Mobiles Hardware-Praktikum 2023

Marc Pfeifer, Dr. Philipp Scholl, Professur für Rechnerarchitektur,
Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Freiburg im Breisgau

Übungsblatt 5 (26 Punkte)

Wiederholung Quartus

Wenn Sie dieses Übungsblatt beginnen, sollten Sie das Quartus Tutorial und bestenfalls das VHDL Tutorial bereits absolviert haben. Laden Sie dazu "QuartusTutorial.pdf" und "VHDLTutorial.pdf" unter "Tutorials" aus ILIAS herunter. Folgen Sie den Tutorials Schritt für Schritt. Außerdem sollten Sie die (erste) Vorlesungseinheit zu VHDL bereits gehört haben.
Die Quartus Prime Software bietet eine sehr große Anzahl an Einstellungen und Funktionen. Daher lässt sich eine Einarbeitung und etwas trial-and-error nicht vermeiden. Achten Sie insbesondere bei Simulationen darauf die Optionen **exakt** wie im Quartus Tutorial beschrieben zu setzen.

Ziel:

Auf diesem Übungsblatt werden Zähler implementiert. Diese erlauben es zum Beispiel Clock-Zyklen zu zählen und dadurch eine Zeitmessung vorzunehmen (vgl. Übungsblatt 3 - Timer - bei Arduino). Dazu sollten Sie bereits wissen . . .

- . . . wie Sie durch die Teilung einer Clock ein Signal mit einer niedrigeren Taktrate generieren (Übungsblatt 3).
- . . . wie ein VHDL Modul aufgebaut ist (*entity* + *architecture*) und wie die Ports eines entsprechenden Moduls zu definieren sind (Vorlesung + VHDL Tutorial).
- . . . wie Sie eine Testbench erzeugen und dort entsprechend die Input-Stimuli setzen, sowie eine System-Clock simulieren (Vorlesung + Quartus Tutorial).

Teil 1: VHDL Zähler

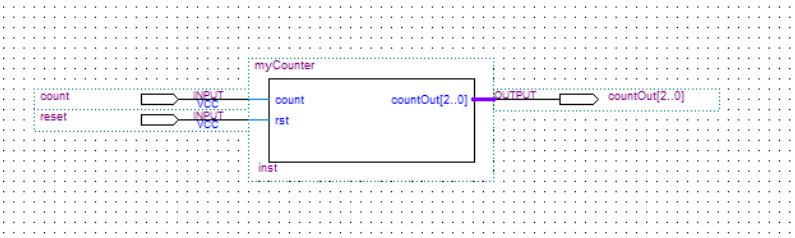


Abbildung 1: Block Diagramm "counter.bdf".

Öffnen Sie das mitgegebene Quartus Archiv "Aufgabe1CounterTemplate.qar". Darin enthalten (siehe Files Tab) ist das in Abbildung 1 gezeigte Block-Diagramm "Counter.bdf".

Aufgabe 1 (4 Punkte):

Öffnen Sie das Modul “myCounter” und setzen sie es als Top-Level Entity. Implementieren Sie hier die Funktionalität eines 6-Bit-Zählers: Über das Eingangssignal “count” wird der Wert des Zählers um 1 inkrementiert. Bei einem Überlauf wird der Wert des Zählers wieder auf 0 gesetzt. Das Eingangssignal “rst” soll den Zähler auf 0 setzen, wenn dieses Signal den Wert ‘0’ hat (active low). Das Ausgangssignal “countOut” soll den Wert des Zählers als 6-Bit Vektor ausgeben.
Simulieren Sie Ihren 6-Bit-Zähler mit einer Testbench. Zeigen Sie hierbei explizit das Hochzählen, sowie das Zurücksetzen des Zählers über reset-Signal sowie den Überlauf. Fügen Sie Ihrer Abgabe ein Bild der Simulation bei, welches alle Fälle zeigt.

Aufgabe 2 (2 Punkt):

Verbinden Sie nun die Taster und die LEDs auf dem Roboter mit dem DE0-Nano wie in Tabelle 1 gelistet:

Roboter	DE0-Nano
Buttons	A0
LED 1	B16
LED 2	A14
LED 3	C16
LED 4	C14
LED 5	D16
LED 6	C15

Tabelle 1: Benötigte Pin-Verbindungen zwischen dem DE0-Nano und den LEDs bzw. der Button Buchse.

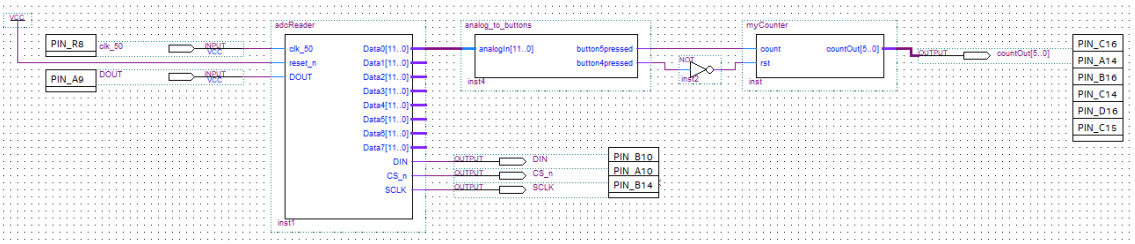


Abbildung 2: Block Diagramm “counterWithOutput.bdf”

