Cortex-R4 processor

High-performance and high-reliability for deeply-embedded real-time systems

Chris Turner, ARM   **May 2010**

This paper presents the Cortex[TM]-R4 processor core from ARM Limited. It describes the processor's features and intended applications and gives detailed explanation of the core architecture, its configurability, implementation and system behaviour including memory interfaces and floating point unit. Some comparisons are made with the earlier ARM9 and ARM11 processors used in similar applications.

Contents

Introduction

The ARM® Cortex™-R4 processor enables a wide range of deeply-embedded, high performance and high reliability products to be implemented with reduced system and development costs. Applications for Cortex-R4 include consumer, mobile, storage, networking and automotive devices.

The Cortex-R4 implements the ARMv7R architecture for compatibility throughout the Cortex processor family and also for backward compatibility with the ARM9 and ARM11 series processors. Cortex-R4 is highly configurable for precise application requirements with synthesis options that include Instruction and Data cache controllers, Tightly-Coupled Memory (TCM) interfaces, memory protection, error correction or parity checking, debug and trace and a Floating-Point Unit (FPU).

The processor's advanced micro-architecture and feature set includes the Thumb-2 instruction set architecture, which significantly reduces Cortex-R4's Cycles per Instruction (CPI) compared with other processors in its class. Efficient implementation ensures excellent power, performance and area attributes for the Cortex-R4. Other features such as a single AMBA® 3 AXI™ interface and memory system enhancements make the Cortex-R4 easy to integrate in a System on Chip.

This paper explores the market and technology drivers for deeply-embedded products and expands on the technical features that make the Cortex-R4 processor highly suited to a broad range of cost-sensitive, deeply embedded applications. In this paper, the processor is always referred to as Cortex-R4, although elsewhere it has also been called Cortex-R4F when the FPU is present.

Deeply embedded applications

Deeply embedded applications are found across a broad range of markets. Products such as hard disk drives, cell-phones and other wireless modems, printers and home gateways are all essentially consumer products and strong consumer demand has increased semiconductor shipments in many of these markets. Other key markets require high performance coupled with high reliability, for example aerospace or automotive applications such as chassis and braking systems.

To be competitive, these embedded processor products must deliver optimum performance at the lowest possible cost. Engineers must avoid creating a design with excess performance or additional features that the application does not need as it results in unnecessary silicon area, cost and power dissipation. The Cortex-R4 can be optimised for its target applications and it enables embedded CPU system designers to deliver highly competitive products.

The Cortex-R4 processor was released in 2006 and is intended for implementation in advanced semiconductor processes from the 90 nm node onward. As the semiconductor industry advances, it enables ever-higher gate count products at lower cost and the Cortex-R4 puts these affordable gates to good use, delivering high processing performance within acceptable limits of clock frequency and power dissipation.

Achieving faster time-to-market through configurability and advanced debug support, reducing overall system costs via improved code density to reduce memory size, as well as harnessing design expertise through industry standards such as AMBA, are all factors which will enable system designers using Cortex-R4 to satisfy the demands of intensely competitive markets.

Examples of deeply embedded applications that are growing in complexity and require increased performance and functionality from processor cores include:

Imaging

The personal imaging and printing markets have evolved from their traditional computer-centric peripherals toward a new class of intelligent stand-alone networked products.

Imaging devices, ranging from low-end inkjet printers and digital cameras to high-performing laser printers and multifunction peripherals, perform more computing-intensive operations. Tasks such as image processing, colour correction, JPEG interpretation, and PDL/PCL handling are now performed autonomously, allowing image-rich content such as web pages, digital photos, and scanned documents to be captured, stored, viewed, transmitted, and printed without using a PC.

Manufacturers of imaging products operate in highly competitive consumer markets and developers need to be ahead of the industry with advanced products that anticipate consumer demand.

Automotive

In the automotive market, advanced electronic control units are driving innovation in areas such as in-car entertainment systems, navigation systems, telematics, in-car safety devices and diagnostics.

For safety-critical body, chassis and power-train applications, such as anti-lock braking systems (ABS), reliability is paramount and qualification cycles are long. These automotive systems are becoming increasingly complex; for example, electronic stability control (ESP) is an order of magnitude more complex than ABS. Designers must be able to integrate testing logic within a processor-based product and then demonstrate that it has a high level of fault coverage.

Storage

Advances in disk drive technology and the spiralling demand for increased storage capacity in both enterprise and consumer electronics is driving the use of disk drives for data intensive and high-speed interface applications. Demand is driven by products such as Network-Attached Storage (NAS) drives and portable consumer electronics such as media players and digital cameras.

These storage devices are being deployed in applications with wide-ranging constraints; for example, a PC disk drive will typically be optimized for random data access, whereas a portable media player requires the same hardware but requires configuration for streaming data access, low power, low noise and robust head-crash protection.

Communications

Use of wireless communications continues to expand as deployment of cellular phones at 3G and beyond increases and as new products such as wireless access points and media devices appear. Wired networking shows continued growth as use of the Internet increases and customers require high performance from broadband networks with increasing demand for enterprise networking equipment. All of these product segments require higher processing performance as data rates increase, protocols become richer and modulation schemes become more complex. Low latency and a rapid response to interrupts are essential for all of these products.

ARM has considerable experience in the deeply-embedded market which has enabled it to specify the Cortex-R4 to meet requirements throughout these, and other, deeply embedded applications. Cortex-R4 brings competitive advantage to products through the following features:

- Reduced cost: The Cortex-R4 Thumb-2 instruction set gives high code density, reducing memory size and cost. The processor's micro-architecture also features relaxed memory system timing allowing smaller, lower cost RAMs to be compiled. Cortex-R4 is also highly configurable so designers can minimise die area by only including the features they require.

- High performance at low power: Cortex-R4 provides increased processing performance at lower clock frequency and a very fast interrupt response. So, in many applications the processor clock frequency can be minimised for low dynamic power consumption.

- High reliability: Cortex-R4 and its local memory system have integrated soft-error management and always exhibit deterministic behaviour with a guaranteed interrupt response.

- Design flexibility: Cortex-R4 features are selected by designers prior to synthesis and include a configurable local memory architecture using Tightly Coupled Memories (TCM) and Level-1 Instruction and data caches. The TCMs are highly flexible for partitioning of instructions, program literals and data across one to three physical memories that can be ROM, RAM or Flash.

- Safety critical: Cortex-R4 has specific support for OSEK and AutoSAR automotive software with its deterministic performance, high-reliability and fault-tolerant configurations, including a dual core lock-step option. Semiconductor partners use Cortex-R4 with FPU to deliver a versatile and high-performing range of cost-effective, fully-featured automotive microcontrollers.

