2008/7

电子工程专辑

嵌入式视频处理基本原理

ADI 公司向电子工程专辑供稿

目录

By: *David Katz 和 Rick Gentile*

ADI 公司

Part-1

作为消费者，我们对于各种形式的视频系统都已经了如指掌。不过，从嵌入式开发者的角度来看，视频技术就好象是一个具有不同的分辨率、格式、标准、信源和显示的复杂网络。

章中，我们只对视频中的某些方面进行具体阐述，这些方面都是当今多媒体处理系统中比较常见的问题。首先，简要介绍视频方面的基本知识，然后，将会重点讨论一些在嵌入式多媒体设计中常见的问题，同时，也将针对一些具有挑战性的视频设计问题，提供一些技巧与窍门。

人类视觉感知

来讨论一些简单的生理学问题。正如我们将看到的那样，对我们的眼睛的工作原理的理解为视频和成像技术的发展铺设出一条重要的道路。

眼睛包含两种视觉细胞：杆状细胞和视锥细胞。杆状细胞主要对亮度信息敏感，而对颜色信息不敏感，它们使我们具备夜视能力。与此相反，视锥细胞对亮度并不敏感，但对 400nm（紫光）～770nm（红光）波长范围内的光比较敏感。因此，这些视锥细胞使我们能够感知色彩。

视锥细胞有 3 种，每一种都带有不同的色素，分别对红光、绿光或者蓝光波长敏感，虽然这 3 种细胞的响应特性有重叠区域。总的说来，视锥细胞对波长在 555nm 左右的绿光区域最为敏感。这也就是为什么在 LCD 显示器中，绿色通道的分辨率高于红色和蓝色通道。

红色、绿色和蓝色视锥细胞的发现大大促进了*三色理论*的发展，该理论认为，任何一种有色光，可以通过不同比例的红光、绿光和蓝光的组合生成。

由于人眼含有的杆状细胞的数量要远多于视锥细胞，故眼睛对亮度的敏感度要高于对色彩的敏感度。这使得我们可以借助对色彩信息的*子采样*来节省视频和图像信息的带宽。

我们对亮度的感受特性是对数性的，而非线性的。换句话说，用于产生 50％灰度图（恰好在全黑和全白之间的正中）所需的实际的光强仅为我们需要产生全白图像所需的光强的 18％。这一特性在相机传感器和显示技术中尤为重要，正如我们将在后面的伽马校正中讨论的。此外，这一效应还将导致人眼对高亮度环境下的量化失真的感知度下降，导致这一特性被许多媒体编码算法所利用。

视觉方面的另一新奇之处在于，人眼可以适应环境，创建自己的白光参考，即使在低照明或者人工照明的情况下也是如此。因为摄像传感器自身并不具有这一特性，因此它需要使用参考量作为绝对白色，并对传感器进行调整，这一过程称为*白平衡控制*。

人眼对高频信息的敏感性要低于对低频信息。而且，虽然它可以检测出静态图像中细节和彩色部分的分辨率，但对于快速移动的图像，却无法做到这一点。于是，人们可以利用*变换编码*（DCT、FFT 等）以及低通滤波技术来降低呈现一幅图像或者视频序列时所需的总带宽。

当图像的刷新速率低于 50～60 次/s 时，我们的眼睛会感受到一种亮光“闪烁”的效应。在光线较暗的情况下，该频率值降低到 24Hz。此外，我们更倾向于观察到大而均匀的区域内的闪烁，相比之下，对局部区域的闪烁敏感度较低。这些特性对于*隔行*视频、刷新速率和显示技术具有重要的潜在作用。

何谓视频信号？

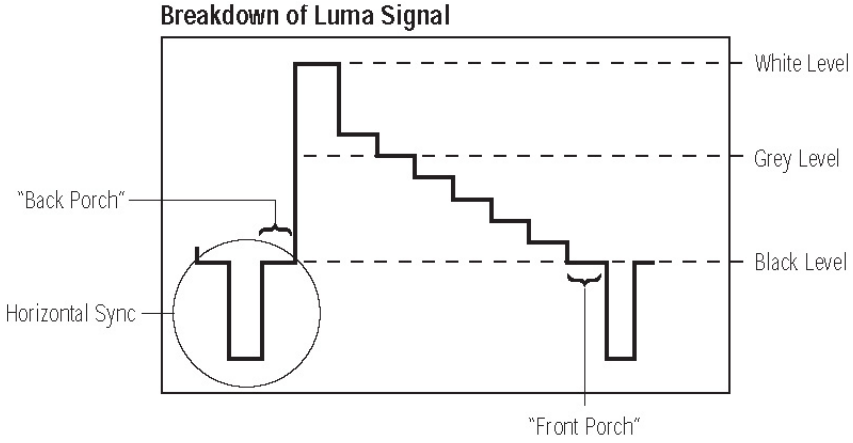


图 1　亮度信号的组成:

图中：Breakdown of Luma Signal——亮度信号的分类，Back Porch——后沿，Horizontal syc——水平同步，White level——白色级，Grey Level——灰色级，Black Level——黑色级

从根本上来说，一个视频信号基本上只是由亮度和色彩数据构成的 2 维阵列，该阵列以一定帧率的刷新变化来描述运动。在传统的阴极射线管（CRT）电视和显示器中，屏幕上的磷粉由一个电子束从上到下、从左到右的方式激发产生光亮。该电子束是由一个如图 1 所示的模拟视频信号调制生成。嵌入该模拟信号中的同步信号，决定了电子束什么时候激亮磷粉，什么时候停止操作。这样电子束可以在下一行由右向左回程扫描，或者从下到上开始对下一帧视频场或帧信号进行扫描。这些同步信号如图 2 所示。

HSYNC 是水平同步信号。它界定了视频帧每一行中（从左到右）有效视频的起始位置。*水平消隐*为电子枪从屏幕右侧回扫至下一行左侧的时间间隔。

VSYNC 是垂直同步信号。它定义了一个新的视频图像的起始位置（从上到下）。*垂直消隐*为电子枪从屏幕图像的右下角返回左上角所需的时间间隔。

FIELD 用于在隔行视频信号中区分出目前所显示的场。该信号并不适用于逐行扫描视频系统。

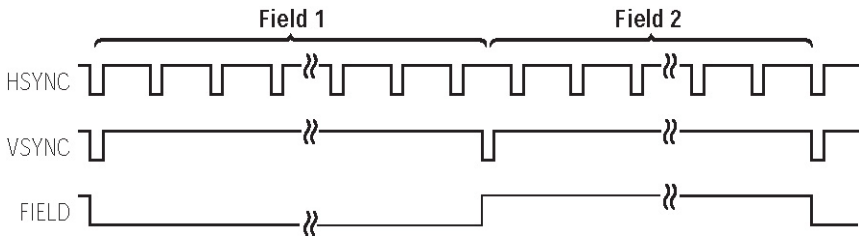


图2　HSYNC、VSYNC、FIELD 信号之间的时序关系

视频信息的传输起源于由黑到白的相关亮度显示，黑白电视系统也是这样产生的。在空间中的一个给定点处的电压水平则与该点图像的亮度水平相关。

当彩色电视出现后，它必须保证与黑白电视的后向兼容，因此彩色脉冲信息被添加到已有的亮度信号顶部，如图 3 所示。色彩信息也被称为色度。我们将在关于色彩空间的讨论中更多的探讨这一问题（见该系列文章的第 2 部分）。

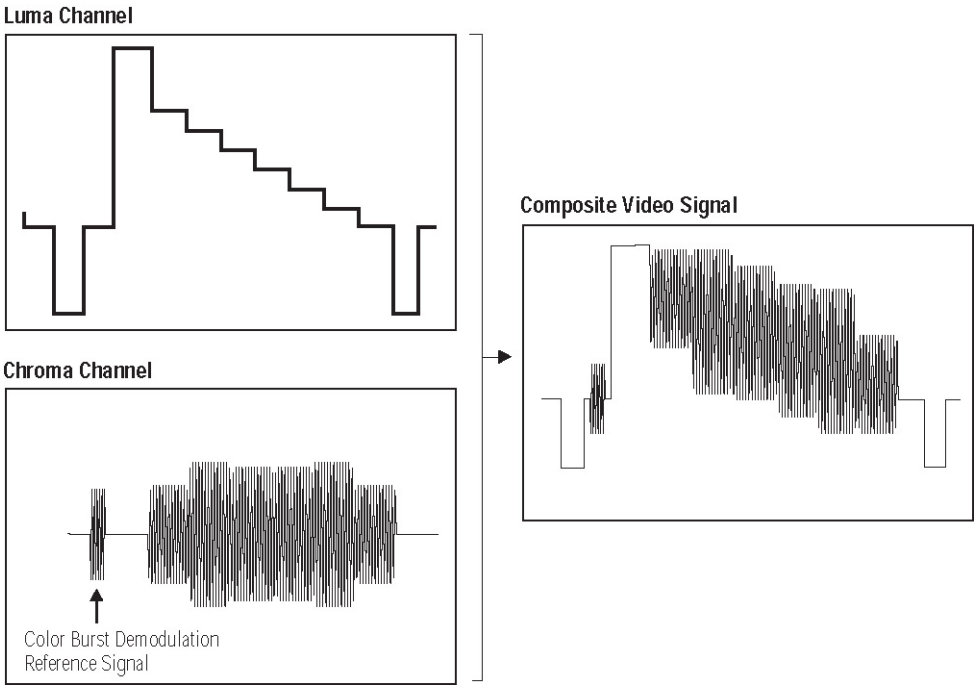


图3　　带色同步信号的模拟视频信号

图中：Luma Channle——亮度通道，Chroma Channel——色度通道，Composite Video signal——复合视频信号。

Color Burst Demodulation Reference Signal——色彩脉冲解调参考信号

广播电视——NTSC和PAL制式

模拟视频标准的区别在于它们各自对亮度和彩色信息的编码方式。目前广播电视领域占统治地位的是两种标准——NTSC 和 PAL。NTSC 由美国国家电视系统委员会提出，在亚洲和北美广泛使用，而 PAL 是 NTSC 的一个分支，在欧洲和南美占据统治地位。另外一种制式，SECAM，则在法国和东欧部分地区流行，不过，

在这些地区中，许多也都采用 PAL。我们的讨论将集中在 NTSC 制上，但讨论的结果也适用于基于 PAL 制的系统。

视频分辨率

水平分辨率是指图像每行的像素个数，而垂直分辨率则是指显示完整一帧时屏幕上出现的水平线的数量。标清 NTSC 系统采用隔行扫描方式，具有 480 线有效像素，每条线上有 720 个有效的像素（即总计 720×480 像素）。

高清系统常常采用逐行扫描方式，其水平和垂直分辨率要远高于标清系统。我们将专注于标清系统，而非高清系统，但我们讨论的大部分，也将推广到具有更高帧和像素传输率的高清系统。

在讨论视频技术时，分辨率和帧速率的提升是沿着两条主要的分支发展的，即计算机图形图像格式和广播视频格式。表 1 给出了各种常见的屏幕分辨率和帧率的比较。即使这两路分支源于不同的领域，而且要求也不同（例如，计算机图形显示使用 RGB 逐行扫描方法，而广播视频则使用 YCbCr 隔行扫描方法），如今在嵌入式领域，它们在使用上几乎可以是互换的。也就是说，VGA 与 NTSC“D-1”广播格式相当， QVGA 对应的则是 CIF。应该注意的是，虽然 D-1 是 720 像素×480 行格式，但它通常被称为 720×480 像素（这实际上是针对 DVD 和其他数字视频的 NTSC“DV”格式）。

视频源	视频标准	水平分辨率（像素）	垂直分辨率（像素）	总像素
广播	QCIF	176	144	25344
计算机图形	QVGA	320	240	76800
广播	CIF	352	288	101376
计算机图形	VGA	640	480	307200
广播	NTSC	720	480	345600
广播	PAL	720	576	414720
计算机图形	SVGA	800	600	480000
计算机图形	XGA	1024	768	786432
广播	HDTV（720P）	1280	720	921600
计算机图形	SXGA	1280	1024	1310720
计算机图形	UXGA	1600	1200	1920000
计算机图形	QXGA	2048	1536	3145728

表 1　图形图像与广播标准之间的对比

隔行和逐行扫描

隔行扫描方式源于早期的模拟电视广播技术，这种技术需要对图像进行快速扫描，以便最大限度地降低视觉上的闪烁感，但是当时可以运用的技术并不能以如此之快的速度对整个屏幕进行刷新。于是，将每帧图像进行“交错”排列或分为

两场，一个由奇数扫描线构成，而另一个由偶数扫描线构成，如图 4 所示。NTSC/（PAL）的帧刷新速率设定为约 30/（25）帧/秒。于是，大片图像区域的刷新率为 60（50）Hz，而局部区域的刷新率为 30（25）Hz，这也是出于节省带宽的折中考虑，因为人眼对大面积区域的闪烁更为敏感。

隔行扫描方式不仅会产生闪烁现象，也会带来其它问题。例如，扫描线本身也常常可见。因为 NTSC 中每场信号就是 1/60s 时间间隔内的快照，故一幅视频帧通常包括两个不同的时间场。当正常观看显示屏时，这并不是一个问题，因为它所呈现的视频在时间上是近似一致的。然而，当画面中存在运动物体时，把隔行场转换为逐行帧（即解交织过程），会产生锯齿边缘。解交织过程非常重要，因为将视频帧作为一系列相邻的线来处理，这将带来更高的效率。

随着数字电视的出现，逐行（即非隔行）扫描已经成为一种具有更高图像品质的流行的输入和输出视频格式。在这种方式下，整幅图像将从上到下依次刷新，其扫描速率约为相应隔行系统的扫描速率的两倍，这消除了隔行扫描产生的许多弊病。在逐行扫描中，由两场信号来表示一帧视频的方式不再使用。

