WHITE PAPER ARM

Cortex-R4

A comparison with the ARM9E processor family

Peter Lewin, ARM

May 2006

Introduction

The ARM9E family of processors has been successfully used for many years in a wide range of applications, including hard disk drives, imaging, automotive, networking and wireless devices. Their low power, small size and high performance, combined with industry standard architecture make them ideal for these applications.

The Cortex-R4 processor builds on this foundation, by increasing performance while keeping system costs low. This is achieved through a variety of new technologies and design improvements which increase the computing efficiency of the processor. By providing a number of synthesis time configuration options, the Cortex-R4 is able to address applications previously covered by the ARM946E-S, ARM966E-S and ARM968E-S processors, and expand this applicability into more demanding situations. This white paper will examine the key differences between the ARM9E processor family and the Cortex-R4 processor.

Performance and efficiency

Pipeline

The pipeline length of the Cortex-R4 processor is increased from the five stages used in the ARM9E family, to eight. This reduces the amount of logic required in each stage, allowing a higher operating frequency on a given process and library. For details of the maximum frequency of each core on various processes, see http://www.arm.com/products/CPUs.

In addition to the length of the pipeline being increased, the later stages of the pipeline are split into four parallel pipelines, each handling different types of instruction (in some cases concurrently):

1. Load Store: This pipeline handles all memory accesses. Memory accesses are split across two pipeline stages, to allow longer RAM accesses times without loss of bandwidth.

2. MAC: Multiply operations are split over three pipeline stages, the final one of which also updates the register bank.

3. ALU: Arithmetic operations use an operand pre-shift stage, a basic ALU operation stage, then optionally saturate before updating the register bank.

4. Divider: The divider uses a Radix-4 algorithm, with a typical 32 bit divide taking around 6 cycles in a single pipeline stage.

This is in contrast to the ARM9E processor pipeline, where each of the 5 pipeline stages only process one instruction at a time.

With the exception of the divider, the separate pipelines advance together. This keeps the instruction execution in order and avoids the need for extensive logic associated with out of order completion. However, the divider is decoupled to prevent the other pipelines stalling while a divide is completed. Data hazards are detected, resulting in the other pipelines stalling if they require the result from a divide operation which has not yet completed.

The load-store pipeline is skewed relative to the other pipelines, by performing the address generation in the issue stage. This keeps the load-use penalty to one for common loads; this is the same as on the ARM9E processor family. Load-use penalty refers to the delay caused when data loaded is required immediately by following instructions.

