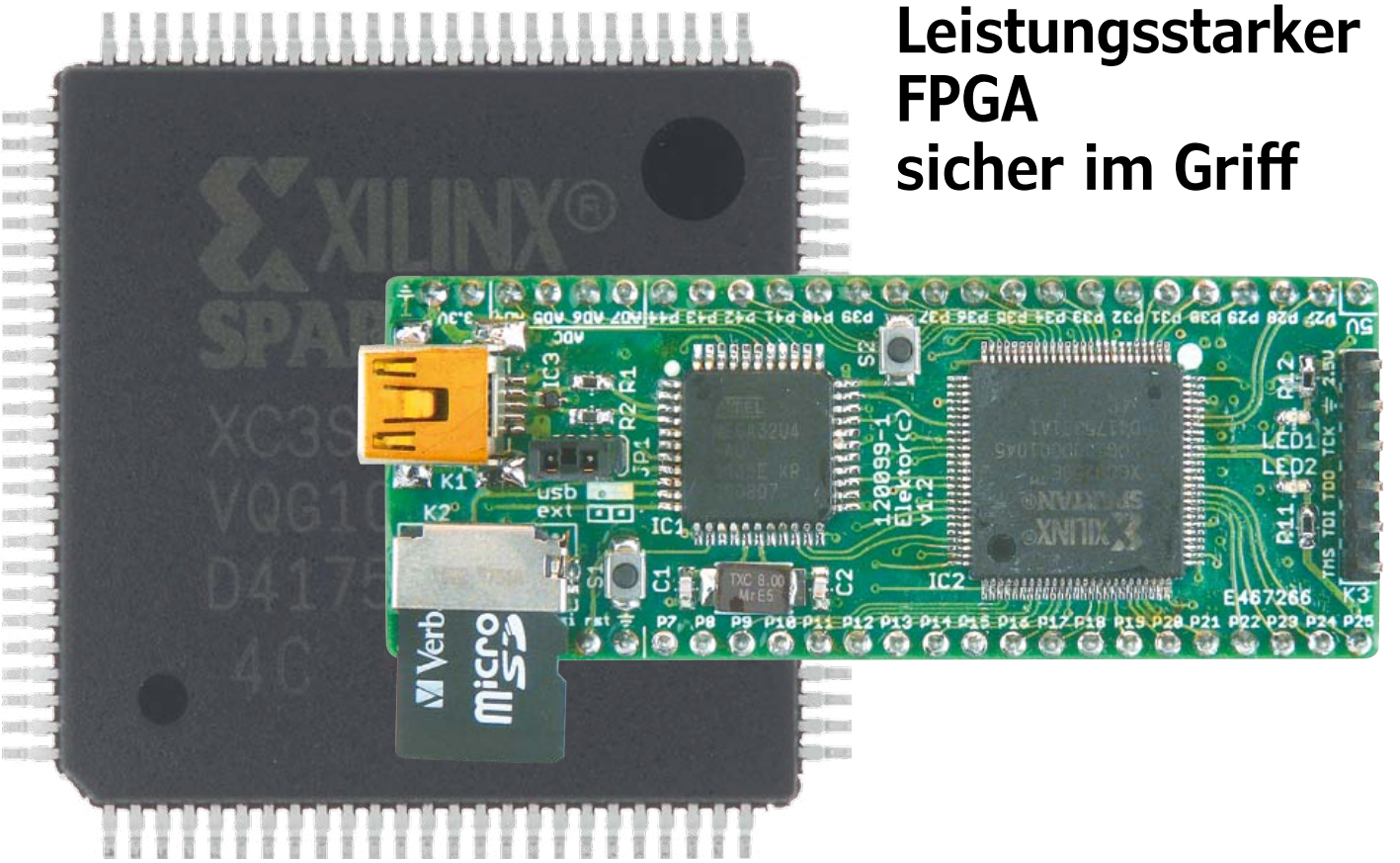
Bauen Sie Ihren Chip!

Leistungsstarker FPGA sicher im Griff



Text:
Clemens Valens
(Elektor-Labor)
Projekt:
Raymond Vermeulen
(Elektor-Labor)

Zu den aktiven Komponenten, die sich durch enorme Flexibilität, aber auch durch hohe Komplexität auszeichnen, zählen zweifellos die FPGAs. Das Grundkonzept erlaubt dem Entwickler eine Vorgehensweise, die dem Konstruieren maßgeschneiderter Chips gleichkommt. In dieser Artikelfolge wollen wir uns auf das Terrain der programmierbaren Logik-Bausteine begeben, die Basis ist das im Elektor-Labor entwickelte FPGA-Board. Schwellenangst war gestern, das Elektor-FPGA-Board hilft, diese Hürde zu nehmen!

Die Kurzbezeichnung FPGA steht für *Field Programmable Gate Array*, was wörtlich „Feldprogrammierbare Anordnung logischer Ports“ bedeutet. Eine freiere und vermutlich bessere Übersetzung lautet „Baustein mit digitalen logischen Funktionen, die vom Anwender konfigurierbar sind“. FPGAs vereinen heute eine Vielzahl logischer Ports, Flipflops, Speicherelemente und noch einiges mehr auf dem gleichen Chip, vom Anwender können sie ähnlich dem Baukastenprinzip zu digitalen Systemen zusammengestellt werden. Aus dieser grundlegenden Eigenschaft resultiert der generelle Unterschied zu den Mikrocontrollern: FPGAs führen keine Programme aus, sie müssen als komplexe digitale Schaltungen betrachtet werden.

