The ARM11™ Microarchitecture

David Cormie
ARM Ltd

The ARM11™ microarchitecture is the first implementation of the ARMv6 instruction set architecture, and forms the basis of a new family of ARM11 cores. The microarchitecture is the detailed definition of the internal design, and hardware resources, which supports the ARMv6 architectural specification.

The key objective in developing the ARM11 microarchitecture was to deliver high performance at low power and low cost. This paper examines the specific features of the new microarchitecture, and explains why the design meets the needs of next-generation wireless and portable consumer products.

ARMv6 Architecture

The first step in the implementation of a new generation of microprocessors is the definition of the *architecture***.** The architecture describes the rules for how the microprocessor will behave, but without constraining or specifying how it will be built. The definition of the architecture provides the specification for the interface with the outside world – enabling operating system, application and development support to be planned and implemented. In detailed terms, the microprocessor architecture defines the processor's instruction set, the programmer's model, and how the processor interfaces with its closest memory resources.

The latest ARM® architecture – ARMv6, was announced in October 2001. ARMv6 builds on many of the successful architecture specifications developed by ARM over the last 10 years. Successive ARM architectures have provided the foundation for ARM's market leadership in 16- and 32-bit microprocessor cores.

Architecture	Feature Set			
	Thumb®	DSP	Jazelle™	Media
V4T	√			
V5TE	√	√		
V5TEJ	√	√	√	
V6	√	√	√	√

Figure 1. Evolution of the ARM Architecture

The ARMv6 architecture has been developed specifically with the needs of next-generation consumer, wireless, networking and automotive products in mind.

As well as licensing microprocessor cores, ARM also licenses its architectures to other manufacturers. For example, the Intel® XScale™ Microarchitecture is based on the ARM architecture – ARMv5TE. This paper also provides a summary comparison between the ARM11 microarchitecture and the Intel XScale-based products.

Target Applications

The ARM11 microarchitecture addresses a broad range of applications in the wireless, consumer, networking and automotive segments.

Its media processing capability, and low power characteristics make it particularly suited to wireless and consumer applications. While high data bandwidth coupled with a high performance core are well adapted for networking applications. Features designed to enhance real-time performance, and availability of floating point products are highly attractive for high performance automotive applications.

ARM11 cores will enable the consumerization of next-generation portable and wireless applications. Portable consumer applications are becoming more sophisticated with each new generation. User expectations are for better user interfaces, faster graphics, and larger screens to support a broader range of applications, such as video phones, games, news services, and movies. Typical design requirements then include: better operating-system support, and support for streaming multimedia applications. At the same time, achieving low product cost, and low power, remain critical factors in ensuring that adoption of high-end portable and wireless appliances becomes ‘mainstream’.

Convergence between computing and communication will continue to advance with the progressive rollout of third-generation wireless services. Growth of text messaging (based on GSM’s short message service - SMS) has been phenomenal – current estimates suggest a billion text messages per day are sent[1]. The increased bandwidth available with 3G services will see ‘texting’ supplemented with multimedia messaging (MMS). The need for enhanced performance is also being driven by entertainment applications such as games, digital audio and streaming multimedia applications.

Adding graphics and video services to basic voice and text communication requires a new generation of wireless handsets, and sees convergence with other types of devices such as digital cameras. The effect will be to transition the high-end pocket-PC/PDA type appliance from early-adopter status to the mainstream consumer market.

Key Benefits

For portable and wireless applications, delivering high performance does not come without constraint. As well as minimizing product cost, power consumption is a major concern. The ARM11 microarchitecture represents a major step in system performance – with target performance for the first cores in the 350-500MHz range (Figure 2), and a

[1] GSM Association October 2001 Press Release

roadmap to over 1GHz. The microarchitecture delivers performance efficiently, and allows developers to make the trade-off between performance and power to match their particular application. By scaling both the clock frequency and the supply voltage, the developer can control power consumption and performance. Processor implementations based on the ARM11 microarchitecture will achieve less than 0.4mW/MHz at 1.2V in a 0.13µm process technology.

