OST-Ostschweizer Fachhochschule

Praktikum Digital Design

P4: Hierarchisches Design: motorproject_top

Prof. Dr. Paul Zbinden / Roman Willi

P4: HIERARCHISCHES DESIGN: MOTORPROJECT_TOP

INHALT

EINLEITUNG

In diesem Praktikum widmen wir uns dem Projekt „Motorproject“. Sie sollen in diesem Praktikum verstehen wie man ein hierarchisches Design mit VHDL beschreibt. Dabei ist wichtig, dass sie verstehen, was eine Komponente, eine Instanz und eine „Port Map“ ist. In jeder Beschreibungssprache ist die Hierarchie ein zentrales Element. Wenn wir Schemata zeichnen, dann bauen wir diese meist hierarchisch auf. Ohne Hierarchie würde die Komplexität zu gross werden und unser Verständnis überfordern. Nun hat die Beschreibung einer Schaltung in Form eines Schemas sehr viele Gemeinsamkeiten mit der Beschreibung einer Schaltung in VHDL. Der Unterschied besteht lediglich darin, dass wir das Schema textuell mit einer Sprache beschreiben - so wie beispielsweise eine Internetseite auch mit HTML beschrieben ist. Bei der Internetseite, nimmt der Browser das HTML Textfile und stellt dies gemäss den HTML Regeln dar. Bei VHDL ist es dasselbe. Das CAD Werkzeug (in unserem Fall die Xilinx Vivado Design Suite) nimmt das VHDL Textfile und generiert ein Schema.

Hierarchie bedeutet die Unterteilung der Gesamtschaltung in einfachere und kleinere (Teil-)Komponenten. Diese sind entsprechend der Struktur der Schaltung ineinander verschachtelt. Die Komplexität dieser Komponenten kann von einem einzelnen Gatter (z.B. ein NAND) bis hin zu komplexen Funktionseinheiten (z.B. Prozessorkern, Filter) reichen. Eine Hierarchie stellt selber keine eigene Funktion innerhalb der Schaltung dar, sondern kann als eine Hülle angesehen werden, in der sich ein Schaltungsteil befindet. Die Hülle hat einen Namen und verschiedene Anschlüsse zu der innen liegenden Schaltung, deren Komplexität außerhalb verborgen bleibt und so quasi eine vereinfachte Betrachtung erst ermöglicht. Eine hierarchische Struktur einer Schaltung kann man sich sehr leicht mit Koffern vorstellen, die ineinander gelegt sind und in denen sich jeweils ein Teil des Gesamt-Designs befindet. Die Gesamtschaltung kann man dann mit dem obersten Koffer anfassen, da alle anderen sich in diesem befinden.

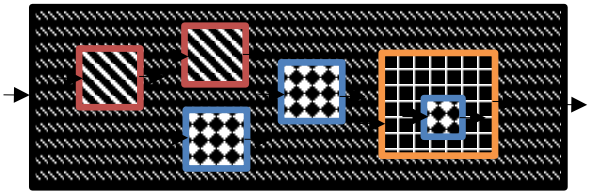

Abbildung 1: Schematische Darstellung eines hierarchischen Designs

AUFGABENSTELLUNG

Gegeben ist das Schema unseres Gesamtprojektes auf höchster Hierarchiestufe, das Schema ***motorproject_top***:

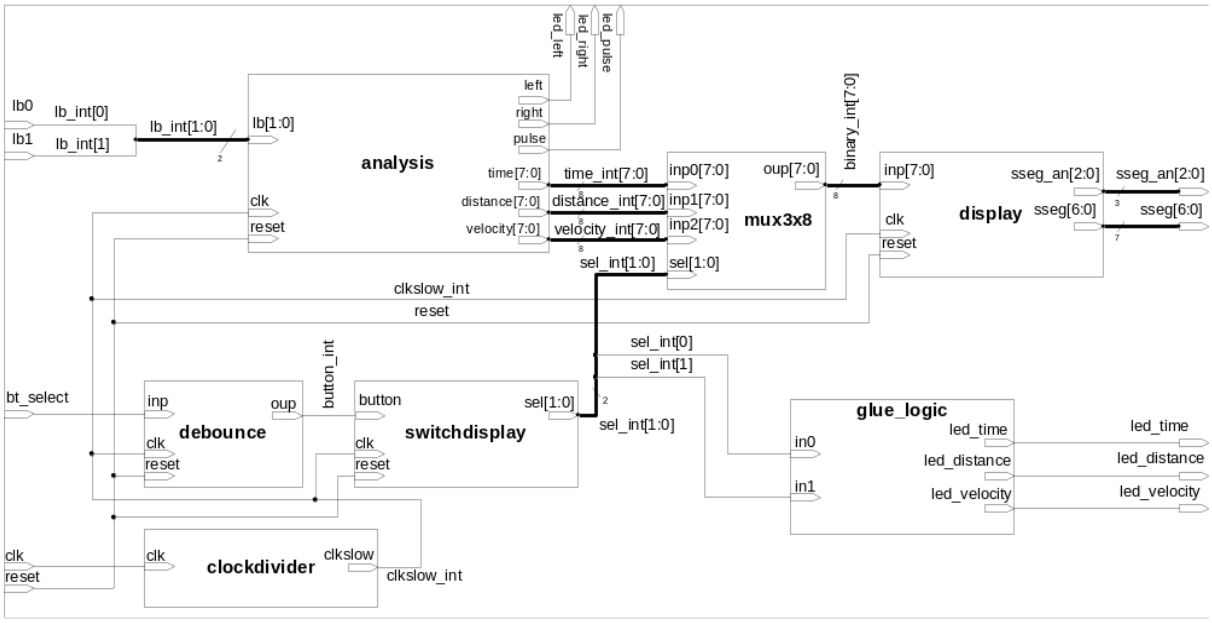

Abbildung 2: Top-Schema des Modules "motorproject".

Realisieren Sie dieses Schema in VHDL. Die in diesem Schema enthaltenen Blöcke (Komponenten) sind natürlich noch nicht ausprogrammiert. Das wollen wir ja im Verlaufe des Semesters erst machen. Wir können diese Komponenten trotzdem schon in unser Top- Schema einbinden. Die Eingänge und Ausgänge der Komponenten sind im Sinne eines Top-Down - Designs schon definiert. Sie finden alle diese **.vhd**-Files im Praktikumsordner (***~/Praktika/DigDes/vorgabe/Lab04/***).

Der Block ***glue_logic*** hat folgende Ausgänge: ***led_time***, ***led_distance***, ***led_velocity***. Diese sollen eine logische Funktion der Eingänge ***in0*** und ***in1*** sein. Folgende Tabelle beschreibt diesen Block, den Sie funktional in VHDL entwerfen.

Selektionsleitung (Eingänge)		Ausgänge		
in1	***in0***	***led_time***	***led_distance***	***led_velocity***
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	1	0	0	0

Tabelle 1: Wahrheitstabelle Glue_Logic

Tipp: Logische Funktionen in VHDL beschreibt man folgendermassen:

```
ausgang <= eingang(0) and eingang(1) or (not eingang(2)) xor eingang(3)
```

Wobei in diesem Beispiel ***eingang*** ein Bus mit 4 Leitungen ist. ***eingang(0)*** ist die Leitung mit der Nummer 0 des Busses ***eingang***, also normalerweise das LSB.

PreLAB: VORBEREITUNG

1. Lesen Sie das Dokument „*Beschreibung DC Motor Board.pdf*“. In diesem Dokument ist die Funktionsweise des Motorprojektboards beschrieben. Sie finden das Dokument auf Moodle. Wir werden im Verlaufe des Praktikums in Digital Design das gesamte Motorprojekt realisieren und am Schluss mit der gegebenen Hardware testen.

2. Bestimmen Sie die Instanznamen der Blöcke (Komponenten) im Schema ***motorproject_top***.

3. Beschreiben Sie die logische Funktion der Ausgänge ***led_time***, ***led_distance***, ***led_velocity*** ausgehend von Tabelle 1 in Form einer Booleschen Gleichung.

InLAB: DURCHFÜHRUNG UND DATENSAMMLUNG

1. **Nur falls noch nicht geschehen**: Erstellen Sie in ihrem Praktikaordner ***~/Praktika/DigDes*** einen Projektordner ***Motorproject*** für das Motorenprojekt. Kopieren Sie die bereits vorgegebenen Blöcke aus dem Ordner ***~/Praktika/DigDes/vorgabe/Lab04/*** in den Projektordner ***~/Praktika/DigDes/Motorproject***.

2. Realisieren Sie die im PreLAB entworfene Logik für die Kontroll-LED’s ***led_time***, ***led_distance***, ***led_velocity*** als neue VHDL Entity mit dem Namen *glue_logic* im File ***glue_logic.vhd***. Testen Sie ihre Beschreibung mit der Testbench ***glue_logic_tb.vhd***.

3. Erstellen Sie in Sigasi ein neues Projekt mit dem Namen ***Motorproject*** im Projektordner ***~/Praktika/DigDes/Motorproject***. Realisieren das Schema ***motorproject_top*** aus Abbildung 2 in VHDL im File ***motorproject_top.vhd***. Die Eingänge und Ausgänge der Komponenten sind schon definiert. Achten Sie darauf, dass Sie die Namen der Komponenten und deren Ein- und Ausgängen **nicht** verändern. Dies ist sehr wichtig sonst passen später Ihre Blöcke nicht zusammen.

4. Überprüfen Sie Ihre Arbeit indem Sie in Sigasi das Blockschema grafisch darstellen. Sie können dies unter ***Window > Show View > Block Diagram***.