Erste Ansätze

Programmierbare Logik bestand zunächst aus einer Matrix, in der mehrere Dutzend logischer Ports versammelt waren. Die Ein- und Ausgänge konnten weitgehend frei miteinander gekoppelt

werden. Unterschiedliche logische Funktionen ließen sich mit dem gleichen Chip realisieren, die Vielzahl diskreter logischer Bausteine auf überdimensionalen Platinen war Vergangenheit. Hinzu kam, dass Funktionen durch Neuprogrammieren des Chips schnell angepasst werden konnten, ein eventuelles Durchtrennen von Leiterbahnen und Verlegen neuer Leitungen entfiel. Die Klasse der programmierbaren logischen Chips ist dicht bei den ROMs (*Read-Only Memory*) und den EPROMs (*Erasable Programmable Read-Only Memory*) angesiedelt, die verwandte Eigenschaften haben. Ein einfaches Beispiel ist ein AND-Port mit den Eingängen A und B sowie dem Ausgang Q. Zu diesem Port gehört folgende Wahrheitstabelle:

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

Wenn die Eingänge A und B in A1 und A0 umbenannt werden und Ausgang Q die Bezeichnung D0 erhält, deutet dies auf zwei Adressleitungen hin, die vier Speicherplätze adressieren. Werden in die Speicherplätze die Werte 0, 0, 0 und 1 gesetzt, entsteht eine AND-Funktion. Die Speicherplätze können auch mit anderen Werten belegt werden, die Anzahl der Adressleitungen ist ebenfalls variierbar. Nach diesem Grundprinzip lassen sich auf unkomplizierte Weise auch hochkomplexe logische Funktionen zusammenbauen. Allerdings ist das Speichern eines Ergebnisses, das eine Funktion liefert, noch nicht möglich. Programmierbare Logik-Bausteine wurden deshalb durch Register ergänzt, die Ergebnisse zwischenspeichern, so dass sie von anderen Teilen des gleichen logischen Systems weiterverarbeitet werden können. Die einfachste Art eines Registers ist das Flipflop, es kann genau 1 bit speichern. In **Bild 1** ist ein solches 1-bit-Speicherelement, wie es in FPGAs vorkommt, schematisch dargestellt.

Flipflops können die Grundelemente vieler Funktionen sein, sie können beispielsweise zu Zählern oder Registern beliebiger Breite zusammengeschaltet werden. Das Verzögern von Datenflüssen ist möglich, und auch Schieberegister lassen sich realisieren, die parallele oder serielle Datenformate ineinander konvertieren.

Fazit: Eine aus logischen Ports bestehende Matrix ist der richtige Ansatz, doch erst wenn Speicherelemente hinzukommen, sind neue Ziele erreichbar.

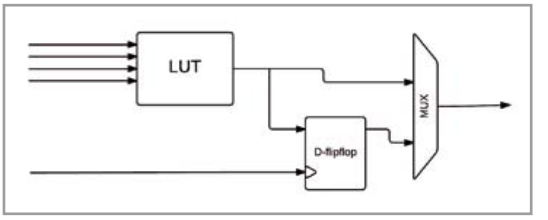


Bild 1.
Stark vereinfachte blockschematische Darstellung einer logischen Zelle in einem FPGA.

Fortentwicklung

Die Komplexität programmierbarer logischer Bausteine wuchs im Laufe der Jahre stetig. Das ist bereits an der Kurzbezeichnung CPLD einer Gruppe solcher Bausteine erkennbar, die *Complex Programmable Logic Device* bedeutet. Die fortschreitende Entwicklung machte es nicht nur möglich, sondern auch notwendig, die Bausteine so zu gestalten, dass sie eher mit den RAMs (*Random Access Memory*) als den ROMs (*Read Only Memory*) verwandt sind. Allerdings verlieren RAMs nach Abschalten der Betriebsspannung ihren Inhalt, und die neuen programmierbaren Logikbausteine verlieren ihre Konfiguration. Die Idee zur Lösung dieses Problems besteht darin, den Chip bei jedem Start innerhalb des Systems zu konfigurieren. Dies war die Geburtsstunde des FPGA. Die spezifische Konfiguration wird oft dadurch gesichert, dass die zugehörigen Daten in einem externen, nicht flüchtigen Speicher stehen und vom FPGA nach jedem Reset neu gelesen werden. Hier liegt die Assoziation zu einem Mikroprozessor nahe, dessen Programm ebenfalls in einem separaten, nicht flüchtigen Speicher untergebracht ist. Inzwischen existieren auch FPGAs, die ihre Konfiguration ohne externe Komponenten nicht-flüchtig speichern können.

Im Lauf der Jahre ist es den FPGA-Herstellern gelungen, eine enorme Anzahl Gatter und Flipflops auf einem einzigen Chip unterzubringen. Die immer noch fortschreitende Integration macht einerseits neue Anwendungen mit extrem schnellen Datenflüssen möglich, andererseits schafft sie neue Probleme, beispielsweise hinsichtlich der internen Chip-Organisation. Das wahlfreie Verknüpfen etlicher Millionen Ports erfordert eine kaum noch überschaubare Anzahl von Verbindungswegen. Die „Leitungen“ fallen lang aus, wenn Ausgänge links oben auf dem Chip mit diagonal gegenüberliegenden Eingängen gekoppelt werden müssen. Bei hohen Datengeschwindigkeiten können die Weglängen Ursache für Verzögerungen und Deformationen der Signale bis zur Unbrauchbarkeit sein. Die Synchronisation der beteiligten Ports ist nur noch mit großer Mühe möglich.

