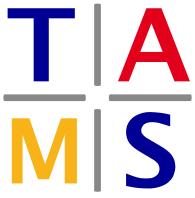
VHDL Schaltungs- und Systementwurf

1

Schaltnetze und Schaltwerke – Simulation



Andreas Mäder

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

https://tams.informatik.uni-hamburg.de

Dieses Aufgabenblatt enthält den ersten Schritt, hin zu einem kompletten top-down VLSI-Entwurf, der dann beispielsweise mit einem FPGA (ein programmierbarer Chip) realisiert werden kann. Dabei wird die Schaltung in der Hardwarebeschreibungssprache VHDL eingegeben und simuliert, wobei Sie die Möglichkeiten von VHDL kennen lernen werden, ein System und dessen Testumgebung zu beschreiben.

Anhand der ersten Schaltungsbeispiele sollen Sie sich in die Bedienung des VHDL Simulators einarbeiten.

Voraussetzungen

- VHDL Grundkenntnisse, beispielsweise aus „VHDL Kompakt“ zu finden auf: `https://tams.informatik.uni-hamburg.de/research/vlsi/vhdl` – `Documentation`

Beispiele

Beispieldateien und Templates für die Aufgaben **1** bis **4** sind über die Web-Seite der Veranstaltung zugänglich. Die Datei `vlsiPrak.tgz` enthält folgende Verzeichnisse:

`fsm-examples` Beispiele zur Automatenbeschreibung mit VHDL. Dabei werden meist zwei Prozesse benutzt: Einer der die Logik der beiden Schaltnetze δ und λ in Form einer großen Case-Anweisung beinhaltet und ein zweiter Prozess, der das Zustandsregister modelliert und über seinen Takteingang die Schaltung synchronisiert.

`delayLine` Der Beispielcode von den VHDL-Einführungsfolien.

`trafficLight` Die (Fußgänger-) Ampelsteuerung aus der Vorführung.

`templates` Entity-Deklarationen für eine Partitionierung der Schaltung entsprechend den Praktikumsunterlagen.

- `dcf77.vhd` enthält ein Modell des DCF-Senders zur Simulation des Decoderautomaten (`dcffsm`).
- `tstClock.vhd` ist die Simulationsumgebung des gesamten Entwurfs.

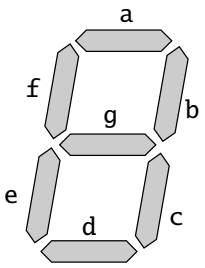
Arbeitsweise

1. Erstellen Sie eine VHDL (Verhaltens-) Beschreibung der Schaltung.
2. Besitzt die Schaltung Clock- oder andere Eingänge, an denen festgelegte Signalfolgen angelegt werden, sollten Sie eine Testumgebung aufbauen, die diese externen Signale über eigene Prozesse erzeugt und die entworfene Schaltung instanziiert (Hierarchie).
3. Simulieren Sie die Schaltung, bzw. deren Testumgebung, um die Funktion zu prüfen.

Aufgabe 1.1

Entwerfen Sie einen **BCD-zu-7 Segment Decoder**. Das Interface der Schaltung ist wie folgt beschrieben: (in `bcddec.vhd`)

```
entity bcddec is
  port( bcdin   : in  std_logic_vector (3 downto 0);
        decoded : out std_logic_vector (6 downto 0));
end entity bcddec;
```




```
decoded(6  downto  0)
        a   ...   g
```

Abbildung 1: Zuordnung der Ausgangssignale

Überprüfen Sie die Korrektheit der entworfenen Schaltung durch geeignete Simulation und lernen Sie dabei die Fähigkeiten des Simulators kennen.

Aufgabe 1.2

Entwerfen Sie einen **Automaten**, der die „Klingel“ eines Weckers steuert und simulieren Sie den Entwurf. Die Funktion der Schaltung ist in dem Zustandsübergangsdiagramm in Abbildung 2 skizziert.

Der (asynchrone) Reset ist in dem Diagramm nicht dargestellt, er wirkt direkt auf das Zustandsregister (siehe Automatenbeschreibung: `mealy.vhd` und `moore.vhd`).

Die Schaltung hat folgende Schnittstellen: (in `alafsm.vhd`)

```
entity alafsm is
  port( reset           : in  std_logic;
        clk1ms          : in  std_logic;
        alarm_tog       : in  std_logic;
        compare         : in  std_logic;
        alarm_act       : out std_logic;
        alarm_out       : out std_logic);
end entity alafsm;
```

Signal	Funktion	Wirkungsweise
`reset`	asynchrones Reset	active Low
`clk1ms`	Clockeingang 1KHz	Vorderflanke
`alarm_tog`	Schaltet Alarm an/aus	active High
`compare`	aktuelle Zeit = Alarmzeit	active High
`alarm_act`	LED: Alarm aktiv	active High
`alarm_out`	Klingel	active High

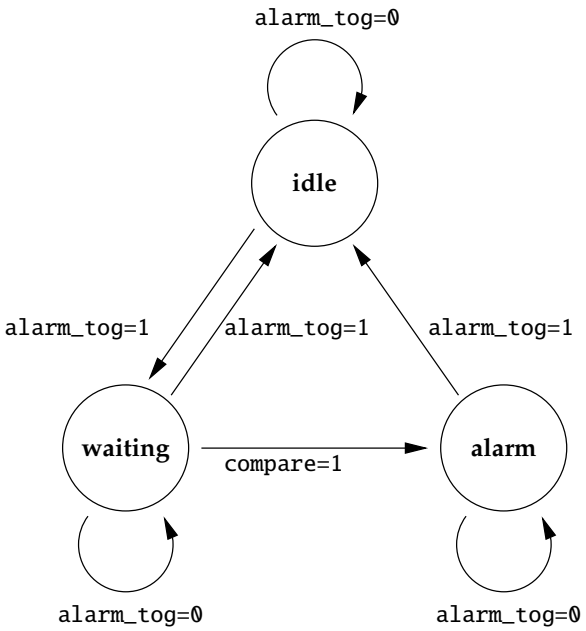


Abbildung 2: Alarm-Automat

Tipps

- Es ist darauf zu achten, dass die Tastenbetätigung von `alarm_tog` sehr viel länger dauert, als der Takt. Eine direkte Umsetzung des oben skizzierten Zustandsübergangsgraphen ist also *nicht* möglich!

  Der Automat muss also so beschrieben werden, dass zwischen den Übergängen durch den Toggle-Eingang die Taste auch wieder losgelassen wird. Dazu können beispielsweise Zustände verdoppelt werden und auf beide Eingangswerte (`1` und `0`) abfragen.

- Für die Ausgangssignale wurde noch nicht festgelegt, ob sie über Mealy- oder Moore-Modelle beschrieben werden. Wegen des synchronen Verhaltens (einfacher handhabbar) ist ein Moore-Automat vorzuziehen.