PART I
ARM Systems and Development

1

The History of ARM

WHAT'S IN THIS CHAPTER?

- The beginnings of Acorn
- How Acorn became ARM
- ARM naming conventions
- ARM processor architecture

In the late 1970s, the computer industry was going through a period of huge change and substantial growth. Up until that time, computers were the size of a room and took dozens of people to operate. The ENIAC weighed 30 tons, took up a surface of 1,800 square feet (167 square meters), and required 150 kilowatts of energy. Huge technological advances were being made, and people started talking about the possibility of creating a computer that didn't weigh more than 2 tons. Then, the transistor revolution happened. Suddenly, all the power held in a room full of electronics (some would say electrics) could be put into a single microchip, just a few centimeters squared. Businesses could afford to have multiple systems; schools bought computers for research and education; and even families could finally enjoy personal computing. The 1970s saw the birth of some of the major players in computers: Apple Computer, Atari, Commodore, and Acorn, to name but a few. It would take only a few years to see some of the mythical names in personal computing: The Amiga, the Atari ST, and the Commodore 64, for example — gift-giving hasn't been the same since.

THE ORIGIN OF ARM

In late 1978, Hermann Hauser and Chris Curry founded Acorn Computers, Ltd. in Cambridge, UK. Initially working as a consultancy group, Hauser and Curry obtained a contract from Ace Coin Equipment to develop a microprocessor-based fruit machine. After an initial research and development phase, Hauser and Curry chose a MOS Technology 6502 processor for the

job. The 6502 was produced in 1974 and was one of the most reliable processors available at that time, which would soon revolutionize the computer industry.

The 6502 was an 8-bit microprocessor that was by far the cheapest microprocessor at the time, while still remaining comparable to competing designs. The 6502 was easy to program, and its overall speed was good. The 6502 was also well known for its low interrupt latency, making it good at handling interrupt-driven events. The 6502 had a simplistic design; it had "only" 3,510 transistors. (Intel's 8085 had 6,500, the Zilog Z80 had 8,500, and the Motorola 6800 had 4,100.) It also had comparatively few registers, requiring frequent RAM access.

The 6502 went on to power some of the most famous names in computing history: the Apple II series, the Atari 2600, and the Commodore VIC-20, among others.

Acorn Computers won multiple contracts and continued to use the 6502 for its projects, capitalizing on its knowledge of this microprocessor.

With its experience with the 6502, Acorn went on to develop the Acorn System 1. It was designed by Sophie Wilson and was based on the 6502. It was a small system (for the time) and was launched into the semi-professional market. With a $96.43 (£65) price tag, it was accessible not only to professionals in laboratories, but also to enthusiasts. It consisted of two simple boards: the top one holding a hex keypad and a small seven-segment LED display standing on a ribbon cable, the bottom one carrying the processor and support circuitry. The simple ROM monitor enabled memory to be edited and software to be stored on cassette tapes. Most were sold as self-assembly kits. With each success, Acorn launched a new System, up to the System 5 in 1983, still featuring a 6502, but rack-mounted, including a disk controller, video card, and RAM adapters.

Acorn then released the Acorn Atom — a personal computer that still used the 6502. By now, Acorn had an excellent knowledge of the 6502, allowing it to push the processor to its limits and sometimes even beyond. Acorn's experience with the 6502 was legendary. However, the 6502 was becoming an old processor, and technological advancements meant that faster processors were available.

From here, Acorn shifted slightly; internal discussions debated on which market to pursue. Curry wanted to target the consumer market, while many other factions did not believe in the consumer market and wanted to target a more professional target, namely laboratories. Several research projects were launched, including the possibility of a 16-bit computer, but Hauser suggested a compromise, an improved 6502-based machine with more expansion capabilities: the Proton.

At the time, the British Broadcasting Corporation, the BBC, took an interest in the microcomputer market and began the BBC Computer Literacy Project. Television episodes on microelectronics and the predicted impact of computers in industry were designed to be made available to students, who had no time to design their own computer system. The BBC wanted a computer to go with their television series and started to look for candidate systems.

The BBC had a long list of subjects that it wanted to demonstrate in its series; graphics capabilities, sound and music, teletext, external I/O, and networking were all requirements. Several companies competed for the contract, and the Proton project was an ideal candidate. The only problem was the Proton didn't actually exist. It was only in the design stage; it wasn't prototyped. Acorn had

little time, only 4 days, and spent those 4 days working night and day, prototyping the design, and getting the Proton ready to show to the BBC Finally, only hours before the meeting with the BBC, the Proton was ready. All the hard work done during that week paid off; not only was the Proton the only machine to meet the BBC's specifications, it also exceeded them. Acorn was awarded the contact, and the project's name was changed. The BBC Micro was born.

