EXPERIMENT DSP5: SIGNAL PROCESSING

Related course: KIE3007 (Digital Signal Processing)

OBJECTIVES:
To execute the program of digital signal processing

EQUIPMENT:
PC equipped with Labview Software

INSTRUCTIONS:
For every test, students are required to print screen the final block diagram and final front panel, and save them in a folder.

REFERENCE(S):
Refer to the main references of KIE3007

EXERCISE:
Exercise 1: Simulating and Analyzing a Signal
Exercise 2: Integrating Simple Filters
Exercise 3: Acquire and Analyze Sound in Real Time
Exercise 4: Generate Audio Output Using LabVIEW's "Play Waveform" Express VI
Exercise 5: Generate and Acquire/Analyze Sound in Real-time

INTRODUCTION:

Digital signal processing (DSP) is concerned with the representation of discrete time signals by a sequence of numbers or symbols and the processing of these signals. Digital signal processing and analog signal processing are subfields of signal processing. DSP includes subfields such as audio and speech signal processing, sonar and radar signal processing, sensor array processing, spectral estimation, statistical signal processing, digital image processing, signal processing for communications, control of systems, biomedical signal processing, seismic data processing, etc.

The goal of DSP is usually to measure, filter and/or compress continuous real-world analog signals. The first step is usually to convert the signal from an analog to a digital form, by sampling it using an analog-to-digital converter (ADC), which turns the analog signal into a stream of numbers. However, the required output signal is another analog output signal, which requires a digital-to-analog converter (DAC). Even if this process is more complex than analog processing and has a discrete value range, the application of computational power to DSP allows for many advantages over analog processing in many applications, such as error detection and correction in transmission as well as data compression.

DSP algorithms have long been run on standard computers, on specialized processors called digital signal processor on purpose-built hardware such as application-specific integrated circuit (ASICs). Today there are additional technologies used for DSP including more powerful general purpose microprocessors, field-programmable gate arrays (FPGAs), digital signal controllers (mostly for industrial apps such as motor control), and stream processors, among others.

NI LabVIEW DSP is a block diagram-based DSP development platform that allows the user to quickly set up complex DSP algorithms. The true power of LabVIEW lies in its ability to interface with external DSP devices and/or internal sound cards that are installed on the PC. The developed algorithms are downloaded to the DSP board, which then runs the algorithm in a real-time environment.

Exercise 1: Simulating and Analyzing a Signal

Complete the following steps to create a VI that simulates a sine wave and displays its time and frequency domains in graphs.

1. Launch LabVIEW 2011.
2. On the **Getting Started** Window, go to **File>>New VI**.

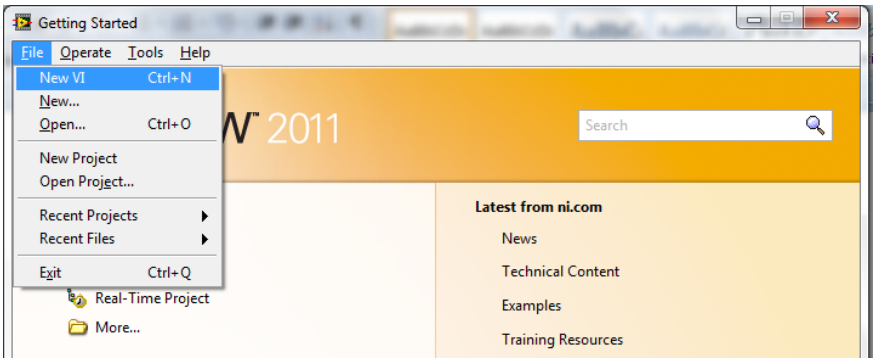

Figure 1.1: Getting Started Window

3. **Blank VI** will occur. There will be two windows**; Front Panel** and **Block Diagram**. **Front Panel** is a windows for Graphical User Interface(GUI) where you can place all controls and indicators. **Block Diagram** is a window for programming development. The next few instructions will you on developing the **Front Panel**.

