RT VIRTUAL LABS
(DIGITAL SIGNAL PROCESSING)
DSP BASICS AND DSK6713

Sagar Bhokre
Department of Electrical Engineering,
IIT Bombay

OUTLINE

- Why DSP?
- Need for the setup
- Features of the board used
- Number Representation
- Fixed point v/s Floating point
- Code Structure
- Sample HPF code

WHY DSP?

Microprocessors	Microcontrollers	DSP
1 operation per instruction	1 operation per instruction	Multiple operations per instruction
Multiple cycles for multiply	Multiple cycles for multiply	single cycle for multiply
Multiple instructions per cycle	Single instruction per cycle	Multiple instructions per cycle
Least peripheral interface	Rich in peripherals	Relatively less number of peripherals compared to Microcontrollers
Expensive	Low cost	Medium to low cost
Very good high-level software support	Rich in on-chip peripherals, good interrupt structure	On-chip bootloader; instruction cache
Fastest clocks	Comparatively slow clocks	Fast clocks
Intel Pentium	MSP430F168	TMS320VC33

NEED FOR THE SETUP

- Most of the times, the programmer is not aware of the faults/erroneous behavior of the program (not reflected by the simulator) when it is implemented on hardware. Our setup helps the user to implement and analyze the performance of the actual system.
  - E.g.: The simulator doesn't reflect a fault in the performance of a filter with very large order (say 2000) but actual implementation results in an erroneous behavior (computation cost and sampling rate being comparable). There are many other instances which the programmer may come across when the program is tested on the actual system.
- This setup makes the costly tools available to the users/students and facilitates them to learn and test the basics of programming the DSP without a need for purchasing the same.
- The setup allows the user to study the timing and speed constraints of the DSP core.

FEATURES OF THE BOARD USED

- TMS320C6713 DSP Starter Kit (DSK)
- I/O interfaces include four 3.5mm audio jacks for microphone, line in, speaker and line out, Digital interface pins, etc.
- Our setup utilizes:
  - The line-in jack for interface with the signal generator
  - Line out jack for interface with the Digital Storage Oscilloscope
  - The JTAG interface for burning the program onto the board
- For future implementation, the two channels of the line-in jack can be utilized independently. (say left channel could be used for interfacing the function generator and the right channel for interfacing the mono input audio signal)

NUMBER REPRESENTATION

32 bit Floating Point Number:

Exponent (e) [x_{31}-x_{24}]	Sign(s) [x_{23}]	Fraction (f) [x_{22}-x_0]

e : 8-bit exponent signed 2's complement integer
s : sign bit
f : fractional part of mantissa

32 bit Fixed Point Number:

Sign (*s*) [x_{31}]	Mantissa (m) [x_{30}-x_0]

s : sign bit
m : mantissa
A scaling factor is used along with the number. The value of the scaling factor decides the precision & range.

FIXED V/S FLOATING POINT

- Fixed point number operations
  - Fast usually single cycle execution (simple hardware)
  - Range is limited by precision (range precision tradeoff)
- Floating point number operations
  - High precision
  - High range
  - Complex hardware for operations

CODE STRUCTURE

```
#include "board support library header files"
#include "Standard header files specific to operations"
#define Macros

Define Global variables

Codec configuration settings

void main()
{
   DSK5510_AIC23_CodecHandle hCodec;
   Variables local to main
   DSK5510_init(); /* Initialize the board support library*/
   hCodec = DSK5510_AIC23_openCodec(0, &config); /*Start codec*/
   DSK5510_AIC23_setFreq(hCodec, DSK5510_AIC23_FREQ_8KHZ);
   Initialization of the variables
```

CODE STRUCTURE

```
while(1)
        {
        /* read samples from the left and right channel */
        while (!DSK5510_AIC23_read16(hCodec, &leftsample));
        while (!DSK5510_AIC23_read16(hCodec, &rightsample));
        USER CODE
        /* Send a sample to the left and right channel */
        while (!DSK5510_AIC23_write16(hCodec,leftsample ));
        while (!DSK5510_AIC23_write16(hCodec,rightsample ));
         }
   /* Close the codec */
   DSK5510_AIC23_closeCodec(hCodec);
}
```

