面向ARM平台的音视频编码优化研究

姜春林[1] 贾维嘉[1,2] 张历卓[1,2] 谷 科[1]
(中南大学信息科学与工程学院 长沙 410083)[1] (香港城市大学计算机科学系 九龙)[2]

摘 要 ARM是性价比很高的嵌入式内核,非常适合作音视频编码系统的CPU。ARM硬件上支持复杂的DSP操作,通常可以在一个时钟周期内完成,极大地提高了音频编码程序的运行速度。在视频编码的数据处理指令中,使用桶形移位器进行移位预处理可以增加代码密度,从而减少取指令次数;而充分利用ARM的16个通用寄存器进行数据操作,也可以减少内存存取操作。实验证明,ARM可以有效提高音视频编码的运行性能。

关键词 ARM内核,音频编码,H.263编码
中图法分类号 TP391 **文献标识码** A

Optimization of Audio and Video Encoder on ARM Platform

JIANG Chun-lin[1] JIA Wei-jia[1,2] ZHANG Li-zhuo[1,2] GU Ke[1]
(College of Information Science and Engineering, Central South University, Changsha 410083, China)[1]
(Department of Computer Science, City University of Hongkong, Kowloon, China)[2]

Abstract ARM is appropriate for audio and video encoder. In audio encoding, ARM supports digital signal processing, and thus can finish one DSP operation in one cycle. In video encoding, the running time can be reduced rapidly by making full use of 16 registers and using shift operation along with other operation in one instruction. Experiment proves that ARM can dramatically improve performance of audio and video encoder.

Keywords ARM core, Audio encoder, H.263 encoder

ARM(Advanced RISC Machines)内核是当今世界上使用得最多的嵌入式内核之一。它应用于众多领域,包括网络、成像系统、车载系统和移动通信等,其中在移动音视频通信领域中的应用尤其广泛。ARM内核比较小巧,功耗也较低,特别适用于手机和PDA中;ARM支持数字信号处理(DSP)指令,特别是ARMv5和ARMv6内核支持增强的扩展指令[1],使得其在音视频通信中占有特别的优势。

1 ARM在语音通信中的应用

在音频通信编码中,信号处理操作主要有饱和操作和16位数乘法操作。其中饱和操作又包括8位饱和加、8位饱和减、16位饱和加、16位饱和减、32位饱和加、32位饱和减、饱和左移、饱和绝对值以及饱和32位到16位等操作;而16位的乘法操作又包括两个16位数相乘以及乘累加等操作。

一般的嵌入式内核是不能支持这些专门的数字信号的处理操作的,只能采用模拟的方法进行软件实现。这样就会增加CPU处理时间,使通话延时,不能达到实时语音通信的要求。为了实时通信,不得不增加一个专门的DSP,因而增加了额外的花费。

而ARM内核就支持这些音频信号的处理操作,特别是ARMv5E,ARMv6版本的内核对音频信号的处理能力更是可以与专业的DSP媲美。下面就以AMR(自适应多速率)语音编码来进行说明。

1.1 ARM支持进位加、进位减操作

AMR源程序中有模拟进位加、进位减的函数。比如进位加函数源程序如下:

```
Word32 L_add_c (Word32 L_var1, Word32 L_var2)
{
  Word32 L_var_out;
  Word32 L_test;
  Flag carry_int=0;
  L_var_out=L_var1+L_var2+Carry;
  L_test=L_var1+L_var2;
  if ((L_var1 > 0) && (L_var2 > 0) && (L_test < 0))
  {
    Overflow=1;
    carry_int=0;
  }
  else
  {
    if ((L_var1 < 0) && (L_var2 < 0))
    {
      if (L_test >= 0)
      {
        Overflow=1;
        carry_int=1;
      }
      else
```

到稿日期:2009-12-23 返修日期:2010-03-02 本文受香港城市大学应用研究项目基金CityU Applied R&D Funding (9681001)资助。
姜春林(1983-),男,博士生,主要研究方向为多媒体通信等,E-mail:linlin24515@gmail.com;贾维嘉(1957-),男,教授,博士生导师,主要研究方向为多媒体通信等;张历卓(1974-),男,博士生,主要研究方向为多媒体通信;谷 科(1980-),男,博士生,主要研究方向为无线网络。

```
        {
          Overflow=0;
          carry_int=1;
        }
      }
      else
      {
        if (((L_var1 ^ L_var2) < 0) && (L_test >= 0))
        {
          Overflow=0;
          carry_int=1;
        }
        else
        {
          Overflow=0;
          carry_int=0;
        }
      }
    }
    if (Carry)
    {
      if (L_test == MAX_32)
      {
        Overflow=1;
        Carry=carry_int;
      }
      else
      {
        if (L_test == (Word32) 0xFFFFFFFFL)
        {
          Carry=1;
        }
        else
        {
          Carry=carry_int;
        }
      }
    }
    else
    {
      Carry=carry_int;
    }
    return (L_var_out);
}
```