The BBC Micro sold so well that Acorn's profits rose from a little more than $4,800 (£3000) to more than $13.6 million (£8.5 million) in July 1983. Acorn anticipated the total sales of the BBC Micro to be approximately 12,000 units, but after 12 years, more than 1.5 million units were sold. By 1984, Acorn claimed sales of 85 percent of the computers in British schools, with deliveries of 40,000 units a month. The BBC Micro was extremely well designed for its use; a robust keyboard that could withstand anything children could throw at it (literally), a carefully designed interface, and the right machine at exactly the right time.

Why Acorn Decided to Create a New Processor

Acorn was now faced with a major problem; almost all its projects had been done on a 6502, so it knew the hardware well. When seeking a new processor to replace the aging 6502, it found that other processors just weren't up to the job. Graphics systems were emerging, and it was clear that the 6502 couldn't keep up in the graphics field. The Motorola 68000 was a 16/32-bit microprocessor that was used in many family and business computers, but slow interrupt response times meant that it couldn't keep up with a communication protocol that the 6502 had no problem running.. Numerous processors were studied and excluded. One by one, the processors on the market were studied, analyzed, and rejected. Finally, the list ran out, and Acorn was left with no choice; if it wanted to do things its way, it had to make its own processor.

Creating processors wasn't necessarily something new; it was the golden age of multipurpose CPUs, and several companies were designing CPUs, using little more than transparent film and pens, but what Acorn was about to do went well beyond simply designing a new CPU with 4000 transistors; ARM set out to create a 32-bit processor.

The project was started in October 1983, with Sophie Wilson designing the instruction set and Steve Furber doing the hardware design using BBC Micros to model and develop the chip, and on April 26, 1985, the first Acorn RISC Machine processor was born, the ARM1. It also worked perfectly the first time, which was rather exceptional for a processor that was basically designed by hand.

The primary application of the ARM1 was to be a coprocessor on BBC Micros and to help create the ARM2. Subsequent chips were sold as specialized coprocessors, or as development boards linked onto BBC Masters. It was in 1987 when the first ARM-based computer was sold, the Acorn Archimedes.

Why Acorn Became ARM

With the education market starting to fall off, Acorn's priority was to open new markets and to promote its processor design. VLSI, Acorn's partner, had been tasked with finding new applications for ARM processors, and Hauser had a separate company, Active Book, that was developing a mobile system based on an ARM2 CPU. Apple Computer's Advanced Technology Group (ATG)

contacted Acorn and started to study ARM processors. Apple's ATG objective was to create an Apple II-like computer, but the project was abandoned for fear of creating too much confusion with Macintosh systems. However, the final report stated that ARM processors were extremely interesting — both their initial design and their power usage and processing power ratio.

Later, Apple Computers again studied the ARM processor. Apple had set strict requirements for its Newton project; requiring a processor that had specific power consumption, performance, and cost, but also a processor that could be completely stopped at any given moment by freezing the system clock. Acorn's design came closest to Apple's requirements but didn't quite fill them. A number of changes were required, but Acorn lacked the resources necessary to make the changes. Apple helped Acorn develop the missing requirements, and after a short collaboration, it was decided that the best move would be to create a new, separate company. In November 1990, with funding from VLSI, Acorn, and Apple, ARM was founded.

Why ARM Doesn't Actually Produce Microprocessors

ARM's original mission wasn't to create a processor but to create an architecture. The subtle difference is that the strategy was not to deliver a CPU with written specifications to a client but more to become a partner, providing a solution to clients who would build their own chips incorporating that solution.

Typical processor designers also manufacture their own processors and let others design systems using their processors and other external components. In some cases, the processor designer also includes peripherals, allowing for a more embedded approach, but it means that a choice must be made between all the processors available.

ARM has a different approach. Although ARM processors are one of the best sellers worldwide, ARM doesn't actually make its own chips; it licenses its intellectual property to other companies. Some of the major players in the field create their own ARM chips: Intel, Nvidia, STMicroelectronics, Texas Instruments and Samsung, to name but a few. This is one of the strong points for ARM; a multitude of ARM-powered processors exist, and they vary greatly in their use and operation. Small ARM-powered processors exist with limited options (few I/O ports, small memory sizes, and so on) and can be found on small systems (ARM-powered processors are common on Bluetooth adapters, for example). And complete systems exist, too, containing everything needed for a small computer. Freescale's i.MX6 processors, for example, contain a DDR controller, ATA controller, Ethernet, flash memory, USB and video controllers, all on one chip, greatly reducing the need for external components.

