Cortex-R4　PK　ARM9E 家族

介绍

在以硬盘驱动器、成像、汽车、网络以及无线设备为代表的，一个宽广的应用领域范围中，ARM9E都已经发挥热量很久了。它们以低功耗、小尺寸、高性能，再配上工业标准的架构，使得它们与生俱来就拥有了胜任这些应用的素质。

在此基础上，凭借在技术和设计理念上的进化，Cortex-R4的计算效率激增，在保持低成本的同时进一步提升了性能。CR4的能力不仅覆盖了所有ARM946E-S, ARM966E-S以及ARM968E-S的应用领域，还进一步扩展了领土，也使得在相同的应用中使产品向更高档次迈进。在这篇文章里，我们将特写CR4与ARM9E的不同之处。

性能与效率

流水线

到了CR4中后，流水线的级数从ARM9E的5级增至8级。这降低了每一级需要的逻辑规模，从而在给定的进程和库上可以获得更高的操作频率。CR4的明星小档案如下所列：

90nm工艺		
	速度优化版	面积优化版
标准Cells	Advantage-HS	Metro
存储器	Advantage	Metro
频率①（MHz）	475	210
带cache时的面积（mm^2）	1.74	1.00
无cache时的面积（mm^2）	1.30	0.73
cache容量	8KB / 8KB[译注]	8KB / 8KB
带cache时的功耗（mW/MHz）②	0.32	0.21
无cache时的功耗（mW/MHz）②	0.26	0.16
FPU占用的面积（mm^2）③	0.51	0.28

[译注]：cache被分为指令cache和数据cache，因此有两个值。

① 这是最坏情况下的指标：90nm工艺，0.9V，120摄氏度，慢硅(slow silicon)

② 这是典型情况下的指标：90nm工艺，25度，普通硅

③ FPU是选配件

增加级数是流水线改造的表面可见现象；另一个内部改造是：比较靠后的级被分为4条独立并行的流水线，有点像一条马路分成4条岔路一般。这4条独立的流水线也分别担负各自的运算，并且有些时候能并发执行两种不同的运算，如下所述：

1. 加载与存储。该流水线负责所有的存储器访问。存储器访问过程又分为两个流水线级，使得即使花在访问RAM上的时间比较久，也不会使带宽被拖慢。
2. MAC。乘法操作在3个流水线级中完成，最后一级还要负责更新寄存器bank。
3. ALU。算术逻辑操作使用的流水线级分别是：操作数预移位级，基本ALU操作级，以及在更新寄存器之前的一个可选的饱和护理级。
4. 除法。CR4的片上除法器使用Radix-4算法，使得32位除法通常需要6周期，并且在单一的

流水线级中完成。

可见，CR4的流水线与ARM9E的还是大相径庭的。ARM9E的5级流水线一次只能处理一条指令。而在CR4中除了除法外，不同指令在各独立的流水线都能同时地行进。有了这项能力，欲使得指令按优化过的顺序执行(instruction execution in order)，就不再需要大量的电路逻辑来计算“乱序执行方案”。

译注：这里的”in order”不是指按照程序字面上给出的顺序来执行指令，而是找出另一种指令的执行顺序，该顺序不一定与程序字面上的顺序相同，但是能得到相同的计算结果，并且能提高并行度，或者避免流水线停工（stall）。这也呼应着intel所谓的“乱序执行”技术。

然而，鉴于除法的不确定性，CR4把这条流水线给解耦了，因此它不适用于刚才讲到的并行化处理。另一方面，CR4的流水线还自行解决数据紊乱危相（data hazards），使得其它的流水线如果需要除法运算的结果，（就不会使用预取的寄存器值），而是会在除法运算完成前停工，（待除法运算把商写到寄存器后，才使用寄存器的值来作相应的后续运算）。

