FPGAs und Mikrocontroller - Ein Vergleich

Proseminar Microcontroller und eingebettete Systeme WS2014/2015

Simon Rehwald
Lehrstuhl für Echtzeitsysteme und Robotik
Fakultät für Informatik
Technische Universität München
Email: simon.rehwald@in.tum.de

Kurzfassung

Diese Arbeit beschäftigt sich mit FPGAs und Mikrocontrollern, wobei schwerpunktmäßig FPGAs betrachtet werden. Zunächst wird der grundlegende Aufbau und die Architektur eines FPGAs näher erläutert. Ergänzend dazu wird auf mögliche Zusatzkomponenten eingegangen. Anschließend wird die Funktionsweise mit LUTs und Multiplexern behandelt, wobei zur Veranschaulichung die jeweilige Implementierungsmöglichkeit mit einem Beispiel näher erklärt wird. Darauf erfolgt eine Auseinandersetzung mit der Entwicklung eines FPGAs. Abschließend werden mögliche Einsatzgebiete betrachtet. Nach jedem dieser Punkte wird das FPGA mit einem Mikrocontroller entsprechend verglichen bzw. gegenübergestellt. Ziel dieser Arbeit ist es, dem Leser grundlegende Aspekte eines FPGAs näher zu bringen sowie Unterschiede zu einem Mikrocontroller deutlich zu machen.

Schlüsselworte

FPGA, Mikrocontroller, LUT, Multiplexer, Logik

INHALTSVERZEICHNIS

I. Einführung

FPGA steht für *Field Programmable Gate Array*. *Gate Arrays* sind dabei vorproduzierte Logikbausteine, welche aus einer Anordung von Transistoren bestehen. Lediglich die Verdrahtung erfolgt kundenspezifisch beim Hersteller. *Field Programmable*, zu deutsch wörtlich *feldprogrammierbar*, soll nun verdeutlichen, dass ein FPGA eben nicht beim Hersteller seine Funktion erhält, sondern beim Kunden, sprich "im Feld". Diese Entwicklung der Logikbausteine hin zu FPGAs begann Ende 1970. Gerade der Wunsch nach ähnlichen Bausteinen wie den beschriebenen Gate Arrays, welche jedoch vom Anwender bzw. Kunden programmiert werden, führte zur Suche nach entsprechenden Lösungen. Schließlich wurde 1985 das erste FPGA auf den Markt gebracht. [1, S.214] [2, S.1f.] [3, S.208]

II. Aufbau eines FPGAs

A. Grundelemente

1) Basiszellen (CLBs):

Grundsätzlich besteht ein FPGA hauptsächlich aus in einer bestimmten Art und Weise angeordneten Configurable Logic Blocks (CLB), welche als Basiszellen bezeichnet werden. In diese Logikblöcke wird später die Funktionalität des FPGAs implementiert werden. Dafür gibt es zwei verbreitete Konzepte, nämlich zum einen sogenannte Lookup Tables (LUTs), zum anderen Multiplexer. Wann und wie beide Konzepte verwendet werden, wird bei Betrachtung der Funktionsweise eines FPGAs näher erläutert.

2) Architektur und Verdrahtung/Verbindung:

Für die Art und Weise, wie die Basiszellen angeordnet sind gibt es nun ebenfalls zwei grundlegende Möglichkeiten. Heute am meisten verbreitet ist das symmetrische Array. Bei dieser Architektur sind die CLBs matrixförmig auf dem Chip angeordnet und zwischen ihnen verlaufen horizontal sowie vertikal Verbindungsleitungen. Dabei gibt es Leitungen unterschiedlicher Länge, sodass sowohl direkte Verbindungen zwischen benachbarten, als auch Verbindungen zu entfernteren Logikblöcken möglich sind. Abb. 1 zeigt den schematischen Aufbau eines FPGAs mit dieser Architektur.
Eine weitere, jedoch aufgrund ihrer geringeren Leistungsfähigkeit nicht mehr häufig verwendete Architektur ist die kanalorientierte Struktur. Hierbei sind die Basiszellen nicht matrixförmig, sondern reihenartig auf dem Chip angeordnet. Zwischen diesen horizontalen Reihen verlaufen die Verbindungselemente, welche ebenfalls in ihrer Länge variieren, um lokale und weiterreichende Verbindungen zu ermöglichen.
Beiden Prinzipien ist allerdings gemeinsam, dass vom Anwender selbst entschieden werden kann, welche Logikblöcke miteinander verdrahtet sind, da es sich um programmierbare Verbindungen handelt.

3) Input/Output:

Um nun sowohl mit entsprechenden Eingangssignalen versorgt werden zu können, als auch verarbeitete Signale ausgeben zu können, befinden sich am Rand eines jeden FPGAs eine Vielzahl von Input/Output-Blöcken (I/O-Blöcke), welche je nach Konfiguration als Eingang, Ausgang oder Ein- und Ausgang verwendet werden können. Die I/O-Pins, welche jeweils hinter den I/O-Blöcken angebracht sind, ermöglichen die Platzierung auf einer Leiterplatine.

