

Die Erfolgsgeschichte von ARM

Seminar: Die Geschichte der Rechnerarchitektur

Benjamin Decker
benjamin.decker@tum.de

Pauline Laßmann
ga27vit@mytum.de

1. EINLEITUNG

Handys sind aus unserem Leben nicht mehr wegzudenken. Der Großteil der Menschen in Deutschland besitzt ein mobiles Telefon und benutzt dieses regelmäßig. Dabei schreitet die Entwicklung rasant voran und in den letzten Jahren haben die Smartphones einen regelrechten Boom erlebt. Diese übernehmen mittlerweile viele Funktionalitäten eines modernen Desktop-PCs. Die Anforderungen an die Prozessoren in Handys sind allerdings anders als bei Desktop- oder Serversystemen. Ein Hauptmerkmal der Geräte ist ihre Mobilität, die durch geringe Bauform und fortschrittliche Akkutechnik gegeben ist. Damit trotz steigender Leistung die Geräte mobil und handlich bleiben, werden an die Hardware spezielle Anforderungen gestellt. So ist z.B. der Prozessor ein wichtiger Faktor im Hinblick auf die Effizienz. Weiter ist ein niedrigerer Stromverbrauch wichtiger als Performance. Motorola, Hitachi, Intel, Texas Instruments, Atmel und eine Anzahl an weiteren Firmen gehören zu den Unternehmen, die sich über die Jahre der Entwicklung und dem Design solcher Mikroprozessoren gewidmet haben. Der wohl erfolgreiche Hersteller solcher Prozessoren ist das Unternehmen ARM Ltd., das mittlerweile auf der ganzen Welt in Länder vertreten ist. Der von ihnen entwickelte ARM Prozessor ist heutzutage in den meisten unserer mobilen Geräte zu finden. Das entwickelt Design basiert auf der RISC-Architektur. ARM Prozessoren werden von dem Unternehmen an andere lizenziert. Zu solchen Firmen zählen unter anderem AMD, Apple, IBM, Intel, Samsung, Qualcomm, Atmel, Toshiba, Renesas oder Texas Instruments. In der folgenden Arbeit wird die Erfolgsgeschichte des ARM behandelt. Von den Anfängen und der Idee, einen kleinen, effektiven Prozessor zu entwickeln bis hin zu dem, was zukünftige Projekte erzielen wollen.

2. GRUNDLAGEN

2.1 Vergleich RISC- und CISC-Architektur

Im folgendem wird die RISC Architektur genauer betrachtet, auf der der in dieser Arbeit behandelte ARM Prozessor basiert. Des weiteren wird die RISC Architektur der CISC Architektur gegenüber gestellt.

RISC und CISC stehen für die Begriffe „Reduced Instruction Set Computer“ und „Complex Instruction Set Computer“. Es sind zwei unterschiedliche Herangehensweisen im Computerbau, welche aber heutzutage in den meisten Computern nicht mehr in ihrer reinen Form vorkommen, sondern eine Kombination aus beiden bilden. Sowohl CISC-Rechner als auch mikroprogrammierbare Rechner benutzen RISC Konzepte[6].

Zunächst wird die RISC Architektur betrachtet.

Diese Architektur beschränkt sich auf wenige, wirklich notwendige Befehle. Es existieren keine, mit denen man direkt auf Speicheradressen arbeiten kann. Register dienen als Halter für Variablen zur Durchführung von Operationen. Alle Operanden liegen damit in Register vor, wodurch die bevorzugte „Register-Register“ Form ermöglicht wird und „Register-Speicher“ Formen vermieden werden. Die Befehle zwischen dem Hauptspeicher und den Registerspeicher beschränken sich auf LOAD, zum laden von Variablen in das Register, und STORE, zum speichern von Variablen im Hauptspeicher. Da keine direkten Speicherzugriffe durchgeführt werden können, muss der Registerspeicher groß genug sein, sodass Variablen und Parameter eines Prozesses gehalten werden können. Auf einem RISC Chip können sich bis zu 256 Register befinden. Die Befehlswortlänge ist dabei immer fest. Mit 32 oder 64 Bit Wortlänge ist es so möglich Befehle einfach zu interpretieren und zu decodieren. Der Programmcode wird durch diese Maßnahme allerdings länger. Anstatt Mikrocode verwendet man eine direkte Verdrahtung der Befehle im Dekodierer. RISC Architekturen haben nur wenige Befehlstypen. Dies in Kombination mit der festen Befehlswortlänge und der direkten Verdrahtung trägt dazu bei, eine möglichst effektive Ausführungszeit von nur einem Takt pro Befehl zu ermöglichen. Eine optimierte Hardware, wie beispielsweise Pipelining, sorgt für weniger Wartezyklen[2].

Ausgehen von diesen grundsätzlichen Merkmalen eines RISC Rechners wird nun im Folgenden ein Vergleich mit CISC aufgestellt.

RISC Rechner benötigt in der Regel mehr Befehle, um den gleichen Effekt wie bei einem CISC Rechner zu erzielen. Dies liegt am einfacheren Befehlssatz, was mit fester Befehlswortlänge dazu führt, dass der Code länger wird. Beim

CISC kann der Code im Gegensatz dazu durch weniger Register und mehr Befehlen, darunter auch sehr mächtige, wie beispielsweise eine Schleife, die mehrere Register bearbeiten kann, sehr kompakt gehalten werden. Beim Dekodieren ist RISC Code jedoch deutlich schneller als der des CISC.

Weiter hat dieser eine Ausführungszeit von nur einem Takt pro Befehl, während es bei CISC wegen immer ungleichen Befehlswortlängen mehrere Takte pro Befehl geben kann.

Wie oben bereits genannt, hat RISC kein Mikroprogramm, sondern läuft über eine optimierte Hardware. CISC dagegen realisiert seine Befehle durch Mikrocode.

Die Datenspeicherbandbreite bei RISC ist allgemein geringer als bei CISC und immer kleiner, je mehr Operanden im Registerspeicher gehalten und mehrfach genutzt werden.

Wie sieht der Systementwicklungsaufwand genauer aus? Dieser hängt von mehreren Punkten ab. Ist die zu realisierende Architektur einfach oder komplex? Werden wenige Befehlsformate verwendet? Wird die Ausführungszeit durch Parallelverarbeitung und Hardware-Optimierungsmaßnahmen minimiert? Werden überwiegend regelmäßige, einfache Hardware Strukturen benutzt? Welche Implementierungstechnologie wird verwendet?

In RISC Prozessoren werden Hardware Steuerwerke benutzt. Bei CISC werden Mikroprogramm Steuerwerke oder optimierte, komplexe Hardwaresteuerwerke bzw. Mischformen verwenden. Der Hardware Entwicklungsaufwand von CISC Prozessoren ist daher höher als bei RISC. Dies liegt an der geringeren Komplexität der Befehle und der wenigen Befehlsformate.

Letztlich betrachtet man noch Hardware Optimierungsmaßnahmen. Je einfacher die Architektur ist, desto einfacher sind Maßnahmen wie Pipelining zu realisieren. Damit ist die RISC Architektur im Vorteil zur CISC Architektur, da diese wie in den oberen Punkten beschrieben, eine einfachere Architektur hat[28].

2.2 Acorn, ARM Ltd. und die Entwicklung des ARM

Acorn Computers fand seinen Anfang mit Hermann Hause und Chris Curry im Jahr 1979. Zusammen mit einigen Studenten und Forschern der Cambridge Universität gründeten sie eine Firma zur Herstellung von Computern. „The Atom“ war der erste von Acorn entwickelte Computer, der für den Heimgebrauch der englischen Bevölkerung gedacht war. 1982, in einer darauffolgenden Kooperation mit der BBC (British Broadcasting Corporation), entstand der BBC micro, der der britischen Bevölkerung Computer näher bringen sollte. Er etablierte sich vor allem in den Schulen Englands[31].

Während ihrer Suche nach einem Nachfolger für den BBC micro, stießen Acorn Entwickler auf die Arbeit „Berkeley RISC 1“ einiger Studenten der University of California, veröffentlicht 1981. Mit den momentan herkömmlichen Designs von CISC nicht zufrieden, erschien Berkeley RISC 1 wesentlich simpler, ohne komplexe Instruktionen und mit weniger Taktzyklen[25].

Aus der Arbeit von Sophie Wilson und Steve Furber, zwei Computerforschern der Cambridge Universität, entstand der erste Mikroprozessor Acorns, die Acorn RISC Maschine (später umbenannt zu Advanced RISC Maschine), auch als ARM bezeichnet. Wilson entwickelte das Instruktion Set, während Furber sich um das Chip Design kümmerte[47].

Die ersten Exemplare wurden 1985 vorgestellt[31]. Besonderes Anliegen des Teams war ein einfaches Design, welches

sich aus einer Kombination aus simpler Hardware und einem Befehlssatz verwurzelt in RISC, mit einigen wenigen CISC Features zusammensetzt. Die gibt dem ARM seinen kleinen Kern und Leistung[25].

Kurz darauf kam bereits ARM2. Mit einem verbesserten Befehlssatz war es der erste ARM Prozessor, welcher in Serienproduktion ging. Der Prozessor wurde als erstes in dem ARM Development System eingesetzt. 1987 brachte Acorn den Archimedes, das zweite ARM basierte Produkt auf den Markt.

Die Entwicklung des ARM wurde immer weiter betrieben. Man suchte nach Wegen um noch bessere Performance zu bekommen. Das Design wurde um ein 4 Kbyte on-chip und einem Instruktions Cache erweitert und die Clock Rate wurde auf 25 MHz erhöht. Der nun entwickelte ARM3 fand 1990 Gebrauch in Acorn's Desktop Computer.

Im selben Jahr entwickelte sich aus Acorn ein neues Unternehmen. Advanced RISC Machines Ltd. wurde in gleichen Teilen von Acorn Computers, Apple Computers (jetzt Apple Inc.) und VLSI Technology unterstützt und getragen. Grund hierfür war das Interesse aller drei Firmen am PDA (Personal Digital Assistant) Markt. Apple Computers hatten schon seinen ersten PDA, den Newton, entwickelt. Er enthielt einen anderen Prozessor als den ARM, doch das Interesse an dem von Acorn entwickelten Prozessor war bei Apple vorhanden, wodurch nach kurzer Zeit eine Zusammenarbeit mit Acorn entstand. Um wettbewerbsfähig zu bleiben beschloss Apple Computers allerdings, dass sie nicht mit Acorn zusammen entwickelt wollten, sondern lediglich ihren Prozessor zu verwenden gedachten. VLSI Technology wurde daraufhin hinzugezogen. Sie brachten die Tools, Apple das Geld und Acorn 12 seiner Entwickler mit in das neue Unternehmen ein. Das Ziel war es, den wachsenden Markt für kostengünstige, stromsparende und leistungsstarke 32-Bit-RISC-Chips bedienen zu können[47].

Als erstes Produkt, im speziellen Auftrag für Apple, entstand der ARM610 (auch ARM6 genannt). Dieser sah einige Änderungen im Vergleich zum ARM2 vor, welcher damals benutzerdefiniert hergestellt wurde. Der ARM610 Chip hatte 32-Bit Adressierung, ein verbesserten Video Controller (VIDC20) und ein Prozessor, welcher mit Gleitkommazahlen arbeiten konnte. Dank der Zusammenarbeit der drei Firmen unter Advanced RISC Machines Ltd., konnte Apple ab 1993 den ARM610 in seinem PDA, Newton, verbauen[31].

Die Advanced RISC Machines Ltd. (ab jetzt kurz ARM Ltd. genannt) merkte jedoch schnell, dass sie sich mit dem Newton als einziges Produkt nicht lange auf dem Markt halten konnten. Der Newton hatte einige Probleme, welche die Nutzbarkeit sehr einschränkten. Deswegen führte der damalige CEO von ARM Ltd. Robin Saxby das sogenannte IP Lizenzmodell ein. Der ARM-Prozessor wurde an viele Halbleiterunternehmen gegen eine Vorablizenzgebühr und später in Form eines Provisionsmodells vergeben. Dies machte ARM Ltd. zu einem Partner aller dieser Unternehmen. Sie versuchten so ihre Markteinführungszeit zu verkürzen, da es sowohl ihnen als auch ihren Partnern zugute kam.

Der entscheidende Durchbruch für ARM Ltd. kam 1993 mit Texas Instruments (TI). Der Deal hat ARM Ltd. dazu veranlasst, sein Lizenzgeschäftsmodell zu standardisieren und gleichzeitig kostengünstigere Produkte herzustellen. Nach TI wandte sich Samsung an ARM Ltd., um eine Lizenz zu erhalten, und nach nur vier Treffen wurde ein Abkommen getroffen. Die Vernetzung innerhalb der Branche war

entscheidend, um Unterstützung für die Produkte von ARM Ltd. zu verbessern und neue Lizenzabkommen zu schließen. Durch neue Deals kam nämlich auch die Möglichkeit die Entwicklung der RISC Architektur weiter voranzutreiben.

1994 machte sich die harte Arbeit der Jahre bezahlt. Mit dem Beginn der Mobilen Revolution waren kleine Geräte eine Wirklichkeit geworden. Die Chips vom ARM waren in Vergleich zu den anderen damaligen Chips relativ klein, da sich ARM Ltd. auf die Entwicklung kleinere Chips schon früh spezialisiert hatte. Nokia wurde geraten, das ARM-basierte Systemdesign von TI für ihr zukünftiges GSM-Mobiltelefon zu verwenden. Aufgrund von Sorgen um den Speicherplatz in ihren Geräten, war Nokia zunächst dagegen ARM Prozessoren zu verwenden. Dies führte dazu, dass ein neuer benutzerdefinierter 16-Bit Befehlssatz entwickelt wurde, der die Speicheranforderungen senkte. Das war das Design, das von TI lizenziert und an Nokia verkauft wurde. Das erste ARM betriebene GSM-Telefon war das Nokia6110 und es war ein großer Erfolg. Der ARM7 wurde zum Flaggschiff des mobilen Designs und wurde seitdem von über 165 Lizenznehmern eingesetzt und wurde seit 1994 über 10 Milliarden mal hergestellt[31].

