Processors for mobile devices

Christopher Pockrandt
Department of Computer Science
Freie Universität Berlin, Germany

23. Juni 2011

Zusammenfassung

Diese Arbeit gibt eine Einführung in aktuelle Prozessor-Architekturen für Mobile-Internet-Devices (MIDs), wie z.B. Smartphones oder Tablet-PCs. Sie stellt die Anforderungen an Prozessor-Architekturen heraus, vergleicht die Architektur des ARM Cortex-A8 mit der des Intel Atom Z510 und zeigt auf, inwiefern sie die Architekturziele erfüllen. Der Schwerpunkt liegt dabei auf dem Pipelining, insbesondere der Sprungzielvorhersage. Anschließend werden aktuelle Entwicklungen und Trends in diesem Markt diskutiert.

Besonderer Dank gebührt dem Betreuer dieser Arbeit, Herrn Prof. Dr.-Ing. Jochen Schiller, für die mehrfachen Rücksprachen und sein Feedback.

1 Motivation

In den letzten Jahren ist die Entwicklung von MIDs enorm vorangeschritten. Die Anforderungen an Smartphones und jüngst auch Tablet-PCs sind stark gestiegen, sodass mittlerweile ähnlich anspruchsvolle Anwendungen auf den Endgeräten laufen wie auf Desktop-Rechnern: 3D-Spiele und Flashanimationen gehören längst zum Alltag eines Mobiltelefones.

Die Hardware eines Desktop-Rechners lässt sich auf Grund ihrer Größe und ihres Stromverbrauches nicht in ein Smartphone integrieren. Doch neben dem Platzverbrauch und der Leistungsaufnahme gibt es noch weitere Anforderungen, die an die Mikroprozessoren gestellt werden. Diese Arbeit macht anhand der Pipeline mit dem Schwerpunkt auf der Sprungzielvorhersage deutlich, welche Ansätze Intel und ARM bei der Entwicklung verfolgen und inwiefern sie diese Anforderungen erfüllen. Grund für diese Fokussierung ist einerseits die starke Relevanz der beiden Aspekte und andererseits die in diesen Punkten zum Teil sehr detaillierten Dokumentationen der Hersteller. Anschließend wird mit dem ARM Cortex-A15 ein Ausblick darauf gegeben, was die Nutzer und Softwareentwickler in Zukunft erwarten wird.

2 Anforderungen [1]

Leistung Smartphone-CPUs verfolgen das gleiche Ziel wie Desktop-Rechner: ein Maximum an Leistung, das sich mit den anderen Zielen vereinbaren lässt, um Multitasking von Internet-, Audio- und Video-Anwendungen zu ermöglichen. Die Leistung von Einkernprozessoren ließ sich im Desktop-Bereich durch die Erhöhung der Taktfrequenz nicht unendlich steigern, da ab Frequenzen von 4 GHz Probleme mit der Wärmeabfuhr entstanden. Um den steigenden Anforderungen weiterhin gerecht zu werden und mit dem Gesetz von Moore mithalten zu können, wurden für Desktop- und Server-Systeme Mehrkernprozessoren eingeführt. Dieser Trend läßt sich seit Kurzem auch bei MIDs erkennen. Hier liegt die Grenzfrequenz allerdings weitaus niedriger; aktuelle Einkernprozessoren haben die Grenze von 2 GHz noch nicht überschritten. Um die Komplexität im Detail gering zu halten, behandelt diese Arbeit mit Ausnahme des letzten Abschnittes ausschließlich Single-Core-CPUs.

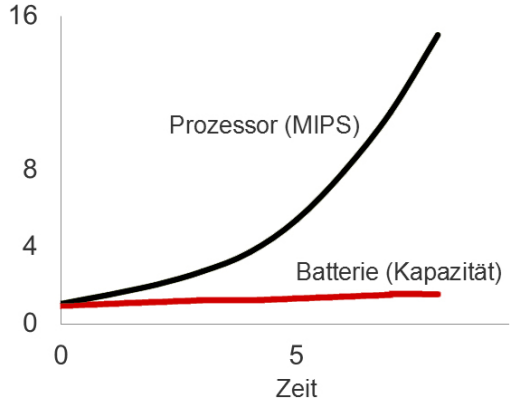

Abbildung 1: Technologiefortschritte der Komponenten im Vergleich (zum Jahr 0)

Energieeffizienz Wie *Abbildung 1* zeigt, hat sich die Batterie-Kapazität im Vergleich zu den Prozessoren nicht wesentlich verbessert. Während die Maßeinheit *Millionen Instruktionen pro Sekunde* (MIPS) stark angestiegen ist, konnten die Akkus den eigentlichen Anforderungen nicht entsprechen. Umso wichtiger ist es, dass die Prozessoren mit den Ressourcen sparsam umgehen.

