Entwurf eines Mikrocontrollers mit dem embedded Prozessorkern FHOP

Dipl.-Ing. (FH) Wolfgang Vollmer, ASIC-Design-Center
Fachhochschule Offenburg, Badstr. 24, 77652 Offenburg
Tel. 0781/205-272, Fax 0781/205-242,
E-Mail: W.Vollmer@fh-offenburg.de

Als Fortsetzung des FHOP-Projektes wurde an der Fachhochschule Offenburg auf Basis des bestehenden Mikroprozessorkerns im Rahmen einer Diplomarbeit ein Mikrocontroller in ES2-0.7 µm-Technologie entworfen. Der Controller wurde modular aufgebaut mit den Komponenten: FHOP-Mikroprozessor, Buscontroller, Waitstate-Chipselect-Einheit, 16x16 Bit Multiplizierer, 2KB ROM, 256 Byte RAM, Watchdog, PIO mit 16 konfigurierbaren Ports, SIO, 2 Timer und ein Interruptcontroller für 8 Interruptquellen.
Der Chip benötigt bei einer Komplexität von ca. 65400 Transistoren eine Siliziumfläche von etwa 27 mm^2. Er wurde im September 1996 zur Fertigung gegeben und mittlerweile erfolgreich getestet. Das interne ROM des Mikrocontrollers enthält das BIOS sowie ein Testprogramm. Zur Erstellung der Software steht eine komplette Entwicklungsumgebung zur Verfügung. Sämtliche Komponenten stehen im FHOP-Design-Kit in Kürze zur Verfügung.

1. Einführung

Ende 1994 begann man an der Fachhochschule Offenburg einen eigenen Prozessor zu entwickeln. Ziel war es, eine Steuerungskomponente zu erhalten, deren Verhalten leicht beeinflußt werden kann und somit flexibel ist. Die Verfügbarkeit der RAM und ROM-Megazellen begünstigte es, daß gleich ein kompletter Prozessor entworfen wurde (Bild 1). Gleichzeitig bekommen die Studenten Einblicke in die Funktionsweise eines Prozessors. Der Prozessor wurde 1995 gefertigt und in seiner Funktion getestet. Er verfügt über folgende wesentliche Leistungsmerkmale:

- 16 bit Architektur (16 bit ALU)
- externer 8 bit breiter Datenbus
- 64 KB Adressraum
- 6 Register
- Steuereingänge HOLD, INT und READY
- Befehle für Transfer, Arithmetik, Logik, Schieben, Sprünge und sonstige Operationen (u. a. Software-Interrupt)
- bis 50 MHz getestet, ca. 8 MIPS
- verfügbar als ES2 0,7 µm Hardmacro

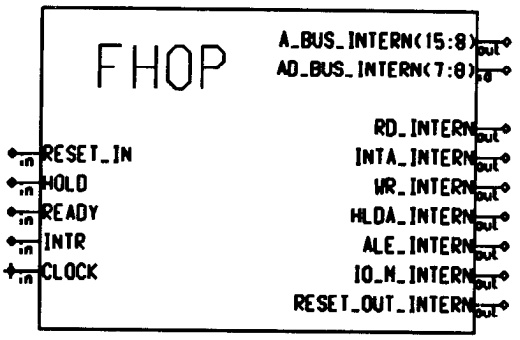

Bild 1: Symbol FHOP

Neben den bereits existierenden Megazellen für RAM und ROM werden häufig Komponenten benötigt, die eine weitere Einsatzmöglichkeit zulassen. Solche Komponenten sind vor allem:

- Parallele Ein-/Ausgabe (PIO) zur Steuerung digitaler Signale
- Serielle Schnittstelle (SIO) zur Kommunikation mit einem PC oder ähnlichem
- Zeitfunktionen (Timer) zur zeitabhängigen Auslösung von Aktionen
- Interruptcontroller zur Verwaltung mehrerer Interrupteingänge
- Watchdog zur Programmüberwachung
- Multiplizierer zur schnellen Multiplikation

Die Aufgabe bestand darin, die einzelnen Komponenten als Hardmakros zu erstellen und dann zu einem Mikrocontroller zusammenzufassen. Desweiteren wurden diese einzelnen Peripheriemodule zu einem Baukastensystem (Design-Kit) zusammengestellt und können somit auch in anderen Anwendungen schnell und einfach eingesetzt werden (Wiederverwertbarkeit).