Bis Ende 1997 war ARM Ltd. zu einem 26,6 Mio. £ Privatunternehmen mit 2,9 Mio. £ Nettoeinkommen herangewachsen. Es war an der Zeit an die Börse zu gehen. Am 17. April 1998 schloss ARM Holdings PLC eine gemeinsame Notierung an der Londoner Börse und der NASDAQ mit einem IPO von £ 5,75 ab. Die gemeinsame Notierung hatte zwei Gründe. Zuerst war NASDAQ der Markt, durch den ARM Ltd. glaubte, dass sie die Anerkennung gewinnen würde, welche ihnen in der Technologieblase der Zeit, die hauptsächlich aus den Vereinigten Staaten herauskam, zustand. Zweitens waren die beiden Hauptaktionäre von ARM Ltd. amerikanisch und englisch. Es sollte den bestehenden Acorn-Aktionären in Großbritannien eine kontinuierliche Beteiligung ermöglicht werden. Der Börsengang von ARM Ltd. ließ die Aktie in die Höhe schnellen und verwandelte das kleine britische Unternehmen in wenigen Monaten zu ein Milliardenunternehmen[31, 47].

Die 90er sahen einen Boom der Technologie. Viele kleine Start-Ups bildeten sich und wollten mit großen Wettbewerbern mithalten. Allerdings geschah das Unvermeidliche und der Technologie Sektor bröckelte und die Aktien sanken insgesamt um 80-90% ab. ARM Ltd. traf seine Ertragsziele und hatte keine Schulden oder finanzielle Enttäuschungen, spürte jedoch auch den Druck der Rezession. So kam es auch zu Entlassungen. ARM Ltd. war in ein neues Zeitalter eingetreten. Sie entwarfen Strategien, welche die Aktivitäten für die nächsten 5 Jahren festlegten und begannen diesem langfristigen Plan zu folgen. 2001 wurde Warren East zum CEO ernannt. Robin Saxby übernahm den Aufsichtsratsvorsitz von ARM Ltd.[48].

Das Ziel des Unternehmens, die Standard-Prozessorarchitektur zu werden, auf die alle zurückgreifen würden, wurde wahr. Die Mikroprozessoren wurden immer kleiner und viele Unternehmen hatte Schwierigkeiten ihre eignen Mikroprozessoren zu bauen oder gar nicht erst die Tools um diese herzustellen. Dies ist einer der Hauptgründe dafür, dass das Geschäft mit Mikroprozessoren eines der ersten war, das das IP-Lizenzmodell nutzte. ARM Prozessoren wurden in immer mehr Chips verwendet, insbesondere im schnell wachsenden Mobilfunkmarkt, wo ARM allmählich zum de-facto-Standard geworden war. Der ARM-Kern war jedoch "hard IP" und

seine Anwendung auf verschiedene Technologien stellte ein Problem da. ARM Ltd. musste einen synthetisierbaren Kern produzieren, der an jeden lizenziert werden konnte, ohne einen technologiespezifischen Port des Kerns zu benötigen. Als Lösung hierfür wurde im Jahr 2001 der ARM926EJ-S angekündigt. Er ist Teil der ARM9 Familie und war vollständig synthetisierbar mit einer 5-stufigen Pipeline und einer integrierten MMU[31]. ARM9 nahm bald den Platz ein, den der ARM7 als Haupteinnahmequelle behauptet hatte. Darauf folgen 2002 ARM9E, ARM10 und ARM11, wobei die Technologie der letzten beiden Prozessoren noch niedrigere Powerlevels und höhere Performance erreichte.

Um im Wettbewerb mit anderen Unternehmen zu bestehen führte ARM Ltd. kurz darauf die Cortex-Familie ein. Anfang des 21. Jahrhunderts boomte der Smartphone-Markt und die Forderung nach mehr Leistung bei gleichzeitig langer Akkulaufzeit stellte eine große Herausforderung dar. Immer leistungsfähigere Single-Core-Architekturen waren keine langwierige Lösung für ARM Ltd.. Der Cortex-A9 MPCore (2004), der zur Cortex-A Familie gehört, wurde entworfen, mit einem Multicore-Prozessor, der den enormen Dynamikumfang in der Verarbeitung besser adressieren konnte. So konnte man den unterschiedlichsten Benutzerbedürfnissen von Smartphones gerechter werden. Cortex-M von 2005 lieferte stromsparende und kostengünstige Kerne für die Mikrocontroller-Industrie. Motivation hierfür war, dass der Markt für Hochleistungsprozessoren riesig war, aber der Markt für kostengünstige Mikrocontroller noch wesentlich größer und dieser nicht von den neuesten ARM Kernen gut bedient wurde. Cortex-R aus dem Jahre 2011 lieferte leistungsstarke Echtzeit-Prozessoren, die den hochspezialisierten Echtzeitanforderungen gerecht wurden. Die achte Version der ARM-Architektur wurde im Oktober 2011 vorgestellt. Für die neusten Prozessoren von ARM wird nun eine 64-Bit-Architektur für die Datenverarbeitung und Speicheradressierung verwendet, nicht wie bei den vorherigen nur eine 32-Bit-Architektur.

Durch ihre vorrausschauende Planung mittels des IP Lizenzmodells ist es ARM Ltd. gelungen, ihre Firma so auszulegen, dass heutzutage keine Prozessoren mehr hergestellt werden, sondern Kerne, CPUs und Mikroprozessorarchitekturen entwickelt und an Hersteller lizenziert werden. So beispielsweise der ARM7, welche immer noch in vielen Geräten neu verbaut wird, während er schon seit Jahren nicht mehr von ARM Ltd. produziert wird[48].

3. DER ARM PROZESSOR

3.1 Die verschiedenen Modelle von ARM und ein genauerer Blick auf ARMv8

Die Architektur von ARM-Prozessoren erfuhr seit 1985 zahlreiche Veränderungen, zum Beispiel bei der Zahl der Register, der Größe des Adressraumes und dem Umfang des Befehlssatzes. Sie wird daher in Versionen unterteilt, abgekürzt mit ARMv[Versionsnummer], die in Table 1 dargestellt sind.

Die achte Version der ARM-Architektur wurde erstmals im Oktober 2011 vorgestellt. Auch hier gibt es wieder eine Unterteilung:

ARMv8-A (Applications) Wird in komplexen Rechenanwendungen wie Servern, Mobiltelefonen und Automobil-Zentraleinheiten eingesetzt.

Architektur	ARM Design/Familie
ARMv1	<ul style="list-style-type: none"> • ARM1
ARMv2	<ul style="list-style-type: none"> • ARM2 • ARM3
ARMv3	<ul style="list-style-type: none"> • ARM6 • ARM7
ARMv4	<ul style="list-style-type: none"> • ARM7TDMI • ARM8 • StrongARM • ARM9TDMI
ARMv5	<ul style="list-style-type: none"> • ARM7EJ • ARM9E • ARM10E
ARMv6	<ul style="list-style-type: none"> • ARM11 (1176, 11 MPCore, 1136, 1156) • ARM Cortex-M (M0, M0+, M1)
ARMv7	<ul style="list-style-type: none"> • ARM Cortex-A (A8, A9, A5, A15, A7, A12, A17) • ARM Cortex-M (M3, M4, M7) • ARM Cortex-R (R4, R5, R7, R8)
ARMv8	<ul style="list-style-type: none"> • ARM Cortex-A (A32, A53, A57, A72, A35, A73, A55, A75, A76) • ARM Cortex-M (M23, M33) • ARM Cortex-R (R52)

Table 1: ARM Modelle

ARMv8-M (Microcontroller) Wird dort eingesetzt, wo Energieeffizienz, Stromverbrauch und die Größe wichtig sind.

ARMv8-R (Real-Time) Wird verwendet, wenn eine Echtzeit-Reaktion erforderlich ist. Zum Beispiel sicherheitskritische Anwendungen oder solche, die eine deterministische Reaktion erfordern, wie medizinische Geräte oder Fahrzeuglenkungen.

Es wird nun besonders auf ARMv8-A eingegangen und ab jetzt als ARMv8 bezeichnet[4].

3.1.1 Grundsätzliches über ARMv8

Mit den ARMv8 wurde die Möglichkeit geschaffen, 64-Bit- und 32-Bit-Architekturen zu verwenden. Die Ausführungszustände werden jeweils als AArch64 bzw. AArch32 bezeichnet. Der AArch64 unterstützt den A64-Befehlssatz und hält Adressen in 64-Bit-Registern. Er ist einzigartig für ARMv8 und nicht auf frühere Architekturen anwendbar. AArch32 bewahrt die Rückwärtskompatibilität mit der ARMv7-Architektur und erweitert diese, so dass es einige im Zustand AArch64 enthaltene Funktionen unterstützen kann. Es unterstützt weiter die Befehlssätze T32 (Thumb Instruction Set) und A32 (ARM Instruction Set) aus der ARMv7 Generation.

Der aktuelle Zustand eines Armv8-Prozessors wird durch das Exception-Level, den ausgewählten Ausführungszustand und dem Security State bestimmt.

Ausführungszustand Der aktuelle Ausführungszustand (AArch64 oder AArch32) definiert die Breite des Universalregisters und die verfügbaren Befehlssätze.

Exception Level Moderne Software erwartet, dass sie in verschiedene Module unterteilt wird, die jeweils unterschiedlich Zugriff auf System- und Prozessorressourcen haben. ARMv8 ermöglicht diese Aufteilung durch die Implementierung verschiedener Berechtigungsstufen. Die aktuelle Berechtigungsstufe kann sich nur ändern, wenn der Prozessor eine Exception annimmt oder von ihrer Bearbeitung zurückkehrt. Daher werden diese Berechtigungsstufen in der ARMv8-Architektur als Exception Level bezeichnet. Das Konzept des Exception Levels ist grundlegend für ARMv8. Alle Operationen finden auf einer definierten Stufe statt, und ein Register kann in einer oder mehreren dieser vorhanden sein. Das Ändern eines Bits in einem Register auf einer Ebene kann auf einer anderen eine andere Wirkung haben. Ein Exception Level (ELn) mit einem größeren Wert n als ein anderer soll auf einem höheren Level liegen. Ein Exception Level mit einem kleineren Wert n als ein anderer wird als auf einem niedrigeren Exception Level beschrieben.

EL0 Enthält die Anwendungen. Entspricht der niedrigsten Ebene und wird oft als unprivilegiert bezeichnet, während die Ausführung auf jeder Exception Level über EL0 oft als privilegierte Ausführung bezeichnet wird.

EL1 Operating System Kernel

EL2 Hypervisor

EL3 Low-level Firmware und Secure Monitor.

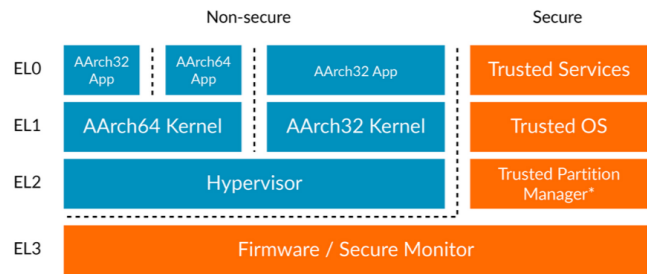


Figure 1: Aufbau Exception Level[4]

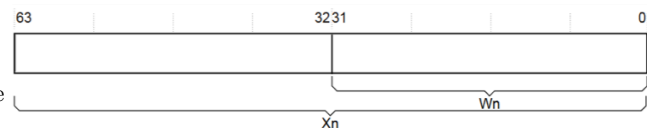


Figure 2: Registeraufbau[4]

Security State Der aktuelle Security State steuert, welches Exception Level aktuell gültig ist und auf welche Speicherbereiche aktuell zugegriffen werden kann. Im Secure Zustand kann ein Verarbeitungselement sowohl auf den sicheren als auch auf den ungesicherten physischen Adressraum zugreifen. In diesem Zustand kann auf die Register des Secure und Non-secure Systems zugegriffen werden. Im Non-Secure Zustand kann ein Verarbeitungselement nur auf den nicht sicheren physikalischen Adressraum zugreifen. Es kann auch nur auf Systemregister zugegriffen, die Non-Secure Zugriffe erlauben vgl. Figure.

Ein Betriebssystem (OS) läuft im Non-Secure Modus, parallel zu einem Betriebssystem (Trusted OS), das in dem Secure Modus auf der gleichen Hardware läuft. Die ARM TrustZone-Technologie ermöglicht die Aufteilung des Systems zwischen Non-Secure und Secure Modi. Dies bietet Schutz vor Soft- und Hardwareangriffen. Der Secure-Monitor dient als Gateway und befindet sich auf einem höheren Exception Level.

Figure 1 zeigt den Aufbau der Exception Levels im Secure und Non-Secure Modus bei AArch64. Bei AArch32 läuft das Trusted OS auf EL3[5].

ARMv8-A ermöglicht außerdem Virtualisierung. Im Non-Secure Modus ermöglicht dies, dass mehr als ein Betriebssystem nebeneinander existieren und auf demselben System betrieben werden. Dies bedeutet, dass der Hypervisor mehrere Gastbetriebssysteme hosten kann. Jedes der Gastbetriebssysteme läuft dann auf einer virtuellen Maschine. Jedes Betriebssystem ist sich nicht bewusst, dass es nicht das einzige ist[4].

3.1.2 Register bei AArch64

ARMv8 bietet 31 x 64-Bit Universalregister, die in allen Exception Leveln jederzeit zugänglich sind. Im AArch64 ist jedes Register (X0-X30) 64 Bit breit. Im AArch32 ist jedes Register (W0-W30) 32 Bit breit vgl. Figure 2.

Lesen aus W-Registern ignoriert die höheren 32 Bit des entsprechenden X-Registers und lässt diese unverändert. Schreiben auf den W Registern setzt die höheren 32 Bit des X-Registers auf null.

