文章编号:1671-4598(2018)05-0120-05 DOI:10.16526/j.cnki.11—4762/tp.2018.05.030 中图分类号:TM93 文献标识码:A

基于 ARM11 的 H.264 硬编解码视频传输系统设计

沈 欢[1],彭 力[1],王 皓[2],朱雪芳[3]
(1. 江南大学 物联网工程学院,江苏 无锡 214122;2. 沈阳航空航天大学 自动化学院,沈阳 110000;
3. 江苏联合职业技术学院 无锡机电分院,江苏 无锡 214000)

摘要:为了解决视频信息冗余、视频传输系统成本较高,视频编解码效率低等缺陷,研究了一种基于 H.264 硬编解码器的视频传输系统;该系统采用以 ARM11 为核心的,包含多媒体硬编解码器 MFC 的 S3C6410 作为处理器,使用 CMOS 摄像头采集实时视频数据;在嵌入式 Linux 操作系统上构建基于 live555 的 RTSP 服务器,监听客户端请求;重点阐述了多媒体硬编解码器 MFC 对视频数据进行的 H.264 硬件压缩编码,以及数据包的 RTP 封装,然后经由 live555 流媒体服务器转发至 PC 机;最后,在 PC 上对接收到的数据流进行解码播放,经实验测试证明,系统设计稳定可靠,具有可扩展性强、性能高、成本和功耗低等优点,图像质量和时延满足实际应用需求。

关键词:ARM;多媒体硬编解码;H.264;RTP

Design of H.264 Hard Codec Video Transmission System Based on ARM11

Shen Huan[1], Peng Li[1], Wang Hao[2], Zhu Xuefang[3]
(1. Institute of Networking Engineering, Jiangnan University, Wuxi 214122, China; 2. Institute of automation, Shenyang University of Aeronautics and Astronautics, Shenyang 110000, China; 3. Institute of Wuxi Electrical and Mechanical Branch, Jiangsu Union Vocational and Technical College, Wuxi 214000, China)

Abstract: In order to solve the problems such as video information redundancy, high cost of video transmission system, low efficiency of video codec and other defects, a video transmission system based on H. 264 hard codec is studied; The system adopts S3C6410 which based on the ARM11 core and includes a multimedia hard codec MFC as the processor, uses CMOS camera to collect real-time data, a live555 RTSP server is built on the embedded Linux operating system to listen to client requests; This paper focuses on the H. 264 hardware compression coding of the multimedia hard codec MFC and the RTP encapsulation of the data packet, and then forwards it to the PC via the live555 streaming media server. Finally, the received data stream is decoded on the PC , after testing, the video transmission system is stable and reliable with strong scalability, high performance, low cost, low power consumption and other advantages. it can effectively achieve real-time video transmission, image quality and video delay can meet the application requirements.

Keywords: ARM; multimedia hard codec; H.264; RTP

0 引言

由于视频原始数据中存在着各种冗余信息,如时间冗余、空间冗余等,故对视频原始数据进行压缩编码十分重要[1]。当前,市场上的嵌入式采集方案大多采用 ASIC、FPGA 等专用方案,虽然这种方案有高可靠性,高效率的优点,但其灵活性较差,整体设计周期较长,系统开发成本高[2-3];目前,市场上也有采用双核 DSP 处理器或者是 ARM 和 DSP 双处理器的方案,这两种方案都使用了 DSP,系统编解码速度快,但系统功耗较大,成本也比较高[4-5]。

采用包含多媒体硬件编解码器的高性能单片嵌入式处理器来进行硬件编解码和智能控制,则既降低了系统的功耗和成本,同时也使得系统具有更易维护和升级的优点,若根据系统的硬件体系结构对软件作进一步地优化,便可实现高效运算[6]。通过对嵌入式技术的深入学习和研究,本文提出了一种基于 S3C6410 单核嵌入式处理器的视频传输系统解决方案。

