Dieter Wecker
Prozessorentwurf mit VHDL
De Gruyter Studium

Weitere empfehlenswerte Titel

FPGA Hardware-Entwurf
F. Kesel, 2018
ISBN 978-3-11-053142-8, e-ISBN (PDF) 978-3-11-053145-9,
e-ISBN (EPUB) 978-3-11-053199-2

Lehrbuch Digitaltechnik, 4. Auflage
J. Reichardt, 2016
ISBN 978-3-11-047800-6, e-ISBN (PDF) 978-3-11-047834-1,
e-ISBN (EPUB) 978-3-11-052997-5

VHDL Synthese, 7. Auflage
J. Reichardt, B. Schwarz, 2015
ISBN 978-3-11-037505-3, e-ISBN (PDF) 978-3-11-037506-0,
e-ISBN (EPUB) 978-3-11-039784-0

Analoge Schaltungstechniken der Elektronik
W. Tenten, 2012
ISBN 978-3-486-70682-6, e-ISBN 978-3-486-85418-3

Dieter Wecker

Prozessorentwurf mit VHDL

Modellierung und Synthese eines 12-Bit-Mikroprozessors

DE GRUYTER
OLDENBOURG

Autor
Prof. Dr. Dieter Wecker
82008 Unterhaching
dweck@online.de

ISBN 978-3-11-058256-7
e-ISBN (PDF) 978-3-11-058306-9
e-ISBN (EPUB) 978-3-11-058283-3

Library of Congress Control Number: 2018941440

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de abrufbar.

Umschlaggestaltung: bpm82/iStock/Getty Images
Satz: le-tex publishing services GmbH, Leipzig
Druck und Bindung: CPI books GmbH, Leck

www.degruyter.com

Vorwort

Der Entschluss zu diesem Buch resultiert aus dem Versuch, einen einfachen funktionsfähigen Mikroprozessor von der Planung bis zum Prototyp zu entwickeln. Es wird ein 12-Bit-Mikroprozessor entworfen, wie er neben anderen Prozessoren in meinem Lehrbuch „Prozessorentwurf“ behandelt wird. Der 12-Bit-Mikroprozessor steht hier im Mittelpunkt und wird mit Hilfe von verschiedenen VHDL-Modellen und einer CAD (Computer Aided Design)- Entwicklungs-Software entworfen.

Die Hardware-Beschreibungssprache VHDL (Very High Speed Integrated Circuit Hardware Description Language) wird weltweit als Beschreibungssprache für digitale Systeme eingesetzt. Im Unterschied zu anderen Hochsprachen ist VHDL eine Hardware-Beschreibungssprache, d. h. die Hardware-Realisierung muss immer mit betrachtet werden.

Für den VHDL-Entwurf eines Prozessors ist es notwendig, sich mit dem VHDL-Code vertraut zu machen. Daher wird für alle erstellten Modelle der Source-Code ausführlich behandelt.

Beim Umgang mit VHDL ist es sinnvoll mit einer Entwicklungs-Software zu arbeiten, die folgende Bedingungen erfüllt:

- VHDL-Editor für den Source-Code
- Compiler für die Umsetzung in einen Binärcode
- Simulator für die Funktionale und Timing Simulation

Die in diesem Buch realisierten Entwürfe wurden mit der Entwicklungs-Software ISE Design Suite der Firma Xilinx erstellt, die als Webpack im Internet kostenlos erhältlich ist. Mit Hilfe der vermittelten Grundlagen und der CAD-Entwicklungs-Software soll der Leser in die Lage versetzt werden, Mikroprozessoren zu entwerfen und für eigene Anwendungen anzupassen. Durch die konsequente Anwendung der strukturierten Entwurfsmethode digitaler Systeme lassen sich auch komplexere Mikroprozessoren entwickeln.

Zusammenfassend werden folgende Themen behandelt:

- Grundlagen des Mikroprozessor-Entwurfs
- VHDL-Entwurf eines 12-Bit-Mikroprozessors
- Simulation und Synthese von VHDL-Modellen

Unterhaching, im Dezember 2017 Dieter Wecker

https://doi.org/10.1515/9783110583069-201

Inhalt

1 Grundlagen

1.1 Einleitung

Das Ziel dieses Buches ist der Entwurf von digitalen Komponenten für Mikroprozessoren. Die Komponenten werden als VHDL-Modelle erstellt und getestet. Es werden mehrere Mikroprozessor-Versionen behandelt, bei denen unterschiedliche VHDL-Modelle für die Bausteine des Prozessors verwendet werden. Dabei werden die Vor- und Nachteile der unterschiedlichen Modelle beim Entwurf sichtbar. Die Synthese-Ergebnisse der verwendeten Entwicklungs-Software und damit die Hardware-Realisierung sind abhängig von den gewählten VHDL-Modellen. Für den Schaltungsentwurf ist es daher wichtig, die geeigneten Modelle zu erstellen. Beim Entwurf digitaler Systeme mit Hilfe der VHDL-Modellierung sind folgende Punkte zu beachten:

- Beschreibung des Modells mit synthetisierbarem VHDL-Code
- Synthese-Berichte analysieren
- Bedingungen für die Hardware festlegen

In der Praxis werden immer komplexere digitale Systeme benötigt in immer kürzeren Entwicklungszeiten. Der Begriff „Time-to-Market“ steht dabei im Vordergrund. Klassische Entwurfsmethoden wie z. B. die graphische Schaltplaneingabe können den Anforderungen nicht mehr gerecht werden. Um diese Bedingungen zu erfüllen, werden leistungsfähige Entwurfssysteme mit einer geeigneten Hardware benötigt. Durch den Einsatz von CAD-Entwicklungs-Software und programmierbaren Logikbausteinen lassen sich diese Anforderungen nahezu erfüllen:

- kurze Entwicklungszeiten
- leichte Änderung des Designs
- Einsatz von IP-Cores
- Frühzeitige Fehlererkennung durch Simulationsmethoden

Die Entwicklungs-Software wird für die Modellierung und für die Umsetzung in die Hardware verwendet. Die Synthese-Tools setzen die VHDL-Modelle in Schaltpläne und Netzlisten um. Die Netzlisten werden für die Umsetzung in die Ziel-Hardware benötigt.

Eine wichtige Rolle für die Beschreibung von Designs spielen die Hardware-Beschreibungssprachen. Für den Entwurf von digitalen Systemen werden häufig die Hardware-Beschreibungssprachen VHDL und Verilog eingesetzt.

Die Hardware-Beschreibungssprache VHDL wurde bereits 1987 als IEEE 1076-87 standardisiert.

Für die vorliegenden Mikroprozessor-Entwürfe werden FPGAs (Field Programmable Gate Array) als Ziel-Hardware verwendet. FPGAs können für hochkomplexe Anwendungen eingesetzt werden. Hier handelt es sich um rekonfigurierbare, d. h. wie-

https://doi.org/10.1515/9783110583069-001

derbeschreibbare FPGA-Technologien. Der FPGA-Entwurf kann somit leicht an veränderte Bedingungen angepasst werden.

Für die Entwürfe kommt die Entwicklungs-Software ISE Design Suite von Xilinx zum Einsatz. Für alle VHDL-Modelle wird hier Standard-VHDL verwendet, so dass man bei den Modellen noch Hersteller-unabhängig ist.

Beim Einsatz von IP-Cores (Intellectual Property) können fertige Komponenten in das eigene Design integriert werden. IP-Cores erleichtern dem Entwickler die Arbeit, er muss nicht für jedes Teildesign die Schaltung selber entwickeln.

Durch den Einsatz von Simulations- und Analyse-Tools der Entwicklungs-Software ist eine frühe Fehlererkennung noch im Entwurfsstadium möglich, es ist eine wichtige Voraussetzung für den Entwurfsprozess.

Beim Entwurf von Mikroprozessoren, die in der Praxis eingesetzt werden, ist es notwendig, die Funktionsfähigkeit des Prototyps als Hardware zu testen. Dazu eignen sich Experimentier-Boards (Demo-Boards), die mit den FPGA-Chips ausgestattet sind, wie sie in der CAD-Entwicklungs-Software verwendet werden. Die Entwürfe des 12-Bit-Mikroprozessors in diesem Buch können auch mit derartigen Experimentier-Boards getestet werden.

1.2 Entwurfsmethoden für digitale Systeme

Im Idealfall könnte der Entwurf komplexer digitaler Systeme wie in Abb. 1.1 aussehen. Am Anfang eines Entwurfs stehen die Systemspezifikationen, d. h. es müssen die Eigenschaften des Systems definiert werden. Mit Hilfe von Entwicklungs-Tools wird ein Modell erstellt, das die Eigenschaften des Systems beschreibt. Dazu können Hardware-Beschreibungssprachen verwendet werden. Für das Layout werden dann Synthese-Tools der Entwicklungs-Software eingesetzt. Dieser Idealfall ist in der Regel für komplexe digitale Systeme nicht möglich, man versucht jedoch, dem idealen Entwurfsverlauf möglichst nahe zu kommen. Dies führt automatisch in die Richtung von formalen Entwurfsmethoden. Ein grober Ansatz sind die beiden folgenden Entwurfsmethoden:

- „Top-down-Entwurf“
- „Bottom-up-Entwurf“

Im ersten Fall geht man von der obersten Entwurfsebene aus. Dabei interessiert man sich nur für die Eigenschaften, die von dem System gefordert werden. Die unteren Ebenen werden zunächst als Black Boxes behandelt, d. h. man betrachtet nur die Funktionen der einzelnen Komponenten mit den zugehörigen Ein- und Ausgängen. Die hierarchische Struktur wird dann immer weiter in Teilkomponenten zerlegt, bis das ganze System nur noch aus Basiskomponenten besteht.