Features and configuration

The Cortex-R4 architecture is shown below with configurable and removable options in dashed boxes. In minimum configuration and synthesised for lower clock frequencies, the core's gate count can be as low as 150k gates. Gate count increases to approximately 290k gates in full configuration, synthesised for maximum clock frequency. On a 40 nm GP process a maximum clock frequency of 934 MHz is possible, yielding a processing benchmark performance of almost 1,500 Dhrystone MIPS.
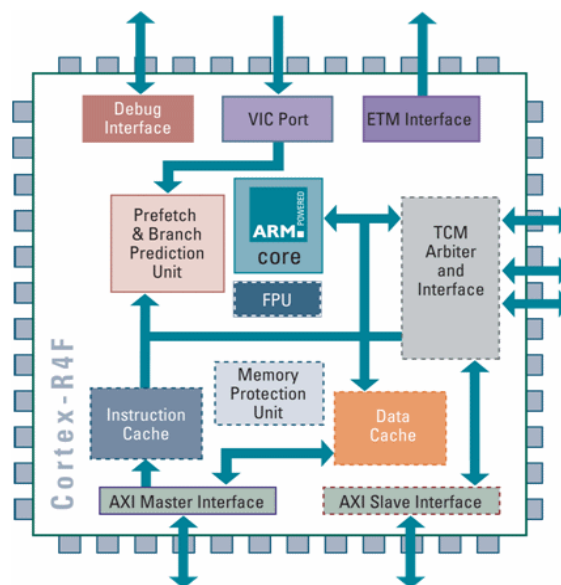

Figure 1: The Cortex-R4 processor core

Based on the ARMv7 instruction set architecture, the Cortex-R4 processor utilizes Thumb-2 technology for enhanced performance and improved code density. Thumb-2 is a blended instruction set that contains all the opcodes from the original 16-bit Thumb instruction set, as well as a large range of 32-bit instructions to provide almost the full functionality of the full 32-bit ARM instruction set. This means that 16- and 32-bit instructions are mixed on an instruction-by-instruction basis with the optimum instruction mix chosen by the compiler. The result is that Thumb-2 code can retain the high performance of ARM code, while achieving the code density benefit of Thumb-1. The ARMv7 architecture also improves exception and interrupt handling and provides better support for non-maskable interrupts (NMI).

The Cortex-R4 processor makes use of several ARM technologies, such as the AMBA 3 AXI protocol for an efficient 64-bit on-chip interconnect. Features in the Cortex-R4 give engineers the flexibility required to optimise the processor for their application:

- A flexible TCM interface with DMA support. The time allowed for RAM accesses is more than doubled compared with ARM9E family processors at the same frequency.

- Support for error detection and correction on RAMs for improved reliability.

- An enhanced Memory Protection Unit (MPU) with 8 or 12 regions providing a fine 32 Byte granularity for stack checking. Regions may also overlap for better memory layout.

- Flexible configuration options for caches, TCM, MPU, error detection/correction, AMBA AXI slave bus and debug which can save up to 70k gates depending on the chosen configuration.

- Optional high-performance IEEE754 FPU, optimised for single-precision but with double-precision capability. This 70k gate unit is an optional component of the Cortex-R4 licence.

- Processor micro-architecture with instruction pre-fetch and queue, branch prediction dual-issue execution to deliver more performance at a given clock frequency.

- A synthesis-time option to generate a redundant copy of the processor logic enabling error detection in safety-critical systems, such as ABS. This specialised feature enables two processors with relevant checking logic to be implemented in a lock-step configuration.

Typical configurations

The Cortex-R4 processor's broad range of applications is addressed by selecting an appropriate configuration prior to logic synthesis. Table 1 summarizes possible configuration options for the processor, with the main parameters listed.

Configuration feature	HDD	Imaging	Wireless	Automotive
TCM	Yes	Probably not	Yes	Yes
Caches	Sometimes	Yes	Sometimes	Sometimes
MPU	No	Sometimes	Yes (8 regions)	Yes (12 regions)
Breakpoints and watch-points	Minimum	Maximum	Maximum	Maximum
ECC/parity	Sometimes	No	No	Yes

Table 1: Configuration options for target applications

Micro-architecture

Pipeline Structure

The Cortex-R4 processor offers superior data processing throughput and excellent power efficiency in real-time embedded applications. Cortex-R4 delivers performance of 1.6 Dhrystone MIPS/MHz and its advanced pipelined architecture enables it to operate at high clock frequencies. This pipeline structure has many advantages over previous ARM processors, targeting performance at reduced cost. The Cortex-R4 is some 30 percent smaller than the ARM1156T2-S and it consumes less power.

The area efficiency comes from a number of micro-architecture optimizations. Compared with the ARM1156T2-S, the Cortex-R4 has one less pipeline stage in its Data Processing Unit (DPU) and a smaller multiplier, as well as a much smaller, though slightly more efficient, Pre-Fetch Unit (PFU). Other contributors to the Cortex-R4's lower gate count come from some optimisation of the Load-Store Unit (LSU) and removal of the Hit Under Miss (HUM) feature. Both the cache controllers and AMBA-3 AXI master are also smaller.
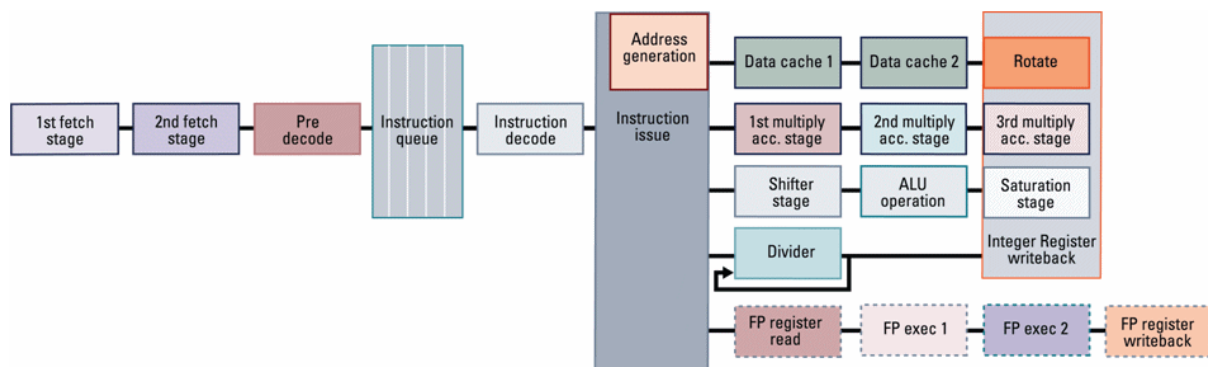

Figure 2: Cortex-R4 pipeline

The pipeline has two cycles for instruction fetch and data access to accommodate the performance of compiled RAMs. Branch and function return prediction takes place at the pre-decode stage. The Instruction Queue (not a pipeline stage) then stores up to eight pre-decoded instructions.