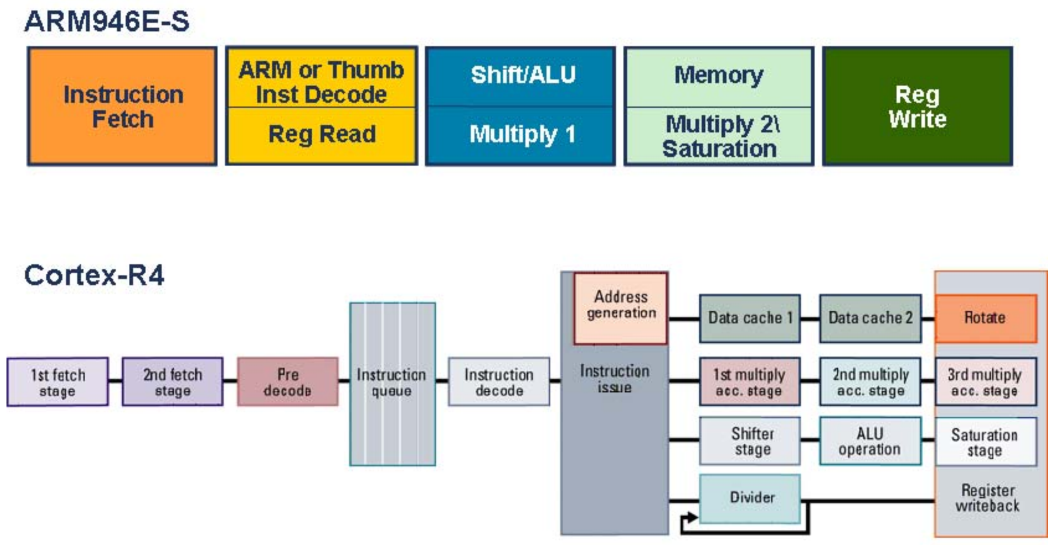

Figure 1: Comparison of ARM9E and Cortex-R4 pipelines

Dual Issue

The Cortex-R4 pipeline structure allows a limited degree of dual issuing without the cost of duplicating execution stages. A second, limited decode unit is provided, which allows certain pairs of instructions to be decoded and issued in parallel. For example, if a load instruction is followed by an add instruction, it may be possible to issue the load instruction to the load-store pipeline at the same time as the add instruction is issued to the ALU pipeline. This results in a large improvement in CPI with very little extra silicon overhead. The dual issue capability is discussed in detail in [1].

As a result of these features, the Dhrystone performance of the Cortex-R4 processor is increased to 1.62 DMIPS/MHz, from 1.14 DMIPS/MHz for the ARM946E-S processor. This allows for either a 42% increase in performance for a given frequency, or a reduction in frequency (and hence power consumption) for a given workload.

Silicon area

The increased complexity of the Cortex-R4 processor pipeline has the potential to increase the silicon area, and therefore cost. Careful design has kept this increase to a minimum – indeed for target frequencies close to the top of the range for the ARM946E-S processor, the Cortex-R4 processor will actually be smaller as synthesis tools find it easier to meet timing constraints and perform more area optimisations. For the latest details on the area of various processors see http://www.arm.com/products/CPUs.

Branch Prediction

A longer pipeline can also have the effect of increasing the processor's CPI (cycles per instruction), by increasing interlocks due to data dependencies (where one instruction can not progress until the result from a previous instruction is available), and by increasing the branch penalty (time taken to refill the pipeline following a branch). A number of measures are taken to offset this effect, including extensive data forwarding and branch prediction. The branch prediction reduces the number of pipeline flushes required by predicting whether each branch instruction will be executed early in the pipeline. When this prediction is correct, this allows the core to fetch instructions from the correct location after the branch, thus avoiding the need to flush the pipeline once the branch is executed. An eight bit global branch history scheme is used, in addition to a return stack to allow the correct prediction of function return addresses. This scheme provides good accuracy without the need for a large cache of previous branch outcomes. The ARM9E processor family does not perform branch prediction, resulting in a pipeline flush each time a branch is taken.

AMBA 3 AXI

The Cortex-R4 has a 64-bit AMBA 3 AXI memory interface, compared to the 32-bit AMBA AHB interface used on the ARM946E-S processor. There are a number of performance gains obtained by the switch to AMBA 3 AXI, including the issuing of multiple outstanding addresses and support for data to be returned out of order. The most significant advantage for many applications will be the fact that a slow memory or peripheral does not block the bus for the duration of its access, allowing the core to perform further accesses rather than waiting for the slow one to complete. Widening the bus to 64-bit also increases the available bandwidth, allowing a cache linefill (8 words) to be completed in four accesses rather than eight.

Interrupt latency

There are a number of features to improve both the worst case and the average interrupt latency of a Cortex-R4 processor system. These include the ability to abandon a load multiple instruction after it has started, new instructions to store and change the processor state at the start of the interrupt handler, and the non blocking nature of the AMBA 3 AXI bus.

