Optimisation techiques

using TI DSP TMS320C55x.

Authors:
Ajay Gupta
Prabindh Sundareson
Krishna Kumar G
Internet Audio Group
TI - India

Optimization techniques using TI DSP C55x

Abstract— *With increasing penetration of the Internet and availability of music in compressed formats, Personal Internet Audio (PIA) players have carved their own niche in the consumer electronics market. PIA players place a heavy premium on power consumption because lower power consumption gives the user the benefit of more battery life. In this paper, an advanced DSP architecture solution from TI is described, which enables the user to reap the benefits of a highly parallel system. The TMS320C55x family has been tailor-made for applications that require efficient algorithmic implementations, compact code storage, and low power consumption. A case study is presented in this paper, based on implementation of an audio decoder on the C55x DSP platform. Results indicate that due to the optimisation levels achieved, the power consumption can reduce by as much as 50% compared to other implementations. Further applications and improvements that can lead to higher optimisations are described. Other value adding features of the TI solution are OEM customisable security plug-ins, programmability for a number of different algorithms and fast turnout times.*

Introduction

Within the last year, the popularity of downloadable, compressed audio formats via the Internet has skyrocketed. The top requirement of the consumer has always been high quality of music and long continuous playtime. This paper discusses our efforts in using the power of TI's TMS320C55x DSP to offer an edge in high quality music and extremely low power consumption. The paper covers:

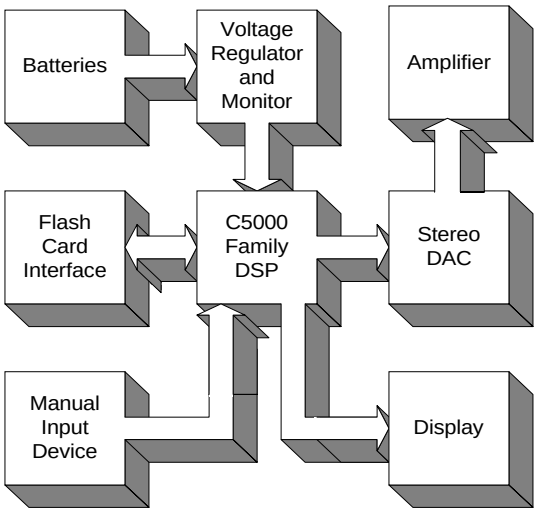
1. A description of the DSP platform for developing audio devices.
2. Power figures of a typical audio player.
3. Technical features of the C55x, which contribute to increased computing efficiency as well as power efficiency.
4. Detailed discussion about the features in the optimisation methods section.
5. Two case studies, which help us to understand how these features can be applied in a practical system.

Platform Description

The target processor for development was the TMS320C55x, TI's low-power, 16-bit fixed-point DSP.

For code development, the TI C5409 Internet Audio EVM, developed within the Internet Audio Group at TI, was used. This board is a single DSP design with a compact flash for holding music and a user interface for controlling playback. Support is provided for audio decompression, sample rate conversion, graphic equalization, and digital volume control. The DSP also handles user interface operations and compact flash I/O.

System Overview



A typical solid state audio system utilizes a Texas Instruments TMS320C5000 family digital signal processor (DSP) as its processing engine and a Compact Flash card for media storage (Ref fig above). In the system, the DSP responds to events

from human input devices and updates a display to provide visual feedback to the user. When the user plays an audio track, the DSP accesses the compressed audio samples on the flash card, decompresses the samples in its internal memory, verifies the digital watermark, and passes the uncompressed audio samples to a D/A. The analog signal is then fed through the amplifier to drive stereo headphones.

Power Consumption

This solid state audio player improves on current portable players by offering longer battery life, ruggedness, large data capacity, and small size.

Battery Life

To compute the typical battery life of the system, some assumptions about the flash operation must be made. An uncompressed 48kHz, 16-bit data stream is equivalent to a data rate of 1.526 Mbits/s. For CD-quality audio, an MPEG-2 AAC compressed data stream requires only 128kbits/s. The MPEG-2 AAC algorithm requires a new frame of less than 688 bytes every 43ms. Assuming the flash sleeps for 40ms, consuming 200µA, and is running for 3ms, consuming 45mA, the average current would be 3.3mA (or 11mW).

Table 1

Power Consumption for typical system.