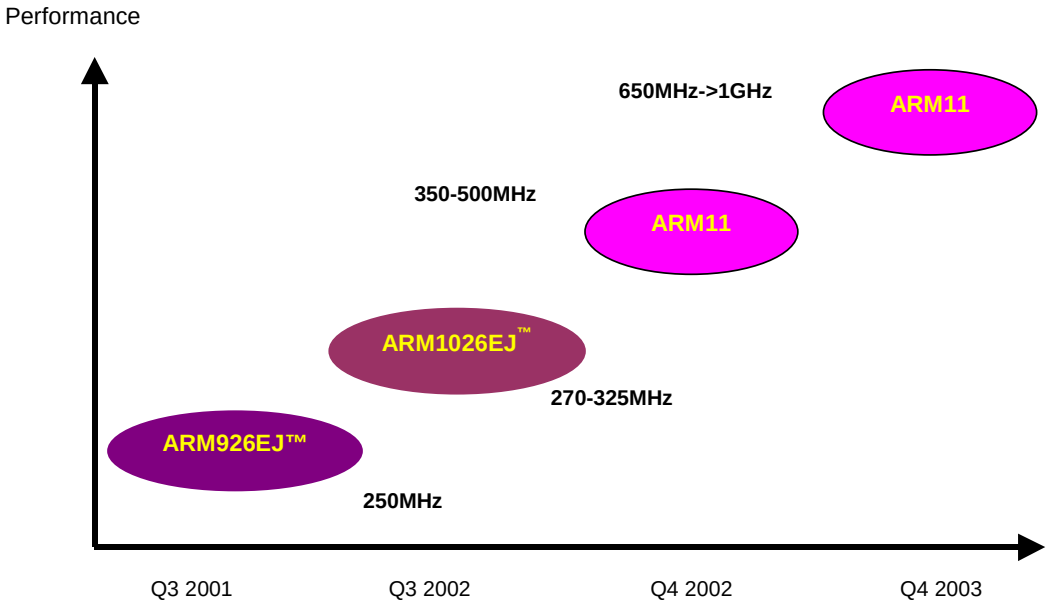

Figure 2. ARM Processor Roadmap

The ARM11 microarchitecture has been developed both for synthesizable and semi-custom hard macrocell implementations. Since, primarily, the cores will be used within a system on a chip (SoC) context, enabling implementation through synthesis allows developers to easily integrate the design. Synthesis also allows licensees to add value to the design by exploiting the particular strengths of their semiconductor processes. Hard macro implementations are targeted at the highest performance, speed-sortable applications.

To better facilitate synthesis, ARM11 cores have a synthesis-friendly pipeline structure, designed to work with commercially available synthesis tools and RAM compilers, that ensures timing closure is readily achieved. The resulting implementation of the synthesized ARM11 core excluding caches will typically occupy less than 2.7mm^2 – a very small footprint in the context of today's complex SoCs.

The ARM11 microarchitecture – through the ARMv6 architecture, also benefits software developers in a number of ways. It includes a large set of media processing instructions to accelerate audio and video applications, it includes a new memory system architecture to accelerate operating system performance, and it includes new instructions to accelerate real-time performance and interrupt response. In addition, since many new applications

require multiprocessor configurations – for example, multiple ARM cores, or ARM plus DSP systems, it is now easier to share data with other processors, and easier to port applications from non-ARM processors.

ARM also provides a complete set of supporting IP for ARM11 processors including the ARM PrimeCell® IP library, as well as a significant amount of third-party AMBA™-compliant IP.

Performance

The superior performance delivered by ARM11 cores is enabled by a combination of architecture and microarchitecture features.

ARMv6 – the Performance Foundation

The ARMv6 architecture provides the foundation for achieving a step-up in system performance. In summary, ARMv6 provides for enhanced performance through:

- Media processing extensions
  - 2x faster MPEG4 encode/decode
  - 2x faster audio DSP
- Improved cache architecture
  - Physically addressed caches
  - Reduction in cache flush/refill
  - Reduced overhead in context switches
- Improved exception and interrupt handling
  - Important for improving performance in real-time tasks
- Unaligned and mixed-endian data support
  - Simpler data sharing, application porting and saves memory

ARMv6 maintains 100% backward compatibility at the binary level for most applications, enabling easy porting of legacy applications. The ARMv6 architecture also provides all Thumb®, and 'E', instructions for code-compression and DSP processing, and implements the ARM Jazelle™ extensions for Java™ acceleration.