Another example can be found with Apple Computer, Inc. Apple had clear requirements for the processor that would power the iPhone and iPad but weren't convinced by what was available at the time. Apple knew that it needed an ARM core for the excellent power-to-energy ratio, but existing solutions either had too many peripherals or too few. Apple's sense of perfection is legendary, so it settled for a different option; it made its own core. By becoming an ARM licensee, it was given the tools necessary to make a custom core; not just by the peripherals surrounding the core, but also with custom cache sizes optimized for speed or power-saving. The result is the A4 processor and subsequent generations.

Just like with Apple, you can create your own ARM-powered chip. ARM technology can be licensed in two formats: synthesizable or hard macro. In hard macro format the cell is provided, and external components can be added around the existing cell. In synthesizable format ARM IP is delivered electronically and can be integrated into an ASIC chip using different parameters: cache size, optimizations, or debug options. There is more work to be done, but it also enables greater creativity and differentiation.

Today, the amount of chip manufacturers proposing (or using) ARM-powered systems is overwhelming. Most of the major manufacturers are ARM licensees. Samsung makes the Exynos line of CPUs, and Nvidia makes the Tegra line of chips. You can find both of these in high-end tablets, and the Exynos is even used in a Chromebook. NEC, Panasonic, NXP, Freescale, and STMicroelectronics are but a few to license ARM cores. Currently, 273 companies have licensed the ARM9 core, and more than 100 have a license to the latest Cortex-A technology.

ARM's strategy to not produce processors but to sell IP means that it can concentrate on what it does best: developing extremely power efficient performance processors. ARM spends almost all its R&D creating processor cores, and because the costs are spread across a number of licensees, ARM core licenses are cost-effective. The end result for engineers is a market full of low-power processors with excellent compatibility. Do you want a processor that can run a small Linux distribution for an embedded project? Well, the choice is yours. There are dozens of processors and System On a Chip (SoC) designs available, and the hardest part will not be to find a processor that fits your project; the hard part will be choosing which one you will go with.

ARM NAMING CONVENTIONS

ARM processors have had a relatively consistent naming convention since their start, but the naming convention can be a little confusing. You must understand the difference between a processor and an architecture.

An *architecture* is a design. It is the combination of years of research that forms a base technology. An architecture can define the programmer's model, covering all aspects of the design. The programmer's model defines registers, addressing, memory architecture, basic operation, and indeed all aspects of the processor, as well as modifications from previous architectures.

A *processor* is a device. It depends on an architecture but also adds other features that may not be common to all devices using a particular architecture. The most common example is the processor pipeline; all processors in the same architecture version can use the same instructions (because they are defined in the architecture), but the pipeline may well be different for each processor; it is not specified in the architecture reference.

An architecture has several processors, all of them using the same basic features, but each processor has a slightly different configuration. An architecture reference says whether a cache system is defined, but each processor may have different cache sizes or configurations. In all cases, the use of cache, the general layout, and anything necessary for its use are defined by the architecture. When studying a processor, it is essential to know two details: which processor family this device belongs to and which architecture the processor is based on.

How to Tell What Processor You Are Dealing With

Although processor names may vary, all ARM cores share a common naming convention, and there have been two main naming conventions during the lifetime of the architecture. Classic ARM cores have the name ARM{x}{labels}, with later variants adopting a name of the form ARM{x}{y}{z}{labels}. Since 2004, all ARM cores have been released under the Cortex brand and have names in the form Cortex-{x}{y}.

The first cores, known as classic processors, use the naming conventions listed in Table 1-1. The first number (x) corresponds to the core version. The second and third numbers (y and z) correspond to the cache/MMU/MPU information and cache size, respectively.

TABLE 1-1: ARM Processor Numbering

X	Y	Z	DESCRIPTION	EXAMPLE
7			ARM7 core version	ARM7
9			ARM9 core version	ARM9
10			ARM10 core version	ARM10
11			ARM11 core version	ARM11
	1		Cache, write buffer and MMU	ARM710
	2		Cache, write buffer and MMU, Process ID support	ARM920
	3		Physically mapped cache and MMU	ARM1136
	4		Cache, write buffer and MPU	ARM940
	5		Cache, write buffer and MPU, error correcting memory	ARM1156
	6		No cache, write buffer	ARM966
	7		AXI bus, physically mapped cache and MMU	ARM1176
		0	Standard cache size	ARM920
		2	Reduced cache	ARM1022
		6	Tightly Coupled Memory	ARM1156
		8	As for ARM966	ARM968

The letters after a processor name are called the label and indicate what optional extensions are available on the processor, as shown in Table 1-2.

