RUN4
Flächenminimierter 4b-µP-Core als VHDL-Modul für die Mikrosystemtechnik und Sensorelektronik

H. Ploog, A.Wassatsch, S. Dolling, D. Timmermann

Universität Rostock
Institut für Angewandte Mikroelektronik und Datentechnik
Richard-Wagner-Str. 31, 18119 Rostock, Tel.: (0381) 498 35 34
email: hp@e-technik.uni-rostock.de

Einleitung

Ein Schwerpunkt in der Mikrosystemtechnik sind seit Jahren Smart-Sensoren. Ziel dieser Anstrengungen ist die Integration von Sensorik, Aktorik und Verarbeitung auf einen VLSI-Schaltkreis. Üblicherweise sind derartige Strukturen derart komplex, daß intelligente Steuereinheiten zum integralem Bestandteil geworden sind. Um den Aufwand für eine jeweils neu zu entwickelnde Einzellösung zu vermeiden, bietet sich die Verwendung eines kommerziell erhältlichen Prozessorkerns, also die Nutzung von fremden Intellectual Property (IP), an. Oft sind existierende Prozessorlösungen aber bzgl. der Funktionalität überdimensioniert oder bieten nicht den gewünschten Umfang von Modifikationsmöglichkeiten an. Die Verwendung von ressourcenreduzierten Elementen ist, trotz der fortgeschrittenen Technologie, immer noch ein relevantes Thema [1]. Es wird ein parametrierbares VHDL-Modul eines 4b-µP-Cores für anwendungsspezifische VLSI-Chips vorgestellt.

1 Architektur

Die wesentlichsten Abhängigkeiten bei der Neuentwicklung von Prozessorarchitekturen zeigt Bild 1. Potentielle Applikationen beeinflussen neben der Chipfläche auch die Kosten einer Neuentwicklung.

Sensorapplikationen benötigen i.a. nur einfache, häufig vorzeichenbehaftete Arithmetik (Multiplikation / Division), sowie Möglichkeiten zur Handhabung von I/O-Aufgaben. Komplexere Aufgaben, wie z.B. die Realisierung serieller Protokolle (RS232), werden statt in Hardware, durch Software gelöst. Der für die Speicherung benötigte Platz (ROM) ist kleiner als der einer direkten Implementierung. Prinzipiell gilt dies auch für aufwendigere Algorithmen mit größeren Wortbreiten.

