Arm training

VC0882 总结之二

张健
2011-12-27

outline

- 代序 : arm 是个特殊的模块
- Arm external api
- Arm debug tools
- Arm development environment build up
- Cortex-A features
- Arm Powered SOC

arm 是个特殊的模块

master, software

VC0882 Chip Block Diagram

arm 是 AXI 总线的一个 master

仲裁机制与
Performance
monitor 会单独介绍

Arm external api

从软件角度需要了解 arm 哪些东西？

Cortex-A8 block diagram

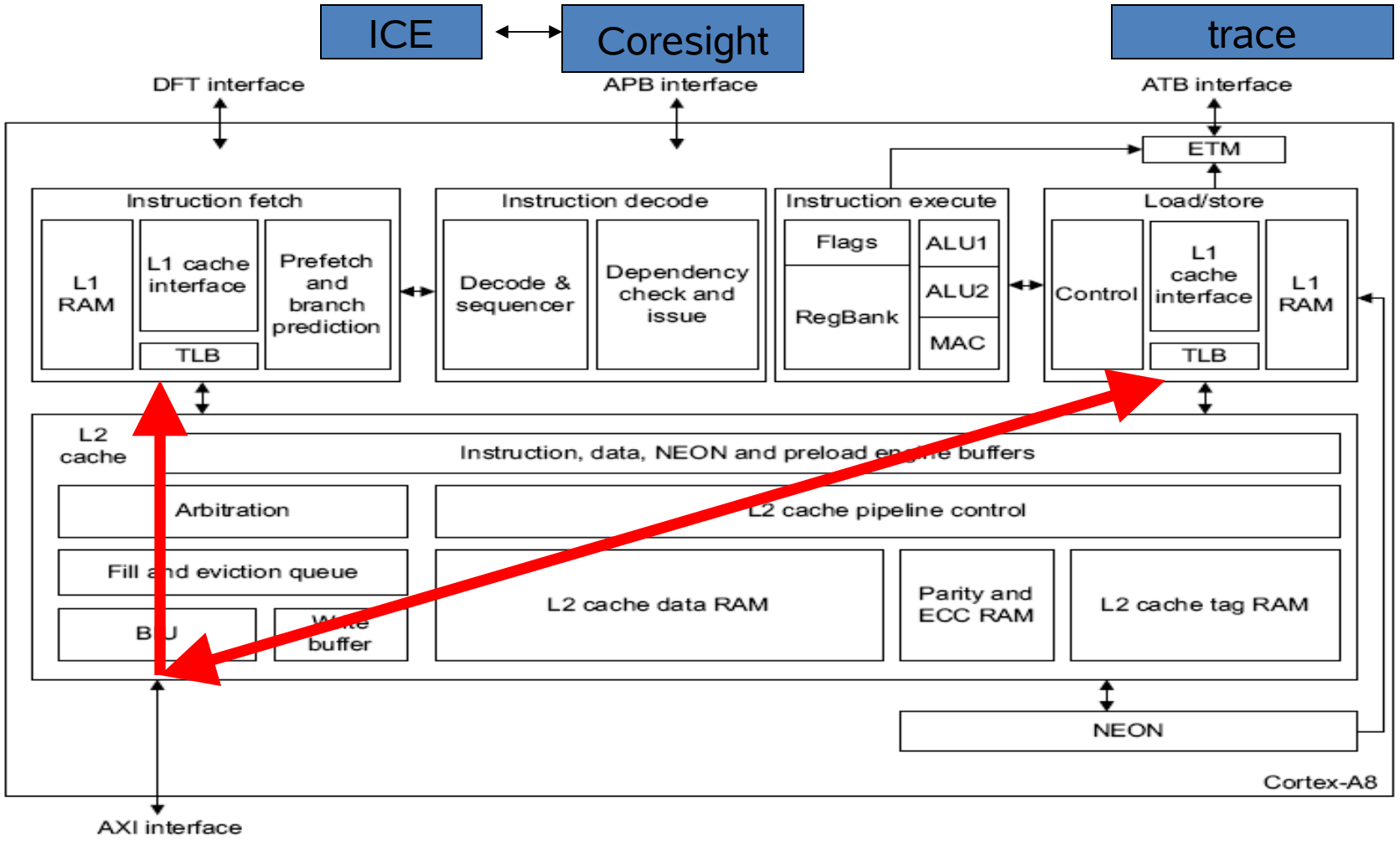


Figure 1-1 Cortex-A8 block diagram

Arm external api

- Memory:
  - memory 区域划分
  - cache align
  - cache maintenance.
  - mmu, cache, tlb 对于软件的影响 .
  - cache debug tools.
  - Memory 属性
- 中断和异常处理
  - OS 对于中断一般有较好的支持 .
  - 异常不等于出错 .

Arm external api

memory 区域划分

Arm 架构是统一寻址的

vc0882 memory map

device	size	start_address	end_address	溢出 / 越界访问的结果
sdram	2GB	0x8000,0000	0xffff,ffff	memory 绕回， ddrc 有 overflow 状态
reserved	256MB	0x7000,0000	0x7fff,ffff	default slave， 不会报错
register	256MB	0x6000,0000	0x6fff,ffff	apb time out. 详见 X2P mas
emi_cs1	256MB	0x5000,0000	0x5fff,ffff	
emi_cs0	256MB	0x4000,0000	0x4fff,ffff	
spi_boot	256MB	0x3000,0000	0x3fff,ffff	
sram	32KB	0x2000,0000	0x2000,7fff	arm 得到 data abort， 其他模块应该会得到 error(\todo 确认)
rom	32KB	0x1000,0000	0x1000,7fff	

访问不允许访问的地址会出错，但 arm 不一定 data abort. 其它模块也不一定得到 error

为了符合 arm 复位后从 0x0 地址开始运行的规则，0 地址需要是某种 mem

memory remap for 0x0000,0000 address

	arm addr		remapped addr		
condition	start_addr	end_addr	start_addr	end_addr	notes
boot_sel=2'b11	0x0000,0000	0x0fff,ffff	0x8000,0000	0x8fff,ffff	remap to sdram
boot_sel=2'b10	0x0000,0000	0x0fff,ffff	0x4000,0000	0x4fff,ffff	remap to emi_cs0
boot_sel=2'b01	0x0000,0000	0x0fff,ffff	0x3000,0000	0x3fff,ffff	remap to spi_boot
boot_sel=2'b00	0x0000,0000	0x0fff,ffff	0x1000,0000	0x1fff,ffff	remap to rom

882 pmu 映射引入的问题

- 除了 arm 以外的模块都看不到 0 地址． 如前述，模块访问 default slave 仍然会 done. 不查询 error 没法知道错了．
- 0x0 和 0x80000000 开始的两个 256Mbytes 区域是同一个 ddr 区域．但是对 arm 来说这是两个区域． 如果两个地址区域都是 cachable，会有 cache 一致性问题．见后面 shuyu 的 cif 问题．

cache

- cache align: cache maintanence operation 只能以 cache line 为单位 .
- cache maintenance: memory 分配由 OS 管理 . 但是 cache maintenance 在 nucleus 里面需要自己做 . Linux 里面有些情况也需要软件自己做 .
- Cortex-A 架构里面不支持单独 invalidate 到 memory 的，只能使用 clean and invalidata 到 memory 。
  - 对软件流程的影响。