4) Takterzeugung und -aufbereitung:

Daneben sind üblicherweise auch mehrere Takterzeuger und -aufbereiter in einem FPGA verbaut, um der großen räumlichen Verteilung im FPGA entgegen zu wirken. Diese Bausteine versorgen Teile des FPGAs mit unterschiedlichen Takten, wodurch jedoch im Gesamten wieder Synchronität erreicht wird. Um eine, wenn auch sehr geringe, aber unvermeidliche Asynchronität möglichst klein zu halten, werden die Takte über spezielle Taktleitungen auf dem FPGA verteilt.

[1, S.205f.] [4, S.79] [3, S.208f., 215ff.] [5, S.41ff.] [6, S.29]

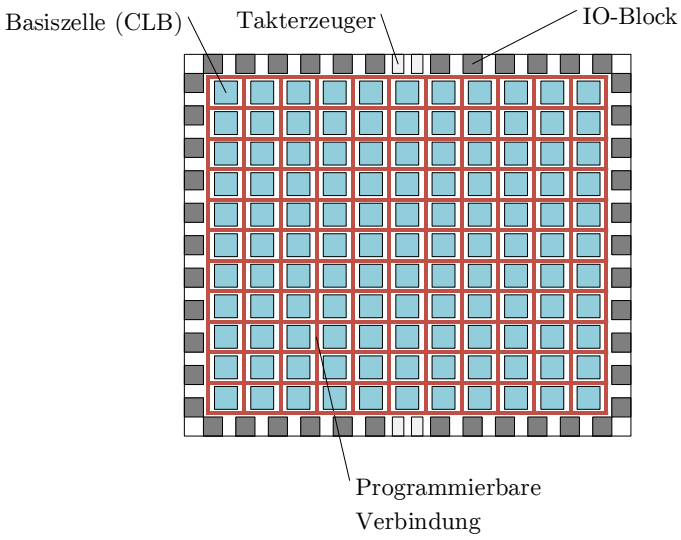


Abb. 1: Aufbau eines FPGA (i.A.a. [3, S.209])

B. Zusatzkomponenten

Neben den genannten Basiskomponenten besitzen FPGAs oft noch weitere Zusatzkomponenten. Dazu gehören beispielsweise dedizierte RAM-Blöcke. Darüber hinaus sind in vielen FPGAs Multiplizierer verbaut, welche eine schnellere Multiplikation zweier Binärzahlen ermöglichen, als es eine Realisierung mit den Logikblöcken könnte. Weiterhin ist es auch möglich, Prozessorkerne in ein FPGA zu integrieren. Dies kann zum einen mit *Soft Cores* geschehen, wobei hier der Prozessor mithilfe der Logikblöcke implementiert wird. In diesem Fall handelt es sich also um keine Hardware-Zusatzkomponente. Zum anderen können auch *Hard Cores* verbaut werden. Bei letzteren handelt es sich im Gegensatz zu den Soft Cores um tatsächliche Hardware, welche entsprechend im FPGA integriert ist. [4, S.79] [7]

C. Vergleich: Aufbau eines Mikrocontrollers

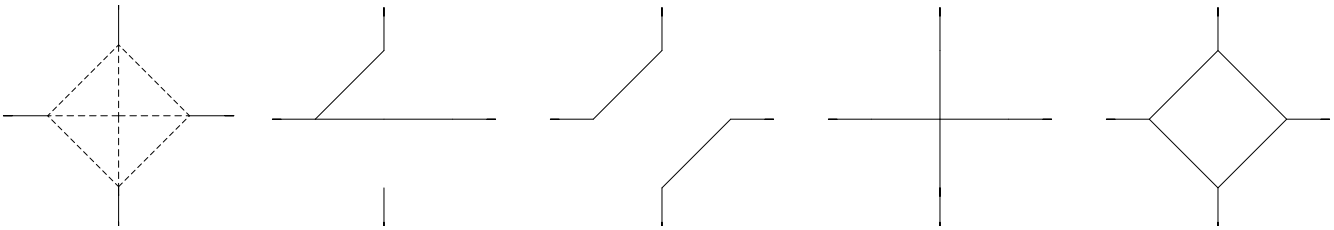
Ein Mikrocontroller besteht in der Regel aus einem zentralen Prozessorkern sowie weiteren Peripheriekomponenten. Dazu gehören Ein- und Ausgabeeinheiten, Programm- und Arbeitsspeicher, Zeitgeberbasierte Einheiten, wie z.B. Counter/Timer oder Watchdogs, sowie eine Unterbrechnungssteuerung. Hier wird erkennbar, dass ein Mikrocontroller letztendlich ein beinahe vollwertiger Rechner ist, nur eben deutlich kleiner und leistungsschwächer. Aus diesem Grund werden Mikrocontroller häufig als “Single-Chip-Computer” bezeichnet.
Wie aufgezeigt, ist dagegen bei einem FPGA die CLB-Matrix das Herzstück, in welche die logischen Funktionen, sprich die Funktionalität, erst implementiert werden müssen. [8, S.258ff.] [9]