The worse case latency for a fast interrupt (FIQ) on an ARM946E-S processor occurs when the FIQ is signalled immediately after a 'Load Multiple" instruction (LDM) of 16 words has started. The ARM946E-S processor can not abandon this instruction, or process the interrupt until it completes. The 10 words may span 3 cache lines (which are 8 words each), causing 24 words to be loaded over the AMBA AHB bus. Additionally, each cache line may contain dirty data in both halves, requiring these lines (24 words) to be written back to memory. If the write buffer is full this must also be drained, requiring a further 8 AHB writes. The final load may cause a data abort, which adds a further 3 cycles to the response time. Assuming a 2:1 core to AMBA AHB clock ratio this will take 118 cycles, even with zero wait state memory. Although these conditions are unlikely to occur frequently, a real-time system must allow for this worst case. Even if the maximum load multiple is limited to 4 words, we assume no external aborts, only half the cache lines being dirty and the write buffer only being half full, the latency will be around 60 cycles. As most of these cycles involve bus accesses, any wait states introduced by the memory system will increase this latency considerably.

The Cortex-R4 processor will abandon a load multiple instruction from normal memory if an interrupt request is received part way through its execution. This avoids the interrupt latency associated with completing up to 16 data reads. The interrupt service routine (ISR) can then be fetched from the instruction cache or TCM while the data cache linefill completes, as the core is no longer waiting on the returned data. In addition, the use of the AMBA AXI ID field also allows the ISR to be fetched over the AMBA AXI bus without waiting for a previous cache line fill to complete. The Vectored Interrupt Controller (VIC) port enables the address of the ISR to be delivered to the pre-fetch unit without accessing a peripheral over the AMBA AXI bus. Even if the VIC port is not used, peripherals can be accessed over the AMBA AXI bus without waiting for the previous linefill to complete. This is enabled by the use of a different AMBA AXI ID for cacheable and non-cacheable reads, which allows these to complete out of order. Providing strongly ordered and device memory (from which a load multiple can not be abandoned) is used carefully, the maximum interrupt latency will be around 20 cycles, with little or no dependency on the access times of AMBA AXI memory and peripherals.

A non-maskable interrupt option is also available on the Cortex-R4 processor, preventing software from disabling the fast interrupt requests (FIQ). This is particularly important for safety critical applications.

System Cost

The cost of using a particular processor in an ASIC is not limited to the silicon area occupied by the processor itself. All processors require various support from other blocks, such as memory, peripherals and bus infrastructure to perform their function. In addition, development costs and times must be factored into the overall cost. One of the major costs in terms of silicon area is memory, and the Cortex-R4 processor includes a number of features to reduce this cost.

Thumb-2

The Cortex-R4 processor implements the ARMv7R architecture, including the Thumb-2 instruction set alongside the original ARM instruction set. The ARM9E family of processors implement the ARMv5TE architecture, which includes ARM and Thumb instruction sets.

The ARM instruction set has fixed instruction width of 32-bits. This allows a very powerful instruction encoding providing maximum performance on the ARM9E processor family. Thumb is an alternative instruction set using a reduced instruction width of 16-bits. This allows for approximately a 35% improvement in code density (Thumb code to implement a given function is approximately 35% smaller than the equivalent ARM code). However, as less functionality can be encoded in a 16 bit opcode, the performance of Thumb code is lower than that of ARM code. In addition, the Thumb instruction set does not give access to all of the architecture (for example, it does not allow you to mask interrupts), so all ARM9E processor family-based systems will use some ARM code. ARM code and Thumb code are generally mixed on a function by function basis, with the software writer responsible for deciding which instruction set is most appropriate for each function.

Thumb-2 contains all the 16-bit instruction opcodes from the Thumb instruction set, and is therefore binary compatible with existing Thumb software; ARM9E processor family code will run on a Cortex-R4 processor without recompilation or re-assembly. However, it supplements these with a large range of 32-bit instructions to provide the full functionality of the ARM instruction set. This means that 16- and 32-bit instructions can be mixed on an instruction by instruction basis and, crucially for development costs, the optimum instruction size mix can be effectively selected by a compiler. The result is that Thumb-2 code can retain the high performance of ARM code, while giving the code density benefit of Thumb. Compared to an ARM9E processor family-based system running ARM code, this allows for a reduction in the amount of program memory required. When compared to an ARM9E processor family-based system running Thumb code, this allows a reduction in the required operating frequency for a given performance point. The benefits of the Thumb-2 instruction set are discussed further in [2].