Wärmeentwicklung Da auf dem engen Raum die Komponenten sehr dicht beieinander sitzen und kein Platz für Lüfter vorhanden ist, sind Mikroprozessoren in ihrer Leistung deutlich beschränkt. Eine zu hohe Wärme, die nicht abgeführt werden kann, könnte umliegende Komponenten beschädigen oder sogar für den Anwender gefährlich werden, insbesondere dann, wenn der Akku zu hohen Temperaturen ausgesetzt ist.

Weitere Punkte, die in diesem Zusammenhang eine Rolle spielen, sind die **Größe** und das **Gewicht** der Prozessoren. Die Größe bezieht sich in der Regel nicht auf den ganzen Prozessor, sondern nur auf den Halbleiterchip ohne Gehäuse, auch Die genannt.

Integrationsgrad Der Integrationsgrad setzt sich zusammen aus der Integrationsdichte (Anzahl der Transistoren pro Flächeneinheit) und der Größe des Chips. Je höher der Integrationsgrad bei gleichbleibender Größe des Dies, umso mehr Platz steht beispielsweise für größere oder zusätzliche Caches zur Verfügung.

Diese vorgestellten Anforderungen sind eng miteinander gekoppelt. Energieeffizienz und Wärmeentwicklung sind komplementäre Ziele: Ist die Leistungsaufnahme geringer, entsteht weniger Abwärme. Zusätzlich gilt: Je kleiner die Strukturgröße, desto geringer die Leistungsaufnahme, da schmalere Leiterbahnen eine geringere Stromstärke erfordern. Gleichzeitig reduziert sich auch die Größe des Dies. Bis jetzt steht dem nur die Leistung als konkurrierendes Ziel gegenüber. Zusätzlich zu diesen Zielen gibt es noch andere Faktoren:

Komplexität Je nach dem zu entwickelnden Produkt kann es sich anbieten einen Prozessor zu wählen, der gewisse Funktionen in Hardware realisiert hat, um so komplexe Berechnungen schneller ausführen zu können, oder aber einen weniger komplexen, der ein schlankes Design hat. Würde beispielsweise ein Hersteller ein Smartphone mit einer Verschlüsselung des nichtflüchtigen Speichers auf den Markt bringen wollen, könnte es sich hinsichtlich der Leistungsfähigkeit lohnen einen Prozessor zu wählen, der entsprechende Verschlüsselungsmethoden bereits in Hardware implementiert hat.

Wiederverwendbarkeit des Designs Beim Entwickeln eines neuen Prozessors wird darauf Wert gelegt ihn so zu entwerfen, dass sich möglichst viele und große Teile davon in späteren Modellen wiederverwenden lassen. So können Kosten für erneute Verifikationen eingespart werden.

Kosten Zum Abschluss seien noch die Lizenzgebühren und Herstellungskosten erwähnt, die insbesondere in der Massenproduktion eine wichtige Rolle spielen.

3 Aktuelle Architekturen

Im Folgenden werden zwei unterschiedliche Prozessoren vorgestellt: Der ARM Cortex-A8 aus der Application-Serie als Vertreter der RISC-Architekturen und der Intel Atom Z510 stellvertretend für die CISC-Prozessoren, basierend auf der Intel x86-Architektur. Ein entscheidender Grund für diese Wahl ist, dass die Hersteller aus verschiedenen Segmenten kommen: ARM aus dem Embedded-Markt, deren Hauptaugenmerk auf einem geringen Stromverbrauch der Prozessoren liegt, und Intel aus dem leistungsorientierten Desktop- und Server-Bereich. Beide versuchen mit unterschiedlichen Ansätzen den Anforderungen gerecht zu werden.