III. Funktionsweise eines FPGAs

A. Grundlegende Funktionsweise

Im Allgemeinen erhält ein FPGA seine spezifische Funktionalität, indem die CLBs mit logischen Funktionen konfiguriert werden. Anschließend werden diese durch programmierbare Verbindungsleitungen miteinander verdrahtet. An den Kreuzungspunkten von Verbindungen gibt es ebenfalls die Möglichkeit bestimmte Leitungen miteinander zu verbinden oder auch nicht. Die in Abb. 2a erkennbaren gestrichelten Linien können nun je nach Bedarf so konfiguriert werden, dass beispielsweise eine der in Abb. 2b aufgezeigten Verbindungsmöglichkeiten entsteht.

Hierbei ist hinzuzufügen, dass bei FPGAs zwischen der reversiblen und irreversiblen Programmiertechnologie unterschieden wird. Wie die Bezeichnungen vermuten lassen, ist bei der reversiblen Programmiertechnologie eine erneute Konfiguration des FPGA möglich, während bei Verwendung der irreversiblen Programmiertechnologie das FPGA nur ein einziges Mal konfiguriert werden kann. Erstere wird realisiert, indem die Konfigurationsdaten in Speicherzellen, meist SRAM[1]-Zellen, abgelegt werden, welche immer wieder neu beschrieben werden können. Bei der irreversiblen Programmiertechnologie dagegen werden sogenannte Antifuses (Anti-Sicherungen) verwendet. Diese werden durchgeschmolzen, wodurch nicht mehr rekonfigurierbare Verbindungen entstehen. Dies wird von einem Programmiergerät erledigt. Ein Vorteil der Antifuse-Technologie ist die sofortige Betriebsbereitschaft nach einem Neustart, während bei SRAM-Zellen die Konfiguration zusätzlich in einem externen Speicher abgelegt und im Falle eines Neustarts erst geladen werden muss. Dafür können FPGAs mit reversibeln Programmiertechnologien an geänderte Anforderungen angepasst werden. [5, S.106ff., S.130f.]



(a) Kreuzung (b) mögliche Verbindungen

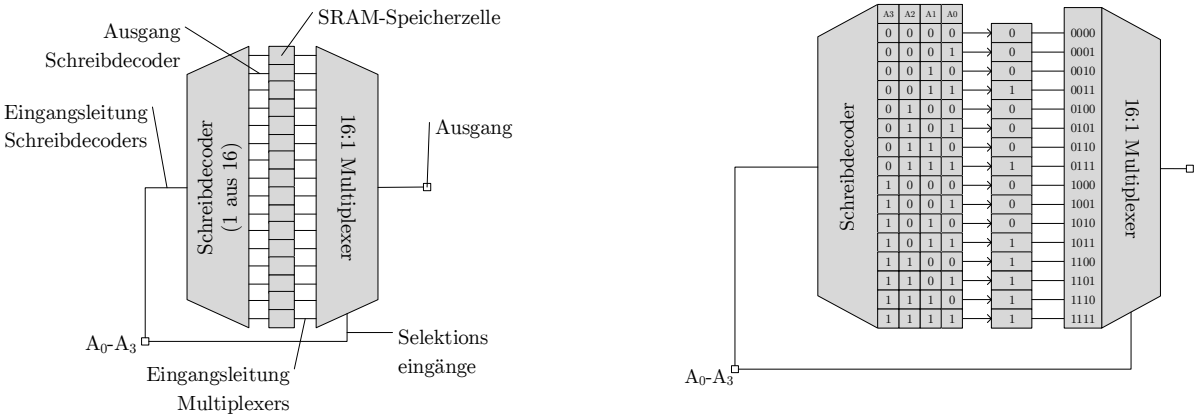
Abb. 2: Kreuzungen im Verbindungsnetzwerk eines FPGAs (nach [4, S.77])

B. FPGAs mit LUT-Basiszellen

LUT[2]-Basiszellen werden meistens in FPGAs mit SRAM-Technologie verwendet, also einer reversiblen Programmiertechnologie. Diese Form der Lösung ist heute am weitesten verbreitet. Eine LUT funktioniert dabei wie eine Wahrheitstabelle. Typischerweise arbeiten LUTs in FPGAs mit drei bis sechs Eingangs- und einem Ausgangssignal. Um daraus beliebige logische Funktionen darstellen zu können, benötigt eine LUT mit k Eingängen (auch k-LUT) folglich 2^k Speicherplätze, da es 2^k mögliche Kombinationen von k Variablen gibt, welche jeweils 0 oder 1 annehmen können. Je nach dem welchen Wert die aus den Eingangssignalen zusammengesetzte Adresse hat, wird ein entsprechender Eintrag aus der Tabelle gewählt und ausgegeben. Diese Auswahl übernimmt dabei ein Multiplexer. Letzterer hat n digitale Eingänge, wobei einer von diesen mit Hilfe von $\lceil \log_2(n) \rceil$ Selektionseingängen auf den Ausgang geschaltet wird.