加载－存储流水线也有一点个性：与其它流水线相比，它的左边多出了一块，在主干流水线的“发布（issue）”级中挤进去了一个“地址生成”的操作。有了它，就使通用的加载操作保持在只有一个“加载-使用”penalty的水平。（penalty可译为性能恶化，但总觉得别扭）。“加载-使用”是指：在一条数据加载指令之后，紧跟着的下一条指令就要使用这个新加载的数据。此时，因为存储器供不上CPU的速度，流不线在执行下一条指令时就不得不停工等待（可见，我们在写汇编程序时，应尽量把要使用的数据提前加载，让加载指令的后面跟上一条不相关的指令。“早起三光，晚起三慌”的道理在高级流水线技术中也同样适用啊——译注）。在ARM9E中，也有此品性。

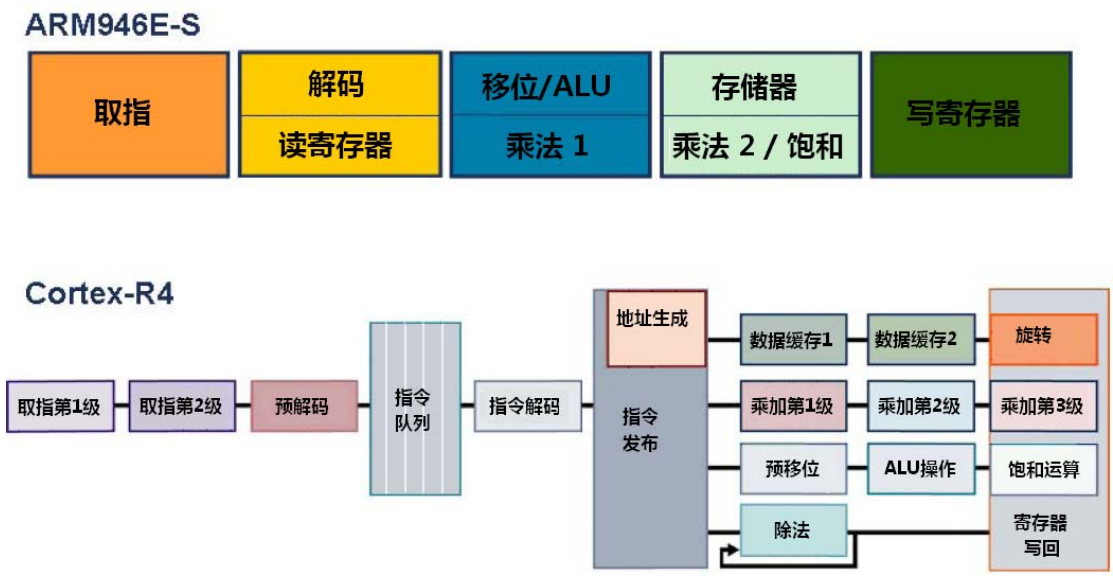

图1. ARM9E与Cortex-R4在流水线上的对比

双重发布（dual issue）

Cortex-R4的流水线能够无需添加重复的执行级，就可以有限地实现双重发布。它内部有一个后备的，（功能）有限的解码单元，它使得特定的一些指令对子可以并行地解码和发布。例如，对于“加载，加法”指令对子(LDR后跟随ADD)，它就可以在把加载指令发布到“加载－存储”流水线的同时，把加法指令也发布到ALU流水线，（从而使这两条指令并行执行）。

这项能力在仅消耗很少硅片面积的代价下，就显著地改善了处理器的CPI（每指令平均周期数）。在文献[1]中，有对双重加载的详细讲解。

有了这些特性的一齐努力，CR4的Dhrystone性能提升到了1.62DMIPS/MHz，从ASRM946E-S的1.14DMIPS/MHz相比，从儿童身高长到了成人身高（女性），使相同主频下性能提升了42%；也使相同的工作负载仅需70%的主频（从而降低了功耗）。

硅片面积

CR4 流水线复杂度比 ARM9 增加许多，若不加以处理，势必伴随着需要更多的硅片面积，这直接影响了成本。因此，设计 CR4 的团队非常仔细地工作，以使这种影响尽可能地小。CR4 的精工细作，使得即便让 CR4 的目标频率接近 ARM946E-S 的上限，CR4 也会更苗条，并且让综合工具更容易就满足时序约束，从而可以把占用面积优化得更狠。