The DPU pipeline is split into four parallel paths plus the floating point path where present. Each path handles different types of instruction and instructions are often processed concurrently:

- 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.

- MAC: Multiply-accumulates are pipelined over three stages. The last stage updates the register bank so there is a single cycle load-use penalty if the next instruction uses the MAC result.

- ALU: Arithmetic and logical operations use an operand pre-shift stage, a basic ALU operation stage, then saturate if required before updating the register bank.

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

The load-store pipeline performs address generation in the issue stage to skew it relative to the other pipelines. This maintains a single cycle load-use penalty for common loads to minimise the delay when data is loaded and required immediately by following instructions.

Pre-Fetch Unit and branch prediction in detail

The Cortex-R4 processor's pipeline has a Pre-Fetch Unit (PFU) incorporating instruction pre-fetching, pre-decoding and program flow prediction. Instructions are fetched from sequential addresses in memory and continue through the pipeline unless there is a branch or a function return when the pipeline must be flushed. An accurate prediction of program flow is required to minimize pipeline flushes and reduce the overall number of cycles per instruction (CPI).

Pre-fetching

The PFU comprises the Fe1, Fe2 and Pd stages of the pipeline. It is responsible for fetching 64-bit wide packets from the instruction memory, extracting instructions from them (including ARM and Thumb-2) and presenting an instruction stream to the DPU. The PFU was designed to be used with high latency level 1 cache memory and TCM, where the typical access time is two cycles rather than one. As such, the PFU incorporates a three-entry 64-bit pre-fetch buffer and as long as there is space in the pre-fetch buffer, the PFU is able to fetch the next instruction packet.

Program Flow Prediction

Two classes of prediction are used: 'Taken-ness' branch prediction and target address prediction.

Taken-ness branch prediction predicts branches where the target address is a fixed offset from the program counter. These branches are handled by the dynamic branch predictor using a global history scheme. A global history prediction scheme is an adaptive predictor that learns the behaviour of branches during program execution. The prediction is checked later in the pipeline once the condition code of the branch has been evaluated. Conditional return instructions will have their taken-ness predicted by the branch predictor. Predicting program-flow changes, such as branches, allows the PFU to start fetching instructions from the branch destination before the DPU has executed the branch, thus saving a number of cycles provided that the prediction was correct.

For branches where the target address is not a fixed offset from the program counter, the target can only be easily predicted if it is a function return. Function return prediction enables instructions to have their target address predicted by the return stack.

The return stack is used to accelerate returns from functions. For each function call the return address is pushed onto a hardware stack. When a function return instruction is recognised by the hardware, an address is popped from the return stack and is used as the predicted return address. The prediction is checked at the end of the pipeline when the correct return address is available. The return stack is four entries deep.

Pre-decoding

Another function of the PFU is to pre-decode the instruction before it is pushed onto the Instruction Queue (IQ). Different classes of instruction are transformed into separate pre-decode formats and pushed onto the IQ. This reduces the complexity of the main decode blocks, which are on the

critical path, and allows data relating to branches, function calls and returns to be extracted for use by the branch predictor and return stack.

On each cycle, up to two instructions can be pre-decoded and moved into the IQ. The IQ is a FIFO that is capable of reading and writing one or two instructions at a time. The IQ decouples the main decode and execution pipeline in the DPU from the instruction fetch and branch prediction in the PFU. The first two instructions from the beginning of the pre-fetch queue are formatted, analyzed and can be moved into the IQ.

Dual issue reduces CPI

Dual issuing, or issuing pairs of instructions in parallel, enables the Cortex-R4 to achieve a measure of superscalar-like performance, thus lowering its CPI without the addition of significant extra resources. Dual issuing improves usage of the register file but is only possible when the next two instructions in the stream use complementary rather than competing data-path resources.

The Cortex-R4 processor's general-purpose register file has four read ports and two write ports, enabling the supply or consumption of 64-bits of data per cycle, to or from the memory system.

For example, a store double instruction (STRD) must read two registers to supply data to the memory system, and a further two registers for the base and offset to provide an address for the access to the memory system. A total of four register file read ports are required to sustain STRD at a throughput of one per cycle when accessing cached or TCM addresses.

Similarly, a number of instructions require two write ports to the register file. Most important are the memory load instructions such as load double (LDRD) and load multiple (LDM) that are capable of reading 64-bits of data per cycle from the memory system. Additionally, any load of a single instruction that also requires the base address register to be updated (e.g. LDR r0, [r13, #4]!) requires two register file write ports if both load and write-back are to occur in the same cycle.

Because the memory system can only return a maximum of 64-bits of data per cycle, there are few cases that require more than two register file write ports. Such operations include loading 64-bits of data at the same time as base register write-back is required. These cases are rare enough that the extra area and speed cost of a third write port is not justified.

Even with just four read and two write ports, there are many instructions that do not utilize all these ports. For example, even a compound data processing instruction such as ADD r5, r6, ASR r0 only requires three read ports and one write port. Any data processing instruction using an immediate shift amount, or no shift at all will require one fewer read ports, and a simple move instruction such as MOV r1, r2 only requires one read port and one write port. Clearly, for a majority of instructions, having four read and two write ports on the register file is overkill. Similarly many instructions only use one of the data-path pipelines in the execute stages of the pipe, leaving the others standing idle.

The dual-issuing of complementary instructions takes advantage of unused register file ports and data-paths to improve the overall utilization of the Cortex-R4 processor resources, providing some superscalar-like benefits at low- cost.

Low Cost Implementation

Keeping the dual-issue control logic simple ensures that the area overhead is minimized. The combinations of instructions that can be dual-issued were carefully chosen through a process of modelling and benchmarking to provide maximum performance improvement (Table 2). Many other pairs of instructions have complementary resource requirements, but adding the capability to dual-issue these adds complexity to the design (and therefore area) while being of little benefit.