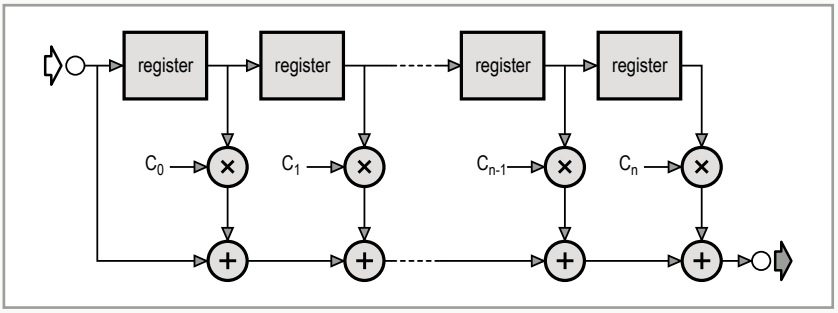


Bild 2.
Digitale Filter bestehen generell aus Verzögerern, Multiplizierern und Zählern. Verzögerer sind mit Registern realisierbar, während die Multiplizierer und Zähler aus logischen Funktionen und Registern aufgebaut werden können. Da dies alles in einem einzigen FPGA implementierbar ist, sind FPGAs für die digitale Signalverarbeitung ideal.

Die beschriebene Situation hat dazu geführt, dass sich eine allgemein gebräuchliche FPGA-Architektur herausgebildet hat. Die Anordnung der Funktionsblöcke auf dem Chip ist nicht willkürlich, sondern wohlüberlegt. Die Hersteller haben vor Augen, dass sich Standard-Produkte auf möglichst breiter Front einsetzen lassen, so dass die Stückzahlen steigen. In den Typen, die für spezielle Aufgaben konzipiert sind, findet der Anwender darüber hinaus besondere Funktionen vor. Eigentlich ist nur dies der Grund, warum kein FPGA-Typ einem anderen vollständig gleicht. Die Funktionsblöcke, die gegenwärtig in praktisch jedem FPGA implementiert sind, lassen sich wie folgt klassifizieren:

- Logik
- I/O
- Arithmetik
- Speicher
- Taktung

Tabelle 1. Funktionsblöcke des FPGA Xilinx Spartan-3E XC3S250E-4VQG100C

Ports	250000
Logische Blöcke	612
I/O-Blöcke	66, davon 7 nur als Eingang
Arithmetische Blöcke	12 Multiplier
Speicherblöcke	12, insgesamt 221184 bit (216 Kbit)
Verteilter Speicher	39168 bit (38,25 Kbit)
Taktblöcke	4

Die Anzahl der I/O-Blöcke hängt von der Gehäusevariante ab, die Angabe bezieht sich auf die Ausführung VQ100 mit 100 Anschlüssen. Ein logischer Block, bei Xilinx *Configurable Logic Block* (CLB) genannt, besteht aus vier *Slices* (Segmente), die jeweils zwei 16-bit-LUTs einschließen. Die Hälfte ist als verteiltes RAM nutzbar, die andere Hälfte für logische Funktionen. Zu jedem *Slice* gehören zwei Flipflops, so dass jeder CLB insgesamt über acht Flipflops verfügt (siehe Bild 3).

Außer Funktionen dieser allgemeinen Kategorien können weitere Komponenten integriert sein, zum Beispiel Prozessoren, Flash-Speicher und auch Peripherie wie Ethernet-Controller, Speichersteuerungen, A/D-Wandler oder Module für die serielle Datenkommunikation (SPI, I²C, USB, CAN oder andere). Mit diesem Funktionsumfang rückt der FPGA in die Nähe der autonomen Chipsysteme, sie werden auch „System-on-Chip“ (SoC) genannt.

Funktionsblöcke

Logik

Die logischen Blöcke machen den umfangreichsten Teil eines FPGAs aus, in der Malerei wären sie die „Leinwand“ (englisch *fabric*), auf der ein Künstler ein Gemälde malt. Komponenten von der Art der Matrix, die zu Beginn betrachtet wurde, gehören diesem Bereich an. Ein logischer Block ist eine konfigurierbare Ansammlung von Ports und Registern, wobei einer so genannten LUT (*Look-Up Table*) eine Schlüsselrolle zukommt. Die LUT entspricht exakt dem Beispiel mit dem AND-Port, nur ist sie meistens etwas größer, und sie hat mehr Eingänge. Anders betrachtet ist sie ein Register, beispielsweise mit der Breite 16 bit, mit dem logische Funktionen ausgeführt werden können. LUTs können auch als Schieberegister oder als Speicher dienen. Im zweiten Fall entsteht so genanntes „verteiltes“ RAM (*Distributed RAM*), der Begriff soll den Unterschied zu den Speicherblöcken andeuten. Ergänzt durch Multiplexer und weitere logische Komponenten beherrschen logische Blöcke auch niedere mathematische Operationen, was ihre Verwendbarkeit und Flexibilität steigert.
Da in einem FPGA sehr viele derartige Blöcke verfügbar sind, können Gruppen zu Multiplizierern oder Zählern zusammengeschaltet werden. Digitale Filter bestehen aus Speicherelementen, Multiplizierern und Zählern, mit ihnen sind auch

digitale Fourier-Transformationen ausführbar (**Bild 2**). Logische Blöcke stellen folglich alle Komponenten bereit, um digitale Signale zu filtern oder zu transformieren. Allerdings werden immer mehrere Blöcke benötigt, ein einzelner Block genügt für solche Prozesse nicht. Die digitale Signalverarbeitung gehört zu den starken Domänen der FPGAs. Wenn ein FPGA zwischen einen A/D-Wandler und einen D/A-Wandler geschaltet wird, kann dies zu überraschenden Ergebnissen führen, beispielsweise wenn das Eingangssignal ein Video- oder Audio-Stream ist.

Aktuelle FPGAs vereinen mehr als eine Million logischer Blöcke auf ihrem Chip, zum Beispiel sind im Virtex-7 XC7V2000T von Xilinx 1,2 Millionen LUTs und 2,4 Millionen Flipflops integriert. Dieser FPGA-Chip besteht aus fast sieben Milliarden Transistoren, während es der 10-Kern-Mikroprozesor Xeon Westmere-EX von Intel auf „nur" 2,6 Milliarden Transistoren bringt.