分支预测

如果某指令在上一条指令的运算结果出来之前无法继续，这时就发生了数据依赖。这会恶化指令互锁，从而增加了CPI。这种现象在较长的流水线中尤其明显。另一方面，更长的流水线在因分支转移而清洗后，也需要更久的时间来重新填充。为避免它俩的不利影响，可以分别求助于“外延数据捷传”（extensive data forwarding）和分支预测的技术。CR4使用了分支预测技术。它会在流水线比较靠前的级中计算分支指令是否的确会发生跳转。如果计算结果认为不会中转，就继续预取紧跟着的下一条指令；如果认为将要跳转，就会从分支目的地预取指令。这样，只要计算正确的概率大，就能减少流水线被清洗的概率。

AMBA 3 AXI

与ARM946E-S所使用的32位AMBA AHB存储器接口相比，Cortex-R4使用了64位的AMBA 3 AXI存储器接口，从而获取了不少的性能增益，包括：可发布多重待决地址，支持数据以乱序返回。其中，对大多数应用程序来说，最鲜亮的优势，就是总线不会在访问慢速的存储器或外设时被阻塞，从而被慢动作的不会拖内核的后腿，使内核可以执行进一步的访问。把总线加宽到64位，也增加了可用的带宽，使得cache填充（8字）只需4次访问，下降了一半。

中断延迟

在Cortex-R4处理器系统中，引入了一系列的特性，可以降低both最坏情况和平均情况下的中断延迟。比如，CR4可以在某个多重加载指令启动后放弃执行该指令；在中断服务程序开始后，可以使用新的指令来存储并改变处理器模式。当然，非阻塞的AMBA 3 AXI总线也功不可没。

（下一段文字精彩地演示了cache对实时系统确定性的破坏——译注）

在ARM946E-S中，如果在某个加载16字的LDM指令刚刚开始执行后，就收到了FIQ请求，那么FIQ就遭遇了最冷的板凳，此时也是FIQ的最坏情况下延迟时间。ARM946E-S不会放弃LDM指令，必须等它结束后才会处理中断。而10个字就有可能跨越3条缓存线（每条8字），使得必须从AMBA总线加载24个字。更加火上浇油的是：有可能需要为加载3条缓存线找出3个victim（在缓存已满时），结果不巧victim还是脏的，于是又得把victim对应的24个字写回到主存。如果再“屋漏偏逢连夜雨”：写缓存也偏偏在这个危难之际满了，就又需要写8个字到主存。最终的加载操作还可能遇到数据流产，又在响应时间上加上了3个周期。假设AMHA AHB时钟比例是常见的2:1，哪怕是理想到零等待周期的主存，也需要(24+24+8+3)*2=118个周期的延迟。请别说我这是鸡蛋里挑骨头，对于实时系统，必须要严肃地找出最坏情况下的

指标。即使限制LDM最多一次只加载4个字，并且没有外部流产，只有一半的缓存线脏了，且写缓冲只满了一半，中断延迟也要60周期左右。而且，因为这60个周期中涉及了大量的总线访问，使得存储器系统引入的任何等待状态都会洋洋洒洒地增加延迟时间。

CR4则不同，如果在LDM指令执行的过程中遇到中断请求，它就会放弃执行该LDM。于是，内核不再等待返回的数据，这就可以在数据cache线的填充过程中，也把ISR从指令cache或TCM中取出。此外，对AMBA AXI ID字段的使用，使得ISR无需等待上一个cache线填充完毕，就可以从AMBA AXI总线中取出。通过向量中断控制器（VIC）端口，可以把ISR的地址传递给预取单元，且此动作无需从AMBA AXI总线上访问外设。即便没有使用VIC端口，也可以在无需等待上一次cache线填充完毕，就可以从AMBA AXI总线上访问外设。要实现此功能，可以对可缓存的读取与不可缓存的读取分配不同的AMBA AXI ID，以允许它们乱序完成。只要小心地使用强顺序化的设备存储器（在它们上面不得放弃多重加载），就可以把最大中断延迟控制在20周期左右，同时对AMBA AXI存储器及外设的访问时间只有很少的依赖，甚至根本没有依赖。在CR4还可以选择让FIQ成为NMI，从而让软件无法掩蔽FIQ。