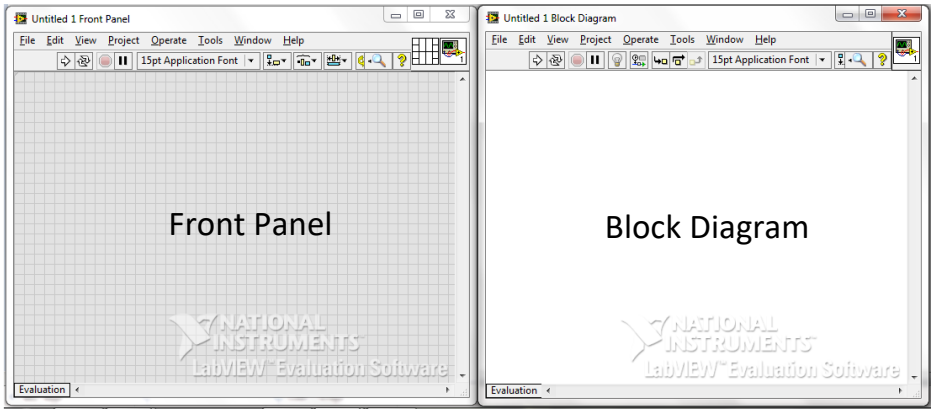

Figure 1.2: Front Panel and Block Diagram Window

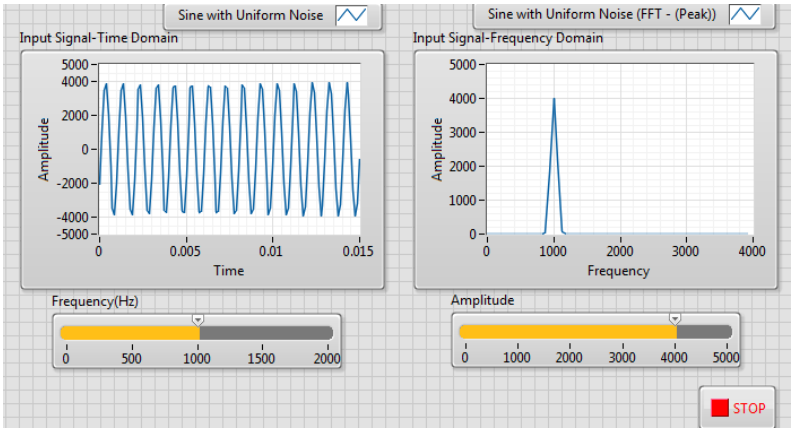

Figure 1.3: Front panel of Exercise 1

4. Place a waveform graph, located on the **Controls»Graphs** palette and change the label from default Waveform Graph to Input Signal - Time Domain (by double clicking on the label). We will simulate a sine wave and display it in this graph. The timing characteristics of the sine wave will be the following: samples per second: 8000, number of samples: 128.
5. From the **Properties** shortcut menu, under **Scales** tab disable the Autoscale options for X and Y Axis. Change the x-axis minimum value to 0 and maximum value to 128. Change the y-axis minimum value to be -5000 and maximum value to 5000.
6. Place another waveform graph to display the frequency domain of the input signal and label it Input Signal - Frequency Domain.
7. From the **Properties** shortcut menu, under the **Scales** tab disable the Autoscale options for the both axis. For x-axis change the name to Frequency (Hz), and set the minimum value to 0 and maximum value to 4000. Since we will sample at 8000 Hz with 128 number of samples, set the Multiplier to 62.5 (8000/128). For y-axis set the minimum value to 0 and maximum value to 5000.

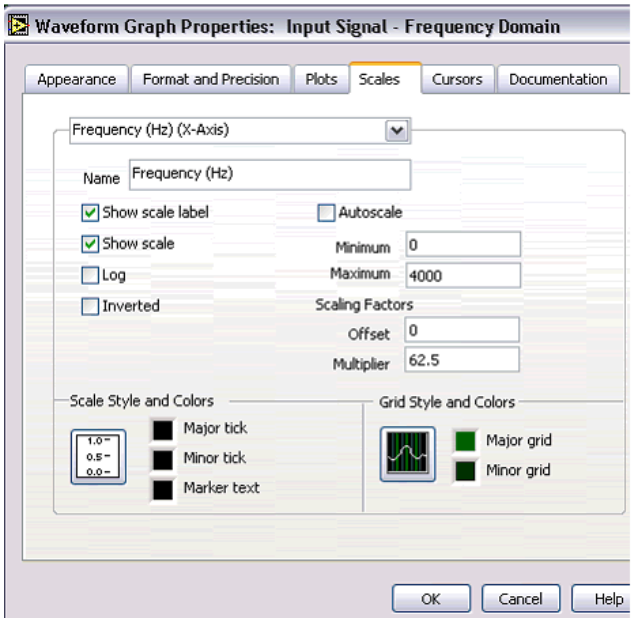

Figure 1.4: Input Signal - Frequency Domain Graph Properties

8. Place two Horizontal Pointer Slides from Control>>Numeric palette and label them Frequency(Hz) and Amplitude. Set the maximum scale range of Frequency(Hz) slider to 2000 and the maximum scale range of Amplitude to 5000.
9. The next few instructions will help you on building the following block diagram.