收稿日期:2017-08-20; **修回日期**:2017-10-18。
基金项目:国家自然科学基金(61203147,61374047,61403168)。
作者简介:沈 欢(1994-),男,江苏苏州人,硕士研究生,主要从事网络视频传输方向的研究。
彭 力(1967-),男,河北唐山人,教授,博士生导师,主要从事视觉传感器网络、人工智能、计算机仿方向的研究。

1 视频传输系统总体设计

本视频传输系统主要由流媒体服务器、网络传输模块和 PC 机三部分组成。流媒体服务器采用以 S3C6410 为处理器的 Tiny6410 开发板。首先 Tiny6410 开发板上需要移植 Linux 操作系统,然后在操作系统上部署 live555 流媒体服务器,采用最新一代编解码方式 H.264 对采集的视频数据进行硬编码,使用 RTP 对数据包进行封装,然后发送给 live555 流媒体服务器等待传输;使用 RTP/RTCP 协议将视频流传输给 PC,PC 端对视频流解码后可进行视频的回放。系统总体结构如图 1 所示。

2 视频传输系统硬件开发平台

系统使用的 Tiny6410 开发板采用 Samsung 公司生产的 S3C6410 处理器,S3C6410 是基于 ARM1176JZF-S 核的高性能处理器,主频 667 Mhz,其内部总线结构由 APB、AHB 和 AXI 组成,外围模块通过这些总线与处理器相连[7]。S3C6410 中包含内存管理器 MMU、LCD 控制器、摄像头控制器、4 路 UART 和 1 路 IIC 等丰富资源,内部集成的多媒体编解码器

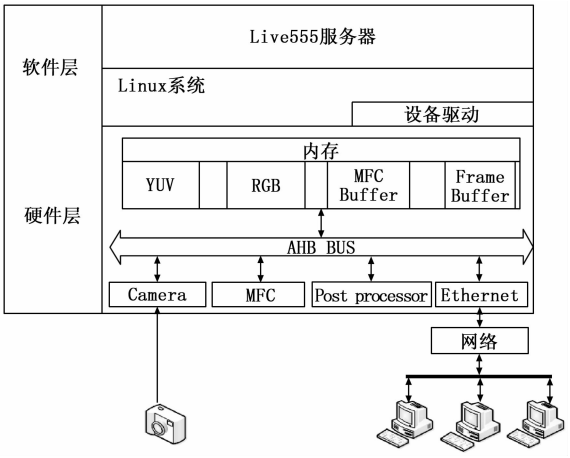

图 1　系统总体结构图

MFC 支持 H.264 等多种音视频格式的编解码，在视频监控、可视电话等开发应用中使用十分广泛，具有体积小、功耗低、处理能力强、速度快等优点。系统硬件框图如图 2 所示。

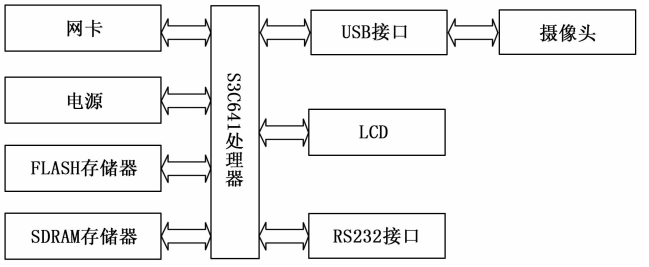

图 2　系统硬件框图

3　视频传输系统软件设计

系统软件设计主要由视频采集、视频编码、视频网络传输以及流媒体服务器四个模块组成，四个模块的设计如下。

3.1　视频采集模块的设计

设备驱动程序是 Linux 内核中的重要组成部分，它主要负责协调操作系统和硬件设备的关系，并提供由操作系统到硬件设备的接口。设备驱动程序为处于用户态的程序屏蔽了硬件的细节，从用户态程序的角度来看，通过类似于系统调用的方式即可实现对设备进行访问和操作。V4L2（Video For Linux two）是内核下编写视频设备驱动程序的规范。通过将视频应用函数封装，V4L2 简化了视频采集程序的开发，提高了系统开发与维护的效率[8]。

