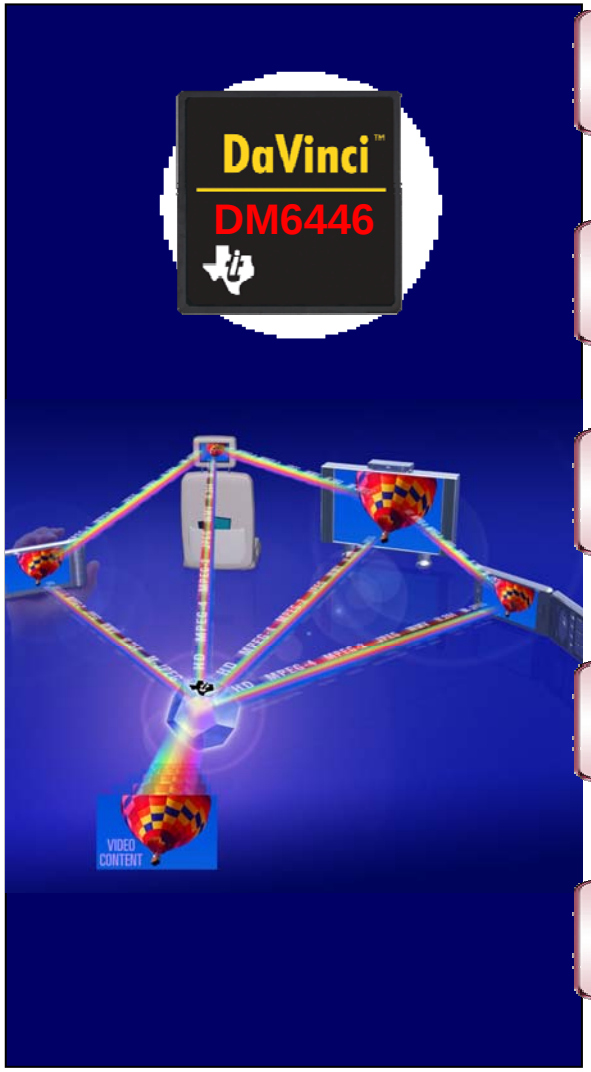
TMS320DM6446 Digital Media Processor
达芬奇技术培训

议题



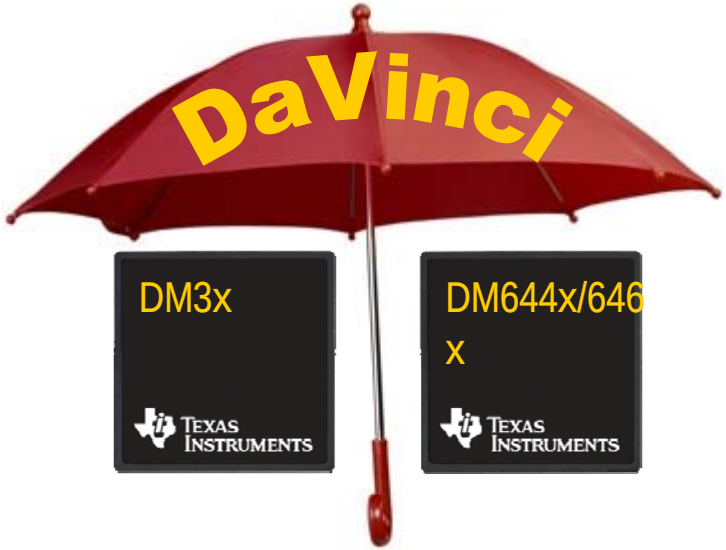
- 达芬奇技术概述
- 达芬奇软件架构
- 应用层（**ARM**）
- 信号处理层（**DSP**）
- **DM6446**开发工具：**DVEVM**和**DVSDK**

达芬奇技术概述

介绍

- 什么是SoC?
  - 通用定义
    - SoC，即片上系统是指集成了所有计算机或电子系统的一些组件（模块）在一个单的集成电路芯片上。
  - 不同的厂商对它的定义有所不同
    - 除了上面的通用定义外，TI对SoC定义也有些不同。比如DSP处理器，协处理器和/或加速器等这样的一些组件集成在一个单芯片上也被称为SoC，这些协处理器或加速器的主要作用是加强DSP的处理性能。

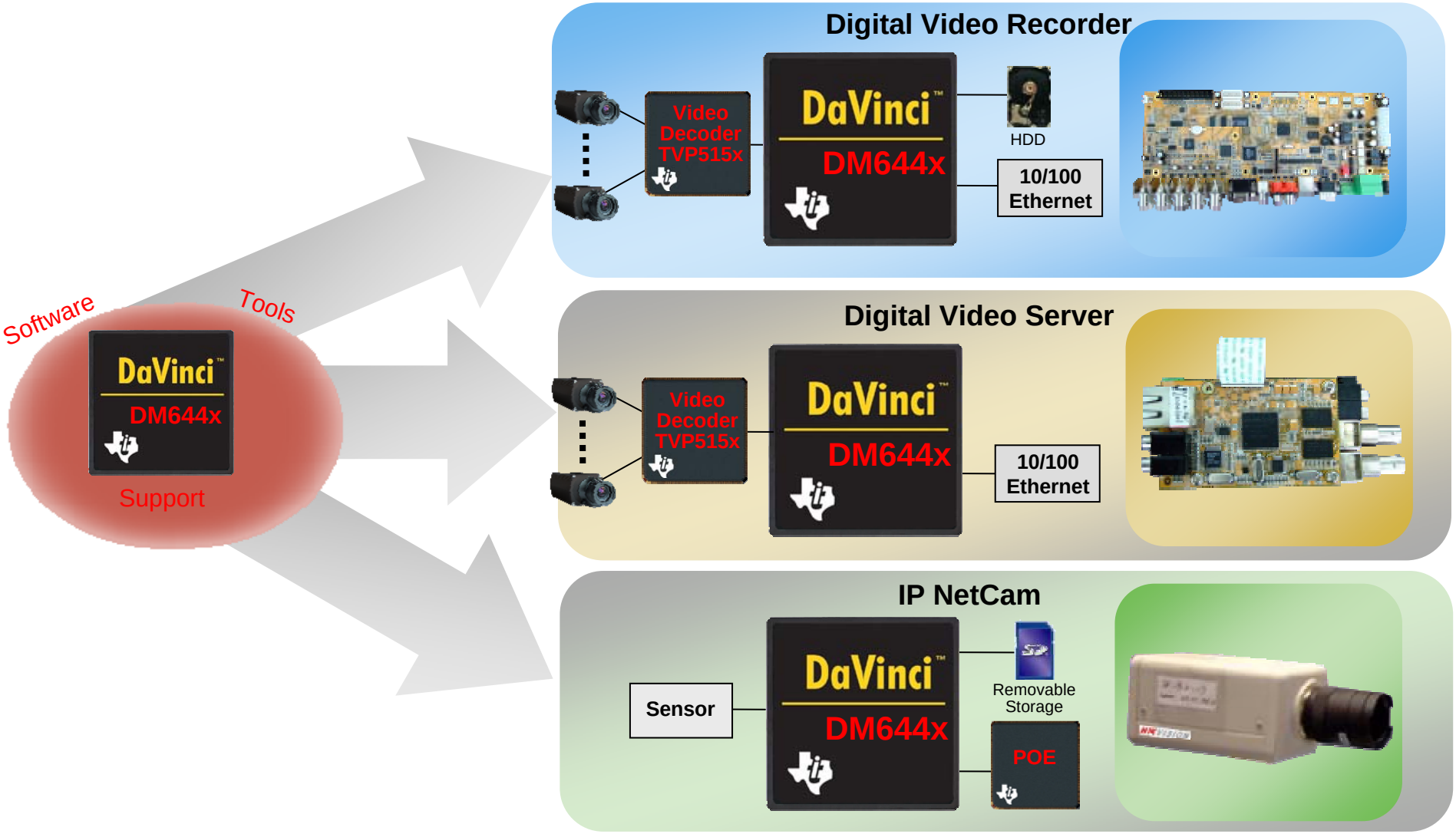
达芬奇技术定义



- 什么是达芬奇技术?
  - 芯片
  - 软件
  - 开发工具
  - 软件支持

达芬奇DM6446典型应用

DaVinci™ DM644x DVR/DVS/IP NetCam应用



TMS320DM6446硬件概述

TMS320DM6446处理器系统结构

特点

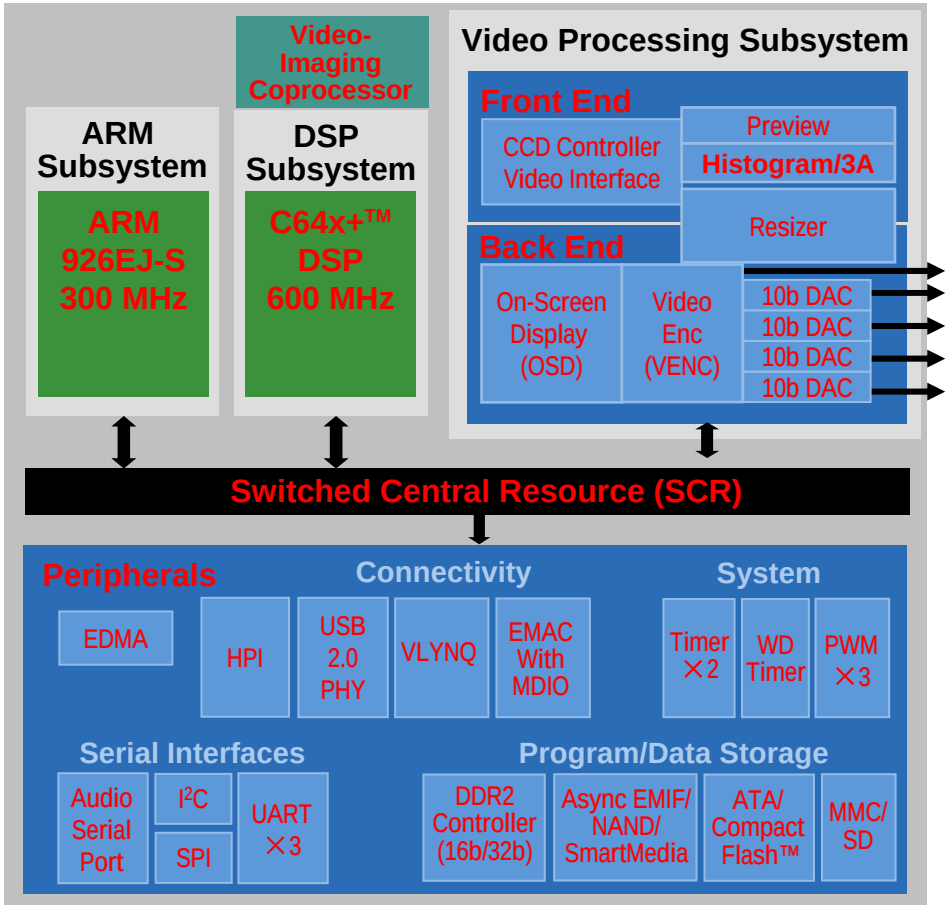
- 处理器核
  - 300MHz ARM926EJ-S™ (MPU) 内核
  - 600MHz TMS320C64x+™ DSP 内核
- 存储资源
  - 片内L1/SRAM: 112 KB DSP, 40 KB ARM
  - 片内L2/SRAM: 64 KB DSP
- 外设
  - 视频处理子系统：VPSS
    - 视频前段– 缩放引擎, 图像处理引擎, 16-bit数字输入
    - 视频后端 – 集成OSD, 4个视频DACs, 24-bit数字RGB输出

优点/性能

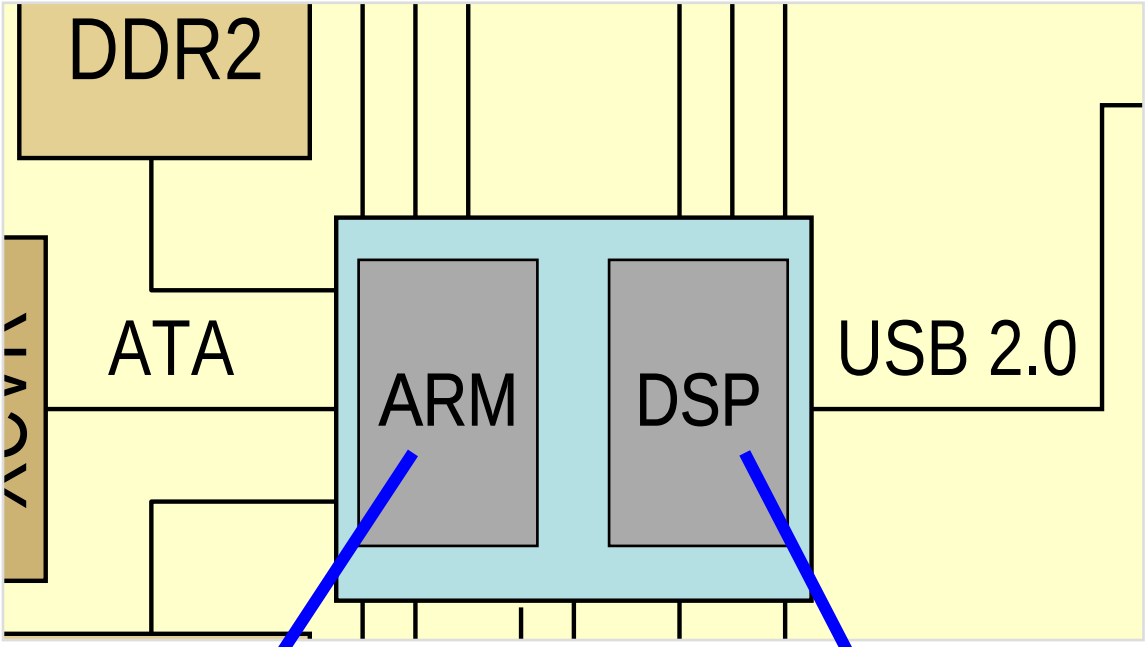
- DM6446的这种高集成度的SoC使得客户可以以较低的成本快速开发自己的产品。
- 视频的编解码能力
  - H.264 BP D1 编/解码
  - MPEG-2 MP SD 解码, MPEG-4 D1 SD 解码

