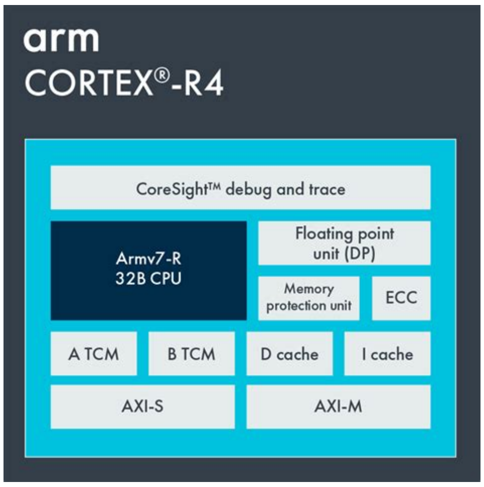
Cortex R4 Reference Manual



File Name: Cortex R4 Reference Manual.pdf
Size: 2790 KB
Type: PDF, ePub, eBook
Category: Book
Uploaded: 19 May 2019, 15:24 PM
Rating: 4.6/5 from 735 votes.

Status: AVAILABLE

Last checked: 4 Minutes ago!

In order to read or download Cortex R4 Reference Manual ebook, you need to create a FREE account.

Download Now!

eBook includes PDF, ePub and Kindle version

Register a free 1 month Trial Account.
Download as many books as you like (Personal use)
Cancel the membership at any time if not satisfied.
Join Over 80000 Happy Readers

Book Descriptions:

We have made it easy for you to find a PDF Ebooks without any digging. And by having access to our ebooks online or by storing it on your computer, you have convenient answers with Cortex R4 Reference Manual . To get started finding Cortex R4 Reference Manual , you are right to find our website which has a comprehensive collection of manuals listed.
Our library is the biggest of these that have literally hundreds of thousands of different products represented.

Book Descriptions:

Cortex R4 Reference Manual

The cores are optimized for hard realtime and safetycritical applications. Cores in this family implement the ARM Realtime R profile, which is one of three architecture profiles, the other two being the Application A profile implemented by the CortexA family and the Microcontroller M profile implemented by the CortexM family. The ARM CortexR family of microprocessors currently consists of ARM CortexR4F, ARM CortexR5F, ARM CortexR7F, ARM CortexR8F, ARM CortexR52F, and ARM CortexR82F.It is similar to the A profile for applications processing but adds features which make it more fault tolerant and suitable for use in hard realtime and safety critical applications.ARM offers a variety of licensing terms, varying in cost and deliverables. To all licensees, ARM provides an integratable hardware description of the ARM core, as well as complete software development toolset and the right to sell manufactured silicon containing the ARM CPU.In this form, they have the ability to perform architectural level optimizations and extensions. This allows the manufacturer to achieve custom design goals, such as higher clock speed, very low power consumption, instruction set extensions, optimizations for size, debug support, etc. To determine which components have been included in a particular ARM CPU chip, consult the manufacturer datasheet and related documentation.The system not only needs to be fast and responsive to a plethora of sensor data input, but is also responsible for human safety. A failure of such a system could lead to severe injury or loss of life.By using this site, you agree to the Terms of Use and Privacy Policy. For some products, newer alternatives may be available. The CortexR4 processor implementation uses the ARM DSP SIMD Single Instruction Multiple Data instruction set and floatingpoint hardware to enable fully the capabilities of the CortexR4 processor for signal processing algorithms.http://www.confortex-distribution.com/userfiles/horton-brotherhood-manual.xml

- **cortex r4 reference manual, cortex r4 technical reference manual, cortex-r4f technical reference manual, arm cortex r4 technical reference manual, 1.0, cortex r4 reference manual, cortex r4 technical reference manual, cortex-r4f technical reference manual, arm cortex r4 technical reference manual.**

The CMSISDSP library, written entirely in C and delivered with source code, enables software programmers to adapt algorithms for specific application requirements and can deliver higher performance for digital signal processing and control applications than can be achieved typically by compilers and run time support libraries. Features An example object code library is included as well. See terms of use. Visit our export site or find a local distributor. Activity Translate Error You dont have JavaScript enabled. This tool uses JavaScript and much of it will not work correctly without it enabled. Please turn JavaScript back on and reload this page. Last modified by Ankur Tomar on May 30, 2013 137 PM. Connect with your peers and get expert answers to your questions. All Rights Reserved. All rights reserved. ARM DDI 0363C CortexR4 and CortexR4FTechnical Reference Manual Copyright 20062008 ARM Limited. All rights reserved.Release Information The following changes have been made to this book.Change History Date 15 May 2006 22 October 2007 16 June 2008 Issue A B C Confidentiality Confidential NonConfidential NonConfidential Restricted Access Change First release for r0p1 First release for r1p2 First release for r1p3 Proprietary Notice Words and logos marked with or are registered trademarks or trademarks of ARM Limited in the EU and other countries, except as otherwise stated below in this proprietary notice. Other brands and names mentioned herein may be the trademarks of their respective owners. Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM in good faith.http://jagtapnursery.com/jagtap-nursery/upload/how-to-manually-start-a-service-in-xp.xml