TABLE 1-2: ARM Label Attributes

ATTRIBUTE	DESCRIPTION
D	Supports debugging via the JTAG interface. Automatic for ARMv5 and above.
E	Supports Enhanced DSP instructions. Automatic for ARMv6 and above.
F	Supports hardware floating point via the VFP coprocessor.
I	Supports hardware breakpoints and watchpoints. Automatic for ARMv5 and above.
J	Supports the Jazelle Java acceleration technology.
M	Supports long multiply instructions. Automatic for ARMv5 and above.
T	Supports Thumb instruction set. Automatic for ARMv5 and above.
-S	This processor uses a synthesizable hardware design.

For newer cores, known as Cortex, the naming convention is different, and easier to follow. There are three Cortex families: Cortex-A, Cortex-R, and Cortex-M.

The Cortex-A family is the computer family; the Application processors. They are designed as fully functional computers, capable of running complex operating systems directly. They are used in mobile telephones, tablets, and laptops.

The Cortex-R family is the fast reacting family, the Real-time processor series. They are often less powerful than the Cortex-A series, but are much more reactive to external stimuli. They adapt better to demanding situations, having lower interrupt latency and more deterministic real-time response, and are often used in critical systems where data interpretation is essential. They are found in medical devices, car systems, and low-level device controllers, such as hard drive controllers.

The Cortex-M family is the ultra-low-powered, small form-factor family, the Micro-controller series. It generally operates at a lower performance point than the A or R series but can run well over 100 MHz. It is usually built into microcontrollers with multiple input and output lines and is designed for small factor systems that rely on heavy digital input and output. The Cortex-M family is found in robotic systems and small consumer electronics and has even been found embedded in data transfer cables. It is often used as support processors in larger devices, for instance, it is not uncommon to find a Cortex-M processor handling power management in a larger application-class device.

Differences between ARM7TDMI and ARM926EJ-S

Traditional naming is useful to know exactly what is available on a processor core. In this case, it serves to compare an ARM7TDMI to an ARM926EJ-S. The ARM926EJ-S was one of the most-used cores, but how does it vary to an ARM7TDMI, or even an ARM1136J-S? How can you tell?

By using the previous tables, you can get a better understanding of the two processors. The ARM7TDMI is an ARM7 core, and because it does not have the {y} or {z} numbering, that means

that it does not have any cache or write buffer. For example, the ARM710 core does have cache, because it has {y} numbering.

Also onboard the ARM7TDMI are four architectural options: T, D, M, and I. T indicates that this core supports the Thumb instruction set; D indicates that this core enables for enhanced debugging via a JTAG port; M indicates that this core supports long multiplication; and finally, I means that this core supports advanced breakpoints and watchpoints.

Also of interest is that the ARM7TDMI belongs to the ARMv4T architecture.

In summary, the ARM7TDMI enables easy debugging and includes some advanced features. However, it does not have any cache or buffering, which might be a problem for some applications, but not all. It also does not have an MMU. The ARM7TDMI powered most of Apple's iPod series, dozens of mobile telephones (especially from Nokia), the Game Boy Advance, most Samsung micro-SD cards, and numerous embedded projects.

The ARM926EJ-S, one of the most popular classic cores along with the ARM7TDMI, belongs to the ARM9 family. Because {y} is 2 and {z} is 6, you know that this processor includes cache and an MMU, and also includes Tightly Coupled Memory. The 926EJ-S also has two options: E meaning that this core includes enhanced DSP instructions, and J meaning that it includes Jazelle Java acceleration technology. The -S at the end of the processor name means that it is delivered as VHDL source code, which is compiled and synthesized by the licensee.

The ARM926EJ-S belongs to the ARMv5TE architecture. Because ARMv5TE includes the Thumb instruction set, this processor also includes Thumb. It is not necessary to specify the extension at the end of the processor name. Because it belongs to the ARMv5 architecture, it also automatically supports JTAG debugging, breakpoints, and long multiplication. In short, the ARM926EJ-S supports all the options that the ARM7TDMI had. The ARM926EJ-S went on to power an entire line of mobile telephones, home and business network devices, and some graphing calculators.

The ARM926EJ-S was a great choice when upgrading from ARM7TDMI systems. They were binary compatible, but the ARM926EJ-S had even better energy efficiency and much higher performance. However, some projects encountered a small problem when comparing the two processors; at equivalent clock rate with all caches disabled on the ARM926EJ-S, the ARM9 was noticeably slower than the ARM7TDMI. The ARM926EJ-S was designed with cache in mind, and it is extremely important to run the ARM926EJ-S with cache enabled, as specified in the Programmer's Model.