应用

- 视频会议, 可视电话, 视频监控, IP机顶盒等



处理器（ARM+DSP）



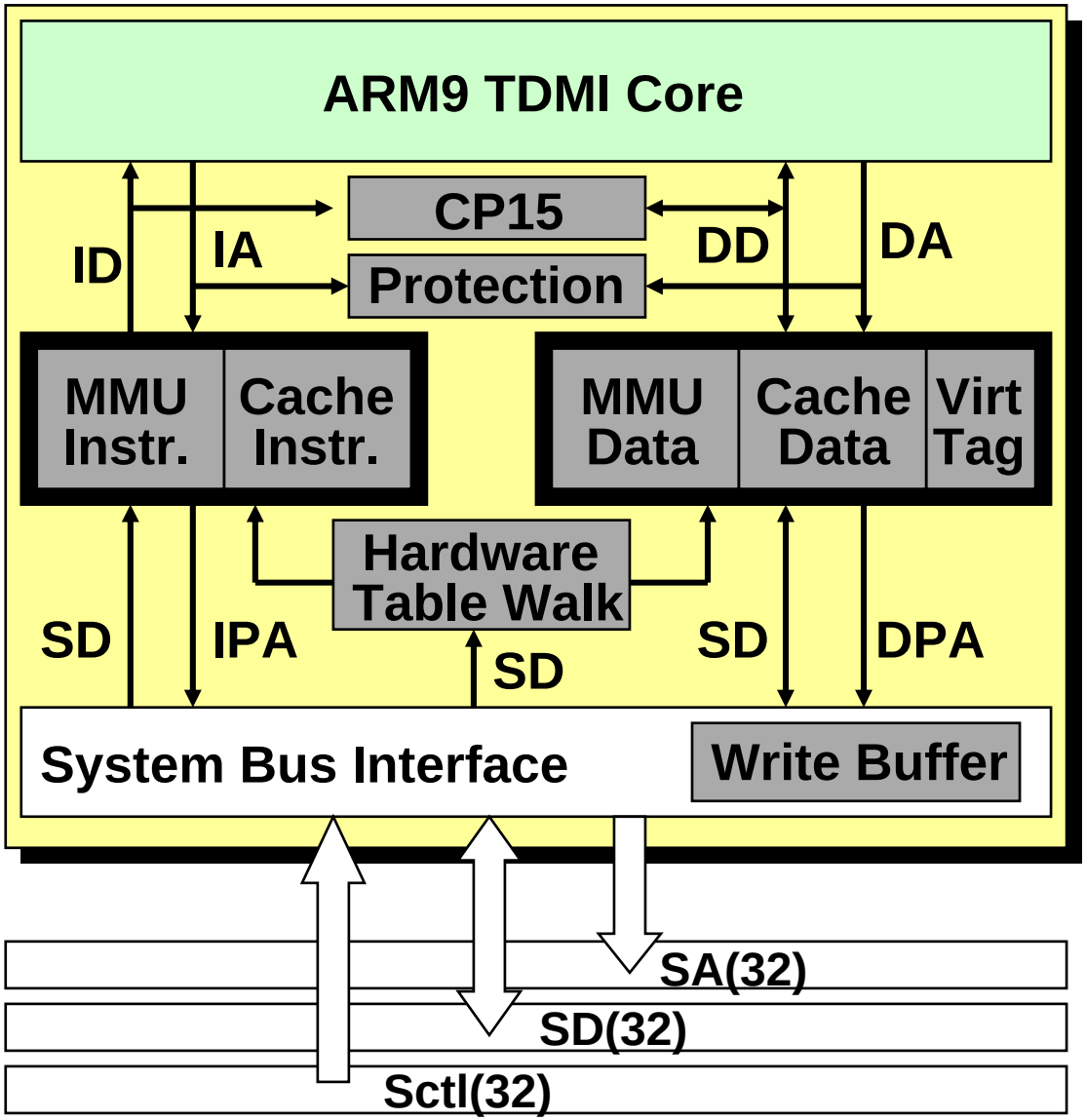
Linux

用户界面, OSD屏幕菜单显示，
设备驱动

DSP/BIOS

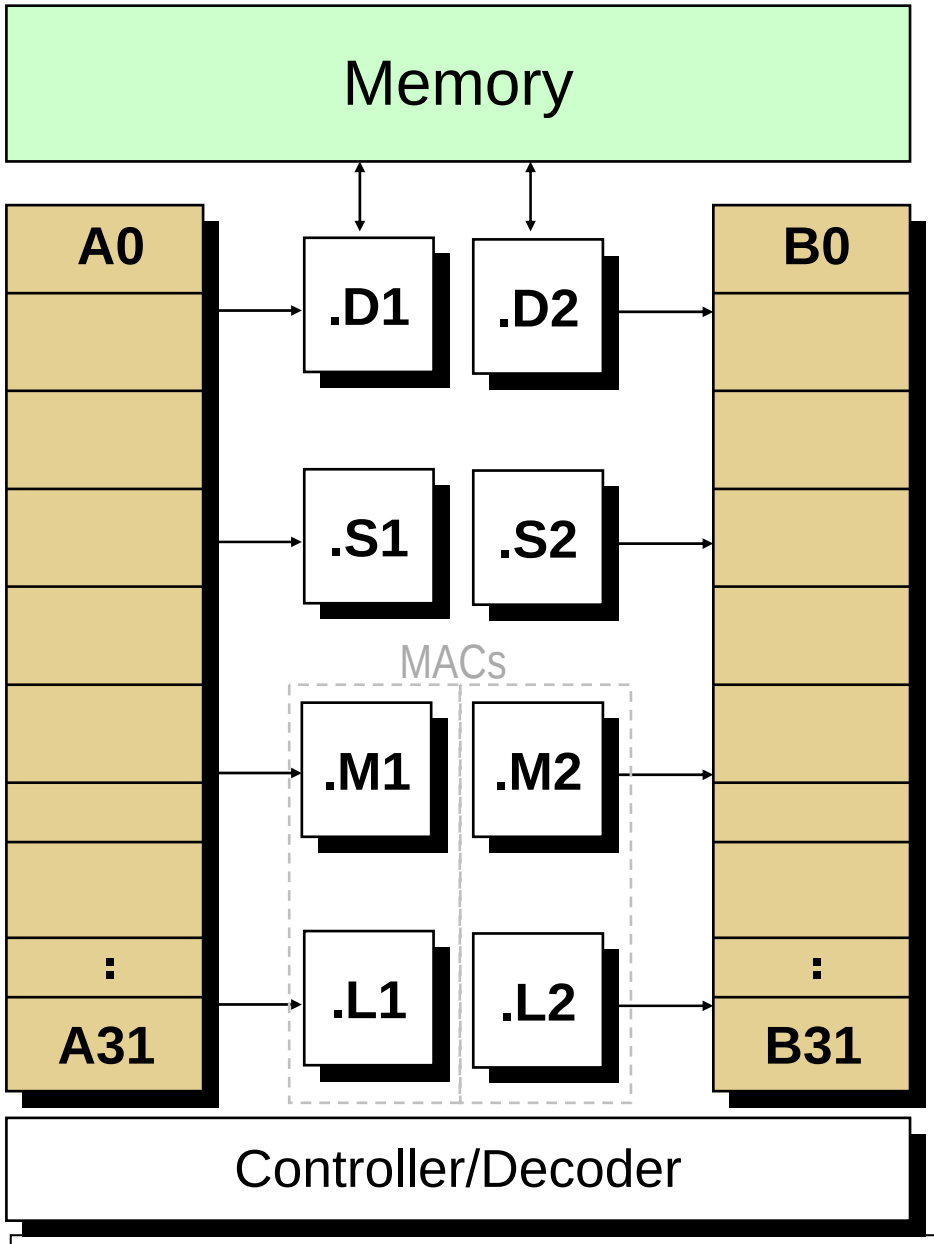
视频处理:编码，解码等
音频处理:编码，解码等

ARM926内核



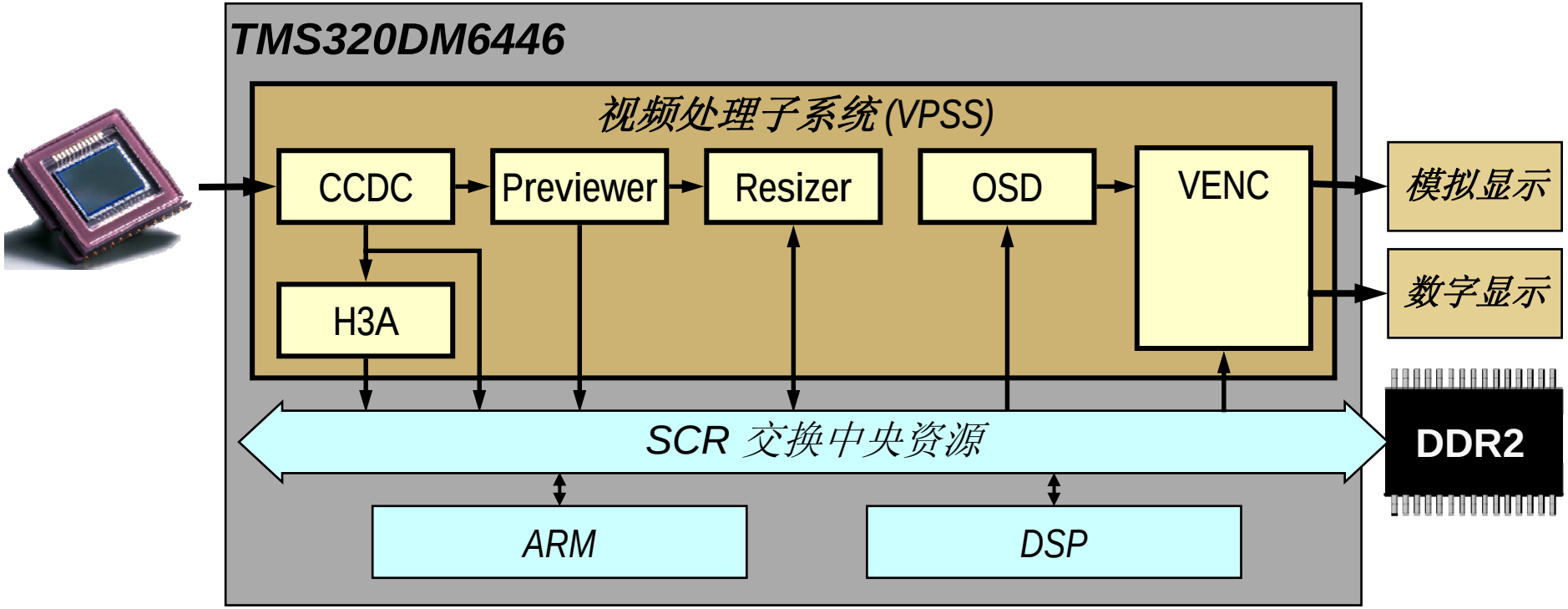
- ARM926EJ-S(ARMV5T核)
- 最大频率: 300 MHz
- 支持多个操作系统: Linux, WinCE等
- 哈佛结构，带有5级流水线
- 指令集包含了一个增强性的16x32位的硬件乘法器
- 能够执行单周期的MAC操作

‘C64x+ CPU 结构

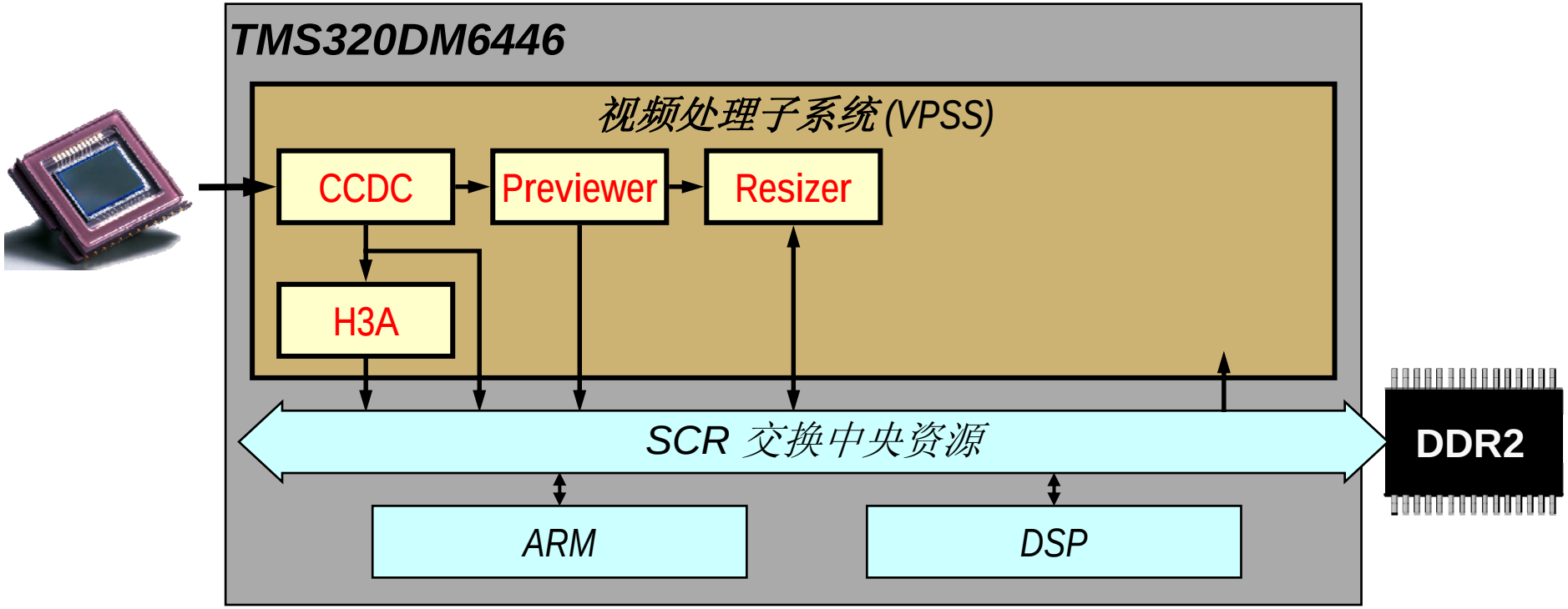


- 8个功能单元：M1,L1,D1,S1,M2,L2,D2,S2
  - 其中乘法 (.M) 和 ALU (.L)提供了最大 8 MACs/cycle (8x8 或 16x16)
  - .L和.S主要是执行一些算术，逻辑，分量的功能
  - .D主要是从Memory中load数据到寄存器文件或把结果从寄存器文件存放到Memory中
- 2个寄存器文件A和B，每个寄存器文件包含32个32位的寄存器。

视频处理子系统 (VPSS)

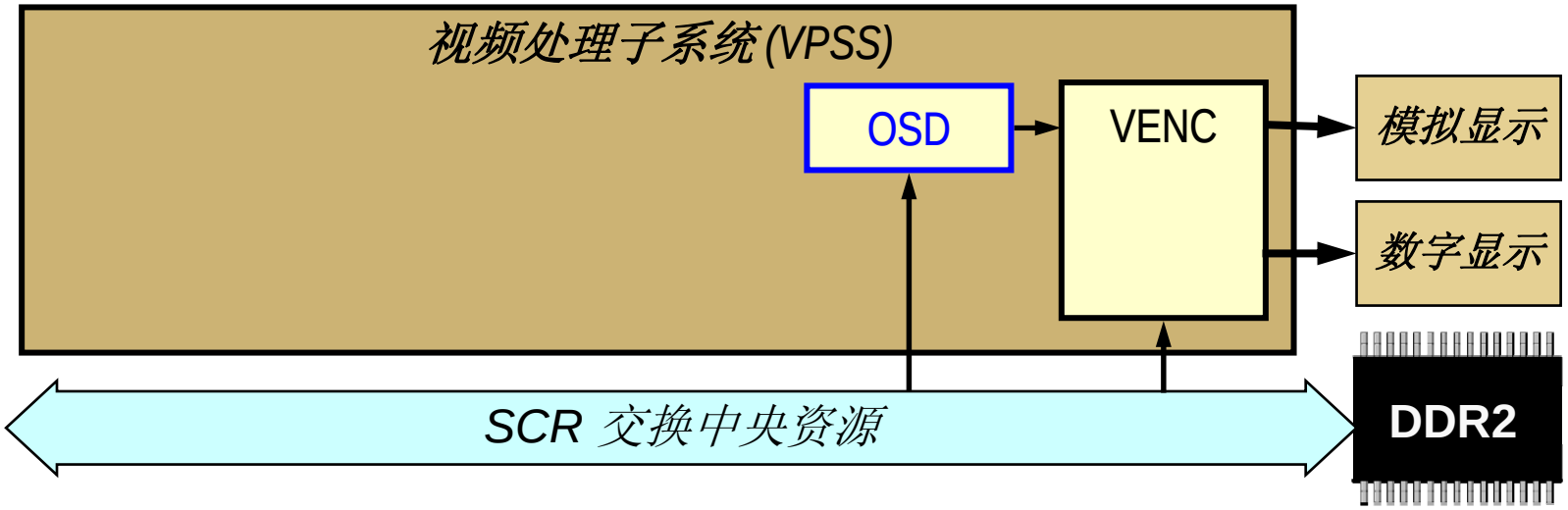


视频处理前端 - VPFE



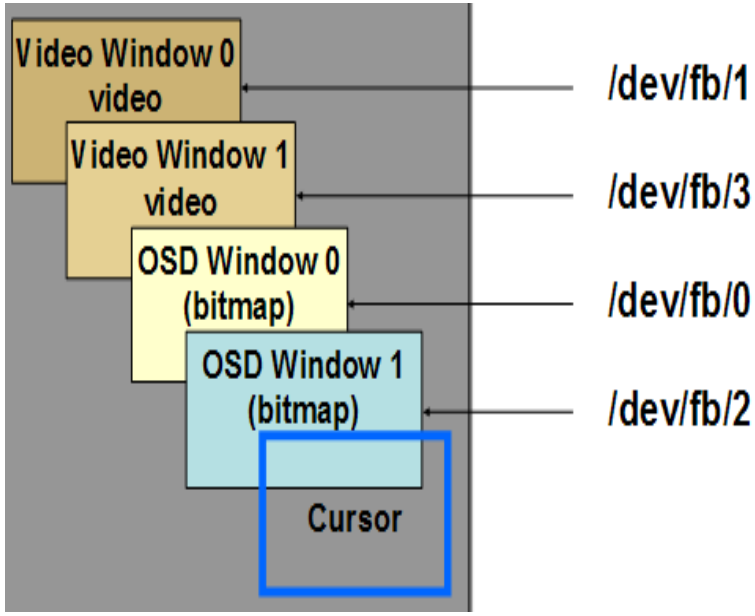
- CCDC - Charge Coupled Devices Controller 电荷耦合设备控制器
- Previewer – 预览引擎
- Resizer – 缩放引擎
- H3A – 自动曝光，自动对焦，自动白平衡

视频处理后端 – VPBE - OSD

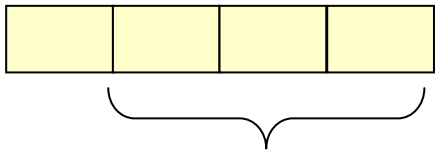


硬件实现的On-Screen Display (OSD)

- 2 个独立的视频窗口：Video Windows 0/1
- 2 个独立的OSD窗口：OSD Windows 0/1
  - 其中OSD Windows 1可配置成属性窗口用来控制视频窗口与OSD Windows 0的混合，即透明度
- 1 个矩形指针窗口
- 同时支持1 个背景窗颜色



