基于 RF5的 DSP开发方法的研究

Research on DSP Development Method Based on RF5

史　英　张卫宁　王　玮　张大志　吕晓平
(山东大学信息科学与工程学院, 济南　250100)

摘　要: DSP是一种数字信号处理器。作为 eXpressDSP软件关键组件的 TMS320DSP算法参考框架 RF, 标准化了算法和系统软件的接口, 使得算法使用者的系统集成任务大为减轻, 产品开发周期相应大为缩短。介绍了适用于大型系统的参考框架 RF5, 并结合 H. 263算法实例, 分析了 RF5在复杂程序中的应用及开发。研究证明, RF5作为一种通用初始化代码, 只需做很少的改动就可用于各种多通道复杂算法的应用程序。

关键词: eXpressDSP软件　参考框架 RF5　H. 263算法实例
中图分类号: TP368 1　　文献标志码: A

Abstract　DSP (digital signal processing), as the eXpressDSP software key module, the TMS320DSP algorithm reference frameworks (RF) standardize the interface of algorithm and system software, thus the system integration assignment of users are decreased dramatically, and the researching and developing period of products is shortened. Reference framework 5 intended for large-scale system is mainly introduced. With H. 263 algorithm as example, the application and development of RF5 in complicated programs are analyzed. The research indicates that as a general initialization code, RF5 can be used in all DSP applications of multi-channels and complicated algorithms only with minor modifications.

Keywords　eXpressDSP software　Reference framework 5 (RF5)　Example of H. 263 algorithm

0　引言

随着 DSP (digital signal processing)芯片类型的增多和技术的不断提高, DSP向着多功能化、高性能化、低功耗化方向发展, DSP硬件技术的更新速度也不断加快, 然而相关软件技术的开发却远远落后于硬件的开发。基于 DSP应用的大型系统往往因其复杂而繁琐的软件设计困扰着设计人员, eXpressDSP是美国 TI公司提出的一项实时软件开发技术, 集成了 CCS(code composer studio)开发平台、DSP BIOS实时软件平台、DSP算法标准 XDAIS(TMS320DSP algorithm standard)和第三方支持等四部分, 利用它可以加快用户开发 DSP软件的过程。Reference Frameworks是一种 eXpressDSP技术下的软件框架, 它将 DSP目标应用程序、XDAIS算法和底层结构结合在一起, 形成一个空间, 该空间就定义为参考框架。图 1是参考框架的总体结构, 最底层是 DSP硬件, 其上是基于标准驱动模块的设备驱动控制器和适配器, 用于完成多线程 I/O请求的序列化功能和同步功能, 上一层是芯片支持库 (CSL)和 DSP/BIOS实时操作系统, 最顶层是应用程序, 在应用程序和 DSP/BIOS之间就是参考框架的各个组件, 如通道管理等。

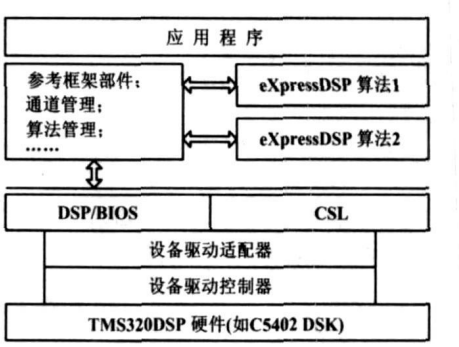

图 1　参考框架的结构

Fig. 1　Structure of a reference framework

参考框架也可以定义为一种使用 DSP/BIOS和 TMS320DSP算法标准的通用 DSP启动程序源代码, 就如同一个应用软件的蓝图, 其中已经包含存储器管理、线程模块和通道封装等复杂的基础结构, 因此, 开发者可以不用考虑这些基础结构是怎样工作的, 而只需要考虑个人的系统特征。到目前为止, TI公司共提供了 RF1、RF3和 RF5三种框架标准, 其中参考框架标准 5 (RF5)支持 1~100个通道和大量符合 eXpressDSP标准的算法, 可用于基于多通道下复杂算法的应用程序

修改稿收到日期: 2006-11-27。

第一作者史英, 女, 1981年生, 现为山东大学信息科学与工程学院在读硕士研究生; 主要研究方向为数字视频处理与通信和基于 DSP的嵌入式系统设计。

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

中。与低标准的参考框架相比, RF5不但支持更多的通道和算法,而且用到了线程模块,因此, RF5非常适合应用于线程间有着复杂联系的应用程序,如视频处理、3G无线基础设备等。

1　RF5

RF5是一种大规模集成的参考框架,其专为多通道多算法的应用程序而设计,它是基于 DSP/BIOS实时操作系统的 TSK模块的应用,因此适用于大型 DSP应用系统。RF5包括: ①XDAIS算法示例; ②可升级的通道管理,通道中 XDAIS算法的数目可增减; ③使用临时内存,可有效地共享 XDAIS临时资源; ④保护线程的结构化控制,微处理器发信息给控制线程,控制线程将收到的信息发给 DSP任务线程。