Im zweiten Fall geht man in der Regel von der Ebene der Basiselemente aus, d. h. den logischen Gattern und entwickelt daraus komplexere Komponenten. Man ver-

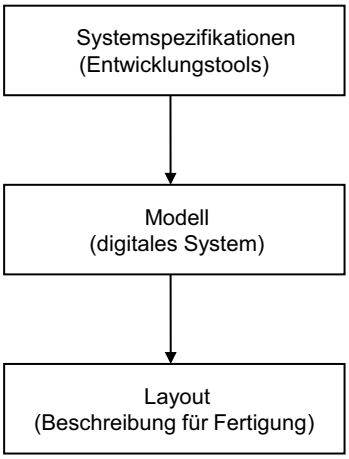


Abb. 1.1: Entwurf von digitalen Systemen.

sucht bei dieser Methode, das System in Teilaufgaben zu zerlegen und dann sukzessiv zum komplexen Gesamtsysten zu kommen. Hier hat man oft den Vorteil, dass bereits geeignete Komponenten aus anderen Entwürfen vorhanden sind, die verwendet werden können. Der Nachteil dieser Methode ist, das der gesamte Entwurf so in Teilaufgaben zerlegt wird, dass das System zu unübersichtlich wird. Diese beiden Entwurfsmethoden „Top-down“ und „Bottom-up“ benötigen viel Erfahrung beim Entwurf von digitalen Systemen und dem Beachten von Entwurfsregeln. Aus diesen Betrachtungen geht hervor, dass man für den Entwurf von komplexen digitalen Systemen formale Ansätze benötigt. Mit den Hardware-Beschreibungssprachen VHDL, Verilog oder anderen geeigneten Hochsprachen wird versucht, mit Hilfe von formalen Ansätzen komplexe Systeme zu beschreiben. Dabei ergeben sich folgende Schwerpunkte:

- Modellierung
- Strukturierung
- Beschreibungsmittel
- Synthese
- Verifikation

Bei der Modellierung können unterschiedliche Beschreibungssprachen für die Beschreibung der Hierarchie-Ebenen und die Strukturierung der Ebenen eingesetzt werden. Die Hierarchie-Ebenen, die auch als Entwurfsebenen bezeichnet werden, sollen den Entwurf in den unterschiedlichen Abstraktionsformen beschreiben. Die Ebenen verlaufen von der Systemspezifikation (Systemebene) bis hin zur Layout-Ebene. Die Layout-Ebene ist beim FPGA-Entwuf die „Place-and-Route“-Ebene. Wichtige Beschreibungsmittel sind:

Netzliste(EDIF) → (VHDL, Verilog) → SystemC → UML

Die Abstraktion nimmt dabei von links nach rechts zu. Auf der untersten Ebene ist ein Netzlistenformat zugeordnet (EDIF: Electronic Design Interchange Format). Auf dieser Ebene befinden sich die logischen Gatter mit den Grundverknüpfungen. Für diese Ebene und die nächst höheren Entwurfsebenen können die Beschreibungssprachen VHDL und Verilog angewendet werden. Der Sprachumfang in VHDL ist größer als in Verilog, es existieren in VHDL z. B. mehr Datentypen für die Beschreibung komplexer Systeme. Für die Beschreibung der Entwurfsebenen kann VHDL bis hinauf zur Systemebene angewendet werden. Eine grobe Einteilung der Entwurfsebenen zeigt die folgende Auflistung:

SK-Ebene → Logikebene → RT-Ebene → Algorith.-Ebene → Systemebene

SK steht für Schaltkreis und RT für Register-Transfer. Die Abstraktion nimmt von links nach rechts zu. Für den FPGA-Entwurf kann man die Logikebene als unterste Ebene ansehen, da die Schaltkreisebene bereits vorstrukturiert ist.

Die als SystemC bezeichnete Beschreibungssprache ist eine Erweiterung der Hochsprachen C und C++. Mit ihr können komplexe digitale Systeme effektiv beschrieben und simuliert werden. Durch den höheren Abstraktionsgrad steigt besonders die Simulationsgeschwindigkeit des Systems. Mit der Komplexität des Systems steigt entsprechend auch die Simulationszeit. Mit den sog. C-Modellen lassen sich Systemspezifikationen effektiver simulieren als z. B. mit VHDL oder Verilog [1, 2]. Mit UML (Unified Modeling Language) lässt sich ein digitales System in abstrakter Form beschreiben. UML ist eine Beschreibungssprache, die komplexe digitale Systeme in strukturierter Form beschreiben kann. Sie ist standardisiert und kann als Beschreibungsmittel auf Systemebene eingesetzt werden. Sie wird auch zunehmend für die Beschreibung und Simulation von SoC(System on Chip)-Entwürfen verwendet. Mit UML versucht man, komplexe digitale Systeme mit formalen Ansätzen zu beschreiben [3, 4].