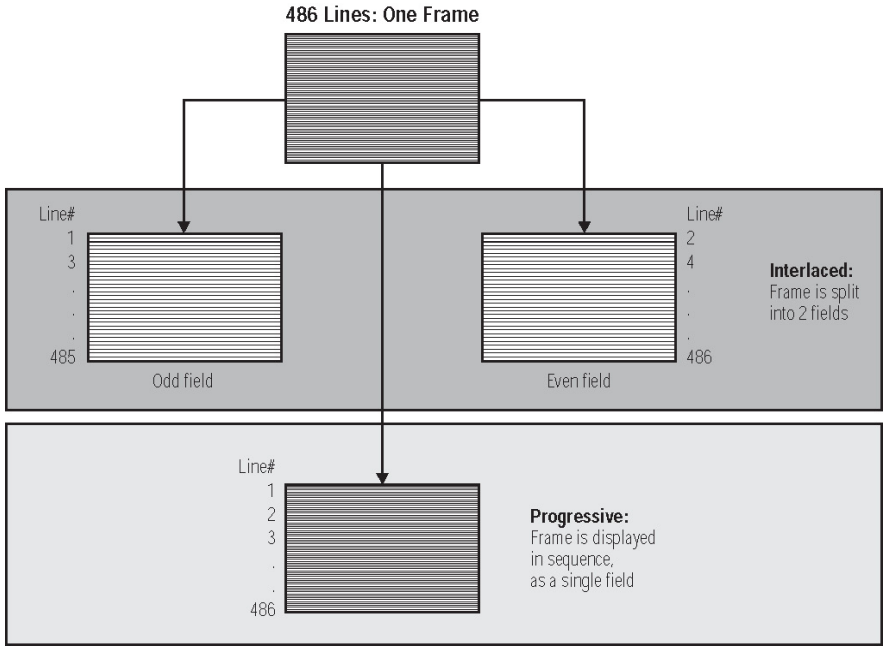


图 4：隔行扫描与逐行扫描方式的对比

图中：486 Lines：One Frame——486 线：1 帧

Line——行，Interlaced：Frame is split into 2 field——隔行:图像帧被分离为两个视场；Progressivew：Frame is displayed in sequence as a single field——逐行:图像帧作为一个视场依序显示；

我们已经简要地讨论了视频信号的基础以及某些常用的术语，接下来，我们将在下一章节开始讨论令人感兴趣的部分——数字视频技术。

Part-2

本文是 5 部分系列文章的第 2 部分，在本文中，我们将讨论数字视频的基本概念。在讨论前，我们需要对与颜色空间有关的几个问题进行一番讲解。

颜色空间

颜色的表达有多种不同的方式，每一种颜色系统所适合的用途都各不相同。最基本的一种表达方式为 RGB 颜色空间。

RGB 代表“红－绿－蓝，”它是相机传感器和计算机图形显示方面常用的一种颜色系统。由于这三种原色相加起来可以形成白光，故可以通过将各原色按不同比例进行调和的办法来形成可见光谱区的大多数颜色。RGB 是所有其他颜色空间的基础，在计算机图形学中，它是颜色空间的首选。

Gamma 校正

在处理与颜色空间有关的问题时，“Gamma（γ）”是一种需要弄懂的关键现象。该术语描述了人们对亮度的感受和显示本身存在的非线性。请注意，这种现象表现在两方面：人眼对亮度的感受是非线性的，而物理输出设备（例如 CRT 和 LCD）对亮度的显示也是非线性的。人们发现，可谓巧合的是，人的视觉对亮度的灵敏度特性几乎恰好与 CRT 的输出特性相反。

换句话说，显示器的亮度大约与输入的模拟信号电压的γ次方成正比。在 CRT 或者 LCD 显示器上，该值一般为 2.2~2.5。因此，相机的预补偿功能，是让 RGB 的量值按照 1/γ次方的关系来变化。

该效应所带来的影响是，视频摄像机和计算机图形学程序，通过一种被称为“Gamma 校正”的流程，可以预先对其 RGB 输出流进行预校正，以便补偿所针对的显示器的非线性，并就眼睛实际感受场景的方式形成一种有现实意义的模型。图 1 示出了这样一种流程。

经过 Gamma 校正后的 RGB 坐标被称为 R’G’B 空间，其中亮度值 Y 可以从这些座标中提取出来。严格来讲，“Luma”一词应该仅指这类经过“Gamma 校正”的亮度值，而真正的“亮度（luminance）”Y 是一个颜色科学方面的术语，它是从 R、G 和 B 的加权和（未经过 Gamma 校正）所获得的。

在本系列文章中，当我们谈论 YCbCr 和 RGB 颜色空间时，我们是指经过 Gamma 校正的分量，换句话说，Y’CbCr 或者 R’G’B’。不过，因为该表示方法会造成我们的困惑，而且并不影响我们的讨论，既然 Gamma 校正必须在传感器和/或显示器与处理器的接口上执行，我们就将仅限于采用 YCbCr/RGB 的说法，即便在完成 Gamma 校正之后也是如此。这一约定的一个例外，是对实际的颜色空间变换方

程进行讨论的时候。

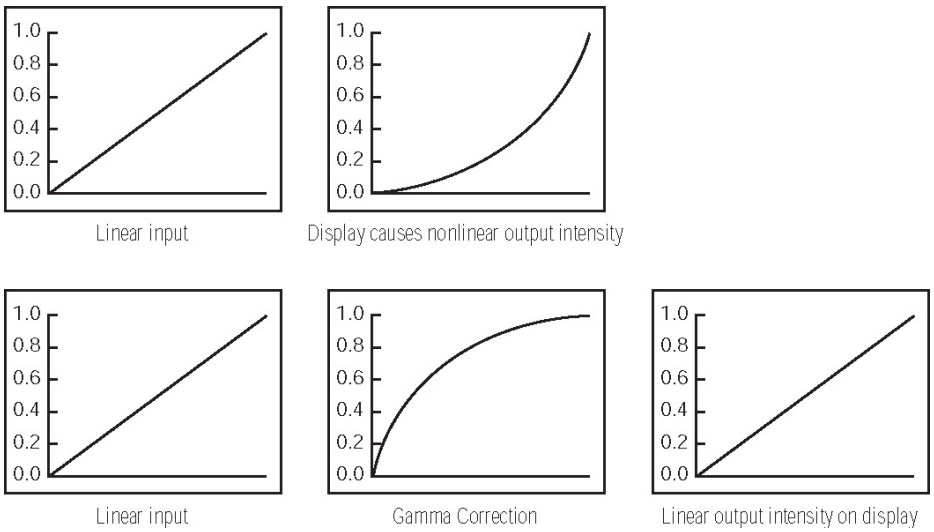


图1. Gamma 校正可实现针对一个给定的输入幅值所产生的光强信号进行线性化
图中：Linear input——线性输入；Display causes nonlinear output intensity——显示器造成了输出强度的非线性；Linear input——线性输入；Gamma Correction——Gamma 校正；Linear output intensity on display——显示器上线性化的输出强度特性。

虽然 RGB 通道格式是呈现现实世界颜色的一种自然而然的方案，但 3 个通道中的每一个都与另外两个高度相关。你独立观看一幅特定图像的 R、G 和 B 通道，就可以发现这一点——你在每个通道中都能感受到整幅图像。另外，RGB 并非图像处理的最佳选择，因为如果要变动一个通道，则也必须在另外两个通道进行更改，而且每个通道的带宽相同。

为了减少所需要的传输带宽并提高视频压缩比，人们提出了其他的颜色空间方案，这些变量是高度非相关的，从而能提供优于 RGB 的压缩特性。其中最流行的一些方案——YPbPr、YCbCr 和 YUV——全都是将亮度信号分量与两个色度分量分离开。这种分离运算是借助等比例缩放的色差因子（B’-Y’）与(R’-Y’)来实施的。Pb/Cb/U 等项对应着（B’-Y’）因子，Pr/Cr/V 等项对应于（R’-Y’）参数。YPbPr 用于分量化的模拟视频中，YUV 则适用于复合的 NTSC 和 PAL 系统，YCbCr 则与分量化数字视频有关。

亮度和色度信息的分离，可以节省图像处理带宽。另外，正如我们马上就会看到的那样，我们可以通过子采样的方法，在视觉效果不会出现较大损失的前提下，大大减小色度信号带宽。这对于需大量处理视频数据的系统来说，是一个受欢迎的特色。

作为在两个颜色空间之间进行转换的一个实例，如下的方程示出了如何在 Y’CbCr 和 R’G’B’颜色空间的 8bit 表示法之间进行转换的方法，式中，Y’，R’，G’ 和 B’通常的变化范围是 16～235，而 Cr 和 Cb 的变化范围是 16～240。

Y′ = (0.299)R′ + (0.587)G′ + (0.114)B′

Cb = -(0.168)R′ - (0.330)G′ + (0.498)B′ + 128

Cr = (0.498)R′ - (0.417)G′ - (0.081)B′ + 128

R′ = Y′ + 1.397(Cr - 128)

G′ = Y′ - 0.711(Cr - 128) - 0.343(Cb - 128)

B′ = Y′ + 1.765(Cb - 128)

色度子采样

由于人眼的杆状细胞要多于锥状细胞，故对于亮度的敏感能力要优于对色差的敏感能力。幸运的是（或者事实上通过设计可实现的是），YCbCr 颜色系统允许我们将更多的注意力投向 Y，而对 Cb 和 Cr 的关注程度不那么高。于是，通过对这些色度值进行子采样的方法，视频标准和压缩算法可以大幅度缩减视频带宽。

在对该问题进行进一步讨论之前，让我们先解析一些术语。假设在进行子采样前，我们拥有一路全带宽的 YCbCr 流，即视频源产生了一路像素分量构成的数据流，其形式如图 2a 所示。这被称为“4:4:4 YCbCr”。这一表示方法看起来甚为离奇，但是可以简单的解释如下：第一个数始终是‘4’，在历史上对应着亮度采样频率与 NTSC 彩色子载波频率之比。第二个数字对应着在某条给定的水平线上的亮度和色度之比：如果并未针对亮度信号来对色度进行下采样，则该数为‘4’。第三个数，如果与第二个数相同的话，则意味着没有垂直方向上的色度子采样。另一方面，如果它等于 0，则两条线之间存在一次 2:1 的色度子采样。于是，4:4:4 意味着在每条线上的像素都有自己独有的 Y、Cr 和 Cb 信号分量。

现在，如果我们通过在水平方向上对色度信号进行因数为 2 的子采样，对一个 4:4:4 YCbCr 信号进行滤波，于是将得到 4:2:2 YCbCr。‘4:2:2’意味着在给定的视频行上对应 2 个色度值有 4 个亮度值。每个(Y,Cb) 或者 (Y,Cr)对都代表着一个像素值。另一种表述方法是：一个色度信号对（chroma pair）在空间上与每隔一个亮度所选取的亮度值相重合，如图 2b 所示。无论您相信与否， 若将 4:2:2 YCbCr 与对应 的 4:4:4 YCbCr 原始信号进行比较，则会发现在图像质量方面出现的损失很小，虽然它相对于 4:4:4 YCbCr 而言在带宽方面缩小了 33%。正如我们马上要讨论的那样，4:2:2 YCbCr 是 ITU-R BT.601 视频推荐方案的一个基础，它是数字视频信号在不同的子系统组件之间进行传输时最常用的一种格式。

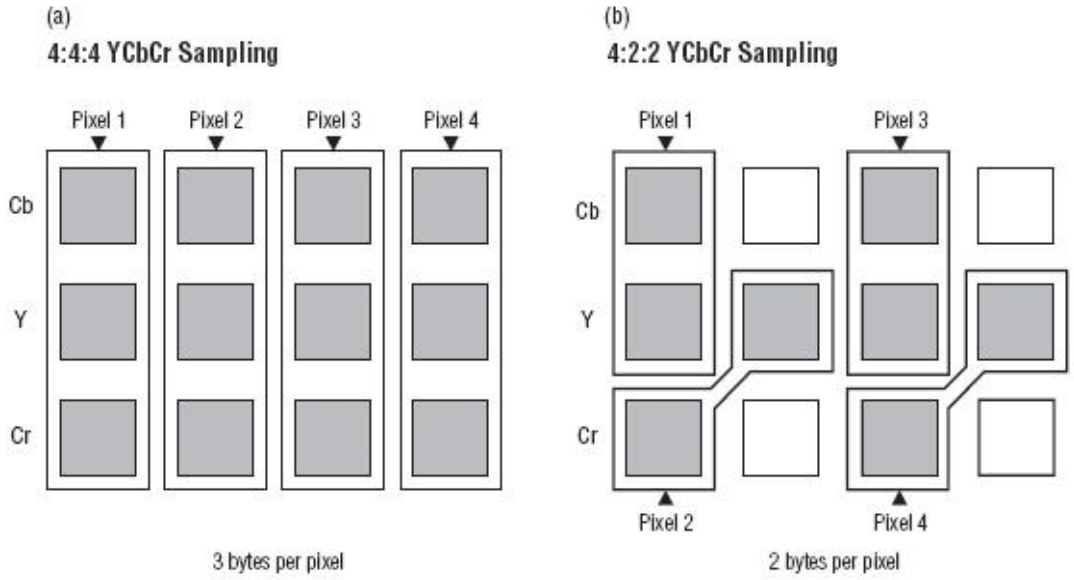


图 2： a） 4:4:4 与 b) 4:2:2 YCbCr 像素采样的比较
图中：Sampling——采样，3bytes per pixel——每像素 3 个字节，2bytes per pixel——每个像素 2 个字节。

应注意的是，4:2:2 并非是唯一的色度子采样方案。图 3 示出了得到广泛应用的其他一些方案。例如，我们可以如图 3c 所示的那样，以比例因数“4”来对一个 4:4:4 YCbCr 流的水平行上的色度进行子采样，从而得到 4:1:1 YCbCr 流。这里，色度对在空间上与每个以 4 为间隔的亮度值相一致。这一色度过滤方案可以节省 50％的带宽。4:1:1 YCbCr 是一种在视频压缩算法的输入和视频解压缩算法的输出方面广受欢迎的格式。

