

Alternative Implementierung des EPP Adapters

Thomas Sailer, HB9JNX/AE4WA, und Johannes Kneipp, DG3RBU

4. April 1998

1 Einleitung

Im Zuge der Einführung breitbandiger Userzugänge bestand die Notwendigkeit, eine Lösung für den User zu schaffen, womit er einfach und preiswert ein Modem an seinen PC anschliessen kann. Daher wurde der EPP-Adapter entwickelt und an der letztjährigen Darmstädter Tagung präsentiert [7].

Messungen auch an älteren PC's (486er) hatten ergeben, daß HDLC mit der geforderten Bandbreite (einige hundert kbit/s) durchaus mit minimalem Aufwand in Software codiert und decodiert werden kann, wurde im Hinblick auf eine preiswerte Lösung darauf verzichtet, einen HDLC-Controller in Hardware zu realisieren. Auf der Suche nach einer möglichst verbreiteten und einfachen Schnittstelle, die die benötigte Bandbreite bewältigen kann, drängte sich der durch die IEEE normierte erweiterte Parallelport [3] geradezu auf. Damit musste der zu entwickelnde Adapter die 8bit-parallele EPP-Schnittstelle an das bit-serielle Modem anpassen und Daten kurzzeitig zwischenspeichern können, damit sie der PC blockweise verarbeiten kann, das Modem jedoch trotzdem einen kontinuierlichen Datenstrom sieht. Eine Suche nach geeigneten Bausteinen endete beinahe zwangsweise bei den Bausteinen 72131 und 72132 von IDT [5]. Diese Bausteine beinhalten ein FIFO mit seriellem Eingang und parallelem Ausgang bzw. parallelem Eingang und seriellem Ausgang. Je einer dieser Bausteine und ein wenig Beigemüse vervollständigten den Adapter.

Doch leider scheinen diese Bausteine nicht dem allgemeinen Trend der Elektronik zu folgen und werden statt billiger immer teurer. Heute sind diese FIFO's mit über 30 DM pro Stück zu veranschlagen (zwei werden pro Adapter benötigt), sodaß sich die Suche nach Alternativen aufdrängt.

Eine Alternative besteht darin, für den Zwischenspeicher anstelle eines speziellen FIFO's ein normales statisches RAM zu verwenden. Da der Adapter nur wenige kbit Speicher benötigt, ist dies die günstigste Speicherart. DRAM's sind in dieser Grössenordnung nicht billiger, verursachen aber einige Schwierigkeiten durch die schmale Schnittstelle (1 oder 4 Bit) und die kompliziertere Ansteuerung (Refresh). Der ganze Rest der Schaltung, also die seriell/parallel-Wandler, die Bedienung der EPP-Schnittstelle und das Einspeichern und Auslesen des RAMs kann mit einem programmierbaren Logikbaustein.

2 Programmierbare Logik: FPGA's

Die geforderte Datenrate von einigen hundert kbit/s und die maximal übertragbaren 1–2 MByte/s über die EPP-Schnittstelle lassen einen Systemtakt von 10 MHz als ausreichend erscheinen. Die zu erwartende Grösse und die moderate Taktfrequenz sind eine Domäne der FPGA (Field Programmable Gate Array) genannten IC's.

FPGA's bestehen aus einer schachbrettartigen Anordnung von Logikblöcken (bei Xilinx CLB genannt), die von einer Anzahl konfigurierbaren Leitungen umgeben sind. Bei der Xilinx XC4000 Serie besteht ein CLB zum Beispiel aus zwei Funktionsgeneratoren, die je eine beliebige boolesche Gleichung mit vier Eingängen und einem Ausgang realisieren können, und zwei D-Flipflops.