882 中遇到的与 cache 的问题

- wangzhengwei, pmu normal->sleep, self-modify code.
- shuyu cif 0x8xxxxxx cache,memory 一致性问题 .
- yuyang dmac 部分 memory 区域没有加 cache 操作 .
- Venc 编码问题 : buffer 使用之前需要先 invalidate ， 即使软件”没有“用过。 AE venc: f_osync
- Suspend/resume: ddr 进入 self-refresh 之前保证后面在 sram 运行 code 所需页表都已经进入了 tlb


美加净
maxam
银耳珍珠滋养霜
天然护肤经典系列

Cache debug tools

- D:\VC0882\document\arm\cache
- 文档由 yangxing 总结，这里不再赘述
- Cache debug tool 可以按照 cache 的寻址方式 (set-way 寻址) 访问 cache. 而不是对软件透明 .
- 该工具在 shuyu cif bug 上实际证明了之前的分析。可以读到预期的错误的数据（为了结果一致，使用 colorbar 作为 pattern ）。

Memory 属性

- 非对齐访问
  - mengfandong，非对齐 dataabort.
  - Yangxing: emi norflash ： norflash 需要特殊的写命令，写命令本身可能是非对齐的。
    - 16bit norflash，写命令可能是 8bit 的。
- AXI 上的乱序访问
  - Memory barrier
  - VC0882 CDC check 时发现的 pmu remap 风险。

中断

- 不支持嵌套
- OS 对于中断一般有较好的支持 .
- by the way， a8 本身也有出去的中断：performance monitor, cross trigger，这些留到后面介绍。

Arm 异常

- 八种异常。
- 再次强调异常不等于出错。
- 从 arm9(?) 开始，支持高地址和低地址两个异常地址。
- 从 Cortex-A8 开始，低地址异常向量可以放在任意符合对齐条件的地址。
  - VC0882 panda_os 支持把异常及打印放到 sram 里面，编译在系统出错时判断是否是 ddr 导致的问题。

Arm 异常

Exception offset	Exception (except monitor exception)	Operating modes	“正常”的异常
0x00	Reset	Supervisor	Software reset
0x04	Undefined Instruction	Undefined	Kgdb break inst
0x08	Supervisor Call (SVC)	Supervisor	Enter kernel space
0x0C	Prefetch Abort	Abort	Break inst
0x10	Data Abort	Abort	Page fault
0x14	Not used	-	-
0x18	IRQ (interrupt)	*Interrupt*	-
0x1C	FIQ (fast interrupt)	*Fast interrupt*	-

后面 debugger 介绍中，会提到如何用 rvdebugger 手工做异常处理

Arm 异常与 monitor 异常对比

Exception offset	Exception that is vectored at that offset from:	
	Monitor exception base addressa	**Base address for all other exceptions**
0x00	Not used	Not used
0x04	Not used	Not used
0x08	Secure	Monitor
0x0C	Prefetch	Abort
0x10	Data	Abort
0x14	Not used	Not used
0x18	IRQ	(interrupt)
0x1C	FIQ	(fast

VC0882 reset

- VC0882 的三种复位
  - 硬件复位
  - Watch dog 复位
  - Software reset, software reset delay
  - 复位时可以选择是否复位 debug logic 。
- Pmu 有寄存器可以读到 reset 原因 .

Undefined Instruction

- Arm 译码时得到未定义的指令 .
- 编译通过的代码，不论 c 还是汇编 (更高层的语言肯定更不可能) 都只会生成合法的 arm 指令。
- 出现原因
  - 编译器用错了。虽然很少见，但是在 Linux 里面可能没有配置交叉编译器。。。
  - Arm 处理器指令集向下兼容。但低版本 arm 不支持高版本 arm 的命令。
  - 代码是动态生成的。比如我写了一个可以写任何 cp 寄存器的 ko 。
  - 系统，尤其是 memory 不稳定。
  - Linux kernel kgdb 用来调试 kernel ，但不能和 gdb 相同。

Supervisor Call (SVC)

- 没有出错情况会导致 SVC 异常。
- 出现原因
  - user space 进入 kernel space，包括但不限于 Linux
  - semihost

Semihost

- Semihost 是 arm 提供的与调试器交互的一种方式
- 882 fpga 阶段需要在 Linux 下进一步测试 vivante gpu。但 AE 开始移植 Linux 后，发现 Linux 非常慢， top 命令几秒才能刷出一次。后来 zhangpu 发现是 rvdebugger 打开了 semihost： linux 运行很慢。原因是 rvdebugger 在每次 arm 进入 svc 状态时判断是否有 semihost 操作。
- semihost demo, semi host 下的 aasp. \todo: 包袱： 882 环境修改的一部分：使 aasp 仅依赖标准库。
- Semihost: 周三看 lishugang 仿真 VC0380 代码，似乎是由于 semihost 导致仿真出错（仿真环境没有像 rvdebugger 接管 semihost 指令，所以会到 undefined instruction 。

Prefetch Abort

- 取指中止：我的理解就是指令突然取不到了
- 出现原因：
  - Arm 地址跳变。（ mmu 开关）。
  - Gdb 断点
  - 系统，尤其是 memory 不稳定。

Data Abort

- 这个异常可能是情况比较多的一种异常了。
- Linux 缺页异常
- Memory 访问权限不满足。
  - Access permission
  - Demain.
  - 访问不存在的地址 .
  - X2p time out.