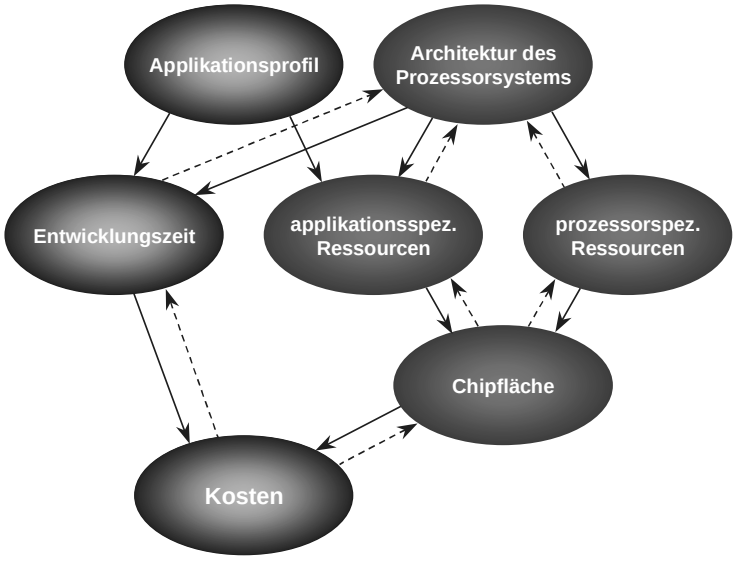


Bild 1 Entwicklungsabhängigkeiten

Zielstellung bei der Entwicklung des RUN4 war es, eine Architektur zu finden, die sowohl schnell (> 8 MIPS) als auch so klein ist (max. 1500 Gatteräquivalente[1]), daß sie die Fläche des eigentlichen Sensors nicht übertrifft. Bezüglich der Fläche besitzen speichernde Elemente eine Schlüsselposition. Die von einem einfachen D-FlipFlop benötigte Grundfläche beträgt ca. 6 GÄ, werden statt dessen Scanpath D-FlipFlops eingesetzt, erhöht sich der Flächenbedarf je nach Technologie auf 10 GÄ. Damit sind speichernde Elementen/Registern[2] restriktiv einzusetzen. Aus Geschwindigkeitsgründen ist eine RISC-ähnliche Struktur aufgrund ihrer Pipeline-Fähigkeit zu bevorzugen, eine reine RISC-Architektur scheidet aber aufgrund der großen Anzahl benötigter Register für Pipeline und Arbeitsregister aus. Andererseits erfordert eine reine Ein-Akkumulator-Maschine nur ein Register, dafür aber eine ungleich höhere Anzahl Taktzyklen, um z.B. effektiv Arithmetik betreiben zu können.
Ein weiter gravierender Nachteil von CISC- gegenüber RISC-Architekturen besteht im nicht einheitlichen Befehlsformat. D.h., die Länge der Befehle ist unterschiedlich und somit auch die Anzahl der Zugriffe auf das ROM zum Abarbeiten eines Befehls. Für kleine Architekturen ist es aus schaltungstechnischer Sicht erstrebenswert, die Anzahl derjenigen Zustände, die auf das ROM zugreifen, gering zu halten.
Nach [2] setzt sich der Kontextwechsel bei Programmen zu 15 % aus absoluten Sprüngen, zu 15 % aus CALL/RETURN-Sequenzen und zu ca. 70 % aus Verzweigungen zusammen. Für Applikationsprogrammierer muß deshalb eine große Anzahl von bedingten Programmverzweigungsmöglichkeiten realisiert werden. Weiterhin sollte u. a. vermieden werden, daß Wertetabellen während des Programmlaufes durch aufwendige Umladesequenzen aus dem ROM über den Umweg von Registern ins RAM geladen werden. Der RUN4-Kern (Bild 2) basiert deshalb auf einer modifizierten Harvard-Architektur. Er weist sowohl RISC-typische Elemente wie die Multiregister-Struktur (keine virtuellen, sondern Arbeitsregister) und einheitliches Befehlsformat als auch CISC-typische Eigenschaften wie die Verwendung komplexer Adressierungsarten für arithmetische Funktionen auf.

Die Architektur besitzt folgende Merkmale:

- Konfigurierbarkeit
- zwei Registergruppen mit je 3 bzw. 4 universellen 4b-Registern
- alternative Verwendung der Registergruppen als Akkumulatorgruppe oder Zeigerregister
- 10-Bit Instruction-Pointer
- effizienter Befehlssatz mit hohe Codedichte
- 4-Bit Flag-Register (I, C, Z, OVN)
- eine/keine Interruptebene
- maximale Kapazität des RAM : 256×4b
- maximale Kapazität des ROM : 1024×12b
- 21 Basis-Instruktionen
- 7 Adressierungsarten
- 10 unterschiedliche bedingungsabhängige SKIP-Befehle

[1] 1 Gatteräquivalent ≙ äquivalente Fläche eines 2fach-NAND

[2] RISC-typisch wäre, daß die Arbeitsregister die Breite des Adreßregisters hätten, bzw. daß das Adreßregister ein Arbeitsregister darstellt

Durch geringfügige Erweiterungen [3] an diesem Basiskonzept besteht die Möglichkeit, auch größere Datenspeicher zu adressieren.