Die gängigsten und für die Entwicklung am angenehmsten FPGA's speichern ihre Konfiguration, also die Gleichung aller Funktionsgeneratoren und die Verdrahtung aller CLB's, mit statischen RAM-ähnlichen Spei-

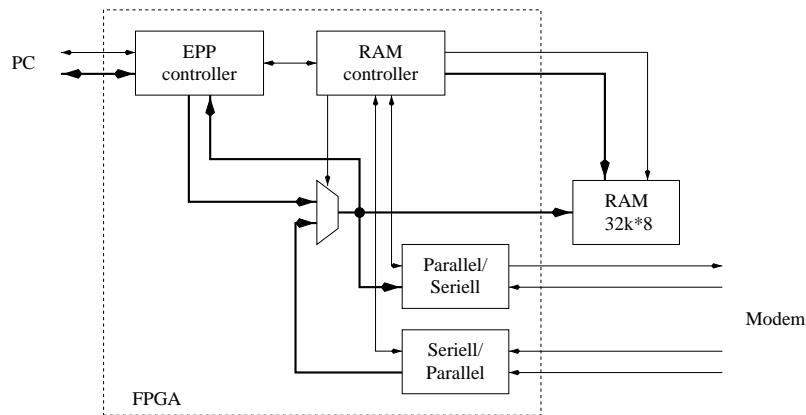


Abbildung 1: Die Schaltung

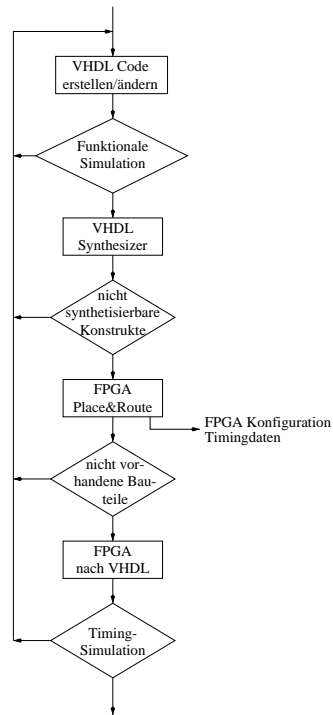


Abbildung 2: Designflow

cherzellen. Sie müssen also beim Einschalten wieder mit ihrer Konfiguration geladen werden. Dies geschieht üblicherweise direkt durch den PC, durch ein normales EPROM, oder durch spezielle serielle PROM's.

Die Marktführer in diesem Gebiet sind die Firmen Xilinx [1] und Altera [2]. Es gibt zwar noch einige andere Anbieter wie Atmel, Lucent oder Motorola, da ich jedoch im QRL nur über die Entwicklungssoftware für Altera- und Xilinx-Bausteine verfüge, habe ich mich auf diese beiden Anbieter beschränkt. Die Tabelle 1 zeigt die in Frage kommenden Bausteine und deren Einzelstückpreise, die aus einer Anfrage bei Schweizer Distributoren stammen.

Hersteller	Baustein	Preis (sFr)
Altera	EPF10K10LC84-3	61.50
	EPF10K20RC208-3	133.50
	EPF8452A	45.30
Xilinx	XC4005E-4	38.60
	XC4006E-4	57.90
	XC5204E-6	26.60
	XC5206E-6	47.80

Tabelle 1: FPGA-Bausteine und deren Preis

Obwohl Altera derzeit eine sehr aggressive Preispolitik bei den grossen FPGA's betreibt, ist diese Firma bei den kleinen Bausteinen aber nicht gerade günstig, weshalb ich mich auf die Xilinx-Bausteine konzentriere.

3 Die Schaltung

Abbildung 1 zeigt schematisch die Schaltung. Die dicken Linien sind Datenpfade, die dünnen Linien Kontrollsignale. Das RAM ist zwar viel zu gross für den Zweck, es gibt aber auch kaum mehr kleinere RAMs, ausserdem ist dieser Baustein sehr gut erhältlich.

3.1 VHDL

Wie wird nun so ein FPGA programmiert? Hier gibt es im Wesentlichen zwei Möglichkeiten:

- Schemaeingabe mit FPGA-spezifischer Komponentenbibliothek
- Hardwarebeschreibungssprache

Die erste Methode ist sehr nah an der Zielarchitektur. Man kann zwar einfach Eigenheiten der verwendeten Zielarchitektur benutzen, ist dann aber auch an sie gebunden.