I/O

Die logischen Blöcke müssen ebenso wie die übrigen Funktionsblöcke mit der FPGA-Umgebung über die Chip-Anschlüsse kommunizieren. Die Brücke zur Außenwelt wird von den so genannten I/O-Blöcken geschlagen. Im erwähnten FPGA-Typ XC7V2000T von Xilinx sind 1200 I/O-Blöcke integriert.

Ein I/O-Block in einem FPGA besteht aus einer Vielzahl von Transistoren in einer Push-Pull-Konfiguration. Die I/O-Blöcke unterstützen nicht nur diverse Schnittstellen-Standards, zum Beispiel DDR, PCI, LVTTL, LVCMOS oder LVDS, sie können Datenströme mit Höchstgeschwindigkeiten sowohl unsymmetrisch (*single-ended*) als auch symmetrisch (*differential*) übertragen. Um ihre Aufgaben zu erfüllen, machen sie Gebrauch von Verzögerungsleitungen, Synchronisationsflipflops und auch Parallel-Seriell-Wandlern (*Serializer-Deserializer, SerDes*). Selbstverständlich sind Pullup- und Pulldown-Widerstände ebenso wie Leitungsabschlusswiderstände auf dem Chip integriert.

I/O-Blöcke sind in Banken organisiert, die sich an unterschiedliche Betriebsspannungen anpassen lassen. Dadurch können die Banken gleichzeitig mehrere Schnittstellenstandards unterstützen, auch dies trägt zur Flexibilität der FPGAs bei. Die Leistungsaufnahme und die Störemission

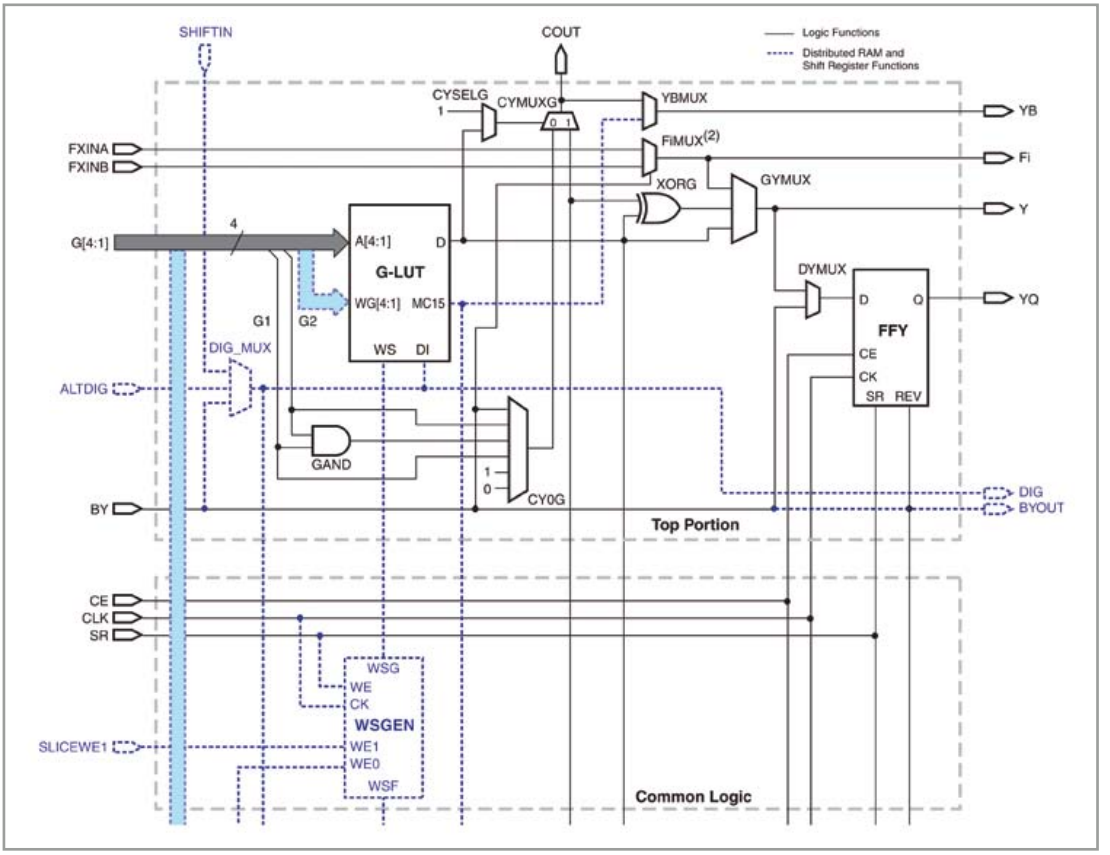


Bild 3. Funktionsschema eines Abschnitts aus einem Slice. Der fehlende Teil ist die „Bottom Portion", sie ist weitgehend mit der umrahmten „Top Portion" identisch.

FPGA-Board

Aus dem Blockschema des FPGA-Boards in **Bild 4** geht hervor, dass ein ATmega32U4 und eine Micro-SD-Karte zum System gehören. Die vollständige Schaltung ist in **Bild 5** wiedergegeben. Der FPGA ist der XC3S250E-4VQG100C aus der Familie „Spartan-3E“ von Xilinx. Auf dem Chip befinden sich 250000 *System Gates*, 5508 *Logic Cells* und 612 *Logic Units*, die maximale Taktfrequenz beträgt 572 MHz. Weitere Eckdaten dieses Typs sind in **Tabelle 1** zusammengefasst. Um die Platinenfläche klein zu halten, haben wir eine FPGA-Variante mit geringen Abmessungen gewählt. Die Bauform 100-VQFP hat 100 Anschlüsse für Oberfächenmontage (SMD), davon gehören 66 Anschlüsse zu Eingängen oder Ausgängen. Das dürfte auch für die Realisierung komplexer Systeme genügen.