V4L2 视频采集的流程如上图 3 所示。视频采集中的关键步骤代码如下：

（1）打开设备文件并查看设备功能

intfd=open("/dev/video0,O_RDWR);

{\mathop{\rm ret}\nolimits} = ioctl(fd, VIDIOC—QUERYSTD, \& std);

（2）设置视频帧格式及制式

{\mathop{\rm if}\nolimits}(ioctl(fd, VIDIOC—S—FMT,\& fmt) < 0)\{

{\mathop{\rm return}\nolimits} 0;

}

（3）向驱动申请帧缓冲

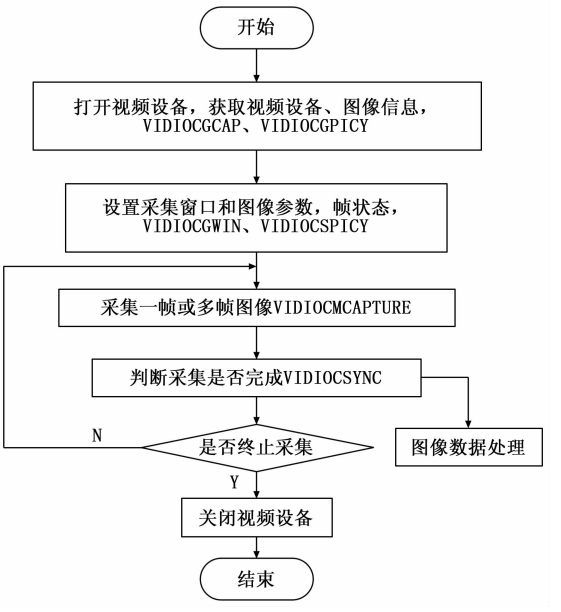

图 3　视频采集流程图

{\mathop{\rm struct}\nolimits} v4l2—requestbuffers req;

{\mathop{\rm memset}\nolimits} (\& req, 0, sizeof (req));

{\mathop{\rm ioctl}\nolimits} (fd,VIDIOC—REQBUFS,\& req)

（4）帧缓冲映射并采集数据

{\mathop{\rm buffers}\nolimits} = calloc(req.count, sizeof(Video Buffer));{\mathop{\rm buffers}\nolimits} [numBufs]. start = mmap(NULL,buf.length,PROT—{\mathop{\rm READ}\nolimits} | PROT_WRITE,MAP_SHARED,fd,buf.m.offset);{\mathop{\rm enum}\nolimits} v4l2_buf_type type;

{\mathop{\rm type}\nolimits} = V4L2_BUF_TYPE_VIDEO_CAPTURE;

{\mathop{\rm f}\nolimits} (ioctl(fd, VIDIOC_STREAMON, \& type) < 0)\{

{\mathop{\rm return}\nolimits} - 1;

}

（5）采集结束，关闭摄像头

{\mathop{\rm ret}\nolimits} = ioctl(fd, VIDIOC_STREAMOFF,\& std);

{\mathop{\rm close}\nolimits} (fd);

3.2　视频编码模块的设计

视频编码主要是将摄像头采集的图像经过压缩编码，形成特定的文件格式或视频数据流。通过对视频数据进行 H.264 编码，既提高了编码效率，同时也增强了网络传输的鲁棒性[9]。

3.2.1　视频编码模型

MFC 主要对存放在缓存中的摄像头数据进行编码。MFC 硬件部分分为 BIT processor 模块和视频编解码核心模块，BIT processor 模块主要负责接硬件加速、编解码速率控制以及接收处理器信号；视频编解码核心模块主要负责运动估算、帧间预测以及帧内预测，并通过 BIT processor 将数据存放到 MFC 的输出缓存中。通过模块的共享，MFC 既能支持多种格式的编解码，又大大降低了硬件的冗余。