Category	First instruction	Second instruction
A	Any instruction except load/store-multiple (LDM/STM), flag-setting multiplies e.g. MULS, various control instructions e.g. MSR and any exception instruction e.g. SVC – was SWI.	B<condition> #immed IT NOP
B1	Single load instructions with immediate or register offset, or register offset with logical-left shift of less than four. No base-register write-back. e.g. LDRGT r10, [r0, r1, LSL #3]	Most data processing instructions, excluding multiplies, that do not require a shift by a value in a register. e.g. RSB r3, r2, r6, LSL #1
B2	Single store instructions with immediate or register offset. No base-register write-back. e.g. STRH r5, [r1, #2]	
C	Any move instruction, immediate or register, that does not require a shift. e.g. MOVW r0, #0x55AA	Most data processing instructions, excluding multiplies. e.g. BFIEQ r10, r11, #4, #5

Table 2: Dual issue instruction pairs

To keep the branching and exception logic simple, the processor only allows dual-issue of instructions in the order shown in Table 2. This means that the decoder for the second instruction is only required to decode a subset of the instruction set. In particular it does not need to be capable of generating control signals for the memory system or decoding load and store instructions. This reduces the area of the decode logic required.

If instructions are in the wrong order to be dual-issued then only the first one is issued. The next pair of instructions in the stream is considered for dual-issue in the next cycle. For example:

```
a) MOV r10, #0x6C
b) LDR r0, [r12, #0x140]
c) ADD r1, r10, #13
d) LDR r2, [r12, #0x144]
e) ADD r1, r1, r3
f) STR r1, [r12, #0x140]
g) B #0x2000
```

The above sequence could, if any order of dual-issuing were allowed, be issued in the sequence: ab, cd, ef, g. Since MOV followed by LDR is not allowed, instruction a) must be single-issued, but this allows subsequent pairs of instructions to dual-issue, in the sequence: a, bc, de, fg. The total number of issue cycles is the same in each case. Because of this effect, the ordering restriction does not significantly impair the ability of the processor to dual-issue.

Because the possible dual-issue cases are strictly defined, the data-path resource allocation for any given instruction can be statically determined from its position in the instruction stream. When the decode stage is presented with a pair of instructions, each instruction is decoded independently in such a way that its allocation of resources complements the allocation of the other. Taking an example from the above table:

- LDRGT r10, [r0, r1, LSL #3] will always be decoded to write the destination register, r10, through write port 0, W0.

- RSB r3, r2, r6, LSL #1 as the second instruction will always be decoded to write the destination register, r3, through write port 1, W1.

In this way, after each instruction is decoded, the resulting control signals can simply be multiplexed together to produce a single control for each data-path resource, rather than requiring complex allocation logic, which might otherwise take more time and require an additional pipeline stage. Most of the data-path logic is agnostic as to whether it is being driven by the first or second instruction of a pair.

Dual-issuing is also limited by data dependencies between the pairs of instructions, a small number of restrictions on the use of r15 (the program counter) and flag setting. Hazard detection logic in the decode stage determines whether or not a pair of instructions can be dual-issued. If the dual-issue conditions are not met, or if only one instruction is available to be decoded, then only one instruction is issued.

Instructions are presented to the decode stage by the instruction queue. When the DPU is busy executing a slow instruction this allows the PFU to queue up instructions for execution, and whenever there are two or more instructions in the IQ the DPU may be able to dual-issue them. Similarly, when the memory system takes a number of cycles to return data to the PFU, the IQ provides a buffer of instructions to keep the execution pipeline fed. The IQ also breaks some of the timing paths from the execution pipeline back to the instruction fetch logic.

The PFU is capable of fetching 64-bits of data from the memory system at a time. This data is held in a pre-fetch queue, and instructions are extracted from it. Each double word of data can contain two or more instructions, and the PFU pre-decode (Pd) stage is capable of extracting two instructions at a time and writing them into the IQ. In this way, the whole pipeline is capable of a sustained throughput of two instructions per cycle, being neither starved by the memory system, nor stalled by the execution pipeline capability.

The dual-issue capabilities, branch prediction mechanism and other features enable the Cortex-R4 processor to achieve its high performance of 1.6 Dhrystone MIPS/MHz.

Hardware divider

The ARMv7R architecture incorporates both signed and unsigned integer divide instructions, and the Cortex-R4 incorporates a hardware divider to execute them. The divider uses a Radix-4 algorithm in which it considers two bits, and therefore four possible values, of the resulting quotient per cycle. For a 32-bit divide, the algorithm takes a maximum of 16 cycles to generate the full result. This maximum can be reduced by counting the number of leading zeros in the dividend and divisor and pre-scaling the data so that the division is only computed for bits that might be non-zero. This early

termination mechanism reduces the number of cycles in many cases, but adds an overhead, since an extra cycle is required to perform the pre-scale.

The division algorithm does not support signed division, so if this is required, any negative numbers must first be negated before the division is performed. At the end of the division, the resulting quotient can finally be negated if the original operands were of different signs.

Even though it employs a radix-4 algorithm and supports early termination, the hardware divider may still take a large number of cycles to operate. To mitigate the effect this has on performance, the divider, in contrast to the rest of the data path, can operate out of order. This allows other instructions (except divides) to continue to execute, provided they are not dependent on the result of the divide, while the divider calculates the result.

When a divide instruction is executed in the Cortex-R4 pipeline, it is split into two parts. A phantom instruction is sent down a control pipeline, in lock-step with the other pipelines. At the same time, the operands for the division are sent to the divide unit, and the calculation is started. The phantom instruction carries with it the condition codes for the divide, and also maintains the ordering of the program. If the phantom fails its condition codes, or is flushed from the pipeline as a result of a preceding mis-predicted branch or an exception, then the divide calculation is terminated. Once the phantom reaches the end of the pipeline, the processor commits to finishing the divide calculation.

At the end of the divide calculation, the divider signals that it has finished, and the pipeline stalls for one cycle in order to allow the result of the divide to be written to the register file. This operation re-uses the logic in the main pipeline to perform the final negation, if required, for a signed divide.

Memory architecture

Selecting the right memory architecture is the key to maximizing system performance and optimizing cost and power. Deciding between the use of SRAM and ROM, the size and partitioning of the cache and configuration of Tightly-Coupled Memory (TCM) are fundamental choices in system design, depending on the application's real-time constraints and the flexibility of the processor.

The Cortex-R4 processor incorporates a Harvard level 1 memory system. This consists of caches, the TCM interfaces and the Memory Protection Unit MPU, each of which is optional and may be omitted in synthesis to save silicon area. This level of configuration provides flexible and efficient support for implementing a memory architecture that is optimized for the application requirements.