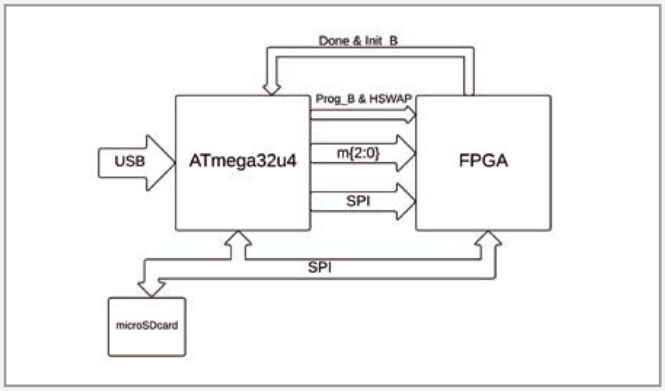


Bild 4.
Bausteine des Elektor-FPGA-Boards: Auf dem Board befinden sich außer dem FPGA ein Mikrocontroller und eine Steckfassung für eine Micro-SD-Karte.

Der Mikrocontroller ATmega32U4 übernimmt die Rolle eines Vermittlers. Er ist für den Datenaustausch mit dem PC zuständig und sorgt dafür, dass der PC mit der Micro-SD-Karte wie mit einem Massenspeicher (*Mass Storage Device*) umgehen kann. Seine wichtigste Aufgabe ist jedoch, die FPGA-Konfigurationsdatei von der Micro-SD-Karte zu lesen und in den FPGA zu laden. Dieser Mikrocontroller kombiniert die bekannten ATmega-Eigenschaften mit der Funktionalität einer USB-Schnittstelle. Als 8-bit-Typ ist er zwar vergleichsweise langsam, doch genügt er für seinen Einsatzzweck voll. Die Firmware steht auf der Elektor-Projektseite [1] zum Download bereit. Der ATmega auf der bestückten und getesteten FPGA-Platine, die der Elektor-Shop liefert, ist bereits programmiert. Die Fassung der Micro-SD-Speicherkarte belegt auf der Platine nur wenig Fläche. Die Karte arbeitet im SPI-Modus, von den meisten Typen wird dieser Modus unterstützt. Die nicht benutzten Leitungen liegen über Pullup-Widerstände an +3,3 V, so dass keine undefinierten Signalzustände auftreten können. Ein Fassungskontakt erkennt, ob eine Karte eingesteckt ist.
Die Spannungsversorgung ist für eine Platine dieses Formats vergleichsweise aufwändig. Der FPGA benötigt die drei Betriebsspannungen 3,3 V, 2,5 V und 1,2 V, die Spannung 3,3 V versorgt auch die Micro-SD-Karte und den Mikrocontroller. Die Entkoppelkondensatoren tragen maßgeblich zur Systemstabilität bei. Wenn Jumper JP1 an seinem Platz ist, wird das FPGA-Board über die USB-Leitung mit Strom versorgt. Die USB-Spannung +5 V liegt auch an Kontakt P26 des Steckverbinders K5. Beim Umgang mit dieser Spannung ist Vorsicht geboten: Die I/O-Anschlüsse des FPGA vertragen +5 V nicht! Wenn JP1 offen ist, kann das Board über P26 von K5 mit +5 V aus einem externen Netzteil versorgt werden. Eine USB-Verbindung ist dann nicht zwingend notwendig.

Wegen der zahlreichen Leitungswege auf enger Fläche hat das Elektor-Labor eine vierlagige Platine entwickelt (**Bild 6**). Die Platine muss nicht in Eigenregie angefertigt und bestückt werden, denn der Elektor-Shop (siehe [1]) liefert das fertig montierte und getestete FPGA-Board zu einem attraktiven Preis.

Konfiguration laden

Normalerweise können FPGAs nicht dauerhaft konfiguriert werden, dies muss nach jedem Reset neu geschehen. Die meisten FPGA-Entwicklungsboards benutzen hierzu einen PC und einen JTAG-Programmer, dies ist hier über Anschluss K3 ebenfalls möglich. Systeme mit FPGAs verwenden zum Speichern der Konfiguration oft Flash-Speicher, der mit dem SPI-Bus verbunden ist. In der Startphase liest der FPGA die Konfiguration aus dem Flash-Speicher. Auf dem Elektor-FPGA-Board setzt der Mikrocontroller die Konfiguration nach dem SPI-Protokoll in den FPGA, dieser Modus wird auch *Serial-Slave-Mode* genannt. Die Methode wird überwiegend in größeren Systemen mit mehreren FPGAs praktiziert, bei nur einem FPGA ist sie ebenfalls anwendbar.

Auch mit der Micro-SD-Karte kann der Mikrocontroller über SPI kommunizieren. Damit die Karte unter einem PC-Betriebssystem einsatzfähig ist, muss sie für die Dateisysteme FAT16 oder FAT32 formatiert werden. Der Mikrocontroller muss in der Lage sein, mit dem Dateisystem umzugehen, damit er die Karte lesen kann. Da die Karte auch über das PC- Betriebssystem zugänglich sein soll, muss der Mikrocontroller die so genannten

SCSI-Kommandos interpretieren und ausführen.
Das gewählte Systemkonzept macht den Arbeitsablauf einfach. Nachdem in dem Tool Xilinx ISE die Konfigurationsdatei erstellt wurde, wird sie im Windows Explorer in „config.bin“ umbenannt und auf die Micro-SD-Karte übertragen. Im Explorer erscheint das FPGA-Board als USB-Massenspeicher mit zugewiesenem Laufwerkbuchstaben. Nachdem die Datei übertragen ist, wird das „USB-Massenspeichergerät“ (die FPGA-Karte) vom Betriebssystem abgemeldet. Der Mikrocontroller lädt die Konfiguration in den FPGA, sobald das FPGA-Board neu startet.