MFC library、MFC driver 和用户程序共同组成了视频编码的软件部分。其中，MFC driver 通过在内核中注册一个

misc 设备，在文件系统中生成/dev 目录下的 s3c－mfc 设备文件，进而可以对该设备文件进行 iotcl 操作。MFC driver 在初始化时先后完成：进行设备注册、打开系统时钟、进行寄存器物理地址映射、注册中断 interrupt、获取显存 framebuffer、设置位处理器缓存、对 MFC 硬件进行初始化。MFC 硬编解码模型框图 4 所示。

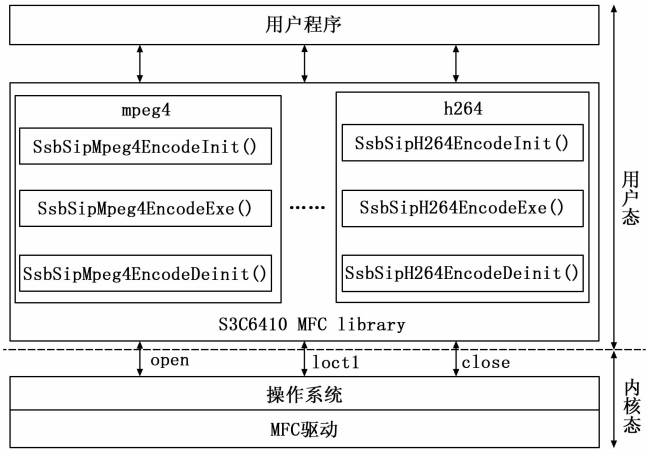

图 4 MFC 硬编解码模型框图

3.2.2 视频编码程序设计

使用 MFC 进行视频的编解码可以通过调用 MFC library 中的函数实现，使得开发人员更多地专注于多媒体应用开发。使用 MFC 进行视频编码的简要步骤如下：

（1）打开输入设备 video

defineCODEC_NODE"/dev/video0"

{\mathop{\rm dev}\nolimits} _fp = open(CODEC_NODE, O_RDWR);

（2）获取设备的 capability 属性

{\mathop{\rm ret}\nolimits} = ioctl(cam_c_fp , VIDIOC_QUERYCAP, \& cap);

（3）设置设备参数

{\mathop{\rm codec}\nolimits} _fmt. type = V4L2_BUF_TYPE_VIDEO_CAPTURE;{\mathop{\rm codec}\nolimits} _fmt. fmt. pix. width = LCD_WIDTH;

{\mathop{\rm codec}\nolimits} _fmt. fmt. pix. height = LCD_HEIGHT;

{\mathop{\rm codec}\nolimits} _fmt. fmt. pix. pixelformat = V4L2_PIX_FMT_YUV420; {\mathop{\rm ret}\nolimits} = ioctl (cam_c_fp , VIDIOC_S_FMT, \& codec_fmt);

（4）MFC 初始化

{\mathop{\rm handle}\nolimits} = mfc_encoder_init(LCD_WIDTH,LCD_HEIG{\mathop{\rm HT}\nolimits} , framerate, 1000, 30); / * Codec start * /

（5）设置 MFC 的状态

{\mathop{\rm start}\nolimits} = 1;

{\mathop{\rm ret}\nolimits} = ioctl(cam_c_fp, VIDIOC_STREAMON, \& start);

（6）循环读取数据

{\mathop{\rm while}\nolimits} (1)\{

{\mathop{\rm read}\nolimits} (cam_c_fp, g_yuv, YUV_FRAME_BUFFER_SIZE);{\mathop{\rm f}\nolimits} (frame_count = = 1) \{

{\mathop{\rm encoded}\nolimits} _buf = (unsignedchar *)mfc_encoder_exe(handl{\mathop{\rm e}\nolimits} ,g_yuv, YUV_FRAME_BUFFER_SIZE,1,\& encoded_size);}else