More information on all of these improvements available in ARMv6 is available [1].

ARM11 Microarchitecture – Key Features

The ARM11 microarchitecture has been designed to deliver high-performance – efficiently. The design of the pipeline is key to enabling this.

The ARM11 microarchitecture pipeline differs from preceding ARM cores. It consists of 8 stages, enabling a throughput of around 40% higher than previous cores. An 8-stage pipeline allows 8 different processing stages to be carried out simultaneously.

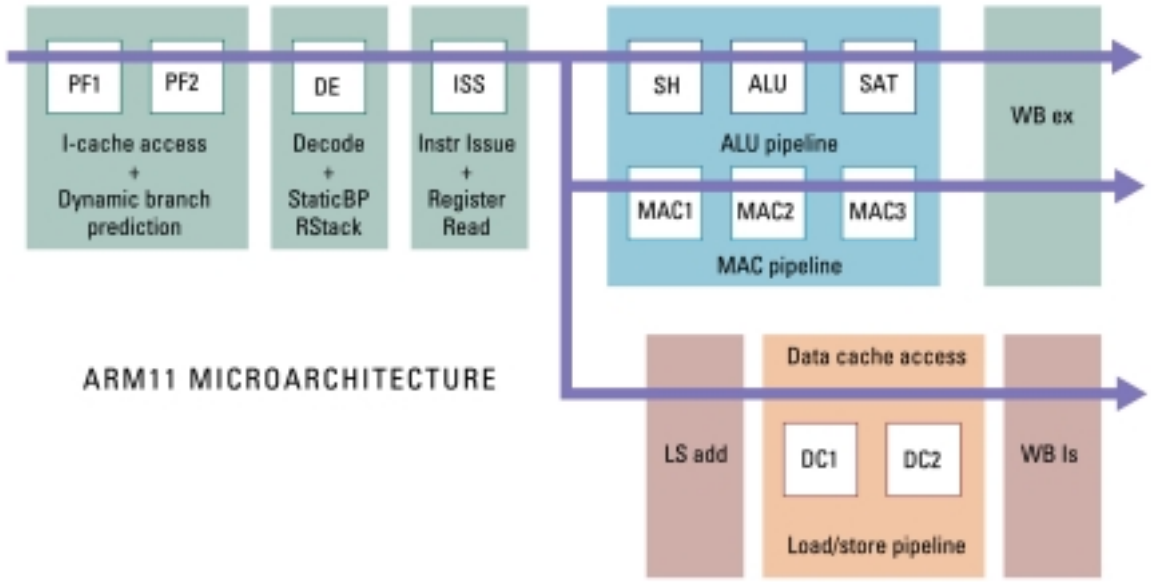

Figure 3. ARM11 Pipeline Structure

Highly-pipelined structures can impair efficiency by introducing excessive delays, or latency, into the system. In general, long pipelines mean that execution of some instructions may be delayed, because they depend on the results of previous instructions - the ARM11 pipeline avoids these delays by extensive use of *forwarding* within the pipeline. A processor with a long pipeline may also lose performance when something happens to interrupt the smooth flow of instructions through the pipeline – for example, a branch instruction. The ARM11 pipeline avoids these delays by using branch prediction to predict the flow of instructions. These forwarding and prediction techniques maintain good pipeline efficiency by reducing pipeline 'stalls' – i.e. situations where the processor has to wait because its next instruction is still rippling through the pipeline.

Together, these pipeline optimizations deliver much higher throughput, but at the same effective latency as a 5-stage pipeline, such as that found in the ARM9™ microarchitecture.

Single Instruction Issue

The ARM11 microarchitecture pipeline is *scalar*, in other words it issues a single instruction at a time (single issue). Some pipeline architectures are capable of issuing multiple instructions at a time – for example, to the ALU and MAC pipelines.

In theory, *multiple issue* microarchitectures should have an efficiency advantage. In practice, multiple issue machines vastly increase the complexity in the front-end pipeline instruction decoding stages, since the processing of instruction dependencies becomes much more involved. This increased complexity has a significant impact on the power consumption and area of the processor.