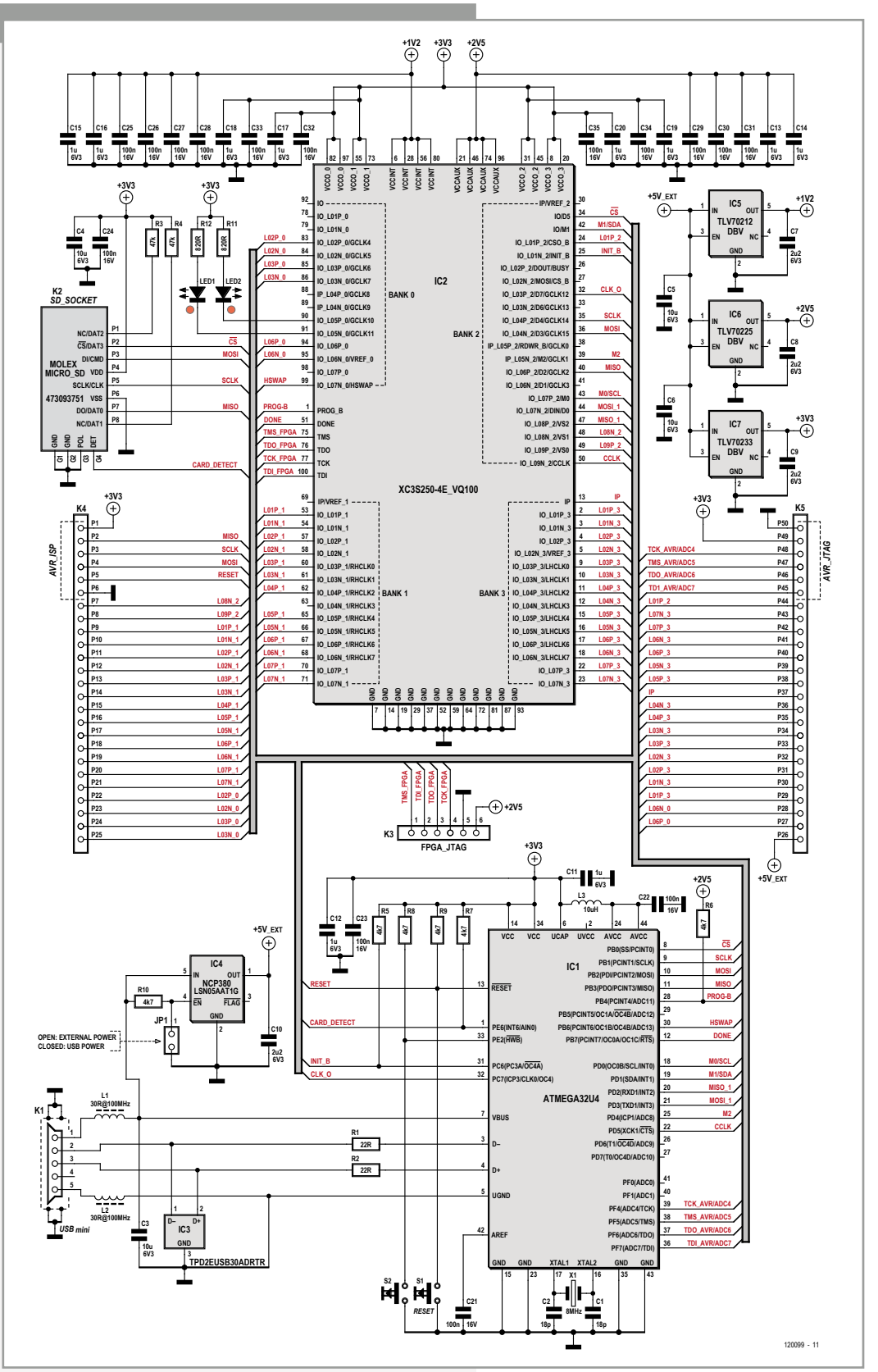


Bild 5.
Die Schaltung des Elektor-FPGA-Boards passt gerade auf eine Druckseite, die zugehörige Platine ist nur 65 mm mal 25 mm groß.

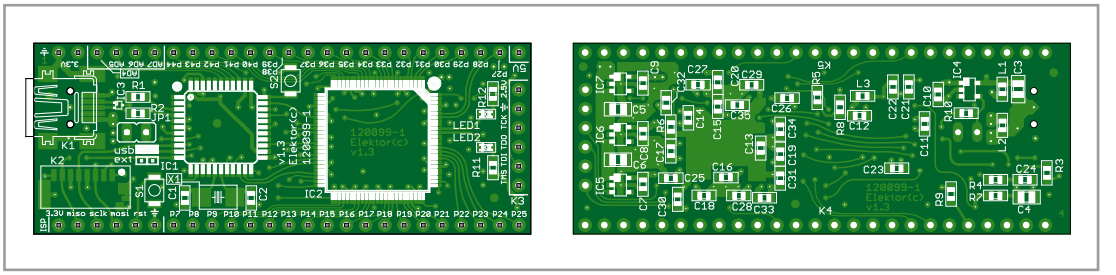


Bild 6.
Die vielen Leitungen lassen sich bei der kleinen Fläche nur mit einer vierlagigen Platine realisieren. Die Platine ist aufgebaut und getestet lieferbar.

sind vom Anwender beeinflussbar, indem er die Flankensteilheit der Signale und den Strom der Ausgangsleitungen konfiguriert.

Arithmetik

Da FPGA-Anwender logische Funktionsblöcke häufig für arithmetische Operationen einsetzen, haben die FPGA-Hersteller in viele Typen vollständige arithmetische Einheiten implementiert, die bequemer konfigurierbar sind. Meistens handelt es sich um Multiplizierer, manchmal sind auch DSP-Funktionen vorhanden, die oft vorkommende Operationen wie a + (b · c) → a (*multiply-accumulate*) beherrschen.