SanDisk SDCFB-48-101 CompactFlash	11mW
Texas Instruments TMS320C5409 DSP	58mW
AKM AKM4350 DAC	8mW
Texas Instruments TPA152 Stereo Amplifier	32mW
Densitron LM4012-TN 16x1 LCD Display	5mW
Device I/O	1mW
Voltage Regulators (80% efficiency)	23mW
Total	138mW

Table 1 shows the power consumption for the individual components in the solid state audio system. For a 3V system, two "AA" batteries would be needed. For the 138mW system shown here, this would offer about 39 operating hours. For comparison, a typical CD player consumes 600mW and operates for approximately 9 hours on 2 "AA" batteries.

From the data we see that DSP is the most power hungry component in IA system. The power consumed by the DSP is proportional to the clock frequency supplied to the DSP. Thus the challenge to the software designer is to use very low MIPS for decoding. Low MIPS in decoder will aid:

1. Low power consumption.
2. Spare MIPS for OEM customization.
3. Spare MIPS for adding new features like equalizer.

Technical Features

The C55x DSP core's low power/high performance makes possible feature-rich, miniaturized personal and portable applications:

Key Features

- **Advanced power management and design**: Runs as low as 0.05 mW/MIPS @ 0.9V, with performance ranging from 140-800 MIPS.
- **Increased performance for tight power budgets:**
  The C55x™ DSP core will speed up development of exciting new miniaturized applications. For example, the C55x DSP core will make possible Cochlear implants so small they may be completely implanted in the ear, enabling people who are profoundly deaf to hear again.
- **New instructions:**
  These reduce code size, increase compiler efficiency, cut power usage, and increase parallelism.
  The processor also has advanced on-chip emulation capabilities for debugging.

Optimization methods

The following features of C55x provide the designer with a number of methods to optimize DSP algorithms:

1. **Multiple Buses**
   The C55x family has 3 Data Read buses, 2 Data Write buses and 1 Program bus. With more buses, and combined with instructions which execute in parallel, more data can be accessed.
   Consider the case where we have to do the following tasks in a single cycle:
   a. Transfer a double word from one

memory location to another

b. Increment both auxiliary register pointers, so that they point to the next data location.

This is a typical example of a function for memory copy (memcpy). This can be accomplished by using the following single C55x instruction:

MOV dbl(*AR6+), dbl(*AR1+); move a double word

This instruction uses four buses in the processor at the same time for effecting the transfer i.e., two simultaneous memory accesses are made for both reading and writing.

2. Variable length instruction set

The instructions can have varying lengths, ranging from one byte to six bytes. The program space in C55x is byte addressable. This effectively means that the processor allows the user to "pack" more instructions into the memory, than say, having a fixed length instruction set

This ultimately results in lesser code size.

3. Higher operating frequency

The C55x has operating frequency ranges upto 400 MHz. This results in fast execution of instructions.

4. Pipeline Protection Unit

The Pipeline Protection Unit (PPU) allows for faster execution of instructions in the user program. The pipeline conflicts are avoided, thus making debugging easier. This makes it easier to program the processor.

5. Separate access to data coefficients

A dedicated register is provided for accessing coefficients from data memory locations. This register is functionally equivalent to the auxiliary registers. This register, called the Coefficient Data Pointer (CDP) can be used in instructions, which simultaneously use two coefficients say, in a Dual MAC instruction.

The CDP can be efficiently used when the same coefficients are used in two separate output calculations. To be specific, operations like block-FIR filters can be implemented using this approach.

6. Direct compare instructions

Some tasks involve a lot of conditional processing using IF statements. In most cases, we would be doing the following steps to accomplish this:

a. Move the data memory value to a register
b. Subtract or do some other operation to see whether the condition is true or false
c. Restore the value to the value before comparison
d. Do the conditional task

It can be seen that if a number of conditions have to be tested, especially inside loops, the overhead can be quite large. In view of this, C55x provides a method of direct comparison of the values from memory with a specific register.

For example, the following instruction compares the value of AC3 and AC2, and sets TC1 if AC3 > AC2.

CMPU AC3 < AC2, TC1; check AC3 and AC1

7. Multiple Repeat options

Loops can be implemented in many ways in a processor. Decrement-check for zero and then branch conditionally is one approach, or a C54x compatible RPT instruction can be used. In C55x, recognising the need for fast looping instructions, a new instruction, RPTBLOCAL has been added. This instruction takes less number of cycles to execute than the standard RPT instruction, and it can be used where the number of words inside the loop fits within 28 words.