系统成本

处理器在一个ASIC中所占用的硅片面积，只是整体成本的一部分。所有的处理器还都需要其它的功能块来“捧”，如：存储器，外设，总线基础设施等。此外，开发成本以及所需的时间也为整体成本作出很大的贡献。光就硅片面积带来的成本来说，存储器乃是烧钱大户。CR4通过包含一系列的特性来减少成本。

Thumb-2

CR4处理器实现了ARMv7-R架构，它在支持ARM指令集的基础上，还包含了Thumb-2指令集。ARM9E则是基于ARMv5TE架构的，它包含的Thumb指令集是第一代的Thumb。

ARM指令集中的指令都是定长的32位的。由于有充足的比特数，使得ARM指令集的编码非常强大，它很容易就能淋漓尽致地发挥ARM9E族处理器的性能。Thumb则是16位编码的一个替补，它可以使代码密度提高约35%。也就是说，相同的函数，如果使用Thumb指令，则平均能瘦身35%。然而，由于16位的编码只能实现ARM指令集功能的一个子集，Thumb代码的性能则不如ARM代码。而且，Thumb指令集的功能不足以覆盖ARM架构的方方面面。例如，在Thumb下就不支持掩蔽中断（中断服务例程也不能使用Thumb写就）；因此，所有ARM9E的处理器的程序都必不可少地会使用一些ARM代码。如果不同的函数使用不同的指令集，则程序员就要根据每个函数的特点，以及优化的目标来作出选择。

Thumb-2指令集是Thumb指令集的一个真超集，因此与Thumb目标码在机器语言的水平上是后向兼容的。ARM9E处理器的代码无需重新编译或重新汇编，就可以直接上CR4。而Thumb-2决定性的优势，其实是在于它还提供了很多32位指令，使得只使用Thumb-2也可以完成全部的操作（Cortex-M3就干脆只支持Thumb-2——译者注）。有了Thumb-2，16位与32位指令之间的隔阂不复存在，可以搂抱在一起了。因此，无需不停地切换处理器模式，这对于简化开发无疑是个激动人心的好消息，同时也使得编译器能在优化代码尺寸时，能更得心应手地控制优化力度。通过上述的优势，Thumb-2既可以做到ARM代码的高性能，又保持了Thumb的高代码密度。与运行ARM代码的ARM9E处理器比起来，这下对程序存储器容量的需求变少了。另一方面，与运行Thumb代码的ARM9E处理器比起来，这下“血压”也降下来了——需要的工作频率可以更低。在文献［2］中有对Thumb-2指令集的详细讨论。

RAM需求

ARM9E和CR4都支持本地存储器（local memory），本地存储器可以通过TCM或cache的形式来呈现。片上的RAM的访问速度必须能跟得上内核的频率。在ARM946E-S中，这种RAM需要40%左右的内核时钟周期时间来作出呼应。例如，如果某ARM946E-S处理器运行在200MHz的频率上（周期为5ns），则RAM的呼应时间不能超过2ns。

在CR4中，流水线会在逾2个周期的时间中访问本地RAM，这大大地放松了对RAM呼应速度的要求：给它们100%的内核时钟周期时间来完成呼应。所以对于运行在200MHz的CR4，允许RAM可以慢到5ns才作出呼应。即便把CR4的主频升到400MHz，也能留给RAM 2.5ns的反应时间——比ARM946E-S在200MHz下的要求还要松20%呢！于是，我们可以选择低速的存储器，比如，我们可以使用Artisan Metro RAM来取代臃肿的Artisan Advantage RAM。这就节省了35%的面积，还节省了54%的功耗。站在设计人员的立场上看，这还使得时序约束更容易满足，为他们大大减负，从而设计周期和风险都可以降低。

灵活性

在综合ARM946E-S处理器时，只有少数可以选择的参数，主要就是cache和TCM的容量。CR4则提供了更多的配置选项，从而允许处理器更能为应用的需求而量身定制，从而扩大了CR4能胜任领域的范围。