另外一种在视频压缩/解压缩方面得到广泛应用的算法是 4:2:0 YCbCr，由于几个原因，它要比我们已经阐述过的其他方案更为复杂。其中一个原因是，Cb 和 Cr 分量均在水平和垂直方向上受到了因数为 2 的子采样。这意味着我们不得不存储多条视频线，才能产生出该子采样流。而且，对应 4:2:0 YCbCr 信号还存在两种广受欢迎的格式。MPEG-2 压缩使用了一种水平方向上共位的方案（图 3d，顶图），而 MPEG-1 和 JPEG 算法所采用的格式则是一种色度信号位于 Y 采样之间的中点上的方案（图 3d，底图）。

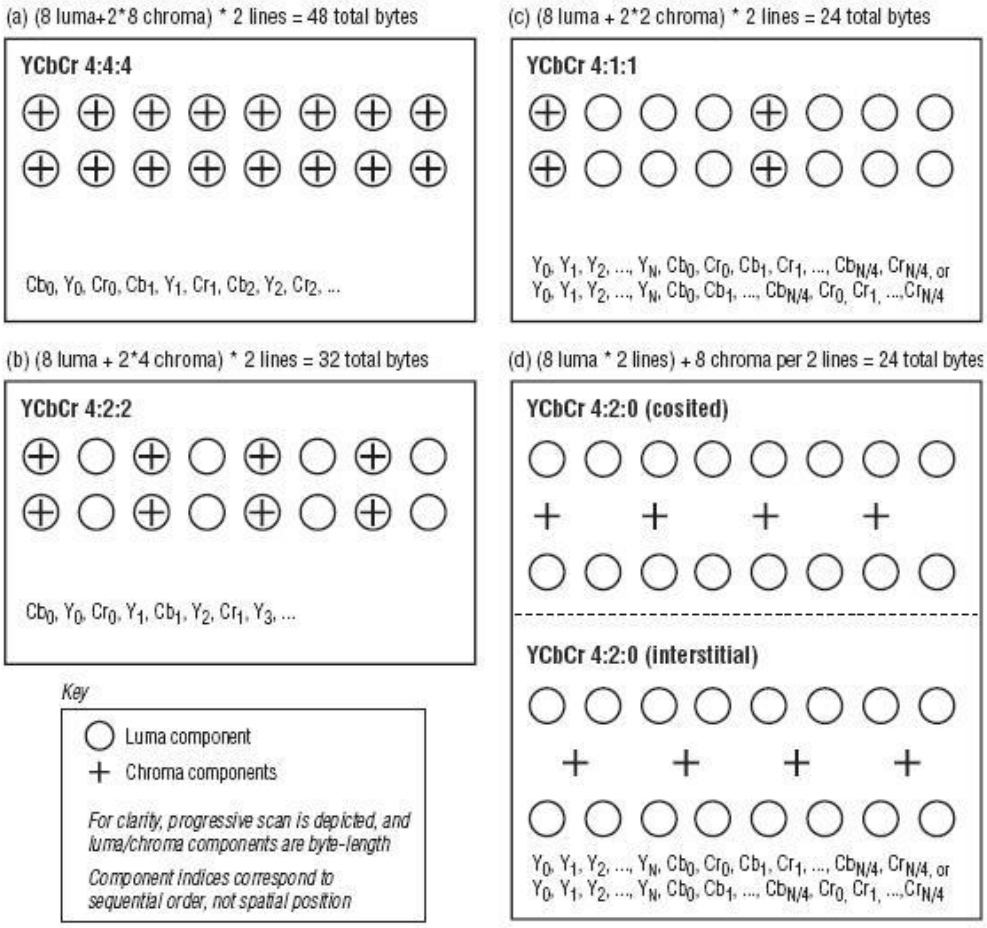


图3：（a） YCbCr 4:4:4流以及经过色度子采样后的导出信号；（b）4:2:2；（c）4:1:1； (d) 4:2:0
图中：Luma——亮度，Chroma——色度，Total bytes——总字节数；
Luma +Chroma Component——亮度＋色度分量，Cosited——共位的；Interstitial——间插的。
左下脚说明文字：为了清楚起见，描述了逐行扫描，亮度/色度分量都具有1个字节的长度；分量的标记对应着串序化的次序，而并非空间位置。

数字视频

在上世纪 90 年代中期以前，几乎所有的视频都采用了模拟格式。正是在那之后，诸如 MPEG-2 压缩的出现、互联网上流媒体的繁荣以及 FCC 采用“数字电视（DTV）”标准等推动力形成了一场“完美风暴”，将数字视频表现方式的优点呈现给整个视频世界。这些超越模拟信号的优点包括：更好的信噪比性能，更好的带宽利用（可以将若干路数字视频通道融入每个现有的视频频道中），而且通过数字压缩技术来减少存储空间。

从根本上来说，视频的数字化同时包括了对模拟视频信号的采样和量化。在视频帧的 2D 框架之中，采样最终体现为将栅格状的图像空间划分为小块的区域，并根据每个区域中颜色空间分量的强度来为其分配相对幅值。请注意，模拟视频信号已经被从垂直方向上（离散的多行扫描）和时间上（每秒分立的多帧图像）

做了采样。

量化是指在采样过程中设法确定这些离散幅值的过程。8bit 视频是消费类应用中常用的格式，在每个颜色通道（R、G、B 或者 YCbCr）中，0 代表最暗（全黑），255 代表最亮（白）。不过，应该指出的是，每单色通道 10bit 和 12bit 的量化水平，也正在迅速融入主流的视频产品中，从而带来更高的精度，这对于避免其截断误差从而降低接收到的图像噪声来说，是非常有效的。

数字视频的出现，在很大程度上为 NTSC 和 PAL 系统接口的标准化提供了极佳的机会。当 ITU（国际电信联盟）开会以确定关于数字视频标准方面的推荐方案时，它把重点放在如何在 NTSC 和 PAL 格式之间实现高度的共享性，这样使得两种标准都可以采用同一种编码格式。

他们确定了 2 种独立的推荐方案－ITU-R BT.601 和 ITU-R BT656。这两种方案合起来，就定义了一种结构，该结构能让不同的数字视频系统部件实现互操作。鉴于 BT.601 定义了数字视频传输所用的参数，BT.656 定义了接口本身。

ITU-R BT.601（前 CCIR-601）

BT.601 规定了对视频信号进行数字化编码的方法，它利用了 YCbCr 颜色空间，以更好地利用通道带宽。它建议将 4:2:2 YCbCr 作为广播视频的首选格式。同时也提供了同步信号（HSYNC，VSYNC，FIELD）和时钟信号，以便划定有效视频区的边界。图 4 示出了同步信号、时钟和数字信号之间典型的时序关系。

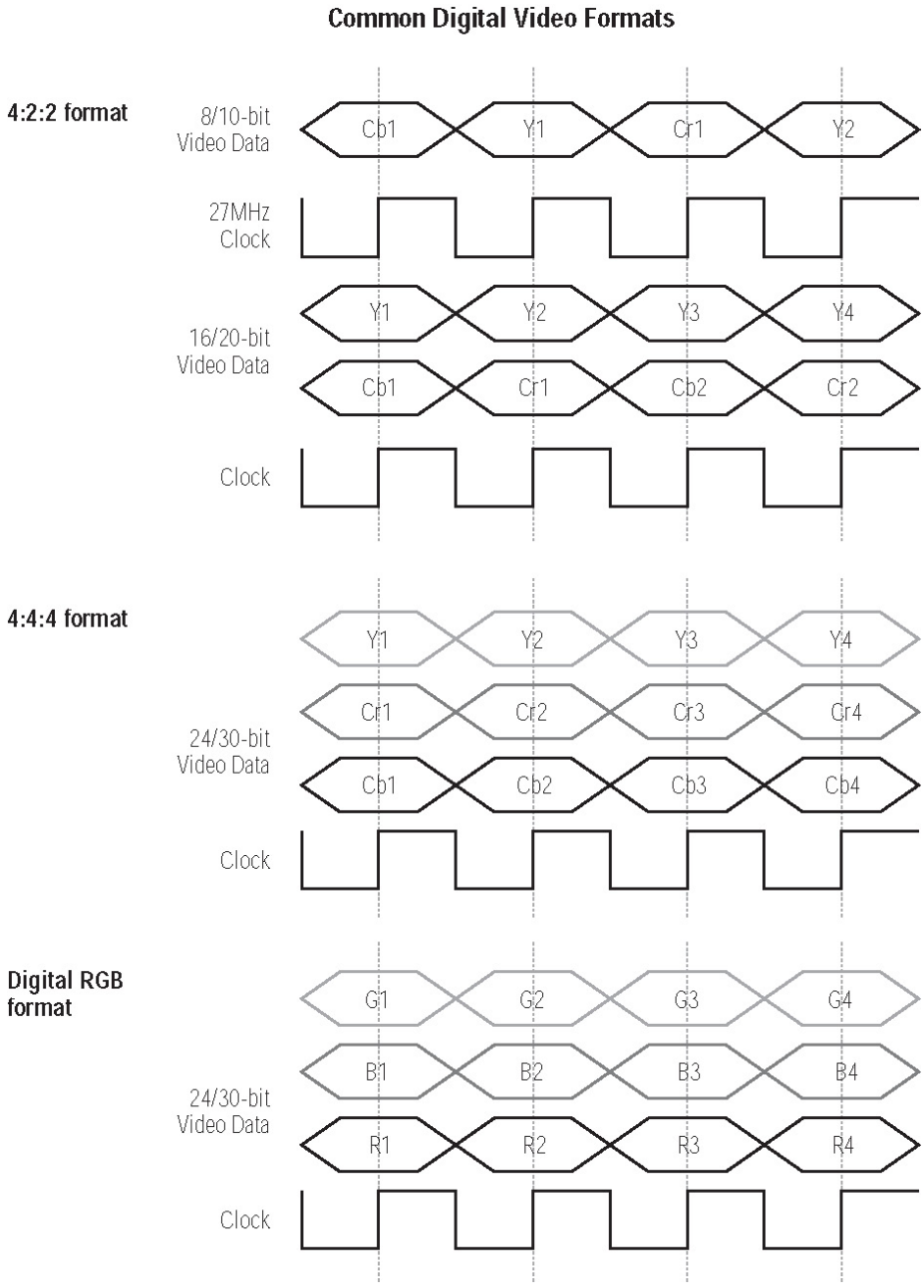


图4　常见的数字视频格式的时序关系

Video Data——视频数据，Clock——时钟信号，format——格式，Digital RGB format——数字RGB格式。Video Data——视频数据，Clock——时钟。

每个 BT.601 像素分量（Y、Cr 或 Cb）被量化为 8 或者 10bit 信息，NTSC 和 PAL 的有效视频画面中每行有 720 个像素 。不过，它们在垂直分辨率方面存在差异。30 帧/s 的 NTSC 有 525 行（线）（包括垂直消隐或者回扫区），而通过在 PAL 帧上添加 100 条线，或者说，使之总共达到 625 条线，使 PAL 的 25 帧/s 的速率适应同样的标准。

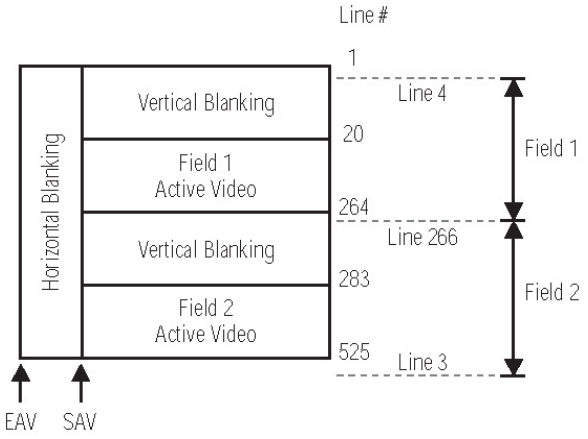
BT.601规定Y值的额定值范围从16（全黑）一直到235（全白）。颜色分量Cb和Cr则从16到240，但128的量值对应着无颜色。有时，由于噪声或者截断误差的存在，一个量值可能会超出额定值边界之外，但永远不会取值到0或者255。

ITU-R BT.656 （前 CCIR-656）

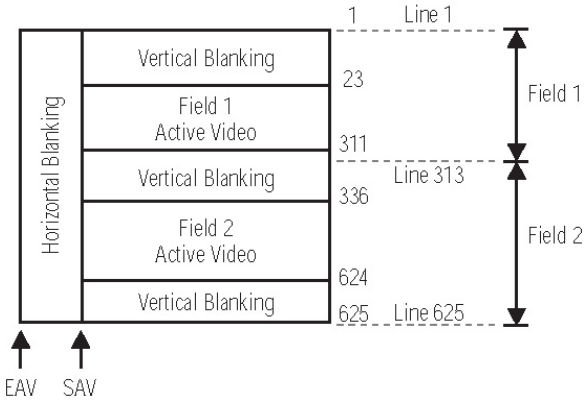
BT.601规划了对视频进行数字编码的方法，而BT.656则实际定义了实施BT.601所必需的物理接口和数据流。它同时定义了位并行和位串行模式。位并行模式只需要27MHz的时钟（在NTSC 30 帧/s条件下）以及8或10条连线（具体取决于像素的分辨率）。所有的同步化信号都嵌入到数据流中，因此无需额外添加硬件连线。

位串行模式只需要在单个通道上传输一路复用化的10bit/像素串行数据流，不过它需要运用复杂的同步化、频谱整形和时钟恢复调理等技术手段。此外，其位时钟速率接近300MHz，因此要在很多系统中实施基于采用串行位形式的BT.656是极富挑战性的任务。从我们的目标出发，我们将把注意力仅放在位并行模式上。