Die zweite Variante ist noch relativ neu, scheint aber, da jetzt aber langsam bezahlbare Software dafür erhältlich ist, an Attraktivität zu gewinnen. Der Vorteil dieser Methode ist deren recht grosse Unabhängigkeit von der Zielarchitektur. Eine Schaltung in einer Hardwarebeschreibungssprache kann nachher im Prinzip unverändert mit jedem Logikbaustein oder sogar ASIC realisiert werden, doch so ganz trifft das Idealbild allerdings selten zu.

Die hier verwendete Hardwarebeschreibungssprache heisst VHDL. Sie wurde ursprünglich zur Simulation und Dokumentation von Digitalschaltungen entwickelt und durch die IEEE standardisiert. Erst im Nachhinein kam dann der Wunsch auf, direkt Beschreibungen in VHDL automatisch in eine Netzliste und damit in eine Implementation überführen zu können. Dies erklärt einige Eigentümlichkeiten von VHDL.

VHDL sieht auf den ersten Blick aus wie eine recht normale Programmiersprache. Im Gegensatz zu einer normalen Programmiersprache, deren Anweisungen sequentiell abgearbeitet werden, ist Hardware aber sehr parallel, d.h. alle Logikbausteine einer Schaltung arbeiten gleichzeitig. Daher besitzt auch VHDL ausgeprägte parallele Konstrukte.

Da VHDL die Wurzeln bei der Simulation hat, gibt es einige Konstrukte, die zwar für die Simulation nötig sind, sich jedoch nicht in Hardware implementieren lassen. Ein Beispiel dafür ist `wait for 10 ns;`. Dieses Konstrukt ist zwar nötig, um Stimuli für die Simulation zu generieren, lässt sich aber nicht in eine synchrone Digitalschaltung umsetzen.

Ein Programm, das den VHDL-Code in eine Netzliste umsetzt (ähnlich einem Compiler, der ein Programm in den Objektcode umsetzt), nennt man Synthesizer. Synthesizer verschiedener Hersteller variieren qualitativ recht stark; VHDL Konstrukte, die zwar eigentlich in Hardware umgesetzt werden könnten, werden nicht alle unterstützt, auch hinsichtlich der Übersetzungsgeschwindigkeit gibt es grosse Streuungen.

Abbildung 2 zeigt den typischen VHDL-Designflow. Zuerst wird die Beschreibung der Schaltung und eine sogenannte Testbench erstellt. Die Testbench ist ebenfalls ein VHDL-“Programm”, in das die Schaltung eingebettet wird und das die Umgebung der Schaltung simuliert.

Beim EPP-Adapter bestand die Testbench aus folgenden Funktionen:

- Taktgenerator für den Systemtakt
- Taktgenerator für den Modemeingang
- “Drahtbrücke” vom Modemausgang zum Modemeingang
- RAM
- PC-Simulator, der pseudozufällige Datenbits in den Adapter hineinschreibt und die empfangenen Datenbits wieder ausliest und auf Korrektheit überprüft.

Ein VHDL-Simulator¹ wird dann zur funktionalen Simulation herbeigezogen. Hier kann überprüft werden, ob die Schaltung grundsätzlich funktioniert. Es sind aber noch keine Timingdaten vorhanden – ob die Schaltung die geforderte Geschwindigkeit erreicht, kann man hier noch nicht beurteilen. Im Vergleich zur Timingsimulation ist die funktionale Simulation jedoch deutlich schneller.

Ein Synthesizer² übersetzt dann den VHDL-Code in eine Netzliste. Ein FPGA-spezifisches Place&Route-Programm generiert aus dieser Netzliste dann ein FPGA-Konfigurationsfile. Auch hier kann es Komplikationen geben: Bei der Xilinx XC4000-Serie ist es günstig, den Reset asynchron zu realisieren, da dies keine zusätzlichen Logikressourcen benötigt. Bei der XC5200-Serie gibt es aber gar keine FlipFlops mit asynchronem Reset, also lässt sich hier der Reset nur synchron realisieren.