OSD 属性窗口（OSD Windows 1 /dev/fb/2）



8级混合

000: 00.0%, 100% Video

001: 12.5%, 87.5% Video

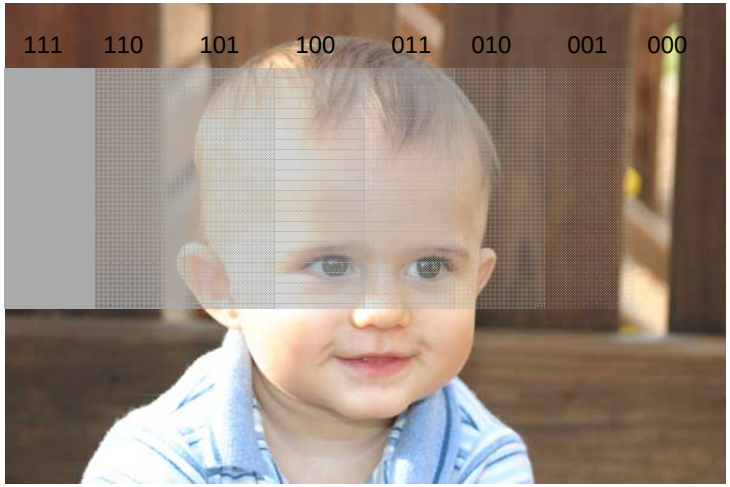
010: 25.0%, 75.0% Video

…

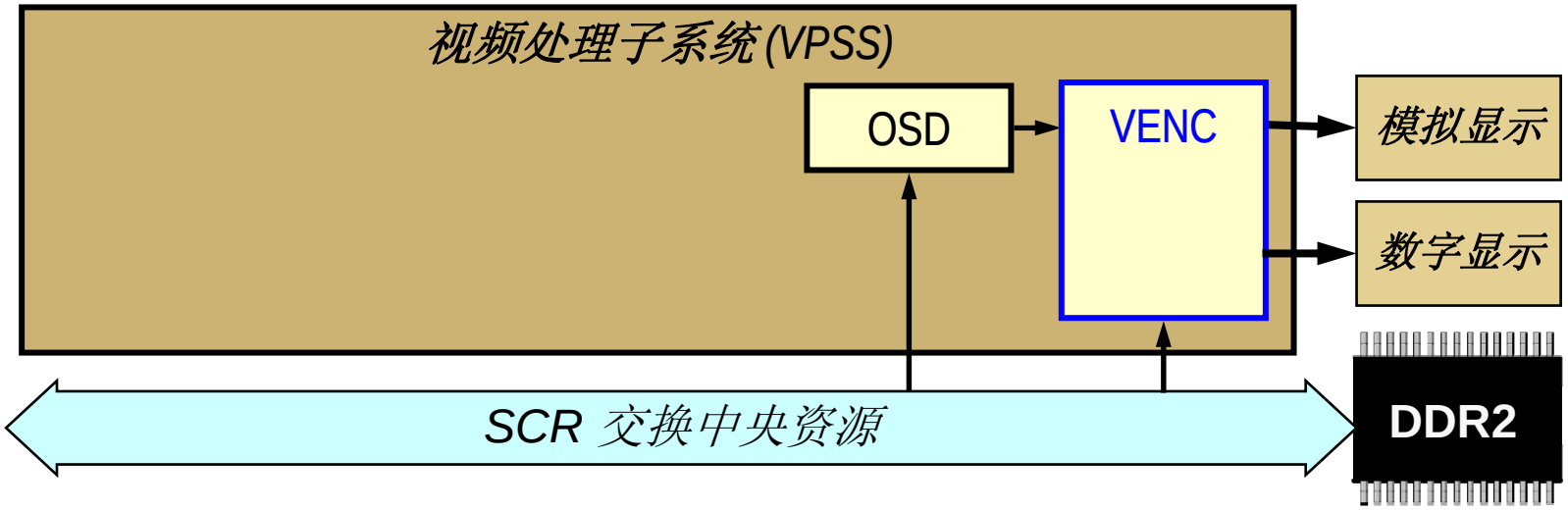
110: 75.0%, 25.0% Video

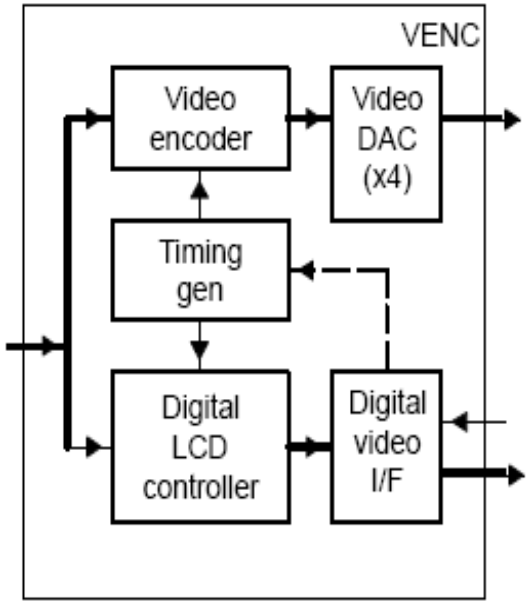
111: 100%, 00.0% Video

Cat /dev/zero /dev/fb/2



视频处理后端 – VPBE - VENC





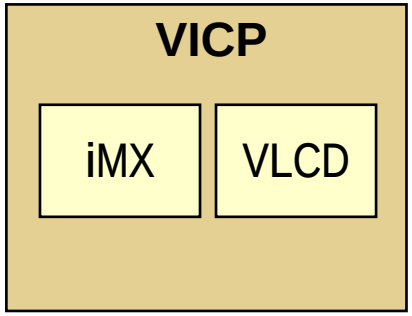
模拟输出:

- 4个54Mhz 10位的DACS
- 复合NTSC/PAL视频（1DAC）
- S-Video(2 DAC)
- 分量(YPbPr) or RGB(3 DAC)

数字输出:

- Max Pixel clock rate: 75 Mhz (720p and 1080i use 74.25 Mhz)
- 8/16 bit YUV
- 达到24-bit RGB
- BT.656 NTSC/PAL

VICP: 视频/影像协处理器



iMX (Imaging Extension)

- 相当于一个**300MHz**的**Encoder**
- 色域空间转变
- 过滤
- 动态评估

VLCD* (Variable Length Codec Decoder)

- 相当于一个**300MHz**的**Decoder**
- **Huffman**哈弗曼编解码

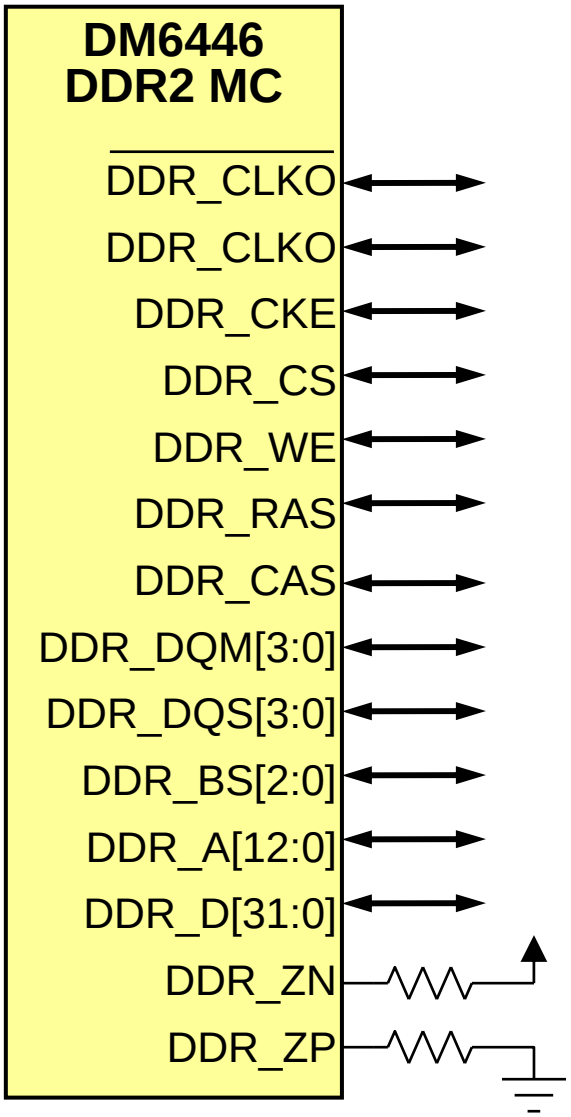
DDR2存储控制器

DDR2 memory controller

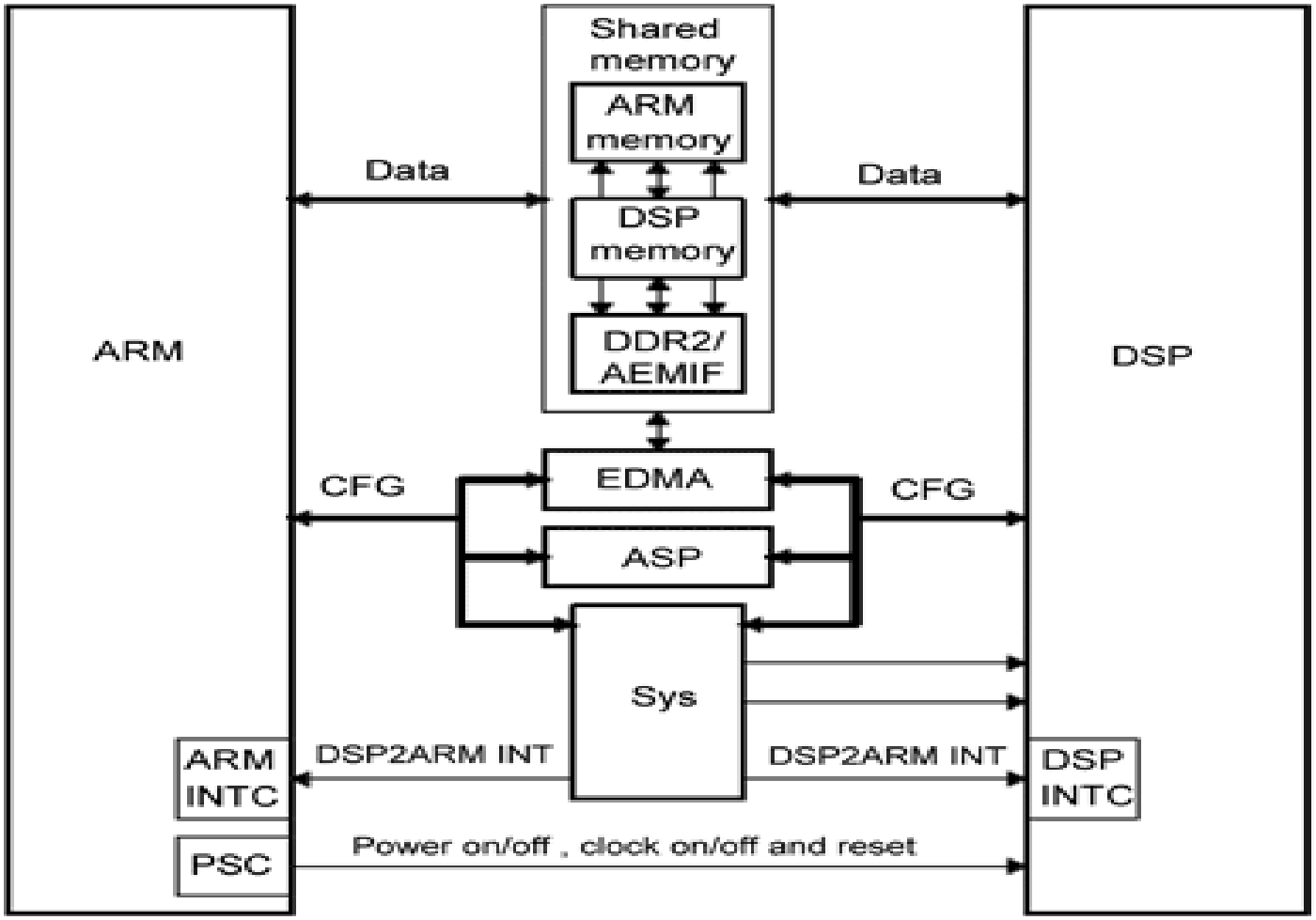
- 256 Mbyte的存储空间
- 16/32-bit数据总线
- Sequential burst length of 8
- 页大小: 256, 512, 1024, 2048

DDR2 设计

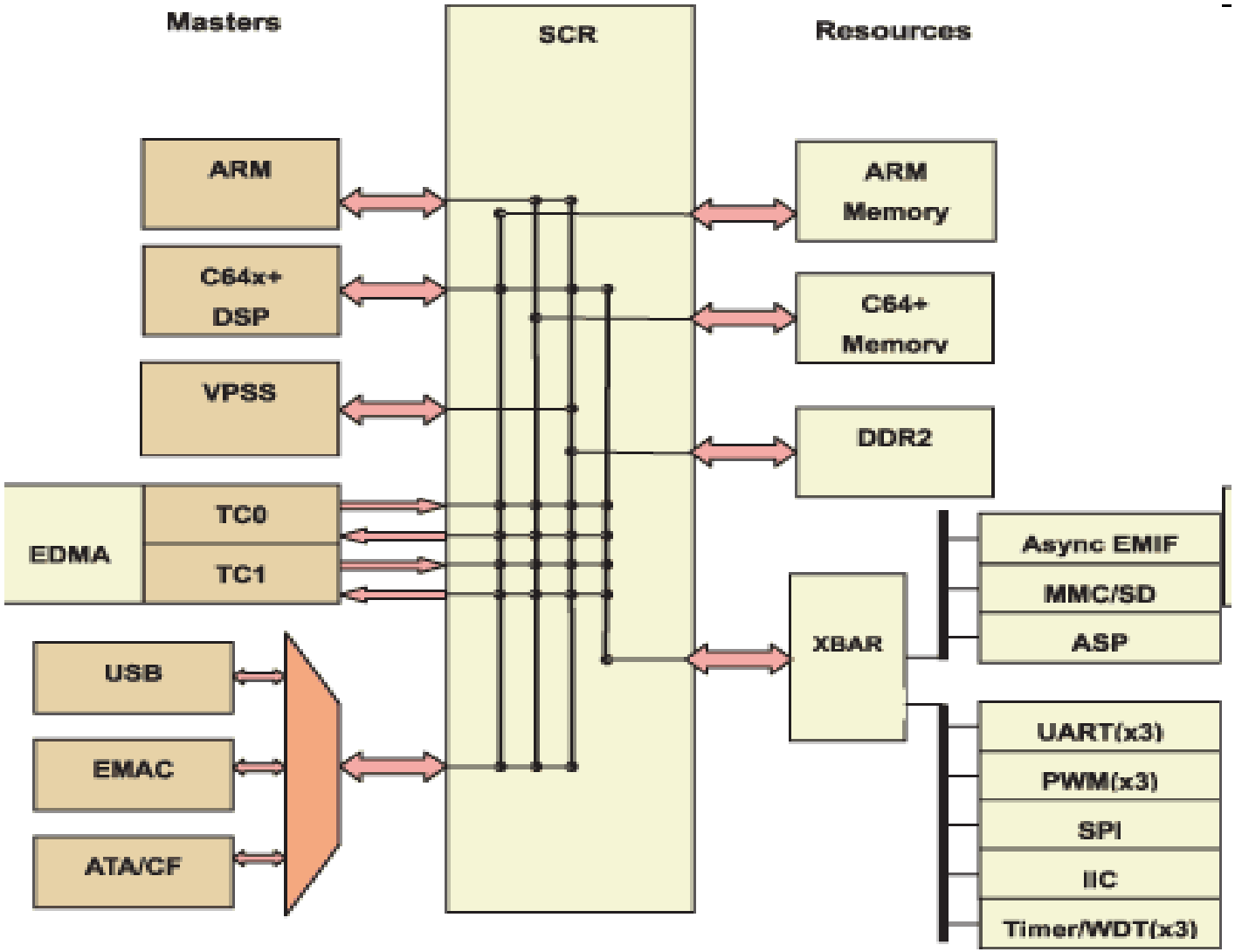
- 信号时序
- 信号完整性
- U-boot在启动时设置DDR2 控制器的参数，包括始终频率，总线宽度