图5和图6示出了ITU-R　BT. 656中分别针对525/60 （NTSC）和625/50 （PAL）系统的帧划分方法和数据流的特性



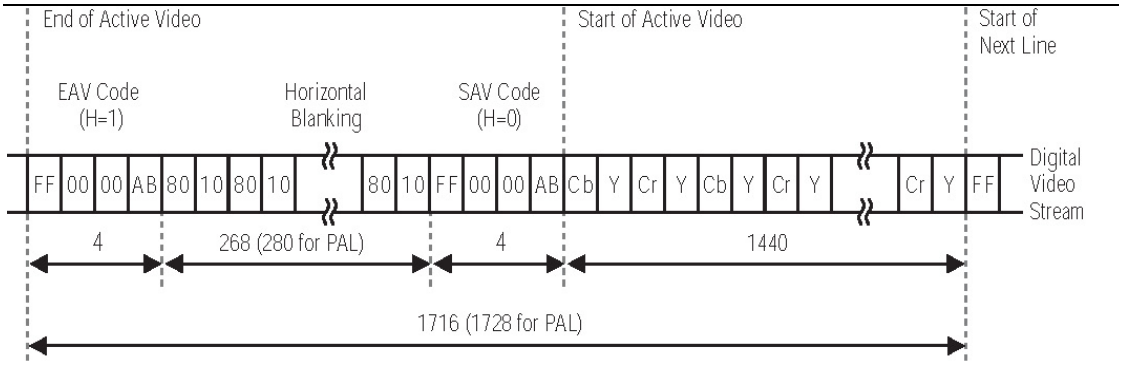
Line Number	F	V	H (EAV)	H (SAV)
1-3, 266-282	1	1	1	0
4-19, 264-265	0	1	1	0
20-263	0	0	1	0
283-525	1	0	1	0



Line Number	F	V	H (EAV)	H (SAV)
1-22, 311-312	0	1	1	0
23-310	0	0	1	0
313-335, 624-625	1	1	1	0
336-623	1	0	1	0

图5　ITU－R BT.656 帧划分

图中：horizontal blanking——水平消隐，Vertical Blancking——垂直消隐，Field——场，Active Video——有效的视频，



AB = Control Byte

图6： ITU－R BT.656数据流

End of active video——有效视频终点；horizontal blanking——水平消隐，Start of active video——有效视频起点，Start of next line——下一行的起点。Digital Video Stream——数字视频流。Control Byte——控制字节。

在BT.656标准中，水平（H）、垂直（V）和场（F）信号作为嵌入到视频数据流中的一串字节来发送，这一串字节构成了一个控制字。有效视频起点（SAV）和有效视频终点（EAV）信号指示了每行读入的数据单元的开始和结束。SAV出现在H发生1-0切换时，EAV出现在H发生0-1切换时。整个视频场由有效的视频＋水平消隐（EAV和SAV代码之间的空间）以及垂直消隐（V＝1的空间）组成。

视频的场从F位的切换开始。“奇数场”由F＝0表示，而F＝1则表示偶数场。逐行扫描的视频并不区分场1和场2，而隔行扫描的视频则要求专门对每个场进行独立的处理，因为每个场交替的扫描行组合起来最终形成实际的视频图像。

图7更为详细地示出了SAV和EAV代码。请注意，视频数据有一个由三个字节构成的前导码（8bit视频是0xFF, 0x00，0x00，而10bit视频则是0x3FF, 0x000, 0x000），后面跟随着XY状态字，这个字除了包含F （场）, V （垂直消隐） 和 H （水平消隐）位之外，还包含了4个保护位，以实现单位错误的检测和纠正。请注意，F和V只能作为EAV序列的一部分来变化（即，从H = 0切换到H = 1）。此外，请注意，对于10bit视频来说，增加的两位实际上是最低位，而不是最高位。

	8-bit Data								10-bit Data	
	D9 (MSB)	D8	D7	D6	D5	D4	D3	D2	D1	D0
Preamble	1	1	1	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0
Control Byte	1	F	V	H	P3	P2	P1	P0	0	0

P3:P0 = Error detection/correction bits

图7 SAV/EAV 前导码

图中：8－bit Data——8位数据，10bit Data——10位数据

各个数位的定义如下：

• F = 0 for Field 1
• F = 0 ，场 1
• F = 1 for Field 2
• F = 1， 场 2
• V = 1 during Vertical Blanking
• V = 1 垂直消隐期间
• V = 0 when not in Vertical Blanking
• V = 0 未在垂直消隐期内
• H = 0 at SAV
• H = 0 @ SAV
• H = 1 at EAV
• H = 1 @ EAV
• P3 = V XOR H
• P2 = F XOR H
• P1 = F XOR V
• P0 = F XOR V XOR H

垂直消隐间隔（V＝1的时间）可以被用来发送非视频的信息，如音频、文字电视广播（teletext）、字幕（closed-captioning）或者甚至交互电视应用所需的数据。BT.656借助辅助性的数据包实现这些功能。这些辅助性数据包并未采用通常在控制代码前的那些“0xFF, 0x00, 0x00”前导码，而是以“0x00, 0xFF, 0xFF”前导码为开头。

假定并不发送辅助数据，则在水平和垂直消隐间隔期间，（Cb, Y, Cr, Y, Cb, Y, …）流是（0x80, 0x10, 0x80, 0x10, 0x80, 0x10…）。另外，请注意，由于0x00 和 0xFF 等量值专门用于控制前导码，故不能用作有效视频流的一部分。在10bit系统中，（0x000 到 0x003）以及（0x3FC 到0x3FF）等量值也都被专门留出，以免给8bit的视频引用造成问题。

上述就是我们关于数字视频的概念介绍。在下一部分中，我们将把注意力放在如何从系统角度来考察视频技术上，讨论视频流是如何进入和输出嵌入式系统的。

Part-3

本文是 5 部分系列文章的第 3 部分，我们将从系统层次来考察视频流，讨论嵌入式视频应用所包含的各种视频源和显示装置。

图 1 描述了一个典型的端到端嵌入式数字视频系统。在这种情况下，一个视频源被输入到一个媒体处理器中（必要时，可经过视频解码器的数字化处理）。此时，可以通过软件编码操作来对其进行压缩，然后将其存储到本地或者通过网络进行传输。

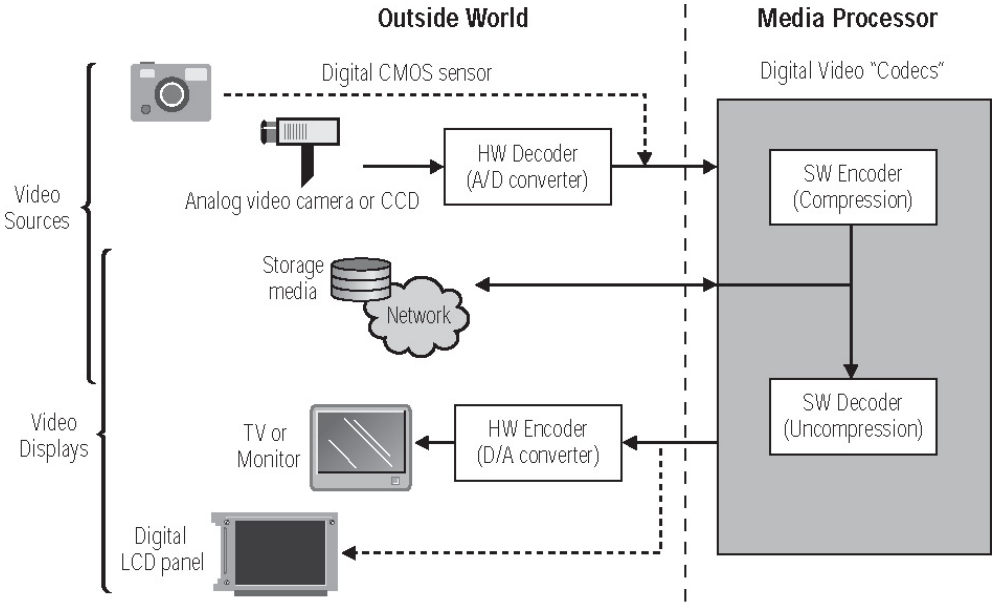


图 1: 模拟/数字源和显示器的系统视频流
Outside World——媒体处理器外部，Media Processor——媒体处理器 Video Soures——视频源，Video Displays——视频显示设备，Digital LCD panel——数字 LCD 平板，Digital CMOS sensor——数字 CMOS 传感器，Analog Video camera or CCD——模拟视频摄像机或者 CCD，HW Decode（A/D converter）r——硬件解码器（A/D 转换器），Storage media——存储介质，TV or Monitor—— 电视机或监视器，Digital LCD panel——数字 LCD 平板显示，H/W Encoder（D/A 转换器）——硬件编码器（D/A 转换器），Media Processor——媒体处理器，SW Encoder（Compression）——软件编码器（压缩），SW Decoder——软件解码器（解压缩）

与此过程相反，我们可以从网络或硬盘存储设备得到一段经过压缩的码流。然后通过软件解码操作实现解压缩，并直接传送到一个视频输出显示设备器上（如 TFT－LCD 平板），或者通过视频编码器转换为模拟信号，从而在传统的 CRT 上显示。

需要注意的是，压缩/解压缩仅仅代表了媒体处理器上可实现的视频处理算法中的一部分。不过，就我们的目标而言，它们为我们的讨论提供了一个合适的模板。接下来，让我们更为详细的探讨这些数据流中专门针对视频的部分。

模拟视频源

嵌入式处理器内核无法直接对模拟视频信号进行处理。视频信号必须首先通过视频解码器数字化，将模拟视频信号（例如，NTSC、PAL、CVBS、S-Video）转换为数字信号形式（通常是ITU-R BT.601/656 YCbCr或者RGB）。这是一个复杂的、多级的处理过程，包括从输入信号中提取时间信息、亮度与色度的分离、色度信息分离为Cr和Cb分量、输出数据的采样，以及为其分配适当的格式等。通过串行接口，如SPI或者I^2C，可以对解码器的操作参数进行配置。图 2 所示的是视频解码器的典型方框图。

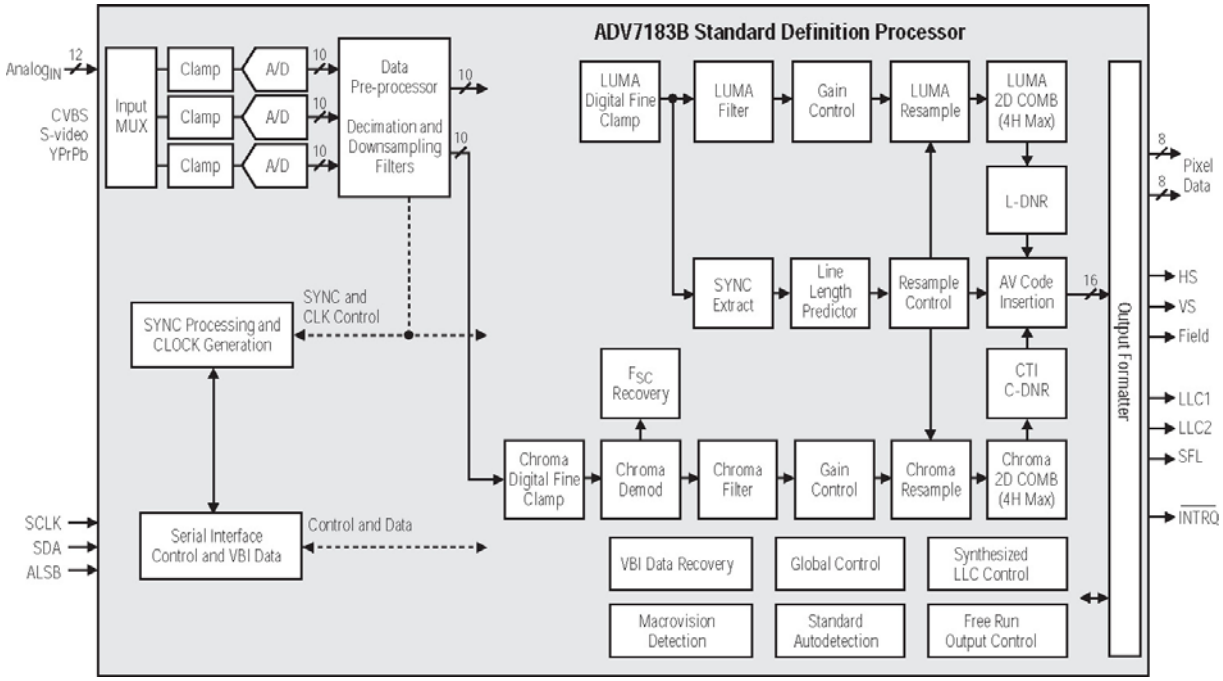


图 2　ADV7183B 视频解码器，

图中：Analog IN——模拟输入，Input MUX——输入复用器，Data Pre-processor——数据预处理器，Decimation and Downsampling Filters——抽取和下采样滤波器，SYNC and CLK——同步与时钟，SYNC Processing and CLOCK Generation——同步处理和时钟生成，Serial Interface Control and VB1 Data——串行接口控制和VBI数据，Chroma Digital Fine Clamp——Chroma数字精细嵌位电路，LUMA Digital Fine Clamp——LUMA数字精细嵌位电路，LUMA Filter——LUMA滤波器，Gain Control——增益控制，LUMA Resample——LUMA重采样，LUMA 2D COMB（4H MAX）——LUMA 2D COMB，SYNC Extract——同步提取，Line Length Predictor—数据线长度预测，Resample Control——重采样控制，AV Code Insertion——AV码插入，F_{SC} Recovery——F_{SC} 恢复，Chroma Demod—Chroma解调，Chroma Filter——Chroma滤波器，Gain Control——增益控制，Chroma Resample——Chroma重采样，Chroma 2D COMB （4H MAX）——Chroma 2D COMB（4H MAX），VBI Data Recovery——VBI数据恢复，Macrovision Detection——Macrovision 检测，Global Control——全局控制，Standard Autodetection——标准自动检测，Synthesized LLC Control——同步LLC控制，Free Run Output Control——自由运行输出控制，Output Formatter——输出格式化。

