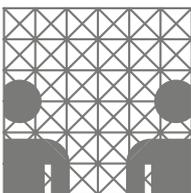


# Praktikum: VLSI-Entwurf

6

## VHDL Standardzell Entwurf Entwurf einer Weckuhr



Andreas Mäder

Universität Hamburg – MIN – Fachbereich Informatik  
Arbeitsbereich Technische Aspekte Multimodaler Systeme

<http://tams.informatik.uni-hamburg.de>

In diesem Versuch wird ein Wecker mit Hilfe der VHDL Simulations- und Syntheseprogramme entworfen. Dabei sollen die Erfahrungen der letzten beiden Versuche, zum Umgang mit den Werkzeugen sowie zur synthesesgerechten VHDL-Modellierung, eingebracht werden.

### Arbeitsweise

Es soll ein hierarchisch aufgebauter, synthetisierbarer VHDL-Entwurf erstellt werden — in Hinblick auf ein späteres Standardzelldesign.

1. Beschreiben Sie die einzelnen Komponenten (Hauptblöcke) der Schaltung durch VHDL Verhaltensbeschreibungen auf Register-Transfer Ebene.
2. Simulieren Sie diese Schaltungen in geeigneten Testumgebungen, um die Funktionalität zu prüfen.
3. (optional) Neben der Funktion ist auch die Synthetisierbarkeit der VHDL-Eingabe, durch Abbildung der (Teil-)Entwürfe auf die Zellbibliothek, zu überprüfen.
4. Anschließend wird der Wecker aus den einzelnen Komponenten als Strukturbeschreibung zusammengesetzt.
5. Auch diese Beschreibung muss simuliert werden.
6. Die Gesamtschaltung ist zu synthetisieren und anschließend als VHDL-Gatternetzliste zu simulieren.

**Tipp:** Da diese Simulation unter Umständen schon länger dauern kann, empfiehlt es sich nur einzelne Komponenten auf der Gatterebene zu simulieren, während andere als ursprüngliche Verhaltensbeschreibung simuliert werden.

Dies ist in VHDL durch Konfigurationen möglich, wobei die Beschreibung der gesamten Schaltung – die hierarchische Beschreibung des Weckers – als Testumgebung für die jeweiligen (synthetisierten) Komponenten dient.

## Aufgaben

VI-1 Entsprechend der oben beschriebenen Vorgehensweise sollen Sie einen **Wecker** entwerfen. Dabei sind keine besonderen Vorgaben einzuhalten, sie können also Ihren Ideen freien Lauf lassen. . .

Eine mögliche Unterteilung der Schaltung in einzelne Komponenten ist in Abb. 1 dargestellt, dazu gleich einige Anmerkungen:

- Die Skizze stellt hier nur einen ersten Schritt in einem hierarchischen Entwurf dar. Weitere Unterteilungen, außer der skizzierten Referenz auf bcddec in outmux, können zwar vorgenommen werden, sind aber nach den bisherigen Erfahrungen nicht notwendig.
- Zusätzliche Blöcke auf dieser Ebene könnten bei erweiterter Funktionalität benötigt werden, beispielsweise für zentrale Kontrolleinheiten.
- Die nachfolgenden Beschreibungen der einzelnen Komponenten sollen jeweils nur einen ersten Eindruck von einer „Mindestfunktionalität“ liefern. Erweiterungen sind sicherlich denkbar für:
  - komfortablere Einstellmöglichkeiten für die Uhr- und Weckzeit:  
vorwärts-/rückwärts zählen  
mehrere Geschwindigkeiten  
...
  - erweiterte Weckfunktionen:  
Nachweckautomatik  
mehrere Weckzeiten (datumsabhängige Weckzeiten, s. Aufgabe 7)  
...
- Zusätzliche Ein- und Ausgänge sind zwar möglich, es ist aber darauf zu achten, dass ihre Zahl nicht zu sehr steigt, da der Entwurf sonst als fertiges IC *padlimitiert* wird (d.h. die Fläche des Entwurfs wird durch die Anzahl der I/O-Pads bestimmt und nicht durch die Komplexität der internen Logik).

Werden noch viele zusätzliche Ein- oder Ausgänge benötigt, so ist zu überlegen, ob sich durch geeignete Multiplexlösungen, die zu einer Mehrfachbenutzung der Eingänge führen, nicht wieder externe Anschlüsse einsparen lassen.

Um die Ausgabe einfacher zu realisieren und um das spätere Interface zu dem DCF 77 Decoder zu vereinfachen, sollte für die interne Zahlendarstellung eine BCD-Codierung gewählt werden. Die Ein- und Ausgänge der einzelnen Strukturelemente können jeweils hinsichtlich der tatsächlich benötigten Bitbreiten minimiert werden — dementsprechend wird keine „richtige“ BCD-Codierung benutzt.

Wie schon oben erwähnt ist die Beschreibung der Blöcke als ein erster Vorschlag und nicht als vollständige Spezifikation zu verstehen — entsprechende Beispiele sind in dem Verzeichnis `templates` vorhanden.