The analysis made in developing the ARM11 microarchitecture determined that the potential for increased performance from multiple instruction issue did not justify the penalty in increased power and area.

The following sections explain some of the specific pipeline optimizations in more detail.

Managing Instruction Branches

Branch instructions are usually conditional instructions. In other words, they require some condition to be tested – such as examining a condition code register, before jumping to another instruction, or just continuing through the current sequence. Branching can cause delays in the pipeline, since the result of the condition code needed by the branch instruction may not be available until three or four cycles after the instruction decoder encounters the branch. Predicting where a branch should be directed (its *destination*), will help avoid this delay.

The ARM11 microarchitecture uses two techniques to predict branches. First, the dynamic branch predictor uses a historical record to see whether the branch has been seen before, and whether it was most-frequently taken, or most frequently not taken

A 64-entry, 4-state branch target address cache (BTAC) is maintained. This table is sufficient to hold the majority of most recent branches – those that are most likely to be seen again

If the branch prediction has been encountered before, a prediction is made based on the previous outcome.

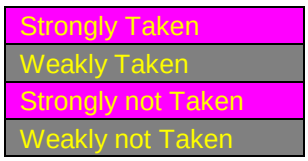

Figure 4. Branch Prediction States

If the dynamic branch predictor cannot find a record of the branch instruction, a static branch prediction procedure takes over.

Very simply, the static prediction looks at the branch to see whether it is going backwards or forwards. If backwards, it assumes it's a loop, and takes the branch. If it's a forward branch, it doesn't take the branch.

The return stack manages branch prediction for returns from up to three procedure calls. Similarly, these are treated as conditional branches.

As well as predicting branches, the ARM pipeline will also *fold* the branches. This means that if the result of the branch prediction is that the branch won't be taken, the original branch instruction is removed or 'folded' from the pipeline, as there is no point in executing it. This saves a clock cycle for a correctly predicted branch.

The net benefit of the dynamic and static branch prediction employed in the ARM11 microarchitecture is that around 85% of branches are correctly predicted – saving typically five processor clock cycles for every correctly-predicted branch.

Improved Memory Access

System design and performance is heavily affected by the way that memory is managed. The memory management architectural enhancements improve the overall processor performance significantly – especially for platform-type applications where operating systems need to manage frequent task changes.

With the changes in ARMv6, instructions and data for more tasks can remain resident in the cache for longer, so cache misses are much more infrequent, and the performance of the processor is increased.

One of the microarchitectural improvements in ARM11 cores is the *non-blocking*, and *hit-under-miss* operation of the memory system. When an instruction requests data from a cache, if the data is not there, a cache 'miss' results, and the pipeline of a simpler processor design would stop. However, ARM11 treats this as a *non-blocking* operation.

The cache is instructed to get the missing data, then the pipeline execution can continue as long as the next instructions are not dependent on the missing data. Even if the next instruction is another data load, the ARM11 microarchitecture permits this operation if the data is in the cache (i.e. a hit-under-miss). Only if three successive data misses are encountered, will the pipeline stall. For most applications, this is a very infrequent occurrence.

Pipeline Parallelism

Although the pipeline is single issue, parallelism is exploited in the back end of the pipeline architecture, with separate processing units for the ALU, multiply-accumulate (MAC) and Load-Store (LS) instructions.

The LS pipeline is dedicated to processing load and store instructions. Decoupling the execution of arithmetic instructions from memory instructions (LS) enables more efficient processing as Execution of LS instructions can often be constrained by the availability of external memory (see the explanation of non-blocking and hit-under-miss operation above for how the ARM11 microarchitecture is optimised to cope with this).

In microarchitectures where the LS unit is part of the main pipeline, the pipeline can be easily stalled waiting for memory operations to complete. With a separate LS pipeline as implemented in the ARM11 microarchitecture, the execution of an ALU or MAC instruction will not be delayed by a waiting LS instruction. This also allows a software compiler more freedom in scheduling code, which also helps to improve performance.
Processing LS instructions is split over two cycles for data cache access. Pipelining the LS unit in this way, means that ARM11 cores are suitable for use in ASIC designs built with commercially available RAM compilers: high-speed, custom memory is not necessary to ensure maximum processor performance.