数字视频源

当今的视频信号源基本上都是基于电荷耦合设备（CCD）或者 CMOS 技术基础上的。这些技术都可以将光转换为电信号，但它们在转换机理方面存在差异。

CMOS 传感器一般会输出并行的数字信号流，该信号流通常包括 YCbCr 或者 RGB 格式的像素分量，以及水平/垂直同步和像素时钟。有时，它们还允许采用一路外部的时钟和同步信号，以控制图像帧从传感器向外部传输。

另一方面，CCD 往往连接到 “模拟前端”（AFE）芯片上，如 AD9948，该类芯片处理模拟输出信号，对其进行数字化，并产生恰当的时序信号来扫描 CCD 阵列。AFE 的同步信号则由处理器来提供，AFE 需要利用该路控制信号来管理 CCD 阵列。经过 AFE 数字化后的并行输出流可以达到每像素分量 10bit、甚至 12bit 的分辨率。

欲了解关于在 CMOS 和 CCD 传感器之间如何进行折衷的更为详细的讨论，以及典型的图像处理流水线操作方面的综述，请参考如下的文章：

http://www.videsignline.com/howto/sensorsoptics/189600793

模拟视频显示

视频编码器

视频编码器用于将数字视频流转换为一路模拟视频信号，输入一般为ITU-R.656或者BT.601 格式的YCbCr或者RGB视频流，根据不同的输出标准（如NTSC、PAL、SECAM）对信号进行转换。一个主控处理器可以通过 2 线或者 3 线串行接口（SPI 或者I^2C）来对编码器进行控制，如对像素时序、输入/输出格式以及亮度/色度滤波等设置进行编程。图 3 所示的是典型编码器的结构框图。视频编码器比较常见的模拟输出格式如下：

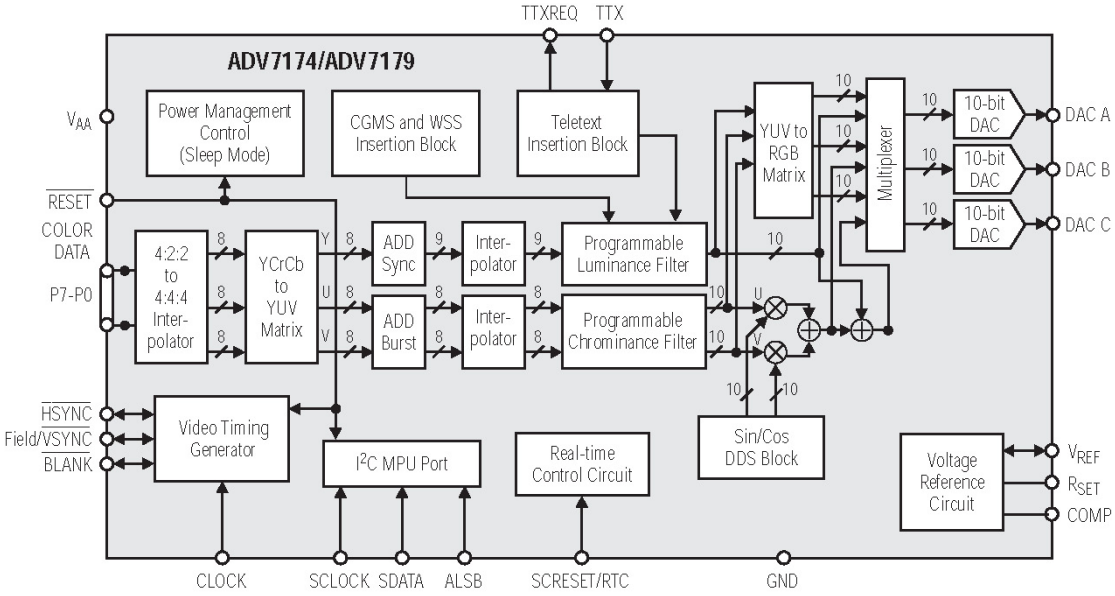


图3：ADV7179 视频编码器的结构框图。

图中 Power Management Control——电源管理控制，Sleep Mode——休眠模式，Interpolator——内插器，CGMS and WSS Insertion Block ——CGMS 与 WSS 插入模块，Teletext Insertion Block——Teletext 插入模块，Interpolator——内插器，Programmable Luminance Filter——可编程亮度滤波器，

Programmable Chrominance Filter——可编程色度滤波器，YCrCb to YUV Matrix——YCrCb至YUV转换矩阵，Video Timing Generator——视频时序发生器，I²C MPU Port——I²C MPU端口，Real－time Control Circuit——实时控制电路，Sin/Cos DDS Block——正弦/余弦DDS模块，Voltage Reference Circuit——电压基准电路，YUV to RGB Matrix——YUV至RGB转换矩阵，Multiplexer——复用器，Voltage

Reference Circuits——电压基准电路

CVBS：复合视频基带信号（或复合视频消隐与同步）。复合的视频是通过图 4a 所示的专用黄色 RCA 接头来连接的。它将亮度、色度、同步和色彩脉冲信息整合到一根电缆内。

S Video：使用图 4b 所示的接头进行连接，可以分别传送亮度和色度内容。将亮度信息与色差信号分离开来，可以大幅改善图像质量，这也正是 S Video 连接在当今的家庭影院系统中流行的原因。

分量视频，也称为 YPbPr，这是 YCbCr 数字视频的的模拟版本。在这种视频中，每个亮度与色度通道都是单独提取、输出的，每路都带有自己的时序。这就保证了模拟传输后图像的高品质。分量连接在高端家用影院系统组件，如 DVD 播放器和 A/V 接收机中，是非常常见的（图 4c）。

模拟 RGB 具有分离的红、绿、蓝信号通道。这可以提供类似于分量视频的图像质量，但它一般用于计算机图形图像领域，而分量视频则主要应用

于消费类电子方面。RGB 连接器往往是 BNC 插座的改型，如图 4d 所示。

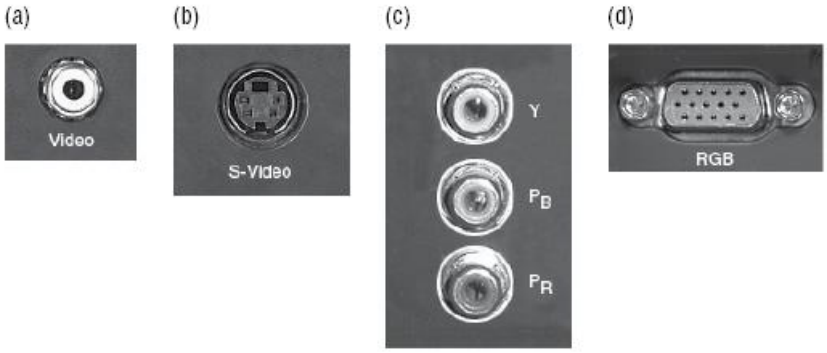


图 4：常见的模拟视频连接器

阴极射线管(CRT)

在显示端，RGB 是计算机监视器和 LCD 平板显示最常用的接口。大多数较老式的计算机监视器都是用 3 根独立的插针，从 PC 视频卡和调制三相电子枪波束接收模拟 RGB 输入产生所需的图像。电子束激励屏幕上的磷点，则该点将发出红、绿、蓝或者三色的组合，具体的发光则取决于是哪路电子束击中了该点。这与模拟电视不同，后者使用的是一种复合信号，将所有色彩信息分层整合为一个单一的输入，并调制成一个单一的电子束。最近出现的计算机显示器使用 DVI 即数字可视接口，来接收数字和模拟的 RGB 信息。

CRT 的主要优势在于它们是廉价的，而且可以产生的颜色也要比同等尺寸的 LCD 平板要丰富。此外，与 LCD 不同的是，CRT 的可视角度非常宽广。CRT 的不足在于，体积较为庞大，电磁辐射较强，由于刷新时的闪烁效应，也会导致眼睛疲劳。

液晶显示（LCD）平板

LCD技术主要有两类：无源阵列和有源阵列。在前者中（最常用的一种类型是 STN，或超级螺旋向列及其衍生种类），往往采用由一块印刷出行电极引线结构的玻璃衬底与另一块印刷出列电极引线结构的玻璃衬底共同组成的“液晶三明治”结构。这些行列交叉点构成了像素点。于是，为了激活特定的像素，时序电路将像素点处的列上电，而将行接地。所形成的电压差则使得该像素点上的液晶产生翻转，于是该点变得不透明，阻止光线穿过。

无源阵列技术虽然简单，但也存在一些缺点。例如，屏幕的刷新时间较慢（这会造成快速物体显示时的重影现象）。此外，行-列交点处的电压也容易泄露到相邻的像素点上，这会在一定程度上造成周围区域像素的液晶变得不透明，阻碍光线的通过。对于观察者而言，图像会变得模糊不清，且对比度变低，另外，可视角度也相对较小。

在这些方面，有源阵列LCD技术可以很大程度上改进无源技术的缺点。从基本结构来看，每个像素点都由一个电容和晶体管开关构成，这种结构使之获得了一个更常用的名称“薄膜晶体管（TFT）显示”。为了对特定的像素进行定位，需要使能其所在行，然后向其所在列施加一个电压，这样可以带来隔离感兴趣的像素点的效果，而周围的其它像素都不会被影响。由于控制特定像素的电流被降低，该像素点的开关速度也更高，这就使得TFT技术具备了高于无源显示的刷新速率。此外，对施加到像素点上电压高低的调制也可以实现对多种亮度级的显示。如今，对于8bit的亮度信息，其显示的亮度级通常有256个。

由于涉及多种不同的组件，连接TFT-LCD的任务繁杂，令人感到混乱不堪。首先，考虑平板本身，需要根据像素时钟频率，对像素阵列的行和列加载脉冲。

TFT-LCD的背光常常是CCFL（冷阴极荧光灯），其内部的气体分子被激发并发光，而相应产生的热量很少。它们适合于LCD平板显示的其他原因还包括：耐用性、长寿命和简单易行的驱动要求。LED也是一种流行的背光方法，主要用于小到中等尺寸的平板，它们的优点在于成本低、工作电压低、寿命长、亮度控制性好等。不过，当平板尺寸更大时，与CCFL相比，LED背光的功耗就显得过高。

一个LCD控制器包含了将一路输入视频信号转换为LCD平板所需格式的大部分电路，它往往包括一个时序发生器，用于控制平板上各像素的同步和时钟信号的时序。不过，为了满足LCD平板在尺寸和成本方面的要求，有时需要由“时钟发生器”或者“时序发生ASIC”芯片从外部提供时序生成电路。除了实现标准的同步以及数据线外，驱动LCD平板上各行和列也需要定时信号。有时，媒体处理器上富余的通用性PWM（脉宽调制），可以取代这一分立芯片，降低系统的成本。

LCD控制芯片还提供屏幕显示支持、图形层叠混合、颜色速查表、抖动与图像旋转等其他一些特色功能。结构复杂的芯片，其价格也极为昂贵，常常超出了与之相连的处理器的成本。

为了给LCD平板提供恰当的电平，需要采用合适的LCD驱动器芯片。它起到LCD控制输出和LCD平板之间的“转换器”的功能。行与列往往是分开驱动的，其时序由时序信号发生器来控制。液晶必须用周期性的极性翻转信号来驱动，因为直流电流会给液晶结构带来应力，并最终使之损坏。于是，施加到每个像素的电压的极性必须满足基于每帧、每行或者每个像素变化一次的要求，具体采取何种方式则取决于实现方案。

随着多媒体设备向着更小型、更廉价的方向发展，促使人们将上述这些LCD系统部件集成起来。如今，集成的TFT-LCD模块包含时序信号产生与驱动电路，只需要一路数据总线连接、时钟/同步化线和电源。一个集成化的TFT-LCD显示模块的电气接口简单易懂，它一般包含数据线、同步线、电源线和一路时钟线。有些平板除了支持并行的视频数字输入信号外，还支持复合模拟视频输入。

OLED（有机发光二极管）显示

OLED 中的“有机”是对夹在两个电极之间的材料而言。当电荷穿过这一有机材料时，这种有机物就会发光。该显示技术仍属一种新生技术，但它有望大大改善 LCD 显示的若干不足。例如，它是一种自发光的技术，无需背光，这就意味着将大大降低显示器的功耗、成本及其重量——一个 OLED 平板可以非常轻薄。此外，它可以比 LCD 平板提供更丰富的色彩，对运动图像的显示效果也优于 LCD。此外，它支持更宽的视角和更高的对比度。OLED 的电子时序发生器和数据接口与 TFT LCD 平板类似。

尽管具有上述种种优点，目前限制 OLED 显示应用的最主要因素是其有限的寿命。在使用几千小时后，有机材料就会击穿，尽管现在某些显示器的这一量值已经超过了 10 000 小时——非常适合于许多便携式多媒体应用。因此，OLED 在手机、数码相机等产品中，具有广泛的应用前景。然而，我们相信在不久的将来，会看到基于 OLED 技术的电视或者计算机显视器。目前，随着 LCD 技术的不断发展，OLED 在大规模生产方面的安排与规划也日趋清晰。

既然我们已经阐述了系统内部的视频流连接方面的基础知识，下面就该对处理器的内部进行一番考察，了解其是如何能有效处理视频信号的。这将是本系列文章下一部分的主题。

Part-4

本文是5部分系列文章的第4部分。文中我们将从处理器的角度来考察视频流，把注意力放在那些能实现高效数据处理与转移的结构和功能特色上。

视频端口的功能特色

要处理视频流，处理器就必须具有适当的接口，以保证数据能高速输入和输出该模块。有些处理器是通过一片连接到处理器的外部存储接口上的 FPGA 和/或 FIFO 来实现上述接口的。这种器件一般用于协调出入处理器的、速度恒定且较慢的视频流（NTSC 视频的速度约～27 MB/s）与零星而高速的、猝发性的外部存储控制器数据流（其容量达到约 133 MWords/s 或者 266 MB/s 的水平）之间的关系。