Es gibt einen separaten Satz von 32 Registern, die für

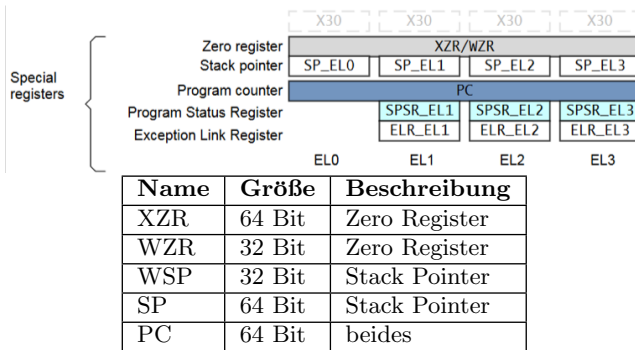


Figure 3: Spezialregister[4]

Gleitkomma- und Vektoroperationen verwendet werden. Diese Register sind 128 Bit breit, können aber wie die Universalregister auf verschiedene Weisen angesprochen werden. Für Gleitkommaberechnungen entspricht die Notation Bx: 8 Bit, Hx: 16 Bit, Sx: 32 Bit, Dx: 64 Bit und Qx: 128 Bit. Soll mit Vektoren gerechnet werden wird die Notation Vx verwendet[17].

Des weiteren gibt es noch fünf Spezialregister, vgl. Figure 3:

Zero Register Ignoriert alle Schreibvorgänge und alle Lesevorgänge geben 0 zurück. Das Nullregister kann in den meisten, aber nicht allen Instruktionen verwendet werden.

Stack Pointer Kann als Basisadresse für Load und Store Vorgänge verwendet werden. Er kann auch mit einem begrenzten Satz von Datenverarbeitungsanweisungen benutzt werden, aber er ist kein normales Universalregister. In ARMv8 ist sowohl Big- als auch Little Endian implementierbar. Die Wahl des zu verwendenen Stack Pointers ist teilweise von dem Exception Level abhängig. Jede Ebene hat ihren eigenen Stack Pointer. Standardmäßig wird der Stack Pointer auf der Ebene gewählt, auf der man sich gerade befindet. Wenn der Prozessor jedoch in AArch64 auf einer anderen Ebene als EL0 arbeitet, kann er entweder den 64-Bit-Stack Pointer, der dieser Ebene zugeordnet ist (SP_ELn), oder den Stack Pointer, der EL0 zugeordnet ist (SP_EL0), verwenden. EL0 kann nur auf SP_EL0 zugreifen.

Programm Counter Hält die aktuelle Programmadresse. Es kann nicht numerisch als Teil der allgemeinen Registerdatei bezeichnet werden und kann daher nicht als Quelle oder Ziel von arithmetischen Anweisungen oder als Basis-, Index- oder Transferregister von Lade- und Speicheranweisungen verwendet werden.

Exception Link Register Enthält die Adresse, an die nach einer Exception zurückgekehrt werden soll.

Program Status Register Auch Saved Processor State Register genannt. Bei einer Exception wird der Prozessorzustand im entsprechenden Saved Program Status Register (SPSR) gespeichert.

3.1.3 Register bei AArch32

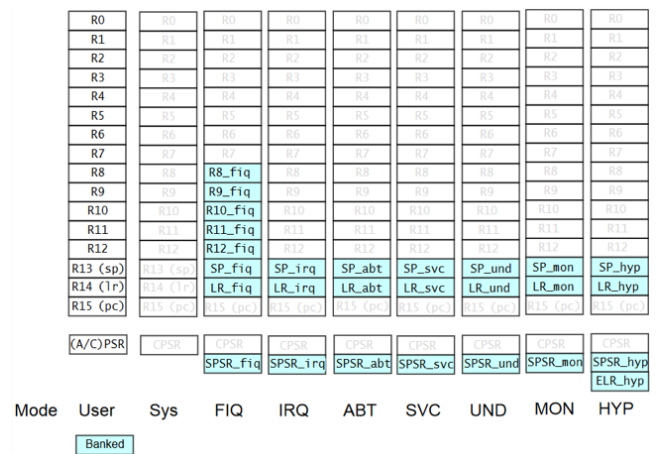


Figure 4: ARMv7 Registeraufbau[4]

Die Kompatibilität mit ARMv7 bedeutet, dass für einen Prozessor, der in AArch32 arbeitet, eine gewisse Übereinstimmung mit den Registern von ARMv7 bestehen muss.

Bei der ARMv7-Architektur gibt es sechzehn 32-Bit-Register (R0-R15). Fünfzehn davon (R0-R14) können für die Datenspeicherung verwendet werden. Das verbleibende Register, R15, ist der Program Counter (PC). Auf welches dieser Register zugegriffen wird und wo, hängt vom Process Mode, in dem sich die Software befindet, und dem Register selbst ab. Dies wird als Banking bezeichnet. Die schattierten Register in Figure 4 sind die, welche in dem angegebenen Modi verwendet werden. Bei ARMv7 gibt es 9 Process Modes die man mit den in ARMv8 verwendeten EL0-EL3 vergleichen kann. Diese werden im nächsten Teilabschnitt genauer erläutert.

Bits[63:32] der X-Register sind im Zustand AArch32 nicht verfügbar und enthalten entweder 0 oder den letzten Wert, der in AArch64 geschrieben wird. Es ist üblich, auf AArch32-Register als W-Register zuzugreifen[4].

3.1.4 System Register

Neben Universalregistern sind Systemregister definiert. Diese werden zur Konfiguration des Prozessors und zur Steuerung von Systemen wie MMU und Ausnahmebehandlung verwendet. Systemregister können nicht direkt von der Datenverarbeitung oder von Lade-/Speicheranweisungen verwendet werden. Stattdessen muss der Inhalt eines Systemregisters in ein X-Register eingelesen, verarbeitet und dann in das Systemregister zurückgeschrieben werden[5].

3.1.5 Exception Typen

Eine Exception ist jedes Ereignis, das dazu führen kann, dass das aktuell ausgeführte Programm ausgesetzt wird. Eine Zustandsänderung zur Behandlung dieser Exception wird dann ausgeführt. Die Armv8-Architektur gliedert Exceptions in zwei Typen: synchrone und asynchrone Exceptions.

Synchrone Exceptions: werden durch die gerade ausgeführte Anweisung verursacht oder hängen damit zusammen. Das bedeutet, dass synchrone Exceptions synchron zum Ausführungsstrom sind.

Asynchrone Exceptions: werden extern generiert und sind daher nicht synchron zum aktuellen Befehlsstrom. Das bedeutet, dass es nicht möglich ist, genau zu sagen, wann eine asynchrone Exception auftritt. Sie können temporär

Modus	Funktion	Security State	ARMv7 Level
User	Hier werden die meisten Anwendungen ausgeführt	beides	PL0
Fast Interrupt Mode (FIQ)	Für Prozessor-Interrupts, die schnell ausgeführt werden müssen	beides	PL1
Interrupt (IRQ)	Für Standard-Interrupts	beides	PL1
Supervisor (SVC)	Für Software-Interrupts, die von Anwendungen angestoßen werden	beides	PL1
Monitor (MON)	a	beides	PL1
Abort (ABT)	Um Speicherfehler zu behandeln	beides	PL1
Undefined (UND)	Wenn ein undefinierter Befehl auftritt	beides	PL1
System (SYS)	Teilt sich die Register mit User Modus	beides	PL1
Hypervisor (HYP)	Wird mit Aufruf vom Hypervisor betreten	beides	PL2

Table 2: ARMv7 Process Modi

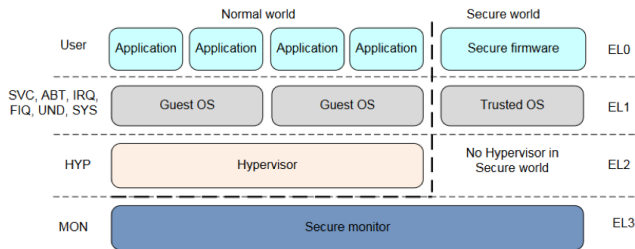


Figure 5: Übertragung von ARMv7 Process Modi auf ARMv8 Exception Modi[4]

ausgeblendet und in einem ausstehenden Zustand belassen werden, bevor sie angenommen werden. Asynchrone Exceptions sind[5]:

Physikalische Interrupts Werden als Reaktion auf ein Signal erzeugt, das außerhalb des Verarbeitungselements erzeugt wird.

- SError (Systemfehler)
- IRQ
- FIQ

Virtuelle Interrupts Können extern oder durch Software erzeugt werden, die auf EL2 ausgeführt wird.

- vSError (Virtual System Error)
- vIRQ (Virtueller IRQ)
- vFIQ (Virtual FIQ)

Die ARMv7-Architektur verwendet Process Level PL0 bis PL2. In ARMv8 wurde diese durch die Exception Levels ersetzt. Einige der in PL1 enthaltenen Modi finden sich als Interrupts in ARMv8 wieder. Table 2 zeigt den vollständigen Satz von Process Modi für einen ARMv7-Prozessor sowie deren Security State[13]:

Figure 5 zeigt wie die Modi des ARMv7 zu ARMv8 passen.

Hier muss allerdings nochmal darauf geachtet werden, ob EL3 AArch64 oder AArch32 benutzt. Wird AArch32 verwendet, erhöht sich das Level von SVC, ABT, IRQ, FIQ, UND und SYS von EL1 auf EL3. Dies liegt daran, dass auf EL3 die Secure Monitor-Funktionalität ist. Die mit dem ARMv6 eingeführten Security Extensions haben den Monitor Modus als Secure State definiert, weshalb die oben genannten Modi zusammen mit der Secure Monitor als EL3 erscheinen[4].

3.1.6 Behandlung von Exceptions

Bei Auftreten einer Exception wird der aktuelle Programmablauf unterbrochen. Für jede Exception wird der Modus (EL0-EL3) definiert, in den die Exception bearbeitet wird. Dieser Modus wird als Zielmodus für die Exception bezeichnet. So ist es beispielsweise möglich, eine Exception von AArch32 EL0 in AArch64 EL1 zu bearbeiten. Wenn eine Exception auftritt:

- wird der aktuelle Exception Level in der SPSR_ELx des Zielmodus gespeichert, wobei x das zu betretende Level ist
- wird die Rückgabeadresse für die Exception im Link Register (LR) des Zielmodus gespeichert

Die drei physikalischen Interrupt-Typen können unabhängig voneinander auf einen der privilegierten Exception-Level EL1, EL2 oder EL3 geroutet werden.

Exceptions, die auf einem niedrigeren Exception Level als die momentane Ebene geroutet werden, werden angehalten. Sie werden bearbeitet, wenn zu einem Exception Level, das gleich oder niedriger als das ist, an das geroutet wurde gewechselt wird.

Die Software kann eine Rückkehr von einer Exception auslösen, indem sie einen ERET-Befehl von AArch64 ausführt. Dadurch wird der zurückgegebene Exception Level basierend auf dem Wert der vorher in SPSR_ELx gespeichert wurde konfiguriert, wobei x das Level ist, auf dem sich momentan befindet. Das Register enthält die Zielebene und den Ausführungszustand, auf die zurückgegriffen werden sollen. Bei Ausführung des ERET-Befehls wird der Zustand von SPSR_ELx wiederhergestellt und der Programmzähler auf den Wert in ELR_ELx aktualisiert. Diese beiden Aktualisierungen werden unteilbar durchgeführt[5].

3.1.7 Wechseln zwischen AArch64 und AArch32

Es gibt die Möglichkeit zwischen der 64-Bit Architektur und der 32-Bit Architektur zu wechseln. Es wird als Interprocessing bezeichnet. Man kann nur zwischen den Ausführungszuständen wechseln, indem man das Exception Level ändert oder resettet. Der Wechsel erfolgt auf der Ebene des Secure-Monitors, Hypervisors oder des Betriebssystems. Ein Hypervisor oder ein Betriebssystem, das in AArch64 ausgeführt wird, kann den AArch32-Betrieb auf niedrigeren Berechtigungsstufen unterstützen. Das bedeutet, dass ein unter AArch64 laufendes Betriebssystem sowohl AArch32- als auch AArch64-Anwendungen hosten kann. Ebenso kann

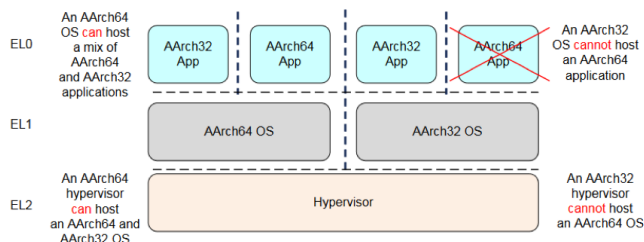


Figure 6

ein AArch64-Hypervisor sowohl AArch32 als auch AArch64-Gastbetriebssysteme hosten. Ein 32-Bit-Betriebssystem kann jedoch keine 64-Bit-Anwendung und ein 32-Bit-Hypervisor kein 64-Bit-Gastbetriebssystem hosten. Dies ist in Figure 6 nochmal verdeutlicht. Um zwischen den Ausführungszuständen auf der gleichen Exception Level zu wechseln, muss das System auf eine andere Ebene wechseln und dann zur ursprünglichen zurückkehren.

Die wichtigsten Punkte, die beim Wechsel zwischen AArch64 und AArch32 zu beachten sind, lassen sich wie folgt zusammenfassen[4]:

- Der Wechsel zu AArch32 erfordert den Wechsel von einem höheren zu einem niedrigeren Exception Level.
- Der Wechsel zu AArch64 erfordert den Wechsel von einem niedrigeren zu einem höheren Exception Level.
- Wenn bei einer Exception oder bei der Rückkehr aus dieser das Exception Level gleich bleibt kann sich auch der Ausführungszustand nicht ändern.
- Sowohl AArch64 als auch AArch32 haben einen ähnlichen Exception Aufbau. Aber es gibt einige Unterschiede zwischen den Secure und Non-Secure Bereichen. Der Ausführungszustand, in dem sich der Prozessor befindet, wenn die Exception erzeugt wird, kann die Exception Level begrenzen, die für den anderen Ausführungszustand verfügbar sind.
- Wenn ein ARMv8-Prozessor sich in AArch32 befindet und auf einer bestimmten Ebene arbeitet, verwendet er das gleiche Exception Modell wie in ARMv7, also die Process Level.
- Code auf Ebene EL3 kann keine Exceptions auf eine höhere Ebene bringen, also kann er den Ausführungsstatus nicht ändern, außer durch einen Reset.