Hier wird im Folgenden ein Mittelweg gewählt zwischen den Entwürfen „Top-down“ und „Bottom-up“. Man bezeichnet diesen Entwurfsstil auch als „Meet-in-the-Middle“. Die Mitte kann z. B. die RT-Ebene sein, wo sich die beiden Entwurfsmethoden treffen. Der Vorteil dabei ist, dass schon vorhandene und getestete Komponenten für einen neuen Entwurf verwendet werden können. Diese Vorgehensweise geht von einer Strukturierung des Mikroprozessor-Systems aus in Komponenten und Subsysteme. Für die Modellierung des Mikroprozessor-Systems wird wie angekündigt die Hardware-Beschreibungssprache VHDL verwendet. Bei der Modellierung können unterschiedliche VHDL-Modelle erstellt werden. Dabei unterscheidet man zwischen der Structural- und der Behavioral-Methode. Bei der Structural-Methode verwendet man Strukturbeschreibungen in Form von Komponenten. Die Methode ist Hardware-orientiert. Die Behavioral-Methode ist eine Verhaltensbeschreibung, hier steht die Funktion des digitalen Systems im Vordergrund. Die verwendeten VHDL-Strukturen haben

auch unterschiedliche Synthese-Ergebnisse zur Folge, d. h. man bekommt auch unterschiedliche Hardware-Ergebnisse [5, 6].

1.3 Definition der Schnittstellen für die Subsysteme

Komplexe digitale Systeme lassen sich in der Regel in die folgenden Subsysteme aufteilen:

- Ein- und Ausgabe-Einheiten
- Operationswerk
- Steuerwerk

In Abb. 1.2 ist das Blockdiagramm mit den Schnittstellen für die Subsysteme dargestellt. Sie können auch als Komponenten des digitalen Systems betrachtet werden. Die Spezifikation der Schnittstellen für die Komponenten wird durch eine externe und interne Schnittstelle definiert. Die externe Schnittstelle ist für das Ein- und Ausgabeprotokoll zuständig, die interne Schnittstelle für die Kommunikation zwischen Operationswerk und Steuerwerk. Der Datenaustausch zwischen den Komponenten wird mit Hilfe von Daten- und Steuerleitungen realisiert.

Das digitale System hat die interne Schnittstelle (S, A) für den Datenaustausch zwischen dem Operationswerk und dem Steuerwerk sowie die externe Schnittstelle (X, Y) für den Datenaustausch über die Ein- und Ausgabeeinheit. Der Eingangsvektor X ist hier vereinfacht dargestellt. Er stellt sowohl den externen Datentransfer über Inputregister als auch den Datentransfer über eine Speichereinheit dar. Der Ausgangs-

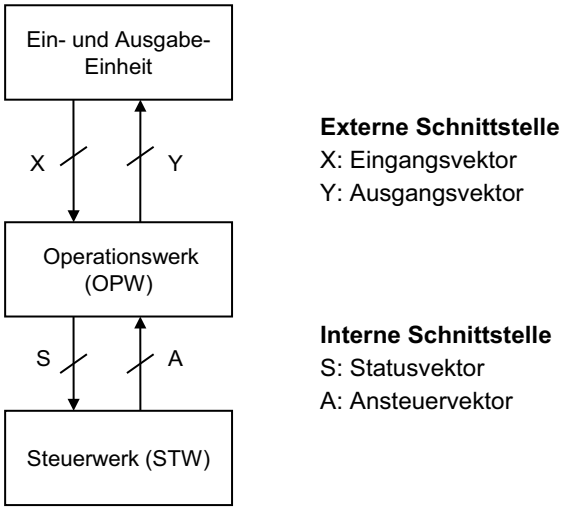


Abb. 1.2: Schnittstellen der Subsysteme.

vektor Y stellt ebenfalls den Datentransfer in ein Output-Register oder in eine Speichereinheit dar. Die Ein- und Ausgabe-Einheit besteht somit aus Ein- und Ausgaberegistern und einer Speichereinheit. Die Speichereinheit wird für die Daten und den Programm-Code benötigt.

Das Steuerwerk ist für den Programmablauf bzw. den Steuerungsalgorithmus zuständig, wobei das Operationswerk die Steuerinformation über den Ansteuervektor A bekommt. Das Steuerwerk generiert die Steuersignale des Ansteuervektors A und steuert den zeitlichen Ablauf der Mikrooperationen im Operationswerk.