不过，这样的安排仍然存在问题。例如，FPGA 和 FIFO 价格昂贵，往往与视频处理器本身相当。此外，使用外部存储器的接口来进行视频信号的传递，也会夺走其在这些系统中的另一项主要用途——在处理器内核与外部存储之间来回传送视频缓冲数据——所需的带宽。

因此，媒体处理系统最好采用一个专有的视频接口。例如在 Blackfin 处理器上就采用了“并行外设接口（PPI）”。PPI 是一种多功能的并行接口，它可以配置为 8 和 16bit 两种带宽。它支持双向的数据流，并包含了 3 条同步线以及一个与外部时钟相连的时钟引脚。PPI 可以对 ITU-R BT.656 数据进行无缝解码，而且还可以与 ITU-R BT.601 视频源与显示设备，例如 TFT LCD 平板显示器接口。它可以用作高速模数转换器（ADC）和数模转换器（DAC）的连接管道。它还可以为外部处理器仿真出主控接口。

PPI 的一些内置的功能可以降低系统的成本，并改善数据的流动。例如，在 BT.656 模式中，PPI 可以对输入视频流进行解码并自动忽略有效视频之外的任何信号，从而可以有效地将一路 NTSC 输入视频流的码率从 27MB/s 降低到 20MB/s，并显著减少处理视频所需的片外存储器的大小。或者，它也可以忽略有效的视频区，而只读入那些嵌入到垂直消隐间隔中的辅助数据。图 1 对这些模式进行了图示。

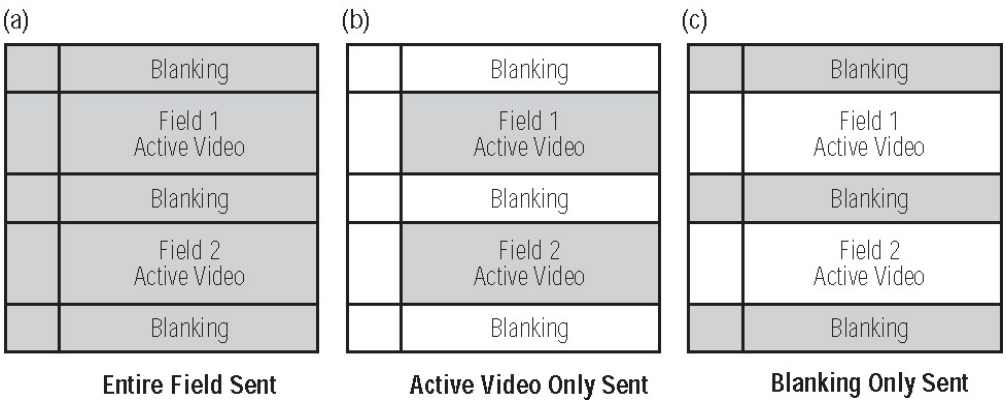


图 1：PPI 接口对 BT.656 各区域的有选择性的掩蔽

图中：Entire Field Sent 发送整个场，Active Video Only Sent——仅发送有效的视频，Blanking Only Sent——仅发送消隐信号。Blanking——消隐，Field 1 Active Video——场 1 的有效视频，Field 2 Active Video——场 2 的有效视频

类似的，在隔行扫描中 PPI 可以每隔一个场“忽略”一个场，换句话说，它不会把这些数据继续提交给 DMA 控制器。虽然这种方法可以立即让输入带宽的要求降低 50%，但它也删除了 50%的视频源内容，因此有时这种折衷是无法令人接受的。不过，当输入视频的分辨率远高于所需要的输出分辨率时，这是一项有用的功能。

类似的，PPI 允许“跳过”编号为奇数或者偶数的分量，再次将所跳过的像素单元对应的 DMA 带宽节约下来。例如，在一个 4:2:2 的 YCbCr 流中，该功能将只容许亮度或者色度信号被读入，提供了在各种不同处理器之间进行算法划分的便捷方法；一个处理器可以读入亮度信号，而另一个则可以读入色度信号。它还提供了将一幅图像或者一路视频流转换为灰度级（仅有亮度信号）的简单方法。最后，在带有交织的 I/Q 数据的高速转换器应用上，这一特色使得在这些同相和正交分量之间进行划分成为可能。

重要的是，PPI并不知晓数据格式，因为它并未针对特定的视频标准来进行硬件连线。它容许用户对行长度和帧长度进行编程设定。这就给那些需要诸如CIF或者QCIF视频，而非标准的NTSC/PAL格式的应用帮了大忙。一般而言，只要输入的视频具有恰当的EAV./SAV代码（BT. 656视频）或者硬件同步信号（BT.601视频），PPI就可以对其进行处理。

封包

虽然BT.656和BT.601推荐方案可以提供10bit的像素单元，但这对于处理来说不是一个友好的字长度。这里的问题是，大多数处理器在处理按8bit、16bit或者32bit分块的数据方面效率很高，而对于处于这些值之间的那些量值都会造成数据搬移方面的效率低下问题。例如，即使一个10bit像素值只比8bit像素值宽2bit，但大多数处理器都会将其作为一个6个高位（MSB）为0的16bit数据来处理。这不仅会浪费内部数据传输总线（DMA）的带宽，而且还会浪费很

多内存——这对于视频应用来说是不利的，此时，若干个整帧的缓冲数据往往要存储在外部存储器中。

数据尺寸大于8bit所带来的效率低下，还表现在数据打包无法达到最佳性能。往往一个高性能的媒体处理器将在其周边形成数据包封机制，该包封居于外部世界和处理器内部的数据传送总线之间，其目标是尽可能减少数据出入外设时总体上给这些总线的带宽造成的负担。因此，一路8bit的视频流以27MB/s的时钟频率输入外设时，可以被打包到32bit 的内部数据搬移总线上，相应的，只需该总线提供27/4或者说6.75MHz 的服务即可。请注意，总的数据传输速率仍然相同（6.75MHz * 32bit＝27MB/s）。与之相反，一个以27MB/s传输的10bit的视频流将只能以2个16bit数据片的形式封装到32bit内部总线上，这就将总的传输速率降低至27/2或者说13.5MHz。在这种情况下，在每16位数据中只有10bit数据是相关的，故内部总线带宽的37.5%被浪费了。

可能的数据流

对多媒体系统中的视频端口的一些连接方式进行考察，展示系统作为一个整体与构成系统的每种数据流动之间的相互依存关系，将是很有益的。在图 2a 中，图像源将数据发送到 PPI 中，在该处 DMA 引擎将其配属 L1 内存，在 L1 内存中，数据经过处理，转换为其最终形式，然后通过一个高速串口发送出去。该模型对于低分辨率的视频处理和 JPEG 等图像压缩算法来说是十分有效的，此类算法可以对小规模的视频数据块（对应若干行）进行处理，此后，这些数据块在随后的过程中将不再为系统所需要。上述的流程对于有些数据转换器应用来说也是十分有效的。

在图 2b 中，视频数据并没有被路由到 L1 内存，而是被引向 L3 存储器。这种配置可以支持 MPEG-2 和 MPEG-4 等算法，这些算法需要将中间视频帧存储在存储器中，以执行时间域的压缩。在这种情形中，L1 和 L3 存储器之间的双向 DMA 流就可以实现像素宏块以及其他中间性数据的传送。

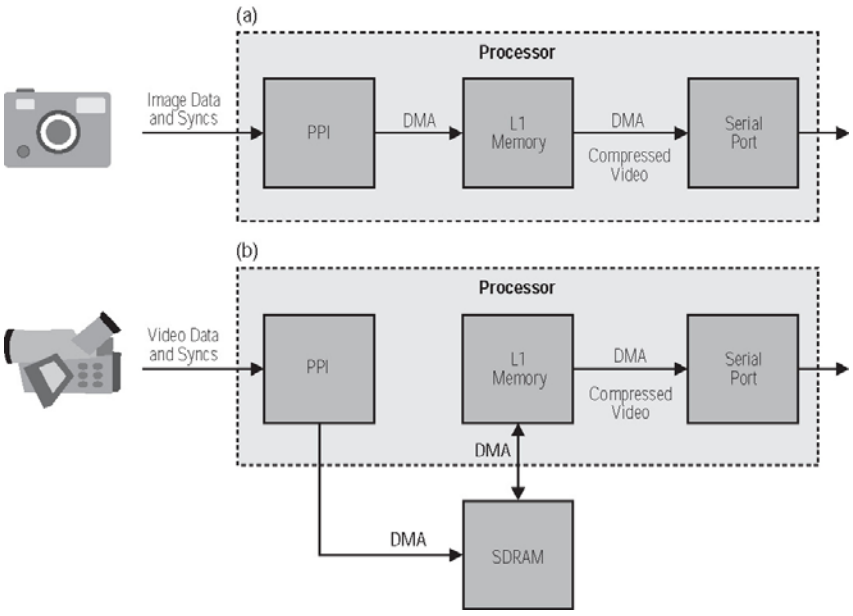


图 2:　可能的视频端口数据传输情形

图中：Image Data and Syncs——图像数据和同步信号；Processor——处理器，Memory——内存，Serial Port——串行端口，Compressed Video——经过压缩的视频；Video Data and Syncs——视频数据和同步信号。

视频 ALU

大多数视频应用都需要对 8bit 数据进行处理，因为各像素分量（无论是 RGB 或是 YCbCr）往往都是按字节计量的。因此，8bit 视频 ALU 和基于字节的地址生成可以带来像素处理方式上的巨大差异。这一点并非无足轻重，因为嵌入式处理器一般工作在 16bit 或者 32bit 的范围内。

嵌入式媒体处理器有时支持那些能高效地处理 8bit 视频和数据的语句。以表 1 为例，其中总结出专门的 Blackfin 指令，综合运用这些指令，就可以实现各种不同类型的视频操作。

Table 1: Native Blackfin video instructions

指令	描述	算法的应用
Byte 对准	将连续的 4 字节未对准的字从两个数据寄存器的组合中拷贝出来。	用于对数据字节进行对准，以满足随后执行的 SIMD 指令的需要。
双 16bit 加/减	向两个 16bit、带符号的量值添加两个 8bit 无符号的量值，然后限制在 8bit 的无符号范围中。	主要用于视频运动压缩算法。
双 16 bit 累加器提取外加加法	将每个累加器的字的上半部分和下半部分相加到一起，然后载入目标寄存器中。	与 Quad SAA 指令一起使用，实现运动的估计
4 重 8bit 加法器	将两个无符号的 4 重字节数组相加	对于提供视频处理应用中典型的包封数据运算来说非常有用。
4 重 8bit 平均值－字节	逐个字节计算出两个 4 重字节数组的算术平均值	支持分形运动搜索和运动估计中所使用的二进制内插法。
4 重 8bit 平均值—半个字	逐个字节计算出两个 4 重字节数组的算术平均值，而且把结果放置在半个字的边界上。	支持分形运动搜索和运动估计中所使用的二进制内插法。
4 重 8bit Pack	将 4 个 8bit 值包封为 32bit 的寄存器	为 ALU 操作准备数据
4 重 8bit 减法	两个 4 字节数组的逐字节相减	在视频应用中提供包封的数据运算
4 重 8bit 减法－绝对值－累加	实现 4 对值的相减，取出其绝对值，然后可以进行累加。	对于基于块的视频运动估计来说非常有效。
4 重 8Bit 解包封	从一对源寄存器处拷贝 4 个连续的字节	

让我们考察几个关于这些指令的运用的示例。

4 重的 8bit 减法－绝对值－累加（SAA）指令非常适合于基于块的视频运动估算。该指令先对 4 对字节进行减法操作，并取每个差值的绝对值，然后将结果累加起来。这都是在一个周期内发生的。下面是其确切的表达式。

$$SAD = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|a(i,j) - b(i,j)\right|$$

试考虑如图 3a 所示的宏块。16 像素×16 像素的基准帧可以进一步分为 4 组。一个非常合理的假设是相邻的视频帧是彼此相关的。也就是说，如果存在运动，则每个帧的各片区域将相对于前面诸帧的宏块发生移动。对宏块的运动进行编码所需要的信息，要少于将每个视频帧视为单独对象进行编码时所需的信息 – MPEG 压缩使用了此技术。

这种对宏块进行的运动检测可分解为两个基本步骤。设定一个帧中的基准宏块之后，我们就能搜寻下一帧中所有在其周围的宏块（目标宏块），以决定最接近的匹配。基准宏块（在帧 n 中）的位置与可与之实现最佳匹配的目标宏块（在帧 n＋1 中）之间存在的偏移，就是运动矢量。

图 3b 图示出了系统中该方法的具体体现方式。

- 圆 = 视频帧中的某些对象
- 实线方框 = 基准宏块
- 虚线方框 = 对区域进行搜索，以发现可能的宏块。
- 点方框 = 最佳匹配的目标宏块（例如代表圆物体的运动矢量宏块）

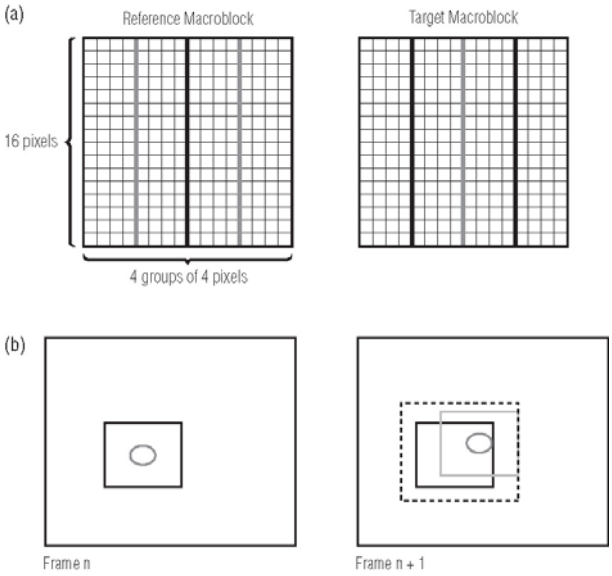


图 3: 减法－绝对值－累加（ SAA）指令的图示
图中：Reference Macroblock——基准宏块，Target Macroblock——目标宏块。
16 pixel——16 像素，4Groups of 4 pixel——4 组，每组 4 个像素，Frame——帧