该函数至少需要 9 个时钟周期才能完成，效率很低。而 ARM 支持汇编指令 ADCS，只需要一个时钟周期就可实现进位加操作。把汇编指令直接嵌入 C 源文件中即可。程序修改如下：

```
Word32 L_add_c (Word32 L_var1,Word32 L_var2)
{
  Word32 L_var_out;
  asm("ADCS %0,%1,%2 ":"=r"(L_var_out) :"r"(L_var1),"r"
(L_var2) :"cc");
  return (L_var_out);
}
```

同样，进位减函数 L_sub_c()可以用 SBCS 指令替换。

1.2 ARMv5E、ARMv6 支持 DSP 操作

ARMv5E 提供的扩展指令支持一些 DSP 操作，比如 32 位饱和指令(QADD，QDADD，QSUB，QDSUB)和 16 位乘法指令(SMLAxy，SMLAWy，SMLALxy，SMULxy，SMULWy)等。这些指令只要一到两个时钟周期，用在音频编码算法中可以大幅度提高算法的执行效率。

ARMv6 兼容 ARMv5E，因此也提供了 ARMv5E 的扩展指令。在此基础上，ARMv6 又新增了多达 64 条新指令，其中仅增加的饱和指令就有 20 条，比如 8 位的饱和加(QADD8)、饱和减指令(QSUB8)、16 位的饱和加(QADD16)、饱和减(QSUB16)指令以及饱和 32 位到任意位指令(SSAT)等。这些指令使得 ARM 具有了更加强大的信号处理能力。

比如 AMR 编码算法中的饱和 32 位数到 16 位数的函数 saturate()，源程序如下：

```
Word16 saturate (Word32 L_var1)
{
  Word16 var_out;
  if (L_var1 > 0X00007fffL)
  {
    var_out=MAX_16;
  }
  else if (L_var1 < (Word32) 0xffff8000L)
  {
    var_out=MIN_16;
  }
  else
  {
    var_out=(Word16) L_var1;
  }
  return (var_out);
}
```

saturate()函数有 3 个条件语句，需要 2 到 4 个时钟周期。而 ARMv6 的饱和指令 SSAT 只需要一个时钟周期就可以完成该函数操作，效率提高了至少一倍。代码修改如下：

```
Word16 saturate (Word32 L_var1)
{
  Word16 var_out;
  asm("SSAT %0,#16,%1 ":"=r"(var_out) :"r"(L_var1) :"
cc");
  return (var_out);
}
```

再比如 16 位饱和加函数 add()，源代码如下：

```
Word16 add (Word16 var1,Word16 var2)
{
  Word16 var_out;
  Word32 L_sum;
  L_sum=(Word32) var1+var2;
  var_out=saturate (L_sum);
  return (var_out);
}
```

即使我们对于 saturate()函数进行了优化，使得该函数只需一个时钟周期，但 add()函数还是需要两个时钟周期，如果用 ARMv6 提供的 QADD16 指令，就只需要一个时钟周期，节省了一半的时间。改写后的代码如下：

```
Word16 add (Word16 var1,Word16 var2)
{
  Word16 var_out;
  asm("QADD16 %0,%1,%2 ":"=r"(var_out) :"r"(var1),"r"
(var2) :"cc");
  return (var_out);
}
```

(下转第 306 页)

结束语 本文使用 FPGA 求解和实现了一个经典的动力学问题——N 体问题，从而提供了一种新型的解决计算密集型任务的方法。实验表明，此方法具有可行性，求解速度较一般软件编程实现要快，而且硬件资源占用率也不是很大，开发过程中编写的一些程序具有可重复使用性。

随着高性能计算的进一步发展，不仅是 FPGA，还有 GPU，CELL 等都令人关注。将它们混合集成到一起，构成异构重构计算系统，开展高性能并行计算，是本课题组将来的研究工作。

参 考 文 献

[1] Lindholm T. N-body algorithms [C]// Seminar presentation. 1999

[2] 潘松，黄继业. EDA 技术与 VHDL[M]. 北京：清华大学出版社，2006

[3] 郑有泉. 现场可编程门阵列第一讲现场可编程门阵列 FPGA 概述[J]. 世界电子元器件，2005，9(1)：40-45

[4] 张博. 基于逐次逼近的 VHDL 开平方算法[J]. 微计算机信息，2008，24(5)：196-197

[5] 李旭. 基于 FPGA 的流水线技术应用研究[J]. 电子测量技术，2007，30(2)：131-132

[6] Hamada T，Benkrid K，Nitadori K，et al. A comparative study on ASIC，FPGAs，GPUs and general purpose processors in the O (N^2) gravitational N-body simulation[C]// NASA/ESA Conference on Adapative Hardware and System. 2009：447-452