TCM灵活性

ARM946E-S和Cortex-R4都支持所谓的“紧密耦合存储器（TCM）”，这是一种本地存储器架构。对TCM的支持体现在both指令和数据上。然而在ARM946E-S中，必须在物理实现上为指令和数据分别开出各自的RAM，代码不能在数据TCM中执行。并且虽然允许在指令TCM中访问数据，这样做却会恶化性能。于是，必须在二者容量的分配上做出艰难的抉择。

在Cortex-R4中，TCM要远远地灵活得多。CR4有三个存储器ports，而程序员则可以看到两个分开的存储器regions。其中两个ports可以在访问单一的地址region时合并，合并方式既可以是基于bottom/top的对半式，又可以是细密交织式（finely interleaved）。在细密交织式中，偶数序号的（evenly addressed）双字存储在一个RAM中；而奇数序号的（oddly addressed）双字则存储在另一个RAM中。在综合的时候，设计者可以选择只实现一个RAM，也可以两个，甚至三个都实现。当实现了多个独立的RAMs时，内核可以并行地访问它们，于是提高了性能。并且和ARM946E-S不同的是，在综合期间不再需要把它们按指令与数据分开对待。处理器中有一个内部的总线矩阵，它可以把任意的访问类型派到任意一块RAM中。只有当数据访问和指令访问的地址指向同一块RAM中时，才会出现延时。

在CR4中，这些存储器是64位宽的，而在ARM946E-S中则是32位的，带宽在CR4中得到了提高。另外，ARMv7指令集的特点还降低了对文字池的访问频度（更多位数的数据能够存储在程序代码中），这更使得设计者在只实现一个TCM RAM时，性能的恶化降到最低。但是在大多数情况下，表面上看起来是单一的逻辑存储器，内部实际上实现两个单独的RAM，这对于改善布局和时序都有好处。在这些情况下，不会因为使用了两个TCM ports就产生了辅加的成本。连接方式非常直来直去，不再需要MUX。