{\mathop{\rm encoded}\nolimits} _buf = (unsignedchar *)mfc_encoder_exe(h

{\mathop{\rm andle}\nolimits} ,g_yuv, YUV_FRAME_BUFFER_SIZE, 0,

&encoded_size);

}

其中，MFC 编码初始化函数 mfc _ encoder _ init 通过调用 SsbSipH264EncodeInit 函数对 MFC 进行初始化；mfc _ encoder _ exe 则调用 MFClibrary 中的 SsbSipH264EncodeGetInBuf 函数将要编码的数据放入 MFC 的输入缓存，再通过 SsbSipH264EncodeExe 函数对帧数据进行编码，最后通过 SsbSipH264EncodeGetOutBuf 函数获得编码后帧的地址。

3.3 视频网络传输模块的设计

3.3.1 H.264 的 RTP 封装

NALU（NAL unit）数据流是由网络适配层对 H.264 的原始数据进行处理后形成的。下面对 H.264 视频流三种 RTP 负载格式进行介绍：当一个 RTP 包能放置一个 NALU 而不能放置两个 NALU 的时候，采用单个 NAL 单元包封装方式；聚合包主要是在 RTP 负载包能够放置两个或两个以上的 NALU 的时候，为了提高网络的传输效率，将多个 NALU 打包到一个 RTP 包中。当 RTP 负载包无法装载一个 NALU 时，则需要对 NALU 进行分片。H.264 数据包分片步骤可见于图 5，其分片规则如下：

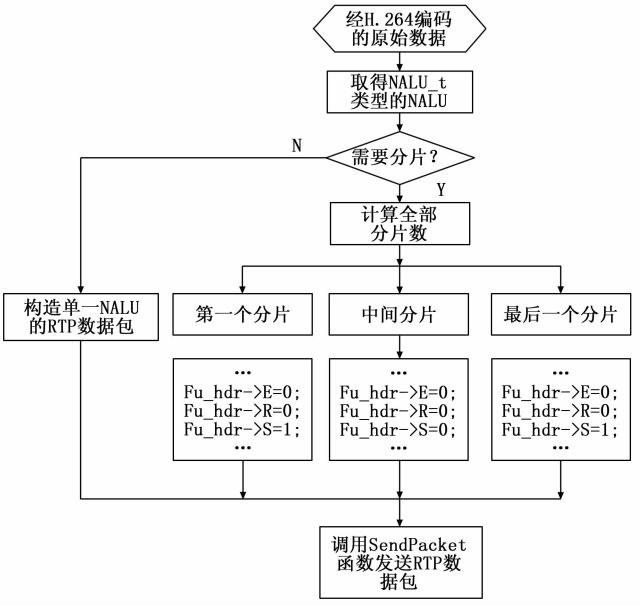

图 5 H.264 数据包的分片步骤框图

1）聚合包无法分片，只有单个 NALU 可进行分片。

2）NALU 字节是连续的，且整数个 NALU 字节构成一个 NALU。

3）发送时，NALU 不同分片的 RTP 序列号是递增有序排列的；接收时，则必须按照与发送时相同的序列号排列合并。

3.3.2 JRTPLIB 库的移植与使用

JRTPLIB 内部用 C++实现了 RTCP 机制，为视频的网络传输提供了一个可靠的 RTP 协议库[11]。JRTPLIB 库的移植步骤如下：

1）下载并解压 jthread、jrtplib 的源代码

2）配置编译环境，对 jthread 进行编译

./configure－host＝arm－linuxCC＝arm－linux－gccCX{\mathop{\rm X}\nolimits} ＝ arm － linux － g ＋ ＋