Abb. 3a zeigt den schematischen Aufbau einer LUT mit SRAM-Zellen. Der 1-aus-n-(Schreib)Decoder wird bei der Konfiguration verwendet, um eine Speicherzelle anzusteuern, sodass sie anschließend mit den entsprechenden Daten beschrieben werden kann, welche zuvor festgelegt werden müssen. Wie in Abb. 3b erkennbar wurde also je nach Kombination aus Eingangssignalen eine bestimmte Speicherzelle beschrieben, wobei in diesem Beispiel die logische Funktion $(A_3 \wedge A_2) \vee (A_1 \wedge A_0)$ implementiert wurde. Die Eingänge des dahinterliegenden Multiplexers sind bei LUTs gerade die jeweiligen Inhalte der Speicherzellen und die Selektionseingänge die Eingangssignale (hier: A_0 bis A_3), welche von außen in die Basiszelle gelangen. Erreicht nun eine bestimmte Kombination letzterer den Multiplexer, prüft dieser, welche Speicherzelle auf den Ausgang geschalten werden muss, indem von oben (im Beispiel: 0000) nach unten (im Beispiel: 1111) die Eingangskombination mit den verschiedenen Möglichkeiten an Kombination (im Beispiel: 16) verglichen wird. Bei Übereinstimmung wird die dahinterliegende Zelle

[1] Static Random Access Memory
[2] Lookup Table



(a) Aufbau einer LUT (i.A.a. [3, S.211])

(b) Beispiel für eine LUT (i.A.a. [3, S.515])

Abb. 3: Aufbau und Beispiel einer Lookup Table

über den Ausgang ausgegeben.
Beispielsweise stellt der Multiplexer in Abb. 3b für die Eingangssignale $A_3 = 0$, $A_2 = 0$, $A_1 = 1$ und $A_0 = 1$, also die Kombination 0011, erst beim vierten Vergleich eine Übereinstimmung fest und gibt entsprechend die dahinterliegende $'1'$ aus. Da $(0 \wedge 0) \vee (1 \wedge 1)$ gerade 1 bzw. `true` als Ergebnis hat, liefert der Multiplexer das erwartete und richtige Ergebnis, wie für alle anderen Kombinationen an Eingangssignalen auch. Der Multiplexer schlägt also das richtige Ergebnis in der "Wahrheitstabelle" nach, wodurch mit LUTs sämtliche logische Funktionen realisiert werden können.
Optional kann noch ein Flipflop hinter die LUT verschaltet werden, um Taktsynchronität zu erreichen. Außerdem befinden sich üblicherweise mehrere LUTs in einer CLB, beispielsweise sind bei FPGAs von Xilinx zwei sogenannte Slices pro CLB vorhanden, welche wiederum jeweils aus zwei 4-LUT/Flipflop Einheiten bestehen (zur Erinnerung: k-LUT: k Eingänge). Das FPGA erhält nun seine Funktionalität, indem die verschiedenen LUTs innerhalb der CLB, sowie außerhalb entsprechend miteinander verschalten werden. [3, S.209ff., S.514f.] [6, S.23f.]

C. FPGAs mit Multiplexer-Basiszellen

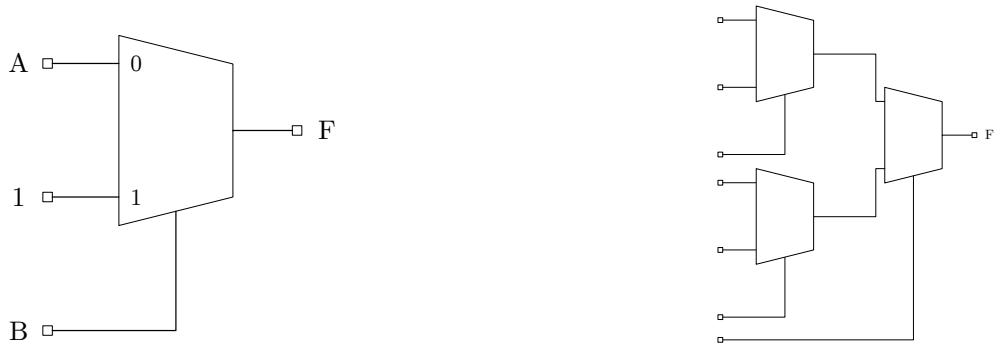
Neben der LUT-Technologie gibt es auch FPGAs, welche Multiplexer als Basiszellen verwenden. Letztere werden zwar - wie erwähnt - auch in FPGAs mit LUT-Basiszellen eingesetzt, in diesem Fall aber lediglich zur Auswahl eines entsprechenden Eintrags aus der SRAM-Speicherzelle. Multiplexer können jedoch auch benutzt werden, um logische Funktionen zu implementieren. Auf diese Art und Weise werden sie zumeist in Verbindung mit Antifuse-FPGAs verwendet.
Abb. 4a zeigt die mit einem Multiplexer realisierte OR-Funktion. A und $'1'$ sind dabei die Eingangssignale und B der Selektionseingang. Je nach dem, ob B 0 oder 1 annimt, werden A oder $'1'$ auf den Ausgang F geschalten, was ja gerade die OR-Funktion repräsentiert. In diesem Zusammenhang ist das Shannon'sche Expansionstheorem von Bedeutung, welches wie folgt definiert ist [3, S.210]:

$$\begin{aligned} f(x_{n-1}, \ldots, x_1, x_0) \;=\; & (\overline{x_i} \;\wedge\; f(x_{n-2}, \ldots, x_i =' 0', x_1, x_0)) \;\vee \\ & (x_i \;\wedge\; f(x_{n-1}, \ldots, x_i =' 1', x_1, x_0)) \end{aligned}$$

Damit lässt sich nun $F = A \vee B$ umschreiben in

$$F = (\overline{B} \wedge (A \vee' 0')) \vee (B \wedge (A \vee' 1')) = (\overline{B} \wedge A) \vee (B \wedge' 1').$$

F wurde also in diesem Fall nach B entwickelt. Nun wird auch klar, wie genau F mit einem Multiplexer wie in Abb. 4a realisiert wurde: Die Variable, nach der entwickelt wurde, wird als Selektionseingang



(a) $A \vee B$ mit Multiplexer

(b) Komplexere Funktion mit Multiplexer

Abb. 4: Logische Funktionen mit Multiplexern

gewählt, während die übrig gebliebenen Funktionen f als Eingänge gewählt werden. In diesem vereinfachten Beispiel sind letztere einzelne Variablen, wäre dem nicht so, so müssten diese wiederum nach einer Variable entwickelt werden. Anschließend werden als Eingänge zwei Multiplexer verwendet, welche gerade diese Teilfunktionen f implementieren (Abb. 4b). Dieser Schritt muss entsprechend oft wiederholt werden. Man erhält also einen n-Mutiplexer, indem mehrere 2-Multiplexer zusammengeschaltet werden. Wie bei den LUT-Basiszellen befinden sich auch bei den Multiplexer-Basiszellen mehrere Multiplexer in einer CLB. Auch hier können durch interne und externe Verschaltung beliebige logische Funktionen realisiert werden. [5, S.44] [3, S.210f.] [1, S.112]

D. Vergleich: Funktionsweise eines Mikrocontrollers

Der größte Unterschied zwischen FPGAs und Mikrocontrollern bezüglich der Funktionsweise liegt zunächst in der Art und Weise, wie beide Bausteine diese erhalten. Wie aufgezeigt wird beim FPGA Hardware konfiguriert und die Funktion rein durch Verknüpfung logischer Funktionen realisiert, sei es mit LUTs oder Multiplexern, während beim Mikrocontroller die Funktion mittels Software implementiert wird. Die CPU des Mikrocontrollers arbeitet dann das im Speicher abgelegte Programm entsprechend ab. Ein Mikrocontroller ist also, was die Hardware betrifft, bereits voll funktionsfähig. Ein FPGA dagegen ist im unkonfigurierten Zustand nicht einsatzbereit. [5, S.55f.]

IV. Programmierung/Konfigurierung eines FPGAs

A. FPGA-Entwurfsablauf

Bereits mehrfach wurde nicht von der “Programmierung”, sondern von der “Konfigurierung” eines FPGAs gesprochen. Dies hat auch so seine Richtigkeit, da ein FPGA über die Hardware seine Funktionalität erhält, nicht über Software, was bereits als grundlegender Unterschied zu Mikrocontrollern genannt wurde. Dennoch wird häufig auch von der “Programmierung” eines FPGA gesprochen.

Im Folgenden wird nun der Entwursablauf eines FPGAs behandelt.

1) Schaltungsentwurf:

Nach der Planung und Dokumentation der Schaltung erfolgt zunächst ein Schaltungsentwurf des FPGAs. Dies geschieht meistens über ein Hardwarebeschreibungssprache (Hardware Description Language, kurz HDL). Üblicherweise wird hierfür VHDL oder Verilog eingesetzt. Sämtliche Bausteine, beispielsweise die LUTs, werden damit abstrakt beschrieben. Daneben gibt es noch die Möglichkeit, die gewünschte Schaltung grafisch mit einem entsprechenden Editor einzugeben. Bei hoher Komplexität erweist sich diese

Methode allerdings als eher ungünstig. Das darauffolgende Vorgehen ist aber der üblicheren Methodik mit HDLs ähnlich.