8. Parallel instructions

Usage of parallelism in the instructions can help increase the efficiency of the programs. Apart from the parallel instructions offered in the earlier versions of the C5xx family, the C55x also offers the possibility of making our own parallel instructions. Thus the instruction set becomes more flexible.

Case Studies

The following case studies reflect the methods adopted for optimising the MP3 decoder algorithms. The first case study indicates the optimisation done for implementing the MDCT algorithm on to the C55x. The second case study indicates the results achieved with the Synthesis Filterbank.

Case Study 1: MDCT Optimisation

The MDCT (Modified Discrete Cosine Transform) is used in the MP3 decoder for frequency-time domain transformation. The Fix-Point MDCT computes end sub-band – start sub-band DCT outputs using matrix multiply. This function is usually called once per granule (of 576 samples) per channel and computes all DCT outputs for each sub band[2]. In the case of a mixed-block (low 2 subbands only are LONG) this function will be called twice per granule per channel. Computing all the DCT outputs inside this function avoids function call overhead of calling a single DCT function 32 times.

The pseudo-code for the function is as below:

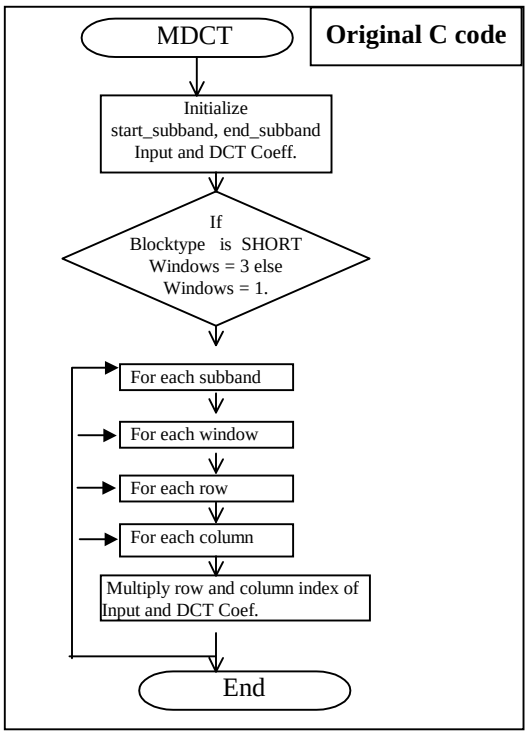
```
FixptMDCT(
FixPt *in,          /* Input vector*/
int blocktype,      /* Size: either 18 or 6*/
int start_subband, /* Beginning subband*/
int end_subband    /* Ending subband*/
) {

    if blocktype == SHORT_WINDOW
        windows = 3;
    else
        windows = 1;

    for (start sub-band to end sub-band)
        for (each window)
            FixPtMatrixMultiply (input, DCTcoeff);
}
```

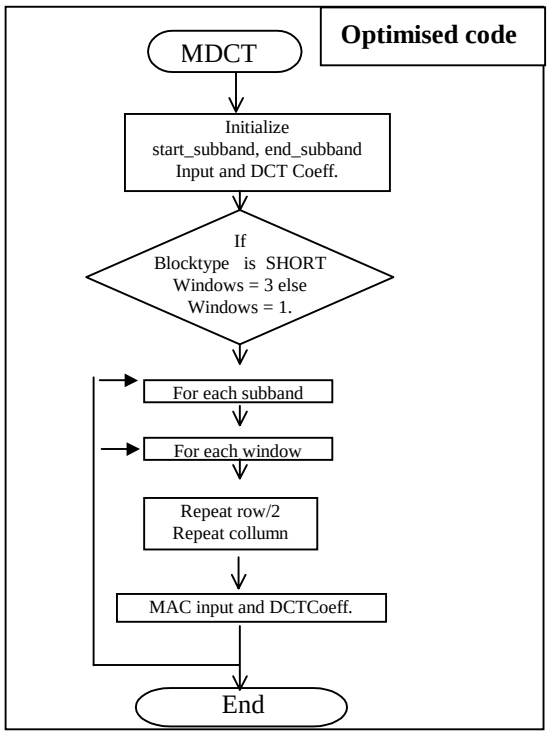
The main component of MDCT is a Fixed-Point matrix multiplication whose order depends on the size of the window, which can be either long (18 samples) or short (6 samples). This multiplication involves a product of two fractional numbers, represented in Q31 format. The conventional method of proceeding with this is to have two nested *for* loops.