However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded. This document is intended only to assist the reader in the use of the product. ARM shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product. The IEEE disclaims any responsibility or liability resulting from the placement and use in the described manner. Where the term ARM is used it means ARM or any of its subsidiaries as appropriate. Confidentiality Status This document is NonConfidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to. Restricted Access is an ARM internal classification.All rights reserved. NonConfidential ARM DDI 0363CRestricted Access Product Status The information in this document is final, that is for a developed product. Web Address ARM DDI 0363CRestricted Access Copyright 20062008 ARM Limited. NonConfidential iii iv Copyright 20062008 ARM Limited. NonConfidential ARM DDI 0363CRestricted Access Contents CortexR4 and CortexR4F Technical Reference Manual PrefaceAbout this book.All rights reserved. Their low power, small size and high performance, combined with industry standard architecture make them ideal for these applications. The CortexR4 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 CortexR4 is able to address applications previously covered by the ARM946ES, ARM966ES and ARM968ES processors, and expand this applicability into more demanding situations.

This white paper will examine the key differences between the ARM9E processor family and the CortexR4 processor. Performance and efficiency Pipeline The pipeline length of the CortexR4 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 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 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 operations are split over three pipeline stages, the final one of which also updates the register bank. ALU Arithmetic operations use an operand preshift stage, a basic ALU operation stage, then optionally saturate before updating the register bank. Divider The divider uses a Radix4 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 loadstore pipeline is skewed relative to the other pipelines, by performing the address generation in the issue stage.

http://dev.pb-adcon.de/node/16201

This keeps the loaduse penalty to one for common loads; this is the same as on the ARM9E processor family. Loaduse penalty refers to the delay caused when data loaded is required immediately by following instructions. Figure 1 Comparison of ARM9E and CortexR4 pipelines Dual Issue The CortexR4 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 loadstore 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. 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 CortexR4 processor pipeline has the potential to increase the silicon area, and therefore cost. For the latest details on the area of various processors see. 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.

http://elmariachimexican.com/images/brand-dispenser-manual.pdf

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 CortexR4 has a 64bit AMBA 3 AXI memory interface, compared to the 32bit AMBA AHB interface used on the ARM946ES 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 64bit 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 CortexR4 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 ARM946ES 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.

http://elreehavia.com/images/brand-identity-manuals.pdf

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 21 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 realtime 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 CortexR4 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 prefetch 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 noncacheable 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 nonmaskable interrupt option is also available on the CortexR4 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 CortexR4 processor includes a number of features to reduce this cost. Thumb2 The CortexR4 processor implements the ARMv7R architecture, including the Thumb2 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 32bits. 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 16bits. 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 familybased 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. Thumb2 contains all the 16bit 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 reassembly. However, it supplements these with a large range of 32bit instructions to provide the full functionality of the ARM instruction set. This means that 16 and 32bit 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 Thumb2 code can retain the high performance of ARM code, while giving the code density benefit of Thumb. Compared to an ARM9E processor familybased system running ARM code, this allows for a reduction in the amount of program memory required. When compared to an ARM9E processor familybased system running Thumb code, this allows a reduction in the required operating frequency for a given performance point. RAM requirements The ARM9E processor family and the CortexR4 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 ARM946ES processor, this RAM requires a response time of approximately 40% of the core clock cycle time. For example, an ARM946ES processor running at 200MHz a cycle time of 5ns requires a RAM response time of 2ns. The CortexR4 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 ARM946ES 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 ARM946ES processor, there are a few configuration options that can be altered, such as the cache size and TCM size. The CortexR4 processor extends these configuration options to allow the processor to be more closely aligned with the application's requirements. This configurability also allows the CortexR4 processor to address a wider range of applications. TCM flexibility Both the ARM946ES processor and the CortexR4 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 ARM946ES 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 CortexR4 processor is far more flexible. There are three memory ports, appearing to the programmer as two separate memory regions. 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 ARM946ES 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 64bits wide on a CortexR4 processor, compared to 32bits wide on the ARM946ES 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 CortexR4 processor also introduces a DMA port that was not present on the ARM946ES processor or the ARM966ES 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 ARM946ES 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 ARM946ES processorbased system alternate members of the ARM9E processor family are available without an MPU, though these do not have caches. The MPU on the CortexR4 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 CortexR4 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 CortexR4 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 ARM946ES processor. This means that code compiled for the ARM946ES processor will run on the CortexR4 processor without recompilation. Recompilation will allow the code to be optimised for the CortexR4 processor and targeted at Thumb2. Summary The CortexR4 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 CortexR4 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.Ask us a question or get help. All ri ghts reserved. Release Information The following changes have been made to this book. Other brands and names mentio ned herein may be the trademarks of their respective owners. Neither the whole nor any part of the information contained i n, or the prod uct described in, this docum ent may be adapted or reproduced in any material form except w ith the prior w ritten permission of the copy right holder. The product described in this document is subject to contin uous developments and impr ovem ents. All particulars of the product and its use contained in this documen t are given by ARM in good faith. However, all warranties implied or expressed, including but not limited to implied warra nties of merchantab ility, or fitness for purpose, are excluded. This document is intended only to assist the reader in the use of the produc t. ARM shall not be liable for any loss or damage arising from the us e of any information in th is docume nt, or any error o r omission in such informatio n, or any incorrect use of the product. Where the term ARM is used it means "ARM or any of its subsidia r ies as appropriate". The IEEE disclaims any responsib ility or liability resulting from the placement and use in the described manner. Confidentiality Status This document is NonConfid ential. The right to use, co py and disclose this document may be subject to license restrictions in accordance with the te rms of the agreement entere d into by ARM and the party that A RM delivered this document to. Product Status The information in this do cument is final, that is for a developed product. Web A ddr es s.arm.