ARM，DSP共享的资源



DMSoC交换中心资源（SCR）

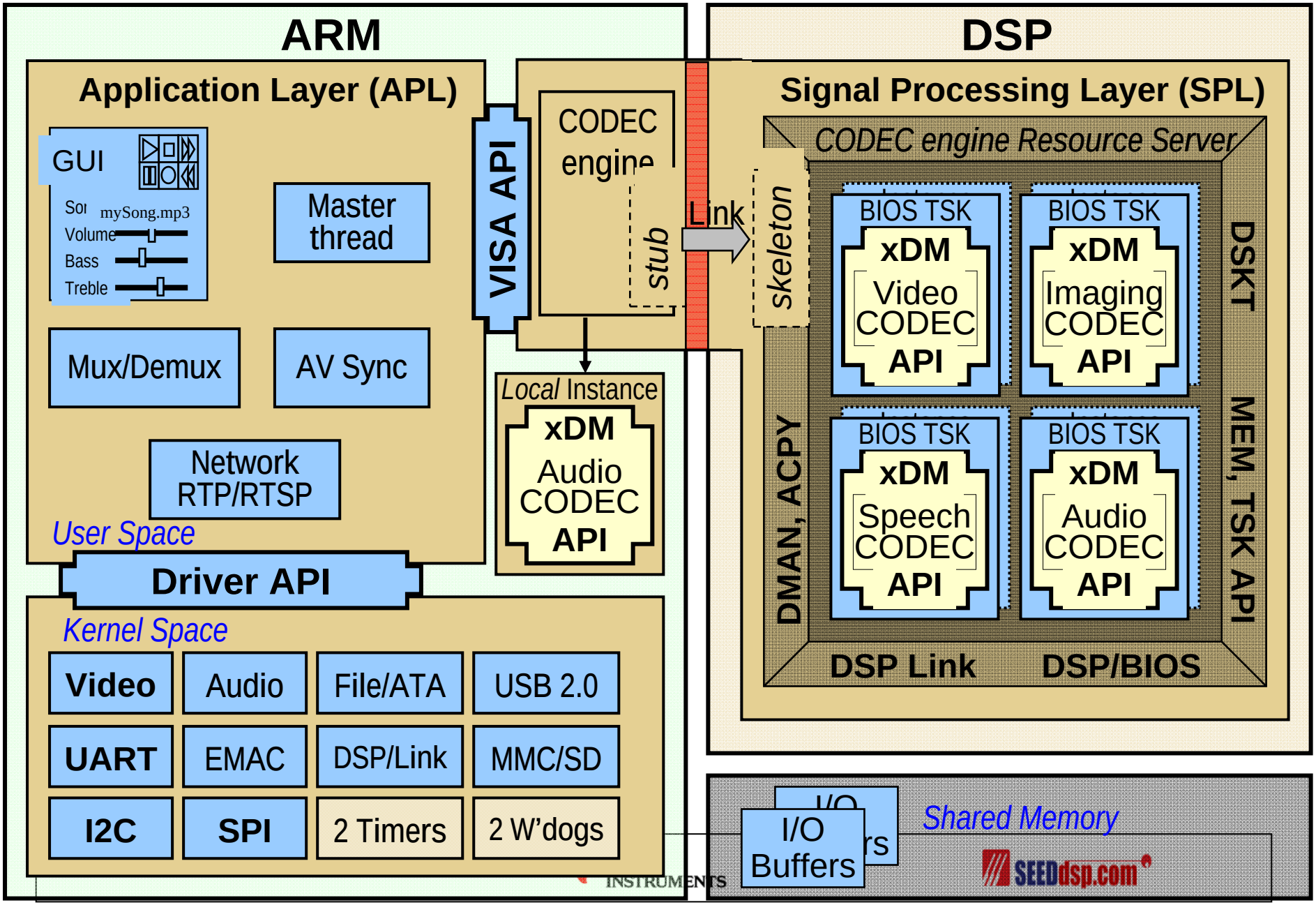


达芬奇DM6446初始化流程

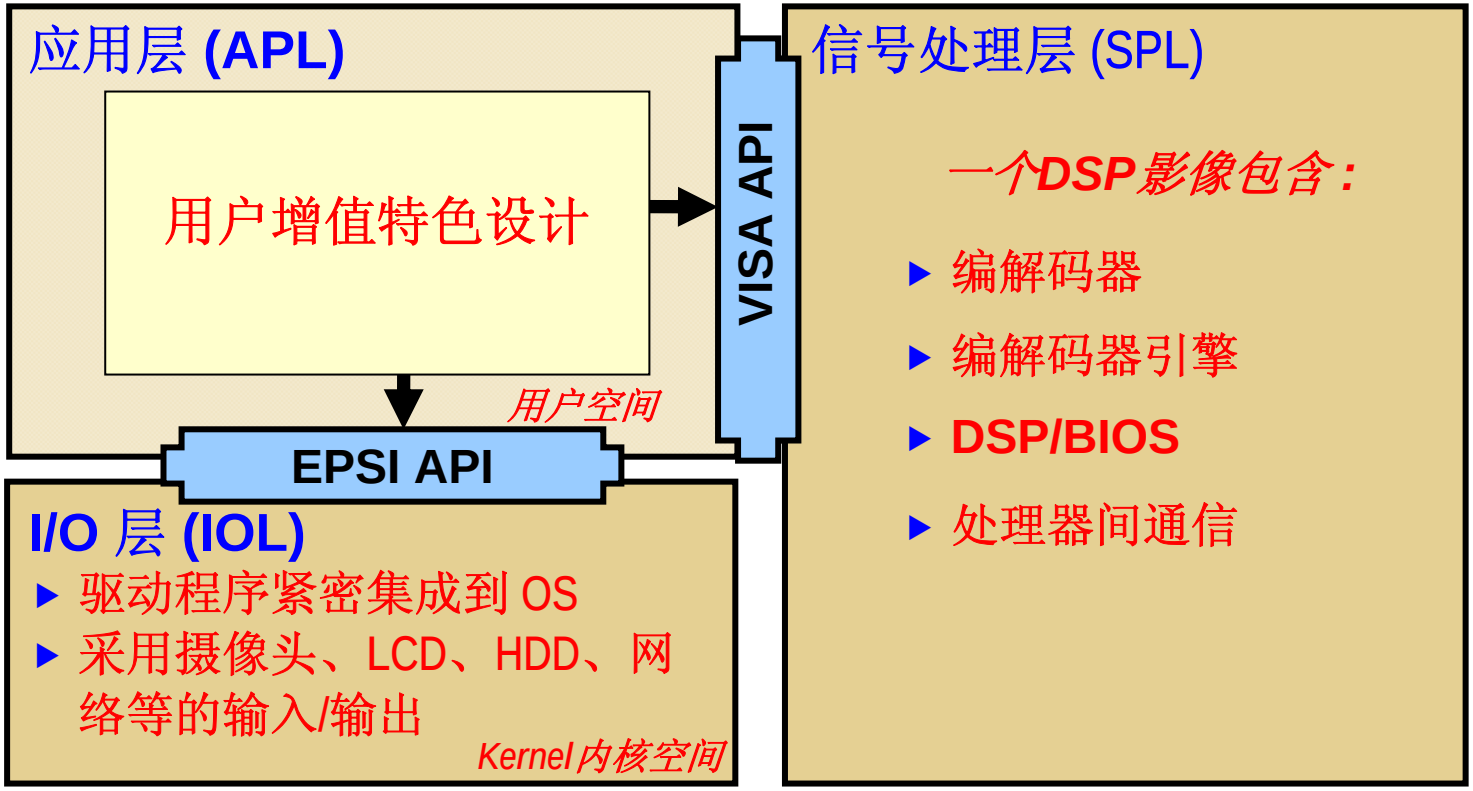
- RBL阶段（ARM ROM Boot Loader）
  - DM6446有4种启动方式：NAND Flash，EMIFA（NOR ），HPI和UART。
  - 系统复位后，保存在片内ROM的RBL程序开始运行，RBL程序根据BTSEL[1:0]管脚的电平来判断相应的启动方式。
  - 如果是BTSEL=00表明是NAND启动方式，RBL程序便从外接NAND Flash中读取UBL的数据到内部RAM中（UBL最大可达14K），然后转至UBL代码运行。其它启动方式请参考DM6446数据手册。
- UBL阶段(User Boot Loader)
  - 即u-boot阶段。
  - U-boot中最初阶段主要完成系统时钟，DDR频率的初始化，准备加载C程序运行的环境，这时候程序运行在ARM RAM中。
  - 拷贝u-boot代码到DDR中，并跳转到C程序的start_armboot处运行。
- Kernel
  - U-boot传递引导参数到Linux Kernel，Kernel会根据从u-boot传递过来的参数决定启动Kernel的方式。比如可以通过TFTP下载Kernel执行；从烧写到Flash中的Kernel启动执行等。等Kernel起来后，Kernel同样会根据u-boot提供的参数来挂载根文件系统。

达芬奇软件架构

TMS320DM6446 达芬奇技术软件架构

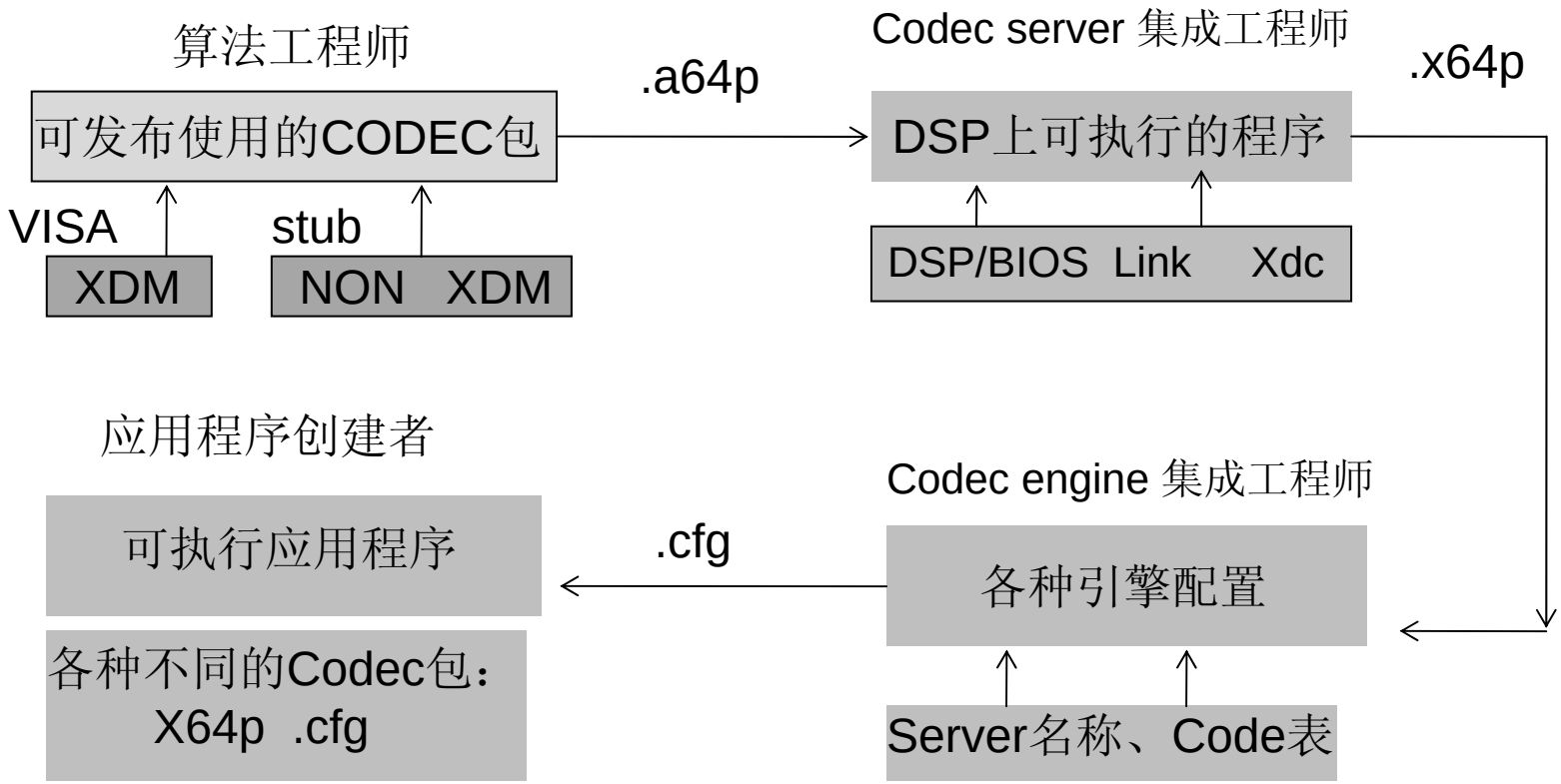


DaVinci 软件开发基本概念



DaVinci 软件开发流程

- 软件开发分为应用层、信号处理层和I/O层三部分，达芬奇软件开发通常需要以下四个步骤：



DaVinci软件开发步骤

- DaVinci的软件开发通常需要4个步骤
  - 第一步：工程师需要基于DSP利用CCS开发自己的音视频编解码算法，编译生成一个编解码算法的库文件*.lib(等同于Linux环境下的*.a64P，直接在Linux环境下修改文件名即可)。
  - 第二步：编译生成一个在DSP上运行的可执行程序*.x64P(即.out文件)，也就是DSP Server。
  - 第三步：根据DSP Server的名字及其中包含的具体的音视频编解码算法创建Codec Engine的配置文件*.cfg。
  - 第四步：应用工程师收到不同的Codec包、DSP Server和Engine配置文件*.cfg，连同自己的应用程序一起编译、链接，最终生成在ARM侧可执行的一个文件。

xDC – Express DSP Component

- 在Linux下封装DSP端可执行程序的工具，xDC根据一套编译指令来生成可执行的文件。xDC的源文件可以是C程序、CPP程序、汇编程序和库文件等。
  - 创建DSP Server源文件
  - 设置xDC的配置文件
  - Linux下编译生成.x64P
- 在DVS6446 SDK开发包中集成了xDC工具，而且有配套的程序示例了如何封装DSP端的算法(DSP Server)

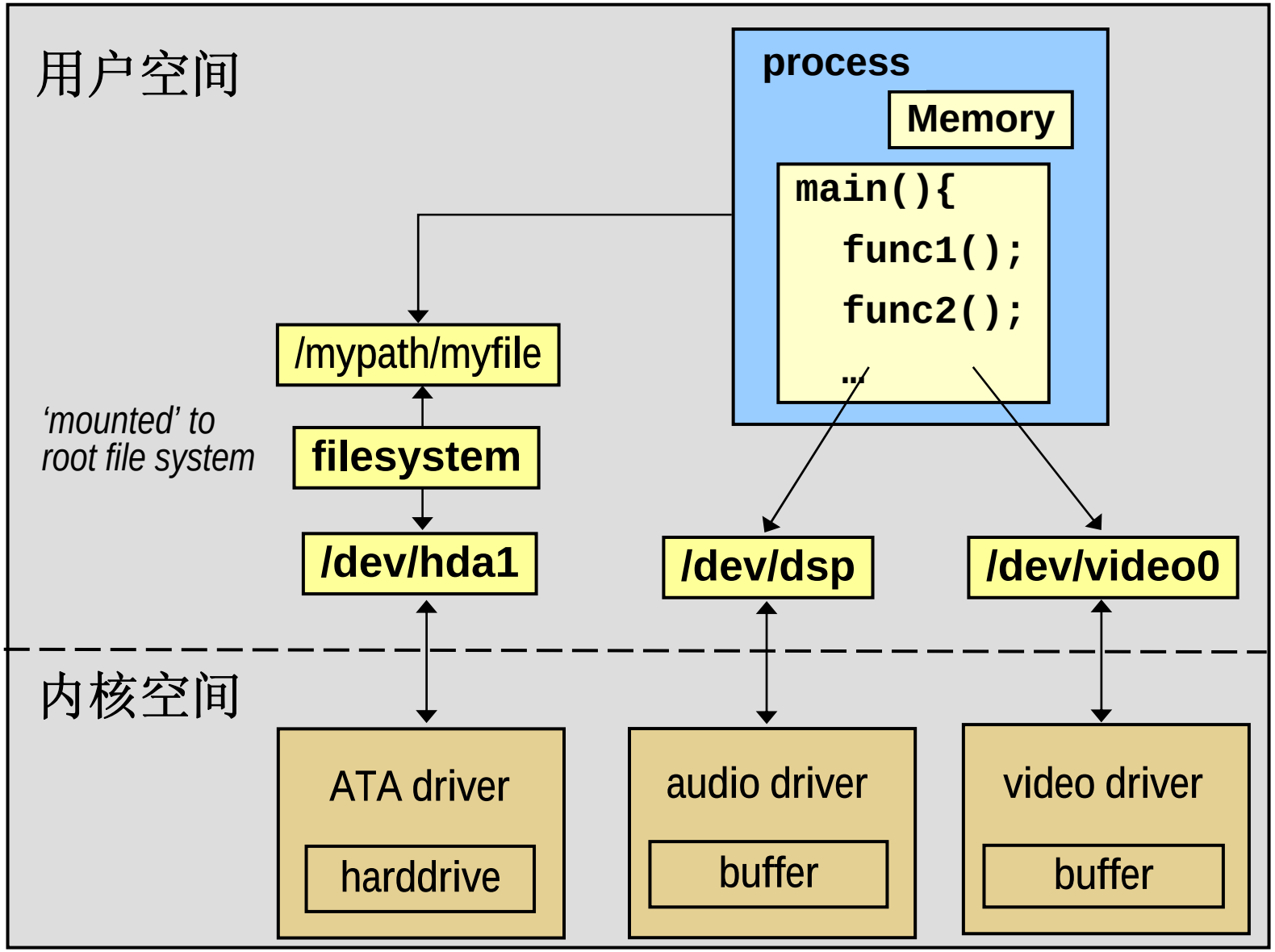
EPSI – I/O层接口

- EPSI
  - **E**asy **P**eripheral **S**ystem **I**nterface
- Linux Drivers(DEVEM LSP)
  - Serial – UART,I2C,SPI
  - Storage – ATA,NAND,MMC/SD
  - Network – 10/100 Ethernet
  - USB – Host and Gadget Drivers
  - Boot – Das UBoot 1.1.3(open source linux boot-loader)
  - Audio – OSS Audio Driver
  - Video – V4L2 for Capture and FBDev for display

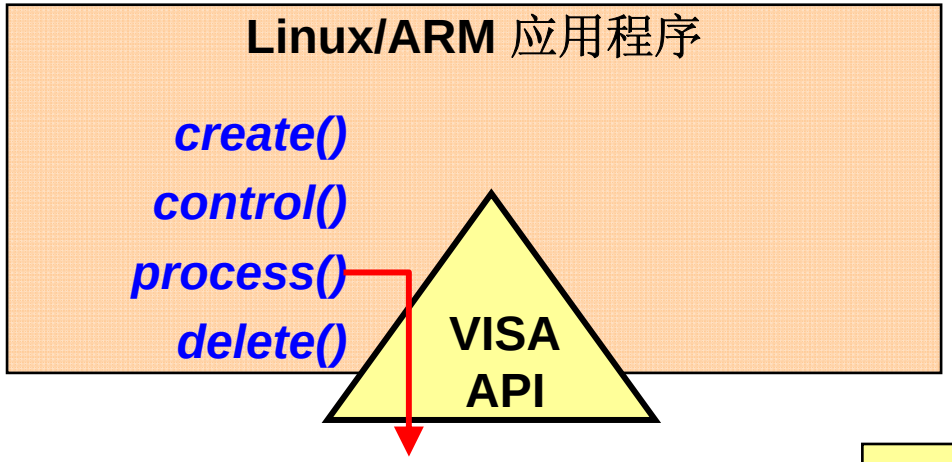
Linux – 基本的设备 Driver API

- 设备：字符设备和块设备
  - 字符设备
    - /dev/video0 视频流设备
    - /dev/dsp 音频流设备
  - 块设备
    - /dev/had ATA->harddrive
    - /dev/ram 外部RAM
- Linux下对设备的访问是通过访问文件的方法来进行的
  - myFileFd = fopen(“/mnt/harddrive/myfile”,”rw”);
  - Fread,fwrite,fclose

访问设备示例


用户空间
process
Memory
main(){
func1();
func2();
…
/mypath/myfile
'mounted' to
root file system
filesystem
/dev/hda1
/dev/dsp
/dev/video0
内核空间
ATA driver
harddrive
audio driver
buffer
video driver
buffer