Das Operationswerk führt die einzelnen Operationen aus und meldet dem Steuerwerk den jeweiligen Status über den Statusvektor S. Durch die Aufteilung des Mikroprozessors in die Komponenten Operationswerk und Steuerwerk besteht die Möglichkeit, die Komponenten getrennt zu behandeln. Das Steuerwerk kann formal nach einem Automatenmodell entworfen werden. Der Ansteuervektor liefert eine eindeutige Zuordnung für die Funktionen im Operationswerk, die vom Steuerwerk kontrolliert werden. Die interne Schnittstelle zwischen dem Operationswerk und dem Steuerwerk wird damit zum fundamentalen Bestandteil des Mikroprozessors.

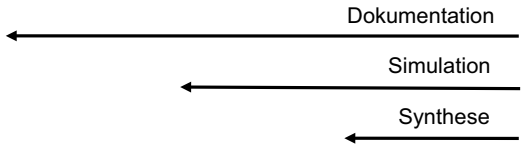
Sowohl das Steuerwerk als auch das Operationswerk werden für derartige Systeme immer als getaktete Automaten aufgebaut. Diese Strukturierung des digitalen Systems hat folgende Vorteile:

- Änderungen und Erweiterungen können leichter durchgeführt werden.
- Das Steuerwerk kann formal nach einem Automatenmodell erstellt werden.
- Durch die Aufgabenteilung können Fehler leichter lokalisiert werden.

Im Kap. 2.2.1.1 wird die Schnittstelle zwischen Operationswerk und Steuerwerk ausführlich behandelt.

1.4 Simulation und Synthese mit VHDL

VHDL wurde ursprünglich für die Dokumentation und Simulation von komplexen digitalen Systemen entwickelt. Die Synthesefähigkeit der VHDL-Konstrukte wurde erst später gefordert. Der Sprachumfang von VHDL teilt sich etwa in der folgenden Form auf:



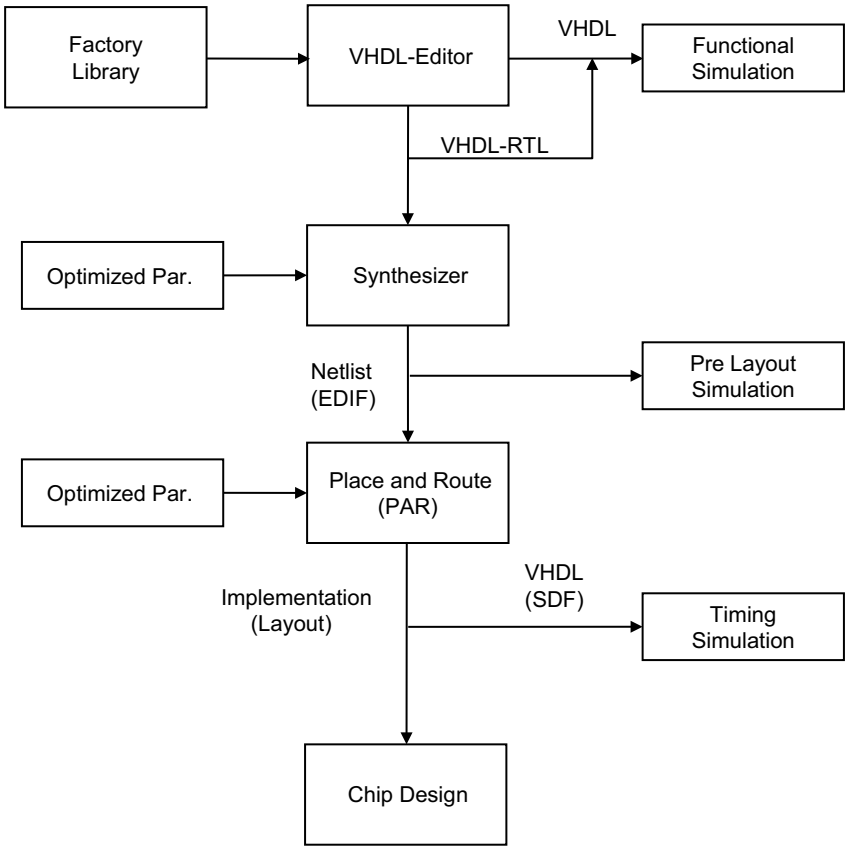


Abb. 1.3: Synthese und Simulation; EDIF: Electronic Design Interchange Format, SDF: Standard Delay Format, RTL: Register Transfer Level.

Daraus folgt, dass die meisten VHDL-Konstrukte simulierbar sind, aber nicht unbedingt synthetisierbar. Die Modellierung mit VHDL wird auf allen Abstraktionsebenen beim hierarchischen Entwurf unterstützt, von der Logikebene bis hinauf zur Systemebene [5, 7].

