HARDWAREUNTERSTÜTZUNG FÜR JAVACARDS: DER JAVAPROZESSOR JSM

Frank Golatowski, Nico Bannow*, Dirk Timmermann

Universität Rostock
Fachbereich Elektrotechnik u. Informationstechnik
18119 Rostock, Richard-Wagner-Str.31
Email: Frank.Golatowski@uni-rostock.de

*Infineon Technologies, CMD DSP
Postfach 80 09 49
81609 München
Email: Nico.Bannow@infineon.com

KURZFASSUNG

In diesem Beitrag wird der am Institut für Angewandte Mikroelektronik und Datentechnik entwickelte 32-Bit-Java-Prozessor JSM beschrieben. Bei diesem Prozessor handelt es sich um eine der kleinsten Implementierungen eines Javasystems für den mobilen Bereich. Der Prozessor wurde als SmartCard-Prozessor (Javacard) bzw. für den Einsatz in eingebetteten Systemen entwickelt. Die realisierte Funktionalität entspricht der SUN-Javacard-Spezifikation 2.1.1. Die in diesem Standard definierten Java-Bytecode-Befehle sind vollständig implementiert. Der Prozessor-Kern ist vollständig synthetisierbar. Er wurde in einem FPGA der Virtex1000E-Reihe implementiert und seine Funktionsweise mit dem Rapid-Prototyping-System Aptix MP3 nachgewiesen.. Der vorliegende Java-Prozessor-Kern benötigt in einem 0.18u Prozess eine Fläche von weniger als 0.2 mm².

1. EINFÜHRUNG

Die Mobilfunkanbieter beginnen Java-basierte Mobildienste anzubieten. Diese Dienste erfordern, dass die eingesetzten mobilen Systeme Java-Befehle ausführen. Dadurch wird u. a. der Bedarf an Prozessoren, die direkt JAVA-Code ausführen, in den nächsten Jahren stark anwachsen. Ein wesentlicher Nachteil bisheriger Java-Systeme liegt häufig im schlechten Laufzeitverhalten. Dies liegt u.a. daran, dass die Java-Programme erst durch eine virtuelle Maschine interpretiert werden müssen. Wird eine solche virtuelle Maschine jedoch in

Hardware umgesetzt, ist eine enorme Leistungssteigerung möglich [03, 04]. Entsprechend dem erwähnten Trend wurden insbesondere in den letzten beiden Jahren verschiedenen Prozessoren vorgestellt. Die Firma ARM hat mit der dreisprachigen Jazelle ARM926EJ-S [01] ihre Migrationslösung vorgestellt. MIPS Technologies bietet den MIPS32™4Ksc [05] und SGS Thomsen den SmartJ [09] an. Diesen drei Lösungen ist gemeinsam, dass sie sich in das entsprechenden Firmenportfolio einbetten. JStar und JSmart von der Firma Nazomi [08] sind Java-Coprozessoren, mit denen prinzipiell jeder andere Prozessor um die Fähigkeit, Java in Hardware auszuführen, erweitert werden kann. Die Firma XILINX bietet als einer der führenden FPGA-Hersteller seit Anfang 2001 den synthetisierbaren Java-Kern LavaCore [10] an.

2. AUFBAU DES JAVA-PROZESSORS JSM (JAVA SILICON MACHINE)

Der Java Prozessor (JSM) (Abb. 1) ist modular aufgebaut. Der Prozessor wird in mehrere lose gekoppelte Module (FSMs) aufgeteilt, um ein strukturiertes, leicht zu änderndes und einfach zu testendes Modell zu erhalten.

Der JSM-Prozessor ist durch folgende **Eigenschaften** charakterisiert:

- vollständige Implementation des Javacard-Standards
- sehr kleines Design und modularer Aufbau
- konfigurierbar und an jeweiligen Anwendungsfall adaptierbar
- hardwareunterstützte Sicherheitsmechanismen
- Anschluss zusätzlicher Peripherieeinheiten.

Herausragende Eigenschaft des Prozessors ist seine minimale Größe und die Möglichkeit des Ausbaus und der Erweiterung um zusätzliche Peripherieeinheiten. Dadurch ist er ebenso in kleinen eingebetteten Systemen einsetzbar. Exemplarisch wurden in dieses Design ein Triple-DES- Kryptographieprozessor und I^2C-Interface angeschlossen.