RF5主要有数据处理和数据通信两个功能。

1.1　RF5中的数据处理

RF5中的数据处理包括任务 (task)、通道 (channel)、单元 (cell)和 XDAIS算法四个层面。图 2所示的 RF5处理元素示意图说明了任务、通道、单元和 XDAIS算法的关系。

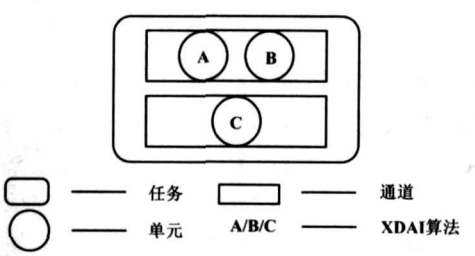

图 2　RF5处理元素

Fig. 2　Processing elements in RF5

每一个任务都是由若干个通道组成的,用以完成特定的操作。通道是单元的集合,单元在通道中顺序执行。单元是 XDAIS算法的封装,而每一个 XDAIS算法负责完成一项特定的数据处理功能。

XDAIS算法是按照 eXpressDSP算法标准开发的适用于 TI的 DSP器件的算法,由 TI公司的第三方提供,可以重复利用,这种按标准开发的 DSP算法无需或只需很少的修改就可以集成在新的 DSP系统中。eXpressDSP算法标准保证了用户的当前程序同其它程序的无缝连接,同时也保证了程序的延续性。为了适应 eXpressDSP算法标准的要求, XDAIS算法不允许直接访问硬件外设,必须通过标准的资源管理接口来实现,例如内存管理使用 IALG接口,而对 DMA资源管理则使用 IDMA接口。XDAIS算法可以实现相对复杂的功能,如 JPEG的编解码等,也可实现相对简单的功能,如声音的放大等。每个算法执行的功能各不相同,为简化算法的使用, RF5引入了“单元”的概念,一个单元是一个 XDAIS算法的标准封装,位于 XDAIS算法核心的数据处理功能在功能特征和数据处理量上都不相同,单元的作用就是为算法和外界提供一个标准接口 (ICELL),通过该接口,算法可与外界进行数据传输。每一个单元都对应一个 ICELL接口,该接口可为每一个单元定义打开 (open)、执行 (execute)、关闭 (close)和控制 (control)四个函数,除了执行函数,其他函数都是可选的,用户必须在应用程序中为每一个算法创建一个 ICELL_Fxns数据结构和 ICELL_Obj数据结构,前者定义单元的四个功能函数,后者则包含各单元的相关信息,结构如下:

```
typedef struct ICELL_Fxns{
        Bool(* celClose);
        Int (* celControl);
        Bool(* celExecute);
        Bool(* celOpen);
} ICELL_Fxns;
typedef struct ICELL_Obj{
        Int     size;
        String  name;
        ICELL_Fxns * cellFxns;
        Ptr     cellEnv;
        IALG_Fxns * algFxns;
        IALG_Params * algParams;
        IALG_Handle algHandle;
        Uns     scrBucketIndex;
    ICC_Handle * inputIcc;
    Uns     inputIccCnt;
    ICC_Handle * outputIcc;
    Uns     outputIccCnt;
} ICELL_Obj;
```

通道是单元的集合,其作用是通过依次调用算法接口,连续执行包含在单元之内的 XDAIS算法。用户不需要对通道写额外的代码,但需要给每一个通道创建相应的通道对象数据结构 CHAN_Obj下面是一个通道的对象结构体:

```
typedef struct CHAN_Obj{
    ICELL_Obj * cellSet;
    Uns     cellCnt;
    CHAN_State  state;
Bool(* chanControlCB) (CHAN_Handle chanHandle);
} CHAN_Obj;
```

任务中包含一个或多个通道,任务将依次执行这些通道,并与设备驱动及其他任务进行数据通信。任

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

务有具体的代码, 该代码由用户编写, 负责与外界进行数据交互及发送控制消息。任务中可以不含任何通道, 此时用户不需要创建通道对象。任务的另一重要特征就是除了传输常规的数据流之外, 它可以根据情况发送控制消息到其他任务。基于此种特性, 每一个任务都循环采用"获取数据"、"执行通道"、"发送数据"的操作, 在循环起始前检查一下是否收到控制消息, 如果有, 任务则根据该消息修改其控制逻辑或修改通道内所包含的单元。