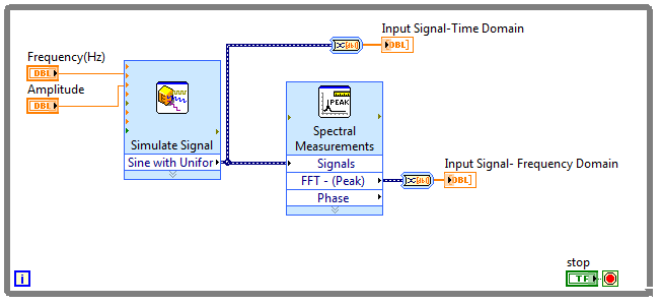

Figure 1.5: Block diagram of Exercise 1

10. Place the **Simulate Signal** Express VI located on the **Functions>>Express>>Input**. Double click on the Express VI and configure it to have a **Signal Type** of **Sine**. Check the **Add noise** box and choose **Uniform White Noise** for the **Noise type**. Wire the **Amplitude andFrequency (Hz)** terminals to corresponding inputs of the **Simulate Signal** Express VI. Wire the **Sine with Uniform** output of the VI to **Input Signal – Time Domain** graph terminal. Refer to Figure 1.6 for other configurations.

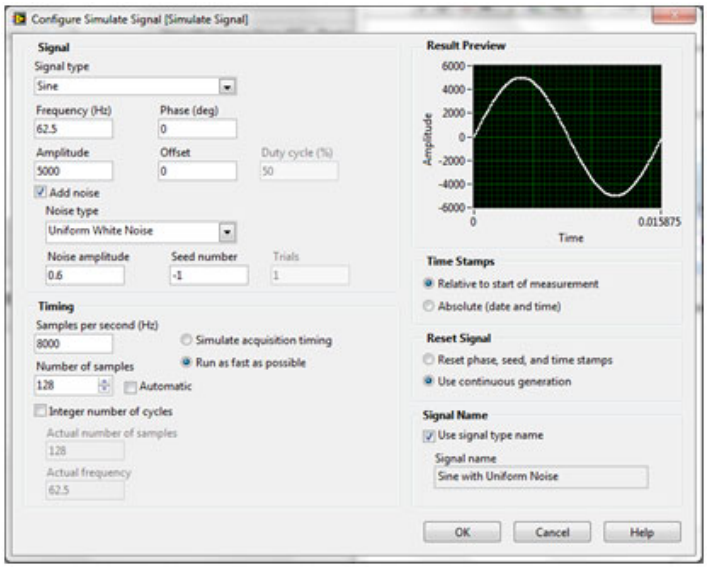

Figure 1.6: Configuration Window for Simulate Signal Express VI

11. Place the **Spectral Measurements** Express VI from **Functions»Express>>Signal Analysis**. Double click on the Express VI to configure it. Select the **Magnitude(peak)** option under **Spectral Measurement** and click OK. Wire the **Sine with Uniform** output of **Simulate Signal** VI to **Signals** input of **Spectral Measurements**.
12. Wire the **FFT – (Peak)** output of the Express VI to **Input Signal – Frequency Domain**graph terminal. To convert the Dynamic Data to Array Data, place a **Convert From Dynamic Data**Palette from **Functions>>Express>Signal Manipulation**
13. Place a **While Loop** located under **Functions»Structures** palette, on the block diagram.
14. Save the VI as Exercise1.vi.
15. Adjust the **Frequency** and **Amplitude** values and run the VI.
16. Run the VI again with various values of Frequency and Amplitude.
17. Save and close Exercise1.vi. You will use it again in the next example.