Der Cortex-A8 ist der aktuellste Einkernprozessor von ARM, der A9 und der A15 sind Mehrkernprozessoren – letzter wird im Kapitel 4 vorgestellt. Der Cortex-A9 ist zwar auch als Single-Core-CPU erhältlich, als solcher allerdings weniger weit verbreitet. Der Z510 wird hier als Vergleich gewählt, da die Z-Serie eigens für MIDs eingeführt wurde und es sich bei dem Modell um das erste dieser Serie handelt. Zudem ist der Z510 der einzige Prozessor seiner Familie auf dem nur 1 Thread zur gleichen Zeit laufen kann, was die Sprungzielvorhersage im Detail weniger komplex werden lässt. Es ist aber zu beachten, dass der A8 mit der Veröffentlichung Ende 2005 etwa 2 Jahre älter ist als der Z510. Einen vom Erscheinungsjahr vergleichbaren Prozessor aus der Z-Serie von Intel gibt es nicht. Da Intel keine Angaben zu Unterschieden in der Pipeline und den Sprungvorhersagen innerhalb der Atom-Familie macht, spielt die Serie und das Modell nur eine sekundäre Rolle.

3.1 ARM Cortex-A8 [2][3][4][5]

Bei der A-Serie handelt es sich nach Angaben des Unternehmens ARM Limited um eine Modellreihe, die für betriebssystembasierte Systeme, wie z.B. Smartphones oer Tablet-PCs, geeignet sind. Der Name ARM (**A**dvanced **R**ISC **M**achines) lässt schon vermuten, dass es sich um einen RISC-Prozessor handelt. Der kleinere Befehlssatz ermöglicht durch einen niedrigeren Dekodieraufwand eine hohe Ausführungsgeschwindigkeit der Instruktionen und somit eine geringere Befehlslatenz.

Die RISC-Architektur hat noch weitere positive Auswirkungen: Durch den kleineren Befehlssatz müssen weniger Funktionen in Hardware implementiert werden. Dies erfordert weniger Transistoren. So ist der Prozessor einerseits kleiner, stromsparender und in der Herstellung kostengünstiger, andererseits hat er ein schlankeres Design, das leichter zu optimieren und anzupassen ist. Die statische Befehlslänge bei RISC-Architekturen ermöglicht eine effiziente Auslastung der Pipeline, weshalb in den folgenden Abschnitten auf das Pipelining – mit besonderem Augenmerk auf der Sprungzielvorhersage – eingegangen wird:

Im Gegensatz zu den Vorgängern der ARMv7-Architektur hat diese eine Dual-Issue Integer Pipeline, die eine Ausführung von 2 Befehlen gleichzeitig ermöglicht. Dadurch erhöht sich die Anzahl der ausgeführten Befehle pro Zyklus deutlich. Wie sich in *Abbildung 2* erkennen lässt, sind die 13 Stufen in 3 Kategorien eingeteilt: Instruction Fetch (F), Instruction Decode (D) und Instruction

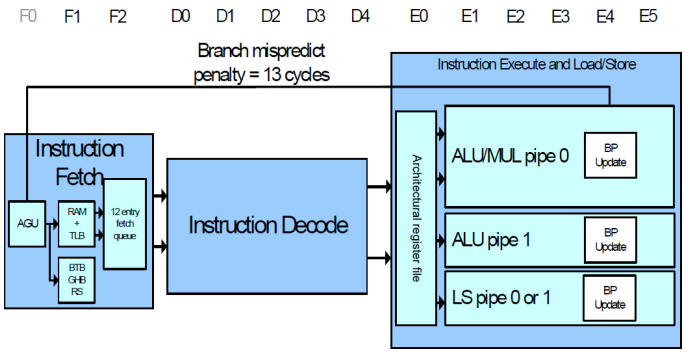

Abbildung 2: Cortex-A8: Dual-issue, in-order Integer Pipeline[8]

Execute (E). Der Pipeline stehen mehrere Funktionseinheiten zur Verfügung: 2 ALU-Pipelines, eine Multiplikations-Pipeline, sowie eine Load-Store-Pipeline. Die Address-Generation-Unit (F0) wird nicht mitgezählt, da der Zugriff auf den Instruction Cache stets als die 1. Stufe bezeichnet wird.

Die verwendete Harvard-Speicher-Architektur trennt Daten- und Programmcode und eignet sich so besonders für Out-of-Order-Execution. Dennoch werden die Befehle in-order ausgeführt, um den Overhead durch Vorhersagen und fälschlicherweise ausgeführte Berechnungen gering zu halten und die Leistungsaufnahme zu drosseln.