1.2　RF5中的数据通信

RF5中的数据通信分为任务级的数据通信和单元级的数据通信。

1.2.1　任务级的数据通信

RF5采用 SIO对象和 SCOM消息支持任务间的数据传送。SIO对象是设备驱动和任务之间的接口, 如图 3所示, 这些对象都是标准的 DSP/BIOS对象, 采用乒乓式的双缓冲区结构。任务将为数据分配两个缓冲区, 将其中空的缓冲区提供给输入设备驱动的同时, 从设备驱动取走装载数据后的缓冲区。

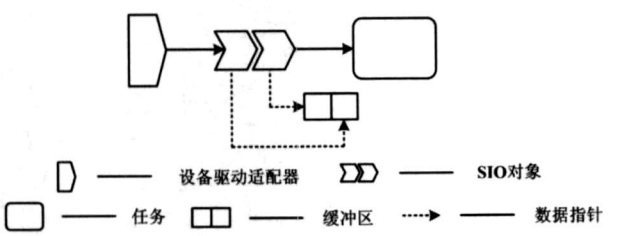

图 3　任务和设备驱动间通过 SIO对象的通信

Fig 3　Communication between a task and a device driver via an SIO object

同步通信 SCOM(synchronization communication)消息, 是由用户定义的结构化对象, 可在任务之间互相传递。SCOM消息可如下定义:

```
structmyMsg {
QUE_Elem      elem;
/* 必须作为 SCOM消息结构体的第一变量* /
Int      someField;
        ……
        } myMsg
```

任务之间可通过 SCOM消息实现任务的同步和互斥访问。程序运行期间, 每一个任务将分配一些缓冲区供其它任务读取或者写入数据, 而 SCOM消息的功能就是指明这些缓冲区的位置, 即消息中包含缓冲区的地址, 一旦其他的任务获取到该 SCOM消息, 就可自由地访问这些缓冲区。每个任务都有自己的消息接收队列, 队列中存放着各任务发送来的 SCOM消息, 当程序运行到该处时, 任务将检查自己的消息接收队列中是否有 SCOM消息, 有则读取, 没则等待。

以下是 RF5使用 SCOM的一个简单例子, 如图 4所示, 写任务可分配一个缓冲区, 并向该缓冲区写入数据, 当缓冲区中充满数据后, 写任务会发送 SCOM消息给读任务, 消息中包含着缓冲区的地址指针, 读任务从自己的接收队列中检测到写任务发来的 SCOM消息后, 将读取消息中指针所指向的缓冲区中的内容。

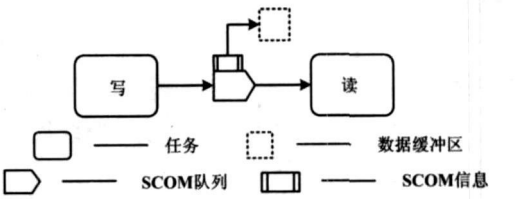

图 4　两个任务之间的通信

Fig 4　Communication between two tasks

1.2.2　单元级的数据通信

单元级的数据通信是由 ICC(inter-cell communication)对象实现的。ICC对象的功能是描述单元读取数据或者写入数据的缓冲区, 每一个单元都有一个 ICC对象的输入列表和一个输出列表, 列表中可有一个或多个输入/输出 ICC对象。ICC对象所指向的缓冲区可由用户来定义。如果某个单元有一个 ICC对象在其输出列表中, 则单元可以通过这个 ICC对象来写缓冲区。如果 ICC对象在单元的输入列表中, 则单元可以通过这个 ICC对象来读缓冲区。对于同一任务内不同通道之间的单元也可以这样通信。最简单的 ICC对象只包括了缓冲区的位置和大小, 又叫线性 ICC缓冲区。使用 ICC对象可以使通道内的数据通信更加灵活, 以图 5为例, 通道中有 *X*、*Y*和 *Z*三个单元, *X*有一个输入两个输出, *Y*有一个输入一个输出, *Z*有两个输入一个输出, 但程序只使用了 5个 ICC对象就实现了各个单元之间的数据通信。

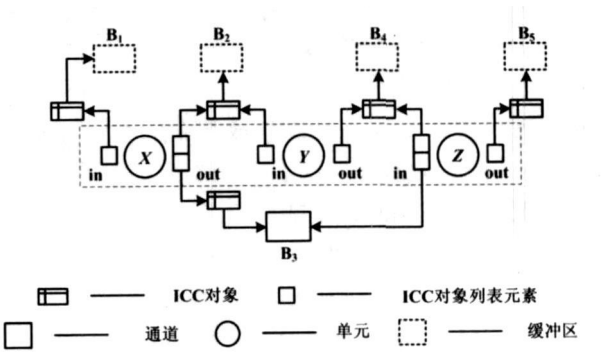

图 5　ICC对象使用实例

Fig 5　Example using ICC objects

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

2　H. 263编码实例

实验程序使用 RF5框架来整合 H. 263编码库解码库。图 6为 DSP端的软件框架,主要包括输入任务(tskInput)、处理任务(tskProcess)、输出任务(tskOutput)三个任务线程。在进入 DSP/BIOS任务调度程序前,程序初始化系统中用到的多个模块。初始化之后,系统会进入由 DSP/BIOS调度管理的三个任务线程。这三个任务线程通过 RF5中的 SCOM 模块互相发送消息:

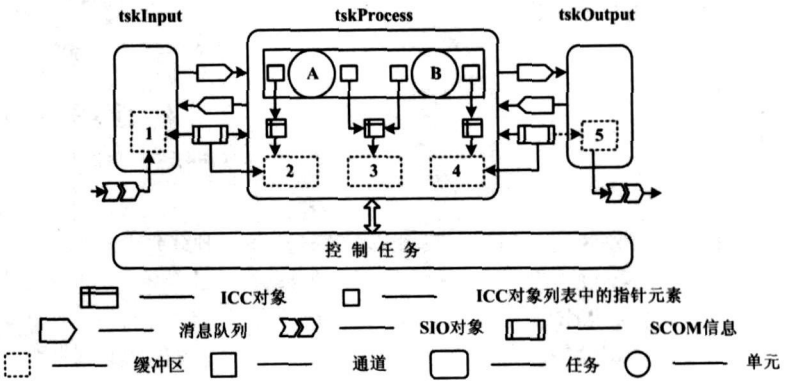

图 6　H. 263软件模块组成

Fig. 6　Composition of H. 263 software module

① 输入任务通过 SIO对象从输入设备驱动程序获得视频图像,存放在缓冲区 1中,并发送 SCOM消息给处理任务,消息中包含图像数据的存储地址,然后等待处理任务发送来的消息以继续接收数据。

② 处理任务包含编码单元和解码单元,负责对图像数据进行编码,并传送编码码流到解码模块,然后将解码后的图像传输到输出模块。处理任务首先检查接收 SCOM队列中是否有输入任务发送来的 SCOM消息,如果有,处理任务将读取消息所指向的缓冲区中的内容,并通过执行 RF5通道来实现视频的编码解码和显示。程序中的 RF5通道包含一个 H. 263编码单元 A和一个 H. 263解码单元 B。在通道执行阶段,首先执行单元 A,产生编码码流,并将码流放入单元 A的输出 ICC对象指向的缓冲区 3中,单元 B从中取出编码码流进行解码,生成图像,将图像数据放入缓冲区 4中,ICC模块管理着编码单元 A产生的码流传送到解码单元 B这一过程,至此,处理任务完成了一帧图像的编解码处理,需要发送 SCOM消息给输出任务,消息中需含有解码后所生成的图像的存储地址,并等待从输入任务来的新消息以继续下一帧图像的处理。

③ 输出任务收到处理任务发来的 SCOM消息,从缓冲区 4中读取图像数据到自己的缓冲区 5,然后通过底层驱动,把数据送给 DSP视频输出端口,并给处理任务发送消息,通知其进行下一帧图像的处理,并等待处理任务来的消息以继续运行。

3　结束语

由上述介绍可以看到,RF5是一个扩展性很强的软件参考框架,其设计目的是让开发人员避开复杂的底层设计,创建基于多通道下复杂算法的应用程序。完全由 C代码编写的特性使得框架易于移植和修改,开发者可以将一个或多个框架作为自己的应用程序的基础。可升级的通道管理特性使得程序可包含大量算法的同时使用尽可能少的任务线程,算法之间能高效地共享临时数据内存,而且通过 ICELL接口可以方便地更换各种算法。SCOM模块可以帮助完成高效的任务间通信,便于多任务多线程应用程序的开发。这些特性必将使得该软件框架得到更广泛的应用。

参考文献

1　彭启琮,管　庆. DSP集成开发环境: CCS及 DSP/BIOS的原理与应用[M]. 北京: 电子工业出版社, 2004: 195-350.

2　朱海涛,戚文芽. 基于 TMS320DM642的 H. 264视频解码器设计[J]. 电视技术, 2005, (4): 25-28.

3　Texas Instruments Incorporated. Reference Frameworks for eXpressDSP Software: RF5, An Extensive, Higlr-Density System[EB/OL]. [2003-04]. http://www.ti.com.

4　Texas Instruments Incorporated. Reference Frameworks for eXpressDSP Software: API Reference[EB/OL]. [2003-04]. http://www.ti.com.

5　Texas Instruments Incorporated. Reference Frameworks for eXpressDSP Software: A White Paper[EB/OL]. [2003-04]. http://www.ti.com.

6　范运兴,王忠勇. 基于实时数据交换技术的数据通信[J]. 自动化仪表, 2006, 27(4): 17-19.

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net