2. Hardwareentwurf

Der Mikrocontroller besteht aus mehreren Komponenten (Bild 2). Das Herzstück bildet der Mikroprozessor FHOP. Er ist über den Buscontroller mit dem

externen Bus, d. h. der Außenwelt verbunden und kann über den internen Bus alle Peripheriemodule des Mikrocontrollers ansprechen. Die Chipselect-Einheit generiert Signale für alle internen Einheiten. Diese Signale kennzeichnen, wann der Prozessor auf die entsprechende Komponente zugreift. Desweiteren stehen acht externe CS-Signale zur Verfügung, deren aktiver Speicherbereich konfiguriert werden kann. Eine externe Adressdecodierung entfällt somit. Ebenso kann ein externer Waitstate-Generator entfallen, da dieser ebenfalls intern realisiert und getrennt für Speicher und Ports auf bis zu 14 Waitstates konfiguriert werden kann.

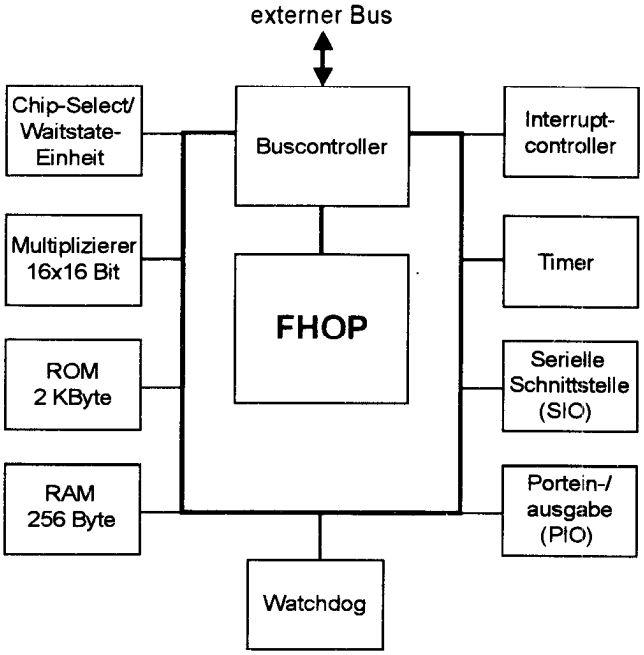

Bild 2: Architektur des Mikrocontrollers

Der Multiplizierer ermöglicht eine 16x16 Bit Hardwaremultiplikation im Integerformat. Er besteht im wesentlichen aus einer Megazelle wie auch das 2 KB ROM und 256 Byte RAM. Diese beiden Speicherbausteine werden benötigt, um das Programm, Daten, Variablen, Interruptvektoren und den Stack abzulegen.
Der Watchdog kontrolliert den Programmablauf und löst einen Reset aus, falls er nicht regelmässig getriggert wird.
Die PIO besteht aus zwei Einheiten mit jeweils 8 Bit, deren Port bitweise auf Ein- oder Ausgang konfiguriert werden können.
Die SIO entspricht dem RS232 Standard, arbeitet vollduplex bei 8 Datenbits, 1 Start- und 1 Stopbit, gerader oder ungerader Parität und verfügt über zwei Interruptausgänge, die den Empfang bzw. das Ende einer Sendebotschaft signalisieren.
Die Timerkomponente verfügt über zwei voneinander unabhängige Timer, ist über mehrere Dekaden programmierbar und kann auch zum Zählen von externen Ereignissen verwendet werden. Jeder Timer verfügt über einen Interruptausgang, der aktiviert wird, wenn die Zeit bzw. ein bestimmter Zählerstand erreicht ist.
