Laborversuch 1

Einführung Laborsystem und Entwurfssoftware / RTL

1 Ziel des Laborversuchs

In diesem Laborversuch führen Sie ein einfaches Logik-Design-Projekt durch, um mit dem Laborsystem und der Entwicklungsumgebung Vivado vertraut zu werden. Die VHDL-Beschreibung eines 4-Bit-Zählers wird Ihnen dabei zur Verfügung gestellt.

Anhand dieser Aufgabe lernen Sie die Bedienung der Software Schritt für Schritt kennen. Erste VHDL-Kenntnisse aus der Vorlesung Hardwarebeschreibungssprachen werden vorausgesetzt: Entity, Architecture, Prozesse, Signale, Sensitivitätsliste, Testbench.

Die Herausforderung bei der Entwicklung digitaler Schaltungen steckt in der Umsetzung der Aufgabenstellung in eine Register-Transfer-Beschreibung. Eine solche Beschreibung lässt sich dann prinzipiell direkt in VHDL übersetzen. Das Erstellen von RTL-Modellen werden Sie in diesem und den weiteren Versuchen ausführlich üben.

Um für diesen ersten Laborversuch ausreichend vorbereitet zu sein, bearbeiten Sie **alle** Vorbereitungsaufgaben **rechtzeitig** vor Ihrem Labortermin (siehe Kapitel 2.3).

2 Einleitung

2.1 Entwurfsprozess

Als Ingenieur oder Ingenieurin wird es später Ihre Aufgabe sein, Lösungen für komplexe Aufgabenstellungen zu entwickeln. Sie bedienen sich dabei der passenden Werkzeuge (z.B. Entwicklungssoftware, Messgeräte), wählen eine geeignete Vorgehensweise (Methodik) und arbeiten sich strukturiert von einer abstrakten Beschreibung des Problems zu einer konkreten Lösung vor.

Grundsätzlich lässt sich der Entwurfsprozess bei der Entwicklung digitaler Schaltungen in folgende Schritte unterteilen:

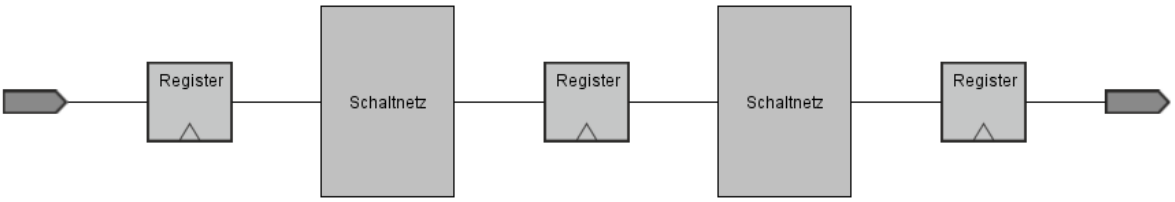
1. Konzeptphase
   - Spezifikation
   - Partitionierung
   - RTL-Entwurf
2. Implementierungsphase
   - VHDL-Kodierung
   - VHDL-Simulation
   - Elaboration / RTL-Analyse
   - Synthese / Implementierung
   - Konfiguration der Zielhardware
   - Test auf der Zielhardware

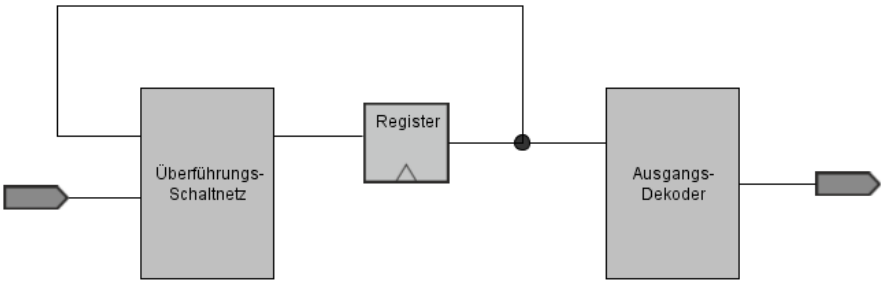
Der RTL-Entwurf wird im folgenden Kapitel beschrieben. Alle wesentlichen Schritte der Implementierungsphase lernen Sie im Rahmen dieses Laborversuchs kennen.