The TCMs in Cortex-R4 provide best-in-class, high-performance, deterministic, hard real-time processing for cost-optimized deeply embedded systems. TCMs are used for storing instructions, program literals or data that must be immediately available for processing and not stored away in main memory and at risk of not being ready in cache. TCMs are often used for storing a fast Interrupt Service Routine (ISR) or they can be used to stream data in or out of the processor using a dedicated AXI bus port. In the Cortex-R (Real-time profile) processors, deterministic behaviour is not compromised by inclusion of a Memory Management Unit (MMU) for virtual memory addressing, as used by multi-tasking software operating systems. An MMU introduces additional and less predictable delay between the processor and its memory system and is not consistent with TCM architecture.

By pipelining memory access over two cycles in both PFU and DPU, the Cortex-R4 relaxes the required response time of the RAM compared with the processor's clock frequency. This flexibility to specify slower on-chip memory leads to a significant reduction in memory area, power consumption and cost. Physical implementation is also simplified by Cortex-R4's less stringent timing demands as timing closure is easier to achieve.
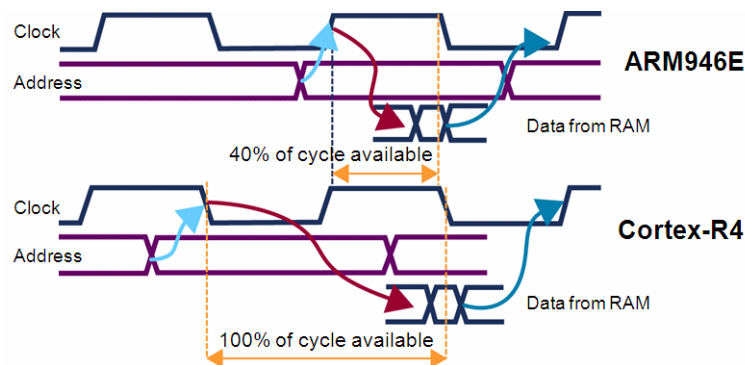
**Figure 3: Cortex-R4 memory access timing compared with ARM9**

Tightly-Coupled Memory support

The Cortex-R4 provides flexible support for Tightly-Coupled Memories (TCM). TCMs are used in real-time embedded systems where instructions and data are held close to the processor for guaranteed access time. Each TCM can be up to 8 MB in size but they are normally limited by memory performance to 256 KB or less as larger SRAM size makes for slower access and increased cost. Three 64-bit TCM ports support memory configurations ranging from a single unified memory for both instructions and data through to separate memories capable of simultaneous DMA access. Each port has independent wait and error signals for connecting RAM, ROM, e-DRAM, and error correction logic.

TCM ports are grouped in two interfaces, A and B. Interface B has two ports for greater bandwidth using interleaved access. This enables TCMs to be organized as either a single bank of memory on interface A for the lowest system cost (which is just 5 percent slower than the fastest configuration), or as two interleaved banks on ports B0 and B1 for a medium cost system (that is just 2 percent slower than the fastest configuration), or as three banks using all the ports for best performance.

The benefit of supporting interleaved TCM-B is apparent when DMA into the TCM (via the slave port) occurs at the same time as data access from the LSU to the TCM. Interleaving makes the most difference when sequential data is being accessed. If both DMA and the LSU try to access the same RAM the slave will stall once on the initial clash, and then the two will continue accessing complementary RAMs. Interleaved memory support is particularly useful if the processor is working on frames of data. The processor can work on the current frame while the next frame is written into the TCM.

The TCM interfaces differ from the main AMBA memory interface in that they are optimized for low latency memory access. The Cortex-R4 issues speculative read accesses on these interfaces and as such they are not suitable for read-sensitive memory or peripherals. Memory access attributes are not exported on this interface. The TCM interfaces have wait and error signals to support error detection, correction and slow memories.
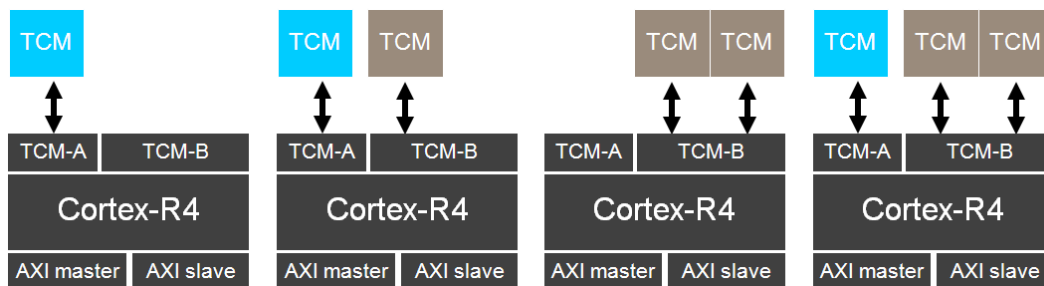
Figure 4: TCM configurations from low cost to maximum performance

The TCM-A and TCM-B interfaces may be connected to zero wait state memory. Each interface is also waitable, and has an error input allowing complex external error identification and correction. The processor also supports parity generation and parity error detection and, as with the ECC support, is configurable on a per port basis.

Unlike previous ARM processors, there is no restriction on the data type that can be stored in the TCMs, allowing symmetrical access from the processor's I- and D-sides. On earlier processors, access to the ITCM and DTCM from the opposing side would result in larger latencies, and instructions fetches could not be made from the DTCM. With Cortex-R4, the only restriction is that any LSU access takes priority over the PFU.

Other ARM processors, such as ARM9E series, have used a fixed-size memory map for the TCM regions, which means that a smaller TCM would be aliased over the region. With Cortex-R4, the TCM size is pin configurable which means that the TCM regions can exactly match each TCM size.

Caches

The Cortex-R4 supports separate Instruction and Data caches. Each cache is a physically addressed 4-way set associative cache with a line length of 8 words. Cache sizes can be independently varied from 4KB to 64KB. Both I and D caches are capable of providing 64-bits per cycle to the processor. The caches may be disabled independently from the TCM.

A cache miss results in the requests required to do the line fill being made to the level 2 (L2) memory interface, and a write-back occurring if the line to be replaced contains dirty data. The cache supports allocate on read miss and allocate on write miss. The write-back data is transferred to the write buffer, which is arranged to handle this data as a sequential burst.

The caches perform critical word first cache refilling. The internal bandwidth from the L2 data read port to the data caches is 8 bytes/cycle, and supports streaming.