Der Interruptcontroller besitzt 8 Eingänge, wovon einer als NMI-Eingang definiert ist. Die Eingänge sind konfigurierbar in der Freigabe, Priorität und Flanke zur Auslösung. 4 Eingänge werden bereits für SIO und Timer benötigt, die restlichen vier stehen extern zur Verfügung.
Der Entwurf sämtlicher Komponenten wurde mit dem Design Architekt auf Schematic-Ebene sowie über VHDL und anschliessender Synthetisierung durchgeführt. Das Routing des kompletten Mikrocontrollers benötigt etwa 27 mm^2 und wurde mit der ES2 0,7 µm Technologie gefertigt (Bild 3). Dabei sind die einzelnen Peripheriemodule separat als Hardmacro geroutet, können so auch in anderen Entwürfen eingesetzt werden.

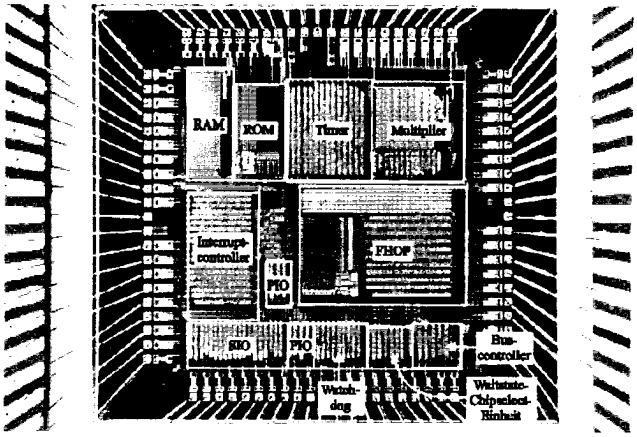

Bild 3: Routing und Chipaufteilung

3. Software des Mikrocontrollers

Im Routing integriert ist bereits ein ROM von 2 KB. Dieses Rom beinhaltet ein BIOS, das Funktionen zur Verfügung stellt, um sämtliche Peripheriemodule ansprechen zu können (Bild 4).
Häufig benötigte Funktionen sind somit bereits implementiert und können bequem auch von einem externen ROM mittels Software-Interrupt (SWI) aufgerufen werden. Die hardwarenahe Programmierung reduziert sich somit auf ein Minimum, wodurch in kürzester Zeit ein Ergebnis erreicht werden kann. Eventuell notwendige Interruptroutinen sind ebenfalls im BIOS implementiert und stellen somit Standardroutinen zur Verfügung. Sollen leistungsstärkere Interruptroutinen verwendet werden, so muß nur der Interruptvektor im RAM umgebogen werden.
Neben dem BIOS enthält das ROM noch ein Testprogramm, das mittels der BIOS-Routionen die Peripheriemodule per Software testet. Über die PIO-Ein-

gabe kann entschieden werden, welches Testprogramm aktiviert wird. Das Ergebnis kann teilweise am PIO-PORT B abgelesen werden. Mit einem Pin kann die Testfunktion disabled werden. Dann wird zu einer externen Adresse gesprungen, wo ein EPROM die Kontrolle über den Prozessor übernimmt. Somit ist der Chip auch für andere Anwendungen einsetzbar.