Arm debug tools

ICE, rvdebugger, qemu, trace

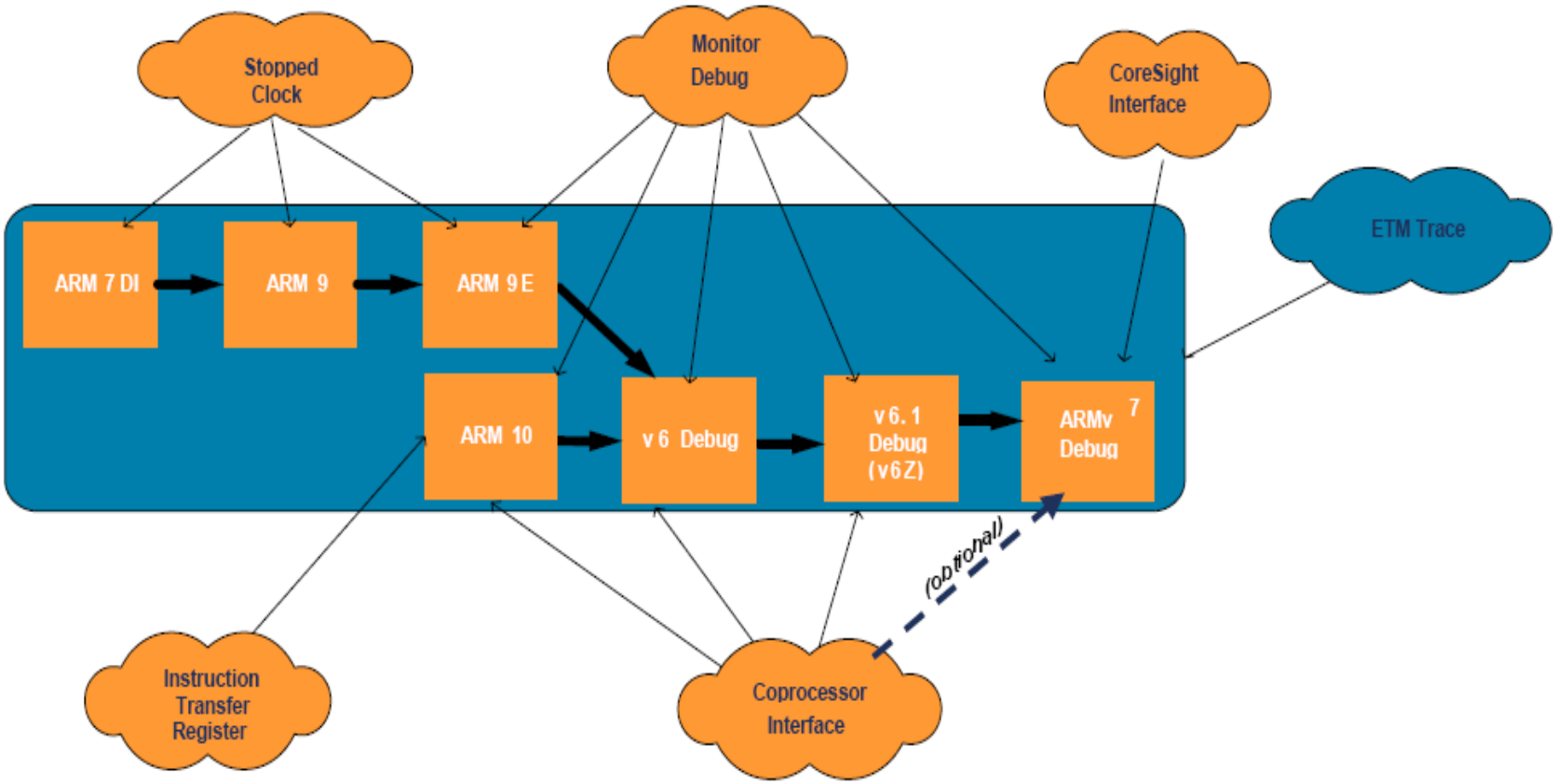
Breakpoint and trace

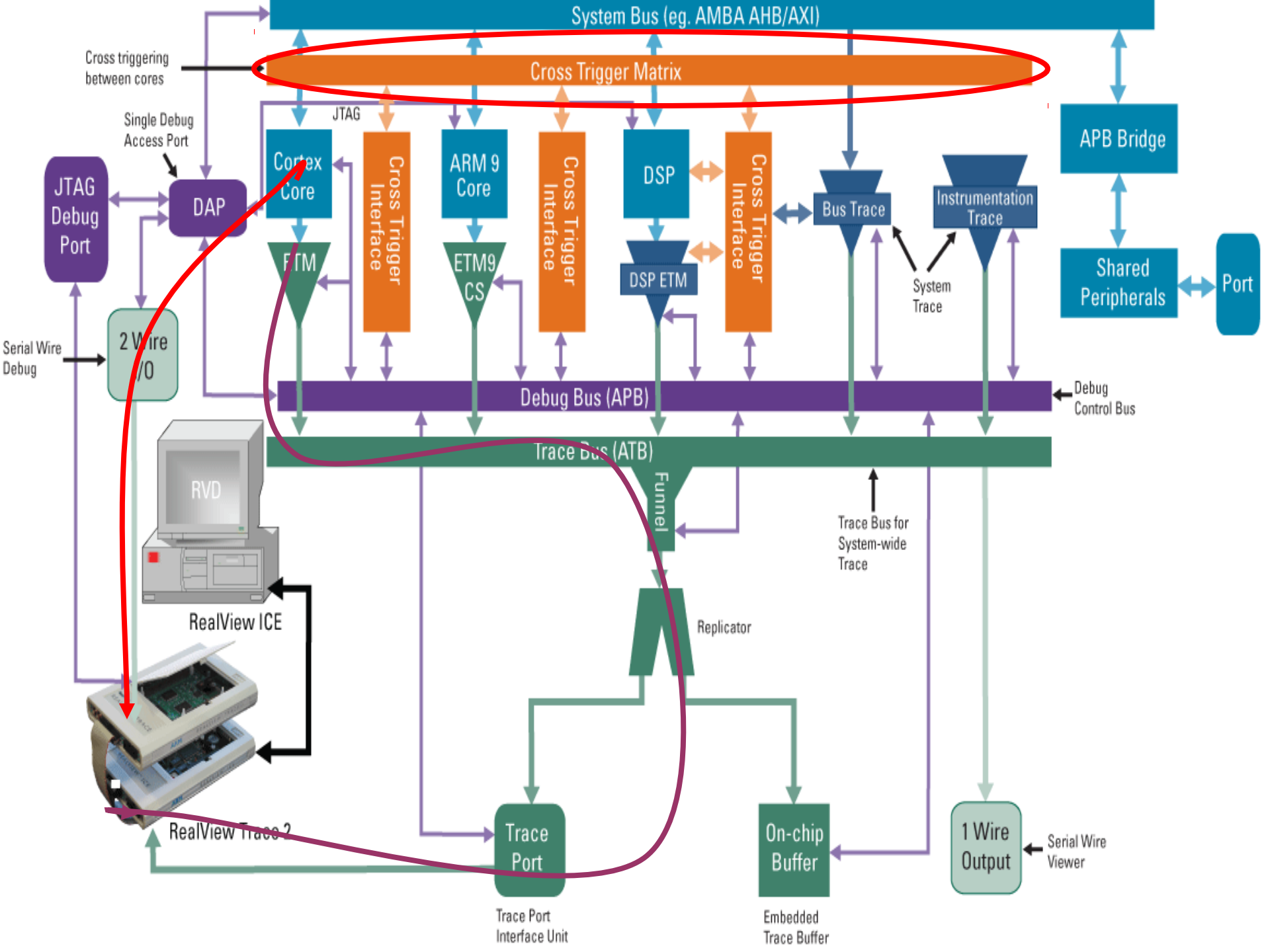
- 这里没有提到代码检查的工具，我不太了解。
- 调试工具其实包括很多，像 realview ICE, jlink, openocd 都是硬件调试器，此外还有 profiling 工具，例如 realview trace, trace32 等等 .
- 结合 caijin 的实践介绍 DS-5 的 profile 功能。
- Linux kernel 里面的调试工具 , kgdb, ftrace(这个之前有文档).
- 像 valgrind 这种纯软件 debug 工具 (memory 溢出检测，profile)，就不详细介绍了 . \todo 我之前试用过，可以简单说说？
- 模拟器其实是提供了一个理想的硬件，便于大家调试软件 .