make

makeinstall

3）对 JRTPLIB 进行编译

./configure－host＝arm－linuxCC＝arm－linux－gcc{\mathop{\rm CXX}\nolimits} ＝ {\mathop{\rm arm}\nolimits} － {\mathop{\rm linux}\nolimits} － {\mathop{\rm g}\nolimits} ＋ ＋ － {\mathop{\rm with}\nolimits} － {\mathop{\rm jthread}\nolimits} － {\mathop{\rm includes}\nolimits} ＝ /usr/local/{\mathop{\rm include}\nolimits} /jthread CPPFLAGS ＝ － I/usr/local/include/jth{\mathop{\rm read}\nolimits} LDFLAGS ＝ － L/usr/local/lib － ljthread

make

makeinstall

4）在库目录中添加编译生成的 .a 文件和 .so 文件，完成 JRTPLIB 的移植。

在使用 JRTPLIB 库之前，先要对一些参数进行相应的设置。初始化 RTP 会话中的一个重要工作是设置合适的时间戳，通过调用 RTPSession 类中的 SetTimestampUnit（）函数可实现以秒为单元的时戳单元的设置；RTP 会话的实例可通过 RTPSession 类中的 Create（）函数来实现初始化；会话的通信端口和通信地址则可通过 AddDestination（）函数实现。最后，应用程序在传输 RTP 数据包时可通过调用 SendPacket 函数实现。

3.4　基于 live555 的流媒体服务器构建

本文在嵌入式 Linux 操作系统上部署 live555 服务器，完成了流媒体服务器的构建。live555 是一个针对流媒体服务而提出的 C++解决方案，它支持多种多媒体传输协议。该服务器框架中主要包含四个模块：UsageEnvironment 模块包含抽象类 TaskScheduler，主要是对系统环境进行抽象；BasicUsageEnvironment 模块主要完成对输入输出信号响应的实现；而 GroupSock 模块则主要用于数据包的接收和发送；最重要的是 GroupSock 模块，其包含了 RTCPInstance、RTSPClient、Sink、MediaSession 和 Source 众多派生类。

该流媒体服务器工作在阻塞模式，服务器初始化时，首先需要进行 RTSP 端口的绑定，然后进行 RTSP 监听，当有用户连接时，服务器会先通知 RTP 打包模块对数据进行打包，接着服务器再将 RTP 打包好的数据包转发。live555 服务器运行流程如图 6 所示。

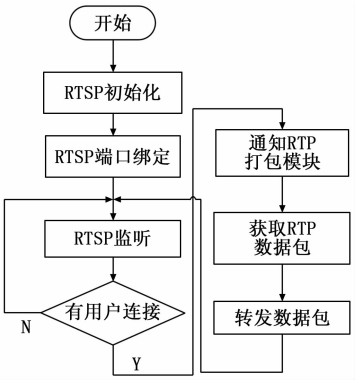

图 6　live555 服务器运行流程图

4　系统测试

在 PC 上安装支持多种流协议的 VLC 播放器，在 VLC 播放器中打开流文件并选择 RTP 协议，即可接受解码并播放 tiny6410 开发板通过局域网传送过来的 RTP 流。设置流媒体服务器 IP 为 192.168.1.230，端口为 8000。在 PC 端进行如下测试：

先运行 tiny6410 上的流媒体服务器，SecureCRT 显示“The server is running”，在 PC 端运行 VLC 播放器，输入 URL：RTSP：//192.168.1.230：8000/，点击播放。视频传输实时性良好，系统实物测试和 VLC 端显示如图 7 所示。

图 7　系统实物测试和 VLC 端显示图

由于时间及条件等方面的限制，该视频传输系统存在一定的时延问题。根据系统测试环境，预先在系统接收端设置 500 ms 的缓冲区域，但系统播放的实际时延高于 500 ms，在 1～1.5 s 之间。VLC 视频播放统计信息如图 8 所示。系统运行 2 小时，平均传输速率为 165 kb/s，平均丢包率为 2.2%，满足国家视频监控的标准。由于视频数据是经压缩编码后再进行传输的，这极大地提高了数据传输的效率，同时也降低了网络的压力。