SAMPLE HPF CODE

```
#include "tonecfg.h"
#include <mathf.h>
#include <math.h>
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define filter_length 51
Int32 input[filter_length+1];
short gain = 1;
float frequency=1000.0/8000.0; // normalized cutoff frequency, Fs = 8000 hz
DSK6713_AIC23_Config config = {/* Codec configuration settings */
   0x0017, // 0 DSK6713_AIC23_LEFTINVOL  Left line input channel volume
   0x0017, // 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume
   0x00d8, // 2 DSK6713_AIC23_LEFTHPVOL  Left channel headphone volume
   0x00d8, // 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume
   0x0011, // 4 DSK6713_AIC23_ANAPATH    Analog audio path control
   0x0000, // 5 DSK6713_AIC23_DIGPATH    Digital audio path control
   0x0000, // 6 DSK6713_AIC23_POWERDOWN  Power down control
   0x0043, // 7 DSK6713_AIC23_DIGIF      Digital audio interface format
   0x0001, // 8 DSK6713_AIC23_SAMPLERATE Sample rate control
   0x0001};  // 9 DSK6713_AIC23_DIGACT     Digital interface activation
```

SAMPLE HPF CODE

```
float hanning;
float filter_coeffs[filter_length];
float filter_coeffs_hanning[filter_length];
void calculate_coefficients_hanning(void)
{Int32 i;
Int32 temp=_trunc(filter_length/2);
/* Start making changes here*/
for(i=0;i<temp;i++)
filter_coeffs[i]=2*frequency*sinf((i-temp)*2*3.14159*frequency)/((i-temp)*2*3.14159*frequency);
filter_coeffs[temp]=2*frequency;
/* End */
for(i=0;i<temp;i++)
filter_coeffs[filter_length-i-1]=filter_coeffs[i];//  exploiting the symmetry

for(i=0;i<filter_length;i++)
            {    hanning=.54-.46*cosf(2*3.14159*i/filter_length);
                 filter_coeffs_hanning[i]=filter_coeffs[i]*hanning;
            }
}
```

SAMPLE HPF CODE

```
void main()
{
  DSK6713_AIC23_CodecHandle hCodec;
  int i,n;                                                    //Local Variables
  Uint32 value1,value;
  Int32 valueOut;
  float output;
  DSK6713_init(); /* Initialize the board support library, must be called first */
  hCodec = DSK6713_AIC23_openCodec(0, &config); /* Start the codec */
  DSK6713_AIC23_setFreq(hCodec, DSK6713_AIC23_FREQ_8KHZ);
        i=0;                                                  //Initialization
        for(n=0;n<filter_length;n++)
        input[n]=0;
        calculate_coefficients_hanning();
        while(1)
              {
                    while(!DSK6713_AIC23_read(hCodec, &value));
                    while(!DSK6713_AIC23_read(hCodec, &value1));
                    if(value > (1<<15))
                          input[i]=-65536+(Int32)value;
                    else
                          input[i]=value;
```

SAMPLE HPF CODE

```
                {
                    for(n=0;n<filter_length;n++)
                      output=output+input[abs(i-n+filter_length)%filter_length]*filter_coeffs_hanning[n];
                }
                output=(Uint16)valueOut;
                while (!DSK6713_AIC23_write(hCodec,gain*valueOut));
                while (!DSK6713_AIC23_write(hCodec,gain*valueOut));
                i++;
                i%=filter_length;
                output=0;
            }
    /* Close the codec */
    DSK6713_AIC23_closeCodec(hCodec);
}
```

Thank You

REFERENCES

- TMS320DSK6713 User Manual
- TMS320DSK6713 Example Codes