Coresight and Jtag

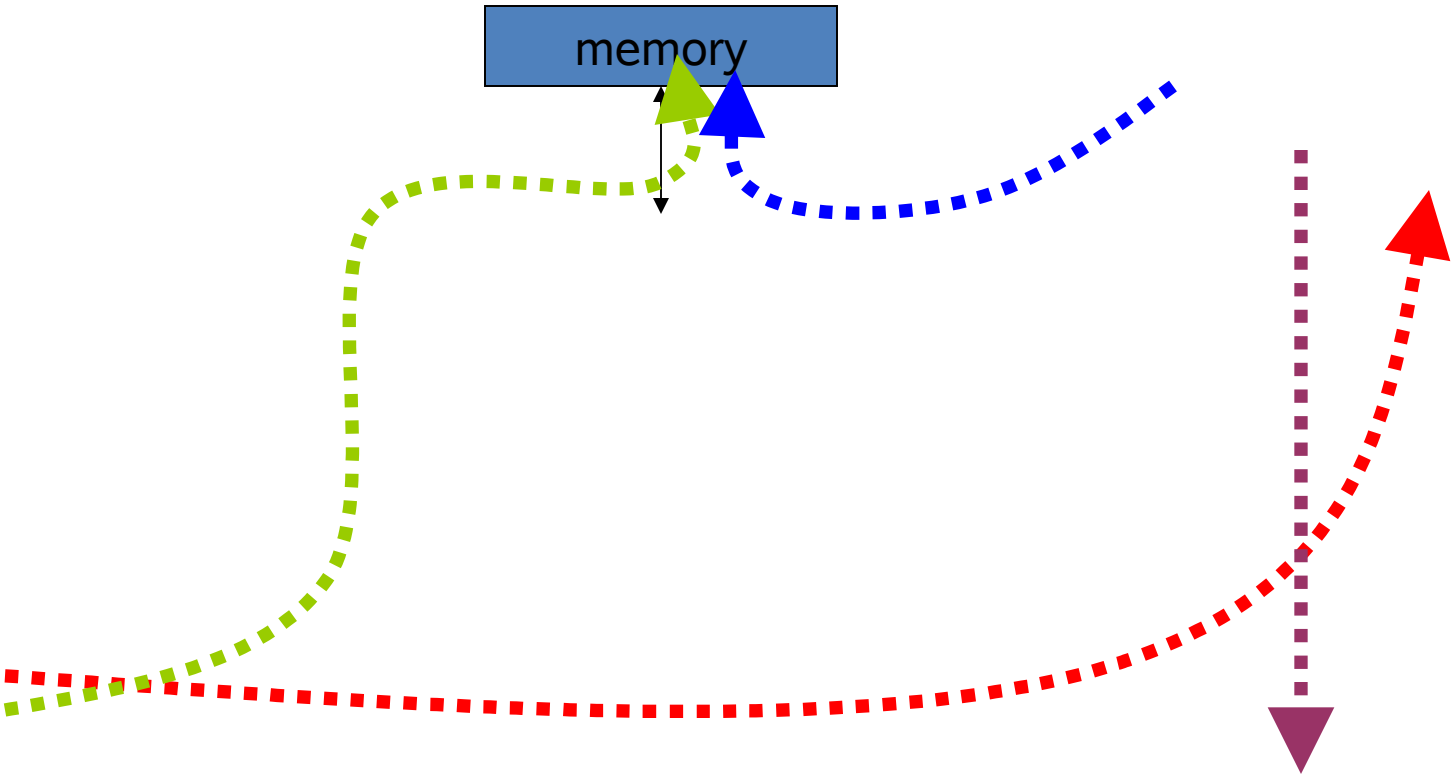
- Jtag: 基于扫描链的调试方式，广泛用于各种芯片。但是只支持调试调试，不支持 trace 。
- CoreSight 是 memory-mapped debug architecture. 通过 Cross Trigger 机制对多核调试有很好的支持 . 支持 trace 。
- debug mode and monitor mode. 理论上后者说可以在系统运行时通过 debugger 读回系统状态 . 只是没有见到应用 .

Arm SOC debug evolution




System Bus (eg. AMBA AHB/AXI)
Cross triggering between cores
Cross Trigger Matrix
JTAG
Single Debug Access Port
JTAG Debug Port
DAP
Cortex Core
Cross Trigger Interface
ARM 9 Core
Cross Trigger Interface
DSP
Cross Trigger Interface
Bus Trace
Instrumentation Trace
APB Bridge
Shared Peripherals
Port
ETM
ETM9 CS
DSP ETM
System Trace
Serial Wire Debug
2 Wire I/O
Debug Bus (APB)
Debug Control Bus
Trace Bus (ATB)
Funnel
Trace Bus for System-wide Trace
RVD
RealView ICE
Replicator
RealView Trace 2
Trace Port
On-chip Buffer
1 Wire Output
Serial Wire Viewer
Trace Port Interface Unit
Embedded Trace Buffer

VC0882 Cortex-A8 and Coresight



Realview ICE and debugger

- 除了硬件断点，所有调试器的基本原理都是使用指令停止 arm 运行，然后由调试器接管 arm. 只是这个指令，在硬件调试器 (ICE, jlink) 和裸奔的软件调试器 (qemu gdb) 等都是通过 break 指令，在 Linux 里面通过 media 的一条未定义指令实现 .
- 所有的调试器只能暂停 arm(通过 coresight cross trigger 也可能停止其他处理器), 但是其它模块仍然继续运行 .

- CVS 目录 : D:\VC0882\document\arm\debug\
- rvi 使用方法 .doc(log: "20:43 2010-8-9"), rvdebugger 问题集 .doc(log: "11:01 2010-6-8").
- 都是 882 项目期间整理的，开始遇到问题是直接支持，后来发现时间不够用，所以写成文档，建议大家先看文档 . 再找我 .

- \todo 写文档，加入到 "rvi 使用方法 .doc"
- 1), ICE demo: 需要一个异常调试的过程 .
- 2), 设置条件断点，断点事件 . \todo 联想 : gdb 调试也有类似技巧 . 举例子 .

- VC0882, VC0718 遇到的部分 rvdebugger 相关问题整理 :
- D:\VC0882\document\arm\debug\rvdebugger 问题集 .doc

ICE 技巧

- rvds 添加新的 RVI target
- rvdebugger 连接 target 运行程序完整流程
- rvdebugger 读写寄存器
  - 读写 arm 寄存器
  - Rvdebugger 读写 memory map 寄存器
    - 通过 memory 窗口（ view 菜单 ->memory 选项卡）读写：
    - 通过 rvdebugger 命令读写
    - 通过 rvdebugger 脚本读写
- **选择 rvds 是否在异常停止**
- **使用 rvdebugger load 符号表并调试**
- **rvdebugger 设置软件断点和硬件断点**
- rvdebugger 设置 connect mode， disconnect mode
- **条件断点与断点事件**
- **手工恢复异常现场**
- **rvdebugger 脚本， rvdebugger 宏定义**

选择 rvds 是否在异常停止

- 像 Linux 这样的系统，有自己的异常处理流程， rvdebugger 不应该干预。

符号表

- 作用
- 使用 rvdebugger load 符号表并调试
- 如何调试 rom 代码： VC0882 bootloader 调试。

软件断点和硬件断点

- 断点的概念
  - 硬件断点
  - 软件断点： break 指令（ 0xe1200b70 ）
- 单步调试其实是利用硬件断点。如果系统中硬件断点都用尽了，就没法单步（单步后系统不会停止）。

