Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

VHDL Schaltungs- und Systementwurf

2

Schaltnetze und Schaltwerke – Synthese

Andreas Mäder

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

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

Ausgehend von den Schaltungen des letzten Aufgabenblatts soll eine Netzliste aus Elementen einer Standardzellbibliothek synthetisiert werden. Unter Beibehaltung der Testumgebung ist eine anschließende VHDL Simulation dieser synthetisierten Netzliste möglich.

Synthese aus VHDL-Code

Die ganze „Kunst" des VHDL-Designs besteht darin *synthesegerechte Eingaben* zu schreiben.

Viele EDA-Werkzeuge können VHDL-Beschreibungen synthetisieren, wobei die Logiksynthese, die Synthese endlicher Automaten und die Synthese von Datenpfaden sehr gut beherrscht wird. Die Umsetzung beliebiger VHDL-Verhaltensmodelle in eine „optimale" Hardwarerealisierung ist aber wegen des, mit der Abstraktion größer werdenden Suchraums nicht möglich — was ist optimal? Nutzt man eine ALU und berechnet Werte sequenziell (kleiner, aber langsamer, ggf. Zwischenspeicher nötig) oder werden mehrere Rechenwerke parallel genutzt (schneller, aber mehr Fläche). Für eingeschränkte Anwendungsfälle sind aber auch hier schon Werkzeuge verfügbar, die „Hand"-Entwürfen vergleichbare Ergebnisse produzieren (in einem Bruchteil der Zeit).

Bei der Beschreibung von für die Hardwaresynthese geeigneten VHDL-Eingaben spielen zwei Faktoren eine Rolle:

1. (a) Die Eingabe auf einer Abstraktionsebene, die von dem Synthesewerkzeug bearbeitet werden kann.

Dies sind in der Regel funktionale Blöcke als Register-Transfer Modelle und darauf aufbauende Strukturbeschreibungen. Beim Entwurf hat der Designer ohnehin schon bestimmte Vorstellungen von Hardwarestrukturen und Funktionseinheiten.

1. (b) Darüberhinaus muss die VHDL-Eingabe in einer Art und Weise beschrieben sein, die vom Synthesesystem verarbeitet werden kann. Dies ist neben dem Verbot/der Präferenz bestimmter Formulierungen meist auch eine Einschränkung des Sprachumfangs (VHDL-Subset).

2. Die Funktionseinheiten sollten so beschrieben werden, dass sie von den Synthesewerkzeugen *gut* umgesetzt werden können.

Analog zu den Programmiersprachen kann das Verhalten der Hardware (des Programms) auf beliebig viele, verschiedene Arten formuliert werden. Nach der Synthese (der Compilation) ergeben sich dann unterschiedlich effiziente Lösungen.

Arbeitsweise

1. Synthetisieren Sie eine Gatternetzliste für den AMS $0,35\,\mu m$-Prozess.

2. Bewerten Sie die synthetisierten Strukturen, also: *Entspricht das Syntheseergebnis der gewünschten Schaltung?*

Hiermit ist gemeint, dass beispielsweise die speichernden Elemente (eingefügte Register) auch wirklich den taktabhängigen Zuweisungen an Signale oder Variablen im Code entsprechen und nicht Folge einer *ungeschickten* VHDL-Beschreibung sind.

Hier ist insbesondere darauf zu achten, ob nicht ungewollt *Latches* in die Schaltung eingefügt worden sind — Typischerweise passiert dies, wenn eigentlich ein Schaltnetz beschrieben werden sollte, wegen bedingter Ausführung eine Zuweisung aber nicht in allen Fällen stattfindet.

3. Sichern Sie das Syntheseergebnis als VHDL-Beschreibung.
4. Simulieren Sie anschließend die synthetisierte Netzliste, wenn möglich unter Beibehaltung der „alten“ Testumgebung.

Aufgabe 2.1

Synthetisieren Sie den **BCD-zu-7 Segment Decoder** aus Aufgabe **1.1** und simulieren Sie das Ergebnis.

Versuchen Sie eine möglichst effiziente Gatterrealisierung zu erreichen (schnell, klein…) indem Sie den VHDL-Code für die Synthese optimieren:

- keine Latches
- *don't-care* Werte ausnutzen
- Zustandscodierung von Automaten (s.u.)
- …

Aufgabe 2.2

Synthetisieren Sie den **Automaten** aus Aufgabe **1.2** und simulieren Sie das Ergebnis.

Achten Sie darauf, dass keine unnötigen speichernden Elemente (Flipflops) generiert worden sind — deren Anzahl ergibt sich aus der Zahl der Zustände, entsprechend den Zustandsbits für eine Codierung.