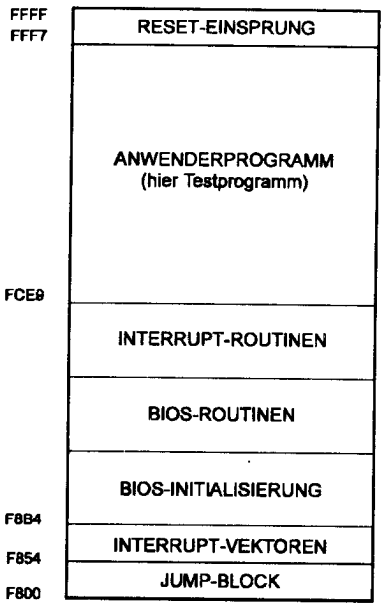

Bild 4: ROM-Aufteilung

4. Hardware-Software-Codesign

Durch die Möglichkeit, Software direkt als Megazelle, d .h. als Siliziumfläche abzulegen, entsteht eine neue Situation. Die Software muß gleich mit dem ersten Entwurf 100% korrekt sein. Eine nachträgliche Änderung ist nur über eine erneute Chipfertigung mit geändertem Programm möglich. Ist die Software falsch, kann dies ein Mißerfolg für einen 100% korrekten Hardwareentwurf bedeuten. Deshalb wurde im Rahmen von zwei weiteren Diplomarbeiten eine Entwicklungsumgebung geschaffen, die dieses Risiko auf ein Minimum reduziert. Genau wie bei der Hardware der Entwurf mittels eines Simulators (Quicksim) kontrolliert werden kann, ist es möglich, den Softwareentwurf, der mit dem Assembler erstellt wurde, mit dem Simulator zu prüfen (Bild 5 und 6). Der Simulator stellt ein abstraktes Abbild des FHOP-Prozessors dar. Es besteht die Möglichkeit, jeder aktuelle Schritt des Prozessors im Programmlisting sowie im „Trace"-Fenster zu kontrollieren. Hier werden ebenfalls asynchrone Ereignisse (Interrupt, Reset) des Prozessors erfaßt, die über spezielle Benutzereingaben aktiviert werden können. Der Programmablauf ist dabei schrittweise, im „Slow"-Betrieb oder „Run"-Betrieb möglich. Durch das Setzen von Breakpoints ist eine gezielte Programmunterbrechung möglich. In weiteren Fenstern können die Registerinhalte, der Speicher, der Stackbereich sowie die Portzustände überwacht bzw. beeinflußt werden. Somit sind alle Ereignisse, die auf den FHOP wirken bzw. von ihm ausgehen, zu erzwingen bzw. auszuwerten.

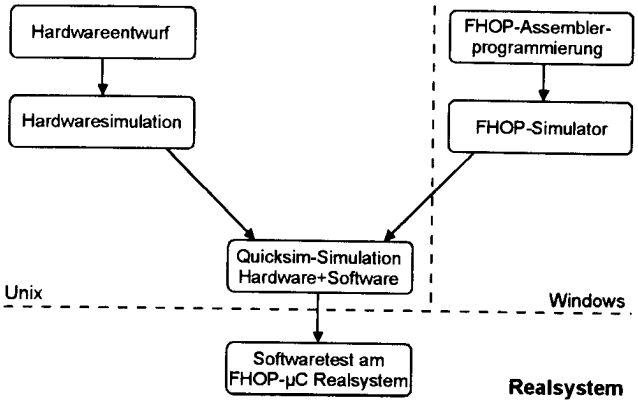

Bild 5: Simulationsübersicht (Hardware-Software-Codesign)