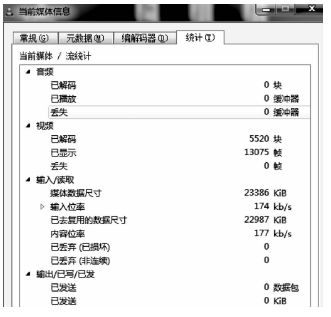

图 8　VLC 统计信息

5　结论

本文分别从硬件和软件两个角度出发，阐述了基于 ARM11 的 H.264 硬编解码视频传输系统的设计与实现。本系统采用以 ARM11 为核心的 S3C6410 作为处理器，使用外设摄像头获取实时视频数据；在 Tiny6410 上移植 Linux 操作系统并构建 live555 流媒体服务器，监听用户请求；采用处理器内置的多媒体硬编解码器 MFC 对视频数据进行 H.264 编码，使

编码效率得到了有效的提高；使用 RTP 对数据包进行封装，然后经由 live555 流媒体服务器转发至 PC 机；经测试，视频还原流畅，传输实时性良好，满足实际应用需求。

参考文献：

[1] 高 敏. 视频图像压缩中熵编码技术研究 [D]. 哈尔滨：哈尔滨工业大学，2016.

[2] 郭诚欣，陈 红，孙 辉，等. 基于现代硬件的并行内存排序方法综述 [J]. 计算机学报，2016 (1)：24.

[3] 杜晓婧，李树国. SHA－1 算法的高速 ASIC 实现 [J]. 微电子学与计机，2016 (10)：19－23.

[4] 任志玲，朱光泽. 基于嵌入式和运动检测的井下视频监控系统 [J]. 计算机测量与控制，2014，22 (5)：1398－1400.

[5] 何登平，黄凌云，何 策. 基于 DM3730 的高清智能视频采集处理系统的设计与应用 [J]. 光电子技术，2015 (4)：283－288.

[6] Xia T，Prevotet J C，Nouvel F. Mini－NOVA：A Lightweight ARM－based Virtualization Microkernel Supporting Dynamic Partial Reconfiguration [A]. Parallel and Distributed Processing Symposium Workshop [C]. IEEE，2015：71－80.

[7] Yin Q，Zhang J. Development of remote video monitoring system based on TCP/IP [A]. International Conference on Computer Science & Education [C]. IEEE，2015：596－600.

[8] 田时舜，章明朝，周 跃，等. 基于 DM8148 的嵌入式网络视频服务器设计 [J]. 计算机工程与设计，2015 (5)：1192－1196.

[9] Zeng B，Yeung S K A，Zhu S，et al. Perceptual Encryption of H.264 Videos：Embedding Sign－Flips Into the Integer－Based Transforms [J]. IEEE Transactions on Information Forensics & Security，2014，9 (2)：309－320.

[10] 张纪宽，彭 力，陈志勇. 动态复杂背景下的智能视频监控系统设计与实现 [J]. 计算机测量与控制，2016，24 (7)：100－104.

[11] Belyaev E，Vinel A，Surak A，et al. Robust Vehicle－to－Infrastructure Video Transmission for Road Surveillance Applications [J]. IEEE Transactions on Vehicular Technology，2015，64 (7)：2991－3003.

(上接第 92 页)