一个 Blackfin 处理器的 SAA 指令执行速度很快，因为它在每个时钟周期中可利用 4 个 8bit ALU。我们可以实现如下的循环，以对图 3b 所示的 4 个对象中的每一个进行迭代处理。

```
/* 用于对图像组块进行迭代处理的循环中 */
SAA (R1:0,R3:2) || R1 = [I0++]  || R2 = [I1++]; /*计算出绝对值并累加*/
SAA (R1:0,R3:2) (R) || R0 = [I0++] || R3 = [I1++];
SAA (R1:0,R3:2) || R1 = [I0 ++ M3] || R2 = [I1++M1];  /*取出目标组块的第 4 个字后，指针指向下一行*/
SAA (R1:0,R3:2) (R) || R0 = [I0++] || R2 = [I1++];
```

现在让我们考虑另一个实例，即求 4 个相邻值平均值的运算，图 4a 示出其基本内核。通常情况下，必须进行 4 次加法和一次除法（或者乘法或者移位），才能计算出平均值。BYTEOP2P 指令可以加快该滤波器的实现。

图 4b 中心像素的量值被定义为：

x = Average(xN, xS, xE, xW)

BYTEOP2P 可以在一个周期内对两个像素执行这种求平均值处理（图 6.21 c,d）。因此，如果 x1 = Average(x1N, x1S, x1E, x1W)，x2 = Average(x2N, x2S, x2E, x2W)，则

R3 = BYTEOP2P(R1:0, R3:2);

将在单个周期内同时计算出两个像素的平均值，前提是 x1 (N, S, E, W)信息存储在 R1 和 R0，x2 (N, S, E, W)则从 R3 和 R2 取出。

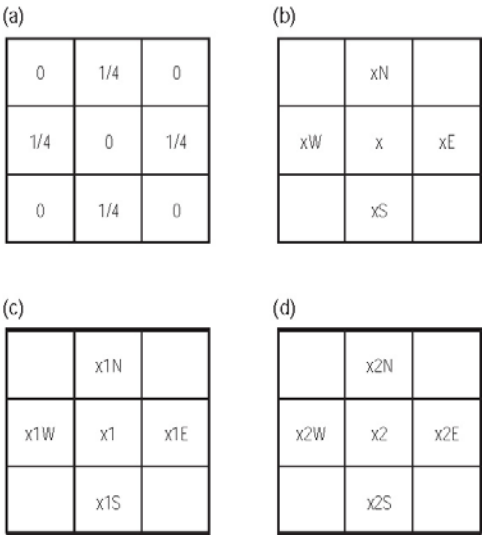


图 4　相邻像素的平均值计算

关于 DMA 的考虑

具备二维 DMA（2D DMA）能力的嵌入式媒体处理器可以在系统层次上提供若干优点。对于新手而言，2D DMA 可以实现宏块对应于外部存储器的来回传递，从而使得对数据的处理成为实际传输的一部分。这消除了传输非连续数据时相应出现的开销。它还可以让系统进行有选择性的传输，即仅传输输入图像中为人们所需要的区域，而不是传输整幅图像，从而尽可能减小数据的带宽。

另一个实例是，2D DMA 可以让数据置入存储器的顺序更顺应处理的需求。例如，如图 5 所示，RGB 数据可以从一个 CCD 传感器以交织的 RGB444 格式进入处理器的 L2 存储器，但是在使用 2D DMA 的情况下，它可以以分离的 R、G 和 B 平面形式传送到 L3 存储器中。对视频和图象数据颜色空间的分量进行交织/解交织化在处理前节省了额外的数据搬移。

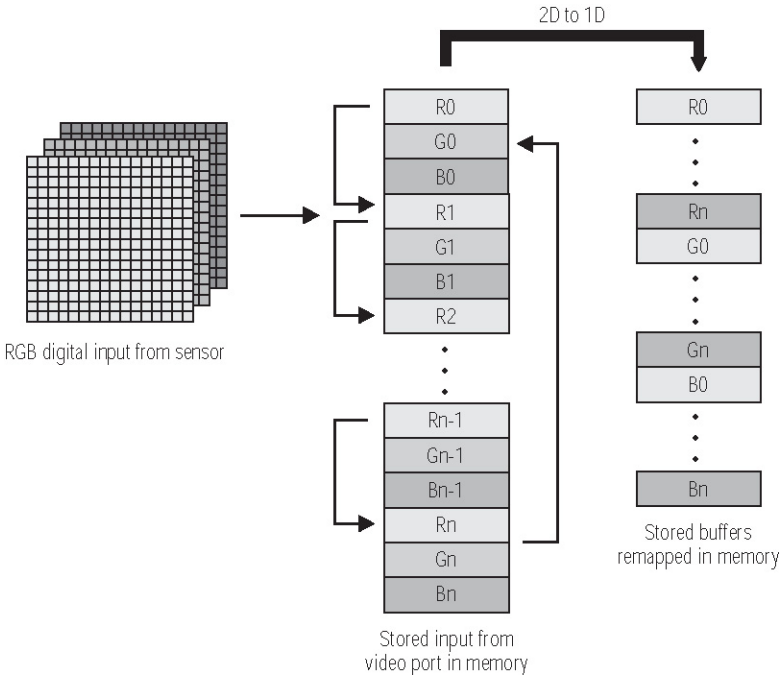


图 5： 利用 2D DMA 实现数据的解交织
图中：说明文字，从左到右，来自于传感器的 RGB 数字输入，存储器中来自视频端口的输入，在存储器中进行重新映射的、存储的缓冲信号。

平面化和间插式缓冲格式的对比

在“存储缓冲采用交织的结构还是平面化的结构？”这一问题上，你应该如何做出决策？交织的数据的优点是，它是图像传感器的自然输出格式，而且也是视频编码器自然而然的输入格式。不过，平面化的缓冲器（即每个像素分量存储在存储器中相互分离的区域中）对许多视频算法而言是更为有效的结构，因为它们中的许多（包括 JPEG 和 MPEG）都是依靠单独的亮度和色度信号来工作的。此

外，对 L3 中平面化缓冲器的访问，其效率要比跨越交织数据进行读取的操作更高，这是因为，当缓冲器采用平面化结构时， SDRAM 页面缺失造成的延迟会随着图样尺寸的大幅增加而发散。

双重缓冲

我们在前面已经讨论了对双重缓冲的需求，这是因为它可以确保当前数据不被新的数据所覆盖，直到你已经为这种覆盖做好准备为止。对视频显示缓冲区的管理就是这一方法的绝好实例。通常，在系统中，如果各种视频源与最终显示的内容之间存在传输速率差异的话，就应该保证在老的内容和新的视频帧之间实现平滑的切换。这是利用双缓冲管理方法来实现的。一个缓冲区指向目前的视频帧，该帧被以一定的刷新速率送到显示器上。第二个缓冲区则用最新输出的帧来填充。当后一个缓冲器被填满时，DMA 发出中断信号，指示现在应该将新的帧发送到显示器上。此时，第一个缓冲区开始填充经过处理的、用于显示的视频信号，而第二个缓冲区则输出当前的显示帧。这两个缓冲区以“乒乓”方式来回切换。

我们应该注意的是，系统可以使用多个缓冲区而不仅仅是两个，以便为同步化提供更大的余地，并降低中断发生的频率及其相应造成的延迟。

现在，我们已经讨论了一些与嵌入式应用中有效的视频数据的搬移有关的基本问题和功能特色。在本系列文章中的第 4 部分，我们将把这些思想扩展到一个示例性的嵌入式视频应用的“排演”中。

Part-5

本文是 5 部分系列文章的最后一部分。在本文中，我们将考察一个视频应用示例，就此探讨一些所涉及的主要的处理模块。

让我们对图 1 所示的系统进行一番游历，以之向读者展示，在嵌入式视频应用中，某些基本的视频处理步骤是如何形成不同的组合的。在该图中，一个隔行扫描的 CMOS 传感器通过处理器的视频端口发送一个 4:2:2 YCbCr 视频流，此时它完成了去隔行操作及扫描速率转换。它随后接受某些计算算法的处理，为输出到 LCD 平板显示做好准备。这种准备过程涉及色度的再采样，Gamma 校正、颜色转换、缩放、与图形的混合，然后封装为恰当的输出格式，以方便在 LCD 平板上的显示。请注意，该系统仅是一个示例性的，一个特定系统并不需要配备上述所有这些元部件。此外，这些步骤也可以以不同于此处的顺序来执行。

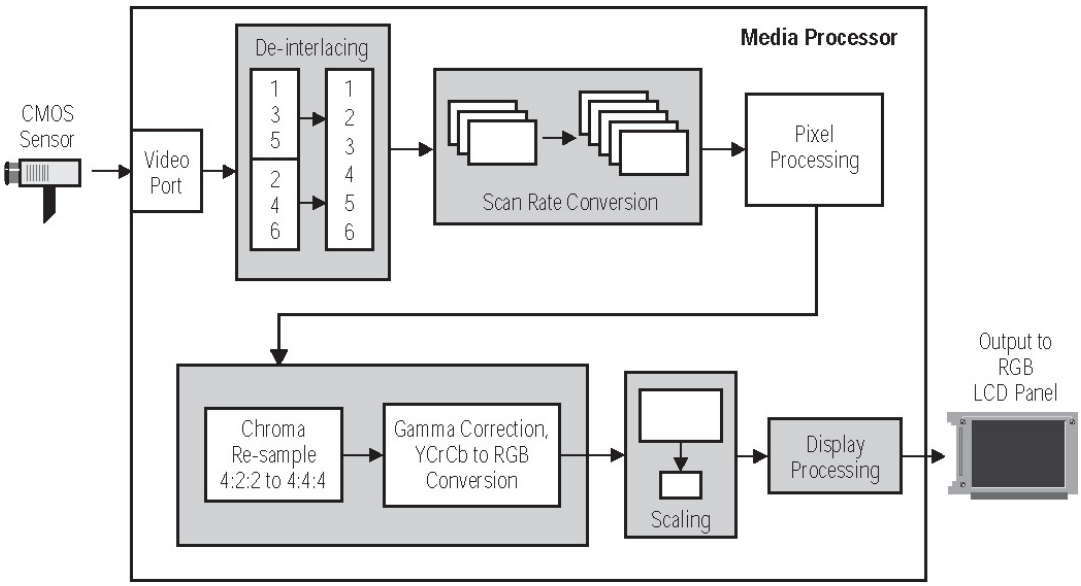


图 1　摄像机输入到 LCD 输出的数据流示例，包括了位于其间的各处理阶段。Sensor——传感器，Video Port——视频端口，De－interlacing——去隔行，Scan Rate Conversion——扫描速率转换，Media Processor——媒体处理器，Pixel Processing——像素处理，Chroma Re－sample——色度信号的再采样。Gamma Correction，YCrCb to RGB Conversion——Gamma 校正，YCrCb 到 RGB 的转换，Scaling——缩放，Display Processing——显示处理，Output to RGB LCD Panel——输出到 RGB LCD 平板。

去隔行

将视频源数据从一个输出隔行 NTSC 数据的摄像机处取出时，往往需要对其进行去隔行处理，这样奇数行和偶数行将交织排列在存储中，而不是分别位于两个分离的视场缓冲区中。去隔行不仅仅是高效率的、基于块的视频处理所需要的，而且也是在逐行扫描格式中显示隔行视频所必需的（例如在一个 LCD 平板上）。去隔行处理有多种方法，包括行倍增、行平均、中值滤波和运动补偿。

扫描速率的转换

一旦视频完成了去隔行化，就有必要进行扫描速率的转换，以确保输入的帧速率与输出显示的刷新速率相匹配。为了实现两者的均衡化，可能需要丢弃场或者复制场。当然，与去隔行化类似的是，最好采用某种形式的滤波，以便消除由于造成突然的帧切换而出现的、高频的人为干扰。

帧速率转换的一个具体情况，即将一个 24 帧/s 的视频流（通常是对应着 35mm 和 70mm 的电影录制），转换为 30 帧/s 的视频流，满足 NTSC 视频要求，这属于 *3:2 下拉（pulldown）。例如，如果每个胶片（帧）在 NTSC 视频系统中只被用到一次，则以 24fps 记录的电影的运动速度将提高 25％（＝30/24）。于是，*3:2 下拉被认为是一种让 24fps 视频流转换为 30fps 的视频序列的转换过程。它是通过以一定的周期化的样式重复各帧来实现的，如图 2 所示。

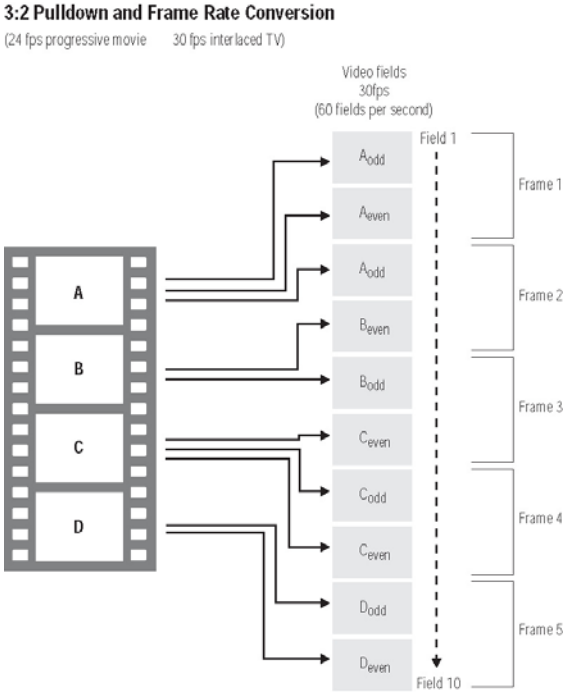


图 2: 3:2 下拉帧重复模式

3:2 下拉和帧速率转换

（24fps 逐行电影），视频场，30fps（每秒 60 场），Field——场，Frame——帧

像素处理

正如我们上面曾讨论过的那样，许多视频算法得到了广泛应用，它们可以被划分为空间和时间方面的诸多类别。一个极为常见的视频算子是二维（2D）卷积内核，它已经用于多种不同形式的图像滤波。