The processor can also abandon a line-fill request in order to accelerate interrupt entry. The line-fill will still complete on the bus (and be written into the cache) because an AXI burst will have been committed to, but the processor can continue fetching data and instructions from the cache or TCM.

AMBA 3 AXI memory interface

The Cortex-R4 has a 64-bit AMBA 3 AXI advanced extensible interface, which supports high-bandwidth transfer to second level caches, on-chip RAM, peripherals and interfaces to external memory.

AMBA 3 AXI supports the issuing of multiple outstanding addresses and for data to be returned out of order. The most significant advantage for many applications will be that a slow slave does not block the bus for the duration of its access, allowing the processor to perform further accesses rather than waiting for completion of the slow access.

In addition, there is an optional AMBA AXI slave port allowing masters on the AMBA AXI system access the processor's level 1 memory. This interface supports full DMA access to the TCMs.

Memory Protection Unit

The optional Memory Protection Unit (MPU) enables memory to be partitioned into 8 or 12 regions with individual protection attributes set for each region. The MPU is programmed from privileged mode and the minimum size of an MPU region is 32-bytes, allowing fine control and minimising memory wastage. If the MPU is omitted there is a fixed mapping of protection attributes.

An MPU is vital for certain applications, such as safety-critical automotive devices. It is also a useful tool for software development to trap out-of-range memory accesses.

Built-in self test

All RAM blocks can be tested by Built-In Self test (BIST) through a Cortex-R4 BIST interface. This enables flexibility in selecting the BIST methodology, allowing appropriate algorithms to be selected for different RAM topologies. Dedicated BIST ports provide an optimised path to the controls of the RAMs, which avoids introducing frequency-limiting critical paths into the design.

Error management

Modern semiconductor processes are susceptible to soft errors caused by radiation or other disturbances and a minimum level of error detection through parity protection is a requirement for many safety-critical systems. The Cortex-R4 includes extensive support for parity detection and it also supports Error Correction Codes (ECC) that can detect and then automatically correct errors.

Memory ECC and parity checking

The Cortex-R4 has options for either parity protection or ECC on its level 1 memory, i.e. caches and TCMs. The ECC is integrated into the processor's pipeline such that its operation is fast and transparent with the processor waiting while the data is corrected and re-read. Other logic prevents the ECC from blocking the system in case of a hard error. Designers can also implement their own ECC scheme using error and wait signals.
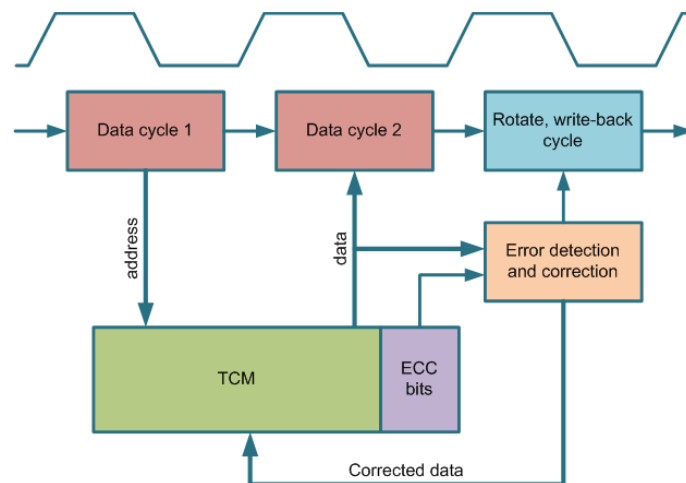
Figure 5: ECC operation in the pipeline

Integrating ECC into the pipeline avoids the circa 20 percent performance penalty that is usually associated with an external implementation of protection to this level. Integrated ECC also simplifies certification processes for safety and reliability standards in automotive and other markets.

The read-modify-write operation necessary for a low cost ECC implementation incurs a small power consumption penalty and so there may be benefits in using only parity protection if it is adequate for an application. This is often the case when there is an alternative method for correcting errors such as refreshing the instruction cache from external memory.

Parity generation and checking in Cortex-R4 is facilitated through byte-wise parity inputs and outputs. Parity logic is only generated when this configuration option is selected. Cache RAMs include a synthesis option to enable parity support when the RAMs become wider to incorporate the parity bits. One parity bit is stored per byte in the data RAMs, while one parity bit is stored per set/way combination in the tag and dirty RAMs.

Odd or even parity is selected by pin configuration. The processor generates parity for each byte of data and the address tag, and checks the parity bits returned. With the parity bits present in each RAM, it is possible to detect and correct a parity error at the expense of a line-fill.

When the processor detects a parity error, there are two options available. These can be selected and controlled in software. If both options are turned off, parity is disabled.

* Hardware correction: The processor can be configured to automatically invalidate the faulting cache line (forcing the cache to be Write Through). The corresponding instruction is re-executed. Second time around the instruction misses in the cache and the correct data is fetched from the external memory system.

* Software notification: A precise abort can be generated on the instruction or data access to allow software to fix the error. This allows error profiling and process termination in fatal circumstances. The Data Fault Status Register (DFSR) allows the abort type to be determined – AFSR indicates a set and way fault. The Data Fault Address Register (DFAR) indicates the address that generated the abort. Provided the cache line is not dirty, the data abort handler can invalidate it and rely on external memory to provide an error-free copy of the data when next required by the processor. If the cache line is dirty, then the parity error may be unrecoverable.

Hardware correction can be enabled at the same time as software notification. In this case, the cache line is automatically invalidated, but a data abort is also taken which is useful for monitoring error rates. Parity operation can be configured separately for the instruction and data caches.

Logic errors

The Cortex-R4 supports a synthesis option to instantiate a redundant copy of the processor in 'lock-step' configuration for use in safety-critical systems. External checking logic then compares both processors' outputs to detect soft or hard errors. The interface helps to keep the overall system area down by allowing the redundant processor to share the RAMs of the master processor.

Interrupt handling

A fast interrupt response is critical to most deeply embedded systems and the Cortex-R4 minimises interrupt latency as far as possible to provide deterministic system behaviour.

The Cortex-R4 has an interrupt controller interface for accepting the vector address of an Interrupt Service Routine (ISR). This enables a dedicated external hardware unit, such as ARM's Vector Interrupt Controller (VIC), to look up the start address of the ISR for the highest priority outstanding interrupt. This interface has a handshake from the Cortex-R4 to acknowledge when a particular interrupt has been taken and it supports controllers running asynchronously to the main Cortex-R4 clock, so systems can be either synchronous or asynchronous with the processor clock and AXI clock.