Considering the fact that for each 32 bit fractional multiplication three 16-bit multiplication instructions are required, the total MIPS required for the matrix multiplication can be calculated as follows:

Window type	LONG
32b mults reqd	18*18
16b mults reqd	18*18*3
Total incldg overhead	1230
MIPS @ 44.1 KHz	6.02

`





We have implemented the above algorithm in C55x, and the flowchart of the optimised algorithm is given. The core loop in assembly is given in the appendix. (For the instruction set, refer to [3]). The code utilises the following important features of the C55x processor for achieving reduction in MIPS requirement:

a. Parallel instructions
b. Dual MACs
c. Nested loops
d. Concurrent usage of more temporary registers
e. Concurrent usage of more Accumulators
f. Less overhead in C-ASM calls
g. Faster local Repeat blocks
h. Logical optimisations

The comparison of the MIPS requirements for the optimized code and the unoptimized code is as below:

MDCT	W/o using Special instructions	C55x
CLOCKS	1230	580
MIPS	6.02	2.84

The results show an improvement of about 52% compared to an implementation, which does not use the C55x features.

Case Study 2: FDCT Optimisation

The FDCT (Fast Discrete Cosine Transform)[5] function is used in the Polyphase Synthesis Filter Bank.

The features of C55x as applied in the MDCT optimisation were used here, and the results are given below:

FDCT	W/o using Special instructions	C55x
CLOCKS	1380	810
MIPS	3.87	2.27

The results show an improvement of about 41% compared to an implementation, which does not use the C55x features.

Summary

This paper looks briefly at power considerations in a typical Portable Audio player, which has become popular in the recent past. The TI TMS320C55x processor fits into the requirements for a processing device to be used in such applications, where constraints on memory size, power, MIPS are present. In this paper, we have introduced the various features of the TMS320C55x processor. Application of the features of the processor in an MP3 decoder algorithm to achieve reduction in processing power has been discussed. Utilising these features can help implementers to save on MIPS and hence on power.

Appendix

1. Assembly routine for modified MDCT code:

```
_MDCT55X:
BSET FRCT
BCLR C54CM:: AMOV #3,T1
BSET SXMD
;C54CM is set to make it to c55x mode
;After each block output is got to go to next block we need to go 3+36
;This is used as the outer loop count  size/2-1 =8
; Should be changed depending on the value of size.
; Should be size/2-1
MOV T0, AC0                    ; AC0 = T0
ADD AC0, #-1,AC0   ; AC0 = T0/2
MOV AC0-1, BRC0;  for 6x6 change 8 to 2
MOV T0*2,T2
;This is the inner loop count. Size (18)-1=17
MOV T0-1, BRC1      ; for 6x6 change 17 to 5
;Allocate output memory pointer.After each inner loop
;Two long locations will be updated.
;Make ar2 and ar3 point to LSByte .That's why adding 1
AMOV #(_x18DCT4+T0*2+1), XAR3
;for change 37 to 13
AMOV #(_x18DCT4+1),XAR2
MOV #(32*2),T0
RPTBLOCAL Outer-1
;For each outer loop CDP increments from 0 to 544
;steps of 32.So loop will be performed 18 times.
MOV #0,AC0 :: XOR AC1
MOV #0,AC2 :: XOR AC3
MOV XAR0,XCDP
RPTBLOCAL MyInner-1
MAC *AR2-,*CDP+,AC0
::MAC *AR3-,*CDP+,AC1
MAC *AR2,*CDP-,AC0
::MAC *AR3,*CDP-,AC1
MAC *(AR2+T1),*(CDP+T0),AC2
::MAC *(AR3+T1),*(CDP+T0),AC3
MyInner:
ADD AC0,-16,AC2   :: ADD T2,AR2
ADD AC1,-16,AC3   :: ADD T2,AR3
MOV AC2, dbl(*AR1+)
MOV AC3, dbl(*AR1+)
MyOuter:
RET
```

References

1. "CPU Technical Brief - TMS320C55x Digital Signal Processor Core" - Texas Instruments technical brief #SPRT183
2. ISO MPEG Layer III standard CD 11172-3
3. "TMS320c55x Mnemonic Instruction set Reference guide" - Texas Instruments publication #SPRU374A
4. "Virtual Music Rocks" - A Forrester Research Report, March 1999
5. "A Fast computational algorithm for DCT" - Chen, Smith, Sralick, 1977, IEEE Publications