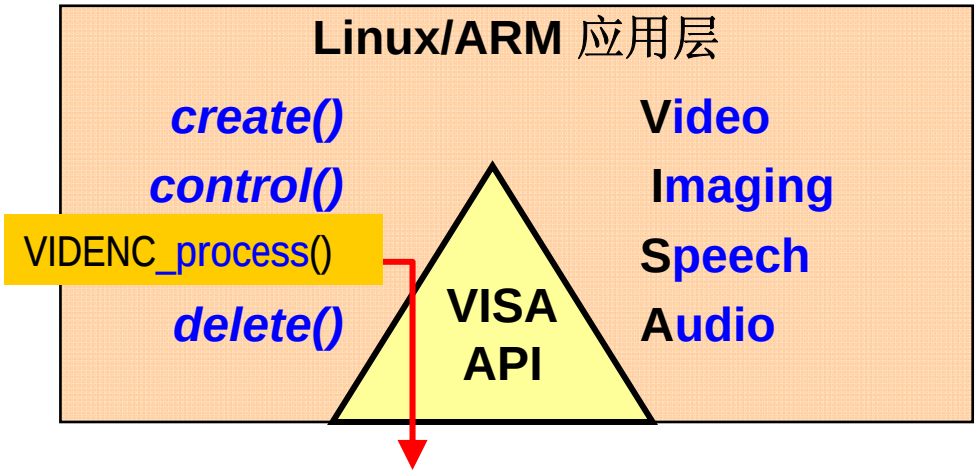
VISA – 4个SPL函数



- 信号处理层的复杂性被抽象为四个处理函数:
  `_create` `_delete`
  `_process` `_control`

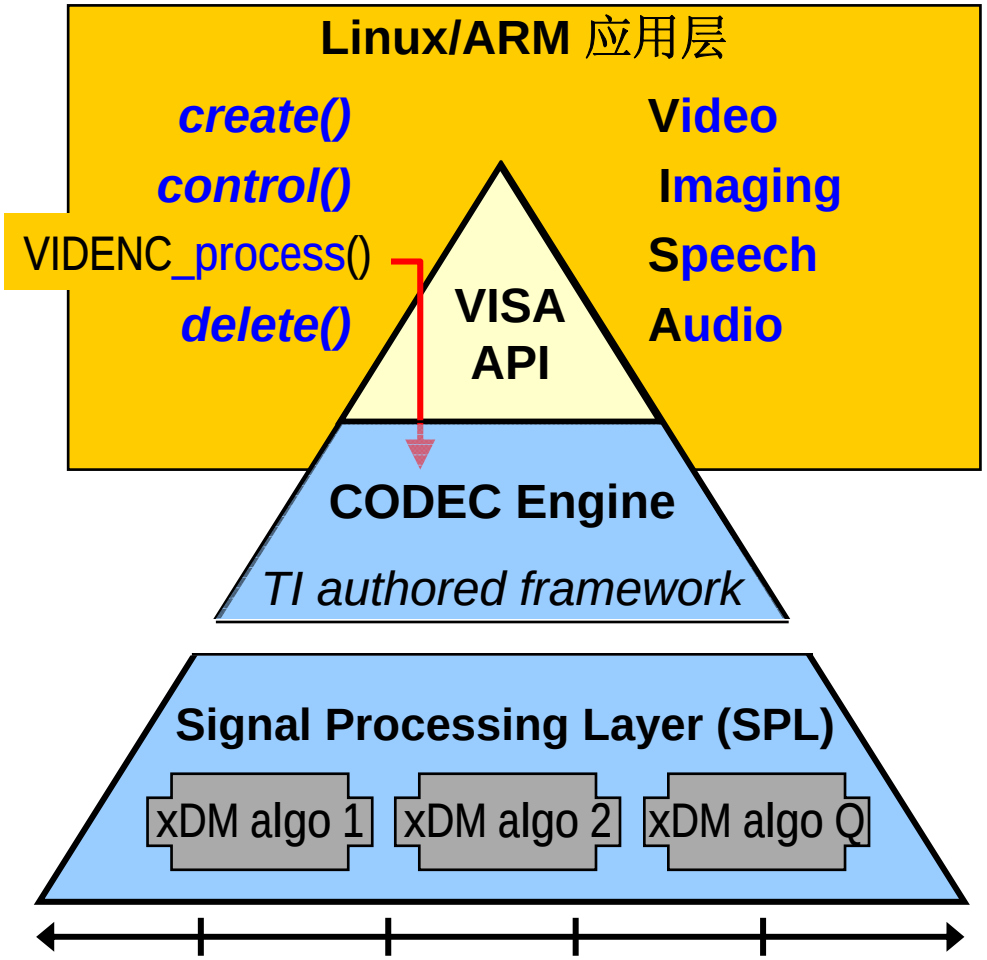
- Create(): 创建一个算法的实例。即为该算法分配请求的存储器空间并初始化该算法
- Process(): 调用算法处理函数，传递输入/输出buffer描述符
- Control(): 用来改变算法的属性设置
- Delete(): 删除算法的实例
  是Create()函数的反操作，会释放掉先前分配给该算法实例的存储器空间

VISA – 8个处理类



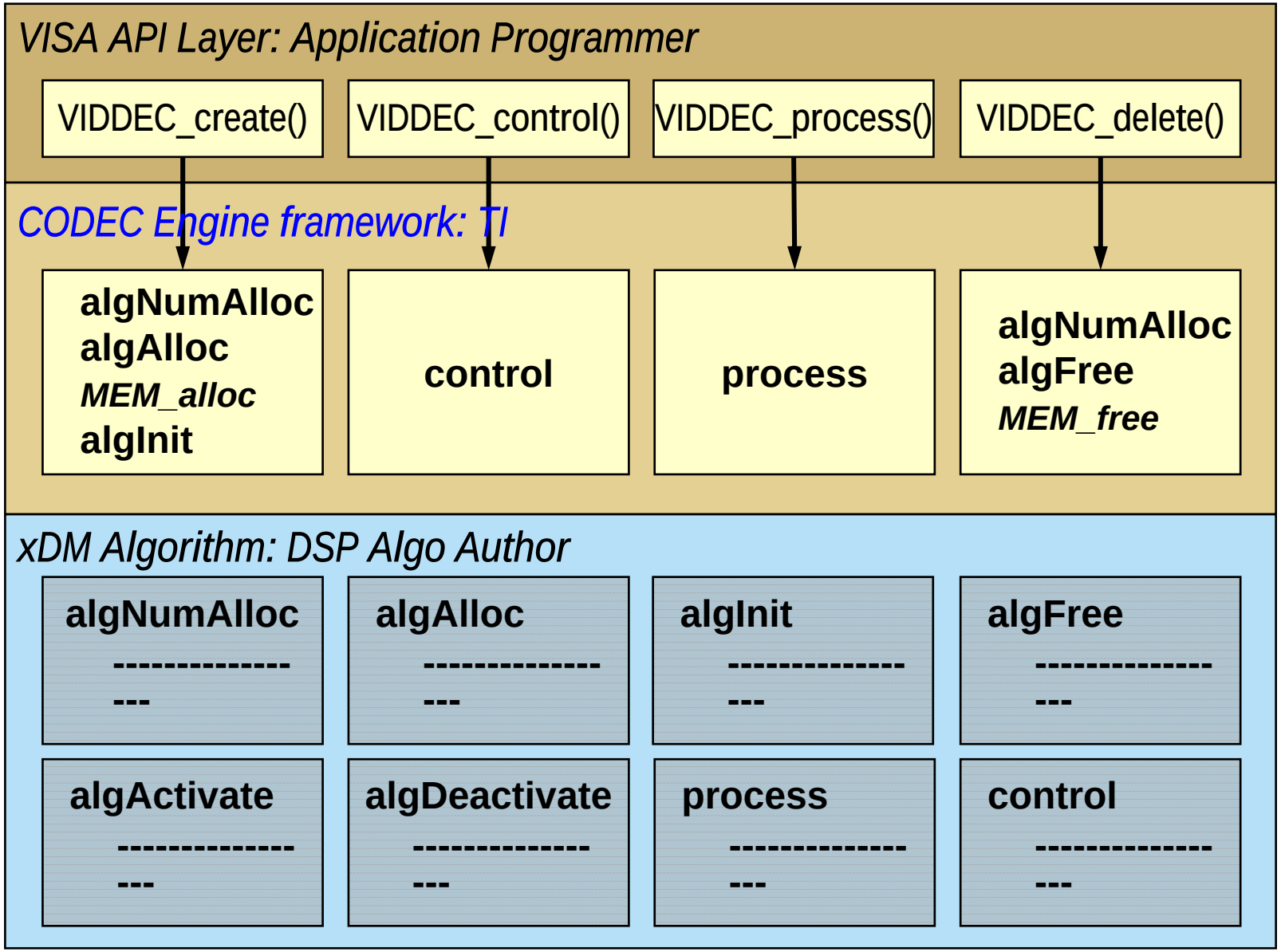
- 复杂的信号处理层被抽象为四个函数:
- `_create` `_delete` `_process` `_control`
- VISA = 4 个处理域 :
  Video Imaging Speech Audio
- Separate API set for encode and decode thus, a total of 8 API classes:
  VIDENC IMGENC SPHENC AUDENC
  VIDDEC IMGDEC SPHDEC AUDDEC

VISA – SPL Interface



- CODEC引擎(CE) 是 VISA 和a算法之间的一个抽象层

VISA – CODEC Engine - xDM



VISA – CODEC Engine - xDM

VISA API Layer: Application Programmer

VIDDEC_create() | VIDDEC_control() | VIDDEC_process() | VIDDEC_delete()

CODEC engine is provided by TI

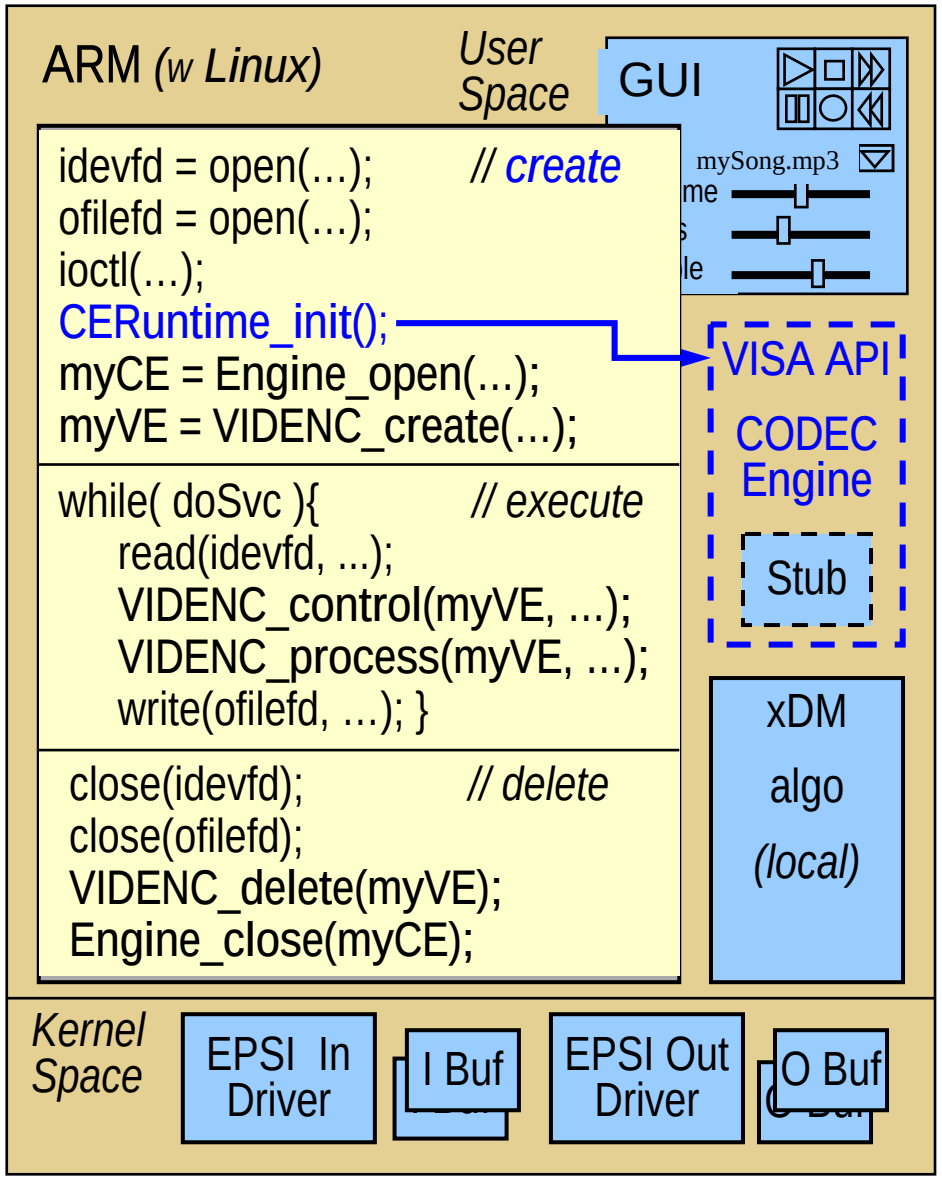
You need only be concerned with

VISA or xDM

xDM Algorithm: DSP Algo Author

algNumAlloc ---------- ---	**algAlloc** ---------- ---	**algInit** ---------- ---	**algFree** ---------- ---
algActivate ---------- ---	**algDeactivate** ---------- ---	**process** ---------- ---	**control** ---------- ---

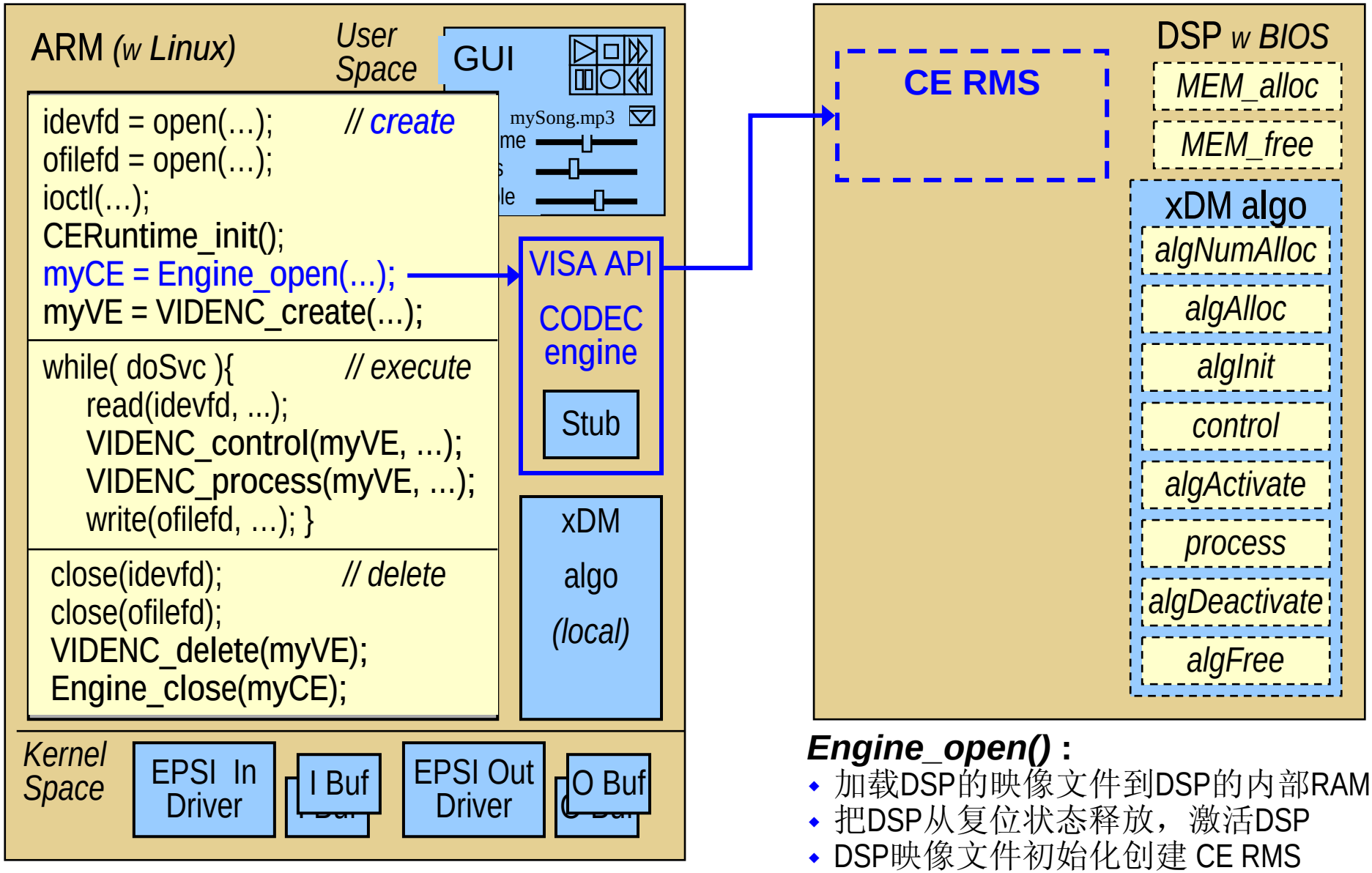
CODEC Engine: CERuntime_init()



CERuntime_init() **:**

- 创建Codec Engine线程
- 仅需调用一次

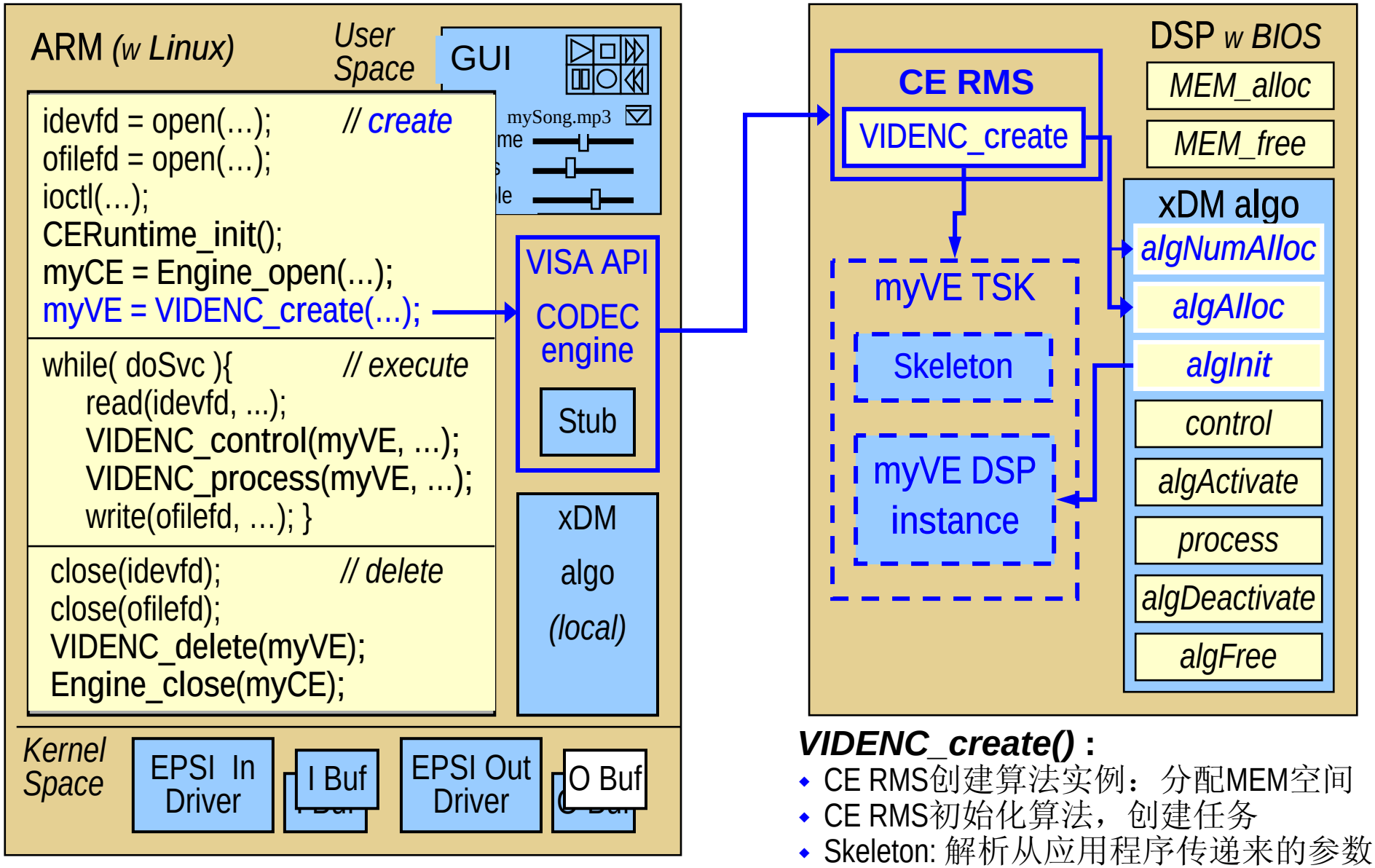
CODEC Engine: Engine_open()