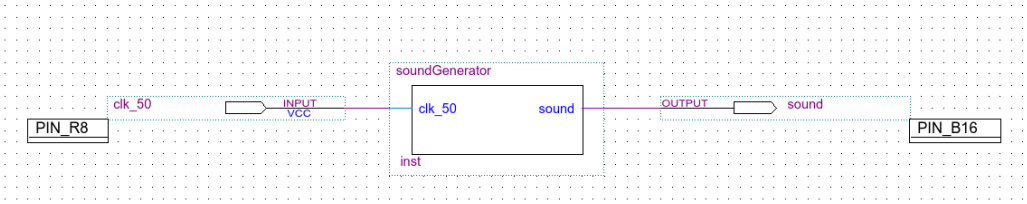
Öffnen Sie nun das in Abbildung 2 gezeigte Block-Diagramm “CounterWithOutput.bdf” und setzen Sie es als *Top-Level Entity*. Implementieren Sie das Modul “analog_to_buttons” so, dass die Ausgänge (*button5Pressed* und *button4Pressed*) ‘1’ sind, wenn die entsprechenden Buttons (*S5* und *S4*) gedrückt sind. Sie können auch das von Ihnen entwickelte Modul von Übungsblatt 4 verwenden. Das in Aufgabe 1 von ihnen vervollständigte Modul “myCounter” ist bereits in das Block-Diagramm eingebunden.
Nachdem Sie das synthetisierte Projekt auf den DE0-Nano übertragen haben, können Sie mit dem Button **S5** den Zähler inkrementieren, und ihn mit **S4** zurücksetzen.

Hinweise:

- Gehen Sie am besten rein kombinatorisch vor. Sie haben die Button-Erkennung bereits für einen 10 Bit ADC in Arduino gelöst (Übungsblatt 1) und können die Ergebnisse daraus entsprechend auf einen 12 Bit ADC in VHDL übertragen.
- Es kann bei ihrem Roboter sein, dass ein Tastendruck nicht direkt oder mehrfach erkannt wird. Je nach Alter und Umgang mit dem Roboter sind die Knöpfe unterschiedlich stark beansprucht worden. Solange die Simulation (mit perfekten Knöpfen) ein funktionierendes Verhalten zeigt, gilt die Aufgabe als abgeschlossen. Die Implementierung einer Entprellung ist nicht vorgesehen.

Teil 2: Sound

Erstellen Sie ein neues Quartus Projekt wie im Quartus Tutorial beschrieben. Erstellen Sie eine neue VHDL-Datei “soundGenerator” mit einem Eingang “clk_50” und einem Ausgang “sound”. Erstellen Sie ein Symbol-File für “soundGenerator” und fügen Sie dieses zu einem neuen Block-Diagramm hinzu. Verbinden Sie die Ein- und Ausgänge wie in der Abbildung:



Verbinden Sie außerdem den Piepser auf dem Roboter mit FPGA Pin B16.

Aufgabe 3 (6 Punkte):

Implementieren Sie das Modul “soundGenerator”. Dieses soll am Ausgang “sound” ein Signal mit einer Frequenz von 400 Hz ausgegeben. Der Input-Pin “clk_50” ist mit einer externen 50 MHz Clock verbunden, welche vom DE0-Nano zur Verfügung gestellt wird (PIN_R8).

Verwenden Sie einen Zähler der bei jeder steigenden Flanke von “clk_50” inkrementiert wird. Erreicht der Zähler einen bestimmten Wert soll der Ausgang von ‘1’ zu ‘0’ (oder umgekehrt) gewechselt werden.

Aufgabe 4 (6 Punkte):

Erweitern Sie Aufgabe 3 so, dass das Drücken der Buttons die Tonfrequenz ändert. Es soll mit Hilfe von Buttons *S1-S5* zwischen fünf verschiedenen Frequenzen gewechselt werden können. Wird ein Button gedrückt soll zu einer der fünf Frequenzen gewechselt werden. Der Ton soll dauerhauft abgespielt werden (auch wenn nichts gedürckt wird). Sie können die Frequenzen frei wählen, sie sollten jedoch alle im hörbaren Bereich liegen (f_x zwischen $200 \leqslant f_x \leqslant 4000$ Hz).
Sie können dazu den “adcReader” und “analog_to_buttons” von Aufgaben Teil 1 verwenden.

Teil 3: Stoppuhr

Aufgabe 5 (8 Punkte):

Erstellen Sie ein neues Quartus Projekt. Implementieren Sie in diesem eine Stoppuhr mit einem 6-Bit Zähler. Verwenden Sie “clk_50”, um diesen jede Sekunde zu inkrementieren.
Die Stoppuhr soll mit zwei Inputs “rst” (reset, setzt den Zähler auf 0) und “startPause” (startet und pausiert das Zählen) gesteuert werden. Diese sollen wieder mit den Buttons **S5** und **S4** verbunden werden. Verbinden Sie den Zähler Ausgang so wie in Tabelle 1 mit den 6 LEDs auf dem Roboter.
Wie Sie die Stoppuhr implementieren steht Ihnen frei. Sie können zum Beispiel Ihre Ergebnisse von Teil 1 und Teil 2 kombinieren. Stellen Sie sicher, dass die Stoppuhr nach Reset bei 0s beginnt und bei einer Pause bei zB 1.9s soll die Stoppuhr bei 1.9s weiterläuft. Beachten Sie das jeder Tastendruck die Stoppuhr entweder startet oder stoppt und bei gedrücktem Taster keine weitere Aktion erfolgt (Tipp: Edge Detektion). Eventuell müssen die Taster dazu entprellt werden, zum Beispiel mit einem weiteren Counter.

Abgabe

Archivieren Sie Ihre Implementierung jeweils von Teil 1-3. Fügen Sie alle drei Quartus Archiv-Dateien zu einer ZIP-Datei hinzu. Laden Sie diese im Übungsportal hoch. Überprüfen Sie die Archiv-Datei auf Vollständigkeit.