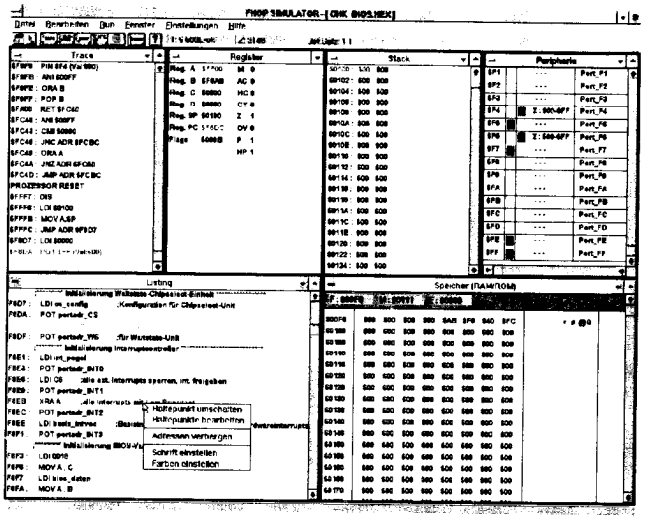

Bild 6: FHOP-Simulator

Über die DDE-Schnittstelle von Windows ist es außerdem möglich, zusätzliche Peripheriemodule (z. B. Tastatur oder LCD-Display), die über Speicher- oder Portzugriffe mit dem Prozessor in Verbindung stehen, als separate Windows-Anwendungen zu ergänzen. Diese Anwendungen sollen dann ein abstraktes Abbild des Peripheriemoduls darstellen. Dadurch ist es möglich, ein komplettes Hardwaresystem nachzubilden und die Software zu testen. Lediglich das Timing muß nochmals kontrolliert werden. Dies ist über eine Quicksim-Simulation möglich, wobei gleichzeitig die Hardware mit der Software simuliert werden kann. Die Simulation von Hardware-Software-Codesigns ist somit möglich.
Es kann in der Simulation ein komplettes Programm oder Programmteile simuliert werden und dadurch das Zusammenspiel zwischen Hardware und Software kontrolliert werden. Es können einzelne Befehle

nachverfolgt werden (Bild 7), bzw. komplette Programmteile auf ihr Resultat (z. B. PIO-Ausgabe) kontrolliert werden. Der Nachteil besteht noch darin, daß eine solche Simulation je nach Workstation und Hardwareentwurf etwa 10 Minuten benötigt, um 1 ms Programmablauf zu simulieren. Deshalb ist man bestrebt die Software an einem realem System zu testen. Dies ist nun mit der Fertigstellung des Mikrocontrollers möglich. Alle zukünftigen Projekte können nun die Software am Mikrocontroller testen und dabei schon auf die Peripheriemodule zurückgreifen. Neue Komponenten können eventuell mittels einem FPGA ergänzt werden.

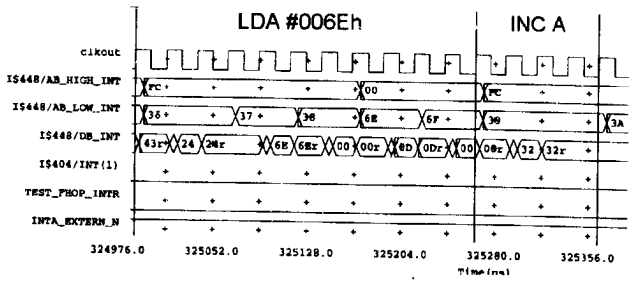

Bild 7: Einzelbefehle simulieren

5. FHOP-Design-Kit

Ist die Software am Realsystem getestet und das Konzept kontrolliert und realisierbar, kann das Projekt als Ein-Chip-Lösung verwirklicht werden. Dazu werden lediglich die Komponenten aus dem Design-Kit übernommen, die für das Projekt benötigt werden (Bild 8). Ergänzt werden sie um die anwenderspezifischen Komponenten.