Engine_open() **:**

- 加载DSP的映像文件到DSP的内部RAM
- 把DSP从复位状态释放，激活DSP
- DSP映像文件初始化创建 CE RMS

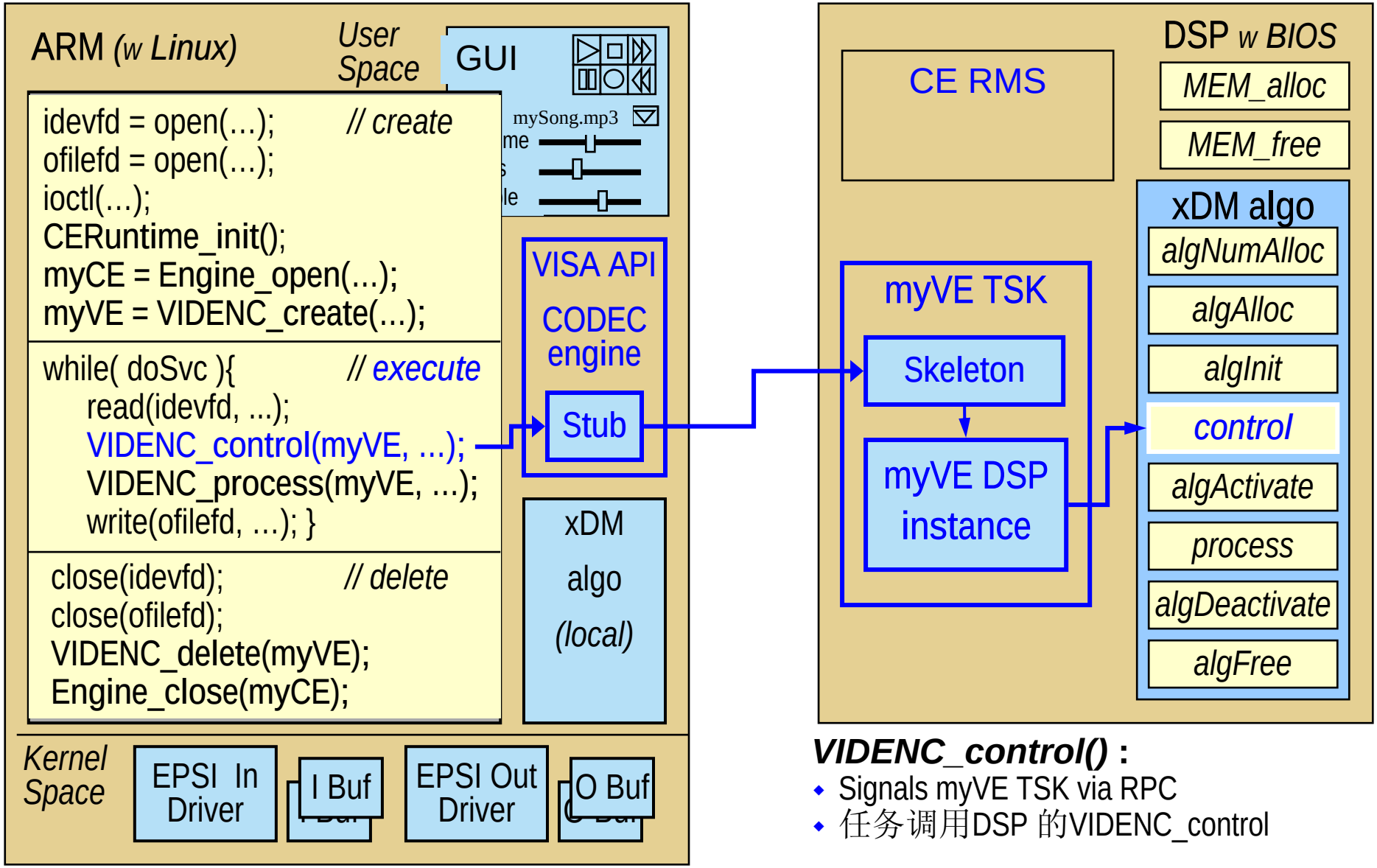
CODEC Engine: VIDENC_create()



VIDENC_create() **:**

- CE RMS创建算法实例：分配MEM空间
- CE RMS初始化算法，创建任务
- Skeleton: 解析从应用程序传递来的参数

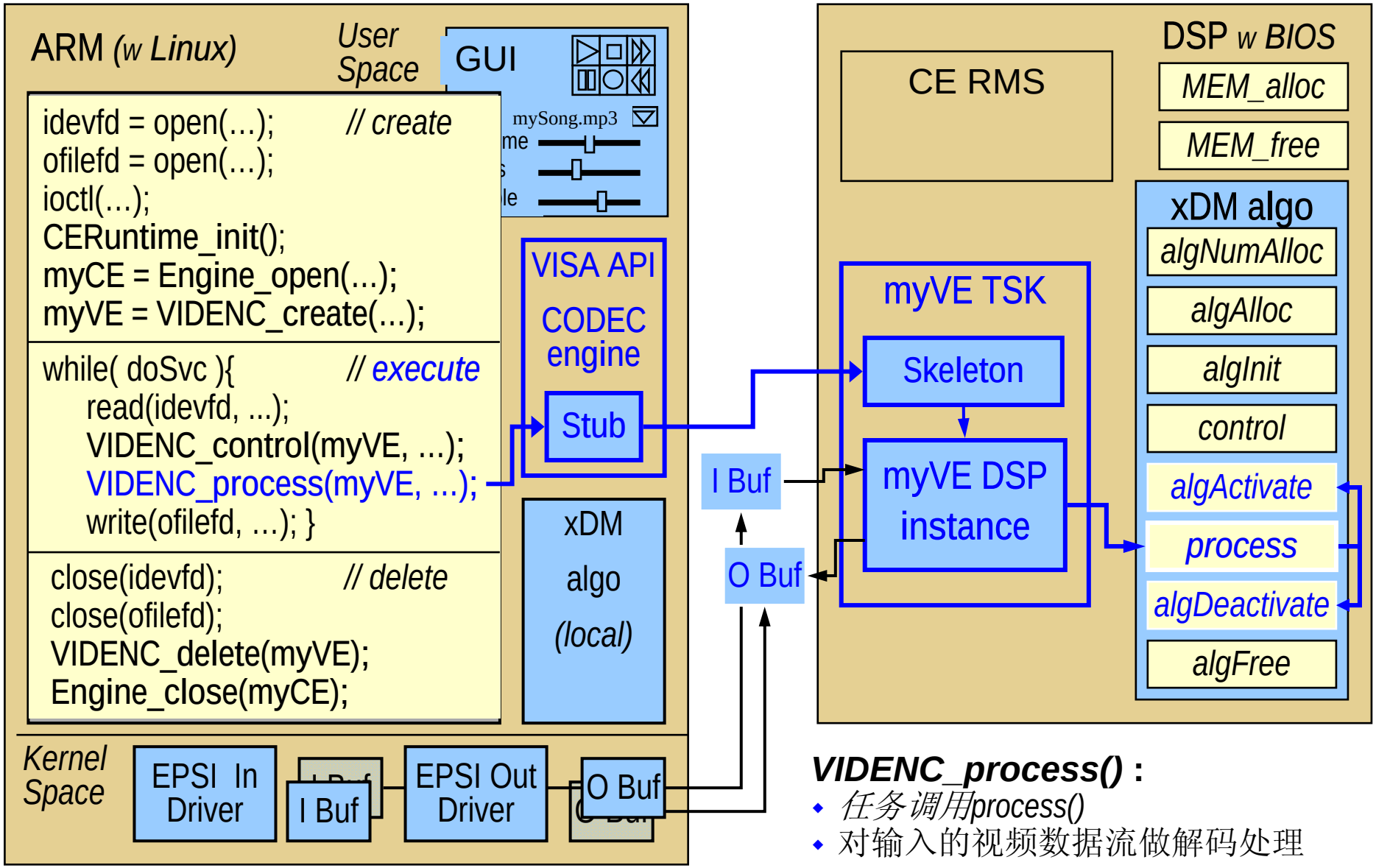
CODEC Engine: VIDENC_control()



VIDENC_control() **:**

- Signals myVE TSK via RPC
- 任务调用DSP 的VIDENC_control

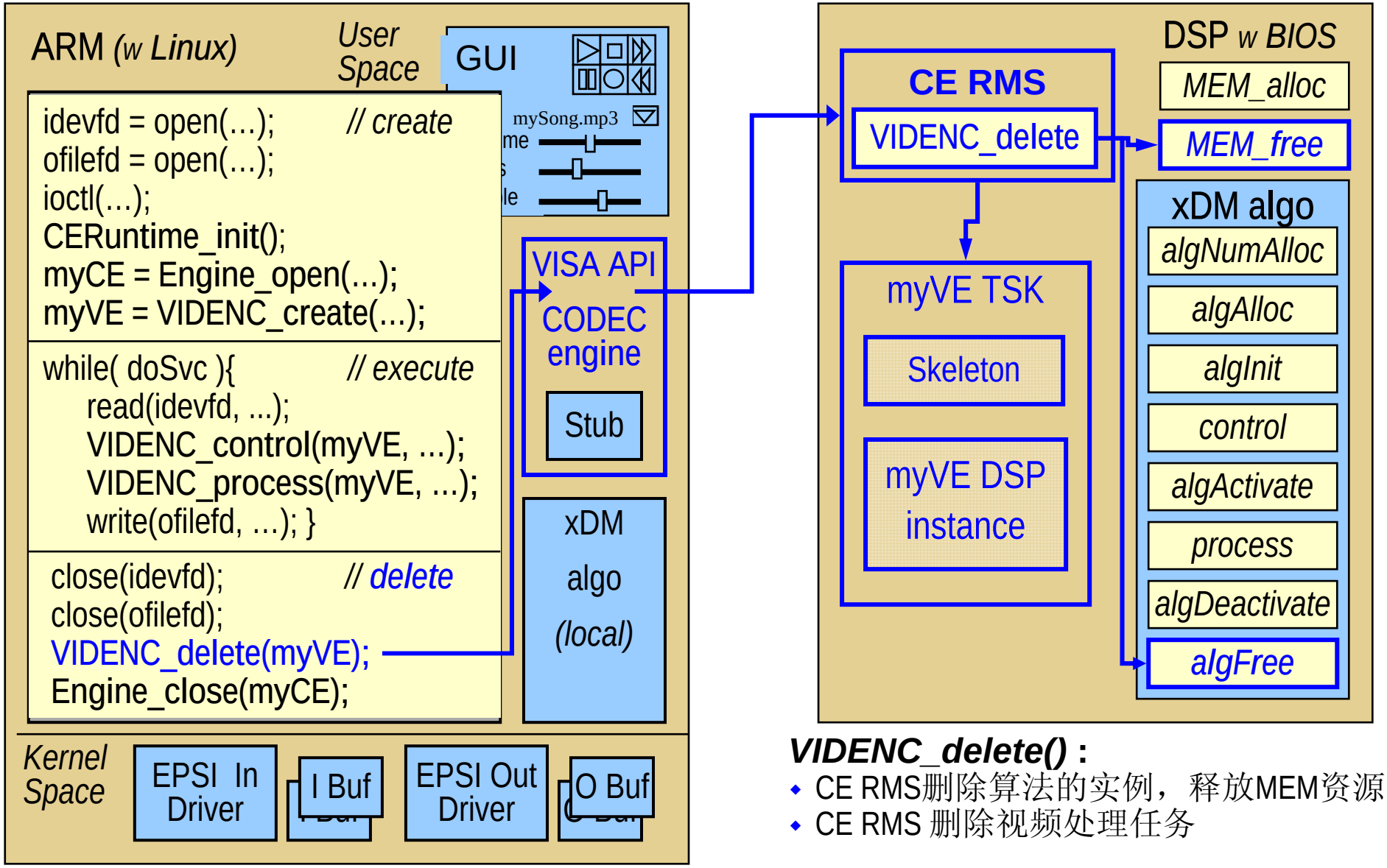
CODEC Engine: VIDENC_process()



VIDENC_process() **:**

- *任务调用process()*
- 对输入的视频数据流做解码处理

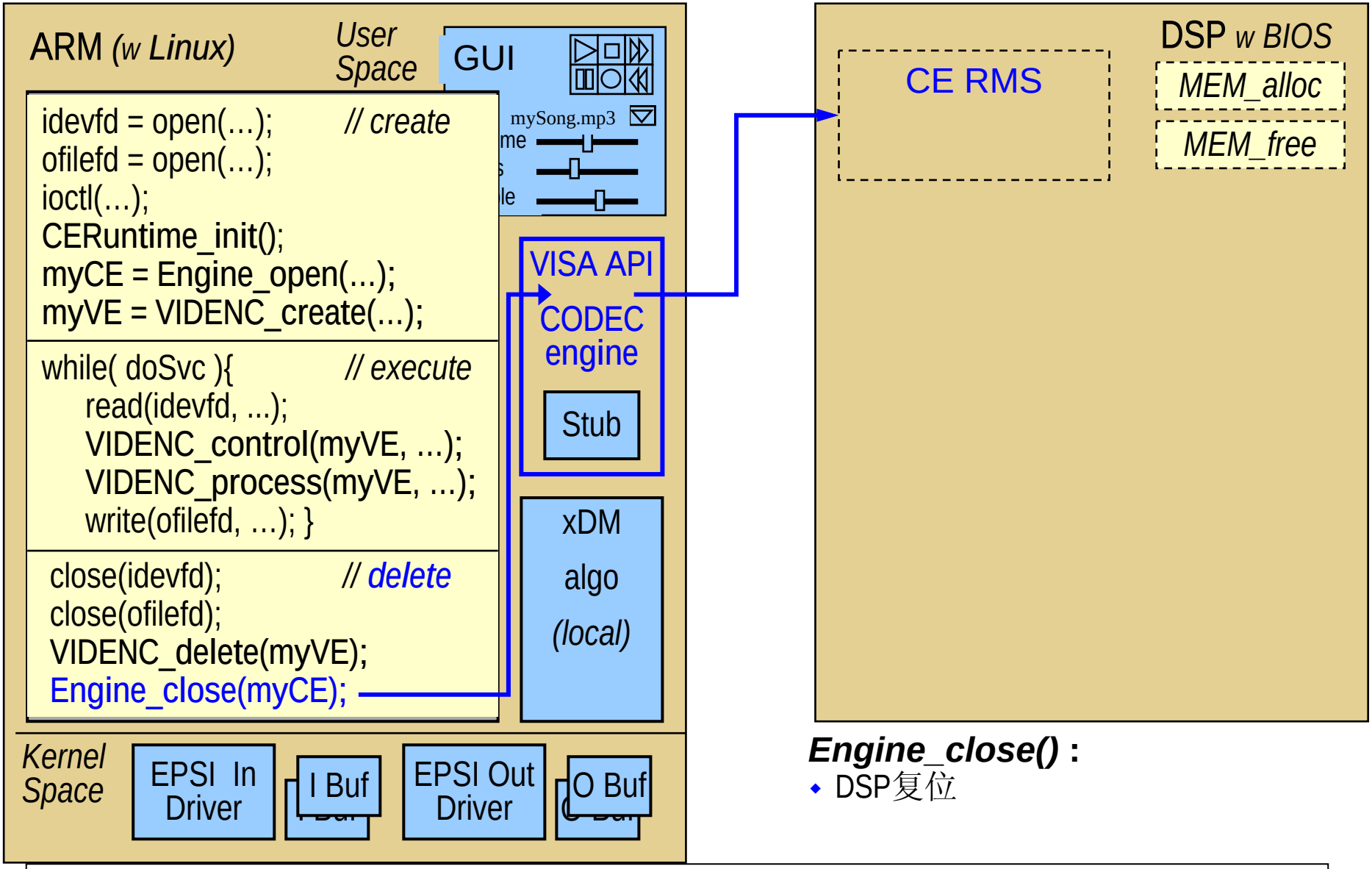
CODEC Engine: VIDENC_delete()