条件断点与断点事件

- Pass count
- Condition
- Break point event

出错调试

- 当 arm 出现 undefined instruction, prefetch abort, data abort 等错误时，可以通过分析如下寄存器确定出错具体原因：
  - arm register: lr, lr 说明了从何处跳转此处．可以用于定位在哪里出错．如果 lr 没有对应任何代码，可能是程序先跑飞到其它地址，然后跑死的．
  - rvdebugger register 选项卡的 control 页有 DFSR, DFAR, IFSR, IFAR. 这四个寄存器作用在 "D:\VC0882\document\arm\Cortex-a8\DDI0344J_cortex_a8_r3p2_trm.pdf" 有详细说明．
    - 它们分别说明在 data 或 intruction 出错时的具体错误．
    - 例如查 TRM, 可以知道，DFSR[11]=0 表示是 read data 出错．=1 表示 write data 出错．
    - [12][10][3:0] 这 5 个 bit 可以知道具体错误类型．例如 b000001 是 alignment fault.
    - DFAR: The purpose of the Data Fault Address Register (DFAR) is to hold the Modified Virtual Address (MVA) of the fault when a precise abort occurs.
    - 在 VC0882 里面，DFAR 通常就是产生 data abort 时读写的地址．
  - 如果跳回 0 地址，也可以做类似分析．有时系统不稳定会导致跳回 0 地址．

demo

- 手工恢复异常现场
- **rvdebugger** 脚本， **rvdebugger** 宏定义

模拟器

- 比较有名的例如 google 提供的 android 模拟器 (goldfish). 这个模拟器是基于 qemu 进一步开发的，在 android4.0 里面，通过 qemu 还做了 3D 加速 .
- 这里希望介绍 qemu 这个模拟器 . 我在 vimicro 期间，借助 qemu 做了一些事情，对我提供效率有很大帮助 .
- 此外还有 arm 提供的 ISSM, RTSM 等模拟器 . 在最新的 arm DS5 开发环境里面 . 可以用 RTSM 做应用程序调试的 demo. 只是这个 RTSM 似乎没有开源的 kernel 版本，当然也可以自己移植，工作量就是大一些。

qemu

qemu 是一个开源的模拟器．支持 x86, arm, mips 等架构．支持完全模拟和直接运行 Linux application 两种模式．我使用的是第一个方式．

- 之前和 LiaoZhiCheng 一起做 Linux porting 时，当时是希望完全独立完成．所以基本功能 ok 之前没有考虑 AE zhaoyuan team 的移植．
  - Linux porting 已经总结过文档，包括使用 low level debug 等方式调试．
  - 当时 kernel 运行起来以后，文件系统的 helloworld 迟迟没有打印．
  - 为了便于调试，制作了一个在 qemu realview 和 830sv 通用的文件系统 (rootfs, cpio 压缩，只是 inittab 里面支持两个 realview 和 830 的串口)．然后同步对比 kernel 加载文件系统的完整过程，后来发现其实文件系统 helloworld 已经执行了，只是没有打印．
- linaro 优化了 qemu，保证 linaro 编译的 beagle 和 realview Cortex-A9 MP 的影响在 qemu 和实际板子都可以运行．
  - 我实验过 linaro 提供的 image，可以正确运行到 android 启动，但是 usb 鼠标键盘，usbnet 有问题．看起来是可以的．在 beagle board qemu 开发其实比 goldfish 好一些．因为前者是有实际硬件的．
  - \todo: demo．我自己编译的 linaro beagle kernel 3.1，正确运行．
  - \todo 希望可以和 DS5 的 profile 实验．
  - "10:22 2011-10-24", linaro qemu + android 运行成功 (但输入设备无反应)．至少说明除了 goldfish 这种纯为模拟器设计的 SOC 模拟器，在比较真实的硬件上运行 android 也有现成的环境．对于需要考虑硬件环境的软件开发，在 linaro qemu 上开发应该好于 android goldfish 模拟器．
- qemu 调试 beagle board aasp.
- 其实如果稍加修改，可以在 qemu 上测试 panda_os 软件是否可用．有没有什么纯软件的 bug.

openocd

- openocd 是一套开源的调试工具 .
- openocd 与 arm 验证： 用 openocd 读出 OMAP3530 cp15 寄存器，判断该 SOC Cortex-A8 L2 cache 是否支持 parity/ECC.

Openocd

- 从 arm linux kgdb 分析可以知道，支持 gdb 调试是个分层架构．上层是对 gdb 调试协议的接口，下层是根据 gdb 调试协议的请求控制具体硬件进行相应动作．对于 openocd 这类支持多种 architecture 和多种 server 的硬件调试器 .
  - 上层除了接 gdb 还可以接 tcl 等 .
  - 下层则支持不同 architechture. 可以细分 :

```
------cortex-a8-----
          |
--------adi v5------
          |
---jtag interface---
          |
------openocd-------
```

其它

- kgdb 原理，和 gdb 使用 swi 指令不同，kgdb 使用未定义指令 .
- trace 调试，使用 .
- jtag 调试问题记录 : "16:07 2009-11-17"-1-1)-(3).
- jlink 使用文档
  - "12:37 2009-11-10“
  - "10:50 2009-11-4": openocd, jlink 使用 .

trace

- ARM SOC 支持 TRACE 功能需要三个条件
- arm core 支持 ETM
- arm soc 把 ETM 引到了 padc
- PCB 连出 ETM 接口 .
- Demo(板子信号有些问题，下次再演示)

DS-5

- DS-5 profile: xaos

Environment build up

Environment build up

- 以 882 为例说明，软件的修改 (包括 panda_os, mini_env), semihost 对于环境升级调试的帮助 .
- 882 项目从之前 830 项目的 arm926ej-s 升级到 Cortex-A8，为了支持 Cortex-A8，同时开发工具从 rvds3.0 升级到 rvds4.0，又由于 rvds4.0 只支持 Realview ICE，所以 debugger 从 multiICE, jlink 升级到 RealviewICE.

