第35卷 第21期 计 算 机 工 程 2009年11月
Vol.35 No.21 Computer Engineering November 2009
·工程应用技术与实现· 文章编号：1000—3428(2009)21—0231—04 文献标识码：A 中图分类号：TP368

基于SOPC及图形加速引擎的座舱显示系统

胡小龙，李列文，周俊明
(中南大学信息科学与工程学院，长沙 410075)

摘 要：提出一种基于可编程片上系统和图形加速引擎的飞机座舱综合显示系统设计方案。为避免图形加速引擎直接对帧存储器进行零碎操作导致的存储器操作瓶颈，引入图形缓存机制。根据图形像素的存储特点提出“远区域优先”图形缓存页面淘汰算法。对汉字及自定义位图等操作采取软硬件结合的方式达到系统性能和资源利用的平衡，利用硬件锁保证帧存储器一致性。通过对模块进行波形仿真实现系统级仿真结果的可视化验证。

关键词：可编程片上系统；图形加速；现场可编程门阵列

Cockpit Display System Based on SOPC and Graphics Accelerating Engine

HU Xiao-long, LI Lie-wen, ZHOU Jun-ming
(School of Information Science and Engineering, Central South University, Changsha 410075)

【Abstract】A design scheme of aircraft cockpit display system based on System on Programmable Chip(SOPC) and graphics accelerating engine is put forward. In order to avoid the memory bottleneck caused by the fragmentary operations of graphics accelerating engine to frame buffer directly, a graphics cache mechanism is introduced in the graphics accelerating engine. According to the pixel graphics storage characteristics, it proposes a “Farthest Area First(FAF)” buffering replacement algorithms. A hardware/software co-design scheme is introduced for Chinese characters and bitmap display by hardware lock mechanism. A visual verification method for system-level simulation is proposed based on logic wave form simulation.

【Key words】System on Programmable Chip(SOPC); graphics accelerating; Field Programmable Gate Array(FPGA)

1 概述

低功耗、低成本的嵌入式图形硬件加速不仅在移动应用领域(GPS、手持游戏、汽车导航等)获得广泛的应用，也在军事应用(飞机、潜艇、航空航天等高可靠应用)领域得到普遍关注。尽管专门的图形加速ASIC早已用于3D游戏、CAD等高端图形应用，但中等规模图形及应用环境苛刻的一些图形处理目前大多采用CPU和简单的显示控制器来实现。由于现场可编程门阵列(Field Programmable Gate Array, FPGA)成本越来越低且具有可重配置、高可靠以及上市时间短等特征，因此它更能满足飞行座舱显示之类的中等规模图形图像硬件的加速处理要求。随着FPGA器件及设计技术的发展，以前需要采用CPU和专用ASIC图形处理器的应用现在可以在单个可编程芯片上实现[1]，即可编程片上系统(System on Programmable Chip, SOPC)。

本文提出一个可用于飞行座舱仪表液晶显示系统的SOPC方案。现代飞机座舱显示系统正在向大屏幕综合化方向发展，飞行员可以在有限的视域内依靠1台或几台显示器及时地获得来自雷达、光电系统、电子战系统、导航和识别等系统的各种信息[2]。传统的飞行座舱图形显示方法大多通过CPU和软件技术实现，而对于实时信息处理系统，CPU还需要进行繁重的数据处理和数据通信工作[3]，这会导致图形显示性能受到影响。本文用FPGA硬件实现图形显示处理，处理器可以像调用函数一样调用图形加速引擎在显示帧存储器中生成图形画面并在显示设备上显示，从而大大减轻了处理器的负担，提高了显示性能。

2 SOPC体系结构

基于FPGA的SOPC图形加速系统体系结构如图1所示。SOPC平台是Xilinx公司新一代Virtex4系列器件XC4VFX12，具有低功耗和高集成度特点。XC4VFX12具有一个PPC405 CPU硬核，主频达到450 MHz，可以满足飞控显示系统嵌入式计算能力要求，并支持VxWorks实时嵌入式操作系统和Linux操作系统。