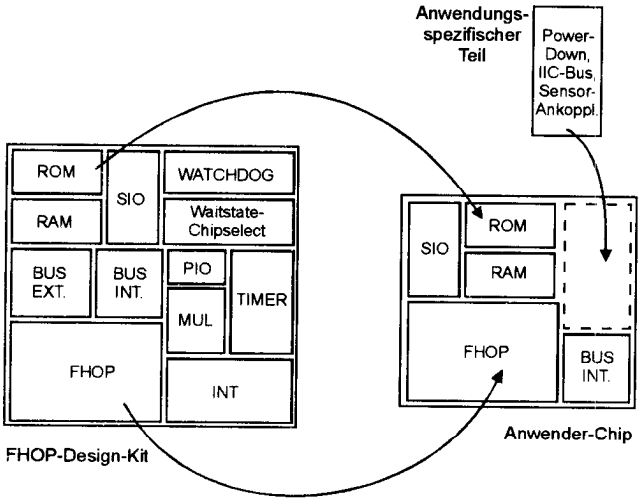

Bild 8: Baukastensytem

Ein solcher Entwurf kann soweit integriert werden, daß an externer Beschaltung lediglich eine Batterie, ein Quarz und ein Reset-Impuls nötig ist (Bild 9).

Entsprechend dem Anwendungsfall kommen weitere externe Bauteile hinzu (z. B. Tastatur, Display usw).

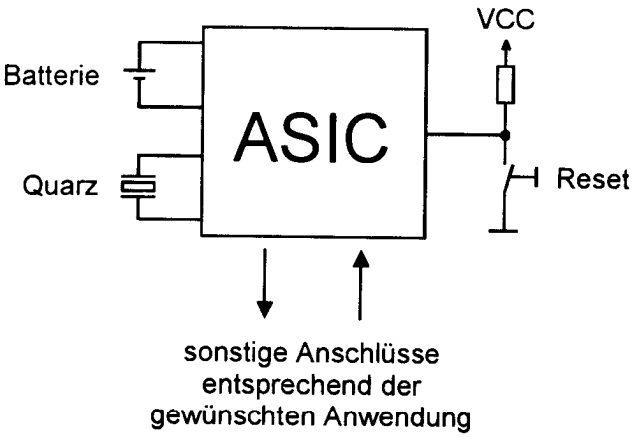

Bild 9: Ein-Chip-Lösung

Zur Zeit arbeiten wir an der Fachhochschule Offenburg an der Zusammenstellung und Dokumentation des FHOP-Design-Kits (Bild 10). Er wird in Kürze erhältlich sein. Interessenten können sich an mich oder Herrn Jansen wenden.

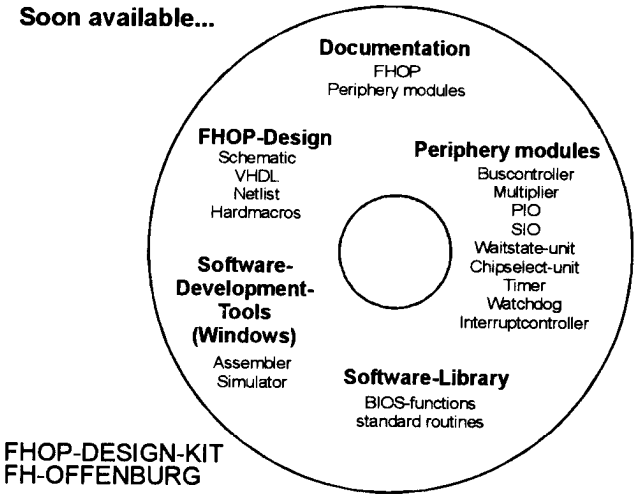

Bild 10: FHOP-Design-Kit auf CD-ROM

Literaturverweis:

[1] Aufbau eines Mikrocontrollersystems auf der Basis des Mikroprozessorkernels FHOP
Diplomarbeit W. Vollmer, FH-Offenburg, 1996

[2] Entwicklung der Steuerbaugruppe eines 16-Bit Mikroprozessor-Chips mit VHDL
Diplomarbeit Th. Gieringer, FH-Offenburg, 1994

[3] Entwicklung des Datenpfads eines 16-Bit Mikroprozessors (FHOP) mit Hilfe von VHDL
Diplomarbeit F. Zimpfer, FH-Offenburg, 1994

[4] A microprocessor in four month, Veröffentlichung auf der IEEE International ASIC Conference and exhibit, Jansen, Gieringer u. Zimpfer, 1994