1 Einleitung

Software-Entwicklung und Schaltungsentwurf

Software-Entwicklung für Mikroprozessoren und Schaltungsentwurf für FPGAs[1]! Zwei Technologien zur Programmierung der beiden wichtigsten Rechenmaschinen haben sich in der Vergangenheit weitgehend unabhängig voneinander entwickelt und werden jeweils von zwei verschiedenen Anwendergruppen genutzt: den Informatikern, Software-Entwicklern und Programmierern einerseits und den Elektrotechnikern und Hardware-Entwicklern andererseits. Viele Hochschulen lehren beide Disziplinen getrennt – in unterschiedlichen Studiengängen, in verschiedenen Vorlesungen. Oft setzen Unternehmen entweder ganz auf die eine oder ganz auf die andere Technologie oder weisen beide Technologien jeweils unterschiedlichen, spezialisierten Abteilungen zu. Während viele Computermagazine und Software-Bücher nur den Mikroprozessor zu kennen scheinen, sucht man in der Literatur, die sich der FPGA-Thematik widmet, vergeblich nach modernen Software-Konzepten, die der Lösung von komplexen Aufgaben auf einer angemessenen Abstraktionsstufe dienen.

Hardware-unabhängige Entwicklung

Dies mag überraschen! Software-Entwicklung *und* Schaltungsentwurf zielen darauf ab, eine vorgegebene Aufgabe zu lösen: Algorithmen zu finden, in einer Programmiersprache zu beschreiben und sie auf einer Hardware auszuführen. In dem einen Fall ist die Programmiersprache z.B. C++ und im anderen Fall VHDL[2]. Ein wichtiger Anteil des Entwicklungsprozesses ist weitgehend unabhängig davon, ob die Aufgabe durch Mikroprozessoren oder FPGAs gelöst wird: die Analyse und Verwaltung der Anforderungen, der Entwurf der Software-Architektur, das Testen der Lösung und das der Entwicklung zugrunde liegende Vorgehensmodell. Auch der Herstellungsprozess der beiden Rechenmaschinen ähnelt sich. Mikroprozessoren und FPGAs werden beide als integrierte Schaltungen (IC[3]) auf Silizium-Halbleiterbasis hergestellt.

Software ist immateriell

Die Teilung der Anwender in zwei Gruppen liegt eher an der unterschiedlichen Entwicklung der beiden Technologien während der letzten Jahrzehnte. Die Software-Pioniere programmierten Mitte des 20. Jahrhunderts Mikroprozessoren in einer Maschinensprache und schufen damit – ohne die Rechenmaschine physikalisch zu verändern[4] – ein nicht gegenständliches Gut: die Software. Anfangs war die Software noch stark von der Hardware abhängig (Maschinensprache!), im Laufe der Zeit entstanden aber immer neue Verfahren, um die Abhängigkeit der Software von der Hardware zu reduzieren: Assemblersprache, höhere Programmiersprachen wie C, virtuelle Java-Maschine oder modellgetriebene automatische Codegenerierung. Diese Verlagerung des Schwerpunktes von der Orientierung an der Hardware hin zur Software ist in Abb. 1.1 durch den untersten Pfeil angedeutet.

steigender Abstraktionsgrad der Software-Entwicklung

Der Schaltungsentwurf ist aus der Hardware-Entwicklung, der Entwicklung von ma-

[1] FPGA = **F**ield **P**rogrammable **G**ate **A**rray

[2] VHDL (**V**HSIC **H**ardware Description Language), VHSIC (**V**ery **H**igh **S**peed **I**ntegrated Circuit)

[3] IC = **I**ntegrated **C**ircuit *(deutsch: Integrierter Schaltkreis)*

[4] Zumindest nicht auf atomarer Ebene! Die Elektronenkonfiguration ändert sich durch die Programmierung natürlich schon!