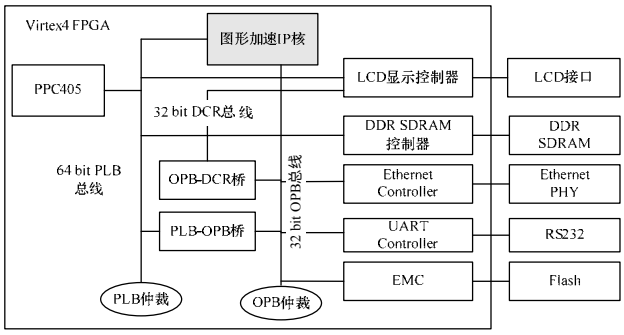

图1 SOPC体系结构

外部接口包括1个100 Mb/s以太网、1个RS422和1个

作者简介：胡小龙(1969－)，男，副教授、博士，主研方向：嵌入式系统，计算机体系结构；李列文，博士研究生；周俊明，硕士研究生

收稿日期：2009-04-14 **E-mail：**lilw168@126.com

LCD 液晶显示器接口。以太网和 RS422 接口用于与其他飞行控制单元的数据传输，LCD 液晶显示器接口与 1 个 1 024×768 的 LCD 显示屏连接进行显示输出。系统配置 64 MB 的 DDR SDRAM 作为系统存储器和显示帧存。片上系统总线为 IBM 开发的 CoreConnect 总线体系， CoreConnect 由高性能的 64 bit PLB(Processor Local Bus)总线、32 bit 的在片外围设备总线(On-chip Peripheral Bus, OPB)和用于设备寄存器操作的 DCR(Device Control Register)总线组成。PPC405 CPU 的指令 Cache 和数据 Cache 分别通过专用接口连接到 PLB 以支持其哈夫体系结构，PLB 总线挂接到 DDR SDRAM 控制器控制的高速存储子系统。以太网控制器、UART 控制器及 Flash 存储控制器等外部设备挂接在 OPB 总线，OPB 总线经由 PLB-OPB 桥连接到 PLB 总线。为简化系统设计，没有设置独立的显示帧存，而是将帧存设置在系统 DDR SDRAM 存储器中，LCD 显示控制器和图形加速引擎也直接挂接到 PLB 总线。由于 PLB 总线及 DDR SDRAM 存储器工作在 100 MHz 时峰值带宽高达 1.6 GB/s，因此可以满足 LCD 控制器和加速引擎对存储带宽的要求。LCD 显示控制器只是简单地按照屏幕刷新频率将帧存储器中的数据读出，并按照分辨率为 1 024×768 的 VGA 显示时序要求进行数据串行化，最后通过 LCD 接口输出。

SOPC 系统中 DDR SDRAM 控制器、UART 控制器、LCD 控制器和以太网控制器均采用 Xilinx XPS8.1 提供的 IP 核。本文主要完成图形加速引擎的设计，并将其集成在 SOPC 系统中。

3 图形加速 IP 核

3.1 图形加速引擎接口

图形加速引擎一方面需要以高带宽对帧存储器进行操作，另一方面还要从 CPU 接收图形加速命令流。为保证图形加速引擎不会因总线竞争而产生延迟，采取命令接口和数据接口分离的原则，即图形加速引擎的数据接口连接到 PLB 总线，通过 PLB 总线对帧存进行操作；而命令接口连接到 OPB 总线，CPU 通过 OPB 总线给出图形加速命令流。CPU 与图形加速处理器通过 FIFO 和控制状态寄存器交换数据，并采用状态机对 FIFO 中的图形命令进行容错处理，对命令参数序列进行格式化后送图形命令处理器。图形命令处理器根据图形命令和参数的内容，将图形命令分为 2 类：(1)画椭圆、椭圆弧等，需要调用相应的图形命令处理单元，将图形命令分解成基本图元，送到基本图元处理单元；(2)清屏等解释性图形命令，直接转换成 DDR SDRAM 操作命令，通过 Master PLB 的 DMA 操作刷新帧存储，见图 2。

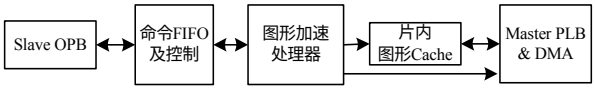

图 2 图形加速 IP 核的数据流

由于图形加速引擎、LCD 显示控制器以及 CPU 共享 DDR SDRAM 中的显示帧存储器，因此为了避免图形加速引擎对帧存储器的零碎存储导致的存储器操作瓶颈，系统中采用了 Cache 机制。

3.2 图形加速处理器

图形加速处理器完成对点、线段、圆、圆弧、椭圆、椭圆弧、梯形填充和清屏等硬件加速处理，内部包括水平线输出单元、打点控制单元、线段命令处理单元、椭圆弧命令处理单元、矩形和梯形填充命令处理单元等。

