Experiment No:12 Date: 24-10-2024

KINEAR CONVOLUTION USING DSP KIT

Aim

To generate a sine wave using DSP kit.

Theory

Linear convolution is one of the fundamental operations used extensively in signal and system in electrical engineering. It has applications in areas like audio processing, signal filtering, imaging, communication systems and more.

In simple terms, linear convolution is the process of combining two signals or functions to produce a third signal or function. Formally, the linear convolution of two functions f(t) and g(t) is defined as:

The formula for linear convolution of two discrete signals x[n] and h[n] is given by:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k].h[n-k]$$

where:

• x[n] is the input signal.

• h[n] is the impulse response of the system.

• y[n] is the output signal.

In the context of linear convolution in DSP, this operation is applied to digital signals. DSP systems utilize algorithms to perform convolution efficiently, often leveraging Fast Convolution methods to handle large datasets and real-time processing.

Applications of Linear Convolution :

• Filtering: Used in digital filters to process signals.

• Image Processing: Applied for edge detection and blurring.

• System Analysis: Helps in analyzing LTI systems in response to inputs

Procedure

1. Open Code Composer Studio,
   Click on File - New – CCS Project
   Select the Target – C674X Floating point DSP , TMS320C6748 , and
   Connection – Texas Instruments XDS 100v2 USB Debug Probe and Verify.
   Give the project name and select Finish.

2. Type the code program for generating the sine wave and choose File – Save As and then save the program with a name including 'main.c'.
   Delete the already existing main.c program.
3. Select Debug and once finished, select the Run option.
4. In the Debug perspective, click Resume to run the code on DSP.
   Observe the console output to verify the convolution result.

Program

```
#include<math.h>

void main()

{

int *Xn,*Hn,*Output;

int *XnLength,*HnLength;

int i,k,n,l,m;

Xn=(int *)0x80010000; //input x(n)

Hn=(int *)0x80011000; //input h(n)

XnLength=(int *)0x80012000; //x(n) length

HnLength=(int *)0x80012004; //h(n) length

Output=(int *)0x80013000; // output address

l=*XnLength; // copy x(n) from memory address to variable l

m=*HnLength; // copy h(n) from memory address to variable m

for(i=0;i<(l+m-1);i++) // memory clear

{

Output[i]=0; // o/p array

Xn[l+i]=0; // i/p array

Hn[m+i]=0; // i/p array

}

for(n=0;n<(l+m-1);n++)

{

for(k=0;k<=n;k++)

{

Output[n] =Output[n] + (Xn[k]*Hn[n-k]); // convolution operation.
```

```
}
}
}
```

Result

Performed linear convolution of two sequences using DSP kit.

Observation

Xn

0x80010000 - 1

0x80010004 - 2

0x80010008 - 3

Hn

0x80011000 - 1

0x80011004 - 2

XnLength

0x80012000 - 3

HnLength

0x80012004 - 2

Output

0x80013000 - 1

0x80013004 - 4

0x80013008 - 7

0x8001300C - 6