In Abb. 1.3 ist die Einordnung des Synthese-Tools für ein FPGA-Design dargestellt. Mit dem VHDL-Editor können zunächst alle VHDL-Konstrukte verarbeitet werden. Dabei hat man i. a. auch die Möglichkeit, VHDL-Module aus einer Herstellerbibliothek zu verwenden.

Der vom Compiler akzeptierte VHDL-Code kann dann in einer funktionalen Simulation getestet werden. Dabei werden nur die Logikfunktionen getestet, unabhängig von der Synthetisierbarkeit der Logik. Das Synthese-Tool akzeptiert nur einen synthetisierbaren VHDL-Code (VHDL-RTL). Die synthetisierte Netzliste dient nach der Synthese als Input für das „Place-and-Route“-Tool. Hier wird in der Regel das Netzlisten-

format EDIF (Electronic Design Interchange Format) verwendet. Die Synthese-Tools werden häufig von den Halbleiterherstellern mitgeliefert und sind damit nicht mehr Hardware-unabhängig.

Optimierungsbedingungen können bei der Synthese mit eingegeben werden. Es können meistens auch VHDL-Module aus denBibliotheken der Entwurfs-Software mit eingebunden werden. Die Timing Simulation kann erst durchgeführt werden, wenn das „Place-and-Route“ (PAR)-Tool eine VHDL-Netzliste mit den berechneten Verzögerungszeiten erzeugt hat (SDF-Datei). Bei der Pre-Layout-Simulation wird die synthetisierte Netzliste mit Verzögerungszeiten für die logischen Gatter verknüpft. Hier kann bereits entschieden werden, ob die geforderten Zeitabhängigkeiten für das Design eingehalten werden können [8].

Synthesefähiger VHDL-Code

Der Umgang mit VHDL hat in der Regel das Ziel, eine digitale Schaltung zu realisieren. Damit das VHDL-Design sinnvoll ist, sollte der Benutzer sowohl die Ziel-Hardware als auch die wichtigsten Regeln für die Synthetisierbarkeit des VHDL-Codes kennen.

Durch das Setzen von Optimierungsparametern kann die Steuerung des Syntheseprozesses bezüglich Chipfläche und Signallaufzeiten beeinflusst werden. Die Verwendung von Herstellerbibliotheken ist i. a. wichtig, um die Ziel-Hardware optimal ausnutzen zu können. Durch diese Abhängigkeiten der Entwurfswerkzeuge und der Ziel-Hardware können hier nur allgemeine Regeln für einen synthetisierbaren VHDL-Code angegeben werden:

- VHDL-Beschreibungen auf RT-Ebene sind synthetisierbar
- Verzögerungszeiten bei Signalzuweisungen sind nicht synthetisierbar
- Physikalische und Datei-Datentypen werden bei der Synthese nicht unterstützt
- **assert-**Anweisungen werden vom Synthese-Tool ignoriert
- direkte Signalzuweisungen werden in Schaltnetze umgesetzt
- Signalzuweisungen, die an Bedingungen gekoppelt sind, werden in Schaltwerke umgesetzt

Auf der RT-Ebene lassen sich meistens Strukturbeschreibungen von Komponenten ohne Probleme synthetisieren, da sie in ein festes Taktschema eingebunden sind. Bei Signalzuweisungen, die z. B. mit den Ausdrücken **after** oder **wait for** verknüpft sind, werden Zeitangaben gemacht, die nicht synthetisierbar sind. Physikalische Datentypen sind nicht synthetisierbar, da sie mit einer Maßeinheit verknüpft sind, z. B. die Maßeinheit time, die vom Synthese-Tool nicht verarbeitet werden kann. Auch Dateien sind aus verständlichen Gründen nicht synthesefähig, da sie eine Ansammlung von Textdaten enthalten, die für die Ein- und Ausgabe von Daten bei der Simulation verwendet werden können.

Die **assert**-Anweisungen werden verwendet, um bei der Simulation Meldungen bei der Abarbeitung des VHDL-Codes auszugeben. Hierfür existiert ein vordefinierter

Aufzählungstyp im **package** standard in der Bibliothek std. Er enthält die Elemente: note, warning, error, failure.

Vom Synthese-Tool wird die **assert**-Anweisung ignoriert. Direkte Signalzuweisungen werden in Schaltnetze synthetisiert, wenn einem Signal oder einer Variablen in allen Fällen ein Wert zugewiesen wird.

Sind Signalzuweisungen an Bedingungen gekoppelt, bei denen Werte zwischengespeichert werden müssen, so werden Schaltwerke synthetisiert.