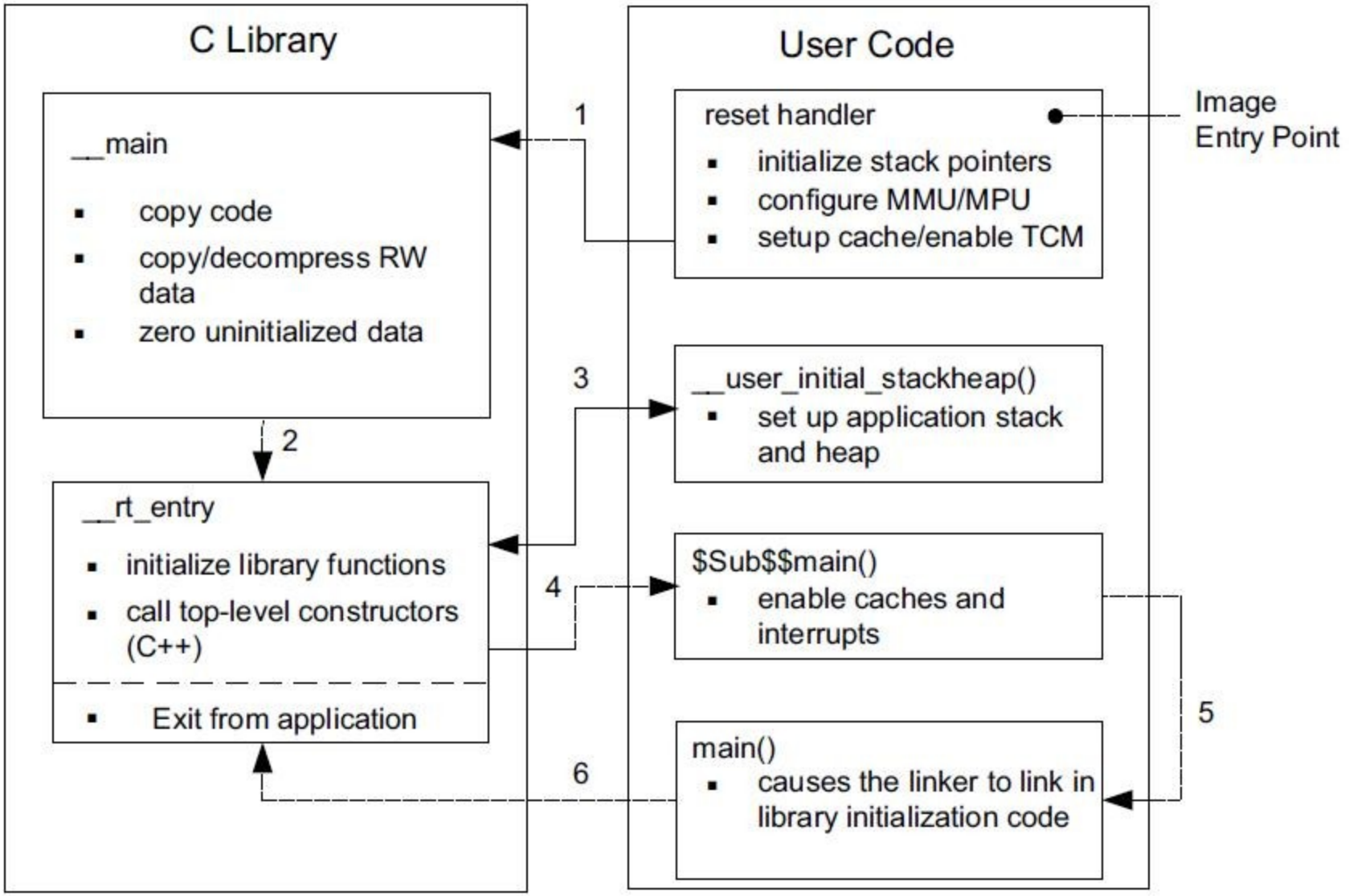


Figure 3-9 Initialization sequence

- 1，环境 .
- 这里介绍 VC0882 项目环境修改的内容．大的方便是说为了一个新项目准备环境需要做哪些事情，具体还是以 arm 为主，兼顾其它 (FPGA 稳定性等).
- 882 项目从之前 830 项目的 arm926ej-s 升级到 Cortex-A8，为了支持 Cortex-A8，同时开发工具从 rvds3.0 升级到 rvds4.0，又由于 rvds4.0 只支持 Realview ICE，所以 debugger 从 multiICE, jlink 升级到 RealviewICE.
- 这里主要说软件的修改 , panda_os, mini_env.
- semihost 对于环境升级调试的帮助 .
- ICE 和 rvdebugger 已经在上次的介绍用介绍了 .

- 3，除了上述应用外，如果项目更换了新的 arm 处理器，环境部分需要做些修改．这里以 882 为例说明．见 "00:39 2010-7-2". "15:39 2010-2-4"

Emi bootloader

Cortex-A features

结合 VC0882 Cortex-A8 test plan

Cortex-A8 block diagram

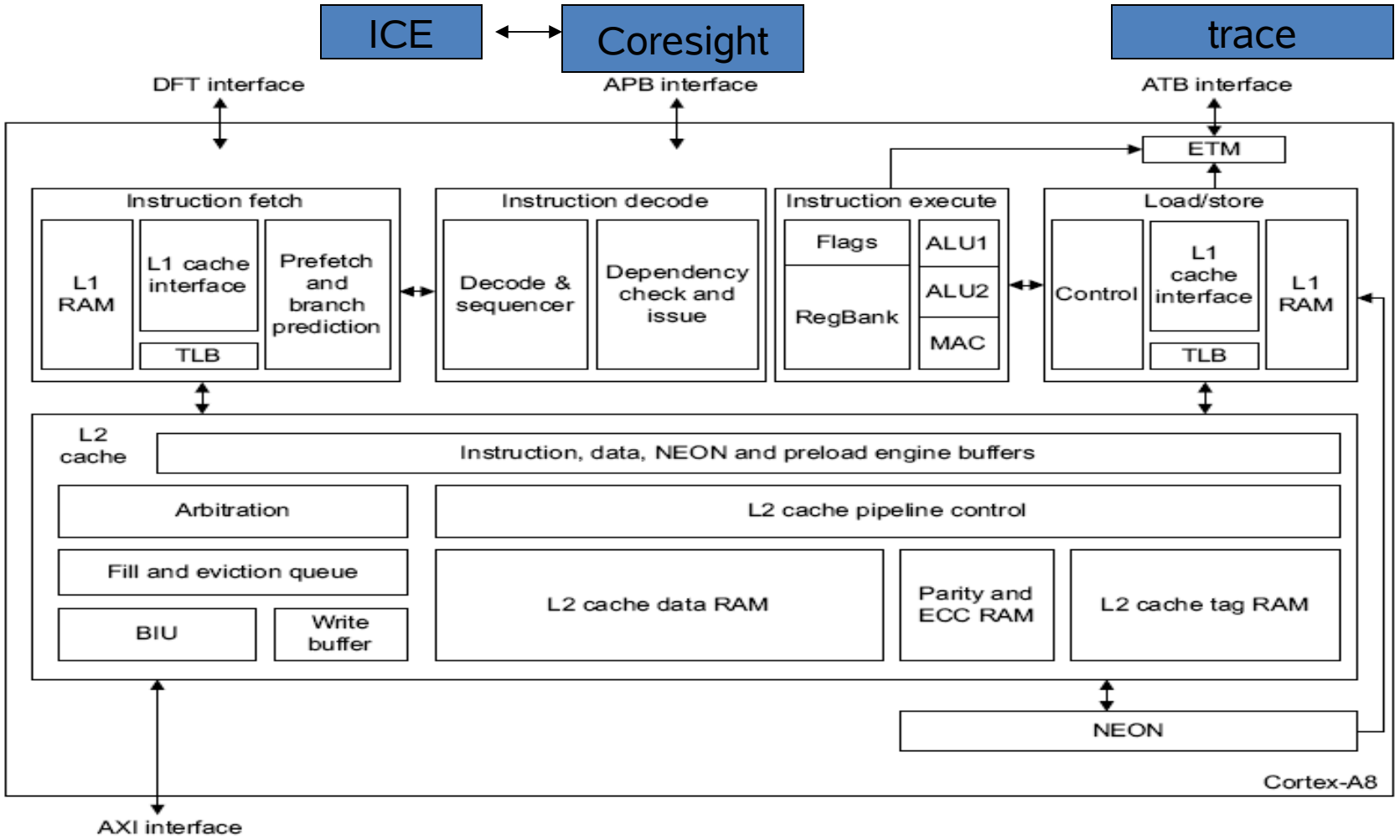


Figure 1-1 Cortex-A8 block diagram

cortex-a8 ecosystem

- ISA(armv7)，支持 thumb-2, thumbEE, NEON.
- ARM core(cortex-a8): 13 级双发射整数流水线 , L2cache, 64/128 bit AXI bus.
  - Cortex-A8 在 AXI 上的行为
- SOC：支持 GPU(OpenGL, OpenVG).
- Coresight：新的调试架构 .
- 对 OS 的支持