水平线输出单元进行最基本的图元处理，是其他图形加速操作(甚至画点)的基础，其他所有图形操作命令最终都通过水平线操作实现光栅化。水平线的宽度为 1 个像素，长度为 1 至屏幕宽度，其属性包括颜色和线型掩码。水平线输出单元的功能是根据水平线在屏幕的起点 X 坐标和 Y 坐标、终点 X 坐标计算起止像素点在显示帧存储器中的物理地址，然后根据水平线的颜色属性和线型掩码获得水平线各像素的像素值，并生成高速缓存指令将像素值写入高速缓存。

打点控制也是一个基本图元处理过程。本文定义的“点”是由多个像素组成的，具有大小和颜色属性。打点控制将来自其他图形加速指令处理单元的打点申请转换成一系列水平线，送到水平线输出单元。

线段、圆、椭圆和弧的处理过程是首先将它们分解为“点”，分解的同时“点”也继承线段、椭圆等图形的画笔颜色、大小属性，然后水平线调用打点控制单元，最终通过水平线输出单元光栅化。而矩形和梯形的填充处理是直接将被填充区分解为水平线，然后调用输出单元光栅化。因此，这些命令的处理过程实际上是一个将图形分解为点或水平线的过程。

3.2.1 线段命令处理

线段命令的最终结果是在给定的起止点以当前画笔属性在屏幕上画线，因此，线段命令参数包括屏幕坐标系起点坐标、终点坐标和画笔属性(颜色、大小、线型)。有多种绘制线段的算法，如数字微分法、中点画线法和 Bresenham 算法，考虑到 FPGA 资源受限和方便硬件实现的特点，本文采用 Bresenham 算法绘制线段。Bresenham 算法的基本思想是通过直线上的当前点以及直线的方位(而不是精确的斜率)以步进的方式确定直线上的下一点。具体过程如下：

(1)通过起点坐标($x0$, $y0$)和终点坐标(xn, yn)的线段可以表达为直线方程：

$$f(x, y)=ax+by+c=0$$

其中，$a=(y0-yn)$; $b=(xn, x0)$; $c=x0\times yn-xn\times y0$。

(2)通过点($x0$, $y0$)确定线段上的下一临近点坐标，该点在线段所在方位最临近的 3 个点中选取。例如线段方位角在 0～$\pi/2$ 之间，则这 3 个点为($x0+1$, $y0$), ($x0$, $y0+1$)和($x0+1$, $y0+1$)。可以得到 $f(x0+1, y0)=a$, $f(x0, y0+1)=b$ 和 $f(x0+1, y0+1)=a+b$。显然这 3 点不一定恰好在该线段上，但是选取这 3 点中离线段最近的一个点，即 $f(x, y)$绝对值最小的点作为线段的下一点，记该点坐标为($x1$, $y1$)，函数值 $f(x1, y1)$为 $\delta 0$。

(3)同样，根据点($x1$, $y1$)再确定下一点坐标，下一点在($x1+1$, $y1$), ($x1$, $y1+1$)和($x1+1$, $y1+1$)中选取，可以得到以下 3 个表达式：

$$f(x1+1,y1)=a+\delta 0$$
$$f(x1,y1+1)=b+\delta 0$$
$$f(x1+1,y1+1)=a+b+\delta 0$$

选取 3 个点中函数值绝对值最小的点作为线段的下一点，记该点坐标为($x2$, $y2$)，函数值 $f(x2, y2)$为 $\delta 1$。依次步进，直到线段终点(xn, yn)。

由此可见，该算法不必计算直线斜率，不必做乘除法，只使用整数加减法操作和一个累加器，占用硬件资源非常少。因此，算法速度快，适合用硬件实现。

3.2.2 椭圆弧命令处理

椭圆弧命令处理单元可以完成任意倾斜角度和任意弧大

小的椭圆弧生成。椭圆弧参数包括椭圆中心点坐标、椭圆的2个半轴长度、椭圆的倾斜角度以及弧的起点和终点。

为简便起见，将任意椭圆弧命令处理分2个步骤进行，首先根据椭圆2个半轴长度参数计算出中心在原点、偏转角为0的椭圆弧，然后进行椭圆中心位置变换和椭圆角度旋转操作。中心在原点、偏转角为0的椭圆弧按照步进式的Bresenham算法生成，具体算法参见文献[4]。由于椭圆算法需要做大量的乘法运算，而椭圆旋转需要做三角函数运算，因此计算量远比线段算法开销大。