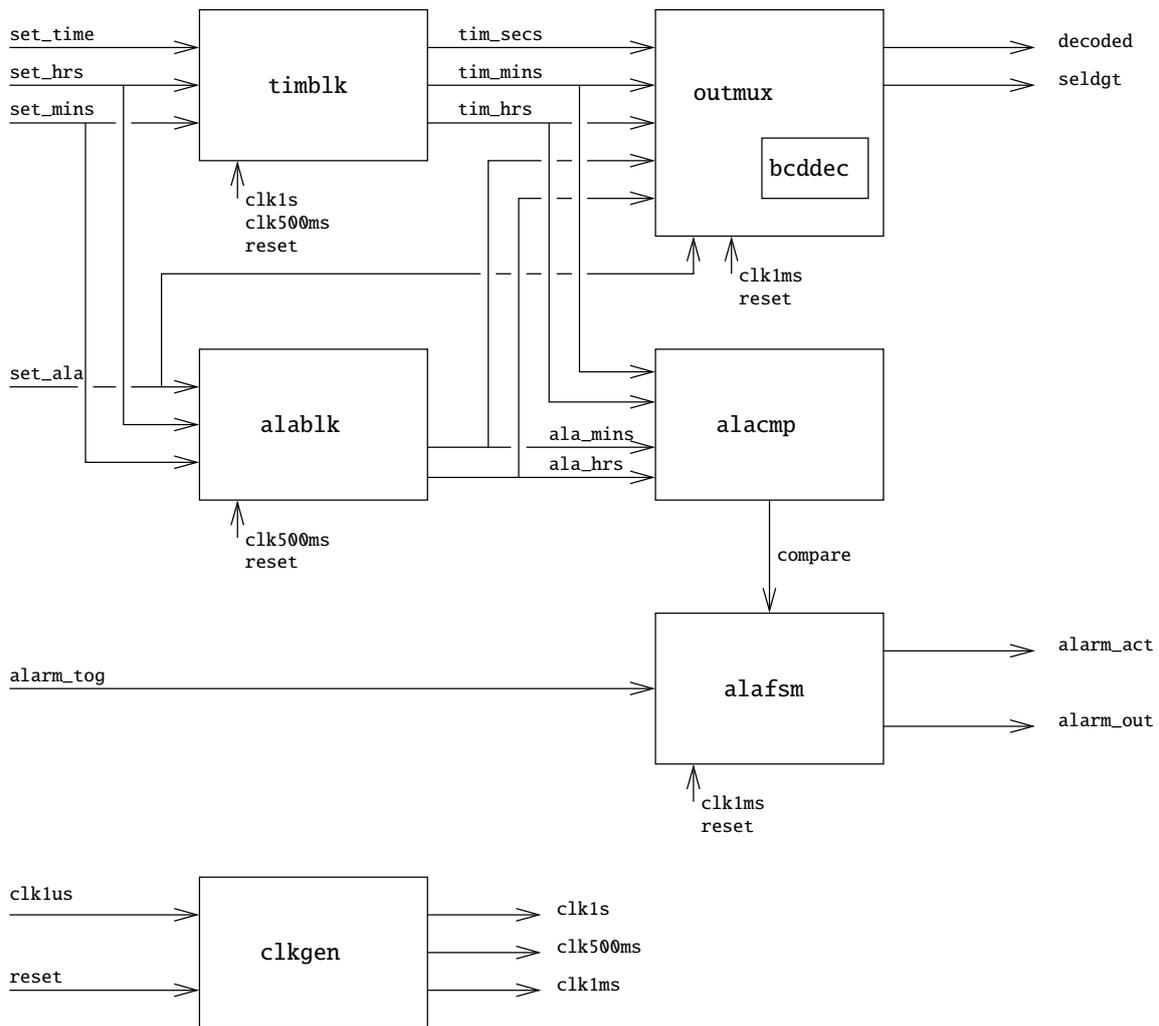


Abbildung 1: Blockstruktur der Weckuhr

**clkgen** : Taktuntersetzer, für die intern benötigten Arbeitstakte.

Signal	Wirkung	Funktion
reset	active Low	asynchrones Reset
clk1us	Vorderflanke	externer Takt 1MHz
clk1ms	Flanke	interner Takt 1kHz
clk500ms	Flanke	interner Takt 2Hz
clk1s	Flanke	interner Takt 1Hz

**alafsm** : Automat zur Wecksteuerung, siehe Versuche 4 und 5.

Neben der hier vorgestellten Partitionierung in `alafsm` und `alacmp`, ist es auch möglich die Funktion des Vergleichers direkt in `alafsm` zu integrieren.

**timblk** : entspricht dem eigentlichen Uhrwerk.

set\_time = '0' die Uhrzeit wird im Sekundentakt raufgezählt, dazu sind entsprechende Modulo-Zähler zu entwerfen.

set\_time = '1' setzt die Sekunden auf "00" zurück und bereitet das manuelle Stellen der Uhrzeit vor.

set\_time = '1'  $\wedge$  set\_hrs = '1' Uhrzeit einstellen: zählt die Stunden mit der 500ms-Clock (rauf).

set\_time = '1'  $\wedge$  set\_mins = '1' Uhrzeit einstellen: zählt die Minuten mit der 500ms-Clock (rauf), dabei findet kein Stundenübertrag statt.