Damit die Pipeline möglichst optimal ausgelastet wird, erfordert es eine Sprungzielvorhersage. Um die Anzahl der falschen Vorhersagen zu minimieren, hat sich ARM entschieden eine dynamische Sprungvorhersage zu implementieren. Die Sprungzielvorhersage übernimmt ein *2-Level Global History Branch Predictor* in der 1. Stufe (Instruction Fetch F1). Er besteht aus 2 Einheiten: dem Branch Target Buffer (BTB) und dem Global History Buffer (GHB).

Bei jeder Instruktion wird im Branch Target Buffer (max. 512 Einträge) überprüft, ob die aktuelle Instruktion schon einmal einen Sprung ausgelöst hat. Falls nicht, wird der Program Counter (PC) um eins erhöht und die nächste Instruktion geladen. Existiert aber im BTB ein Eintrag, bedeutet dies, dass es sich bei dieser Instruktion um einen Sprungbefehl handelt. Daraufhin liefert der BTB die Zieladresse und die Information, ob es sich um einen bedingten (z.B. If-Abfrage) oder einen unbedingten Sprung (z.B. GOTO-Anweisungen) handelt. Bei einem unbedingten wird die

Instruktion der Zieladresse geladen, bei einem bedingten Sprung ist noch nicht bekannt, ob die Bedingung überhaupt zutreffen wird.

Dazu gibt es den Global History Buffer. Er besteht aus 2-bit trägen Automaten (2-bit saturating counters), die ein Indikator dafür sind, ob der bedingte Sprung ausgeführt werden sollte oder nicht. *Abbildung 3* zeigt eine Veranschaulichung durch einen Zustandsautomaten. Er speichert, ob die aktuelle Instruktion in den letzten Ausführungen tatsächlich zu einem Sprung geführt hat oder nicht. Befindet sich der Automat in einem der beiden linken Zustände, wird der Sprung nicht ausgeführt und der PC wird um eins erhöht – befindet er sich in den beiden rechten, wird der Sprung ausgeführt. Stellt sich am Ende heraus, dass es sich bei der Instruktion, unabhängig von der Vorhersage, um einen Sprung gehandelt hat, liest der Automat die Eingabe „T“ (taken) ein, andernfalls „NT“ (not taken).

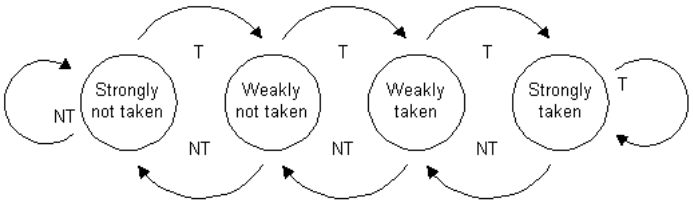

Abbildung 3: 2-bit saturating counter[5]

Diese Vorhersage ist allerdings optimierbar. Einige Fälle führen zu einer geringen Genauigkeit. Ist der Automat z.B. auf dem Zustand „weakly taken“ und die Instruktion führt bei jedem 2. Aufruf einen Sprung aus, führt das zu einem Pendeln zwischen den Zuständen „weakly taken“ und „weakly not taken“: Beim ersten Ausführen der Instruktion geht der Prozessor von einem Sprung aus. Später stellt sich heraus, dass der bedingte Sprung nicht hätte ausgeführt werden dürfen; die Pipeline muss geleert werden. Nun ist der Automat im Zustand „weakly not taken“. Beim nächsten Mal wird der Sprung daher nicht ausgeführt, tatsächlich muss er aber ausgeführt werden. Die Pipeline muss erneut geleert werden. Der Automat geht wieder in den Zustand „weakly taken“ über und beginnt danach dieses Schema von Vorne. Dadurch ergibt sich eine Vorhersagegenauigkeit von 0 %, die Pipeline wird nicht optimal genutzt.