3.1.8 ARMv8 Befehlssatz

Eine der wichtigsten Änderungen in der ARMv8-Architektur war das Hinzufügen eines neuen Befehlssatzes A64. Dieser Befehlssatz enthält viele der gleichen Funktionen wie der vorhandene 32-Bit-Befehlssatz (A32). In AArch64 wird nur A64 verwendet, während AArch32 auf die alten Befehlssätze T32 und A32 zurückgreift.

Der A64 Befehlssatz bietet Zugriff auf 64-Bit große Integer Register und Datenoperationen und die Möglichkeit, 64-Bit große Pointer auf den Speicher zu verwenden. A64 fügt einige zusätzliche Funktionen hinzu und entfernt gleichzeitig andere Funktionen, die die Geschwindigkeit oder Energieeffizienz beeinträchtigen könnten. Der A64-Befehlssatz ist ähnlich dem bestehenden A32-Befehlssatz. Die Anweisungen selbst

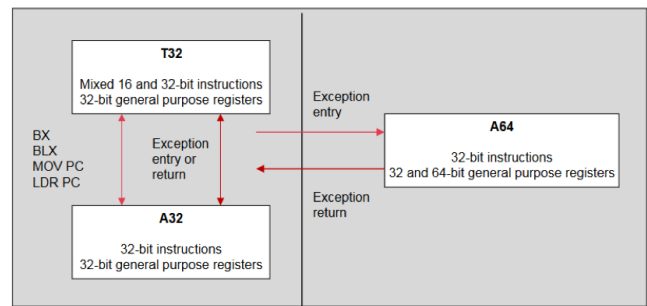


Figure 7: Befehlssatzwechsel[4]

sind immer noch 32 Bit breit und haben eine ähnliche Syntax. Alle A64-Befehle haben die gleiche Länge, im Gegensatz zu T32, dass ein Befehlssatz mit variabler Länge ist. Dies erleichtert die Verwaltung und Verfolgung der generierten Codesequenzen, insbesondere bei dynamischen Codegeneratoren[4].

In AArch32 ist der Befehlssatz A32 weitgehend kompatibel mit ARMv7-A. Es enthält auch einige neue Anweisungen zur Anpassung an einige der Funktionen, die im A64-Befehlssatz eingeführt wurden. Der Thumb-Befehlssatz wurde erstmals in den ARM7TDMI-Prozessor aufgenommen und enthielt ursprünglich nur 16-Bit-Befehle. Diese ermöglichten viel kleinere Programme jedoch auf Kosten der Performance. ARMv7-Prozessoren, einschließlich der Prozessoren der Cortex-A-Serie, unterstützen die Thumb-2-Technologie, die den Thumb-Befehlssatz erweitert und eine Mischung aus 16-Bit- und 32-Bit-Anweisungen bereitstellt.

Es ist nicht möglich, Code aus AArch64 und AArch32 innerhalb einer einzigen Anwendung zu verwenden. Es gibt keine Zusammenarbeit zwischen A64 und A32 oder T32, wie bei A32 und T32. Code, der in A64 für die ARMv8-Prozessoren geschrieben wurde, kann nicht auf Prozessoren der ARMv7 Architektur ausgeführt werden. Code, der für ARMv7-Prozessoren geschrieben wurde, kann jedoch auf ARMv8-Prozessoren im Ausführungszustand AArch32 ausgeführt werden. Dies ist in Figure 7 zusammengefasst.

3.2 Pipelining

Aus Effizienzgründen wird in fast allen modernen CPUs ein Pipelining-System verbaut. In einer RISC Architektur kann fast in jedem Taktzyklus eine neue Instruktion gestartet werden. Eine Instruktion "durchwandert" während der Ausführung die Stufen der Pipeline welche jeweils einen Taktzyklus zum Erledigen ihrer jeweiligen Aufgaben benötigen. Vom ersten ARM Prozessor bis zum ARM7TDMI Kern aus dem Jahr 1994 wurde mit 3 Stufen die einfachste Variante einer Pipeline benutzt. Dabei werden die Befehle in folgenden Stufen abgearbeitet vgl. Figure 8:

fetch Zuerst wird der nächste Befehl aus dem Speicher geladen. Dazu wird an der im Programmzähler angegebenen Stelle im Speicher die noch kodierte Instruktion in das instruction-read Register kopiert. Anschließend wird der Programmzähler inkrementiert und zeigt danach auf den nächsten Befehl im Speicher.

decode Die Instruktion liegt noch in kodierter Form, als Binärzahl, vor. Diese Zahl wird nun der entsprechenden Aktion zugeordnet und es wird alles für die Ausführung

des Befehls vorbereitet. Es werden zum Beispiel Konstanten aus dem Befehl zur weiteren Verarbeitung in einem Register abgespeichert.

execute Anschließend wird der Befehl ausgeführt. Hierbei werden je nach Befehl unterschiedliche Komponenten der execute-Stufe benutzt.

Aktuellere ARM Prozessoren wie zum Beispiel der ARM11 MPCore besitzen mehr als 3 Pipeline Stufen[3]. Längere Pipelines sind nötig um beispielsweise Branch-Prediction zu ermöglichen. Dazu werden dem fetch-Schritt einige Stufen hinzugefügt. Weil in jedem Taktzyklus ein neuer Befehl in die Pipeline aufgenommen wird, liegen in den zusätzlichen Stufen die nach dem aktuellen Befehl folgenden Instruktionen. So lässt sich bei einem konditioniertem Sprung noch bevor dieser im execute-Schritt ausgeführt wird abschätzen ob gesprungen wird oder nicht, also ob die Sprungbedingung erfüllt ist oder nicht. Mit dieser Aufnahme beginnt der entsprechenden Befehle in der Pipeline. Ohne Branch-Prediction müsste vor dem konditionalen Sprung gewartet werden bis bekannt ist ob die Bedingung erfüllt ist oder nicht. Vor Durchführung des Sprungs wird noch einmal überprüft ob die Bedingung wirklich mit der Vorhersage übereinstimmt. Bei einem Fehler muss die komplette Pipeline geleert werden und der Status des Systems vor der Prediction wiederhergestellt werden. Anschließend wird die Pipeline mit dem Programmcode der korrekten Stelle wieder befüllt. Es geht also mindestens so viel Zeit verloren wie die Länge der Pipeline in Taktzyklen entspricht.

4. ARM IM VERGLEICH ZU ANDEREN RISC ARCHITEKTUREN

Da die Anwendung von unterschiedlichen RISC Strukturen immer anders ist, muss man für jede Anwendung alle Faktoren genau betrachten, um zu entscheiden welche Architektur genau verwendet werden soll. So kann die Entscheidung von der Performance, der Verlässlichkeit oder den Kosten abhängen.

Nun werden drei verschieden auf der RISC Strukturen basierend Mikroprozessoren genauer betrachtet und verglichen. Der ARM, um welchen es in dieser Arbeit geht, der MIPS von Wave Computing und der SPARC, der hauptsächlich in Oracle Verwendung findet aber auch in Fujitsu Geräten zu finden ist.

Als ersten werden die allgemeinen Unterschiede der Architekturen betrachtet.

MIPS steht für „Microprozessor without Interlocked Pipeline Stages“ und ging 1980 erstmals in Entwicklung. Er ist vor allem in mobilen Geräten sowie Netzwerk- und Audiogeräte zu finden. Des weiteren ist MIPS Open Source und besitzt einen simplen Coding Syntax, der die Nutzung besonders einfach macht. Die Kosten für Geräte, die MIPS verwenden, sind sehr gering[40].

SPARC heißt „Scalable Prozessor Architecture“ und wurde 1987 entwickelt. Dieser Prozessor wird hauptsächlich von Programmierern und Entwicklern genutzt, da er eine einfache Programmierung hat. Vor allem in Softwares und Geräten, in welchen die Performance von der Durchsatzleistung abhängig ist, sind SPARC Prozessoren zu finden. Allerdings haben diese im Vergleich zu anderen Prozessoren höhere Kosten[49].

Der ARM „Advanced RISC Maschine“ ist eine 32 oder 64 Bit Architektur. der Prozessor kann für verschiedene Märkte

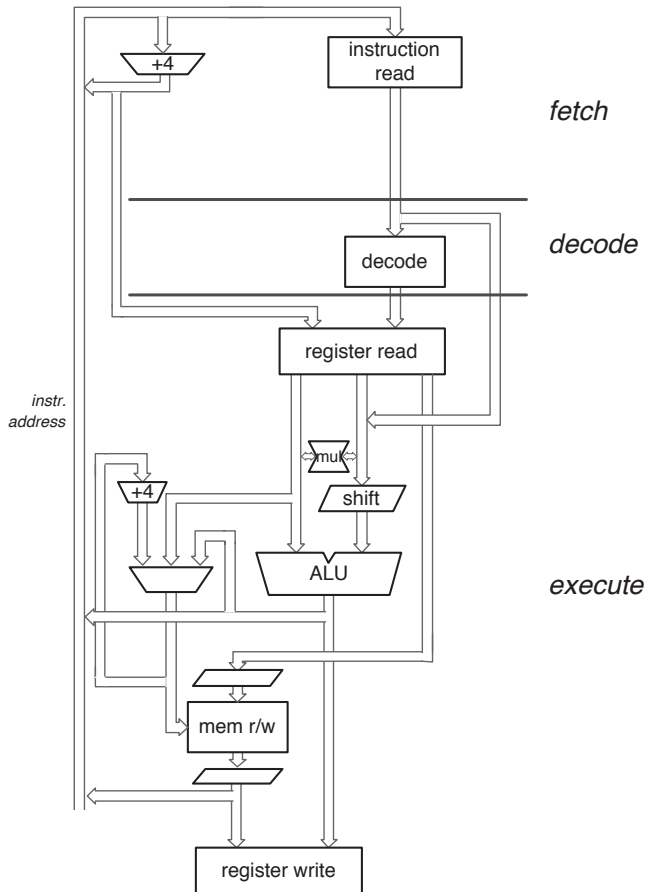


Figure 8: 3-Stufen Pipeline eines einfachen ARM Chips[38]

angepasst und neu entwickelt werden. Es herrscht eine große Gerätevielfalt, so dass je nach Bedarf meist eine ARM Architektur zu finden ist, die die gestellten Anforderungen erfüllt. Jedes Gerät ist außerdem so konfiguriert, dass es so wenig Power wie möglich verbraucht. Hier werden nun zum Vergleich mit den anderen Architekturen nur bis ARMv7 Versionen betrachtet[17].

Die größten Unterschiede der Architekturen sind folgende. MIPS hat einen kleinen Overhead, womit weniger Zeit verbraucht wird. Gleichzeitig hat er eine große Leistungseffizienz, da er eine kleinere Zahl an Instruktionen besitzt. ARM zeichnet sich dadurch aus, dass es einen sehr dichten 16-Bit komprimierten Instruktion Set "Thumb" besitzt, der Anweisungen bedingungslos ausführt. Einer der architektonischen Besonderheiten von ARM besteht darin, dass die verwendeten Kerne zwischen zwei Ausführungszuständen wechseln können: A32(ARM Befehlssatz) oder dem 16-Bit-"Thumb" Befehlssatz. SPARC hat nicht wie die anderen Architekturen ein flaches Register Modell[40].

Betrachtet man die Registernotationen, lässt sich folgendes erkennen.

Die Registernotation des MIPS hat ein \$ vor jedem Registernamen (Beispiel: \$0 entspricht R0). Es gibt 32 Register mit je 32 Bit. Es gibt 2 32-Bit-Register (HI und LO), die Ergebnisse der ganzzahligen Multiplikation und Division enthalten. Register \$0 hat den konstanten Wert 0. Stack und Frame Pointer befinden sich in \$29 und \$30.

Die Registernotation des SPARC hat ein % vor jedem Registernamen (Beispiel: %0 entspricht R0). SPARC besitzt ebenfalls 32 Register der Länge 32 Bit. Diese sind in 4 Gruppen unterteilt: Globale Register (%g0-%g7 sind r[0]-r[7]), Output Register (%o0 bis %o7 sind r[8]-r[15]), lokale Register (%l0 bis %l7 sind r[16]-r[23]) und Input Register(%i0 bis %i7 sind r[24]-r[31]). Der Aufbau der Register ist nicht flach und es existieren sich überlappende Registerfenster.

Die Registernotation des ARM entspricht der realen Registerbenennung (also einfach R0). Die Register sind in Gruppen unterteilt. Sechzehn Register, sogenannte General-Purpose Registers. Sie tragen die Namen R0 bis R15. Alle sind 32 Bit breit. Und zwei Status Register. Das Current Program Status Register (CPSR) und das letzte bezeichnet als saved program status register (SPSR)[40].

Eine Betrachtung der Ausführungsmodi zeigt:

MIPS ist in Kernel Mode und User Mode unterteilt. Beim Kernel Mode ist das Status Bit auf 0 gesetzt. In diesem Modus können alle Register erreicht und verändert werden. Es dient vor allem dem Bearbeiten von Errors, Interrupts und Exceptions. Das Status Bit ist auf 1 gesetzt, wenn der User Mode benutzt wird. Dieser Modus wird vom Benutzer verwendet und hat nicht die gleichen Privilegien wie der Kernel Mode[40].

SPARC besitzt ebenfalls Kernel- und User Mode. Hier ist das Status Bit auch auf 0, wenn man sich im Kernel Mode befindet. Es gewährt vollen Zugriff auf den Speicher, die Prozessor State Register und Systemtabellen. Desweiteren werden auf Input/Output Geräte so zugegriffen. Status Bit = 1 bedeutet User Mode. Hier besitzt der Benutzer nun read/write Zugriff auf die Prozessor State Register, sowie nur einen beschränkten Zugriff auf den Speicher[49].