**Tip:** Der Aufbau als *synchroner* Zähler ist am einfachsten zu implementieren und am sichersten im Betrieb.<sup>1</sup>

Signal	Wirkung	Funktion
reset	active Low	asynchrones Reset
clk1s	Vorderflanke	„normaler“ Takt
clk500ms	Vorderflanke	Takt zum Stellen
set_time	active High	Stellen der Uhrzeit
set_hrs	active High	Stellt Stunden
set_mins	active High	Stellt Minuten
tim_secs1	4-bit Bus	Sekunden 1er
tim_secs10	3-bit Bus	Sekunden 10er
tim_mins1	4-bit Bus	Minuten 1er
tim_mins10	3-bit Bus	Minuten 10er
tim_hrs1	4-bit Bus	Stunden 1er
tim_hrs10	2-bit Bus	Stunden 10er

**alablk** : Register und Zähler für den Weckmechanismus.

set\_alarm = '1'  $\wedge$  set\_hrs = '1' Weckzeit einstellen: zählt die Stunden mit der 500ms-Clock (rauf).

set\_alarm = '1'  $\wedge$  set\_mins = '1' Weckzeit einstellen: zählt die Minuten mit der 500ms-Clock (rauf), dabei findet kein Stundenübertrag statt.

Signal	Wirkung	Funktion
reset	active Low	asynchrones Reset
clk500ms	Vorderflanke	Takt zum Stellen
set_alarm	active High	Stellen der Alarmzeit
set_hrs	active High	Stellt Stunden
set_mins	active High	Stellt Minuten
ala_mins1	4-bit Bus	Minuten 1er
ala_mins10	3-bit Bus	Minuten 10er
ala_hrs1	4-bit Bus	Stunden 1er
ala_hrs10	2-bit Bus	Stunden 10er

<sup>1</sup>Alle Stellen des Zählers werden in einem VHDL-Prozess – mit einer Clock – gesetzt.

**outmux** : steuert die 7-Segment Anzeige an.

set\_alarm = '1' zum Stellen der Weckzeit wird die Ausgabe auf die aktuelle Weckzeit umgeschaltet; normalerweise wird die Uhrzeit angezeigt.

- Um nicht für jede Ziffer einen eigenen 7-bit Bus als Ausgang vorzusehen, soll die Ausgabe im Zeitmultiplex betrieben werden. Dafür gibt es zwei Ausgangsbusse: Einer steuert die 7-Segmente an (decoded), während ein Zweiter eine Ziffer aktiviert (seldgt).

Über eine Zeitschleife wird der Reihe nach jede Ziffer angesprochen, wobei auf dem Bus decoded der decodierte Wert der jeweiligen Stelle ausgegeben wird. Dabei ist darauf zu achten, dass Segmente beim Umschalten der Stelle – Synchronisation von decoded und seldgt – nicht mit falschen Werten angesteuert werden. Gegebenenfalls muss man beim Umschalten der Digits die Segmentleitungen über ein Ausblendsignal dunkel tasten.

- Eine Erweiterung der Funktion wäre, dafür zu sorgen, dass die Ziffer Null an der Stelle der 10-er Stunden nicht angezeigt wird, sondern diese Stelle der Anzeige leer bleibt.

Signal	Wirkung	Funktion
reset	active Low	asynchrones Reset
clk1ms	Vorderflanke	Takt für internen Automaten
set_alarm	active High	Umschalten auf Alarmzeit
tim_secs1	4-bit Bus	Uhr-Sekunden 1er
tim_secs10	3-bit Bus	Uhr-Sekunden 10er
tim_mins1	4-bit Bus	Uhr-Minuten 1er
tim_mins10	3-bit Bus	Uhr-Minuten 10er
tim_hrs1	4-bit Bus	Uhr-Stunden 1er
tim_hrs10	2-bit Bus	Uhr-Stunden 10er
ala_mins1	4-bit Bus	Alarm-Minuten 1er
ala_mins10	3-bit Bus	Alarm-Minuten 10er
ala_hrs1	4-bit Bus	Alarm-Stunden 1er
ala_hrs10	2-bit Bus	Alarm-Stunden 10er
decoded	7-bit Bus	7-Segmente
seldgt	6-bit Bus	6-Stellen

**alacmp** : Vergleichler für das Auslösen des Alarms.

Signal	Wirkung	Funktion
tim_mins1	4-bit Bus	Uhr-Minuten 1er
tim_mins10	3-bit Bus	Uhr-Minuten 10er
tim_hrs1	4-bit Bus	Uhr-Stunden 1er
tim_hrs10	2-bit Bus	Uhr-Stunden 10er
ala_mins1	4-bit Bus	Alarm-Minuten 1er
ala_mins10	3-bit Bus	Alarm-Minuten 10er
ala_hrs1	4-bit Bus	Alarm-Stunden 1er
ala_hrs10	2-bit Bus	Alarm-Stunden 10er
compare	active High	Uhrzeit = Alarmzeit