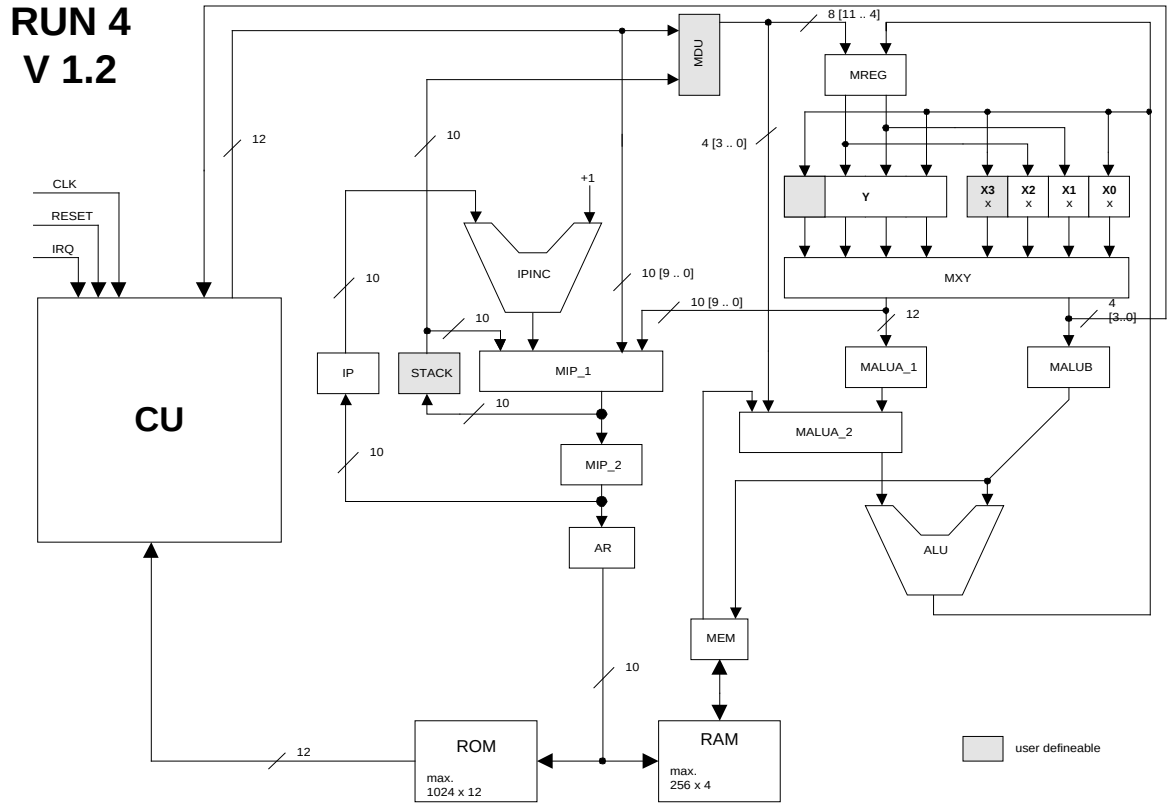


Bild 2 : Parametrierbare Basisarchitektur des RUN4

Bedingt durch das einheitliche Befehlsformat ist das Befehlsformat breiter (12b) als der Adreßraum tief (10b) ist. Dadurch können selbst bei absoluten Sprüngen über den gesamten Programmspeicher Opcode und Zieladresse in einem Zyklus aus dem ROM geladen werden.
Ein weiterer nicht unbedeutender Vorteil ergibt sich damit für das Mapping des Befehlssatzes. Durch die vorgenommene Festlegung war es möglich, bestimmten Bitpositionen im Opcode eine definierte Aufgabe zukommen zu lassen. Beim RUN4 repräsentieren die Bitpositionen 6, 5 und 4 die ALU-Befehle. Sie werden ohne Zwischenbearbeitung direkt an die ALU weitergegeben. Die Control-Unit (CU) vereinfacht sich durch diese ‘horizontale Microprogrammierung’ erheblich.

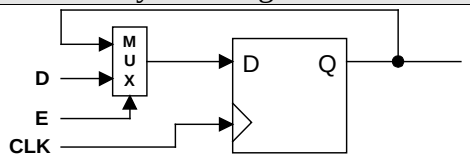
2 VHDL

Die Beschreibung des µP-Cores erfolgte ausschließlich in VHDL. Neben einer guten Simulationsfähigkeit ist damit gewährleistet, daß das Design auf unterschiedlichen Bibliotheken abgebildet werden kann (ASIC, FPGA, ...).