ARM besitzt nicht wie bei den anderen beiden Architekturen nur zwei, sondern 6 Modi. Der User Mode ist der Ort, an dem die meisten Anwendungsinhalte oder Betriebssystem-Tasks ausgeführt werden. Der Privileged Mode lässt sich

erneute in zwei Operationsmodi unterteilen. Dieser sind System Mode für Betriebssystemanwendungen und der Exception Mode, welcher sich erneute nochmal in 5 Untermodi unterteilen lässt[17]

- Supervisor Mode (SVC)
- Abort Mode (ABT)
- Undefined Mode (UND)
- Interrupt Mode (IRQ)
- Fast Interrupt Mode (FIQ)

Die jetzt beschriebenen Adressierungsmodi sind die Art und Weise, wie Architekturen die Adresse eines Objekts angeben, auf das sie zugreifen wollen.

MIPS unterstützt fünf Adressierungsmodi:

Register Addressing Dieser Modus wird hauptsächlich zur Berechnung der effektiven Adresse des Sprungregisterbefehls verwendet (Bsp. Jump \$R3).

Immediate Addressing Dieser Modus greift nicht auf den Speicher zu und ist somit schneller als andere Modi. Der Immediate hat die Größe 16 Bit (Bsp. ADD \$R1, \$R1, 100 → R1=R1+100).

PC-Relative Addressing Dieser Modus wird für bedingte Zweige verwendet. Die Adresse ist die Summe aus dem Programmzähler und einer Konstanten im Befehl.

„Pseudo“- Direct Addressing Dieser Modus berechnet die effektive Adresse, indem die oberen 4 Bits des Programmzählers mit dem 26-Bit-Immediate verkettet.

Base-Addressing Dieser Modus wird für store und load Instruktionen verwendet. Er wird als indirekte Adressierung bezeichnet, da das Register als Zeiger auf einen Speicherort wirkt, dessen Adresse im Register gefunden werden konnte.

SPARC besitzt zwei Adressierungsmodi zur Berechnung der effektiven Adresse:

Register indirect with index Dieser Modus berechnet die effektive Adresse, indem er die Inhalte des Basisregisters zu denen des Indexregisters hinzufügt.

Register indirect with immediate Dieser Modus berechnet die effektive Adresse, indem er die 13 Bit Immediate auf 64 Bit erweitert und dann die Inhalte des Basisregisters hinzufügt.

ARM besitzt drei Hauptadressmodi zur Berechnung der effektiven Adresse:

Pre-indexed Addressing Dies wird verwendet, um das Lesen von sequentiellen Daten in Strukturen wie Arrays, Tabellen und Vektoren zu erleichtern. Ein Zeigerregister wird verwendet, um die Basisadresse zu halten. Ein Offset kann hinzugefügt werden, um die effektive Adresse zu erreichen (Bsp. LDR R0, [R1, #4] → lädt R0 mit dem Inhalt von dem Register an der Stelle R1+4).

	MIPS	SPARC	ARM
Stackwachstumsrichtung	Nach unten wachsend	Nach unten wachsend	Nach unten und oben wachsend
Push und Pop Instruktionen	Nicht vorhanden. Es wird mit dem Stack Pointer gearbeitet	Keine explizierten Push/Pop Instruktionen. Es wird mit dem Stack Pointer gearbeitet	Keine explizierten Push/Pop Instruktionen. Es wird mit dem Stack Pointer gearbeitet

Table 3: Stack Implementierung

Post-indexed Addressing Ähnlich wie das Pre-indexed Addressing, aber es greift zuerst auf den Operanden an der Stelle zu, die durch das Basisregister angezeigt wird, und erhöht dann das Basisregister.

Program Counter Relative Addressing Register R15 ist der Programmzähler. Wenn man R15 als Zeigerregister für den Zugriff auf den Operanden verwenden, wird der resultierende Adressierungsmodus als relative PC-Adressierung bezeichnet. Der Operand wird in Bezug auf die aktuelle Codeposition angegeben (Bsp. LDR R0, [R15, #24] → lädt R0 mit dem Inhalt R15+24).

Als letztes wird die Stack Implementierung betrachtet: In Table 3 werden Stackwachstumsrichtung und Push und Pop Instruktionen der drei Chips betrachtet.

Alle Architekturen haben ihre Vor- und Nachteile. Aufgrund der schnellen Veränderungen in den Technologien der Computerarchitekturen werden in der nahen Zukunft weitere Entwicklungen und Verbesserungen auftauchen. Dies wird die Wahl der zu verwendeten Architektur erschweren, da die Optionen zunehmen und die Spezifikationen sich anfangen zu ähneln[40].

4.1 Konkurrenz zwischen Intel und ARM

In den letzten Jahren ist die Trennung zwischen Mobilgerät und PC immer mehr verschwommen. So erfüllen mobile Geräte mehr rechenintensive Aufgaben als in der Vergangenheit, was zusätzliche Anforderungen an die Chips stellt, die sie betreiben. Bei dieser Arbeit sind mobile Prozessoren erforderlich, die eine hohe Leistung erbringen, aber auch Energieverwaltung und -einsparung beachten, damit die Batterien der Geräte nicht schnell entladen werden. Der in den Handys enthaltene Mikroprozessor steuert alle wichtigen Funktionen. Dementsprechend wichtig ist die Wahl des Hauptprozessors. Damit alles reibungslos funktioniert, muss sie in kürzester Zeit viele Befehle ausführen. Je schneller allerdings der Prozessor, desto höher auch der Energieverbrauch und desto kürzer die Akku-Laufzeit.

Es wird nun ein Vergleich zwischen Intel und ARM Ltd. hergestellt und die Herangehensweise der beiden Unternehmen mit der wachsenden Handyindustrie und den damit steigenden Anforderungen an die Chips.

Im Wesentlichen versuchen ARM Ltd., leistungsfähigere „handheld computing devices“ zu entwickeln, während Intel versucht einen PC auf Basis der x86-Architektur des Unternehmens zu entwickeln, welcher nun tragbare Größe besitzen soll[45].

Intels Herangehensweise:

Intel hat für den Mikroprozessormarkt einen Chip, den Atom, entwickelt, der als einer der Centrino-Prozessoren des Unternehmens vermarktet wird. Es gibt zwei Varianten des Atom: die Atom-N- und Atom-Z-Serie. Während Atom-Z für den kostengünstigen Desktop- und Notebook-Markt en-

twickelt wurde, ist Atom-N auf mobile Geräte ausgerichtet. Der Prozessor ist eine CISC Architektur und unterstützt Video-, Audio- und 3D-Grafiken, sowie Intels „SpeedStep“ Technologie, welche die Prozessorspannung und Kernfrequenz dynamisch an die Anforderungen der Anwendungen anpasst und damit den Gesamtstromverbrauch senkt. Einige Varianten unterstützen auch Hyperthreading für eine verbesserte Parallelisierung[45].

Intel ist es gewohnt, so viele Funktionen wie möglich in seine PC- und Laptop-Chips zu integrieren. Auch beim Atom ist es so, dass anstatt separate Chips für viele verschiedene Aufgaben zu verwenden, die meisten Funktionen auf dem Chip selbst behandelt werden. Dabei muss aber einen übermäßigen Stromverbrauch im Prozess vermeiden werden. Intel hat daher die meisten Funktionen in Atom integriert, nutzt aber immer noch seinen Controller-Hub, um einige Aufgaben wie Grafik, Speicher und drahtlose Kommunikation zu erledigen. Die Atom Prozessoren sind x86-basierte Architekturen. Auf diese Weise könnten mobile Geräte die gleiche x86-basierte Software verwenden, die auch Computer ausführen, was ein Desktop-ähnliches Erlebnis bietet und es den Benutzern ermöglicht, mit den gleichen Anwendungen zu arbeiten, an die sie von ihren PCs gewöhnt sind. Intels Atom verfügt über eine 16-stufige Pipeline, die drei Phasenstufen umfasst. Jeder Atom-Kern ist außerdem mit zwei ALUs und zwei FPUs ausgestattet. Der Atom Chips enthält 47 Millionen Transistoren und ist der kleinste Prozessor, den Intel je gebaut hat. Atom verbraucht 2 Watt Leistung bei Höchstgeschwindigkeit, 100 Milliwatt bei niedriger Geschwindigkeit und durchschnittlich 200 Milliwatt für eine Reihe von Anwendungen[37].

ARM Herangehensweise:

Ein Großteil der führenden Hersteller von mobilen Chips lizenziert die Chip-Architektur von ARM. Diese Prozessoren reichen in Performance, Leistung und Preis von ARM7-Prozessoren bei 15 MHz bis hin Cortex-A8 mit mehr als 1 GHz. Sie verbrauchen maximal etwa 300 Milliwatt, während Laptops 15 bis 35 Watt nur im Idle-Modus verbrauchen. Meist wird der Befehlssatz der ARM-Version 5 oder 6 verwendet[45].

Um neue Funktionen ausführen zu können, werden neue Schaltkreise auf einzelnen Siliziumstücken oder auf Chips eingebaut, auf denen sich der ARM-Kern bereits befindet. Die Hersteller müssen Funktionen so aufbauen, dass sie mit dem ARM-Kern arbeiten können. Angesichts der Trends von intelligenten drahtlosen Geräten hat ARM Chipkerne für Smartphones entwickelt, die über größere Bildschirme, mehr Funktionen und vollständige Internetbrowser verfügen. ARM hat die Cortex-Kerne entwickelt, um mehr Eigenschaften zu unterstützen und mehr Funktionen bereitzustellen, wie höhere Taktraten und neue Energie-Management-Techniken. Darunter gibt es den Cortex-A8, welcher für den mobilen Markt entwickelt wurde. Die bisherigen ARM-Architekturen

wurden modifiziert, um die beste Performance bei geringster Leistung zu erzielen. Um dies zu erreichen, verwendet die Cortex-A8 eine superskalare Architektur mit „in Order“-Anweisungen. Diese sind weniger komplex als eine „Out-of-Order“-Anweisungen, was zu einem geringeren Stromverbrauch führt. Bei mehreren ALU-Pipelines hat der Cortex-A8 einen durchschnittlichen Befehlszyklus von 0,9 Sekunden.

ARM und Intel kommen aus verschiedenen Perspektiven auf den Mobile-Geräte-Chip-Markt. ARM kommt aus dem Smartphone-Bereich und versucht, einen Mini-PC zu erschaffen, während Intel vom Laptop und PC versucht kleiner zu werden. ARM hat den Vorteil, dass es Erfahrung im Designen von mobilen Chips hat. Intel verfügt über beträchtliche Ressourcen, x86-Kompatibilität und PC-ähnliche Leistung. Hersteller werden diese Faktoren berücksichtigen müssen, wenn sie entscheiden, welche Chips sie in ihre zukünftigen mobilen Geräte einsetzen wollen[45].

5. ARM HEUTE

Heutzutage interagiert jeder Mensch täglich mit Geräten in denen ARM Chips stecken. Sie sind in Handys, Fernsehern, Küchengeräten, Ampeln, Autos und vielen anderen Orten teilweise sogar mehrfach verbaut.

In 2006 verwendeten 98% aller Mobiltelefone auf der Welt mindestens einen ARM basierten Chip[29] und in 2014 waren mit 50 Milliarden weltweiten ARM Chips in 60% aller mobilen Geräte ARM Chips verbaut. In 2011 war die 32-bit ARM Architektur die am weitesten verbreitete Architektur in mobilen Geräten und die populärste in eingebetteten Systemen[24]. Der Markt für mobile Geräte und eingebettete Systeme wird von ARM dominiert.

5.1 Lizenzierung von ARM Chips

Im Gegensatz zu den Anfängen der ARM Erfolgsgeschichte, stellt heute der größte Umsatz von ARM Ltd. die Lizenzierung der Architektur und ISA sowie kompletter Kern-Designs, sogenannter intellectual property Cores, kurz IP-Cores, dar. ARM Ltd. verkauft also lediglich Baupläne für Chips. Die Menge der Chips auf dem Markt hängt also nicht von den Ressourcen des Unternehmens zur Herstellung der Schaltkreise ab, sondern von der Menge an vergebenen Lizenzen. ARM Ltd. ist dabei nur auf die Verbesserung der Baupläne bedacht und muss sich keine Gedanken über Herstellung, Modernisierung von Fabriken oder Lieferengpässe machen, was maßgeblich zur Verbreitung der ARM Architektur beiträgt.

Für viele Chiphersteller ist es profitabel ein ARM-Chip-Design zu kaufen und dieses in Geräte zu integrieren. Diese Designs sind getestet, schnell, effizient und lassen sich, je nach Art der Lizenz, beliebig erweitern oder modifizieren. Das Preismodell setzt sich dabei aus einem Kaufpreis für die Lizenz und laufenden Gewinnanteilen der verkauften Chips zusammen. So lohnt es sich oft, vor allem bei kleinen Geräten, auf ARM zurückzugreifen anstatt selbst einen Chip von Grund auf zu designen, zu testen und seine Funktionalität zu verifizieren.

Die Preise sind dabei unter anderem abhängig vom jeweiligen Chip-Design und betragen zwischen 1 Million und 10 Millionen Dollar. Zur Auswahl stehen dabei unterschiedlich leistungsfähige oder spezialisierte Varianten. Die laufenden Kosten betragen zwischen 1% bis 2% der Einnahmen pro verkauftem Chip[42].

Die Herstellung eines ARM Chips läuft in Schritten ab.

