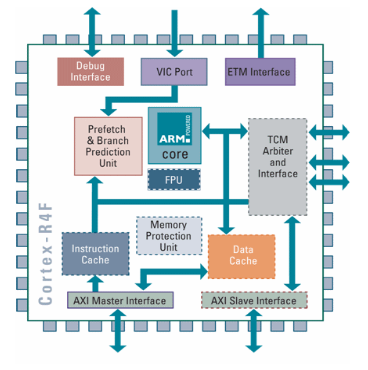
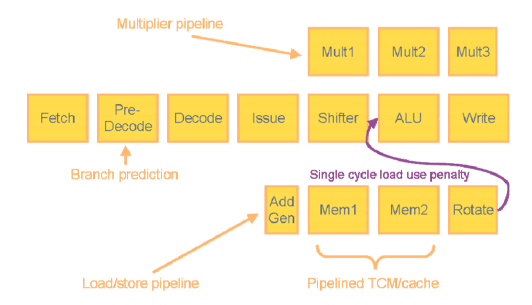
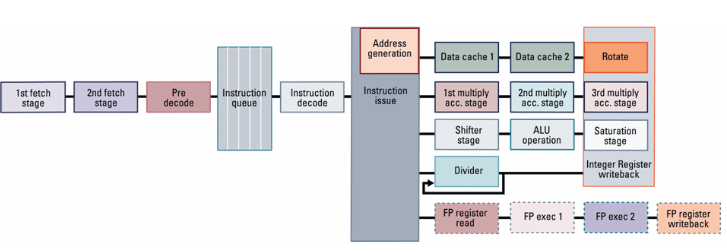
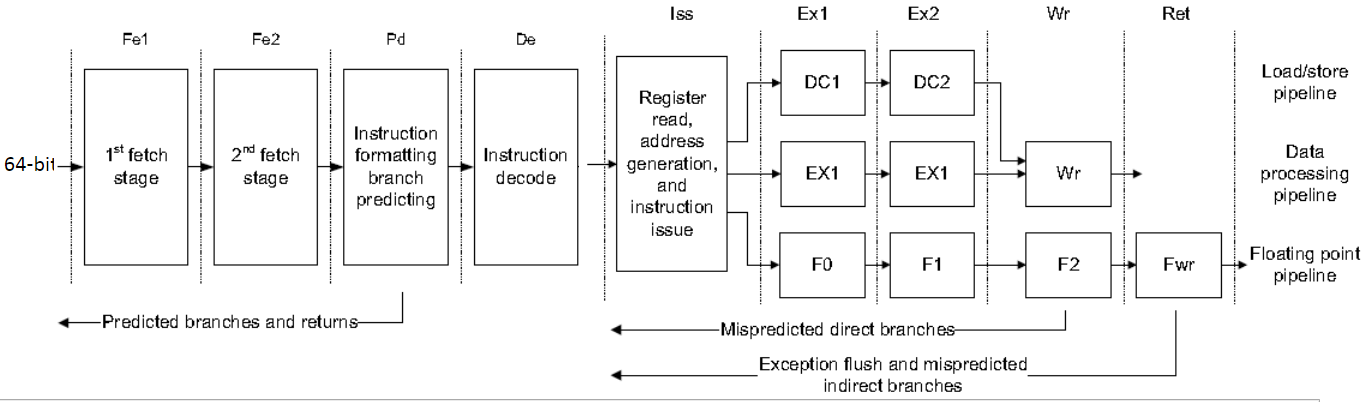
# Cortex-R4内核

Cortex-R4使用64位的AXI 3总线协议，并且TCM、Cache、MPU、AXI-Slave、FPU等都是可配置的。是具有指令预取和指令队列、分支预测结构及双发射的处理器微结构。









八级流水线中有两个周期用于指令获取和数据访问。在DPU(Data Processing Unit)阶段流水线分为四个并行的路径和浮点运算路径(如果存在)，每个路径处理不同类型的指令，指令通常并发处理：

Load/Store: 此路径处理所有内存访问。内存访问分为两个流水线阶段，可使更长的RAM访问时间而不丢失带宽(bandwidth).