Um solche Fälle zu vermeiden, wird im Global History Register (GHR) gespeichert, bei welchem

der letzten 10 bedingten Sprünge es wirklich zu einem solchen gekommen ist. Eine 0 bedeutet, dass der Sprungbefehl nicht ausgeführt wurde, eine 1, dass er ausgeführt wurde, unabhängig von der Voraussage. Mit dieser 10-bit Global Branch History und den 4 niederwertigsten Bits des Program Counters wird einer der 4096 *2-bit saturating counter* im GHB ausgewählt. Anhand von diesem wird entschieden, ob der aktuelle bedingte Sprung ausgeführt werden soll oder nicht. Somit erhöht sich die Vorhersagegenauigkeit im Beispiel auf 100 %, denn schon nach wenigen falschen Voraussagen haben sich die Automaten eingependelt und im GHR steht die alternierende Folge „1010101010“. Es existiert also nicht, wie man vermuten könnte, zu jedem bedingten Sprung im BTB ein träger Automat, der für die Vorhersage ausgewählt wird. Stattdessen hat das Verhalten der letzten 10 Sprünge Einfluss darauf, welcher von den 4096 Automaten betrachtet wird.
Falls im Branch Target Buffer kein Eintrag gefunden wurde, sich beim Instruction Execute aber herausstellt, dass es sich doch um einen Sprung handelt, wird in der Stufe E4 (Branch Prediction Update) ein neuer Eintrag im BTB vorgenommen. Wenn es sich dabei um einen bedingten Sprung gehandelt hat, wird auch der GHB und das GHR aktualisiert. Anschließend wird die Pipeline geleert, was zu einem Verlust von 13 nichtverwertbaren Taktzyklen führt, man spricht auch von einer Branch Misprediction Penalty von 13 Zyklen.

Diese dynamische Sprungvorhersage hat insgesamt eine Genauigkeit von 95 %, d.h. nur 5 % aller Vorhersagen sind fehlerhaft und die falschen Instruktionen müssen in der Pipeline verworfen werden. Somit wird die Pipeline trotz In-Order-Execution – ohne den Overhead einer Out-of-Order-Execution – fast optimal ausgelastet. Leser, die an weiteren Details der im Cortex-A8 implementierten Vorhersage interessiert sind, sollten einen Blick in das Kapitel „ARM Cortex-A8: A High-Performance Processor for Low-Power Applications“ [3] werfen, für Interessierte an Analysen zur Vorhersagegenauigkeit und weiteren Problemen der verschiedenen Vorhersagetechniken, empfiehlt sich [5] S. 10 ff. zur weiteren Vertiefung in diese Thematik.

Zum Abschluss dieses Kapitels sei noch ein gravierender Nachteil der ARM-Architektur genannt. Es wird kein IA-32-Befehlssatz (x86) unterstützt. Somit können Windows-Betriebssysteme und -programme nicht auf ARM-Prozessoren laufen. Dies kann bei der Wahl des Prozessors schon ausschlaggebend sein, Abstand von den ARM-Produkten zu nehmen. Erfordert beispielsweise die

Ausrichtung auf eine bestimmte Zielgruppe die Installation eines Windows-Betriebssystems, spielen die technischen Vorteile wie die geringe Leistungsaufnahme eine untergeordnete Rolle. In Zukunft könnte – laut einem Artikel des Wall Street Journals – Windows ab Version 8 auch auf ARM-Architekturen laufen.[1]

3.2 Intel Atom Z510 [5][6][9]

Bei dem Intel Z510 handelt es sich um einen 32-bit-Prozessor aus der Z-Serie für MIDs mit einem Kern. Im Gegensatz zu den anderen Modellen der Z5xx-Serie kann auf diesem Kern statt zwei Threads nur ein Thread zur gleichen Zeit laufen. Dadurch müssen die Funktionseinheiten und Caches nicht zwischen den Threads geteilt werden, wodurch der Z510 sich leichter mit dem in 3.1 vorgestellten Cortex-A8 vergleichen lässt. Der Atom unterscheidet sich von dem Cortex-A8 in seinem Befehlssatz. Er hat eine CISC-Architektur, die IA-32. Ein CISC-Befehl wird von Desktop-CPUs in der Regel in *mehrere* Mikrobefehle (µ-ops) zerlegt, um so die Out-of-Order-Execution besser ausnutzen zu können. Da der Z510 die Befehle In-Order ausführt, ist dieser Mehraufwand nicht zu rechtfertigen, sodass nur komplexere Befehle in mehr als einen Mikrobefehl umgewandelt werden. Dennoch bedeutet das Übersetzen eines CISC-Befehles in einen Mikrobefehl einen Mehraufwand, der sich in der Leistungsaufnahme widerspiegelt.

Der Intel Atom hat eine 16-stufige Pipeline und kann ebenfalls 2 Instruktionen gleichzeitig abarbeiten. Durch die CSIC-Architektur variiert die Befehlslänge und die Anzahl der Taktzyklen pro Befehl, was das Pipelining schwieriger macht. Die Prozessoren der Atom-Familie haben außerdem mehr Funktionseinheiten als der Cortex-A8: unter anderem 2 ALUs, 2 FPU (Floating Point Units), sowie eine Multiplikations- und eine Divisionseinheit.