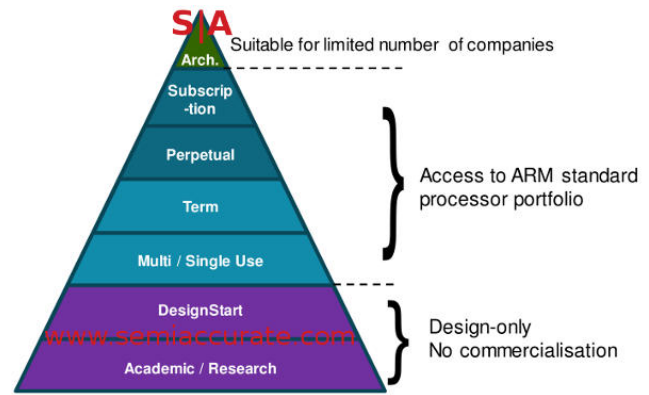


Figure 9: Lizenz Optionen[23]

Zuerst wird entschieden welche Lizenz sich am besten eignet. Hierbei wird auf die Art des infrage kommenden ARM Chip Designs und auf die Laufzeit des Vertrags eingegangen, die auch unbegrenzt sein kann. Anschließend werden Preise und laufende Zahlungen mit ARM Ltd. verhandelt. Bis alles vertraglich abgeklärt ist können bis zu 2 Jahre vergehen. Danach können die Chips aber je nach Anwendungsgebiet bis zu 20 Jahre genutzt werden.

Die Lizenzen können in 3 Kategorien unterteilt werden[23]. vgl. Figure 9:

Design-only licenses **Akademische Lizenzen** und **DesignStart Lizenzen** können gratis oder kostengünstig erworben werden um Zugang zu Bauplänen von unvollständigen ARM Chips oder Chips die noch in der beta-Testphase sind zu Demonstrations- oder Bildungszwecken zu erhalten. Diese Baupläne sind oft in Form einer Hardware-description language wie zum Beispiel vhdl:

```
component armcache
port (
rst      : in  std_logic;
clk      : in  std_logic;
hold     : in  cli_hold;
ici      : in  genic_type_in;
ico      : out genic_type_out;
dci      : in  gendc_type_in;
dco      : out gendc_type_out;
ahbi     : in  ahb_mst_in_type;
ahbo     : out ahb_mst_out_type;
apbi     : in  apb_slv_in_type;
apbo     : out apb_slv_out_type
);
end component;
```

Selbst Baupläne wie der des design-only ARM Chips aus dem dieser Codeausschnitt ist sind bereits sehr komplex. Der gesamte Code mit Kommentaren des Chips aus dem Beispiel ist 176206 Zeilen lang.

Standard licenses Chips die mithilfe von Standard Lizenzen gebaut wurden sind die am weitesten verbreiteten ARM basierten Schaltkreise. Sie sind beispielsweise in Raspberry-Pis verbaut. Anwendung wird dort gefunden wo es einfacher ist ein System um einen funktionierenden Chip herum aufzubauen ohne sich Gedanken

darüber zu machen wie dieser intern funktioniert oder falls man nicht die Mittel besitzt selbst einen Chip zu entwerfen und zu bauen.

Single-use Lizenzen erlauben den Bau und kommerziellen Vertrieb von einem einzelnen Chip Typen, **Multi-use Lizenzen** hingegen für mehrere Chips, die aber nur mit ARM Komponenten eines einzelnen Typs über einen festgelegten Zeitraum gelten.

Term-Lizenzen erlauben je nach Lizenz das Verbauen unterschiedlicher ARM Chip Typen für einen festgelegten Zeitraum.

Perpetual licenses sind äquivalent zu Term Lizenzen aber laufen für einen unbegrenzten Zeitraum. Diese sind sinnvoll für Komponenten die noch über unbekannte Zeit hinweg gewartet und ausgetauscht werden müssen.

Subscription licenses bieten vollen Zugriff auf alle Produkte über einen unbegrenzten Zeitraum. Dadurch ist es möglich schnell auf Marktbewegungen zu reagieren ohne zuerst eine neue Lizenz mit ARM Ltd. auszuhandeln.

Architectural licenses Nur wenige Kunden haben einen Vertrag über eine **Architectural licenses** abgeschlossen und nicht alle sind öffentlich bekannt. Zu bekannten Firmen zählen Apple und Qualcomm. Chips die aus einem solchen Vertrag resultieren sind beispielsweise einige der Apple A-Reihe, CPUs für iPhones oder einige Snapdragon CPUs für Android Handys. Architekturlizenzen sind die teuersten und am wenigsten verbreiteten Lizenzen. Sie erlauben die Modifikation bzw. den Eigenbau von ARM Chips. Optimal, falls ein Chip spezielle, von standardmäßigen ARM Chips nicht gegebene, Anforderungen wie unter anderem geringeren Stromverbrauch oder höhere Performance erfüllen soll. Diese Lizenzen eignen sich nur zusammen mit dem nötigen Wissen zum Bau eines Chips.

Zusammengenommen hat ARM Ltd. zur Zeit mehr als 1000 laufende Lizenzen mit 320 Partnern von denen nur 15 eine Architectural License erworben haben[43].

Neben diesen Lizenzen werden für jede neue Generation von ARM Chips einige wenige Partner ausgewählt die exklusiven Zugang zu den Prototypen bekommen. Im Gegenzug helfen sie ARM Ltd. dabei den neuen Chip zu debuggen und zu testen. Häufig erreichen die Partner durch den technischen Vorsprung einen Marktvorteil gegenüber Konkurrenten. ARM Ltd. bekommt dadurch nützliches Feedback. Eine Win-Win Situation[44].

Die erworbenen oder selbst gebauten Chips werden dann für unterschiedlichste Zwecke verwendet. Zusammen mit einem Modul zur Frequenzanalyse kann ein Radio Empfänger gebaut werden, zusammen mit Beschleunigungssensoren ein Airbag Kontrollsystem[19]. So sind oft auch mehrere ARM Chips in einzelnen Geräten zu finden.

IP-Cores sind meist die Alternative zum Bau eines eigenen Chips und dem Entwurf der einzelnen Kerne und stellen meist die einfachere und auch günstigere Wahl dar.

5.2 SoCs

Mit dem Trend zu immer kleineren elektronischen Geräten wird auch der Bedarf für kleinere Chips immer höher. In diesen Anwendungsgebieten kommen SoCs ins Spiel.

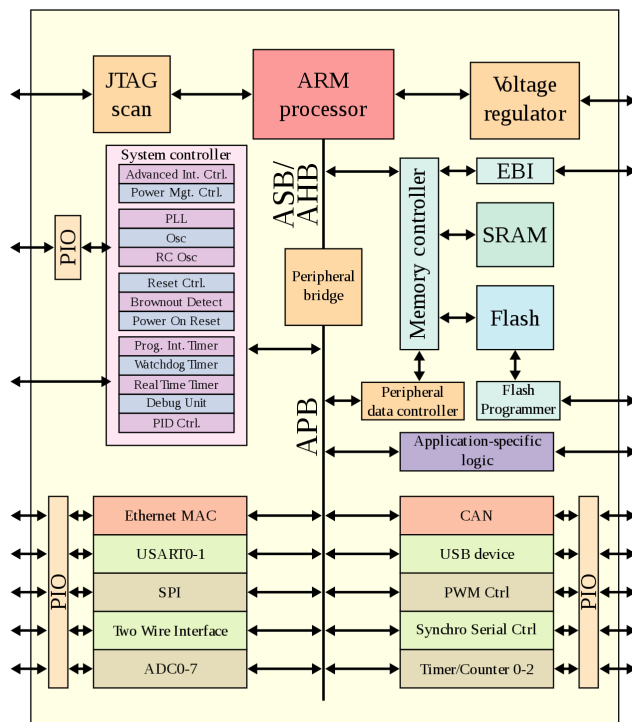


Figure 10: Blockdiagramm eines SoCs basierend auf einem ARM-Prozessor[12]

SoC steht für System on a Chip. Anders als bei Desktop PCs mit einem Mainboard-Design bei dem es eine physikalische Unterteilung zwischen CPU, Grafikkarte, Arbeitsspeicher und anderen Komponenten gibt, die alle modular und austauschbar sind, befindet sich bei einem SoC alles dicht zusammen auf einem einzelnen Chip. So muss beim Ausfall einer einzelnen Komponente der komplette Chip ausgewechselt werden. Außerdem ist es nicht möglich nur Teile des Chips zu upgraden. Allerdings treten durch die Abgeschlossenheit des Systems seltener Defekte auf und in den entsprechenden Anwendungen ist selten ein Upgrade nötig. Eine Kontrolleinheit für eine Ampel muss beispielsweise erst dann erneuert werden wenn die komplette Ampel erneuert wird. Der alte Chip wäre dann wohl ohnehin nicht mehr kompatibel zur neuen Ampel.

CISC Chips benötigen für das Ausführen von komplexen Instruktionen viele verbaute Logikeinheiten für die Interpretation von Befehlen. Mehr Logik läuft auf mehr Transistoren hinaus, was wiederum auf mehr Stromverbrauch und somit weniger Performance pro Watt hinausläuft. Die ARM Chips eignen sich also aufgrund ihrer RISC Architektur besonders für diesen Zweck. Zusätzlich bietet ARM Software-Unterstützung zur Integration und ist deshalb als Haupt- oder Hilfskern in vielen SoCs verbaut. Die Anwendungsgebiete für SoCs sind Umgebungen mit geringer Energiezufuhr bzw. mit hohem Effizienzbedarf oder wenig Raum, also unter anderem: eingebettete Systeme, mobile Geräte, low-power environments und Laptops.

Figure 10 bildet ein SoC mit verbaute ARM-Prozessor ab, und zeigt wie vielseitig sich ein ARM Core einsetzen lässt. Durch Hinzufügen unterschiedlicher Komponenten lässt sich

so gut wie jeder Use-Case abdecken.

Für die Herstellung eines Samsung Handys wird beispielsweise zuerst ein SoC zur Verarbeitung aller Hauptaufgaben auf dem zukünftigen Handy entworfen. Dies passiert mithilfe des Bauplans eines oder mehreren IP-Kerne, oft auf ARM basierend, als Haupt- oder Hilfsrecheneinheit. Die Kerne werden dabei meistens nicht modifiziert sondern lediglich eingebaut und mit anderen Komponenten verbunden. Ein Kern kann so, je nach Entwurf, unterschiedlichste Aufgaben, von arithmetischen Operationen bis zu Video Decoding, übernehmen. Das komplette SoC wird dann von Samsung an einen Handyhersteller verkauft, wo wiederum eigene, möglicherweise wieder auf ARM basierende, Chips eingebaut werden. Oft wird die Herstellung dieser Chips ausgelagert und bei sogenannten Foundrys in Auftrag gegeben. Zusammen mit Bildschirm, Akku und anderen Handykomponenten macht das SoC und somit der Teil des Handys mit ARM Komponenten keinen großen Teil mehr aus. Durch den abgeschlossenen Lizenzvertrag macht ARM aber dennoch mit jedem dieser verkauften Handys Gewinn[14]. Meist ist das ein Kompromiss den viele Hersteller eingehen um den langen und teuren Prozess des Entwurfs eigener Kerne zu umgehen.

Die Komponenten eines SoCs sind

Bus Busse sind für die Datenübertragung innerhalb des SoCs zuständig. Es ist immer ein Bus-System vorhanden. Bei komplizierteren SoCs kann es auch mehrere Busse mit unterschiedlichen Aufgaben und Schnelligkeiten geben, für die auf einzelne Komponenten unterschiedliche Zugriffsrechte oder Zugriffsmöglichkeiten gelten.

Clock Durch regelmäßige Impulse hilft die Clock bei der Synchronisation der Komponenten. Oft sind mehrere, mit unterschiedlichen Frequenzen arbeitende Clocks verbaut. Meistens zählt eine dieser Clocks Sekunden um beispielsweise dem SoC zu ermöglichen eine Uhrzeit anzuzeigen.

CPU Je nach Anwendungsgebiet kann die CPU beliebig performant ausfallen. Oft sind mehrere CPUs in einem SoC verbaut.

Memory Programmcode und temporäre Daten werden im SoC im Speicher abgelegt. Ein SoC kann je nach Anwendungsgebiet auch lediglich read-only Programmcode-Speicher besitzen oder sogar als Schaltnetz komplett ohne Speicher auskommen[8].

Graphics & Audio Bei manchen SoCs ist es nötig eine Grafikeinheit für schnelle parallele Programmausführung oder Video-Verarbeitung, zum Beispiel bei DVD-Playern, oder Audiokerne zur Ton-Wiedergabe, zum Beispiel bei MP3-Playern, zu verbauen.

Interface Von einfachen USB-Anschlüssen bis zu HDMI und Audio Steckplätzen kann die Schnittstelle nach außen beliebig vielfältig sein und hängt sehr stark vom Einsatzort des SoCs ab.

5.3 Das SpiNNaker Projekt

Das SpiNNaker Projekt der Universität von Manchester und anderen Partnern ist der Versuch ein funktionierendes menschliches Gehirn zu simulieren[10].