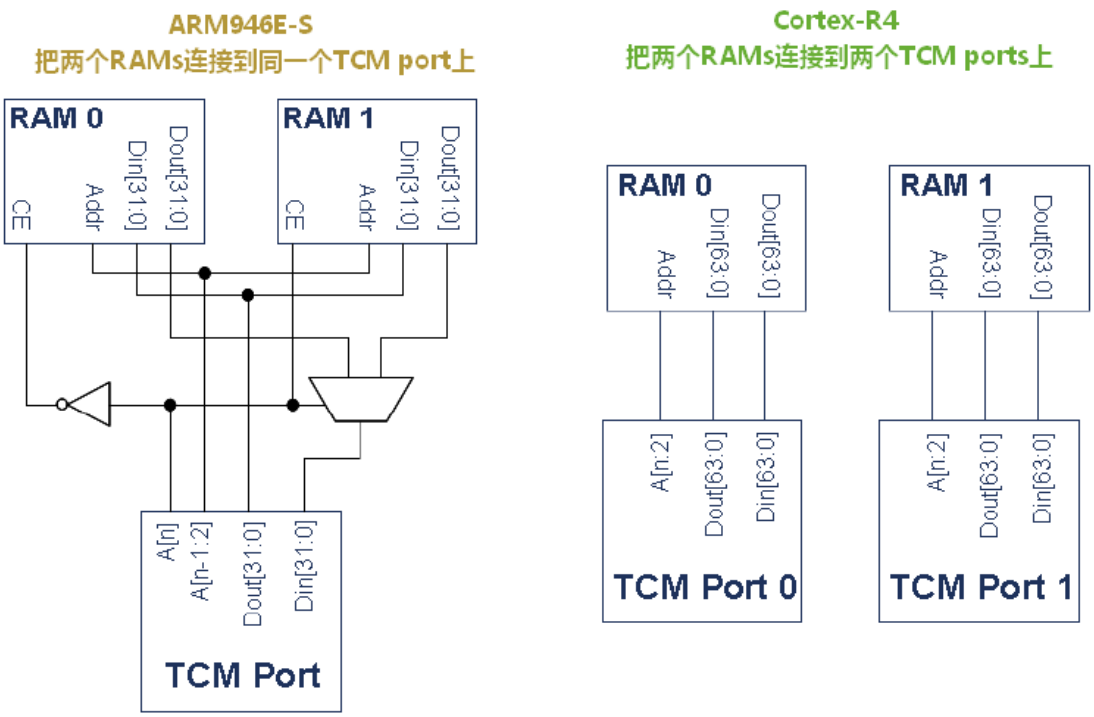

Figure 2: 紧密耦合存储器 - RAM连接方式

DMA

CR4众多创新的其中一项，就是为DMA包含了一个port，这在ARM946E-S和ARM966E-S中都是没有的。这个port是AMBA AXI的一个从port，用于简化外部DMA控制器（或者是其它处理器）访问TCM的难度。使用这个port的访问，与内核的指令和数据访问有相同的地位，都使用内部矩阵并服从它的仲裁，并且可以与它们并行不悖。如果TCM是按交织双字的方式来实现的，则更是佳配：（因为奇偶的伴行），内核与DMA访问可以通过交替地(alternate)访问奇序号和偶序号的双字，从而轮流地访问同一个地址。于是，在使用双口(dual ported)RAM时，就可以做到让它们不争抢总线，大大提高了访问效率。

MPU

ARM946E-S有一个8 region的MPU，每个region最小容易为4KB，这个MPU是必配的，因此在基于ARM946E-S的系统中总会有这个MPU（其它ARM9E的成员有些是可以不带MPU的，虽然它们并没有cache）。CR4上的MPU可以配置成支持8个regions或12个regions，使得在增加灵活性与节省硅片面积的选择之间更加灵活。在CR4中，也可以彻底忽略的MPU，此时则把缺省的保护属性映射到不同的地址区间中。CR4的MPU region最小可以只有32字节，以允许更精细地控制，从而降低了对存储器的浪费。

架构

ARMv7架构包含了3个款式（“应用”款式，“实时”款式，以及“单片机”款式）。Cortex-R4实现了实时款式（ARMv7-R）。这种分法，使得架构及其实现的目标领域都更明确，从而可以进一步地优化配置。例如，实时款式支持在嵌入式控制应用中非常重要的硬件除法指令。ARMv7-R对ARMv5架构是“二进制兼容”的（注意ARMv7-M则不能——译注），这意味着运行在ARM946E-S(v5架构)上的代码可以原封不动地移植到CR4(v7-R架构)上。但如果要

做针对性的优化，则最好为CR4重新编译一下代码（因为有了thumb-2）。另外，操作MPU的代码需要改动。

小结

Cortex-R4比ARM9E族相比，both在主频和计算效率的性能考量上，都有了真切而显著的提高。进一步地，因为可配置的选项更多，CR4能够为具体的应用更加量体裁衣。这些进步并没有建立在牺牲功耗或是占用面积上，这注定了CR4将会很成功。因此，对于需要高性能，低主频（对应了低功耗）以及低成本的应用，CR4是一个理想的选择（按ARM的话，CR4主要用在非常高端(high volume)的深度嵌入式系统中，比如硬盘，inkjet打印机，汽车安全系统以及无线modem等，但译者感觉ARM9E能做的它都能做，可能剩下的比较小型一点的都留给CM3做了吧——译注）。在降低成本方面，CR4不但对RAM的需求量更少，还简化了设计难度。ARMv7-R在保持与ARM代码的兼容性的同时，还提供了优化的余地——主要是在Thumb-2指令集上做文章。

参考文献

[1] “Cortex-R4: A mid-range processor for deeply-embedded applications”（Cortex-R4，为深度嵌入式应用而定制的中级处理器）, ARM 白皮书

[2] “System solutions for a baseband SoC”（基带SoC的系统解决方案）, ARM 白皮书, Dom Pajak.

[3] “Performance of the ARM9TDMI and ARM9E-S cores compared to the ARM7TDMI core”（ARM9TDMI与ARM9E-S的性能与ARM7TDMI的比较）, ARM白皮书

[4] “Cortex-R4 Technical Reference Manual”（Cortex-R4技术参考手册）, ARM

[5] “ARM9E Technical Reference Manual”（ARM9E技术参考手册）, ARM

[6] “ARM946E-S Technical Reference Manual”（ARM946E-S技术参考手册）, ARM

http://www.arm.com/documentation