2.2 Register-Transfer-Logik

Beim Entwurf digitaler Schaltungen bedient man sich unterschiedlicher Abstraktionsebenen. Aus der Vorlesung Hardwarebeschreibungssprachen ist Ihnen der Register-Transfer-Level

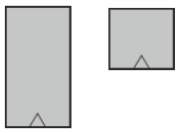
(RTL) bekannt. RTL-Modelle beschreiben synchrone digitale Schaltungen, deren Zustandsänderungen auf die Flanken eines Taktsignals synchronisiert sind. Sie enthalten getaktete Speicher (D-Flipflops) und Überführungslogik (Schaltnetze) zur Berechnung des nächsten Zustandes der Schaltung:



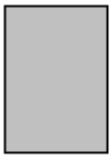


Hinter den Registern verbergen sich die bereits erwähnten D-Flipflops. Die Funktion der Schaltnetze und Dekoder kann z.B. mit Hilfe von Wahrheitstabellen, boole'schen Gleichungen oder Ablaufdiagrammen dargestellt werden.

Als Werkzeug zum Zeichnen von RTL-Blockschaltbildern verwenden Sie das freie Programm *yEd* (www.yworks.com). Für dieses Programm wird Ihnen eine Symbolbibliothek zum Zeichnen von RTL-Blockschaltbildern bereitgestellt. Die Bibliothek enthält folgende Symbole:



Register, D-Flipflop



Überführungslogik / Schaltnetz

Verbindungs-/Verzweigungspunkt für Signale

Negation

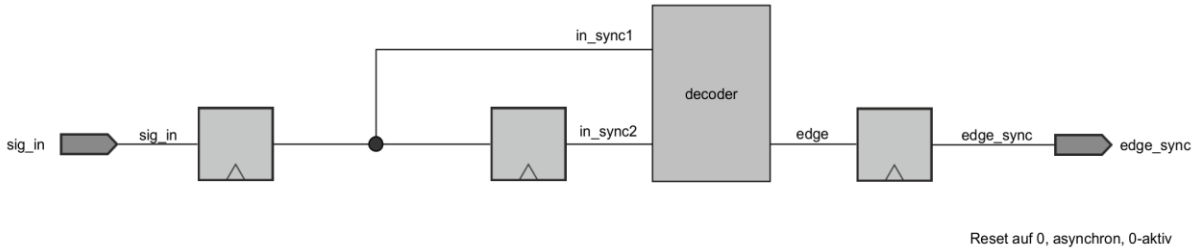
Ein-/Ausgang (Pfeil zeigt in Richtung des Signalflusses)

Die kleinen Dreiecke an Registern und D-Flipflops kennzeichnen den Takteingang. Ist das Dreieck leer, speichert das Flipflop bei der steigenden Taktflanke, ist das Dreieck ausgefüllt, bei der fallenden. Wenn alle Register mit dem gleichen Takt betrieben werden, lässt man die Verdrahtung des Taktsignals zur Verbesserung der Übersichtlichkeit meistens weg.

Aus dem gleichen Grund wird oft auch der asynchrone Reset-Eingang der Register inklusive Verdrahtung weggelassen. Stattdessen sollte ein entsprechender Vermerk in der Legende gemacht werden (z.B. *Reset asynchron, 1-aktiv*).

Zu einem vollständigen RTL-Blockschaltbild gehören außerdem Beschriftungen von Signalen und Ein-/Ausgangsports, sowie Angaben über deren Bitbreite – falls es sich um Bussignale handelt.