RAM requirements

The ARM9E processor family and the Cortex-R4 processor both include support for local memory, in the form of TCM and caches. These require on chip RAM to be implemented that can operate at the core frequency. For the ARM946E-S processor, this RAM requires a response time of approximately 40% of the core clock cycle time. For example, an ARM946E-S processor running at 200MHz (a cycle time of 5ns) requires a RAM response time of 2ns.

The Cortex-R4 processor pipelines accesses to local RAMs over 2 cycles. This means that the access time required of the RAM is increased to 100% of the core clock cycle time. So at 200MHz, a response time of 5ns is required, and even at 400MHz the required response time of 2.5ns is longer than that required by a 200MHz ARM946E-S processor. This enables the choice of a lower speed memory library, which can drastically reduce both silicon area and power consumption. It may, for example, allow the use of Artisan Metro RAM rather than Artisan Advantage RAM, giving a 35% area reduction and a 54% power saving. In addition to the area and power reduction, this will make timing closure much easier, shortening the design cycle and reducing risk.

Flexibility

When synthesising the ARM946E-S processor, there are a few configuration options that can be altered, such as the cache size and TCM size. The Cortex-R4 processor extends these configuration options to allow the processor to be more closely aligned with the application's requirements. This configurability also allows the Cortex-R4 processor to address a wider range of applications.

TCM flexibility

Both the ARM946E-S processor and the Cortex-R4 processor support a local memory architecture, referred to as Tightly Coupled Memory (TCM). Both support the use of TCM for instructions and data. In the case of the ARM946E-S processor these must be implemented as two physically separate RAMs, one for instructions and one for data. Code can not be run from the data TCM, and although data can be accessed in the instruction TCM, there is a performance penalty for doing this. These restrictions dictate that the split between the size of TCM available for instructions and for data is fixed separately when the core is synthesised.

The TCM on the Cortex-R4 processor is far more flexible. There are three memory ports, appearing to the programmer as two separate memory regions. Two of the ports can be combined to access a single address region, either on a bottom / top half basis, or a finely interleaved basis. In the interleaved configuration, evenly addressed double words are stored in one RAM, while oddly addressed double words are stored in the second RAM. At synthesis time, the designer may choose to implement one, two or three separate RAMs. Where separate RAMs are implemented, the core can access these in parallel, hence increasing performance. Unlike the ARM946E-S processor, these RAMs do not have to be designated as instruction or data memory at synthesis time; the processor contains an internal bus matrix that can route either type of access to any of the implemented RAMs. There will only be a delay when simultaneous instruction and data accesses are located in the same RAM. These memories are 64-bits wide on a Cortex-R4 processor, compared to 32-bits wide on the ARM946E-S processor, further increasing bandwidth. Additionally, the ARMv7 instruction set reduces the need for literal pool accesses (data stored with the program code). This means that the performance penalty should the designer choose to implement only one TCM RAM is minimal.

In many cases, a single logical memory will be implemented as two separate blocks in order to improve layout and timing. In these cases, there will be no extra cost associated with using two TCM ports, the connections will be more straightforward, and the need for a MUX will be eliminated.

