ
end if;
```

Xilinx empfiehlt für alle von Ihnen produzierten FPGAs synchrone Resets.[**?**]

Aufgaben:
1.: Überlegen Sie sich, welche Vor- und Nachteile synchrone gegenüber asynchronen Resets haben. Erläutern Sie dabei insbesondere die Auswirkungen auf die Metastabilität der Schaltung.
2.: Implementiere Sie eine Schaltung mit einem synchronen und einem asynchronen Reset. Sehen Sie sich die Schaltung in "Elaborated Design" und "Synthesis/Schematic" an. Was fällt Ihnen auf?
3.: Implementieren Sie die unten stehende State machine in VHDL mit einem synchronen Reset.

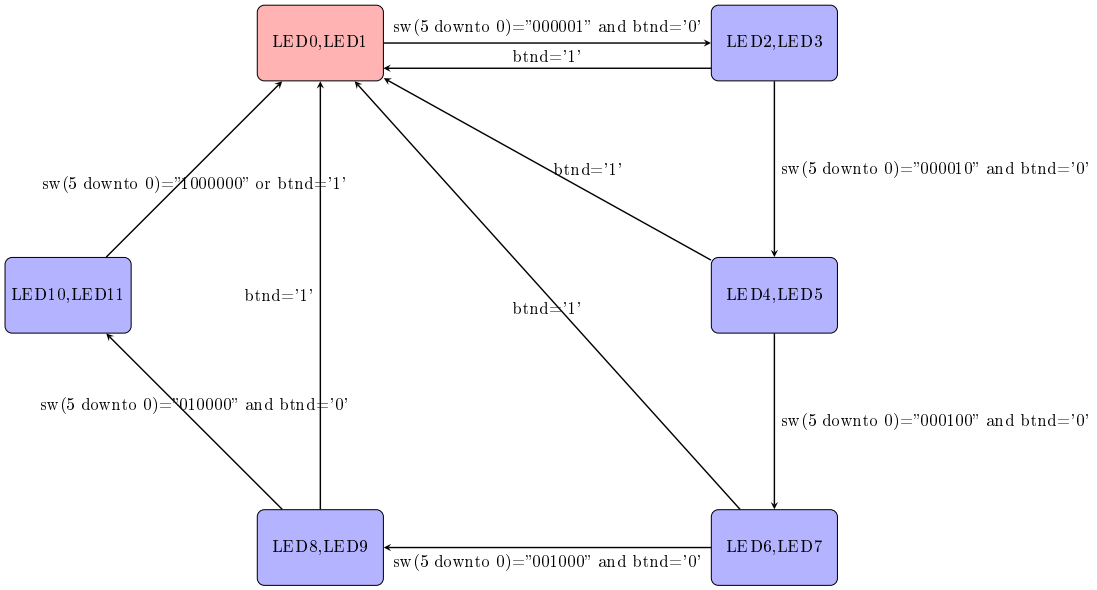

Figure 1: Endlicher Automat

4.: Implementieren Sie eine Stoppuhr mit Reset in VHDL. Stellen Sie Ihre Zeitauflösung auf 0.1s ein. Wie hoch ist die maximale Zeitauflösung, die Sie in Ihrem FPGA erreichen können? Was ist die maximale Zeitspanne, die Sie so messen können?

4 Gatterlaufzeiten

Bei zeitkritischen Designs sind die Verzögerungszeiten in Gattern und Leitungen zu beachten. Typische Verzögerungszeiten lassen sich dem Datenblatt des FPGA entnehmen. Man kann diese aber auch selbst messen.

Aufgabe:
Generieren Sie einen Ringoszillator und einen Frequenzteiler geeigneter Größe, um eine LED so umzuschalten, dass der Schaltvorgang mit dem Auge sichtbar ist. Messen Sie grob die Zeit, die ein Schaltvorgang dauert. Berechnen Sie daraus die Frequenz Ihres Ringoszillators und überlegen Sie sich, wie sich daraus die Gatter und Signal Laufzeiten abschätzen lassen.

5 Vivado Clock Wizard

Legen Sie ein neues Projekt an und wählen Sie unter "Project Manager" den "IP Catalog". Hier finden Sie vorgefertigte Module(Komponenten), die Sie direkt in Ihr Projekt einbinden können.
-Wählen Sie "FPGA Features and Design"
-Wählen Sie "Clocking Wizard"
-Im nun geöffneten Fenster wählen Sie als Optionen PLL, Frequency Synthesis, Phase Alignement, Balanced, Input Frequency =100(MHz)
-Unter "Output Clocks" wählen Sie clk_out1,Output Freq=320(MHz),reset,locked,Active High,Automatic Control On-Chip
-OK
-Out of Context for IP–>"Generate"
-Ok

-Öffnen Sie den Project Manager und wählen Sie Ihr neu erstelltes Modul "clk_wiz_0" und wechseln Sie zu "clk_wiz_0.v"
-Unter "module clk_wiz_0" finden Sie die Port map des Moduls, damit können Sie das Modul genau wie eine Komponente in Ihrem Hauptprogramm einbinden.

Aufgabe:
Instanzieren Sie die soeben angelegte Phasenregelschleife in Ihrem Hauptprogramm. Generierieren Sie dadurch eine schnelle Uhr und testen Sie, ob die Ausgangsfrequenz des Moduls tatsächlich 320MHz ist.