为了在精度、速度和资源消耗方面取得平衡，在具体实现上考虑了3个因素：(1)为满足椭圆计算的精度要求，参数计算过程全部采用64位整数，以保证不会产生累计精度损失；(2)为了达到140 MHz以上的独立综合速度设计要求并将资源消耗控制在合理的水平，本文使用了共享多拍运算器的方法，即多个运算步骤共享1个加法器和1个乘法器，这样既节约了FPGA有限且宝贵的乘法器资源，又可实现在100个时钟周期内计算完所有参数的时间要求；(3)对于椭圆旋转所要求的三角函数运算，采取以空间换时间的策略，即设计三角函数查找表，在保证速度的同时尽可能减少对FPGA资源的占用。

3.2.3 矩形和梯形填充命令处理

矩形填充命令处理将矩形填充区直接分解成若干个画水平线，然后送到画水平线控制单元中完成命令执行。梯形填充命令处理单元同时驱动2个线段Bresenham算法控制单元，以步进方式寻找填充区每一水平线的2个端点，从而将梯形的填充分解为水平线。梯形填充命令是一个相对通用的命令，通过参数的选取和设置可以完成任意三角形的填充，在软件的协助下可以完成一般多边形的填充。

4 高速缓存控制器

如上所述，2D图形被分解为点或水平线需要非常频繁但长度较小地随机访问DDR SDRAM中的帧存，而根据SDRAM存储器的操作特点，其少量而频繁的读/写效率远远低于一次大长度的猝发读/写效率。另外，存储操作还需要保证CPU执行程序及LCD显示控制以60 Hz的屏幕刷新频率读取帧存的带宽需求，所以，尽量避免图形加速引擎对帧存储器的零碎存储导致的SDRAM操作瓶颈是十分必要的，因此，在图形加速引擎中引入了高速缓存机制。高速缓存控制系统为图形加速引擎提供了一个透明的DDR SDRAM访问机制，并将图形命令所需要的对DDR SDRAM小而频繁的访问合并为较大长度的猝发读/写操作，从而提高了存储器访问效率。

4.1 高速缓存结构

高速缓存的大小影响到图形引擎的效率和FPGA内部存储资源的占用。由于汉字处理(支持32×32点阵)也是该图形显示系统的一部分并且共用同一高速缓存系统，同时，文献[5]对典型图形图像Benchmark的统计分析表明，在基于贴块(tile)的3D图形渲染系统中，32×32像素的贴块大小是一个合理的选择，因此在本文的设计中采用32×32像素大小的高速缓存以达到性能和资源利用的平衡。

高速缓存结构如图3所示。FPGA内部采用512×64 bit的RAM作为高速缓冲区，以便与位宽64 bit的DDR SDRAM对应。高速缓冲区分成32个页面，每个页面1 024 bit(对应32个真彩像素)，占用16个位宽64 bit的SDRAM地址。高速缓存与DDR SDRAM之间的基本操作单位是页面，并且采用猝发操作交换数据。高速缓存与DDR SDRAM中帧存的页面之间采用全相关策略，因此，对于4 MB大小的帧存(为操作简便，对1 024×768分辨率的真彩显示分配4 MB大小的帧存)，其页面地址为12 bit。

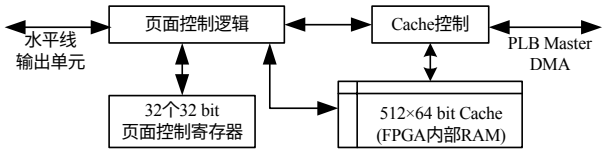

图3 高速缓存控制系统结构

为了对高速缓存进行有效管理，使用32个32 bit的页面控制寄存器，每个缓存页对应一个页面控制寄存器。页面控制寄存器中有12 bit页面地址(Page)、一个修改标志位(Dirty)、一个有效位(Valid)和一个18 bit的计数器(Counter)。Page表示该缓存页中为帧存的第Page页数据，其DDR SDRAM地址为帧存基地址加上页面地址。Dirty指示该缓存页数据是否被修改过。Valid指示该缓存页是否有效。计数器Counter对有效且被修改过的页面在缓存中存在的时间进行计数。