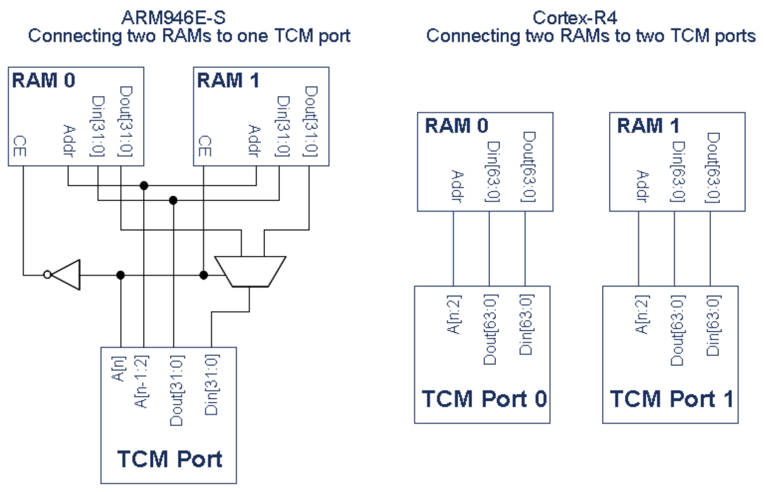

Figure 2: Tightly Coupled Memory - RAM connection

DMA

The Cortex-R4 processor also introduces a DMA port that was not present on the ARM946E-S processor or the ARM966E-S processor. This is a slave AMBA AXI port that allows an external DMA controller (or another processor) simple access to the internal TCM. Accesses through this port are arbitrated using the same internal matrix as the core's instruction and data accesses, and can occur in parallel to these. If the TCM is implemented as interleaved double words, the core and DMA accesses can effectively access the same memory range simultaneously by accessing alternate addresses (for example, when streaming data into the TCM for the core to process). This can provide much of the benefit of using dual ported RAM, without the associated cost.

MPU

The ARM946E-S processor has an 8 region MPU, with a minimum region size of 4Kbytes. The MPU is not optional, so will always be present in an ARM946E-S processor-based system (alternate members of the ARM9E processor family are available without an MPU, though these do not have caches). The MPU on the Cortex-R4 processor can be configured with 8 or 12 regions, allowing additional flexibility if required but avoiding the associated silicon area if not. The MPU can also be omitted completely, resulting in a fixed mapping of protection attributes. The minimum size of an MPU region is 32 bytes, allowing finer control and reduced memory wastage.

Architecture

The ARMv7 architecture consists of 3 profiles (Application profile, Real time profile and Microcontroller profile). The Cortex-R4 processor implements the Real time profile (ARMv7R). This enables the architecture as well as the implementation to be optimised for the particular application space. For example, the Real Time profile (and therefore the Cortex-R4 processor) supports hardware divide instructions, which are particularly useful for embedded control applications. ARMv7R is binary backward compatible with the ARMv5 architecture, as implemented on the ARM946E-S processor. This means that code compiled for the ARM946E-S processor will run on the Cortex-R4 processor without recompilation. Recompilation will allow the code to be optimised for the Cortex-R4 processor (and targeted at Thumb-2). Additionally, some system code may need to be altered – for example to use the new MPU region sizes and attributes.

Summary

The Cortex-R4 processor offers a substantial increase in performance over the ARM9E processor family, both in terms of maximum operating frequency and computing efficiency. In addition, the increased configurability allows the processor to be closely matched to the application's requirements. These improvements have been made without sacrificing the low power consumption and size that have made the ARM9E processor family so successful. The Cortex-R4 processor is therefore ideal for systems that require any combination of higher performance, lower operating frequency (and hence power consumption) and lower costs. Costs can be lowered both by reducing the required amount of RAM, and by reducing development costs.

The ARMv7R architecture ensures compatibility with the existing ARM code base, while allowing optimization – particularly by using the Thumb-2 instruction set.

References

[1] "Cortex-R4: A mid-range processor for deeply-embedded applications", ARM whitepaper.

[2] "System solutions for a baseband SoC", ARM whitepaper, Dom Pajak.

[3] "Performance of the ARM9TDMI and ARM9E-S cores compared to the ARM7TDMI core", ARM whitepaper

[4] "Cortex-R4 Technical Reference Manual", ARM

[5] "ARM9E Technical Reference Manual", ARM

[6] "ARM946E-S Technical Reference Manual", ARM
http://www.arm.com/documentation