2.1 Einfluß auf Synthese

VHDL wird seit ein paar Jahren auch zur automatischen Synthese verwendet. Die zu erzielenden Syntheseergebnisse hängen wesentlich von der Art der Beschreibung ab [4]. Optimale Synthesergebnisse lassen sich dadurch erst dann erreichen, wenn das synthetisierende Programm und die Zielbibliothek bekannt sind. Tabelle 1 zeigt z.B. zwei unterschiedliche Beschreibungen von speichernden Elementen, beide sind formal identisch, liefern aber unterschiedliche Syntheseergebnisse. Die Version 1 wird immer auf ein D-FlipFlop mit vorgeschaltetem Multiplexer abgebildet. Version 2 hingegen kann, wenn entsprechende Elemente in der Zielbibliothek vorhanden sind, auf eben diesen abgebildet werden. Sind diese Elemente nicht vorhanden, wird das Konstrukt wie die Version 1 umgesetzt. Die synthetisierte Schaltung ist bei gleicher Funktionalität bzgl. der Anzahl Gatter kleiner und kann dadurch schneller getaktet werden.

	VHDL-Beschreibung	Syntheseergebnis

DFF-Version 1	`IF (clk'event AND clk='1') THEN`
	`  IF (a_enable='1') THEN`
	`  a_ff <= a_value;`
	`  ELSE`
	`  a_ff <= a_ff;`
	`  END IF;`
	`END IF;`
		

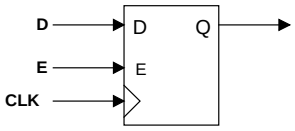
DFF-Version 2	`IF (clk'event AND clk='1') THEN`
	`  IF (a_enable='1') THEN`
	`  a_ff <= a_value;`
	`  END IF;`
	`END IF;`
		

Tabelle 1 Einfluß auf Synthese

2.2 Busstrukturen

Bussysteme bestehen aus Sender-/Empfängermodulen, die über einen Bus kommunizieren können. Die Ausgangsstufen der Sender werden zumeist als Tri-State realisiert. Außer in der Gefahr potentieller Kurzschlüsse, liegt ein weiterer Nachteil in der Testbarkeit des Chips nach Fertigstellung. Die auf Tri-State-Stufen folgende Einheiten können nur bedingt getestet werden, so daß die Gesamttestbarkeit des µP bei ca. 20..30% liegt. Verstärkt werden heute Miroprozessoren deshalb nicht mehr durch ATPG, sondern durch Abarbeitung eines Testprogramms getestet. Die Zeitdauer des Tests verlängert sich dadurch erheblich. Um die genannten Nachteile zu vermeiden, ist beim Beschreiben der internen Busstrukturen auf die Verwendung von Tri-State-Stufen komplett verzichtet worden. Statt dessen führen die Ausgänge einer Modulgruppe jeweils auf einen Multiplexer, der dann den gewünschten Ausgang durchschaltet (1-aus-n-Dekoder). Durch konsequente Anwendung der Transformation liegt die Testbarkeit des Designs bei 98.5%.