MAC: Multiply-accumulates(乘法累加)分为三个流水线阶段，在最后一个阶段更新寄存器堆，所以如果相邻的下一条指令要使用MAC结果，那么会多增加一个周期用于加载.

ALU: 算术和逻辑运算分为三个流水线阶段，第一个阶段使用一个操作数进行预移位(pre-shift)，第二个阶段是基本的ALU运算，第三个阶段是如果需要在更新寄存器堆前饱和(saturate)，即Saturation stage.

Divider: 除法路径使用Radix-4算法，在单个流水线阶段典型的32位除法大约需要6个周期.

Load/Store路径在指令发射阶段就会生成地址，以平衡其他流水线路径，这样可以减少数据冒险的延迟。

ret-stage

dc1-stage

dc2-stage

dc3-stage data cache?

rt2-stage

pd-stage pre-decode

dec-stage decode

iss-stage issue

ex1-stage execute 1

ex2-stage execute 2

wr-stage writeback?

## PFU (Prefertch Unit)

### 结构综述

PFU负责从指令存储器中获取64位宽的数据，并从中提取指令(包括ARM和Thumb-2)，并向DPU提供指令流。PFU设计用于L1 Cache和TCM，其中典型的访问时间是两个周期而不是一个周期。因此，PFU包含一个三输入的64位预取缓冲区(pre-fetch buffer)，只要预取缓冲区中有空间，PFU就能获取下一个64位数据包。

PFU包括两级预取(Fetch)和一级预译码(Pre-decoding)，在预译码之后会将指令推入指令队列(IQ, instruction queue).指令队列(IQ)是一个FIFO，一次可读写一条或两条指令。分支和函数返回预测发生在预译码阶段。指令队列(IQ)最多存储八个预译码指令，IQ并不属于流水线阶段。

PFU可以一次从内存系统中获取64位数据，此数据保存在预取队列(prefetch queue)中，并从中提取指令。每个双字(double word)可以包含两条或多条指令，在PFU预译码(Pd)阶段能够一次提取两条指令并将它们写入IQ中。通过这种方式，整个流水线能够持续每个周期吞吐两条指令，既不会死等内存系统，也不会因等待执行流水线能力而停滞。

### 分支预测器

在PFU中，有两种预测器：Taken-ness预测和目标地址预测。

‘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.

“Taken-ness”分支预测预测目标地址与程序计数器的固定偏移量的分支。这些分支由动态分支预测器使用全局历史方案来处理。全局历史预测方案是一种自适应预测器，它在程序执行期间学习分支的行为。一旦评估了分支的条件代码，预测将在稍后的管道中进行检查。条件返回指令将由分支预测器预测其“取数”。预测程序流变化（如分支）允许PFU在DPU执行分支之前开始从分支目的地获取指令，从而在预测正确的情况下节省了大量周期。

对于目标地址不是程序计数器的固定偏移量的分支，只有当目标是函数返回时，才能容易地预测目标。函数返回预测使指令能够通过返回堆栈预测其目标地址。

返回堆栈用于加速函数的返回。对于每个函数调用，返回地址被推送到硬件堆栈上。当硬件识别到函数返回指令时，将从返回堆栈中弹出一个地址，并将其用作预测返回地址。当正确的返回地址可用时，在管道末端检查预测。返回堆栈有四个条目深。

## DPU (Data Processing Unit)

### 结构综述

DPU负责指令译码、发射、执行等内容，在经过PFU之后得到的是已经预译码的指令，PFU会将这个预译码的指令推往DPU中的IQ，可以缓存指令。指令会经过译码阶段进行完全的译码，得到立即数等数据。之后会进入发射(issue)阶段，此阶段控制信号会根据指令类别的不同发往不同的通路，即MAC、ALU、Store、DP(Divider)。对于MAC、ALU、Store在流水线上都会有ex1(execute1)、ex2(execute2)、wr(writeback)三个流水线阶段，但具体通路间是独立的，可以达到同时执行不同指令的目的，即超标量双发射运行。在最终向寄存器堆写入数据时，是通过一个MUX来选择写入的数据来自哪个通路。

包括ALU、MAC、Store三条数据通路，也包括了选择这些数据的MUX。在Issue发射阶段，有六个数据bus(data buses)，即store pipe/三个alu pipe/两个mac pipe。

Interfaces with LSU to transfer data to or from the memory system.