Der Prozessor besteht aus den **Modulen**

1. Control-Unit (CU),
2. Arithmetic Logical Unit (ALU),
3. Memory Management Unit (MMU),
4. Stack und

5. Input/Output-Unit (I/O-Unit)

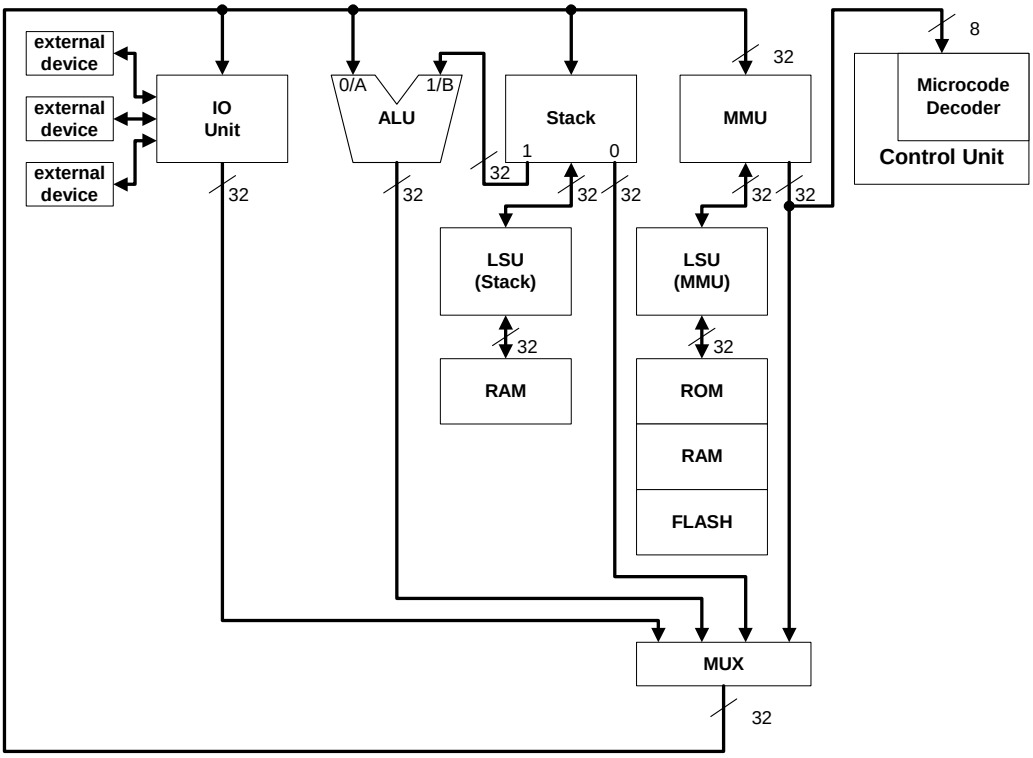

Abb. 1: Aufbau des JSM-Prozessors

2.1 CONTROL-UNIT

Die Control-Unit hat die Aufgabe alle untergeordneten Einheiten zu steuern. Die Abarbeitung von Bytecode ist grundsätzlich mikrocodebasiert. Dadurch muss die Programmausführung nicht mittels einer aufwendig zu implementierenden festverdrahteten Schaltung erfolgen. Durch die mikrocodegesteuerte Strategie sind sowohl eine flexible, schnelle Implementierung und Anpassung an Vorgaben als auch ein einfaches Debugverhalten zur Fehlerbehebung möglich. Die Aufgabe der Control-Unit ist es, den Mikrocode ordnungsgemäß zu laden und auszuführen. Die Control-Unit arbeitet die Mikrocodebefehle schrittweise ab. Dazu werden aus einer Tabelle die Befehle für die zu steuernden Module ausgelesen und an diese für die Dauer eines Taktes weitergeleitet. Danach wird solange der Befehl NOP (no operation) ausgeführt, bis die Module ihre Befehlsausführung beendet haben. Anschließend beginnt der Zyklus von neuem: ein neuer

Mikrocodebefehl wird geladen und ausgeführt. Für jeden Java-Bytecodebefehl muss mindestens ein Mikrocodebefehl abgearbeitet werden. Dies schließt den Befehl NOP mit ein. Die Anzahl der notwendigen Mikrocodebefehle ergibt sich aus der Komplexität des jeweiligen Java-Bytecodes.