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 line-fill to complete. This is enabled by the use of a different AMBA AXI IDs 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 cannot be abandoned) is used carefully, the maximum interrupt latency should be independent of the access times of AMBA AXI memory and peripherals.

The Cortex-R4 can abandon and restart a multi-cycle instruction such as load or store multiple if an interrupt request is made part way through execution. This avoids the interrupt latency associated with completing as many as 16 data reads. The ISR can then be fetched from instruction cache or TCM while the data cache line-fill completes, as the core is no longer waiting on the returned data.

These advanced functions enable Cortex-R4 to respond to an interrupt in as few as 20 cycles using the dedicated fast FIQ input, compared with 54 cycles for the ARM966E-S or 118 cycles for the ARM946E-S (assuming the ISR is immediately available in the R4's TCM or cache). Worst case response increases to 30 cycles when using the VIC on IRQ. These interrupt responses are maintained when a higher priority interrupt occurs whilst Cortex-R4 is already executing an ISR. The cycle count responses quoted are to the first instruction in the ISR after which the ISR entry code takes a further 11 cycles before interrupts are re-enabled. Cortex-R4 allows the ISR to continue in Thumb-2 mode, whereas ARM9E processors must exit Thumb mode to handle any exceptions.

If preferred, the ARM Generic Interrupt Controller (GIC) can be used with Cortex-R4. A non-maskable interrupt option is also available on the Cortex-R4, which prevents disabling of the Fast Interrupt reQuests (FIQ) by software. This is particularly useful in safety-critical applications.

Floating Point Unit

The capability to perform floating-point calculations is a requirement for applications such as automotive electronics that use sophisticated control algorithms. Cortex-R4 supports this with a fully integrated, IEEE754 compliant, Floating Point Unit (FPU) that is an integral part of the processor's DPU, and not an external co-processor. The FPU is backward-compatible with earlier ARM processor FPUs and it offers excellent performance as the Cortex-R4 can dual issue integer and floating point instructions. The load-use penalty for the floating point pipeline is reduced by its one-cycle offset from the integer pipeline. The FPU is a licence (Cortex-R4F) and synthesis option.

Tools such as MATLAB are commonly used to develop models of an entire engine control and power-train subsystem. Sophisticated closed-loop models can be developed that include components such as the engine as well as the control systems, enabling simulation of a complete system. The resulting code will normally contain floating-point data types.

Other high-level programming languages and tools such as C++, UML and Simulink are used to develop re-usable platforms where an application is developed once and retargeted multiple times. Floating-point algorithms need to be mapped directly from the modelling environment such as Simulink or ASCET-SD, and the code auto-generated using tools such as MathWorks' Real Time Workshop Embedded Coder, ASCET-SE or dSPACE Targetlink. Using a code-generation and modelling-based methodology improves quality and reduces development time. A processor with floating-point capability ensures the target platform fits with the methodology.

The Cortex-R4's FPU performs floating-point calculations that allow a greater dynamic range and accuracy than fixed-point calculations. The FPU is optimised for single precision automotive and control systems with performance as near as possible to integer operations. All rounding modes are supported in hardware and it also has hardware divide and square root capability. When required, the FPU can perform double-precision (64-bit) calculations at the expense of some calculation speed.

Development environment

Cortex-R4 developers are supported with a wide range of ARM tools and infrastructure including:

- Fast models. High speed behavioural simulations running on Windows or Linux and capable of simulating memory system delays. These models are developed and co-validated with processor hardware and may be used for application software programming prior to silicon availability.

- CoreTile. A PCB with Cortex-R4 SoC running at 250 MHz with 64 kB level 1 caches, three 64 kB TCMs and 512 MB of SDRAM. These boards are hosted on the RealView Evaluation Baseboard and can be used for prototyping hardware and performance evaluation.

- Design Simulation Models. Timing accurate models for use in Verilog or VHDL simulations during hardware design. DSMs are derived directly from the Register Transfer Level (RTL) code.

- RealView™ Eclipse-based software development environment incorporating the optimizing ARM compiler, documentation, build tools, profiler and debugger.

- CoreLink™ system and fabric IP including AMBA bus, interrupt and memory controllers etc.

- CoreSight™ on-chip Debug Access Port (DAP) and Embedded Trace Module (ETM) for low-overhead, high-visibility monitoring of Cortex-R4 SoC products.

A top quality debug environment is a key success factor for engineering teams bringing up a new product. It is then used to monitor correct operation before releasing systems to the field, which, for example, is important for maintaining safety standards in complex automotive systems.
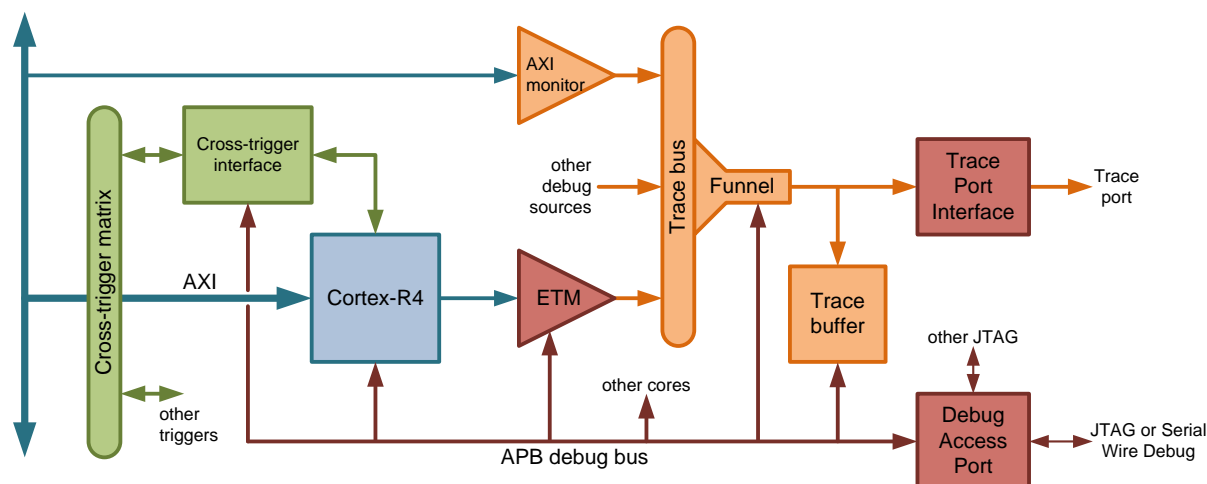

Figure 6: CoreSight debug and trace

CoreSight technology, shown above for Cortex-R4, provides mechanisms to ensure visibility into all parts of the system. The ETM provides both instruction and data traces, time stamped and in order. It has a highly flexible and complex triggering capability including loop counting and sequences.