com Change history Date Issue Confidentiality Change 15 May 2006 A Confidential First release for r0p1 22 October 2007 B NonConfidential First release for r1p2 16 June 2008 C NonConfidential Restricted Access First release for r1p3 11 September 2009 D NonConfidential Second release for r1p3 20 November 2009 E NonConfidential Documentation update for r1p3 12 February 2010 F NonConfidential Documentati on update for r1p3 04 April 2011 G NonConfidential First release for r1p4 Product revision st atus The r n p n identifier indicat es the revision status of the product d escribed in this book, w here r n Identifies the major revision of the product.Intended audience This book is writt en for system designers, system int egrators, and programmers who are designing or programming a SystemonChip SoC that uses the processor. Using this book This book is organized into the following chapt ers Chapter 1 Introduction Read this for an introduction to the processor and descriptions of the major functional blocks. Chapter 2 Functional Descrip tion Read this for a description of the functionali ty of the processor. Chapter 3 Programmers Model Read this for a description of the processo r registers and programming informatio n. Chapter 4 System Control

Read this for a description of the sy stem control coprocessor registers and programming informat ion. Chapter 5 Prefetch Unit Read this for a desc ription of the functi ons of the Pr efetch Unit PFU, including dynamic branch prediction and th e return stack. Chapter 6 Events and Perfor mance Monitor Read this for a description of the Performan ce Monitoring Unit PMU and the event bus. Chapter 7 Memory Protection Unit Read this for a descriptio n of the Memory Pr otection Unit MPU and the access permissions process. Chapter 10 Power Control Read this for a description of the power control facilities. Chapter 1 1 FPU Programmers Model Read this for a description of the Floating Point Unit FPU support in the CortexR4F processor.

Chapter 12 Debug Read this for a description of the debug suppo rt. Chapter 13 Integration T est Registers Read this for a description of the Integr ation T est Registers, and of integration testing of the processor with an ETMR4 trace macrocell. Appendix A Signal Descrip tions Read this for a description of the inputs and outputs of the processo r. Appendix B AC Characteristics Read this for a description of the timing param eters applicable to the processor. Appendix C Cycle T imings and Interlock Behavior Read this for a description of the instruction cycle timing and in struction interlocks. Appendix D ECC Schemes Read this for a descriptio n of how to select the Error Checking and Corr ection ECC scheme depending on the T ightlyCoupled Memory TCM configu ration. Appendix E Revisions Read this for a description of the technical changes between released issues of this book. T ypographical The typographical conventions are ital ic Introduces special terminology, deno tes crossreferences, and citations.Denotes signal names. Also used for terms in de scriptive lists, where appropriate. Y ou can enter the underlined text instead of the full com mand or option name.Enclose replaceable terms for assembler syntax where they appear in code or code fragment s. For example MRC p15, 0,,, Timing diagrams The figure named Key to timing diagram conventions explains th e components used in timing diagrams. V ariations, when they o ccur, have clear labels. Y ou must not assume any timing information that is not expli cit in the diagrams. Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is un important and does not af fect normal operation. Key to timing diagram conventi ons T iming diagrams sometimes sh ow singlebit signals as HIGH and LOW at the same time and they look similar to the bus change shown in Key to timing diagram convent ions.

http://www.familyreunionapp.com/family/events/corti-incubator-manual