2.2 ALU

Das Rechenwerk (ALU) dient dem Ausführen von Grundrechenoperationen. Die Operationen umfassen die Addition, Subtraktion, Negation, Multiplikation, Division, Schiebeoperationen, logische Operationen, Testoperationen sowie Vergleichsoperationen. Die ALU enthält zwei Register, die die Eingangsoperatoren speichern. Dazu müssen beide Register nacheinander oder in einem Zyklus geladen werden. Erst dann ist eine (sinnvolle) Ausführung der arithmetischen oder logischen Funktion möglich. Die ALU erhält die Operanden als vorzeichenbehaftete 32 Bit-Werte.

2.3 MMU

Die Memory Management Unit (MMU) dient der gesamten Speicherverwaltung der JSM mit Ausnahme der Stackframes, die die Stack-Einheit eigenständig verwaltet. Die MMU kann verschiedene Speichertypen ansprechen, die der Speicherung von Betriebssystem, Applets, Klassenvariablen und Objekten dienen. Die MMU kann theoretisch, d.h. in Abhängigkeit von der Größe des vorhandenen Speichers, dem Java Standard entsprechend, 2^{32} Bit adressieren.

2.4 STACK

Der Stack dient der Zwischenspeicherung von Variablen einer Methode bzw. der Übergabe von Daten an eine Methode. Des weiteren werden bei Methodenaufrufen prozessorinterne Daten abgespeichert, wie der Programmzähler (PC), die aufrufende Methode und die Klasse der aufrufenden Methode. Die derzeitige Implementierung sieht eine Begrenzung des Stacks auf $2^{10} = 1024$ Einträge vor.

2.5 IO- UNIT

Die IO- Unit dient der externen Kommunikation des Prozessors über entsprechende Eingabe- und Ausgabegeräte (IO- Geräte), die nicht im Standard von Java definiert bzw. nur über die jeweilig entsprechende API ansprechbar sind. Die IO- Unit stellt die Schnittstelle des

Prozessors zu den jeweiligen Peripherie- Geräten dar. Diese sind zwingend notwendig, da ein Ablauf der Programme auf der Karte ohne externe Kommunikation keinen Sinn macht. Auf die externen Geräte darf nur vom Betriebssystem bzw. von dessen API aus zugegriffen werden. Eine weitere Verwendungsmöglichkeit der IO- Unit ist die Ansteuerung zusätzlicher Hardware, die der Beschleunigung interner Berechnungen (z.B. kryptographischer Koprozessor, Floating Point Unit) oder aufwendiger interner Operationen (z.B. Garbage Collector) dient.

3. RAPID-PROTOTYPING UND ENTWICKLUNGSUMGEBUNG

Der Funktionsnachweis wurde in der in Abb. 2 dargestellten Rapid-Prototyping-Umgebung erbracht [02]. Ausgehend von der VHDL-Beschreibung wurde das Design unter Synopsys synthetisiert und mit den XILINX-M3-Tools auf das FPGA abgebildet. Das Design wird nachfolgend mittels Aptix-System-Explorer auf das Rapid Prototyping System MP3C in das Virtex1000-FPGA geladen. Damit befindet sich der Java-Kern im ausführungsbereiten Zustand. Die Java-Applets werden als Bytecode im externen Speicher (Memory Extension Board) abgelegt. Die Applets können sowohl im ROM als auch im Flash abgelegt werden. Ebenso besteht die Möglichkeit, dass die Java-Applets von einem PC (Terminal) über ein I^2C-Interface in den Speicher geladen werden. In einer Beispielanwendung befindet sich das Applet bereits im ROM. Der erforderliche Bytecode wurde fest in den EPROM abgelegt.