Das RTL-Modell eines Flankendetektors, also einer Schaltung, die für jede steigende Flanke des Eingangssignals am Ausgang einen Impuls erzeugt, könnte beispielsweise so aussehen:

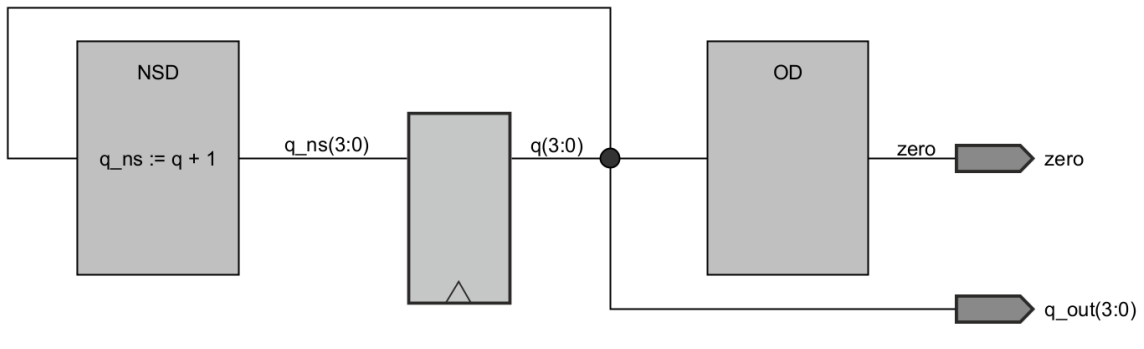


decoder:

in_sync1	in_sync2	edge
0	0	0
0	1	0
1	0	1
1	1	0

Zur Beschreibung der Dekoderfunktion wurde hier eine Wahrheitstabelle verwendet.

Ein weiteres Beispiel ist das RTL-Blockschaltbild eines 4-Bit-Zählers, der mit der fallenden Taktflanke zählt und über einen zusätzlichen Ausgang den Zählerstand `0` signalisiert:



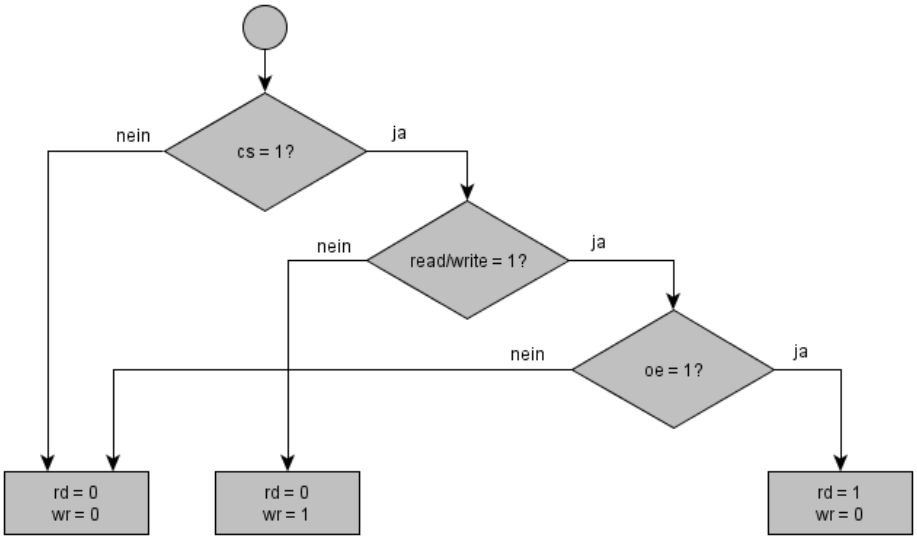

```
OD:

if q = 0 then
  zero := 1
else
  zero := 0
end if
```

NSD und *OD* sind gebräuchliche Abkürzungen für *Next State Decoder* beziehungsweise *Output Decoder*, diese kennzeichnen also Schaltnetze. Alternativ könnten die Blöcke auch mit ÜSN (Überführungsschaltnetz) und ASN (Ausgangsschaltnetz) beschriftet werden. Die Schaltfunktionen der beiden Dekoder wurden als Pseudocode angegeben – die Funktion des NSD

im Block, die des OD in der Legende. Letzteres bietet sich vor allem dann an, wenn die Schaltfunktion wie im nachfolgenden Beispiel etwas komplexer ist:

```
IF cs = 1 THEN
      IF read/write = 1 THEN
            IF oe = 1 THEN
                  rd := 1; wr := 0
            ELSE
                  rd := 0; wr := 0
      ELSE
            rd := 0; wr := 1
ELSE
      rd := 0; wr := 0
```