4.2 高速缓存工作流程

高速缓存处理单元收到来自图形处理单元的写存储器命令后，首先将写命令中地址的高19 bit与页面寄存器中的Page域比较，查看目标地址是否在高速缓存中，如果命中(在高速缓存)，则直接修改相应地址的内容；否则，找一个空闲或可以丢弃的缓存页面，并将该页面从SDRAM载入缓存，Valid位置1，这个过程称为“页面置换”，然后在缓存中完成写操作。页面被修改后，将Dirty位置1。Dirty位为1的页面需要及时回写到SDRAM，以保证屏幕内容的及时更新。

4.3 高速缓存页面置换策略

高速缓存置换策略极大地影响了缓存的使用效率。高速缓存页面置换算法的宗旨是尽量将以后或短时间内不会使用的页面淘汰。目前存在多种高速缓存页面置换算法，如随机算法、LRU(最近最久未使用)算法、FIFO(先进先出)算法，其中，LRU算法性能最优也最常见，大多数CPU的指令和数据Cache、操作系统的虚拟存储管理都采用LRU算法。

从表面上看，在本文的图形系统中采用LRU算法是一种恰当的选择，但由于LRU算法是基于程序代码执行和程序数据使用的“局部性”原理提出的，而在本文设计的图形系统高速缓存中缓存的是图形数据，并且图形加速引擎的屏幕画图操作在一定时间范围内大多局限于小的空间区域，因此本文根据图形加速引擎操作的“空间局部性”原理，提出高速缓存的“远区域优先(Farthest Area First, FAF)页面置换”算法。FAF算法的实质就是将离当前图形操作区域最远的区域页面淘汰，所以，该算法的精确实现与帧存的大小和存储器组织方式有关。为简化FPGA设计，具体实现上采取了一种近似策略，即离当前操作地址最远的页面优先淘汰，为提高地址比较速度，使用了多个比较器进行并发操作。仿真测试表明，在缓存大小为32×32像素时，FAF算法与LRU算法性能相近，但FAF算法使用的FPGA资源较少。

5 软硬件协同设计

软硬件划分是嵌入式系统协同设计中独具特色的一个重要阶段，其实质是通过对软硬件资源的合理分配，实现软硬件的有机融合，从而平衡软硬件资源、提高系统整体性能。本文图形加速系统中位图操作和点阵汉字处理是通过结合软

硬件的方式实现的。

5.1 帧存储器一致性

在共享帧存储器结构中，CPU 软件和图形加速引擎都可以对显示帧存储器进行操作，为了避免两者同时对帧存储器操作带来的显示混乱，在 FPGA 中实现了“ TEST_AND_SET ”硬件锁来保证帧存储器的一致性。为了增加软硬件处理的并发度以减小软件操作延迟，图形加速引擎尽量减小上锁“ 粒度 ”，即在每个水平线操作时进行上锁，而不是对每个硬件加速命令操作上锁。CPU 软件在直接操作帧存储器前也需要调用硬件锁，只有当硬件没有进行水平线操作时，才可以对帧存储器进行写操作。

“ TEST_AND_SET ”硬件锁对来自 CPU 软件和图形加速引擎的上锁请求处理并不完全一样。来自图形加速引擎的上锁请求只需实现与 CPU 软件操作的互斥；而对于来自 CPU 软件的上锁请求，在满足互斥要求的条件下，还需要向图形加速引擎的高速缓存控制器发送 Flush Cache 指令排空高速缓存，以保证图形加速引擎中高速缓存与帧存储器的一致性。

5.2 位图及点阵汉字处理

虽然飞行座舱仪表的绝大部分绘图功能都可以通过硬件实现，但是有关自定义位图、地图、复杂仪表图形的初始化等工作更适合在 CPU 的控制下通过 DMA 方式直接从内存传送到帧存储器。在每个 DMA 操作周期，需要对帧存储器上锁，操作完成后进行解锁。

汉字处理采用软件和图形加速硬件相结合的方式实现，即 CPU 软件根据汉字内码在字库获得汉字的点阵，并将汉字点阵作为水平线掩码，调用图形加速引擎的水平线功能完成汉字输出。这种汉字处理方式虽然会使汉字显示速度有所降低，但由于 CPU 性能相对较高，不会导致显示效果和系统整体性能下降，因此从不需要硬字库，从节省 FPGA 资源上看，采用软硬件结合的方式进行汉字显示是合理的。在汉字显示过程中，CPU 软件不需要直接操作帧存储器，因此，不需要调用互斥锁。