ARM's common architecture helps to reduce the cost of ownership for development systems. Many ARM and third-party tools can be re-used across the entire Cortex processor family.

Implementation

The Cortex-R4 processor is highly configurable in synthesis as shown in the table below. For single core configurations the processor's gate count ranges from approximately 150k gates to 290k gates. These gate counts assume that synthesis is not targeting the absolute maximum clock frequency.

Tightly-Coupled memories	0 to 3, each 0 to 8 MB	4.5k gates
Level 1 cache controllers	I and D, each 4 to 64 kB	20k
Memory Protection Unit	8 or 12 regions, min 32 byte	12k
ECC, Parity error correction	Apply to cache and/or TCM	10k
AMBA3 AXI slave interface	Non-blocking, 64-bit, DMA	11k
Breakpoints and watchpoints	Chose up to 8 of each	13k
IEEE754 Floating Point Unit	SP optimised, DP capable	70k
Safety-critical system	Dual core, lock-step option	-

Table 3: Cortex-R4 configuration options

Designs that target the maximum achievable frequency generally expand the core's gate count, area and power consumption and results vary widely, not only with configuration options but also by process library and technology. Some sample data points are given here:

|	90 nm G	65 nm GP		65 nm LP	40 nm G
Advantage Library	9T	10T	HS	LP 10T	HS 12T
Target frequency	304 MHz	619 MHz	381 MHz *	392 MHz	934 MHz
Standard cell area	0.8 mm2	0.8 mm2	0.4 mm2	0.7 mm2	0.48 mm2
Core power	0.18 mW/MHz	0.12 mW/MHz	0.09 mW/MHz	0.17 mW/MHz	0.08 mW/MHz

Table 4: Cortex-R4 performance, power and area, excluding FPU

The above configurations used eight MPU regions, two breakpoints, one watch-point and controllers for 8k I and D caches. Configuring TCM ports, an AXI slave and parity/ECC on I and D caches will introduce small variations in these results. The FPU increases standard cell area by 0.2 mm^2 on 65 nm LP. Results target maximum frequency except * that targets smallest area and low power.

Comparison with ARM9 and ARM11

The micro-architecture enhancements in Cortex-R4, such as dual issuing, branch prediction and a pipeline that allows single-cycle load-use penalty, deliver more work per cycle. Improved efficiency reduces the CPI, enabling the processor to clock more slowly while delivering 1.6 DMIPS/MHz. Enabling a lower frequency clock reduces dynamic power consumption and also allows lower supply voltages for reduced leakage power and improved reliability.

Cortex-R4 offers more performance than the ARM946E-S and is more than twice as efficient when running Thumb code. Configuration options enable Cortex-R4 to address applications that would have previously used ARM946E-S or ARM1156T2-S. Dhrystone MIPS benchmarks combined with implementation data on 65 nm LP show that Cortex-R4 offers almost twice the performance of ARM946E-S, matching the ARM1156T2-S but with almost three-times the power efficiency!
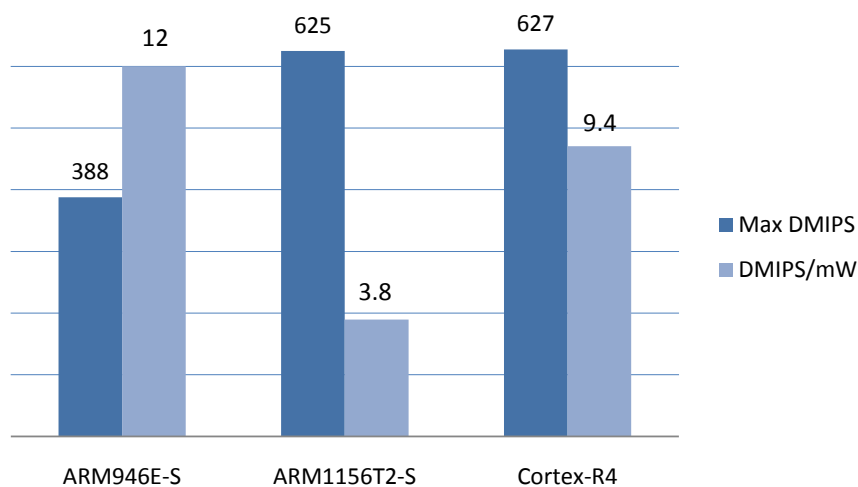
Figure 7: Performance and power comparison

Cortex product family

The Cortex Family of processors provides a range of solutions optimized around specific market applications across the full performance spectrum. This underlines ARM's strategy of aligning technology around specific market applications and performance requirements.

The ARM Cortex family comprises three processor series, which all implement the Thumb-2 instruction set to address the increasing performance and cost demands across a range of markets:

- ARM Cortex-A Series, applications processors for multi-tasking OS and user applications. ARM and Thumb-2 instruction sets.

- ARM Cortex-R Series, embedded processors for high-performance, real-time systems. ARM and Thumb-2 instruction sets.

- ARM Cortex-M Series, embedded processors optimized for cost sensitive applications and microcontrollers. Thumb-2 instruction set only.

Summary

The Cortex-R4 processor delivers high performance combined with cost and power efficiencies across a broad range of deeply-embedded applications. Significant micro-architecture features include Thumb-2, a pipeline design that implements instruction pre-fetch and branch prediction, and low-cost superscalar-like processing through selective dual issuing. This micro-architecture achieves a high level of performance at moderate clock frequencies. This approach is fundamental to delivering a low power and reduced cost implementation.

Key to the Cortex-R4's features are the three TCM interfaces that support a flexible variety of configurations. Code and data stored in TCMs are available to guarantee a fast and deterministic response from the processor.

In addition, the TCM and DMA implementation, as well as the provision of a single AMBA 3 AXI interface, ensures that the Cortex-R4 processor is easier to integrate than other cores. Relaxed level 1 memory timing ensures significant system area and power savings can be achieved, and make it easier to meet synthesis timing constraints during the development process.

Further features enhance the Cortex-R4 processor's deeply embedded credentials for specific applications. The more flexible MPU allows greater levels of RTOS protection. The processor responds quickly to interrupts with a fast and deterministic exception entry time, which is greatly improved over earlier processors such as ARM946E-S. The optional FPU provides for fast single precision or double precision calculations. High-reliability applications are supported through the ECC and lock-step configuration options.

Acknowledgments