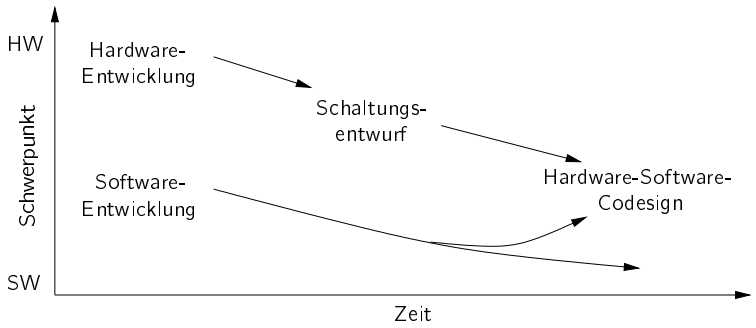

Abbildung 1.1: Historie der Technologien Software-Entwurf für Mikroprozessoren und Schaltungsentwurf für FPGAs: Während die Software-Entwicklung Mitte des 20. Jahrhunderts noch sehr hardware-lastig war, hat sich der Schwerpunkt im Laufe der Zeit immer mehr von der Hardware (HW) hin zur Software (SW) verschoben. Der Schaltungsentwurf für FPGAs ist aus der Hardware-Entwicklung entstanden und hat sich dadurch zunehmend in Richtung Software verlagert. Schaltungsentwurf und Software-Entwicklung verschmelzen heute zu Hardware-Software-Codesign.

Hardware ist materiell

teriellen Bausteinen entstanden. Algorithmen, Berechnungsvorschriften zur Lösung einer Aufgabe, werden in einer digitalen Schaltung abgebildet. Anfangs wurden die Schaltungen dem Baustein noch fest aufgeprägt, indem Leiterbahnen auf atomarer Ebene angelegt wurden – ein praktisch irreversibler Prozess. Wollte man die Berechnungsvorschriften ändern, musste man einen neuen Baustein herstellen. Mit dem Aufkommen der programmierbaren Logikbausteine, PLDs[5], hat sich dies geändert. Die Funktion der Rechenmaschine wurde jetzt nicht mehr über die Anordnung der Atome im Baustein definiert, sondern durch die Konfiguration der Elektronen. Und die ist leicht zu ändern! Auf ein und demselben Baustein können nacheinander viele verschiedene Funktionen programmiert werden. Der Fokus hat sich vom materiellen Prozess der Herstellung eines Stücks Hardware auf den nicht materiellen Prozess des Entwurfs

Schaltungsentwurf ist immateriell

einer Schaltung verschoben. Diese Verlagerung ist in Abb. 1.1 durch den Pfeil von „Hardware-Entwicklung“ hin zu „Schaltungsentwurf“ angedeutet.

Um eine Schaltung zu testen, muss nun kein Hardware-Baustein mehr angefertigt werden – ein relativ langwieriger und teurer Prozess! Die Schaltung braucht jetzt für den Test nur auf ein bereits vorhandenes FPGA geladen zu werden. Das verkürzt die Entwicklungszyklen drastisch und erlaubt es, sich auf diejenigen Aktivitäten zu konzen-

komplexe Schaltungen

trieren, die eine Entwicklung komplexer Systeme ermöglichen: Anforderungsanalyse, Modellieren und Testen! Diese Aktivitäten wurden in den letzten Jahrzehnten auf breiter Ebene untersucht. Getrieben von der Software-Entwicklung für Mikroprozessoren hängen sie aber streng genommen nicht von der Rechenmaschine ab, auf dem die Software schließlich ausgeführt wird! Warum sollten also diese bewährten Methoden

[5] PLD = **P**rogrammable **L**ogic **D**evice *(deutsch: Programmierbarer Logikbaustein)*

nicht auch für den Schaltungsentwurf nützlich sein?

Nicht nur die Vertreter des historischen „Hardware-Lagers“ können von diesem Umstand profitieren. Auch die Software-Entwickler, die bisher mehr oder weniger bewusst Mikroprozessoren als Zielplattform vor Augen hatten, gewinnen jetzt eine alternative Rechenmaschine, die für bestimmte Anwendungsklassen dem Mikroprozessor überlegen ist:

Beschleunigung durch FPGAs

- parallelisierbare Anwendungen: Unter Umständen lässt sich ein parallelisierbarer Algorithmus kostengünstiger auf einem FPGA ausführen als auf einem Mikroprozessor-Cluster.