To get the most out of the parallel pipelines, the ARM11 microarchitecture enables '*out-of-order completion'*. This means that instructions that have no dependency on the outcome of previous instructions can complete their execution without waiting. Completed instructions free up the resources for execution of the next instruction.

64-bit Data Paths

For many of the current target applications, a true 64-bit processor is still considered to be unecessary. Implementation costs would be excessive, as would power and area. The ARM11 microarchitecture exploits 64-bit structures in the processor where this makes sense, without the need for a complete 64-bit solution. The result is delivery of 64-bit effective performance, but at a 32-bit cost.

The ARM11 microarchitecture specifies 64-bit data buses between the processor integer unit and the instruction and data caches, and between coprocessors and the integer unit. These 64-bit paths allow two instructions to be fetched from the cache in a single cycle, and allow load- and store-multiple instructions to transfer 64 bits (two ARM registers) every cycle. This allows ARM11 processors to achieve very high performance on many code sequences, especially those that require data movement in parallel with data processing.

Floating Point Processing

The ARM11 microarchitecture supports floating point as a design option. Products in development include cores with and without floating point processing units. This allows application developers to choose floating point for those applications that demand it, and to optimize cost where floating point is not required.

ARM11 Microarchitecture Comparisons

Figure 5 highlights some key feature comparisons between the ARM11 microarchitecture, and other currently available ARM architecture-based designs. Intel's XScale is based on the ARMv5TE architecture, which Intel licenses from ARM.

As Figure 5 illustrates, the ARM11 provides extended performance compared to the ARM9E™ and ARM10E™, and shares many common features with the Intel XScale microarchitecture.

The key differences in pipeline details stem from the target performance range and intended implementations. ARM11 based cores target high performance, but are also intended to be easily integrated in a broad range of SoC designs. To that end, ARM11 products support a synthesis design flow, as well as semi-custom, hard-macro implementations.

The ARM11 microarchitecture also provides upward compatibility to ensure that OEMs can choose any ARM core-based product, supplier, or implementation with confidence.

Feature	ARM9E™	ARM10E™	Intel® XScale™	ARM11™
Architecture	ARMv5TE(J)	ARMv5TE(J)	ARMv5TE	ARMv6
Pipeline Length	5	6	7	8
Java Decode	(ARM926EJ)	(ARM1026EJ)	No	Yes
V6 SIMD Instructions	No	No	No	Yes
MIA Instructions	No	No	Yes	Available as coprocessor
Branch Prediction	No	Static	Dynamic	Dynamic
Independent Load-Store Unit	No	Yes	Yes	Yes
Instruction Issue	Scalar, in-order	Scalar, in-order	Scalar, in-order	Scalar, in-order
Concurrency	None	ALU/MAC, LSU	ALU, MAC, LSU	ALU/MAC, LSU
Out-of-order completion	No	Yes	Yes	Yes
Target Implementation	Synthesizable	Synthesizable	Custom chip	Synthesizable and Hard macro
Performance Range	Up to 250MHz	Up to 325MHz	200MHz – >1GHz	350MHz - >1GHz

Figure 5. ARM Architecture Feature Comparisons

Summary

The ARM11 microarchitecture is the first implementation of the new ARMv6 instruction set architecture. It forms the basis of a new family of synthesizable cores, for easier SoC integration, and semi-custom cores for the highest performance applications.

ARM partners will be able to differentiate and optimize ARM11 cores for power and performance, exploiting the characteristics of their own process technologies. The new microarchitecture is targeted at next-generation high-end portable and wireless applications, consumer, networking, and automotive applications. There are many features that will also make ARM11 processors highly suited to high-end embedded real-time applications, such as future networking and in-home entertainment products now require.

ARM11 cores benefit from the complete ARM development environment and comprehensive IP solutions, as well as the extensive ARM developer and partner community.

References

[1] The ARM Architecture Version 6 (ARMv6)
David Brash
January 2002
http://www.arm.com/support/White_Papers