6 仿真结果

逻辑设计完成后，采用 Modelsim 6.1 对系统进行了功能和时序仿真。仿真结果表明，汉字、图形显示加速系统在系统时钟为 100 MHz 下，典型的操作时间为：显示 32×32 点阵汉字为 50 μs；清屏为 2 300 μs；在画笔宽度为 1 时画长度为 100 的线段为 130 μs；100×100 的矩形填充时间为 200 μs；画 200×150 的斜椭圆为 230 μs，满足系统设计要求。

为了更方便地进行逻辑功能验证，本文设计了可视化的系统仿真工具。用 Modelsim 对图形操作仿真结束后，将帧存储器模型中的仿真数据转储到一个文本文件中，然后将文本数据转换成 bmp 格式的图片，如图 4 所示。图中第 1 行为 32×32 点阵的幼圆体汉字；第 2 行为 16×16 点阵的宋体汉字；第 3 行显示了 8×16 点阵常用的 ASCII 码字符。第 4 行从左到右显示了 4 个 100×100、分别填充不同的颜色的正方形；画笔宽度为 1 的 20 个点；3 条画笔宽度不同的直线；2 个不同参数的圆；1 条圆弧和 4 个不同参数的椭圆。

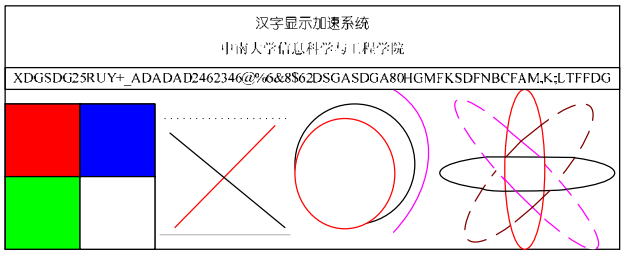

图 4 仿真结果

7 结束语

本文针对当前飞机座舱对图形显示的需求，基于 FPGA 的灵活、可靠和可重配置的特点，提出了基于 SOPC 和图形加速引擎的飞行座舱显示系统方案，通过对基本的 2D 图形处理进行硬件加速来代替以前的纯软件处理，不仅将 CPU 从繁重的图形显示工作中解脱出来，而且大大提高了图形显示性能和系统的稳定性。

参考文献

[1] Chang Chen, Wawrzynek J, Brodersen R W. BEE2: A High-end Reconfigurable Computing System[J]. Design & Test of Computers, 2005, 22(2): 114-125.

[2] Kong Quancun, Li Chenggui, Zhang Fengqing, et al. A New Method of Accelerated Graph Display in Primary Flight Display Based on FPGA[C]//Proc. of the 6th Int'l Symposium on Instrumentation and Control Technology. Beijing, China: [s. n.], 2006.

[3] Zou Xuecheng, Chen Yicheng, Liu Zhenglin, et al. The Design of BitBLT Engine Embedded in Graphics Accelerator for Handheld Devices[J]. Journal of Huazhong Univ. of Sci. & Tech.: Nature Science, 2005, 33(1): 34-38.

[4] David F R. Procedural Elements of Computer Graphics[M]. [S. l.]: McGraw-Hill, 1997.

[5] Iosif A, Ben J, Stamatis V. Selecting the Optimal Tile Size for Low-power Tile-based Rendering[C]//Proc. of the 13th Annual Workshop on Circuits, Systems, and Signal Processing. Veldhoven, Netherlands: [s. n.], 2002.

编辑　张正兴

（上接第 230 页）

参考文献

[1] Chen T, Zhang X, Shi Y Q. Error Concealment Using Refined Boundary Matching Algorithm[C]//Proc. of ITRE'03. Princeton, NJ, USA: [s. n.], 2003.

[2] Jian G T, Chen M J. Effective Error Concealment Algorithm by Boundary Information for H.264 Video Decoder[C]//Proc. of ICME'06. Toronto, Ontario, Canada: IEEE Press, 2006.

[3] Chen M J, Chen C S, Chi M C. Temporal Error Concealment Algorithm by Recursive Block-matching Principle[J]. IEEE Trans. on Circuits and Systems for Video Technology, 2005, 15(11): 1385-1393.

[4] Lam W M, Reilbman A R, Liu B. Recovery of Lost or Roneously Received Motion Vectors[C]//Proc. of ICASSP'93. Minneapolis, Minnesota, USA: IEEE Press, 1993.

[5] Wang Yekui, Hannuksela M M, Varsa V, et al. The Error Concealment Feature in the H.26L Test Model[C]//Proc. of ICIP'02. Rochester, NY, USA: [s. n.], 2002.

编辑　金胡考