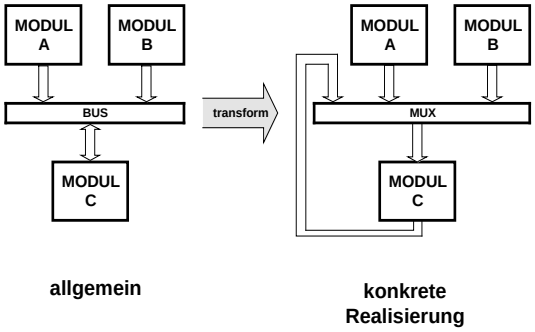


Bild 3 Busstrukturen

3 Parametrierbarkeit

Um den Anforderungen minimaler Applikationen gerecht zu werden, ist es möglich, die Anzahl der Arbeitsregister der Registergruppen wahlweise auf drei bzw. vier einzustellen. Dies hat bei besonders ressourcenkritischen Applikationen den Vorteil, daß ganz auf externes RAM verzichtet werden kann. Um die Parametrierbarkeit zu vereinfachen, wurden alle von der Anzahl der Register abhängigen Komponenten als generische VHDL-Module beschrieben. Das Ein- bzw. Ausschalten bestimmter Eigenschaften (Interrupt/SlowRAM/#Register) erfolgt jeweils durch Auswertung der entsprechenden Schalter. Eine nachträgliche Modifikation der eigentlichen Quellen wird dadurch vermieden, wodurch ein einfacher Reuse des Cores gewährleistet ist. Die Basisarchitektur kann so auf effiziente Weise modifiziert werden. Wird in einer gegebenen Applikation z.B. kein externer Interrupt benötigt, kann die Chipfläche um ca. 25% reduziert werden.
Die Einstellmöglichkeiten sind im folgendem:

- Anzahl der Register einer Registergruppen auf 3 bzw. 4
- Ein- bzw. Ausschalten des Interruptsystems
- SLOW_ROM, SLOW_ROM
- ROM_ENABLE (lowpower-Applikationen)
- direktes Lesen und Schreiben des Stacks

4 Softwaretools

Zur Verifikation der Performance des RUN4 zu vermitteln, wurde in [5] eine Reihe von Test- / Benchmarkprogrammen entwickelt. Diese realisieren einfache Arithmetik (Multiplikation / Division) bis hin zur Implementierung eines UART-Moduls zur seriellen Kommunikation. Speziell für Sensorapplikationen wurden Funktionen zur linearen Interpolation realisiert. Das Testen der einzelnen Programmsequenzen erfolgte dabei mit einer State-of-the-art Programmierumgebung unter Windows-NT, die in [7] entwickelt wurde. Der Simulator entspricht dem heute üblichen Industriestandard und besitzt alle Möglichkeiten einer debug-Umgebung wie Einzelschrittabarbeitung, Setzen und Löschen von Breakpoints, Disassembler usw. Der Assembler-Code kann auch direkt in eine VHDL-Beschreibung überführt werden. Eine Einbindung in die Simulationsumgebung des zu synthetisierenden µP-Cores wird damit möglich. Offline entwickelte Programme können so zur Backannotation benutzt werden.