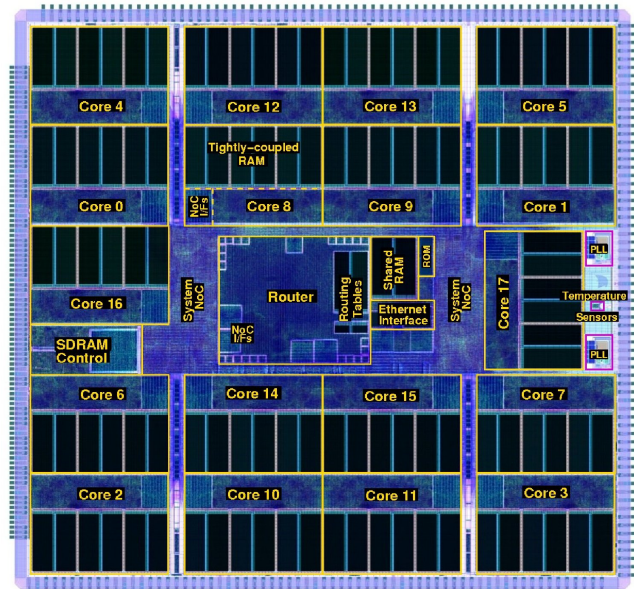


Figure 11: Bauplan einer SpiNNaker ARM CPU[9]

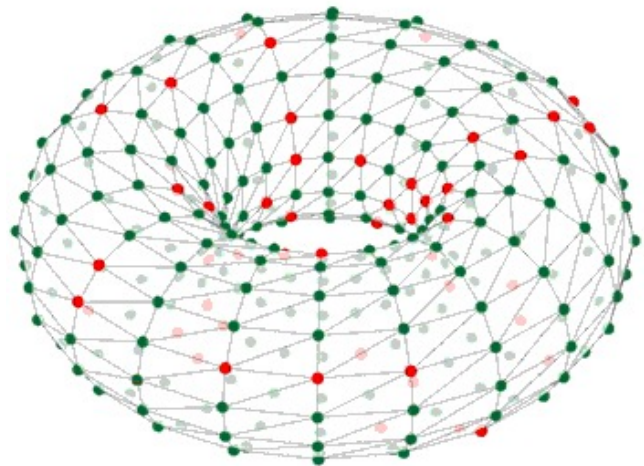


Figure 12: Topologische Anordnung der SpiNNaker Nodes[10]

Für das komplette System werden zunächst 48 ARM-basierte CPUs (siehe Figure 11) auf einem Board befestigt und durch Verbindungen in einem Hexagonalen Netz miteinander verbunden. Die Boards werden anschließend miteinander in einem Rack zu einem toroidalen Netz (siehe Figure 12) zusammen geschlossen. Ab einer gesamten Größe von einer Million auf diese Weise miteinander verbundenen ARM CPUs ist es laut den Projektleitern möglich ein menschliches Gehirn zu simulieren. Die Schwierigkeit bildet hierbei das Fehlen von biologischen Vorgängen im künstlichen Gehirn durch Hormone und äußeren Empfindungen aber vor allem die Programmierung des Systems.

Anders als Computer-Speicher können Neuronen keine Zustände speichern, es wird stattdessen vermutet, dass die Speicherung von Informationen im Gehirn mithilfe von Signalen und Sendemustern geschieht. Alle Knoten schicken dafür Paket-basierte Nachrichten an ihre Nachbarknoten und reagieren auf vorgegebene Weise auf empfangene Nachrichten, ähnlich wie es auch im menschlichen Gehirn passiert. Allerdings ist es schwer vorherzusagen wie ein jeweiliger Knoten sich verhalten soll um der Simulation eines Neurons nahe zu kommen. Man kann erst dann vorhersagen wie sich ein Knoten zu verhalten hat, wenn bekannt ist wie das Gehirn mithilfe von Neuronen Informationen verarbeitet. Gerade das soll mithilfe dieses Projektes erst in Erfahrung gebracht werden. Eine gute Herangehensweise bildet also Trial & Error.

Das Projekt bietet eine Plattform für performante parallele Berechnungen für die Simulation von neuronalen Netzwerken in Echtzeit als ein Forschungswerkzeug für Neuro-, Computer- und Robotikwissenschaftler sowie als Hilfestellung für die Erschließung neuer Computer-Architekturen.

6. DIE ZUKUNFT VON ARM

Mit steigendem Bedarf an energieeffizienten Systemen und immer mehr ARM-kompatibler Software erscheint die RISC Architektur für viele Chiphersteller aber auch Software Entwickler immer sinnvoller. Zwischen 2007 und 2010 hat ARM Ltd. seinen Marktanteil von 13,6 % zu 23,5 % fast verdoppeln[35]. Vielleicht steht ARM mit den aktuellen Entwicklungen erneut ein ähnlicher Sprung bevor. Der offensichtliche große Konkurrent von ARM ist, abseits von anderen auf RISC basierenden Architekturen, Chips mit x86 CISC Befehlssatz. In vielen Marktbereichen ist diese Complex Instruction Set Architecture dem ARM Befehlssatz zur Zeit nur überlegen weil die meiste Software der zugehörigen Anwendungsgebiete für x86 geschrieben wird und nur mithilfe eines Emulators portiert werden kann. Für viele Software Hersteller ist es immernoch einfacher Software ausschließlich für x86 zu schreiben weil in diesen Anwendungsgebieten immer noch die CISC Architektur den Markt dominiert[7]. Durch sich verbessernde RISC Compiler und die dadurch zukunftssichere Entwicklung von Software für unterschiedliche Befehlssätze verschwinden aber immer mehr Nachteile von ARM gegenüber x86.

ARM Ltd. schafft es Jahr für Jahr die Performance ihrer Chips um zweistellige Prozentbeträge zu verbessern[39] und zeigt sich auch für die kommenden Jahre zuversichtlich. So erscheint es in vielen, bisher vom x86 Befehlssatz dominierten Marktgebieten immer sinnvoller auf ARM umzusteigen. Zusammen mit ARMs Einzug in den Server Sektor, Investitionen und Forschung großer Softwarefirmen in die RISC Architektur lässt sich ein Umschwung der marktführenden Architektur

in einigen Bereichen prognostizieren.

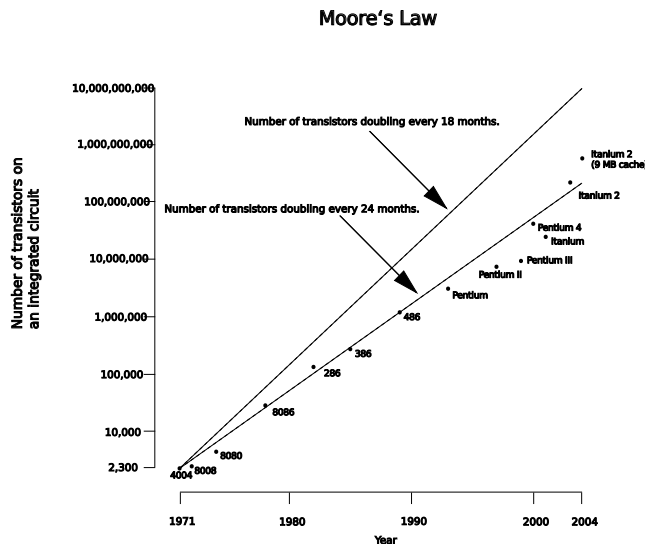


Figure 13: Moores Law[11]

6.1 ARM und das Ende von Moores Law

1965 sagte Gordon Moore, Mitbegründer der Firma Intel, eine jährliche Verdopplung der Transistoren auf Computerchips für die Zukunft voraus. Abgesehen von der Änderung seiner Aussage auf eine zweijährige statt einer einjährigen Verdopplung 1975 behielt Moore bis vor kurzem Recht, siehe Figure 13. Das Phänomen wurde ihm zu Ehren "Moore's Law" genannt. 2010 zeigten sich jedoch die ersten Anzeichen, dass dieser exponentielle Trend nicht länger beibehalten werden kann.

Ein Grund dafür ist die Hitzeentwicklung beim Betrieb der Chips. Für die Erhöhung der Taktrate einer CPU wird in der Regel auch eine erhöhte Spannung benötigt. Falls diese erhöht wird gestaltet es sich immer schwieriger die elektrischen Leitungen so zu konfigurieren, dass nirgends zu viel Spannung anliegt. Durch immer kleiner werdende Transistoren verringert sich jedoch auch die Toleranz der Leitungen für hohe elektrische Ströme und somit höhere Temperaturen. Es gibt also ein Limit für die Größe von Transistoren für eine angestrebte Taktfrequenz ab dem es viel schwerer wird noch weitere Optimierungen bei Taktfrequenz oder Transistorgröße vorzunehmen. Diesem Limit nähern sich moderne CPUs seit 2010 immer mehr an[30].

Ein weiteres Problem ist der sogenannte Tunneleffekt. Die extrem kleinen Größenordnung moderner Transistoren erreichen mit 10 Nanometern eine breite ab der es Elektronen möglich wird durch Quantenübergänge die elektronische Trennung im Inneren eines Transistors zu überwinden[41]. Diese Übergänge treten nur extrem selten auf, werden aber mit abnehmender Größe der Transistoren immer wahrscheinlicher und führen zu fehlerhaften Signalen in der CPU.

Aufgrund dieser Schwierigkeiten fällt es Chipherstellern immer schwerer bei jeder neuen Generation mehr Performance als bei der letzten und somit weniger finanzielle Einnahmen zu erreichen. ARM zeigt sich allerdings wenig besorgt im Hinblick auf diesen Umschwung, wie Mike Muller, chief technology officer und Greg Yeric, director of Future Silicon Technology bei ARM Ltd., 2018 in einem Interview erklären[27]: "Moore's Law is dead. Moore's Law is over. (...) On one level it's true, but I'd say, certainly from my

perspective and Arm's perspective, we don't care". Er gibt dafür einige Gründe an:

3D Chips Um immer mehr Transistoren in einem begrenzten Raum unterzubringen muss entweder die Größe der Transistoren sinken oder die Größe der CPU steigen. Ersteres ist wegen dem Ende von Moores Law nur noch begrenzt möglich, letzteres jedoch schon. Das "stapeln" von CPU Schichten scheint dafür ein vielversprechendes Modell zu sein. Intel plant schon seit einiger Zeit einen Prozessor mit diesem Design[36]. Vorteile beinhalten wegen der größeren Dichte auch kürzere interne Zugriffszeiten. Ein großes Problem dabei ist die Hitzeentwicklung. Durch die größere Dichte und weitere Entfernung zur CPU Oberfläche ist die Temperatur im Inneren des Chips schwerer zu kontrollieren.

Spezialisierte Chips CPUs sind darauf ausgelegt alle möglichen Berechnungen relativ schnell auszuführen. Es gibt allerdings Randfälle und spezielle Berechnungen, in denen eine spezialisierte Komponenten oft um Größenordnungen schneller arbeiten. GPUs sind zum Beispiel zu schnellen parallelen Berechnungen im Stande und übernehmen diese deshalb oft in Systemen in denen sie verbaut. Die CPU führt währenddessen andere Berechnungen durch. Eine zunehmende Auslagerung von speziellen Berechnungen führe auch nach dem Ende von Moores law weiter zu Leistungssteigerungen, wie auch Yerik erklärt.

Alternativen zu Silizium-Chips Silizium Chips sind nur der aktuelle Schritt in der Computer Entwicklung. 1960 wurden die bisher in der Computer-Branche verbreiteten Elektronenröhren von den schnelleren, kleineren und effizienteren Transistoren abgelöst. Nun stehe uns schon innerhalb der nächsten 10-15 Jahre ein weiterer Umschwung bevor. Mit immer fortgeschrittenen Materialien, Fertigungstechniken und Technologien eröffnen sich auch neue Möglichkeiten für Rechenkomponenten.

Das Ende von Moores law ist für ARM also ein überwindbares Hindernis und wird den technischen Fortschritt in der Computer Branche nicht anhalten sondern in neue Richtungen lenken.

6.2 Umstieg zu ARM

In den letzten Jahren wurden einige große Computer-Projekte mit ARM Komponenten als Alternative zu x86 Chips durchgeführt. Japan hat den Bau eines Supercomputers mit ARM CPUs begonnen[33] und Apple verwendet die RISC Chips vermehrt in mobilen Geräten.

Apple ist nicht nur einer der wenigen Besitzer einer ARM architectural license sondern verbaut auch in den meisten seiner mobilen Systeme wie iPhones ARM basierende SoCs. Neu ist allerdings die Verwendung von Prozessoren wie dem "A12X Bionic", einer CPU die Performance in der Größenordnung einer x86 Desktop CPU liefert. Diese befinden sich zum Beispiel im neuen "iPad", einem batteriebetriebenen Tablet[1, 26]. Apple und ARM Ltd. zeigen damit, dass die ARM basierten SoCs es bei weit geringerem Energiebedarf mit Flaggship CPUs der Konkurrenz aufnehmen können.

Der Grund, für diesen Performance pro Watt Vorsprung ist zum großen Teil nur möglich weil Apples iOS Software nativ

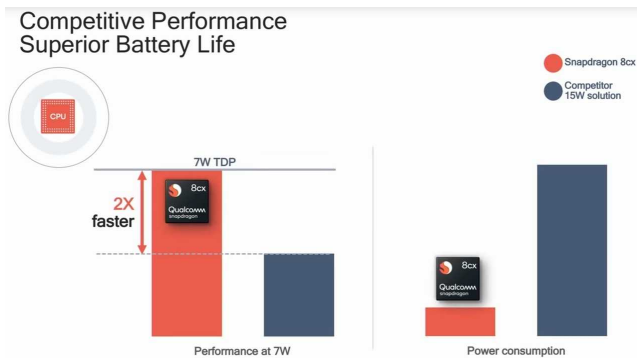


Figure 14: Vergleich zwischen Snapdragon 8cx und einem unbekannten Konkurrenten[32]

auf dem RISC Befehlssatz der ARM Architektur ausgeführt werden kann. Es ist wahrscheinlich, dass Apple auch das MacOS Betriebssystem für den RISC Befehlssatz entwickeln und nach und nach ARM CPUs auf allen Geräten einsetzen werden.

Auch Qualcomm steigt mithilfe von ARM in den Markt der mobilen PCs ein. Schon in der Vergangenheit hat HP mit dem "HP ENVY x2" ein Tablet mit einem von Qualcomm hergestellten ARM basierten Prozessor, dem "Qualcomm® Snapdragon™ 835" ein Windows Tablet mit der RISC Architektur auf den Markt gebracht. Verglichen mit einem äquivalenten Tablet mit x86 Intel CPU für einen um 100 Euro höheren Preis liegt das ARM Tablet bei nativen Tests nur kurz hinter dem x86 Tablet und das bei weit geringerem Stromverbrauch[21]. In den kommenden Jahren wird der von Qualcomm geplante ARM Prozessor "Snapdragon 8cx" viele der bisherigen Nachteile einer ARM Architektur auf Windows PCs lösen. Mit vergrößerten Caches, verbesserter RAM und PCIe Anbindung und 8 performanten Kernen könnte dieser Chip Qualcomm dabei helfen sich zu einem echten Konkurrenten zu Intel und AMD in der Laptop und Tablet Branche zu entwickeln[32, 15]. Figure 14 zeigt ein ungenaues Diagramm der Performance des Chips.