Trotz der tieferen Pipeline beträgt die Branch Mispredict Penalty auch bei diesem Prozessor 13 Taktzyklen. Außerdem ist die Funktionsweise der Sprungzielvorhersage des Z510 mit der des Cortex-A8 identisch. Die Global History Table hat mit 4096 Einträgen genauso viele wie der ARM-Konkurrent, das Global History Register hat 12 Bit. Lediglich bei der Größe des Branch Target Buffers gibt es einen erheblichen Unterschied. Er hat mit 128 statt 512 Einträgen nur ein Viertel der Cachegröße des ARM-Prozessors. Ältere Einträge werden im BTB häufiger durch neue ersetzt,

[1] http://online.wsj.com/article/SB10001424052748704851204576034051605593000.html, 23. Juni 2011

weshalb es bei Sprungbefehlen öfter zu einem *Miss* im BTB kommt, obwohl die Instruktion zuvor schon ausgeführt und im BTB eingetragen wurde. Im Nachhinein muss die Pipeline geleert werden. Aus welchem Grund Intel an dieser Stelle auf einen größeren Cache verzichtet hat, ist nicht bekannt. Denkbar wäre, dass sie mit ihrem – im Vergleich zum Cortex-A8 – ohnehin schon großen Die weniger Platz für Caches zur Verfügung haben.

3.3 Fazit [2][6][8][9]

Da die Herstellerangaben bei der Leistungsaufnahme auf verschiedenen Berechnungen basieren, lassen sich die Werte nicht ohne Weiteres vergleichen: ARM bezieht sich auf die durchschnittliche Leistungsaufnahme, Intel auf die maximale Verlustleistung (Thermal Design Power). Außerdem spielen Power-Management-Technologien eine maßgebliche Rolle. Für einen direkten und brauchbaren Vergleich in Bezug auf Leistungsaufnahme und Rechenleistung sind praktische Tests mit realistischer Benutzung der Geräte notwendig. Ein Benchmark-Vergleich des Cortex-A8 mit dem Atom 330 findet sich in [4].

	ARM Cortex-A8	Intel Atom Z510
Taktrate	600 MHz - 1 GHz	1.1 GHz
L1-Cache (instruction + data)	16 + 16 kb	32 + 24 kb
L2-Cache	256 kb	512 kb
Leistungsaufnahme	300 mW	2 W
Die-Größe in mm^2	$< 4^2$	26
Fertigung	65nm - 90nm	45nm

Tabelle 1: Vergleich [2][6][8][9]

An vielen Stellen wird deutlich, dass die Intel Corporation aus dem Server- und Desktop-Bereich kommt und sich deren jahrelange Entwicklung auf die Rechenleistung konzentriert hat. Ziele wie eine geringere Leistungsaufnahme oder kleinere Chips waren sekundär. Um sich später auch im Embedded-Bereich Marktanteile zu sichern, versuchte Intel die Performance zu reduzieren, um so den Anforderungen hinsichtlich des Stromverbrauches der MID-Hersteller gerecht zu werden. Dabei sind sie ihrer CISC-Architektur und den vergleichsweise hohen Taktraten weiterhin treu geblieben. An Punkten, wie z.B. der Out-of-Order-Execution, wurde hingegen eingespart.

[2] ohne L2-Cache bei einer Fertigung von 65 nm

ARM Limited als Marktführer im Embedded-Prozessor-Markt steht seit Beginn für Low-Power-Designs und kostengünstige Chips. Um den ständig steigenden Anforderungen hinsichtlich Rechenleistung gerecht zu werden, geht die Entwicklung zu immer leistungsfähigeren Prozessoren über, ohne dabei Kompromisse in der Leistungsaufnahme einzugehen.

Neben der angesprochenen Leistungsaufnahme und Rechenleistung gibt es noch 2 weitere Aspekte, die die Wahl des Prozessors maßgeblich beeinflussen. Aktuelle ARM-Prozessoren unterstützen keinen IA-32-Befehlssatz und sind somit nicht mit x86-Software kompatibel – ein mögliches Ausschlusskriterium für die Gerätehersteller. Dafür sind den Herstellern bei ARM-Prozessoren mehr Freiheiten gegeben. Während Intel den Kunden fertig gebaute Prozessoren verkauft, vertreibt ARM lediglich Lizenzen und übergibt den Kunden das Design zur eigenen Fertigung oder zur Herstellung durch Dritte. Dadurch haben die Kunden die Möglichkeit noch vor der Produktion Änderungen am Prozessor vorzunehmen.