4. ZUSAMMENFASSUNG

In diesem Artikel wurde der Javaprozessor JSM vorgestellt, der den Javabytecode direkt ausführt. Der Prozessor entspricht dem Javacard-Standard 2.1.1. [07]. Der Funktionalitätsnachweis wurde auf dem Aptix Rapid-Prototyping-System MP3 erbracht. Der Einsatz von Java erlaubt eine neue Qualität im Erstellen komplexer und doch überschaubarer Programme. Problematisch ist u.U. der hohe Speicherbedarf, die durch die objektorientierte Programmiersprache erforderlich ist. Allerdings wird die Chipfläche mit modernen Technologien immer besser ausgelastet, so dass mehr Fläche für Speicher und Logik bereitsteht. Der vorliegende Java-Prozessor-Kern benötigt in einem 0.18µ Prozess eine Fläche von weniger als 0.2 mm². Die Mikrocodetabellen sind in dieser Betrachtung nicht berücksichtigt.

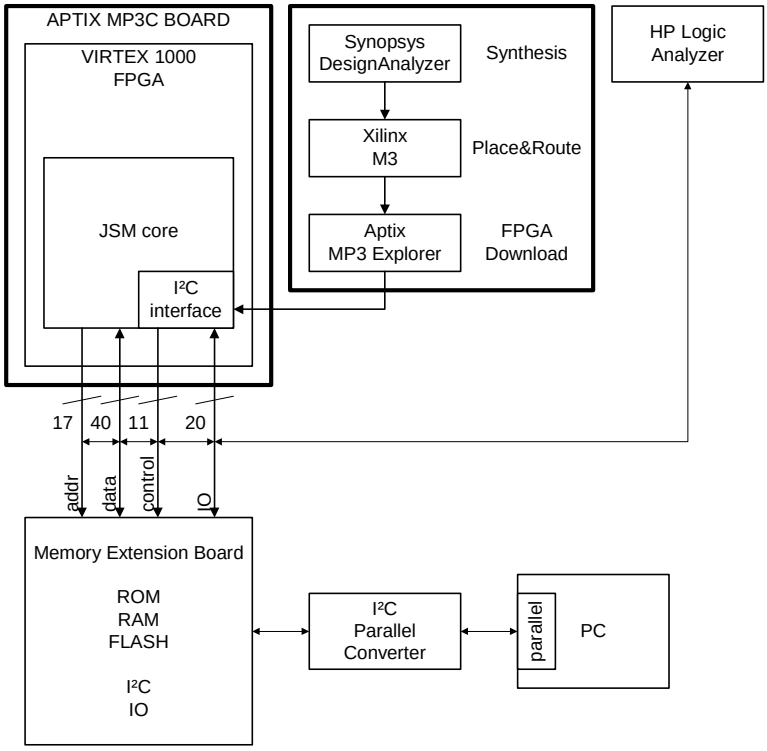

Abb. 2: Rapid-Prototyping

5. LITERATURANGABE

[01] Jazelle ™ – ARM® Architecture Extensions for Java Applications, White Paper, www.arm.com

[02] N. Bannow, Java-Prozessors für Smartcards und kleine eingebettete Systeme. Diplomarbeit, Institut für Angewandte Mikroelektronik und Datentechnik, Universität Rostock, Dez. 2000

[03] H. Ploog, R. Kraudelt, N. Bannow, T. Rachui, F. Golatowski, D. Timmermann: A Two Step Approach in the Development of a Java Silicon Machine (JSM), Workshop on Hardware Support for Objects And Micro architectures for Java. Austin, Texas, Oktober 1999

[04] F. Golatowski, H. Ploog, R. Kraudelt, T. Rachui, O. Hagendorf: Java Virtual Machines für ressourcenkritische eingebettete Systeme und SmartCards, Java Informationstage JIT 99, ITG/GI-Fachtagung, Düsseldorf, September 1999

[05] MIPS Technologies, MIPS Technologies bringt neuen Prozessor- Core für Smartcard-Anwendungen mit besonders niedriger Leistungsaufnahme auf den Markt, Pressemitteilung, www.mips.com/pressrelease/022001H.html.Feb. 2001

[06] Sun Microsystems, Inc., Java Card Technology Home Page http://java.sun.com/products/javacard.,

[07] Sun Microsystems Inc., JavacardTM 2.1.1 Virtual Machine Specification, 2000

[08] Java Coprocessor, Product brief, http://ww.nazomi.com

[09] Instant Java for your smartcard, SGS Thomson, www.st.com/stonline/prodpres/smarcard/insc9901.htm

[10] Bhaskar Bose, M. Esen Tun, LavaCORE- A Configurable Java Processor, http://www.xilinx.com/xcell/xl37/xcell37_20.pdf