与LSU接口，用于将数据传输到内存系统或从内存系统传输数据

Holds general-purpose registers, status registers and control registers (CP15, CP14, etc.)

保存通用寄存器、状态寄存器和控制寄存器（CP15、CP14等）

### ctl (control signal pipeline)

此模块是控制通路信号的流水线。

This module pipelines all the control signals that are generic to the DPU pipeline, ie. not associated with a specific data path.

该模块对DPU流水中通用的所有控制信号进行流水传输，即与特定哪条数据路径(datapath)无关。

It also includes the logic for the Issue stage, the Interlock logic and the Forwarding logic.

它还包括发射阶段的逻辑、Interlock逻辑和前递逻辑。

### dbg (Debug block)

主要包括了debug相关的一些寄存器，包括：

Debug Status and Control Register (DSCR)

Debug Run Control Register (DRCR)

Instruction Transfer Register (ITR)

Data Transfer Registers (DTRRX, DTRTX)

Watchpoint Fault Address Register (WFAR)

Debug State Cache Control Register (DSCCR)

Integration Mode Control Register (ITCTRL)

Integration Mode – ETM interface (ITETMIF)

Integration Mode – Miscellaneous outputs (ITMISCOUT)

Integration Mode – Miscellaneous inputs (ITMISCIN)

### de (Decode)

包括IQ，Decoder，PC/mode/IT 等逻辑部分。主要是包括FSM或IQ中有一些时序逻辑。

### dp (DataPath, containing data muxes)

#### ALU通路

EX1 Stage:

包括了 a) The RBIT unit (set of muxes).

b) SAT \* 2 unit required by {QDADD, QDSUB} instructions.

c) 32-bit Barrel Shifter plus the logic which encodes the type of shift and shift value. This block also contains the Sign extension logic required by the XTEND and XTRACT instructions.

d) Forwarding Mux.

e) Second mask generator

EX2 Stage:

包括了 a) Arithmetic Unit.

b) Logic Unit.

c) The {NZCV} Condition Code flags generation logic and the GE flags generation logic.

d) The Flag Setting Logic.

### etmif (ETM interface module in wr-stage)

此模块为ETM (Embedded Trace Macrocell，嵌入式追踪宏单元)的接口，暂时无需关心。

### ldst (Load/Store phantom pipeline)

此模块主要是LSU单元的控制信号的流水线。

This module pipelines the various control signals required for Load/store transactions as they travel through the LSU pipeline. These signals form a “phantom” of the LSU transaction, and enable the correct operation when the result is produced in the wr-stage.

当Load/Store指令通过LSU流水线时，该模块(ldst)会传送Load/Store所需的各种控制信号。这些信号形成LSU事务的“phantom”，并在写回阶段(write)产生结果时启用正确的操作。

Most of the wr-stage LSU interface signals are handled in this pipeline.

大多数写回阶段的LSU接口信号都在该流水中处理。

### RegBank

4 Read Port and 2 Write Port.

## 其他

### 流水线冲刷flush

流水线冲刷有两个信号，flush\_ret和flush\_wr。

// Flush logic //

There are several sources of flush

- direct branch

- indirect branch

- exception

Different branch signals come from the various events in the various blocks. They are combined here to produce to overall flush signals : flush\_ret (kills everything) and flush\_wr (kills everything de->ex2) In practise, if flush\_ret is asserted, then flush\_wr will certainly be asserted .

# Appendix

dpu Data Processing Unit

lsu Load-Store Unit

pfu Prefetch Unit

mpu Memory Protection Unit

msh Mesh?

lfb linefill buffers

IQ Instruction Queue

fwd Forwarding

agu Address Generator Unit

dp Datapath (ALU&MAC)

rf register file