Aus dem FPGA-Konfigurationsfile kann wieder ein VHDL-File generiert werden. Dieses VHDL-File enthält nun Timinginformationen über die realisierte Schaltung, und auch sonst hat es ausser den Ein- und Ausgangssignalen keine Ähnlichkeiten zur ursprünglichen Beschreibung mehr, modelliert aber das Verhalten des realen FPGA's recht genau. Falls die abschliessende Timingsimulation erfolgreich verläuft, kann man davon ausgehen, dass die von der Testbench getestete Funktionalität sich in der Praxis auch genau so verhalten wird.

4 Die Implementation

Vor einiger Zeit wurde eine VHDL-Beschreibung des Modems erstellt. Das Design passt aber leider nicht mehr in ein XC4006, wodurch das Projekt preislich unattraktiv und auf Eis gelegt wurde. Obwohl die XC5200er Serie schon seit längerem als Low Cost FPGA angepriesen wird, ist es Xilinx erst vor wenigen Wochen gelungen, die dafür nötige Place&Route Software fertigzustellen (Betaversionen sollen angeblich schon seit längerem verfügbar gewesen sein, haben aber nie den Weg zu uns gefunden. . .). Ein XC5204 reicht für den EPP-Adapter aus. Etwa 85% der FPGA-Ressourcen werden für den Adapter benötigt. Damit wird der FPGA-EPP-Adapter wieder interessant. Anstelle zweier FIFO's (~60 DM) kann nun ein FPGA und ein gängiges RAM (~35 DM) treten, wobei beide Adapter kompatibel zueinander sind.

Es existiert derzeit zwar noch keine Hardware, dies soll aber in naher Zukunft nachgeholt werden. Inwiefern das Projekt dann noch weiterverfolgt wird, hängt nicht zuletzt von der Preisentwicklung der betroffenen Bausteine ab.

5 Ausblick

5.1 EPP oder ECP?

Durch die grosse Flexibilität des FPGA's braucht man sich nicht auf eine kompatible Nachbildung des Originaladapters zu beschränken, es können auch Änderungen betrachtet werden.

Ein Problem des Originaladapters sind die Füllstandsbits der FIFO's. Die von den IDT FIFO's gelieferten Abstufungen sind etwas ungünstig für die Treibersoftware, können aber nicht geändert werden. Mit dem FPGA ist dies kein Problem.

Auch kann man sich überlegen, den ECP-Modus der Parallelschnittstelle anstelle des EPP-Modus zu verwenden. ECP ist zwar deutlich komplizierter zu implementieren, der PC-Schnittstellenbaustein bietet aber im ECP-Modus ein FIFO an und kann dieses sogar per DMA füllen und leeren.

¹hier VSYSTEM

²hier COMPASS AsicSyn

5.2 Modemintegration

Eine weitere Möglichkeit wäre, ein FSK-Modem mit in das FPGA zu implementieren. Der dramatische Preissprung zum nächstgrösseren Baustein lässt diese Option derzeit aber auch nicht besonders attraktiv aussehen.

Literatur

- [1] Xilinx Corporation *The Programmable Logic Data Book*
<http://www.xilinx.com>
- [2] Altera Corporation *1998 Data Book*
<http://www.altera.com>
- [3] *IEEE Standards BBS Information Technology*
<http://standards.ieee.org/catalog/it.html>
- [4] *Introduction to the IEEE 1284-1994 Standard*
<http://www.fapo.com/1284int.htm>
- [5] IDT Corporation *Specialized Memories & Modules*
<http://www.idt.com>
- [6] VHDL Tutorials
http://www.regent.e-technik.tu-muenchen.de/forschung/vhdl/VHDL_tutorials.html
- [7] Wolf-Henning Rech, DF9IC, Johannes Kneipp, DG3RBU, Gunter Jost, DK7WJ, und Thomas Sailer, HB9JNX, *Ein Modemadapter für den EPP*, 41. Weinheimer UKW-Tagung, Skriptum, 1996
- [8] <http://www.baycom.de/>