Differences between ARM7 and ARMv7

This is a common question — and a common mistake. There is no comparison possible; the ARM7 is a core, whereas ARMv7 is an architecture.

The ARM7 is a generation of processor designs; some of the most famous cores include the ARM7, ARM7TDMI, and ARM7EJ. The fact that the core name contains the number 7 does not mean that it belongs to the seventh architecture, far from it. Indeed, all three belong to different architectures: the ARM7 belongs to the ARMv3 architecture, the ARM7TDMI belongs to ARMv4T, and the more recent ARM7EJ belongs to ARMv5TE.

More recent cores from ARM use the Cortex naming convention and are easier to categorize.

Differences between Cortex-M and Cortex-A

Using the new naming convention from ARM, it is easier to immediately tell what a core is designed for. There are three branches in the Cortex family: the Cortex-A, the Cortex-R, and the Cortex-M. Bonus points are awarded to anyone noticing that, again, ARM managed to use the same three letters.

A Cortex-A, for Application, is probably connected to a large amount of memory and runs at a relatively high clock speed. It is designed to handle large amounts of applications, while running a complete operating system. It can be used as the primary processor on mobile devices that require fast computational power, while using little power. You can find the Cortex-A in mobile phones, tablets, digital cameras, and just about any consumer mobile device.

A Cortex-M, on the other hand, designed for the microcontroller world, has much less memory, runs at a slower clock speed but requires far less energy to run and is much smaller. It is often used to control hardware devices or to be an interface between hardware and another processor. (Most Bluetooth USB keys have a Cortex-M processor inside.)

These two cores can be used for separate functions, but often they are used together. A smaller Cortex-M could take some of the work of a larger Cortex-A by handling device connection, data output, or power supply regulation.

MANUFACTURER DOCUMENTATION

There are two things to know about the processor that you use:

- Processor family
- Architecture version

ARM provides two documents: the processor manual, called the Technical Reference Manual, and the architecture, called the Architecture Reference Manual.

The Architecture Reference Manual lists all the features common to an architecture version, including, but not limited to, assembly instructions, processor modes, memory management systems, and other subsystems. The Technical Reference Manual gives more detailed information about the options and internal information about the current CPU but does not go into architecture details.

For a system on a chip device, the manufacturer normally has extensive documentation. The SoC will be based on an ARM core and ARM architecture, so ARM's documentation will be necessary, but the manufacturer will also produce a document listing all the electrical characteristics and input/output information, together with any custom or proprietary peripherals that have been included.

WHAT IS ARM DOING TODAY?

Today, you are in the middle of a mobile revolution. You are no longer tied to cables, in some cases even for recharging mobile devices. The amount of mobile devices has exploded, and with those figures, the amount of processors. Today, with the phenomenal amount of ARM licensees, companies are building bigger and better chips. The Samsung Exynos Octa serves as an example of

previously unheard of designs. On one single chip, there are two clusters of processors, a quad-core Cortex-A7 and a quad-core Cortex-A15, for a total of eight cores, and also a graphics processor, as well as numerous peripherals.

CPUs are not the only technology that ARM licenses. One of its concerns when moving away from the 6502 was the chip's inability to provide good graphics (for the time). Although some embedded systems do not have a screen, others depend heavily on one. A tablet today is only as good as the CPU inside the tablet, but also only as good as the graphics chip. Having a fast CPU isn't everything; if the web page isn't displayed fluidly, the system is considered to be useless. ARM develops and licenses Mali graphics processors, achieving more graphics power than some desktop-based graphics cards, while using ultra-low power.

ARM is also heavily focused on its new architecture: ARMv8. ARMv8 is a huge step forward because it introduces 64-bit capability to the ARM ecosystem, while maintaining 32-bit support. ARMv8 will open a new market for ARM. Although the low power Cortex-A57 processor could be integrated into a mobile telephone or a tablet, it can also be used on servers, with multiple vendors already working on solutions using ARM technology. A typical server room presents one of the biggest IT costs of any company; the amount of electricity used by servers is phenomenal. Server processors are power hungry, but they also create a lot of heat that has to be evacuated quickly; estimates say that 25 percent of the electricity used is for cooling. Because ARM processors are low-powered and run cool, they are the ideal candidate.

SUMMARY

In this chapter, I have explained the beginnings of Acorn, how the company became ARM, and what prompted ARM to create a new processor. After years of developing processors, ARM created two different naming conventions, and they have been explained, as well as the architecture system that different processors belong to. It is also important to explain the changes between the different processors, and the processor attributes.

In the next chapter, I will go deeper into what an ARM embedded system is, what makes them different from other systems, and what you will need to know before beginning a new project.