2) Simulation und Verifikation:
Im Anschluss an den Schaltungsentwurf folgt in der Regel die funktionale Simulation und Verifikation des Entwurfs, um etwaige Fehler bereits jetzt zu beheben. Dafür wird die Hardwarebeschreibung der Schaltung in eine Testumgebung eingebunden. In dieser wird die Schaltung mit Hilfe von sogenannten Stimuli-Generatoren mit Eingangssignalen versorgt und die Ausgangssignale werden mit den erwarteten Ergebnissen verglichen. Je nach dem, ob alle Spezifikationen erfüllt werden, muss entweder der Entwurf korrigiert und wiederholt simuliert werden oder die Simulation und Verifikation ist abgeschlossen.

3) Synthese:
Nun kann die Synthese der Schaltung durchgeführt werden. Der verhaltensbeschreibende Schaltungsentwurf wird dabei in eine strukturbeschreibende Netzliste "übersetzt". Das verwendete Werkzeug muss selbstverständlich auf den zu konfigurierenden FPGA abgestimmt sein, da diese in ihrem Aufbau je nach Hersteller unterscheiden. Aus diesem Grund empfiehlt es sich die von letzteren bereitgestellten Programme zu verwenden. Weitherhin muss darauf geachtet werden, dass die Schaltung überhaupt synthesefähig ist, d.h., dass die HDL-Beschreibung tatsächlich derartig umgesetzt werden kann. Aus diesem Grund ist es oft hilfreich während der Simulation und Verifikation bereits eine Synthese durchzuführen, um gerade dies zu prüfen.

4) Platzierung und Verdrahtung:
Im letzten Schritt erfolgt nun die Platzierung und Verdrahtung, auch Place and Route, der Elemente aus der Netzliste. Diese gibt nämlich noch nicht genau an, wie das FPGA letztendlich aufgebaut ist. Dieser Schritt wird ebenfalls zumeist mit den Programmen und Werkzeugen der entsprechenden Hersteller automatisch durchgeführt. Bei Bedarf kann dann beispielsweise nach der ersten Platzierung manuell optimiert werden. Zuletzt wird dann mit dem verwendeten Programm eine Konfigurationsdatei erzeugt, welche vom FPGA geladen werden kann und es somit entsprechend konfiguriert.

[6, S.63ff., S.119ff.] [3, S.359ff.] [5, S.77ff.]

B. Debugging eines FPGAs

Funktioniert das FPGA nun noch nicht so wie man es sich vorgestellt hat, so bestehen verschiedene Methoden des Debuggings. Zunächst gibt es die bereits angesprochen Möglichkeit der Simulation eines FPGAs innerhalb der Entwicklungsumgebung. Diese kann letztendlich nach jedem der genannten Schritte des Entwurfs durchgeführt werden. Diese Simulation gerät aber schnell an ihre Grenzen, da sie die Realität nur bedingt richtig darstellt. Aus diesem Grund muss der tatsächlich fertig konfigurierte FPGA bzw. die Schaltung getestet werden. Nur so können Probleme und Fehler festgestellt werden, welche im realen Einsatz auftreten. Für diese Art des Debuggings gibt es spezielle Logik-Analysatoren. Grundsätzlich wird hierbei zwischen im FPGA integrierten und externen Logik-Analysatoren unterschieden. Beide zeichnen während des Betriebes Signale auf, welche anschließend entsprechend ausgewertet werden können, um mögliche Fehler zu entdecken. Nur auf diese Art und Weise können Fehler gefunden werden, welche so nicht während der Simulation aufgetreten sind. [10]

C. Vergleich: Programmierung eines Mikrocontrollers

Mikrocontroller können in Hochsprachen wie C/C++ oder teilweise auch Java sowie Assembler programmiert werden. Im FPGA dagegen sind die verwendeten "Sprachen" HDLs. Die Funktion eines

Mikrocontrollers wird somit - wie bereits mehrfach erwähnt - softwareseitig realisiert. Wie die Entwicklungswerkzeuge für FPGAs bieten auch die Werkzeuge für Mikrocontroller die Möglichkeit, den Code vor dem Übertragen zu simulieren und zu testen. Sind alle Anfoderungen erfüllt, wird der Code kompiliert und ein sog. Binder, oft auch Linker, erzeugt eine ausführbare Datei, welche auf den Mikrocontroller übertragen wird. Diese wird in aller Regel im nichtflüchtigen Festwertspeicher abgelegt, sodass der Mikrocontroller auch nach dem Abschalten seine Funktion bewahrt und nicht erneut beschrieben werden muss.
Das Debugging eines Mikrocontrollers gestaltet sich gegenüber einem FPGA deutlich einfacher. In der Regel kann mit einem Debugger Schritt für Schritt die Software von Fehlern befreit werden. Beim FPGA dagegen muss die Hardware auf ihre korrekte Funktionsweise überprüft werden. [9, S.104ff.]