2D 卷积

既然一个视频流事实上是一路以特定速率移动的图像序列，图像滤波器需要以足够快的速度运行，以跟上图像连续不断的输入。于是，重要的是，图像滤波器内核必须被优化，从而在尽可能少的处理器周期内完成。这可以通过考察基于 2 维卷积的简单图像滤波器组来予以演示说明。

卷积是图像处理中的基本操作。在二维卷积中，针对特定像素执行的计算是在其附近的像素点的亮度值的加权和。因为掩膜的周围都是以特定像素为中心的，故掩膜往往具有奇数次阶数。相对于图像而言，掩膜尺寸一般很小，人们通常选择 3×3 掩膜，因为从逐个像素的角度出发，它在计算方面是合理的，同时足够大以能检测出图像的边缘。不过，应该指出的是 5×5、7×7，甚至更多的像素组合，也得到了广泛应用。例如，摄像机的图像流水可以采用 11×11（甚至更大的内核），以便执行极为复杂的滤波操作的。）

图 3a 示出了 3×3 内核的基本结构。作为例子，对一幅图像上第 20 行、第 10 列上的一个像素点的卷积运算过程的输出应该是：

Out(20,10)=A(19,9)+B*(19,10)+C*(19,11)+D*(20,9)+E*(20,10)+F*(20,11)+G*(21,9)+H*(21,10)+I*(21,11)*

The 3x3 convolution mask and how it can be used

A	B	C
D	E	F
G	H	I

(a) 3x3 Convolution Kernel

0	0	0
0	1	0
0	0	0

(b) Delta

-1	0	1
-1	0	1
-1	0	1

1	1	1
0	0	0
-1	-1	-1

(c) Edge Detection

1/8	1/8	1/8
1/8	0	1/8
1/8	1/8	1/8

(d) Smoothing

-1/8	-1/8	-1/8
-1/8	1	-1/8
-1/8	-1/8	-1/8

(e) Edge Enhancement

图 3： 3×3 卷积掩膜及其使用方法
图中（a）3×3 卷积内核，（b）Delta；（c）边缘检测；（d）平滑化；（e）边缘增强。

应当以一种对计算有帮助作用的方式来选择系数。例如，是 2 的幂次（包括分数）的缩放因子是首选方法，这样乘法就可以被替换为简单的移位操作。

图 3b-e 示出了几种有用的 3×3 内核，其中每一种在下面都将予以简要的讲解。

图 3b 中的 Delta 函数是最简单的图像处理方式中的一种，可以让当前的像素通过而无需修改。

图 3c 示出了 2 种最常用的边缘检测掩膜形式。其中第一种用于检测垂直边缘，

而第二种则检测水平边缘。高输出值对应着边缘存在的程度更高。

图 3d 的内核是一种平滑滤波器。它可以执行对周围的 8 个像素进行平均化处理，并将结果放置到当前的像素位置上。这一操作的结果是图像的“平滑化”或者低通滤波。

图 3e 的滤波器被称为一个“非尖锐的掩膜算子。”它被视为一种能产生边缘增强图像的手段，方法是从当前的像素中减去其自身经过平滑化的版本（通过对周围的 8 个像素的平均来获得）。

图像边界的处理

当一个诸如 2D 卷积的函数对图像边界附近的像素点进行操作时会发生什么情况？为了恰当地执行像素滤波，就需要这些边界“之外”的像素信息。出现这一情况，可以有两种补救方式。最简单的一种是，干脆忽略这些边缘区。例如，假定一个 5×5 的卷积内核需要分别选取当前像素的左、右、上、下的各 2 个像素，以保证运算的正常进行。于是，为何不在每个方向上删去 2 行，这样就可以保证该内核能始终对真实数据进行处理。当然，这并不始终是一种理想的方法，因为它丢弃了实际的图像数据。此外，当将滤波器串联起来，以形成更为复杂的像素处理能力时，随着每个新增加的滤波器级被执行，该方法将不断让输入图像幅面变窄。

处理图像在边界上出现的难题时，常用的其它方法是：复制多行和/或多列像素，或者从左边缘（顶）绕回前一图像的右（底）边缘上。虽然这在实践中可能易于实施，但它们也产生了此前并不存在的数据，因此在某种程度上会破坏滤波效果。

也许，最直接的、也是破坏最小的图像边界处理方法，是将处于实际图像之外的一切像素视为 0 值，或者说黑点。虽然这一方案也会造成滤波结果的失真，但它的侵入性并不像产生潜在随机的非 0 值像素的像素行时那么大。

色度再采样和颜色的转换

我们的示例中的数据流最终需要转换到RGB空间中。我们在前面已经讨论了如何借助一个3×3的矩阵乘法在4:4:4 YCbCr 和 RGB空间之间进行相互转换。不过，到目前为止，我们的像素值依然处于4:2:2 YCbCr空间中。因此，我们需要对色度值进行再采样，以达到4:4:4的格式。接下来，到RGB的转换将是直接了当的，正如我们前面看到的那样。

从4:2:2 通过再采样转换为4:4:4格式，需要从那些缺乏Cb和Cr分量之一的Y样本中插值出Cb和Cr值。一个简洁的再采样方法是从最邻近的像素上借助简单的平均化方法插值出缺失的色度值。也就是说，在一个像素点上缺失的Cb值将为最接近

的2个Cb值的平均值所取代。某些应用还需要更高阶的滤波器，但这一简化的方法往往也已经够用了。另一种方法是对邻近的像素点的色度值进行复制，以得到这些在当前像素表示中缺失的量值。

一般来说，从 4:1:1 空间到 4:2:2 或者 4:4:4 格式的转换只需用到一个一维的滤波器（tap 值和数量应与所希望的滤波水平相一致）。不过，从 4:2:0 格式再采样为 4:2:2 或者 4:4:4 格式，则还需要用到垂直采样，于是就有必要采用二维卷积内核。

由于色度的再采样和 YCbCr→RGB 的转换都是线性的运算，有可能将这些步骤组合起来，形成单个数学运算，从而高效率的实现 4:2:2 YCbCr→RGB 的转换

缩放和裁切

视频缩放可以生成一路分辨率与输入格式的分辨率不同的输出流。理想情况下，固定缩放的要求（输入数据分辨率，输出平板显示的分辨率）都是事先已知的，以避免在输入和输出流之间进行任意的缩放所带来的计算上的负担。

缩放可以是缩减和放大，具体则取决于应用。清楚待缩放图像的内容（如文字和细线的存在与否）很重要。不恰当的缩放会导致文字不可阅读或者造成缩放后的图像中某些水平线的消失。

将输入帧尺寸的大小调整为幅面更小的输出帧时，最简单的方法是图像的裁切。例如，如果输入的帧尺寸是 720×480 像素，输出是 VGA 帧（640×480 像素），则你可以把每行的前 40 和后 40 个像素丢弃。这样做的一个优点是像素的丢失和复制不会造成人为的影响。当然，缺点就是，你将损失帧内 80 个像素（约 11%）的内容。有时，这并不是很大的问题，因为，由于显示器的机械外壳的存在，屏幕的最左侧和最右侧的部分（以及顶部和底部区域），往往成为人们无法看清的地方。

如果无法选用裁切的方法，还可以通过若干种方法来对图像进行下采样（减少像素和/或行的数量）或者上采样（增加像素和/或行的数量），这使得人们可以在处理的复杂程度和相应的图像质量方面做出折衷取舍

增加或减少每行的像素数量

缩放方面所采用的一种直接的方法是丢弃像素（下采样）或者复制现有的像素（上采样）。也就是说，当缩减为一个较低的分辨率时，每条线（和/或每帧中的一定数量的线）的一些像素将被丢弃。这当然会减少处理的负担，但结果是造成了混叠和可见的人为影响。

只令复杂程度有少量上升的方法，是利用线性内插法来改进图像质量。例如，在缩减一幅图像时，在水平和垂直方向上的滤波可以获得一个新的输出像素点，该像素随后取代了在插值过程中所用的像素点。与前述技术类似的是，信息仍然会被丢弃，依然会出现人为和混叠问题。

如果图像品质极为重要的话，还有其它方法可在不降低品质的前提下实现缩放。这些方法都是努力维持图像的高频分量，使之在水平和垂直缩放中保持一致，同时减少混叠的影响。例如，假定要求对一幅图像以 Y:X 为比例因子来缩放。为了达到这一目标，图像可以 Y 为因数进行上采样（“插值”），通过滤波以防止混叠，然后以比例因子 X 进行下采样（“抽样滤波”）。在实践中，这 2 个采样过程可以组合成为单个多比例滤波器。

增加和减少每帧的行数

增加或者减少每行的像素点的指导原则，一般可以用于一幅图像每帧的行数的修改。例如每隔一行丢弃一行数据（或者整个隔行场）提供了一种可减少垂直分辨率的快速方法。然而，正如我们以前提到过的那样，在丢弃或者复制行时，应当采用某种垂直滤波方案，因为这些处理会在图像中引入人为影响。这里可以采用的同一种滤波策略是：简单垂直平均、更高阶的 FIR 滤波器或者多比例滤波器，以便垂直方向上得到一个确切的缩放比率。

显示处理

Alpha 混合

在显示之前往往需要将两种图像和/或视频缓冲区组合起来。这样做的一个实用化的例子是蜂窝手机的图形显示上叠加的图标，如信号强度和电池电力指示等。一个涉及两路视频流的示例是：画中画功能。

将两路流组合起来后，你需要决定在这些内容重叠的地方，那路流将“胜出”。这就是 Alpha 混合发挥作用的地方。我们可以定义一个变量（α），它表示出如下在叠放流和背景流之间的“透明因子”，

输出值＝α（前景显示的像素值）＋(1-α)(背景像素值)。

该方程表明，α值为 0 时，可以实现完全透明的叠图，α值为 1 时，则叠图完全是不透明的，完全不显示相应区域的背景图像。

α有时通过单独的通道与各像素的亮度和色度一起发送。这就造成了“4:2:2:4”的情形，其中最后一位值是一个伴随着每个 4:2:2 像素对象的α键。α的编码形式与亮度分量的编码相同，但对于大多数应用来说，常常只需取少数几个透明度的离散级别（也许是 16 个）即可。有时一个视频叠图缓冲区要预先乘以α，或者预先借助速查表进行映射，在这种情况下，它被称为一个“经过整形的”视频缓冲区。

合成操作

在合成操作中，需要定位一个叠图缓冲区到一个对应更大图像的缓冲区中。常

见的一个实例是视频显示中的“画中画”模式，以及图形化图标（如电池和信号强度的指示标志）在背景图像或视频中的位置安排。一般来说，合成功能在输出图像完全完成之前尚需若干次反复循环。换句话说，产生一个复合的视频可能需要将“多层”图像和视频叠放起来。

二维的 DMA 功能对于合成功能来说是非常有用的，因为它容许在更大的缓冲区中定位出任意尺寸的矩形缓冲区。要记住的一点是，任何图像的裁切应该在合成过程之后进行，因为定位后的叠图可能会和此前裁切的边界发生冲突。当然，一个替代方法是确保叠图不至于和边界冲突，但有时这一要求实在可谓过分。

色度键控

“色度键控”一词是指两幅图像合成时，其中一幅图像中的特定颜色（往往是蓝色或者绿色）为另一幅图像中的内容所取代的现象。这提供了一种方便地将两幅视频图像合成起来的方法，其方法是有意识的对第一幅图像进行剪裁，使之为第二幅图像的恰当区域所取代。色度键控可以在媒体处理器上以软件或者硬件形式来实现。

输出格式化

大多数针对消费类应用的彩色LCD显示（TFT LCD）都带有数字RGB接口。显示中的每个像素实际上都有3个子像素——每个都包含红、绿和蓝色滤波器——即人眼可以将其分辨为单色的像素。例如，一个320×240像素的显示事实上具有960×240像素分量，这包含了R、G和B子像素。每个子像素有8bit的亮度信号，这构成了常见的24bit的彩色LCD显示基础。

这3种最常见的配置中，要么是每个通道使用8bit来表示RGB（RGB888格式），要么是每通道6bit表示（RGB666格式），或者R和B每通道用5bit表示，G通道用6bit数据表示（RGB565格式）。

在这3种配置中，RGB888提供了最好的色彩清晰度。这种格式总共有24bit的分辨率，可以提供超过1600万种色彩。它为LCD TV等高性能应用提供所需要的高分辨率和精度。

RGB666格式在便携式电子产品中非常流行。这种格式具有总共18bit的分辨率，可以提供262000种色彩。不过由于其采用的18引脚（6＋6＋6）数据总线并不能很好地与16bit处理器数据通道相兼容，在工业上一种常见的折衷方法是，R和B通道5bit，G通道6bit（5＋6＋5＝16bit总线），以此来实现与RGB666平板的连接。这种情形具有很好的性能，因为在视觉上绿色是3种颜色中最重要的。红色和蓝色通道的最低位连接到平板上对应的最高位上。这确保了每个颜色通道上的全动态范围（全亮一直低至全黑）。

我们希望本系列文章能让您对嵌入式视频处理的基本原理有一个充分的理解。对于媒体处理方面问题的深入讨论，包括数据流和媒体框架和分析，你可能希望参考这篇文章。

End