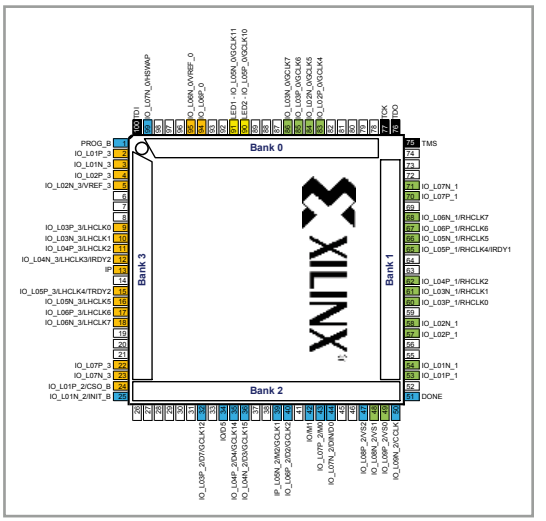


Bild 7.
Anschlüsse des FPGA-Typs XC3S250E-4VQG100C von Xilinx. Die Farben haben folgende Bedeutungen: Orange und Grün führen zu den Board-Steckverbindern, Blau ist die Schnittstelle mit dem Mikrocontroller, an Gelb sind zwei LEDs angeschlossen, und Schwarz ist die JTAG-Schnittstelle, sie ist über einen eigenen Steckverbinder zugänglich.

1	3.3V					GND	50
2	AVR ISP	MISO				3.3V	49
3		SCLK			TCK_AVR/ADC4		48
4		MOSI			TMS_AVR/ADC5	AVR JTAG	47
5		RESET			TDO_AVR/ADC6		46
6	GND				TDI_AVR/ADC7		45
7	48	IO_L08N_2/VS1			IO_L01P_2/CSO_B	24	44
8	49	IO_L09P_2/VS0			IO_L07N_3	23	43
9	53	IO_L01P_1			IO_L07P_3	22	42
10	54	IO_L01N_1			IO_L06N_3/LHCLK7	18	41
11	57	IO_L02P_1			IO_L06P_3/LHCLK6	17	40
12	58	IO_L02N_1			IO_L05N_3/LHCLK5	16	39
13	60	IO_L03P_1/RHCLK0			IO_L05P_3/LHCLK4/TRDY2	15	38
14	61	IO_L03N_1/RHCLK1			IP	13	37
15	62	IO_L04P_1/RHCLK2			IO_L04N_3/LHCLK3/IRDY2	12	36
16	65	IO_L05P_1/RHCLK4/IRDY1			IO_L04P_3/LHCLK2	11	35
17	66	IO_L05N_1/RHCLK5			IO_L03N_3/LHCLK1	10	34
18	67	IO_L06P_1/RHCLK6			IO_L03P_3/LHCLK0	9	33
19	68	IO_L06N_1/RHCLK7			IO_L02N_3/VREF_3	5	32
20	70	IO_L07P_1			IO_L02P_3	4	31
21	71	IO_L07N_1			IO_L01N_3	3	30
22	83	IO_L02P_0/GCLK4			IO_L01P_3	2	29
23	84	IO_L02N_0/GCLK5			IO_L06N_0/VREF_0	95	28
24	85	IO_L03P_0/GCLK6			IO_L06P_0	94	27
25	86	IO_L03N_0/GCLK7				5V	26

Bild 8.
Kontaktbelegungen der Board-Steckverbinder. Die orangen und grünen Pins stimmen mit Bild 7 überein. Der Mikrocontroller kann über diese Steckverbinder programmiert werden. Die Anschlüsse des AVR-JTAG-Ports sind auch als analoge Eingänge nutzbar.

Speicher

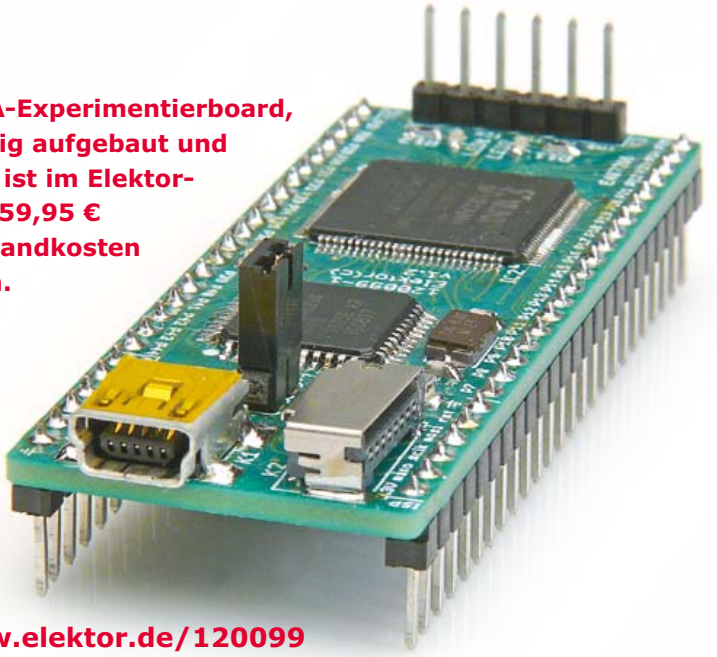
Außer dem verteilten RAM in Form von LUTs sind in FPGAs auch RAM-Sektionen integriert, deren Gesamtgröße im Megabit-Bereich liegen kann. Das Unterbringen von Speicher auf dem Chip macht extrem hohe Datengeschwindigkeiten möglich. Auch funktional vollständige Mikrocontroller-Systeme sind mit einem einzigen FPGA ohne externe Komponenten realisierbar. Wenn RAM-Funktionsblöcke mit zwei voneinander unabhängigen Ports ausgestattet sind, können sie die Funktion von Datenpuffern (FIFOs) übernehmen. Damit sind beispielsweise Systemkomponenten synchronisierbar, die mit unterschiedlichen Geschwindigkeiten arbeiten. Die Speicherwortbreite ist variabel, so dass zum Beispiel 1-bit-Speicher und 32-bit-Speicher gemischt werden können.