V. Einsatzmöglichkeiten eines FPGAs

A. Einsatzgebiete

Ein erstes Einsatzgebiet liegt im Prototyping. Im Zuge der Entwicklung eines elektronischen Bausteins ist es sehr hilfreich den Entwurf vor der Fertigung zu simulieren und zu testen, um die Funktionstüchtigkeit zu verifizieren. Für die Erstellung eines Prototypen eignen sich daher besonders FPGAs, da mit ihnen auch komplexe Schaltungen realisiert und durch die Rekonfigurierbarkeit sehr leicht korrigiert, sprich von Fehlern befreit werden können. Damit kann verhindert werden, dass Fehler erst nach der Fertigung des Bausteins entdeckt werden.
Daneben bietet sich der Einsatz von FPGAs v.a. dann an, wenn die Verarbeitungsgeschwindigkeit eine große Rolle spielt. FPGAs können, anders als Mikrocontroller, Berechnungen parallel durchführen. Bei Anwendungen, welche Echtzeitverarbeitung erfordern, beispielsweise die Echtzeit-Bildverarbeitung, kommen häufig FPGAs zum Einsatz, da sie durch die Parallelität deutlich schneller gewünschte Berechnungen durchführen können. Ein konkretes Einsatzgebiet, welches eben diese Eigenschaft ausnutzt, sind Ultra High Definition (UHD) Fernseher, welche 4K Inhalte anzeigen und verarbeiten können. Auch im militärischen Bereich werden FPGAs eingesetzt, z.B. als Antriebssteuerung in einer (zielsuchenden) Rakete. Des Weiteren finden letztere auch in der Raumfahrttechnologie oder in Digitalkameras Anwendung.
Weiterhin werden FPGAs auch neben einem Prozessor als Co-Prozessoren eingesetzt, um die Datenverarbeitung zu beschleunigen. Bei Bedarf kann der Prozessor Berechnungen auf den FPGA auslagern, was unter Umständen einen deutlichen Geschwindigkeitsvorteil mit sich bringt. Es ist auch möglich, dass das FPGA ohne Unterbrechnung des Systems dynamisch neu konfiguriert werden kann, sodass der Prozessor mehrere verschiedene Berechnungen auslagern kann. Diese Technologie befindet sich allerdings noch in der Entwicklung. [5, S.54ff.] [11, S.8f.] [3, S.223f.] [12]

B. Vor- und Nachteile eines FPGAs gegenüber Mikrocontrollern

Der wohl größte und wichtigste Vorteil von FPGAs gegenüber Mikrocontrollern ist die bereits erwähnte parallele Verarbeitung. Mikrocontroller dagegen arbeiten sequentiell, weshalb FPGAs im Vergleich dazu einen deutlichen Geschwindigkeitsvorteil haben. Abb. 5 veranschaulicht dies. Während die CPU im Mikrocontroller für den Algorithmus zwölf sequentiell durchgeführte Schritte benötigt, braucht das FPGA vereinfacht gesehen lediglich zwei. Im ersten Schritt können die beiden Additionen parallel durchgeführt werden, um anschließend die Differenz aus beiden Teilergebnissen im zweiten Schritt zu bilden.
Ein erster Nachteil dagegen ist, dass mit Mikrocontrollern mehr oder wenig beliebig komplexe Aufgaben gelöst werden können, zumindest sofern der Programmspeicher entsprechend groß ist, während dies bei FPGAs ab einem bestimmten Zeitpunkt an der begrenzten Anzahl an Logikblöcken scheitert. Des Weiteren sind FPGAs in der Anschaffung meist teurer als Mikrocontroller. Hinzu kommen der höhere Stromverbrauch sowie die erneut nötige Konfiguration nach dem Ausschalten, sofern kein FPGA mit Antifuse-Technologie verwendet wurde. [5, S.55f.]

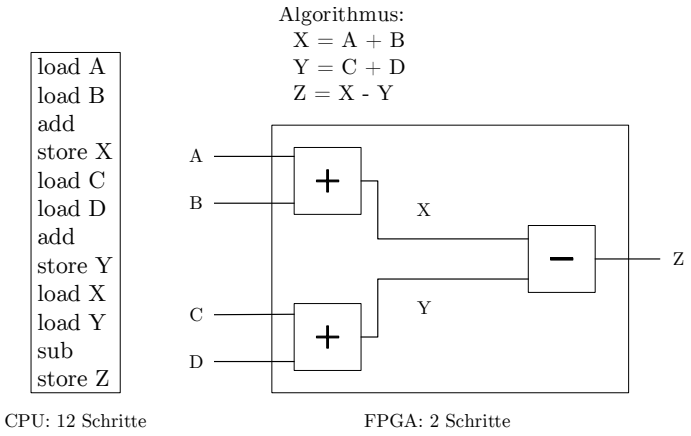


Abb. 5: FPGA vs. Mikrocontroller (i.A.a. [5, S.58])