[7] Lienhart G，Kugel A，Manner R. Using floating-point arithmetic on FPGAs to accelerate scientific N-body simulations[C]// Proceedings of the 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines. 2002：182-191

[8] Liu P F，Bhatt S N. Experiences with parallel N-body simulation [J]. IEEE Transactions on Parallel and Distributed Systems，2000，11(12)：1306-1323

[9] Ho H，Chun C W，Yu P，et al. Floating-point FPGA：architecture and modeling[J]. IEEE Transactions on Very Large Scale Integration Systems，2009，17(12)：1709-1718

[10] Valls J，Sansaloni T，Peiro M，et al. Fast FPGA-based pipelined digit-serial/parallel multipliers [C] // Circuits and Systems，ISCAS'99 Proceedings of the 1999 IEEE International Symposium. 1999：482-485

（上接第 301 页）

同样，饱和减函数 sub()也可以用 QSUB16 指令来替换。

2 ARM 在视频通信中的应用

2.1 充分利用寄存器进行数据操作

H. 263 是 ITU-T(国际电信联盟)发布的视频编码标准，专为中高质量运动图像压缩设计。它采用 DCT（离散余弦变换）和 IDCT(反向离散余弦变换)进行视频压缩，可以得到很不错的视频效果，因此常应用于视频会议和可视电话等通信应用中[2]。

ARM 内核有 16 个寄存器，它们可以作为通用寄存器，在视频编码中，编译器并不能有效、充分地使用这 16 个寄存器。比如 DCT 和 IDCT 函数有很多数组[3]，编译器一般直接编译成内存读取操作。DCT 和 IDCT 在运行过程中多次调用，造成频繁的访存操作，消耗较多的时间。如果能充分利用这 16 个寄存器，用汇编语言来精心编排，把经常用到的数组元素用寄存器来暂时存放，则可以有效提高程序的运行速度。这里有一个原则，就是必须确保最重要的和经常用到的数组元素被分配到寄存器。

2.2 把移位操作和其他操作用一条指令实现

ARM 有一个特征，那就是寄存器可以在进入 ALU(算术逻辑单元)之前先通过桶式移位器进行移位操作，即预处理。这些移位操作包括 LSL(逻辑左移)、ASL(算术左移)、LSR(逻辑右移)、ASR(算术右移)、ROR(循环右移)以及 RRX(带扩展的循环右移)操作，它们可以和数据处理指令(ADC，ADD，AND，BIC，CMN，CMP，EOR，MOV，MVN，ORR，RSB，SBC，SUB，TEQ，TST)一起使用，不占用一个独立的指令，从而提高了代码密度，减少了对存储器的访问次数。

比如在 IDCT 函数中有一条语句：

X0=(blk[0] << 11)+128

数组元素 blk[0]先左移 11 位，然后加上立即数 128，假设 R1 寄存器保存的是 X0，R2 寄存器保存的是 blk[0]，R3 寄存器保存的是 128，则汇编语句如下：

ADD R1，R3，R2，LSL #11

一条汇编指令就实现了两个操作，节省了取指令的时间。

2.3 实验分析

我们把 H. 263 的 DCT 和 IDCT 函数用汇编语言来实现，在汇编过程中采用上面的两种方法，把汇编实现后的函数部署到 HKC 智能手机中进行多次测试，并将结果与汇编前进行对比。HKC 的配置如下：

HKC G801：处理器内核是 ARMv5，主频是 416MHz，内存是 64MB 。

实验结果如表 1 所列。

表 1 汇编前后的效率对比

	DCT 运行 1000 次的时间	IDCT 运行 1000 次的时间	H. 263 的 CPU 使用率
汇编前	20. 5ms	18. 7ms	90%
汇编后	10ms	9. 3ms	52%
优化效率	51. 2%	50. 3%	42. 2%

由该表可知，采用本文的方法使 H. 263 的 CPU 使用率下降了 40%左右，效果还是很明显的。

结束语 ARM 内核在音视频信号处理方面有很大的优势，因此特别适用于音视频编码中，它的性能在某些方面甚至超过了专门的 DSP。

参 考 文 献

[1] Sloss A N，et al. ARM 嵌入式系统开发：软件设计与优化[M]. 北京：北京航空航天出版社，2005：528-536

[2] 余振建，周健，戴梅萼. 面向运动图像远程实时传输的 H. 263 压缩方法的分析与优化[J]. 电子技术应用，2003，29(1)：50-52

[3] 郑晓博，陶品. ARM7 平台实时 MPEG-4 解码系统的实现与优化[J]. 计算机科学，2008，35(1)：246-249