Eine Alternative zum Pseudocode oder zur Wahrheitstabelle ist die grafische Darstellung der Schaltnetzfunktion als Ablaufdiagramm:



2.3 Signale

In VHDL dienen Signale zur Kommunikation zwischen Prozessen und Komponenten.

Signalbündel werden auch als *Busse* bezeichnet und in Blockdiagrammen üblicherweise `signalname(MSB:LSB)` benannt (*MSB* = most significant bit, *LSB* = least significant bit). An den Ausgang eines 4-Bit-Zählers könnte z.B. ein Signal `cnt(3:0)` angeschlossen sein.

Als *High-aktiv*, bzw. *1-aktiv* bezeichnet man ein Signal, wenn die ihm zugeordnete Funktion durch den logischen Signalwert 1 repräsentiert wird. Muss beispielsweise zum Zurücksetzen eines Zählers ein 1-Pegel am Reseteingang anliegen, so kann der Reset nur durch ein 1-aktives Resetsignal erfolgen. Entsprechendes gilt für *Low-aktive* bzw. *0-aktive* Signale.

3 Vorbereitungsaufgaben

V1 Machen Sie sich anhand des Dokuments *Technischer Anhang* mit der im Labor verwendeten Hardware vertraut.

V2 Mit dem Programm *yEd* werden Sie RTL-Blockschaltbilder erstellen. Absolvieren Sie die E-Learning-Einheit „*yEd*“ auf der E-Learning-Plattform der Hochschule.

V3 Bearbeiten Sie das Dokument *Vivado-Einführungskurs, Teil 1* bis zum Kapitel *Erzeugen des Bitstreams*.

Beantworten Sie folgende Fragen zum Vivado-Einführungskurs (die Fragen beziehen sich auf die unveränderten Quellcodes aus den Vorlagen). Begründen Sie Ihre Antworten anhand passender Quellcode-Abschnitte.

a) Wie reagiert der Zähler `counter4` auf das Taktsignal? Reagiert der Zähler auf den Pegel oder auf eine Flanke des Taktsignals?

b) Wie reagiert der Zähler `counter4` auf das Resetsignal? Reagiert der Zähler auf den Pegel oder auf eine Flanke des Resetsignals?

c) Wie heißen die Signale in der Testbench (nicht die Ports der Komponente `counter4`!), die an den Takt- und Reseteingang des Zählers angeschlossen sind?

d) In welchem der Testbench-Prozesse wird das Takt-, in welchem das Resetsignal generiert? Beschreiben Sie die Arbeitsweise der beiden Prozesse!

e) Das FPGA auf dem Laborsystem kann extern mit einer Frequenz von 100 MHz getaktet werden (siehe Dokumentation zum BASYS3-Board). Welche Periodendauer des Taktes ergibt sich daraus? Wo könnten Sie diese Periodendauer im Quellcode der Testbench einstellen?

f) Welchen Signalpegel hat das von der Testbench generierte Reset-Signal in seiner aktiven Phase? Passt der Pegel des Reset-Impulses zu den Anforderungen der VHDL-Komponente, die Sie testen möchten? Wo können Sie ggf. den Signalpegel des generierten Reset-Signals ändern?

g) Wie können Sie den Taktgeber des Peripherieboards mit einem FPGA-Anschlusspin verbinden?

V4 Erstellen Sie mit *yEd* das RTL-Blockschaltbild eines 6-Bit-Zählers (Reset 0-aktiv, asynchron, setzt den Zähler auf 0 zurück; Zustandswechsel mit steigender Taktflanke), der in Zweierschritten abwärts zählt und seinen Zählerstand über das Signal `data_out(5:0)` ausgibt. Über einen Ausgangsdekoder soll ein zusätzliches Signal `cnt24` mit einem High-Pegel den Zählerstand 24_{10} anzeigen.