Der Dominanz der ARM Architektur im Home-PC Sektors steht also bald nichts mehr im Weg. Computer sind in der Vergangenheit immer schneller, kleiner und effizienter geworden[46] und ARM verspricht genau diese Eigenschaften. ARM CPUs werden in Zukunft vergleichbare, in vielen Anwendungen sogar bessere, Performance bei geringerem Preis und geringerem Energiebedarf liefern. Nativer Code läuft jetzt schon schneller auf ARM als auf x86 Architekturen. Die Wahl zwischen den beiden Konkurrenten entscheidet sich also zum großen Teil durch die Güte der ARM Compiler die Jahr für Jahr besser werden[16].

Auch der Server Sektor zeigt Auswirkungen des Umschwungs. Während Intel und AMD, Firmen mit einem Sortiment an Server CPUs, seit ihrer Gründung das Ziel verfolgen immer schnellere CPUs mit immer mehr Transistoren zu bauen, ist ARMs erste Priorität die Effizienz ihrer Systeme. Im Serverbereich ist Effizienz eines der wichtigsten Optimierungsziele. Server laufen oft 24 Stunden am Tag und die Hauptkosten für Serverbetreiber sind die Anschaffungskosten, die Stromkosten für den Betrieb und die Kühlung[18]. ARM CPUs bieten eine bessere Performance pro Watt als die bisher im Serversektor dominante x86 Architektur und sind günstiger in der Anschaffung. ARM Systeme lassen sich beliebig skalieren

und flexibel zusammenstellen. Intel bietet im Vergleich dazu beispielsweise nur einige Server-Grade CPUs an mit wenigen Modifikationsmöglichkeiten. Der Umstieg von Servern auf die ARM Architektur scheint vielen sinnvoll und hat sogar schon begonnen: Amazon errichtet einen neuen Server, komplett ausgestattet mit den effizienten RISC CPUs[20].

Den Schritt hin zu ARM ist einer den immer mehr Software und Hardware Hersteller gehen. Microsoft entwickelt eine neue Office Version und Adobe arbeitet an einer neuen Photoshop Version, beide mit nativer Unterstützung für ARM[22, 34]. In einigen Chromebooks sind bereits jetzt ARM CPUs verbaut.

7. ZUSAMMENFASSUNG

Der ARM Prozessor hat über die Jahre seiner Existenz viele Veränderungen erlebt. Seine Entwicklung startete mit der Idee eines kleinen, effektiven Prozessors. In einer Blockhütte in Swaffham Bulbeck bei Cambridge begann dann die Entwicklung eines der effektivsten Prozessoren, die uns heute bekannt sind. Mit dem Boom der Handyindustrie und dem Lizenzmodell von ARM Ltd. ist ein kleines britisches Unternehmen zum Millionenverdiener geworden. Es gibt 8 ARM-Architekturmodelle, jedes effektiver und innovativer als das Vorherige. Das neueste Modell unterstützt nun auch 64-Bit, nicht wie vorher nur 32-Bit Architekturen. Die Cortex Familie bildet dieses neue Modell und hat bereits mehrere Spezialisierungen. Heutzutage interagiert jeder Mensch täglich mit Geräten in denen ARM Chips stecken. Sie sind in Handys, Fernsehern, Küchengeräten, Ampeln, Autos und vielen anderen Orten teilweise sogar mehrfach verbaut. Doch nicht nur in alltäglichen Situationen findet man ARM Prozessoren. Das SPiNNer Project der Universität von Manchester versucht mittels einer großen Anzahl von ARM Prozessoren ein funktionierendes menschliches Gehirn zu simulieren. Japan baut mit ARM CPUs einen Supercomputer und Qualcomm steigt mithilfe von ARM in den Markt der mobile PCs ein. Der Schritt hin zu ARM ist einer, den immer mehr Soft- und Hardware Hersteller gehen. Es sieht so aus als ob ARM Prozessoren bald nicht nur im mobilen Sektor, sondern auch im Server- und Home-PC Sektor eine dominante Rolle übernehmen werden.

8. REFERENCES

- [1] Apple iPad Pro Tech Specs . <https://www.apple.com/ipad-pro/specs/>. Online; visited on 13.06.2019.
- [2] Architekturen moderner Prozessoren . http://www.cs3.tf.fau.de/Lehre/GRa/SS2015/Gra_Kap.4.pdf. Online; visited on 13.06.2019.
- [3] ARM11 MPCore Processor Technical Reference Manual . <http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0360e/I1002919.html>. Online; visited on 13.06.2019.
- [4] Fundamentals of ARMv8 . https://static.docs.arm.com/100878/0100/fundamentals_of_armv8_a-100878_0100-en.pdf?_ga=2.256872715.1898384357.1560017423-1620123220.1559401973. Online; visited on 13.06.2019.
- [5] Learn the Architecture . <https://developer.arm.com/architectures/learn-the-architecture>. Online; visited on 13.06.2019.

- [6] RISC Architecture . <https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/riscisc/>. Online; visited on 13.06.2019.
- [7] RISC vs CISC. <https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/riscisc/>. Online; visited on 13.06.2019.
- [8] Schaltnetze und Normalformen . http://www.info-wsf.de/index.php/Schaltnetze_und_Normalformen. Online; visited on 13.06.2019.
- [9] SpiNNaker Board . <http://apt.cs.manchester.ac.uk/projects/SpiNNaker/SpiNNchip/>. Online; visited on 13.06.2019.
- [10] SpiNNaker Project . <http://apt.cs.manchester.ac.uk/projects/SpiNNaker/project/>. Online; visited on 13.06.2019.
- [11] Moore's Law . <https://deacademic.com/dic.nsf/dewiki/974317>, 2004. Online; visited on 13.06.2019.
- [12] SoCPic. <https://en.wikipedia.org/wiki/File:ARMSoCBlockDiagram.gif>, 2006. Online; visited on 13.06.2019.
- [13] ARM Architecture Reference Manual ARMv7-A and ARMv7-R Edition . https://static.docs.arm.com/ddi0406/c/DDI0406C_C_arm_architecture_reference_manual.pdf, 2014. Online; visited on 13.06.2019.
- [14] ARM Don't Make Computer Chips - Computerphile. <https://youtu.be/Jyp6jFCzW44>, 2016. Online; visited on 13.06.2019.
- [15] <https://www.qualcomm.com/snapdragon/always-connected-pc>, 2019. Online; visited on 13.06.2019.
- [16] <https://developer.arm.com/tools-and-software/embedded/arm-compiler/documentation>, 2019. Online; visited on 13.06.2019.
- [17] ARM Architecture Reference Manual ARMv8, for ARMv8-A architecture profile . https://static.docs.arm.com/ddi0487/db/DDI0487D_b_armv8_arm.pdf?_ga=2.247395972.1898384357.1560017423-1620123220.1559401973, 2019. Online; visited on 13.06.2019.
- [18] Cost of Server Ownership: On-Premise vs. IaaS . <https://www.sherweb.com/blog/cloud-server/total-cost-of-ownership-of-servers-iaas-vs-on-premise/>, 2019. Online; visited on 13.06.2019.
- [19] arm.com. cortex-m0+. <https://www.arm.com/products/silicon-ip-cpu/cortex-m/cortex-m0-plus>, 2019. Online; visited on 13.06.2019.
- [20] J.-L. AUFRANC. Amazon Launches 64-bit Arm Server "A1" Instances. <https://www.cnx-software.com/2018/11/27/amazon-arm-server-a1-instances/>, 2018. Online; visited on 13.06.2019.
- [21] Coreteks. Intel is in serious trouble. ARM is the Future. <https://youtu.be/IFHG7bj-CEI>, 2018. Online; visited on 13.06.2019.
- [22] T. Crowley. Why Windows on ARM? <https://hackernoon.com/why-windows-on-arm-6cd9426f0923>, 2017. Online; visited on 13.06.2019.
- [23] C. Demerjian. A long look at how ARM licenses chips. <https://semiaccurate.com/2013/08/07/a-long-look-at-how-arm-licenses-chips/>, 2013. Online; visited on 13.06.2019.
- [24] J. Fitzpatrick. An Interview with Steve Furber . <https://cacm.acm.org/magazines/2011/5/107684-an-interview-with-steve-furber/fulltext>, 2011. Online; visited on 13.06.2019.
- [25] S. B. Furber. ARM System-on-chip Architecture . Addison Wesley, 2000. Online; visited on 13.06.2019.
- [26] M. Günsch. Nur im Geekbench: Apples A12X fast so schnell wie Intels Core i9-8950HK . <https://www.computerbase.de/2018-11/geekbench-apple-a12x/>, 2018. Online; visited on 13.06.2019.
- [27] N. Heath. 'Moore's Law is dead': Three predictions about the computers of tomorrow . <https://www.techrepublic.com/article/moores-law-is-dead-three-predictions-about-the-computers-of-tomorrow/>, 2018. Online; visited on 13.06.2019.
- [28] R. Hoffmann. Rechnerentwurf: Rechenwerke, Mikroprogrammierung, RISC. De Gruyter Oldenbourg, 1993.
- [29] T. KRAZIT. ARMed for the living room . <https://www.cnet.com/news/armed-for-the-living-room/>, 2006. Online; visited on 13.06.2019.
- [30] J. Loeffler. No More Transistors: The End of Moore's Law . <https://interestingengineering.com/no-more-transistors-the-end-of-moores-law>, 2018. Online; visited on 13.06.2019.
- [31] C. P. Markus Levy. The History of The ARM Architecture: From Inception to IPO . <http://reds.heig-vd.ch/share/cours/reco/documents/thehistoryofthearmarchitecture.pdf>. Online; visited on 13.06.2019.
- [32] F. Müssig. Snapdragon 8cx: Qualcomms nächste CPU für Windows-10-Notebooks . <https://www.heise.de/newsticker/meldung/Snapdragon-8cx-Qualcomms-naechste-CPU-fuer-Windows-10-Notebooks-4245468.html>, 2018. Online; visited on 13.06.2019.
- [33] POST-K. Fujitsu zeigt ARM-Prozessor für Supercomputer . <https://www.golem.de/news/post-k-fujitsu-zeigt-arm-prozessor-fuer-supercomputer-1806-135123.html>, 2018. Online; visited on 13.06.2019.
- [34] T. P. Research. ARMed Attack: Intel And AMD Do Not See The Torpedo Headed Their Way. <https://seekingalpha.com/article/4227086-armed-attack-intel-amd-see-torpedo-headed-way>, 2018. Online; visited on 13.06.2019.
- [35] F. Riemenschneider. ARM-Architektur überholt x86- und Power-Architektur. <https://www.elektroniknet.de/elektronik/halbleiter/arm-architektur-ueberholt-x86-und-power-architektur-79002.html>, 2011. Online; visited on 13.06.2019.
- [36] V. Rißka. Foveros: Intel will unterschiedliche Chips in Zukunft stapeln . <https://www.computerbase.de/2018-12/intel-foveros-3d-stacking/>, 2018. Online; visited on 13.06.2019.
- [37] K. Roberts-Hoffman. ARM Cortex-A8 vs. Intel Atom: Architectural and Benchmark Comparisons . <https://pdfs.semanticscholar.org/0303/99fd8a3f623ddd107f5e85500cc56f777576.pdf>, 2009. Online; visited on 13.06.2019.
- [38] L. Ryzhyk. The ARM Architecture .

- <https://www.cse.unsw.edu.au/~cs9244/06/seminars/08-leonidr.pdf>, 2006. Online; visited on 13.06.2019.
- [39] A. Sag. ARM Unveils Latest Cores At Computex 2019 . <https://www.forbes.com/sites/moorinsights/2019/05/31/arm-unveils-latest-cores-at-computex-2019/>, 2019. Online; visited on 13.06.2019.
- [40] M. K. Sarah El Kady and M. Alhafnawi. MIPS, ARM and SPARC- an Architecture Comparison . http://www.iaeng.org/publication/WCE2014/WCE2014_pp174-179.pdf, 2014. Online; visited on 13.06.2019.
- [41] A. Seabaugh. The Tunneling Transistor . <https://spectrum.ieee.org/semiconductors/devices/the-tunneling-transistor>, 2013. Online; visited on 13.06.2019.
- [42] A. L. Shimpi. The ARM Diaries, Part 1: How ARM's Business Model Works: How ARM makes Money. <https://www.anandtech.com/show/7112/the-arm-diaries-part-1-how-arms-business-model-works/2>, 2013. Online; visited on 13.06.2019.
- [43] A. L. Shimpi. The ARM Diaries, Part 1: How ARM's Business Model Works: How ARM works. <https://www.anandtech.com/show/7112/the-arm-diaries-part-1-how-arms-business-model-works>, 2013. Online; visited on 13.06.2019.
- [44] A. L. Shimpi. The ARM Diaries, Part 1: How ARM's Business Model Works: Types of license & the chosen three. <https://www.anandtech.com/show/7112/the-arm-diaries-part-1-how-arms-business-model-works/3>, 2013. Online; visited on 13.06.2019.
- [45] B. Smith. ARM and Intel Battle over the Mobile Chip's Future . <http://www.eng.auburn.edu/~uguin/teaching/READING/E6200/ARM%20and%20Intel.pdf>, 2008. Online; visited on 13.06.2019.
- [46] THRILLIST. The Difference Between Your Computer Now and Just Five Years Ago . <https://www.thrillist.com/tech/nation/the-difference-between-your-computer-now-and-five-years-ago>, 2016. Online; visited on 13.06.2019.
- [47] B. Walshe. A Brief History of Arm: Part 1 . <https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/a-brief-history-of-arm-part-1>, 2015. Online; visited on 13.06.2019.
- [48] B. Walshe. A Brief History of Arm: Part 2 . <https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/a-brief-history-of-arm-part-2>, 2015. Online; visited on 13.06.2019.
- [49] D. L. Weaver and T. Germond. The SPARC Architecture Manual . <https://cr.yp.to/2005-590/sparcv9.pdf>, 2003.