VIDENC_delete() **:**

- CE RMS删除算法的实例，释放MEM资源
- CE RMS 删除视频处理任务

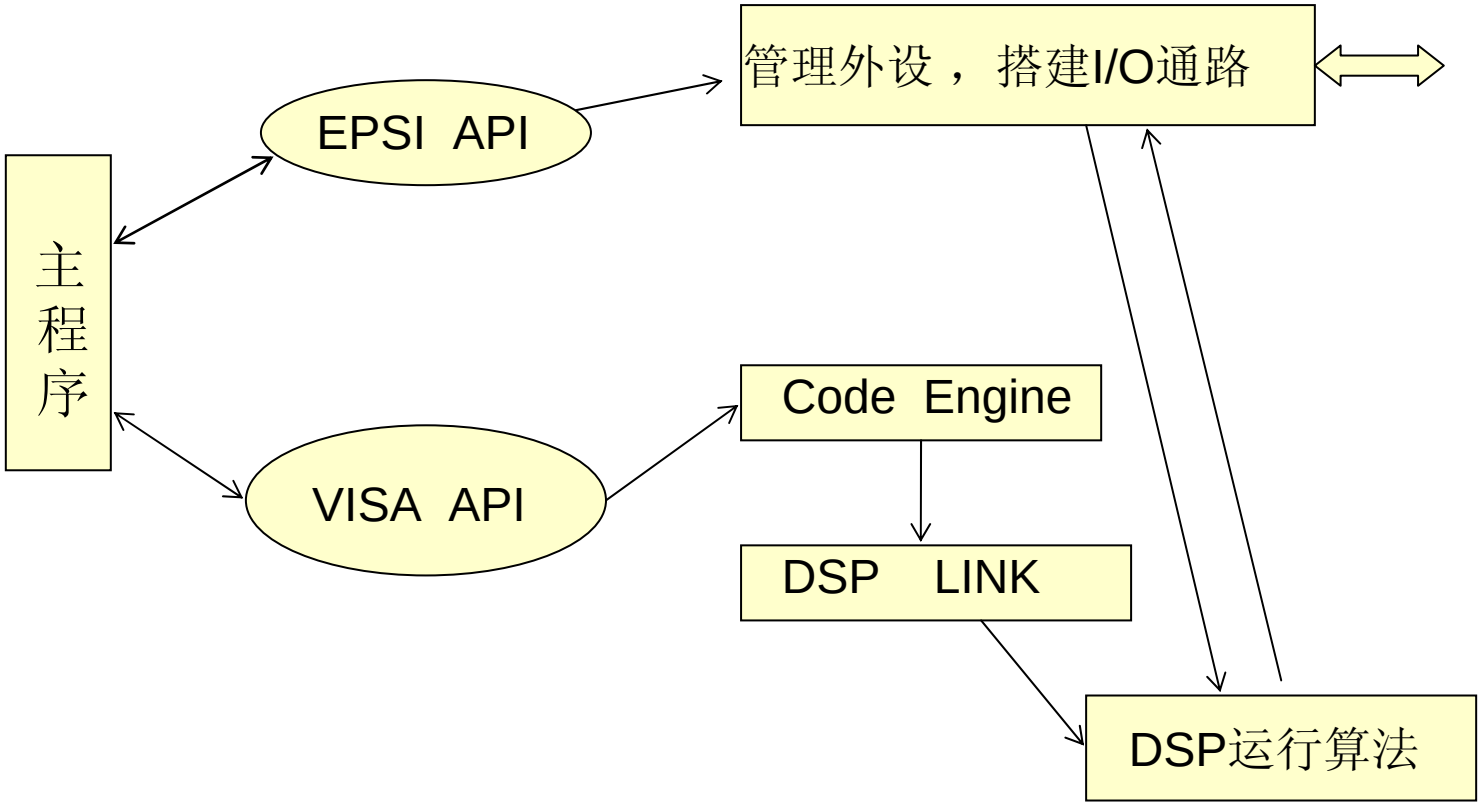
CODEC Engine: Engine_close()



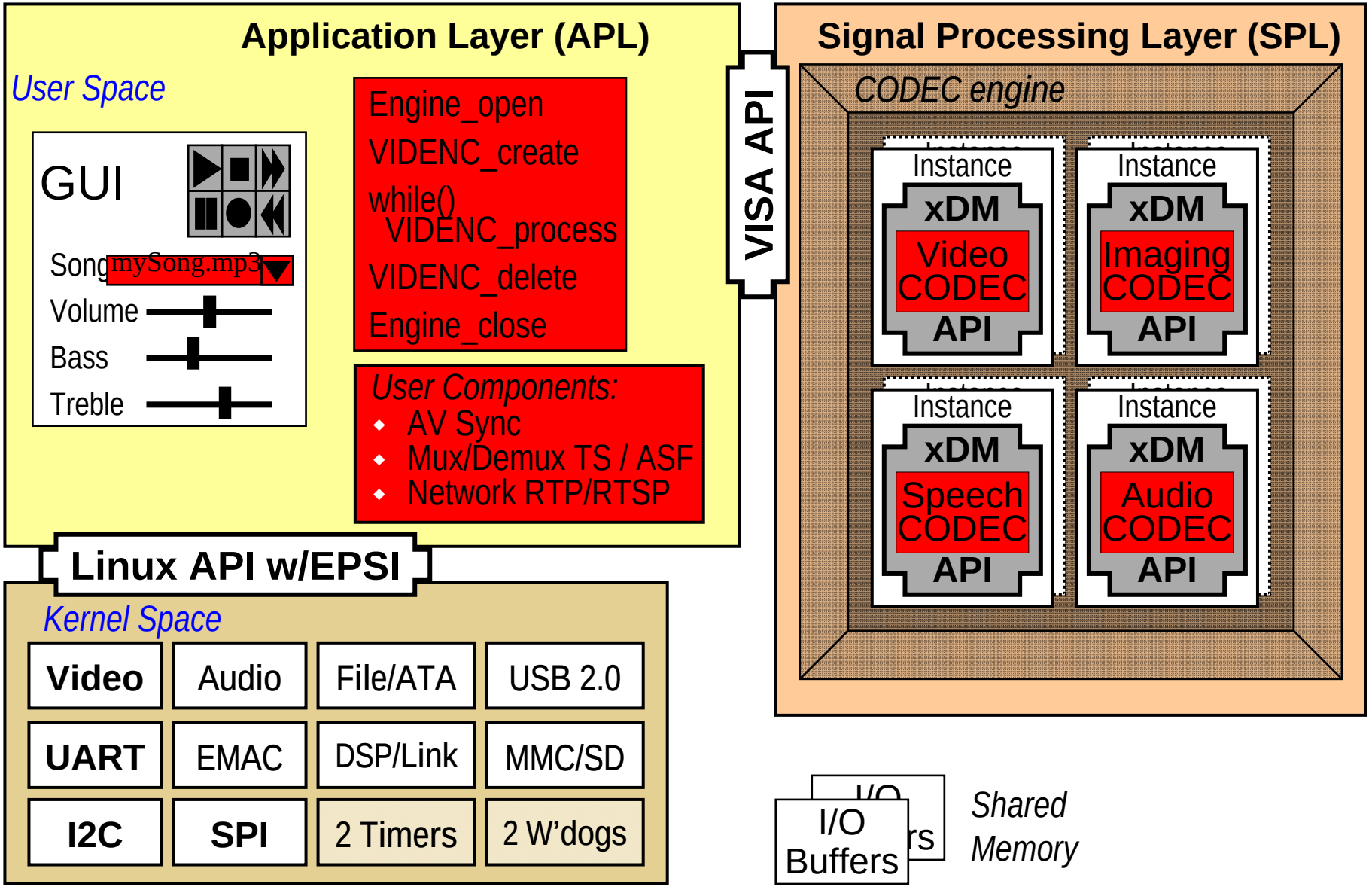
Engine_close() **:**

- DSP复位

总结


管理外设，搭建I/O通路
EPSI API
主程序
Code Engine
VISA API
DSP LINK
DSP运行算法

DaVinci 技术软件架构



示例伪代码

```
idevfd = open("/dev/xxx", O_RDONLY);
ofilefd = open("./fname", O_WRONLY);
ioctl(idev fd, CMD, &args);
myCE = Engine_open("vcr", myCEAttrs);
myVE = VIDENC_create(myCE,"videnc",params);

while( doRecordVideo == 1 ) {
        read(idevfd, &rd, sizeof(rd));
        VIDENC_control(myVE, ...);
        VIDENC_process(myVE, ...);
        write(ofilefd, &wd, sizeof(wd));
}
 close(idevfd);
 close(ofilefd);
 VIDENC_delete(myVE);
 Engine_close(myCE);
```

// Create Phase

// 得到输入设备描述符

// 得到输出设备描述符

// 初始化IO设备...

// 准备VISA环境

// 调用视频编码算法

// Execute phase

// 读输入设备数据

// 执行VISA的控制函数

// 执行VISA处理函数，处理数据流

// 输出结果到输出设备

// Delete phase

// 关闭打开的I/O设备

// 释放分配的存储器空间

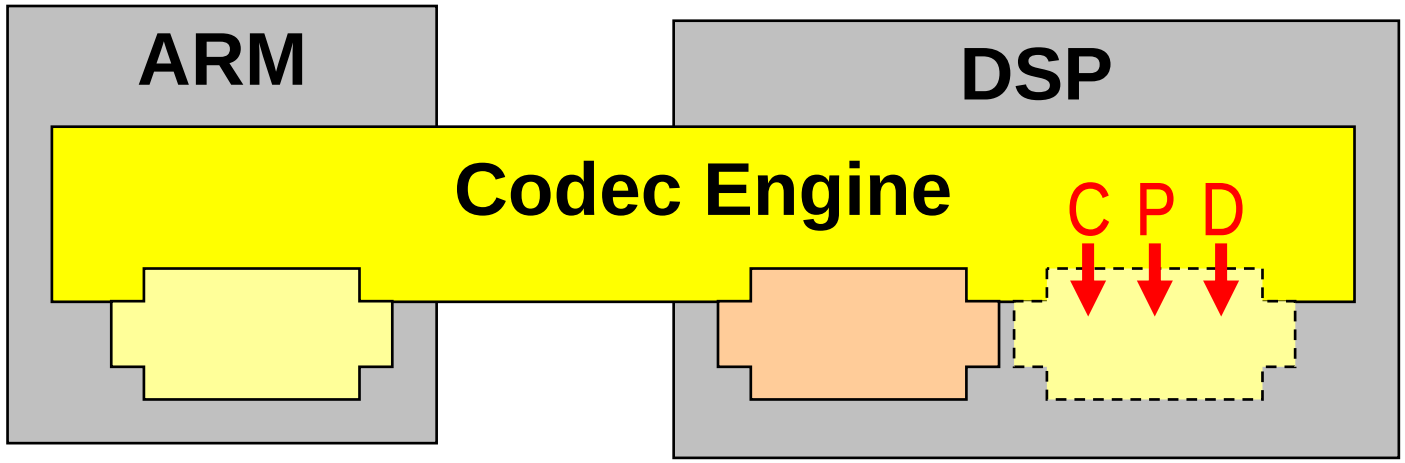
// 关闭引擎

思考的问题

- 编解码算法是在DSP端运行的，那应用程序是如何由ARM加载到DSP端的？(上面讲到的)
- DSP端的算法需要符合什么标准？
- ……接下来我们要讨论的xDAIS/xDM算法标准

编解码算法标准 – xDAIS/xDM

- xDAIS API定义的抽象接口包括2部分：IALG和IDMA2
  - IALG：定义独立于软件架构的抽象接口，完成对算法实例的创建。每个算法都必须实现IALG接口。
  - IDMA2：定义算法使用的DMA资源的接口。
- IALG中定义的函数API的主要功能：
  - 用于创建、初始化和删除实例对象的函数：**C**reate(),**D**elete()
  - 算法处理的函数：**P**rocess()



DVEVM和DVSDK的建立和安装

TI DVSDK and DVEVM

- **DVSDK – 数字视频软件开发包**
  - 开发DSP侧的算法
  - 提供全部的MontaVista Linux
  - 主要内容包括：
    - Mont Vista Linux Professional Edition v4
    - Linux下的DSP/BIOS
    - Linux下的编译，汇编工具
    - 框架组件
- **DVEVM – 数字视频评估模块**
  - 开发ARM侧的应用
  - 主要内容包括：
    - 开发板(TI EVM 或SEED-DVS6446)
    - CCD摄像头
    - LCD显示器
    - A/V 线

ARM端开发环境的建立及配置 - DVEVM

- 对DaVinci平台，TI在硬件上给予双核架构强有力的支撑，在DSP端用DSP/BIOS来支持音视频算法的运行，在ARM端用MontaVista Linux来支持其对外设的管理。对于ARM与DSP之间的数据交互，则用Codec Engine和Codec Server来加以管理。
- DaVinci的开发程序分为Codec部分和应用程序部分。开发应用程序前，需要搭建软硬件开发环境。硬件环境包括：DAVINCI开发板DVEVM、CCD摄像头、LCD显示器、硬盘、串口线。其次是与DVEVM配套的ARM端开发环境。环境搭建好后，需要对Linux主机进行相关配置才能使用DVEVM开发板。
- 配置ARM端软件开发的各个模块，包括：
  - TFTP
  - NFS
  - Bootloader/Linux Kernel的烧写
  - 设置DVEVM的启动参数

DSP开发环境的建立及配置 - DVSDK

- 如果需要开发DSP端的算法还需要安装使用DVSDK。
- 该DVSDK数字视频软件开发包包括：
  - MontaVista Linux Professional Edition v4：相对于DVEVM发布的MontaVista Linux Demo版本来说，这个完全版包含了DevRocket IDE和相关服务支持，要全面的多。
  - TI Codegen for Linux：与DSP相关的一些编译、连接工具。
  - Framework Components：主要用来支持DSP端算法开发的一些模块，能够管理符合xDAIS标准的算法模块，分配内存和DMA资源。这些模块是被CE来使用的，但如果有必要在DSP端程序也可以使用它们
  - DVSDK的版本必须与DVEVM的版本号一致。且最好把相关的DVSDK安装在DVEVM的安装目录下。
- 安装步骤:
  - 将目录下SEED-DVS6446_SDK_tar.gz复制到Linux服务器的/opt目录下进行解压安装.
    默认的安装路径为/opt, 安装完成后创建了3个文件夹:dvevm_1_20, mv_pro_4.0, nfs.
  - 配置SDK.
    - 配置ARM v5t交叉编译器PATH. 进入root根路径,执行命令: Host # cd /opt.
    - 修改root目录下.bash_profile文件. 打开.bash_profile文件: Host # vi .bash_profile, 在PATH=$PATH:$HOME/bin下面添加如下内容:

.bash_profile

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin
PATH="/opt/mv_pro_4.0/montavista/pro/devkit/arm/v5t_le/bin:/opt/mv_pro_4.0/montavista/pro/bin:/
opt/mv_pro_4.0/montavista/common/bin:$PATH"
export PATH
unset USERNAME
```

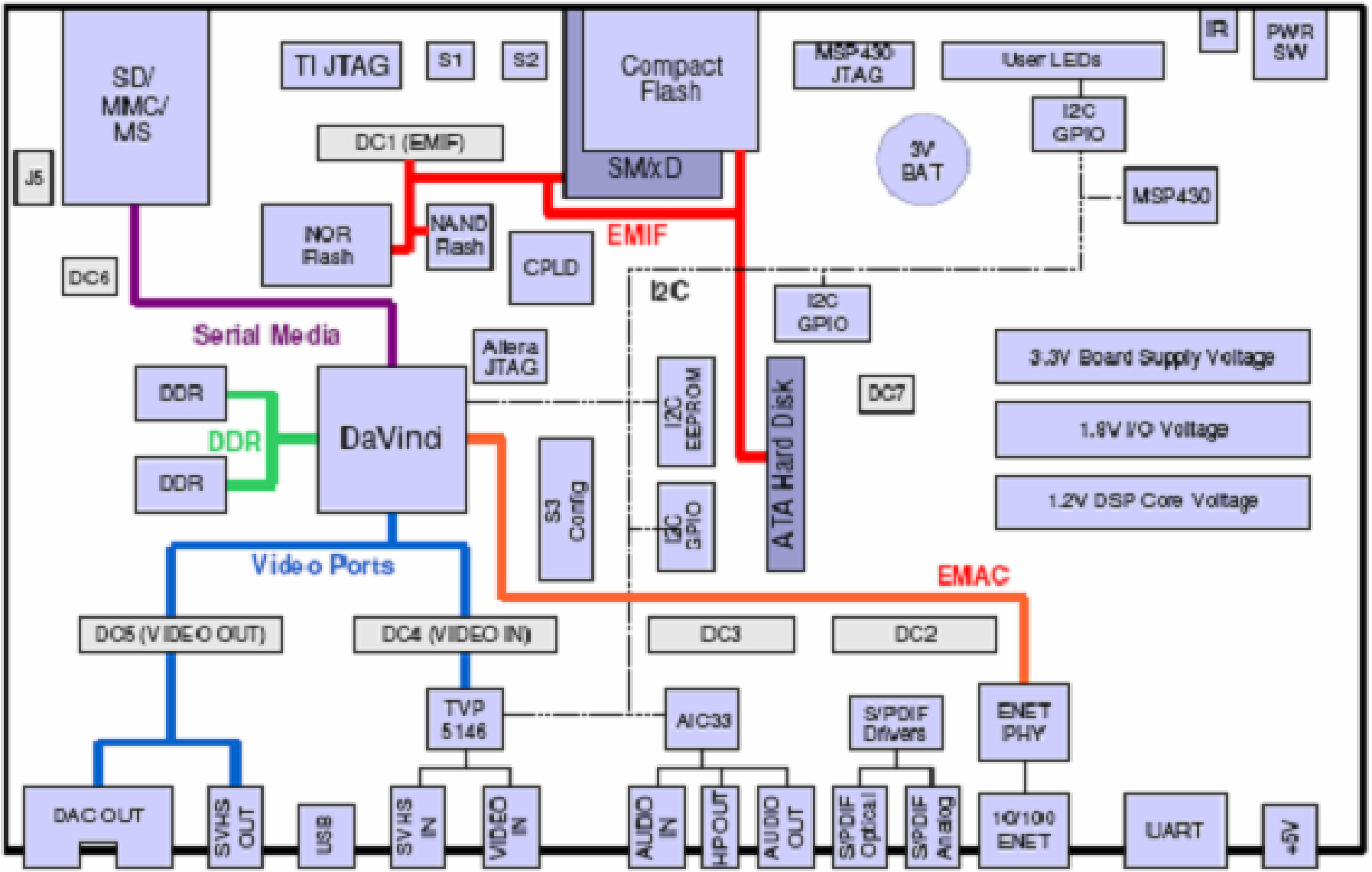
DaVinci工具链的建立及配置 - DVSDK

- 安装完ARM v5t交叉编译环境后可进行简单的测试ARM v5t编译器是否正常工作
  - 在Linux服务器的控制台输入arm_v5t_le-gcc命令,如果显示如下信息则表明安装正常.

  - 配置nfs网络文件服务.
    - 修改/etc/exportfs文件,添加如下内容/opt/nfs *(rw,sync,no_root_squash,no_all_squash).
    - 运行nfs服务器: /user/sbin/exportfs –a, /sbin/service/nfs restart

SEED-DVS6446平台介绍

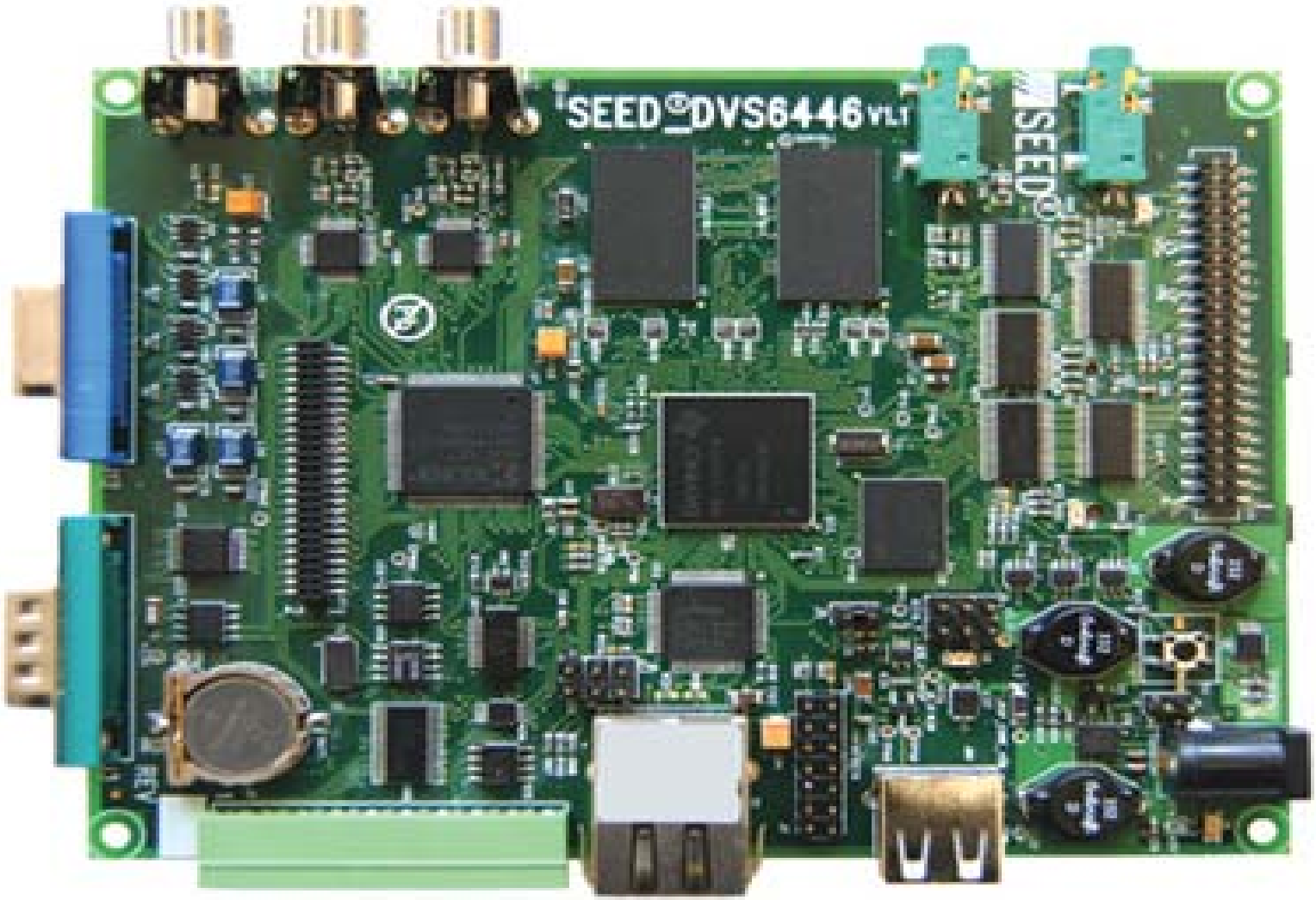
TI EVM框架结构


SD/
MMC/
MS
TI JTAG
S1
S2
Compact
Flash
SM/xD
MSP430
JTAG
User LEDs
IR
PWR
SW
I2C
GPIO
3V
BAT
MSP430
J5
DC1 (EMIF)
NOR
Flash
NAND
Flash
CPLD
EMIF
DC6
I2C
I2C
GPIO
Serial Media
Altera
JTAG
3.3V Board Supply Voltage
DDR
DDR
DDR
DaVinci
I2C
EEPROM
ATA Hard Disk
DC7
1.8V I/O Voltage
S3
Config
I2C
GPIO
1.2V DSP Core Voltage
Video Ports
EMAC
DC5 (VIDEO OUT)
DC4 (VIDEO IN)
DC3
DC2
TVP
5146
AIC33
S/PDIF
Drivers
ENET
PHY
DAC OUT
SVHS
OUT
USB
SVHS
IN
VIDEO
IN
AUDIO
IN
HP OUT
AUDIO
OUT
S/PDIF
Optical
S/PDIF
Analog
10/100
ENET
UART
+5V

SEED-DVS6446

- 在TI EVM板的基础上，重新设计的现场应用的板卡
  - 符合工业标准，长130mm，宽100mm
  - 应用领域
    - 视频安全监控领域
    - IP机顶盒
    - 车载娱乐系统
    - 便携式多媒体播放器
    - 数码相机
    - ……

实际硬件板卡图



特点

- 外扩DDR2 256MB
- 外扩NAND Flash 64MB
- 1路NTSC/PAL标准模拟视频输入，1路标准NTSC/PAL标准模拟视频输出，支持VGA的输出
- 1路音频的输入/输出
- 1路10/100M以太网输出
- 标准的ATA标准硬盘接口
- MMC/SD接口
- 支持主从模式的USB 2.0接口
- 2路串口，支持RS232/RS485两种标准接口
- 数字视频的扩展输入接口

DEMO例程

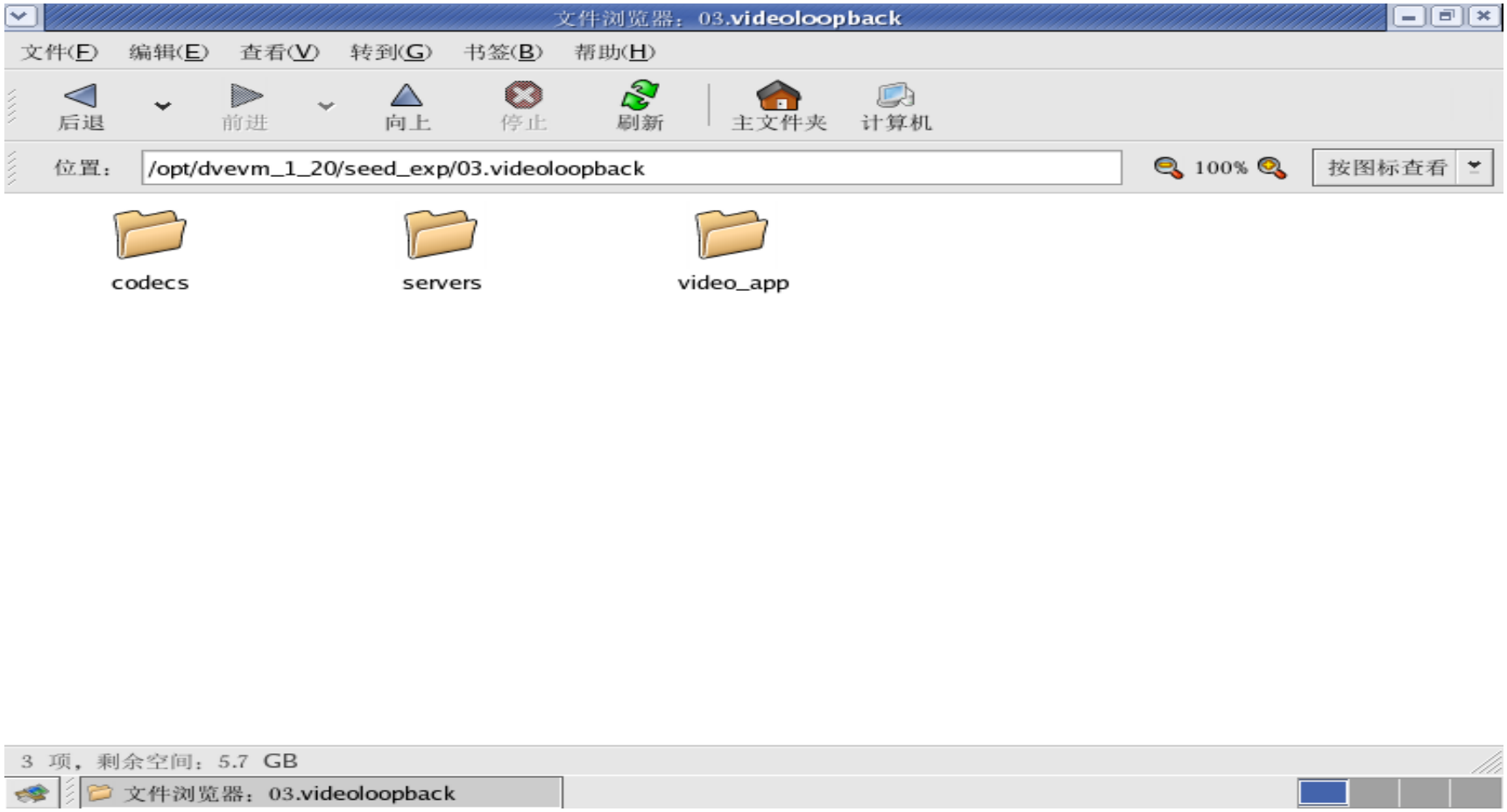
- SDK安装包包含了DEMO程序源码。用户可在此基础上修改实现自己的功能。这部分程序完全是基于ARM端的应用程序，程序中通过Codec Engine提供的机制调用DSP端的音视频算法，实现编码、解码和编解码的功能
- 支持的DEMO有：
  - Encode：编码程序，支持音频G711，视频MPEG2/4，H264的编码标准，存储与文件系统目录下
  - Decode：解码程序，支持音频AAC，G711，视频MPEG2/4、H264标准的解码
  - Encode/Decode：编解码程序，支持视频H264的实时编解码，视频前端采集，算法进行编解码，后端用于显示输出

Linux视频采集回放实验解析

- 目的
  - 学习Codec Engine机制的编程
  - 学习使用XDC编译工具
  - 视频采集和显示驱动的基础

程序包

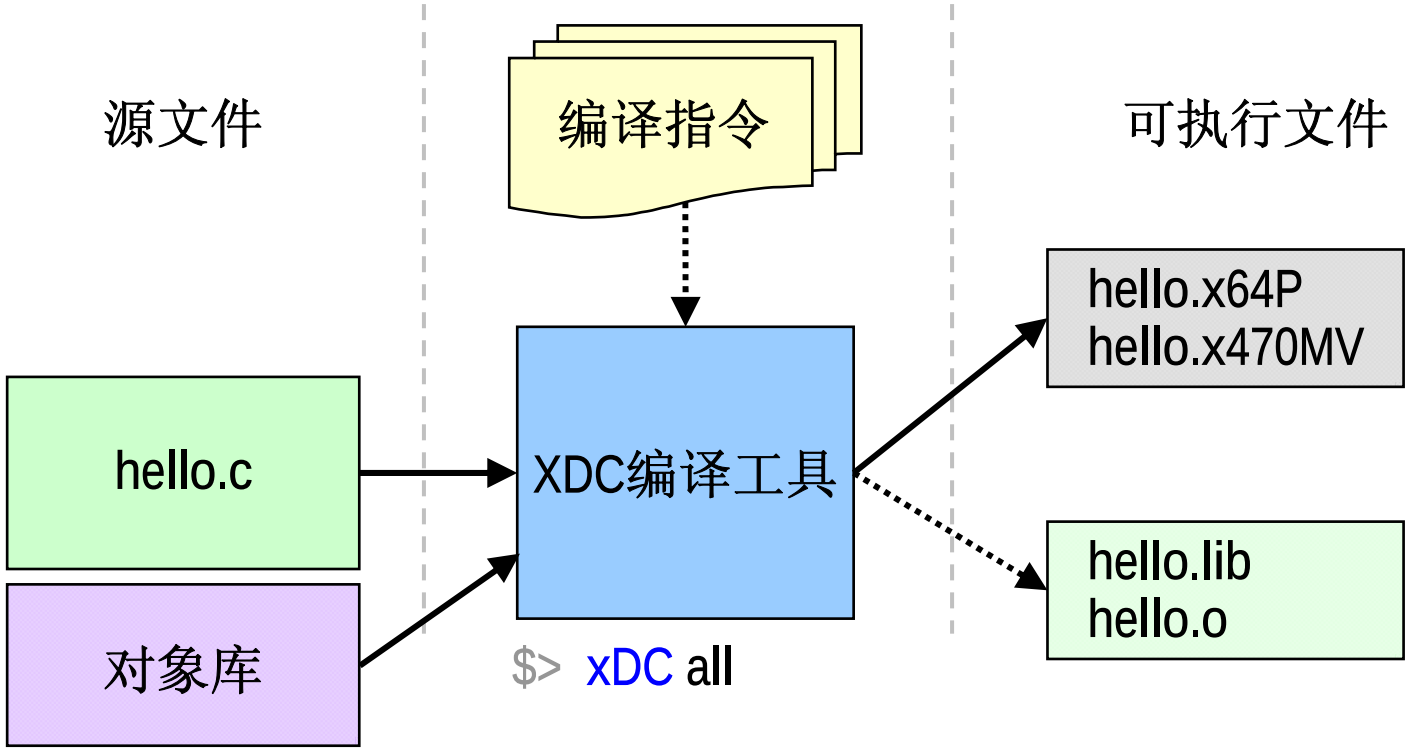
- codecs：包含了codecs算法包
- servers：DSP server封装包
- video_app：ARM端应用程序



Server – DSP Server(Codec Server)

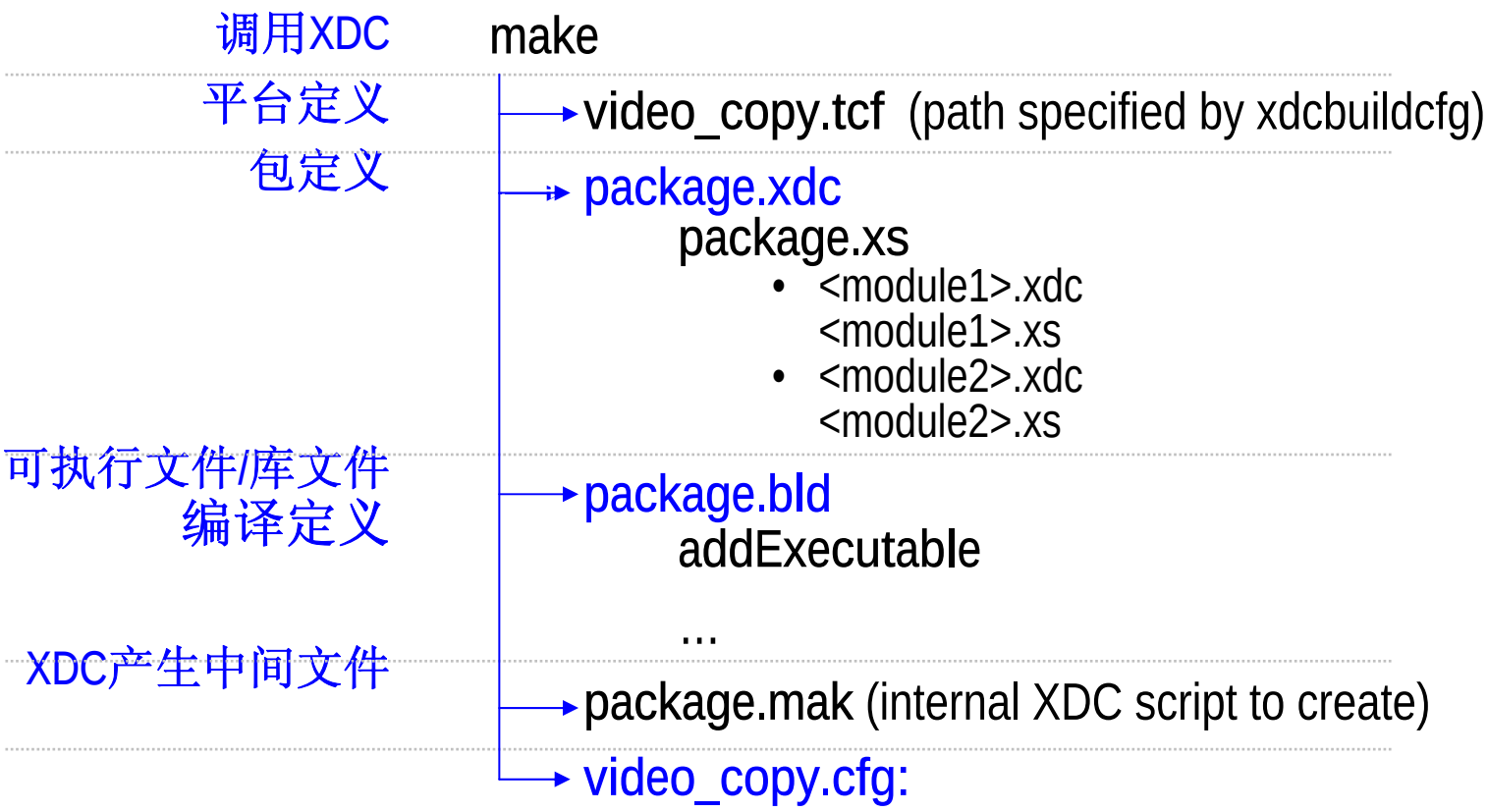
- main.c:打开CE的trace功能，读取或更改main函数的参数等
- Server的DSP/BIOS配置文件.tcf及link.cmd
- **video_copy.tcf**：定义Memory Map、设置DSP的复位/中断向量表，创建和初始化BIOS程序需要的各种数据对象
- **package.bld**：定义target是C64P DSP、要生成针对target的可执行程序，其中的配置脚本文件是video_copy.tcf、链接选项是链接link.cmd文件，同时还要生成main.c的目标代码
- **package.xdc**：声明DSP Server的名字、它的路径及Server的依赖文件

xDC 简介



- xDC的源文件可以是C程序、C++程序、汇编程序和库文件
- xDC根据一套编译指令编译生成可执行文件
- 一次可编译多个目标对象的可执行文件：.x64P是DSP端得可执行文件，.x470MV是ARM端得可执行文件

xDC流程



- xDC根据package.xdc, package.bld,video_copy.cfg三个文件编译生成DSP Server
- servers：DSP server封装包

ARM端程序分析

- ARM端通过Engine配置文件video_copy.cfg来配置使用的codec包、DSP Server
- ARM通过Codec Engine的VISA API完成对DSP端算法的调用，调用前完成Codec Engine的初始化
- 接着调用VIDENC_create(),VIDEO_process()对视频图像编码算法处理

Thanks!