Taktung

Obwohl es theoretisch möglich ist, asynchron ablaufende Funktionen (ohne Taktsignal) zu implementieren, stellt dieser Ansatz für die meisten komplexen Aufgaben keine gangbare Lösung dar. Wegen der unterschiedlich langen Wege, die von den Signalen auf dem Chip zurückgelegt werden müssen, kommt es zu Laufzeitdifferenzen. Flanken, die an der Quelle exakt gleichzeitig auftreten, kommen an der Senke zu unterschiedlichen Zeiten an. Dadurch entstehen nicht definierte Zustände, die zu Fehlfunktionen führen. Die Lösung dieses Problems besteht darin, die Signale zu definierten Zeiten gewissermaßen abzutasten, ähnlich wie ein Foto einen visuellen Zustand zu einem definierten Moment einfriert. Diese Methode geht zwar auf Kosten der Geschwindigkeit, doch sie hält die Signalflanken synchron.

Für die korrekte Synchronisation ist das Taktsignal zusammen mit diversen Flipflops verantwortlich. Allerdings unterliegt das Taktsignal den gleichen physikalischen Gesetzen wie andere Signale, auch hier entstehen Laufzeitdifferenzen. Die FPGA-Hersteller begegnen diesem Problem, indem sie dem Anwender konfigurierbare Verzögerungs- und Schiebemechanismen an die Hand geben. Taktsignale können auch über unabhängige interne Taktnetzwerke transportiert werden. Ein weiterer Grund, komplexe Taktungen auf dem Chip zu integrieren, sind unterschiedlich schnelle Subsysteme (so genannte Taktdomänen). Die Taktung ist konfigurierbar, indem Taktsignale mit unterschiedlichen Frequenzen und Phasen von einem Generaltakt abgeleitet werden. Die Taktsignal-Frequenzen können bis in den GHz-Bereich reichen.

Das FPGA-Experimentierboard, vollständig aufgebaut und getestet, ist im Elektor-Shop für 59,95 € plus Versandkosten erhältlich.



Auf www.elektor.de/120099 steht Näheres!

Die SD-Karte

Nach den allgemeinen Betrachtungen wollen wir uns an die ersten praktischen Versuche mit einem FPGA wagen. Zu diesem Zweck hat unser Entwickler Raymond Vermeulen ein FPGA-Experimentierboard entworfen, das einen schwerwiegenden Nachteil vieler anderer FPGA-Boards geschickt umgeht. Gemeint ist das oft recht komplizierte Verfahren, die entwickelte Konfiguration in den FPGA zu laden und zu erhalten. Ein typisches Merkmal vieler FPGA-Software-Tools ist ihre hohe, stark einarbeitungsbedürftige Komplexität. Hinzu kommt, dass zum Laden der Konfigurationsdatei ein externes JTAG-Programmiergerät notwendig ist. Unsere alternative Methode zielt darauf ab, dass ein PC das FPGA-Board wie einen intelligenten USB-Stick einstuft. Physisch ist der USB-Stick eine auf dem FPGA-Board installierte Micro-SD-Karte. Auf die SD-Karte kann die Konfigurationsdatei wie jede andere Datei geschrieben werden. Immer wenn das System gestartet wird, kopiert der FPGA die Konfigurationsdatei von der SD-Karte in seinen flüchtigen Konfigurationsspeicher. Damit ist gleichzeitig das Problem der Treiber für spezielle Programmier-Hardware gelöst.

Im nächsten Monat

Das Konfigurieren eines FPGAs ist keine ganz einfache Aufgabe. Deshalb werden wir uns in Folgebeiträgen mit den Schritten beschäftigen, die zu praktikablen realen Anwendungen führen. Voraussichtlich im nächsten Monat zeigen wir, wie das Gratis-Tool Xilinx ISE Webpack installiert wird. Als ersten Gehversuch lassen wir die LEDs des FPGA-Board blinken. Die Zeit bis dahin können Sie zum Download der Xilinx-Software [2] nutzen. Die kostenlose Webpack-Version hat einen Umfang von etwas mehr als 6 GB. Alternativ können Sie die Software auf DVD (**Bild 9**) bei Xilinx kostenlos bestellen. In der Regel kommt sie innerhalb von zwei Wochen ins Haus.

(120099)gd

Weblinks

[1] www.elektor.de/120099

[2] www.xilinx.com/support/download/index.htm

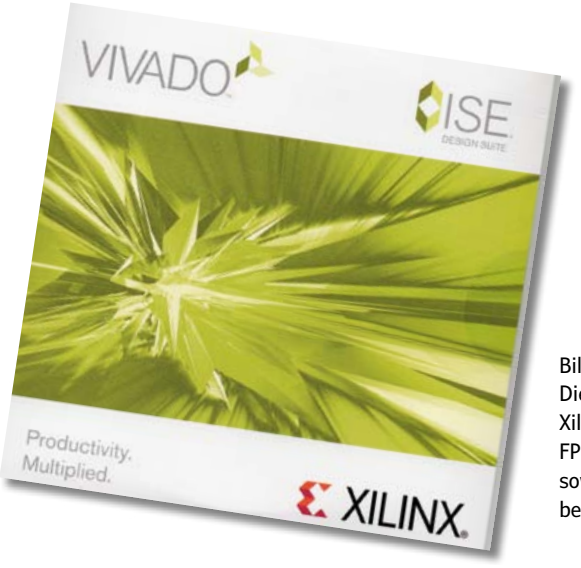


Bild 9.
Die kostenlose DVD von Xilinx, auf der sich das FPGA-Tool „ISE Webpack“ sowie weitere FPGA-Tools befinden.