VIMICRO 中星微 Nasdaq: VIMC

Cortex-A

- Instruction set
- Memory
- Interrupt
- Debug (Coresight)
- Clock, reset, power
- Security: Trust Zone
- Multi-core(A5, A7, A9, A15)

Instruction set

- Arm, thumb, thumb2
  - 每次 arm 架构变化都会多出一些指令
  - 位操作命令 .
  - Memory barrier 命令 : isb, dsb
- Thumb2-EE
  - 加入解释性语言和脚本语言的速度
- **NEON** and vfp3
  - NEON 相当于半个独立的 dsp
  - 比之前 arm 指令集里面的多媒体指令功能更丰富
  - 如果只想加速浮点运算可以只用 vfp3，不配置 NEON.
    - Cortex-A8 没法选择，固定用 NEON
    - NVidia Tegra2 没有 NEON，据说请 arm 专门为其做 flash 优化 .
  - **NEON openmax demo**

pipeline

- Pipeline (Cortex-A8)
  - integer 13 stages
  - neon 10 stages
  - L2 cache 8 stages
- In-order and Out-of-order (Cortex-A8)
  - integer 是顺序执行的 (A9 可以乱序)
  - neon 可以乱序
- Multi issue
  - Cortex-A8 双发射
  - Cortex-A15 三发射，乱序发射

- Branch prediction
  - 预测错误的跳转会造成流水线 flush，这样会有 13cycle 的 penelty．打开对于循环性能的提升很大 **(VC0882 memcpy test 和 bootloader 仿真都有证明).**
  - 如果打开了分支预测，就不能通过跳转来 flush pipeline．需要使用 pipeline flush 或 isb.
- 不建议使用 FCSE，推荐 ASID 方式

Memory--cache

- Cache 基本原理
- Cache sram
  - **VC0882 cache 验证 : memcpy**
- Max 7 level
  - 一套代码，可以做多级 cache 的一致性操作 .
  - 没有提供类似 arm926ej-s 中的一条指令清 dcache 的命令 .
- L1 cache Harvard
  - i$, d$, tlb
  - Cortex-A8: 16k 或 32k, 4 路组相联 .
- L2 cache unified. 可以选择有无 .
  - Cortex-A8: 128k to 1M, 8 路组相联 . L1 dcache 与 L2 dcache 无关，但 L1 icache 是 L2 icache 的子集
  - NEON 默认走 L2 cache. 如果希望用 L1 cache 需要单独设置 cp15.
- Cache maintenance
  - 不支持单独的 invalidate? \todo check.
  - 代价
  - Point of coherence, Point of unification (next page)
- Cache fill 使用 AXI wrap 操作 : critical word first.
- L1 icache 是 VIPT, L1 dcache 和 L2 cache 是 PIPT----PIPT 避免 page coloring.
- Cache timing: L1 cache 与 cpu 同频 , L2 cache 1/3 cpu 频率 (VC0882 情况).
- on/off sequence

PoC and PoU

- The points to which a cache maintenance operation can be defined differ depending on whether the operation is by MVA or by set/way:
- For set/way operations, and for All (entire cache) operations, the point is defined to be to the next level of caching.
- For MVA operations, two conceptual points are defined:
- Point of coherency (POC)
- For a particular MVA, the POC is the point at which all agents that can access memory are guaranteed to see the same copy of a memory location. In many cases, this is effectively the main system memory, although the architecture does not prohibit the implementation of caches beyond the POC that have no effect on the coherence between memory system agents.
- Point of unification (POU)
- The PoU for a processor is the point by which the instruction and data caches and the translation table walks of that processor are guaranteed to see the same copy of a memory location. In many cases, the point of unification is the point in a uniprocessor memory system by which the instruction and data caches and the translation table walks have merged.
- The PoU for an Inner Shareable shareability domain is the point by which the instruction and data caches and the translation table walks of all the processors in that Inner Shareable shareability domain are guaranteed to see the same copy of a memory location. Defining this point permits self-modifying code to ensure future instruction fetches are associated with the modified version of the code by using the standard correctness policy of:
- 1. clean data cache entry by address
- 2. invalidate instruction cache entry by address.
- The PoU also enables a uniprocessor system which does not implement the
- Multiprocessing Extensions to use the clean data cache entry operation to ensure that all writes to the translation tables are visible to the translation table walk hardware.

Memory--MMU

- Mmu 基本原理： 页表 . Tlb.
- add 16MBytes super section. Remove 1k tiny page. 使用更大的 section 能改善 tlb 命中率 .
  - Supersections Consist of 16MB blocks of memory.
  - Sections Consist of 1MB blocks of memory.
  - Large pages Consist of 64KB blocks of memory.
  - Small pages Consist of 4KB blocks of memory.
- strongly-ordered, device, normal
  - 解释三个属性的含义 . 与原来 CB 的关系 .
  - kongyingqi 就因为这个属性发现了一个 IC bug: 配寄存器的时候没有配完就返回 ok 了 .
  - 26 个 Outstanding transactions
- XN, 表示这个区域不能被预取或执行 :
  - Any region of memory that is read-sensitive must be marked as Execute Never, to avoid the possibility of a speculative prefetch accessing the memory region. For example, any memory region that corresponds to a read-sensitive peripheral must be marked as Execute Never.
- nG: =0 表示是 OS kernel 空间 , =1 表示是进程空间 , for ASID.
- Inner and Outer Sharable: 多核才有意义 , 略过 .
- Ttb: 两个 ttb
- on/off sequence: 地址不能跳变 .

MMU 页表

- 1 级页表和 2 级页表
- Tlb 与页表不同 : tlb 不分级 .
- \todo 贴一份完整 mmu 属性

Address extension(A15): 访问超过4G 的地址空间 .

Interrupt

- PLE 的 DMAIRQ 中断
- PLE 的 DMAEXTERRIRQ 中断
- PLE 的 DMASIRQ 中断
- Performance Monitor 中断

Debug

- Debug register
- Coresight validation
- **Lock accesss**
  - 需要解锁才能访问调试寄存器
  - setreg @CS_LOCKACCESS=0xC5ACCE55

Clock, reset, power

- arm cpu clk 和 AXI clk 之间什么关系？ 整数倍
- arm reset: 共有 5 个 reset，包括单独 reset debug logic，其它 logic, neon，详见 cortex-a8 trm p363.
- 882 设计简化为可以选择 reset cpu 时是否 reset debug logic.
- power control: 882 中 A8 是一个 power domain. 只能通过是否给 clock 控制功耗 . neon, vfp, etm.
- **wfi and power control**.
  - wfi 唤醒与屏蔽 irq 无关 .