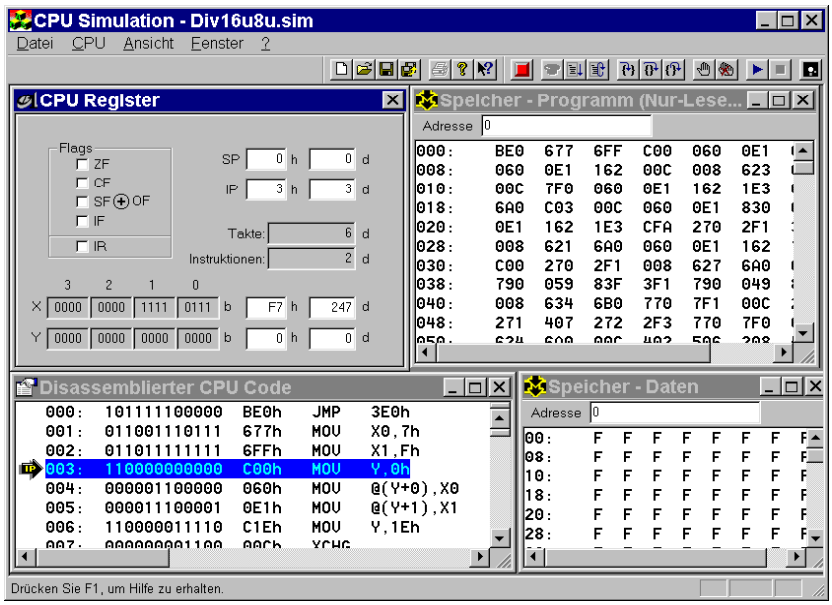


Bild 4: Simulationsumgebung unter NT/Win95

5 Zusammenfassung

Smart-Sensor-Applikationen verarbeiten die Meßdaten in unmittelbarer Nähe des eigentlichen Sensors [6]. Das bedingt das Vorhandensein eines Prozessors auf dem Sensorchip. Um eine optimale Anpassung des Kerns an die zu lösenden Aufgaben zu ermöglichen, sind beim Entwurf des RUN4 die Zielapplikationen in die Spezifikation eingeflossen. Dadurch ist eine minimale, optimal auf die Lösung von Smart-Sensor-Applikationen zugeschnittene Architektur entstanden. Exemplarisch wurde das Design auf eine ASIC-Bibliothek[3] (Bild 5) und auf einem XILINX-FPGA abgebildet. Ein Befehl benötigt durchschnittlich 2 Taktzyklen. Im ASIC beträgt die max. Taktrate ca. 30 MHz, der Durchsatz beträgt ca. 12 MIPS. Die Designvorgaben sind damit deutlich unterschritten worden. Für das FPGA ist eine Prototypen-Entwicklungsumgebung realisiert worden. Durch Wortbreitenkonverter (4 bit ⇔ 8 bit) es möglich, bereits vorhandene Peripherie (I/O, A/D, D/A) an den Core anzuschließen. Die nächsten geplanten Schritte sind Entwicklungen von eigenen softwareunterstützenden Komponenten (Timer, Zähler usw.). Des weiteren ist die HW-Integration unterschiedlicher serieller Busprotokolle, wie z.B. I^2C, RS232 und für Chipkarten T0 und T1 geplant.

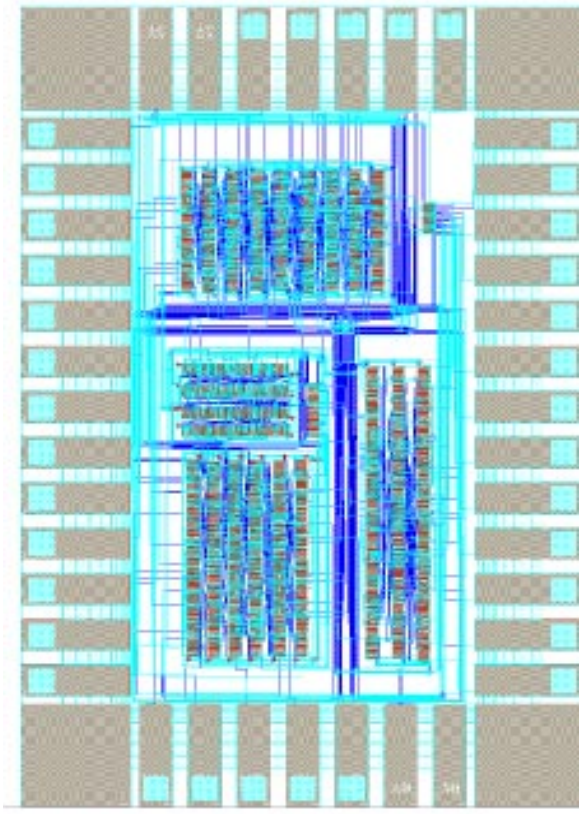


Bild 5 Layout des RUN4

6 Literatur

[1] W. Brockherde, D. Hammerschmidt, B.J. Hosticka: Silicon microsystems for mechatronic applications. Conference on Mechatronics and Robobtics, Teubner (1995),Proceedings S. 446-455

[2] John L. Hennessy und David A. Patterson: Computerarchitecure : A quantitative approach, Morgan Kaufmann, zweite Auflage, 1996

[3] E. Woitzel, H.Ploog : Erweiterungsmöglichkeiten der RUN4-Architektur, Interner Bericht, Uni Rostock,1996

[4] M. Selz : Untersuchungen zur synthesegerechten Verhaltensbeschreibung mit VHDL, Dissertation, Universität Erlangen, 1994

[5] B. Katschke : Benchmarksuite für den Microcontroller RUN4, Kleiner Beleg, Uni Rostock, 1997

[6] P.Schulmeyer: Intelligente Sensoren. Design & Elektronik, Nr. 20/96

[7] F. Papenfuß: Evaluierung minimaler Prozessorarchitekturen, Diplomarbeit, Uni Rostock, 1996

[3] ES2-Bibilothek 1 µm : 1300 GÄ, Fläche : ca. 2.1 mm^2 (je Bank 3 Register, Interruptsystem)