VI. Zusammenfassung und Ausblick

Field Programmable Gate Arrays sind programmierbare Logikbausteine. Sie erhalten ihre Funktion beim Kunden bzw. Anwender und nicht beim Hersteller - daher auch die Bezeichnung "feldprogrammierbar". Sie bestehen heute im Wesentlichen aus matrixförmig angeordneten Basiszellen, einer gitterartigen Verdrahtung letzterer und Input/Output-Blöcken. Ihre Funktion ergibt sich durch die Implementierung von logischen Funktionen in die Basiszellen, welche anschließend über die beschriebene Verdrahtung untereinander verbunden werden. Für die konkrete Realisierung dieser logischen Funktionen bieten sich LUTs und Multiplexer an, wobei erstere bei rekonfigurierbaren und zweiterer bei nicht-rekonfigurierbaren FPGAs zum Einsatz kommt. Die durch parallele Verarbeitung erreichte hohe Rechenleistung eines FPGAs führt dazu, dass diese in vielen Bereichen eingesetzt werden, wo genau dies erforderlich ist, beispielsweise in UHD-Fernsehgeräten. Der größte und wichtigste Unterschied zu Mikrocontrollern ist, dass FPGAs ihre Funktionalität durch die Konfigurierung von Hardware erhalten, während auf Mikrocontroller Software geladen wird, welche die Funktion beinhaltet und entsprechend abgearbeitet wird.
Wie in anderen (elektronischen) Bereichen, werden auch immer wieder neuere, leistungsstärkere FPGAs auf den Markt gebracht. Zum einen wächst die Anzahl der verbauten Logiblöcke, I/O-Blöcke etc. (vgl. Moore'sches Gesetz), zum anderen werden auch immer mehr der beschriebenen Zusatzkomponenten auf einem FPGA verbaut. Es wird sich zeigen welche vielleicht jetzt noch ungeahnten Möglichkeiten FPGAs in der Zukunft bieten werden können.

LITERATUR

[1] K. Fricke, *Digitaltechnik : Lehr- und Übungsbuch für Elektrotechniker und Informatiker*, 7th ed., ser. SpringerLink : Bücher. Wiesbaden: Springer Vieweg, 2014.

[2] H. Flügel, *FPGA-Design mit Verilog*. München: Oldenbourg, 2010.

[3] F. Kesel and R. Bartholomä, *Entwurf von digitalen Schaltungen und Systemen mit HDLs und FPGAs : Einführung mit VHDL und SystemC*, 3rd ed. München: Oldenbourg, 2013.

[4] A. Biere, Ed., *Digitaltechnik - Eine praxisnahe Einführung*. Berlin: Springer, 2008.

[5] M. Wannemacher, *Das FPGA-Kochbuch*, 1st ed. Bonn [u.a.]: Internat. Thomson Publ. Co., 1998.

[6] P. Sauer, *Hardware-Design mit FPGA : eine Einführung in den Schaltungsentwurf mit FPGA*. Aachen: Elektor, 2010.

[7] o.V. Mikrocontroller.net. FPGA. Zuletzt aufgerufen am: 22.12.2014. [Online]. Available: http://www.mikrocontroller.net/articles/FPGA

[8] K. Wüst, *Mikroprozessortechnik : Grundlagen, Architekturen, Schaltungstechnik und Betrieb von Mikroprozessoren und Mikrocontrollern*, 4th ed., ser. SpringerLink : Bücher. Wiesbaden: Vieweg+Teubner, 2011.

[9] U. Brinkschulte and T. Ungerer, *Mikrocontroller und Mikroprozessoren*, ser. eXamen.pressSpringerLink : Bücher. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[10] o.V. Altera. Debugging FPGA Systems. Zuletzt aufgerufen am 21.12.2014. [Online]. Available: ftp://ftp.altera.com/outgoing/download/education/events/highspeed/Tek_ALTERAFPGADEBUG_IPIntegration_final.pdf

[11] R. Dietrich, "SGI RASC: Evaluierung einer Programmierplattform zum Einsatz von FPGAs als Hardware-Beschleuniger im Hochleistungsrechnen," Master's Thesis (Diplomarbeit), Technische Universität Dresden, Jun 2009. [Online]. Available: http://queens.inf.tu-dresden.de/dietrich_diplom.pdf

[12] o.V. Xilinx. Field Programmable Gate Array (FPGA). Zuletzt aufgerufen am: 20.12.2014. [Online]. Available: http://www.xilinx.com/applications.html

[13] R. Gessler, *Entwicklung Eingebetteter Systeme : Vergleich von Entwicklungsprozessen für FPGA- und Mikroprozessor-Systeme Entwurf auf Systemebene*, ser. SpringerLink : Bücher. Wiesbaden: Springer Vieweg, 2014.

[14] W. Plaßmann, *Handbuch Elektrotechnik : Grundlagen und Anwendungen für Elektrotechniker*, 6th ed., ser. SpringerLink : Bücher, D. Schulz, Ed. Wiesbaden: Springer Vieweg, 2013.