Security: Trust Zone

Multi-Core

- 一个 SCU 下面支持 4 个 Core
- 借助 AMBA4 的 CCI (cache coherence interconnect) 可以支持多余一个的 SCU. 针对 Cortex-A15 的多核 .

其他需要注意的

- 11, cp15 里面其它需要注意的寄存器
- 1), "3.2.26 c1, Auxiliary Control Register" 里面有一些高级控制选项 .
- (1), "speculative accesses"? AXI
- 2), "c12, Interrupt Status Register". fiq, irq, external abort.
- 3), "c2, Translation Table Base Control Register"
- [5] PD1 Specifies occurrence of a translation table walk on a TLB miss when using Translation Table Base
- Register 1. When translation table walk is disabled, a section translation fault occurs instead on a TLB miss:
- 0 = The processor performs a translation table walk on a TLB miss, with secure or nonsecure privilege appropriate to the current Secure or Nonsecure state. This is the reset value.
- 1 = The processor does not perform a translation table walk. If a TLB miss occurs with Translation Table Base Register 1 in use, the processor returns a section translation fault.
- [4] PD0 Specifies occurrence of a translation table walk on a TLB miss when using Translation Table Base Register 0. When translation table walk is disabled, a section translation fault occurs instead on a TLB miss:
- 0 = The processor performs a translation table walk on a TLB miss, with secure or nonsecure privilege appropriate to the current Secure or Nonsecure state. This is the reset value.
- 1 = The processor does not perform a translation table walk. If a TLB miss occurs with Translation Table Base Register 0 in use, the processor returns a section translation fault.
- [3] - Reserved. UNP, SBZ.
- [2:0] N Specifies the boundary size of Translation Table Base Register 0:
- b000 = 16KB, reset value
- b001 = 8KB
- b010 = 4KB
- b011 = 2KB
- b100 = 1KB
- b101 = 512B
- b110 = 256B
- b111 = 128B.
- 4), 分析系统出错寄存器 : DFSR, DFAR, I...

Arm 异常与 monitor 异常对比

Exception offset	Exception that is vectored at that offset from:	
	Monitor exception base addressa	**Base address for all other exceptions**
0x00	Not used	Not used
0x04	Not used	Not used
0x08	Secure	Monitor
0x0C	Prefetch	Abort
0x10	Data	Abort
0x14	Not used	Not used
0x18	IRQ	(interrupt)
0x1C	FIQ	(fast

Arm coprocesssor tool

- Read arm coprocessor register
- D:\VC0882\document\arm\coprocessor_tool

Cs and jtag 验证

- \todo: 与 VC0882(Cortex-A8) coresight, arm11 jtag 验证相关部分留到 arm 验证部分详细说明 .

Arm Cortex-A 让人想起了 Intel Pentium

0x413FC082

	ARM	INTEL	MIPS
编号	Cortex	Pentium	
流水线	超标量	超标量	超标量？
多媒体指令	NEON	MMX	ASE
数据宽度	64bit	64bit	
新一代指令压缩技术	Thumb-2		microMIPS

Thumb-2 architecture delivers a 30% reduction in code size while maintaining the same performance of 32-bit ARM code.

A code compression ISA that maintains 98% of MIPS32 performance while reducing code size by 35%, translating to significant silicon cost savings

Armv7 与 armv5 差异

- Armv7 与 armv5 相比变化很大．参 ARM_ARM_7AR.
  - 大端小端变化 (armv7, armv5), 见下页.
  - 不允许 sp, pc 作为通用寄存器使用，
  - Multi load/store 中不建议包含 sp: The SP can be in the list in ARM code, but not in Thumb code. However, ARM instructions
  - that include the SP in the list are deprecated.
  - Armv7 允许对 normal memory 非对齐访问
  - Dataabort: 有 precise/imprecise dataabort 两种．
  - cp15 的寄存器实际数量都多了几倍，从 16 个变为 83 个．寄存器名字还是 c0-c15. 新增功能用 op2 和 CRm 区分．
  - the base restored Data Abort model: 发生 dataabort 后基址寄存器会恢复到 dataabort 前的值．

Table H-2 Bytes accessed by aligned accesses in endian formats

Memory access:		Bytes accessed in endian format:		
Size	**Address**	**LE**	**BE**	**BE-32**
Doubleword	A	ZYXWVUTS	STUVWXYZ	VUTSZYXW
Word	A	VUTS	STUV	VUTS
Word	A+4	ZYXW	WXYZ	ZYXW
Halfword	A	TS	ST	VU
Halfword	A+2	VU	UV	TS
Halfword	A+4	XW	WX	ZY
Halfword	A+6	ZY	YZ	XW
Byte	A	S	S	V
Byte	A+1	T	T	U
Byte	A+2	U	U	T
Byte	A+3	V	V	S
Byte	A+4	W	W	Z
Byte	A+5	X	X	Y
Byte	A+6	Y	Y	X
Byte	A+7	Z	Z	W

Armv7 新 feature

- Thumb-2: 16, 32bit 指令混合，提高代码密度 (同时也提高了 cache 命中率).
- Thumb-EE: 加速解释型语言 (java) 和脚本运行执行速度
- NE
  - 10 级流水线
  - the Cortex-A8 processor can decode MPEG-4 VGA video (including dering, deblock filters and yuv2rgb) at 30frames/sec at 275 MHz, and H.264 video at 350 MHz.
  - 可以用于支持使用范围少的视频格式 .
- 最多支持 7 级 cache.

Android éclair(2.0) 开始使用 ThumbEE 加速 java 执行，根据 0xdroid(http://code.google.com/p/0xdroid/issues/detail?id=56) 数据，性能提高 1.7 倍 .

Cortex-a8

- IEM

Cortex-a8/armv7 对 OS 的支持

- WinCE6 支持 TI OMAP3530，没有使用 Cortex-a8 任何新 feature.
- Android 从 Eclaire 开始对 Cortex-a8 新 feature 有很好的支持 . Java 性能提高，支持 Flash 10 播放 (之前 arm9 是 flash lite3).
- Ubuntu 对 cortex-a8 支持也很好 . Utuntu 支持的第一个 armv7 是 Marvell 的 armv7 处理器 (不是 cortex-a8).

Arm Powered SOC

- 国内外基于 Arm 架构的热门 SOC 对比 (rk2918, amlogic, via, renses, qualcomm, ti, samsung).

reference

- Cortex-A8 TRM:
  - D:\VC0882\document\arm\Cortex-a8\DDI0344J_cortex_a8_r3p2_trm.pdf
- ARM V7-AR ARM:
  - D:\VC0882\document\arm\Cortex-a8\DDI0406B_arm_architecture_reference_manual.pdf
- Zhangjian Arm 工作笔记
  - \\10.0.2.36\sqmshare\share\zhangjian\log200920102011_arm_vimicro
  - 这个文档只是笔记，没有整理．索引见”　17:38 2011-12-12”.
- VC0882 项目早期的 arm 介绍：
  - D:\VC0882\document\VC0882 环境和 Cortex-a8 介绍 .ppt
  - D:\VC0882\document\VC0882 environment freshman guide.doc