- signalverarbeitungsintensive Anwendungen: Auf einem FPGA integrierte DSP[6]-Funktionsblöcke bieten z.B. schnelle MAC[7]-Einheiten an, die unter anderem für Vektoroperationen genutzt werden können.

- nicht sättigbare Anwendungen: Im Gegensatz zu einem Mikroprozessor, dessen Verarbeitung oft durch externe Ereignisse gesteuert wird (Interrupts), verarbeitet ein FPGA die Daten in jedem Takt gleich.

Das bedeutet nicht, die *gesamte* Software soll auf FPGAs ausgeführt werden! Serielle Algorithmen laufen nach wie vor besser auf einem seriell arbeitenden Mikroprozessor, der in der Regel höher getaktet ist als ein technologisch vergleichbares FPGA, und es werden nur die Software-Teile auf FPGAs ausgelagert, für die sich der höhere Implementierungsaufwand lohnt.

Eine engere Koppelung der beiden Strömungen („Software“ und „Hardware“) aus Abbildung 1.1 würde offenbar beide Lager bereichern. Diese Koppelung ist in Abb. 1.1 durch eine Verschmelzung der beiden Pfade zu einer Disziplin symbolisiert, die unter dem Schlagwort „Hardware-Software-Codesign“ bekannt ist. Leider ist dieser Begriff etwas irreführend, denn er suggeriert einen gemeinsamen Entwurf (Codesign) von Hardware und Software. Hardware wird jedoch nicht entworfen und dann aus diesem Entwurf ein Baustein hergestellt, sondern es wird eine vorhandene Hardware programmiert. Gemeinsam entworfen wird vielmehr eine digitale Schaltung (für FPGAs) und Software (für Mikroprozessoren) – also in beiden Fällen nichts Gegenständliches, sondern eine Software! Diese Software läuft auf FPGAs, Mikroprozessoren oder gemischten Mikroprozessor-FPGA-Systemen. Beim Entwerfen eines Hardwaresystems aus einzelnen Hardwarebausteinen könnte man wieder von Hardware-Design im wörtlichen Sinn sprechen und bei einem daran gekoppelten Entwurf der Software von Hardware-Software-Codesign!

Hardware-Software-Codesign

Das Buch ist folgendermaßen aufgebaut:

Kapitel 1 führt in die Thematik des Buches ein.

Kapitel 2 stellt einen wichtigen Einsatzbereich der Rechenmaschinen „Mikroprozessor“ und „FPGA“ vor: eingebettete Systeme.

Kapitel 3 beschreibt die beiden Rechenmaschinen „Mikroprozessor“ und „FPGA“.

[6] DSP = **D**igital **S**ignal **P**rocessing *(deutsch: Digitale Signalverarbeitung)*
[7] MAC = **M**ultiplication-**Ac**cumulation *(deutsch: Multiplikation-Akkumulation)*

Kapitel 4 bietet einen Einblick in den Stand der Software-Entwicklung für Mikroprozessoren. Hierbei beschränken wir uns auf die grundlegenden Konzepte, die das Wesen der modernen Software-Entwicklung charakterisieren: Anforderungsanalyse, Software-Entwurf, Testen und Vorgehensmodelle.

Kapitel 5 liefert einen Überblick über den Stand des Schaltungsentwurfs für FPGAs.

Kapitel 6 stellt Software-Entwicklung und Schaltungsentwurf gegenüber.

Kapitel 7 führt hybride Rechenarchitekturen ein.

Kapitel 8 stellt Werkzeuge zum Entwurf auf Systemebene, zur Simulation und zur automatischen Codegenerierung für Mikroprozessoren und FPGAs vor.

Kapitel 9 beschreibt einen UML-basierten Ansatz zur Software-Entwicklung für hybride Systeme.