由表 1 可求得 DCS 失电故障分散控制系统总风险，由于各个负荷点断电概率和断电损失不同，导致各个负荷点上的断电风险也不同。

为了使风险评估结果更具有可靠性，分别应用传统维护平台与改进维护平台，并对两种平台的控制系统的准确性进行对比，结果如图 5 所示。

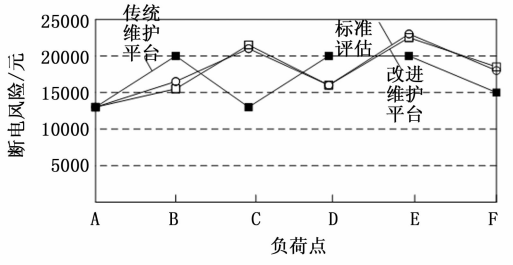

图 5 两种维护平台风险评估准确率对比结果

由图 5 可知：当负荷点为 A 时，两种评估方法的风险值为 13 898.15 元，都与标准评估风险一致；当负荷点为 B 时，传统维护平台的评估风险值为 20 000.00 元，而改进维护平台所得的风险值为 16 000.83 元，而标准评估风险值为 16 130.82 元，明显看出，改进维护平台所得的风险值与标准值更为接近。不同负荷点处的风险值不同，逐渐显示出传统维护平台风险评估不准确，而改进维护平台所得的风险值基本与标准值一致。

4.4 实验结论

根据上述实验内容，得出实验结论：由于 DCS 失电故障分散控制系统出现故障负荷点不同，所说需要测试不同风险值。采用传统维护平台进行风险评估时，所获得的风险值波动范围较大，而采用改进维护平台进行失电故障风险评估时，所获的风险值波动范围较小，且与标准值基本一致，由此可知，改进维护平台比传统维护平台对失电故障风险分析的准确率更高。

5 结束语

改进设计的 DCS 失电故障分散控制系统维护平台能够有效满足失电故障风险评估的需求，通过在实际应用中不断对暴露的故障点进行分析，方便后续相关工作人员对故障风险进行准确评估。针对故障风险需从定义着手开始分析，从引起的外部故障断电风险原因进入到 DCS 失电故障分散控制系统故障断电风险分为外部原因和内部原因，对 DCS 失电故障分散控制系统中常出现的失电故障进行分析，建立风险评价指标，并以该指标为基础构建失电故障风险评估数学模型。实验验证可知，改进的控制系统维护平台比传统维护平台对失电故障风险分析的准确率更高，为其他安全风险监测提供保障。

参考文献：

[1] 王东风，刘 千，韩 璞，等. 基于大数据驱动案例匹配的电站锅炉燃烧优化 [J]. 仪器仪表学报，2016，37 (2)：420－428.

[2] 吕 荣，L (U) Rong. 基于大数据处理技术的 AIS 应用研究 [J]. 海军工程大学学报，2017 (4)：98－102.

[3] 王宁玲，付 鹏，陈德刚，等. 大数据分析方法在厂级负荷分配中的应用 [J]. 中国电机工程学报，2015，35 (1)：68－73.

[4] 付 鹏，王宁玲，李晓恩，等. 基于信息物理融合的火电机组节能环保负荷优化分配 [J]. 中国电机工程学报，2015，35 (14)：3685－3692.

[5] 王 刚，梁正玉，李存文，等. 大数据分析在汽轮机调节门特性参数辨识及优化中的应用 [J]. 中国电力，2016，49 (6)：15－19.

[6] 闫秀侠，巴忠松. 失磁故障对多极少槽永磁同步电机的影响 [J]. 电子设计工程，2017，25 (10)：102－104.

[7] 霍 文. 基于现场总线技术的准能矸石电厂 DCS 系统改造 [J]. 煤炭工程，2015，47 (11)：33－35.

[8] 程保华，张合厂，熊 科，等. 核电站 DCS 主从控制器切换后信号跳变的分析与解决 [J]. 现代电子技术，2015，450 (19)：140－142.

[9] 王玉萍，曾 毅. 基于小型 PLC 的电网智能监测 DCS 控制器设计 [J]. 电网与清洁能源，2016，32 (12)：51－56.

[10] 乐安胜，谢红辉. DCS 控制系统在铜电解生产中的应用 [J]. 中国有色冶金，2017，46 (2)：45－48.

[11] 王玉萍，曾 毅. 基于小型 PLC 的电网智能监测 DCS 控制器设计 [J]. 电网与清洁能源，2016，32 (12)：51－56.