Es erfolgt auch eine Umsetzung in Schaltwerke, wenn das Ausgangssignal auf der rechten Seite der Signalzuweisung steht. Das Gleiche passiert, wenn nicht in allen Abfragen ein Wert zugewiesen wird [9].

2 Das 12-Bit-Mikroprozessor-System (MPU12_S)

Die Anforderungen für das zu entwickelnde Mikroprozessor-System müssen noch festgelegt werden. Als erstes wird die Vorgabe gemacht, das System in die Komponenten Operationswerk, Steuerwerk und Speicher zu strukturieren. Die internen und externen Schnittstellen zwischen den Komponenten wurden bereits in Kap. 1.3 eingeführt. Dabei geht es um folgende Protokolle:

- externe Ein- und Ausgabe über Input- und Output-Register
- Datenaustausch zwischen Operationswerk und Speicher
- interner Datenaustausch zwischen Operationswerk und Steuerwerk

Die konkreten Beschreibungen für die Schnittstellen mit Signalen und den zeitlichen Abläufen müssen noch genauer definiert werden. Das Mikroprozessor-System hat zunächst die vereinfachte Form nach Abb. 2.1:

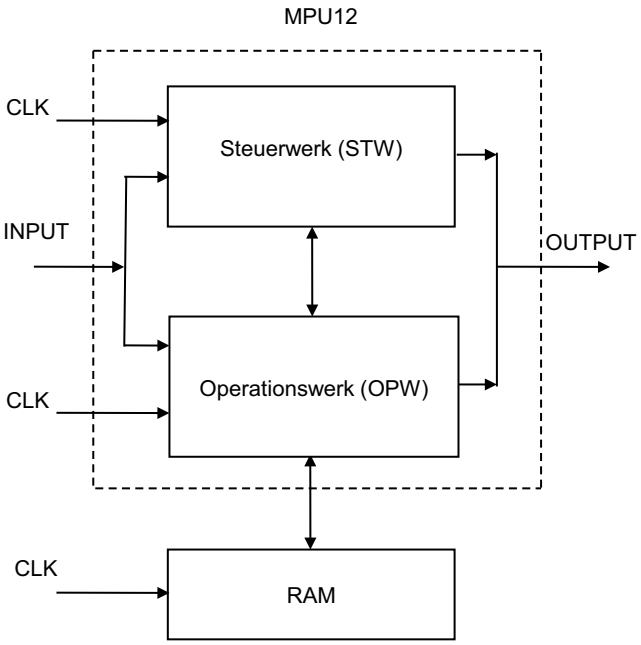


Abb. 2.1: Mikroprozessor-System (MPU12_S).

Als Arbeitsspeicher wird ein synchroner statischer RAM-Speicher verwendet. Das Lesen der Daten aus dem Speicher soll asynchron und das Speichern von Daten synchron, d. h. getaktet erfolgen. Die drei Komponenten OPW, STW und RAM sind formal getaktete Automaten und haben eigene Takteingänge. Da die Befehlsfolge im Mikro-

https://doi.org/10.1515/9783110583069-002

prozessor-System sequenziell abläuft, muss in der Regel die Taktfolge für die Komponenten beachtet werden. Auf diese Punkte wird bei der Simulation des Systems noch ausführlich eingegangen.

2.1 Entwurf eines 12-Bit-Mikroprozessors

Am Anfang eines Entwurfs steht immer eine Anforderungsliste. Es muss zunächst ermittelt werden, welche Spezifikationen das digitale System erfüllen soll. Es wurden bereits allgemeine Anforderungen an das Mikroprozessor-System, wie z. B. die Strukturierung der CPU in ein Operationswerk und ein Steuerwerk gemacht. Ein wichtiges Kriterium ist dabei auch der Punkt, dass der Entwurf, d. h. die Eigenschaften des Systems leicht verändert werden können. Es soll eine 12-Bit-CPU entworfen werden, die als MPU12 (Mikro-Prozessor-Unit) bezeichnet wird. Dabei werden folgende Kriterien festgelegt:

- strukturierter Entwurf für das Mikroprozessor-System
- es existiert nur ein RAM-Speicher für Programme und Daten
- einfacher Befehlssatz mit arithmetischen und logischen Befehlen
- Adressierung direkt und indirekt
- Strukturierung der CPU in Operationswerk (OPW) und Steuerwerk (STW)
- das Ein- und Ausgabeprotokoll ist asynchron
- Befehlsformat, Daten- und Adressformat sind einheitlich
- Befehlsphasen

Die allgemeinen Anforderungen an den Entwurf sind damit festgelegt. Es müssen noch die konkreten Spezifikationen für den Entwurf der MPU12 bestimmt werden:

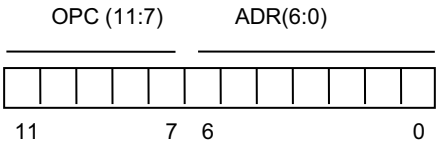
- Befehlssatz
- Befehlsformat
- Adressierung
- Registerstruktur
- Akkumulatorstruktur
- Befehlsphasen (ca. 5 CPU-Takte pro Befehl)
- Ein- und Ausgabeprotokoll

Der Befehlssatz der MPU12 ist in Tab. 2.1 zusammengestellt. In der ersten Spalte steht der 5-Bit-Opcode OPC(4:0) für die Befehle. Es können insgesamt 32 Befehle definiert werden, davon sind 28 Befehle zugeordnet, die 4 restlichen werden als „No Operation“ (NOP) behandelt. Das niederwertige Bit des 5-Bit-Opcodes gibt den Adressierungsmodus an:

OPC(0) = 0 : direkte Adressierung
OPC(0) = 1 : indirekte Adressierung

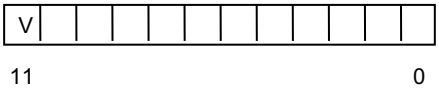
Die nächsten zwei Spalten geben die Kürzel (Mnemonics) für die einzelnen Befehle sowie ihre Bedeutung an. In den letzten Spalten sind die drei Flags und ihre Gültigkeit angegeben. Die Befehle mit einem angehängten ‚I' sind für die indirekte Adressierung.

Befehlsformat (12 Bit)



Operationscode OPC 5 Bit, direkte Adressierung 7 Bit

Datenformat (12 Bit)



Es sollen vorzeichenbehaftete Zahlen verwendet werden. Das oberste Bit (MSB) ist für das Vorzeichen reserviert. Der Datenbereich ist 11 Bit breit.

Abkürzungen in Tab. 2.1

OPC	: Opcode für Befehl
m	: 7-Bit-Adresse
M(m)	: 12-Bit-Operand von Adresse m
M(M(m))	: 12-Bit-Operand von Adresse M(m)
OPR	: 12-Bit-Output-Register
IPR	: 12-Bit-Input-Register
ACR	: 12-Bit-Akkumulator-Register
PC	: 12-Bit-Program-Counter
STA	: 12-Bit-Register-Stack
Z	: Zero-Flag
S	: Vorzeichen-Flag
C	: Carry-Flag
Ci	: Input-Carry
x	: Flag gültig

Tab. 2.1: Befehlssatz der MPU12.

OPC(4:0)	Mnemonic	Bedeutung	Z	S	C
00000	OU m	OPR ← M(m)			
00001	OUI M(m)	OPR ← M(M(m))			
00010	ST m	M(m) ← ACR			
00011	STI M(m)	M(M(m)) ← ACR			
00100	IN m	M(m) ← IPR			
00101	INI M(m)	M(M(m)) ← IPR			
00110	SP	Programmende			
00111	NOP	PC ← PC +1			
01000	JZ m	Z = 1: PC ← m	x		
01001	JZI M(m)	Z = 1: PC ← M(m)	x		
01010	JS m	S = 1: PC ← m		x	
01011	JSI M(m)	S = 1: PC ← M(m)		x	
01100	JC m	C = 1: PC ← m			x
01101	JCI M(m)	C = 1: PC ← M(m)			x
01110	JU m	PC ← m			
01111	JUI M(m)	PC ← M(m)			
10000	CA m	STA ← PC, PC ← m			
10001	CAI M(m)	STA ← PC, PC ← M(m)			
10010	RT	PC ← STA			
10011	NOP	PC ← PC +1			
10100	SHR	ACR ← SHR (ACR)	x	x	
10101	NOP	PC ← PC +1			
10110	SHL	ACR ← SHL (ACR)	x	x	
10111	NOP	PC ACR ← PC +1			
11000	AD m	ACR ← ACR + M(m) + Ci	x	x	x
11001	ADI M(m)	ACR ← ACR + M(M(m)) + Ci	x	x	x
11010	SU m	ACR ← ACR – M(m) – Ci	x	x	x
11011	SUI M(m)	ACR ← ACR – M(M(m)) – Ci	x	x	x
11100	NA m	ACR ← NA (ACR, M(m))	x	x	
11101	NAI M(m)	ACR ← NA (ACR, M(M(m)))	x	x	
11110	LO m	ACR ← M(m)	x	x	
11111	LOI M(m)	ACR ← M(M(m))	x	x	

Mnemonics in Tab. 2.1 (direkte/indirekte Adress.)

OU/OUI : Output
ST/STI : Store
IN/INI : Input
SP : Stop
NOP : No Operation
JZ/JZI : Jump if Z = 1
JS/JSI : Jump if S = 1
JC/JCI : Jump if C = 1
JU/JUI : Jump