5. Ein neues Vivado Project soll den Namen ***Motorproject*** erhalten und unter ***~/Praktika/DigDes/Motorproject*** abgelegt sein. Fügen Sie dem Projekt alle vorgefertigten Blöcke hinzu (inklusive ihrem vorher erstellten Block ***glue_logic.vhd***). Erstellen Sie die VHDL Beschreibung für das Schema ***motorproject_top*** automatisch, indem Sie ein Block-Design in Vivado erzeugen (***Project Manager / IP INTEGRATOR / Create Block Design***). Geben Sie bei *Design name* ***motorproject_top*** an. Die VHDL Module können direkt im Schema platzieren (***Rechtklick im Schema / Add Module…***) und sie anschliessend verdrahten.

   Für das Zusammenführen von einzelnen Signalen zu Bussen, sowie für das Abzweigen einzelner Signale aus einem Bus existieren in VIVADO IP Blöcke. Einen IP Block fügen Sie im Block-Design hinzu, indem Sie entweder analog zum Hinzufügen eines vhdl Moduls wieder mit ***Rechtsklick / Add IP…*** wählen. Die beiden Blöcke nennen sich ***Concat*** für das Zusammenfügen mehrere Signale zu einem Bus und ***Slice*** für das Auskoppeln einzelner Signale oder Signalgruppen.

   Input und Output Pins erhalten Sie mittels ***Rechtsklick / Create Port…***

   Das Top-level VHDL File erhalten Sie schliesslich, wenn Sie das Block-Design im Hierarchiebrowser anwählen und nach einem Rechtsklick die Option *Create HDL Wrapper…* selektieren. VIVADO erstellt auf Knopfdruck für Sie das vhdl-File, das Sie vorgängig mit viel Aufwand von Hand entworfen haben.

   Schlussendlich sollte Ihr Block Design so aussehen:

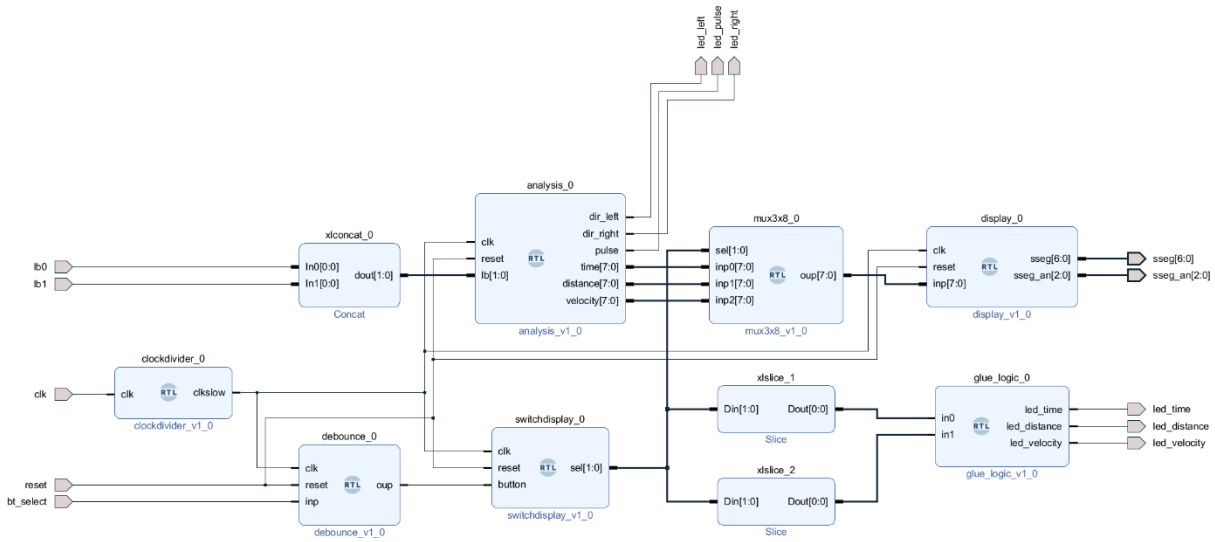

Abbildung 3: Block Design motorproject_top

6. (Optional) Realisieren Sie die ***glue_logic*** auf dem ZedBoard. Verwenden Sie dazu das Constraint File ***glue_logic.xdc*** und setzten sie das File ***glue_logic.vhd*** als Top Modul.

PostLAB: RESULTATE UND DISKUSSION

1. Was beschreibt man in einer Entity?

2. Welchen Namen geben sie sinnvollerweise der in diesem Praktikum erstellten architecture des Top-level Blockes ***motorproject_top***?

3. Wie heissen die Komponenten, die im Block ***motorproject_top*** verwendet werden?

4. Kann man von einer Komponente auch mehrere Instanzen platzieren, d.h. in einem Design verwenden? Wie macht man das?

5. Wie lauten die Instanznamen der Komponenten?

6. Wie viele interne Ein- und Ausgänge hat das ***motorproject_top***? Wie heissen diese?

7. Wie viele interne Signale (Leitungen) verwenden Sie im ***motorproject_top***? Wie heissen diese?

8. Wie deklarieren Sie interne Signale?

9. Weshalb verwenden Sie beim Deklarieren von Bussen das Schlüsselwort *downto* und nicht *to*?