Eingesetzt wird der ARM Cortex-A8 unter anderem im HTC Desire, sowie in Apple-Produkten, die einen Apple A4-Prozessor haben (z.B. iPad und iPhone 4). Bei dem Apple A4 handelt es sich um ein System-on-a-Chip, das von Apple entwickelt wurde und auf einem Cortex-A8 basiert.[3] Die Atom-Familie findet in Smartphones und Tablet-PCs kaum Einsatz, eines der wenigen Produkte überhaupt, das den Z510 verwendet, ist der Archos 9 Tablet-PC. Aktuell hat ARM einen Marktanteil von 95 % im Handy- und Smartphonebereich[4] und konnte im Jahr 2010 6,1 Milliarden Prozessoren bzw. deren Lizenzen verkaufen. 62 % davon waren für MIDs bestimmt, 19 % fanden Anwendung in eingebetteten Systemen.[5]

4 Ausblick [2][7]

Auch der Cortex-A15 MPCore lässt sich nach unternehmenseigenen Angaben nicht nur für MIDs verwenden. Er wird im MID-Segment mit 1 - 2 Kernen bei einer Taktrate von 1 - 1.5 GHz empfohlen, prinzipiell sind aber bis zu 4 Kerne und eine maximale Taktrate von 2.5 GHz möglich. Der L1-Cache

[3] http://online.wsj.com/article/SB10001424052702303912104575164112770784290.html, 23. Juni 2011
[4] http://www.xtoysoft.com/blog/?p=1188, 23. Juni 2011
[5] http://www.xtoysoft.com/blog/?p=560, 23. Juni 2011

soll sowohl für Instruktionen, als auch für Daten jeweils 32 kb groß sein. Wie schon der A9-Prozessor wird auch der A15 eine Out-Of-Order-Execution haben und wahrscheinlich mit einer Fertigung von 28 nm oder 32 nm hergestellt werden können. Über Daten zu der Pipeline, der Leistungsaufnahme oder der Größe des Dies wurden noch keine Angaben von offizieller Seite gemacht. Erste Prototypen werden zum Ende des Jahres erwartet, die serienmäßige Produktion könnte schon 2012 starten.

Sollten ARM – hinsichtlich des Cortex-A15 – und Microsoft – in Bezug auf die ARM-Unterstützung von Windows 8 – ihre Versprechen einhalten, könnte ARM seine Marktanteile im Smartphonebereich weiterhin sichern und sogar den Intel-Atom-Prozessoren im Netbook- und Notebook-Markt Konkurrenz machen.

Literatur

[1] F. Koushanfar, M. Potkonjak, V. Prabhu, J. Rabaey: *Processors for Mobile Applications*, pp. 603, 2000 IEEE International Conference on Computer Design

[2] ARM: *http://arm.com/products/processors*, 23. Juni 2011

[3] E. John, J. Rubio: *Unique Chips and Systems*, pp. 79-96. CRC Press, USA, 2007

[4] K. Roberts-Hoffmann, P. Hegde: *ARM Cortex-A8 vs. Intel Atom – Architectural and Benchmark Comparison*, University of Texas at Dallas, 2009

[5] F. Agner: *The microarchitecture of Intel, AMD and VIA CPUs*, http://agner.org/optimize/microarchitecture.pdf, pp. 10-33, 128-133. Copenhagen Univ. College of Engineering, 23. Juni 2011

[6] Intel: *Intel Atom Z510*, http://ark.intel.com/Product.aspx?id=35469, 23. Juni 2011

[7] W. Wang, T. Dey: *A Survey on ARM Cortex - A - Processors*, http://cs.virginia.edu/s̃kadron/cs8535_s11/ARM_Cortex.pdf, Univ. of Virginia, 23. Juni 2011

[8] Texas Instruments:*Cortex-A8 Architecture*, http://processors.wiki.ti.com/index.php/Cortex-A8_Architecture, 23. Juni 2011

[9] Intel: *Intel Atom Z5xx*, http://download.intel.com/design/processor/datashts/319535.pdf, 23. Juni 2011