V5 Fassen Sie die nachfolgend aufgeführten Informationen und Dokumente in einem PDF-Dokument zusammen.

Bringen Sie das PDF-Dokument zur Laborveranstaltung mit.

Dateiname: `HBS-Laborbericht_V1_Nachname_Vorname.pdf`

- Kopfzeile: *HBS-Labor, V1 – Datum – Nachname, Vorname*
- Schriftliche Beantwortung der Fragen zum Vivado-Einführungskurs

- Korrigierter VHDL-Code des 4-Bit-Zählers aus dem Vivado-Einführungskurs[1]
- VHDL-Code der Testbench zum 4-Bit-Zähler aus dem Vivado-Einführungskurs[1]
- Screenshot der Signalverläufe in der Simulation[2]
- Constraints-Datei `constr.xdc`
- Screenshot des Fensters *Project Summary* (wählen Sie in diesem Fenster vor der Erstellung des Screenshots unter *Utilization* die Darstellung *Table*)
- RTL-Blockschaltbild des 6-Bit-Zählers

4 Freiwillige Vorbereitungsaufgabe zur weiteren Vertiefung

Es soll ein Zähler entwickelt werden, der mit jeder steigenden Taktflanke in Einerschritten von 3 bis 14 aufwärts zählen und nach Erreichen der 14 wieder bei 3 beginnen soll. Der Reset sei asynchron und 1-aktiv und setze den Zähler auf 3 zurück.

- Erstellen Sie zur o.g. Spezifikation des Zählers mit yEd das zugehörige RTL-Blockschaltbild und geben Sie die Funktion des Schaltnetzes an.
- Erstellen Sie ein neues Vivado-Projekt und fügen Sie die vorgegebene Datei `counter_3_14.vhd` zum Projekt hinzu.
- Erstellen Sie eine Testbench mit Hilfe des Testbench-Generators. Erzeugen Sie in der Testbench die notwendigen Stimuli für einen vollständigen Test des Zählers.
- Simulieren Sie das vorgegebene VHDL-Modell des Zählers und korrigieren Sie evtl. vorhandene Fehler. Überprüfen Sie anhand der Signalverläufe, ob der Zähler korrekt arbeitet (Resetverhalten, Startwert, Schrittweite, Zählrichtung, Taktflanke, Überlauf) und korrigieren Sie ggf. den VHDL-Quellcode.
- Überprüfen Sie die Funktion der Schaltung mit der selbsttestenden Testbench `counter_3_14.vhd`.
- Definieren Sie die Randbedingungen für die Synthese. Orientieren Sie sich dabei am *Vivado-Einführungskurs, Teil 1*: Der Zählerstand soll über LEDs auf dem BASYS3-Board ausgegeben werden, der Takt kommt vom Peripherieboard, als Reset-Taster wird BTNC auf dem BASYS3-Board verwendet.
- Synthetisieren und implementieren Sie Ihren Entwurf und erzeugen Sie einen Bitstream.

  Kontrollieren Sie die Warnmeldungen und korrigieren Sie ggf. das VHDL-Modell des Zählers.

  Wählen Sie im Fenster *Project Summary* unter *Utilization* die Darstellung *Table* und überprüfen Sie die Auslastung des Bausteins.

[1] Formatieren Sie Quellcode mit einer nichtproportionalen Schriftart in relativ kleiner Schriftgröße (z.B. Courier New, 9 Punkt).

[2] Achten Sie darauf, dass im Screenshot alle relevanten Informationen deutlich erkennbar sind. Löschen Sie ggf. unnötige Signale. Zum Erstellen des Screenshots ist es hilfreich, das Fenster mit der Darstellung der Signalverläufe vorher auszudocken und die Fensterdimensionen an den Inhalt anzupassen.