Exercise 2: Integrating Simple Filters

1. Open Exercise1.vi that you built in previous exercise. Select **File»Save As** and rename the VI to Exercise2.vi.
2. Modify the **Front Panel** as shown in the following figure. The next few instructions will help you on modifying the **Front Panel.**

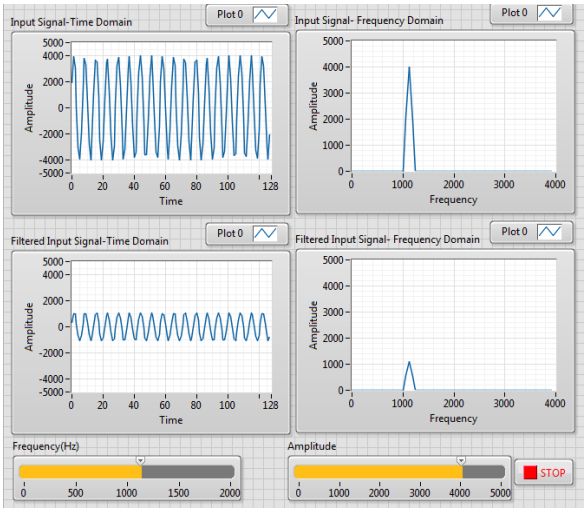

Figure 2.1: Front Panel of Exercise 2

a. Create a copy of **Input Signal – Time Domain** waveform graph (by holding down Ctrl and dragging the graph). Review the Properties of the new graph and note that it has the same x and y scales as **Input Signal – Time Domain** graph. Label the new graph **Filtered Signal – Time Domain**.

b. Create a copy of **Input Signal – Frequency Domain** waveform graph (by holding down Ctrl and dragging the graph) and label the new graph **Filtered Signal – Frequency Domain**.

3. Modify the block diagram as follows. The next few instructions will help you on modifying the **Block Diagram.**

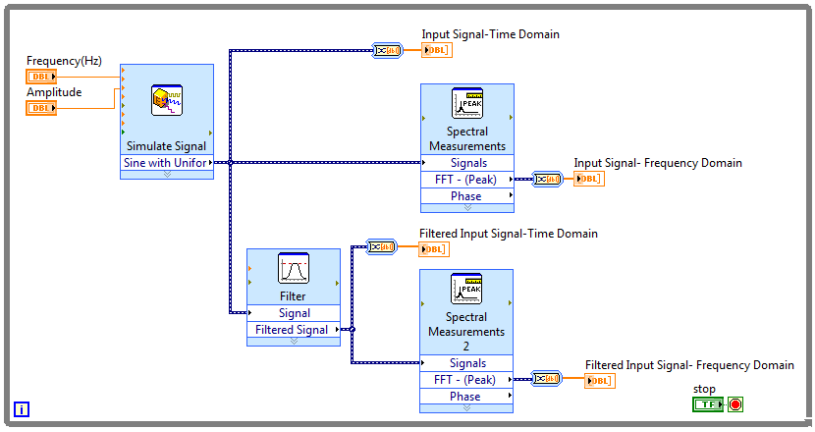

Figure 2.2: Block Diagram of Exercise 2

4. Place a **Filter** Express VI from **Functions»SignalProcessing»Filters.** This Express VI will apply a Lowpass filter to the signal. Double click on the VI and configure it by entering 1000 for **Cutoff Frequency**, and selecting **Infinite impulse response (IIR) filter** with **Chebyshev** topology of order 5. Display the filtered signal by wiring the **Filtered Signal** output of Express VI to **Filtered Signal–Time Domain** terminal.

5. Create another copy of the **Spectral Measurements** Express VI by dragging it while holding the Ctrl key. Wire the **Filtered Signal** output of the **Filter** VI to the **Signals** input of **Spectral Measurements**. Then connect the **FFT - (Peak)** output to **Filtered Signal – Frequency Domain** terminal to display the frequency component of the filtered signal.

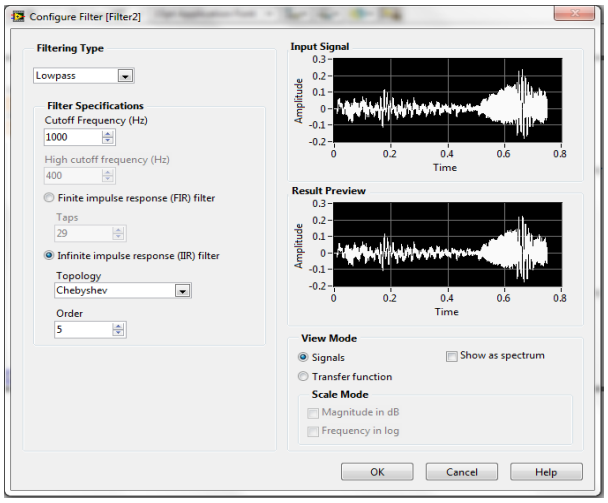

Figure 2.3: Configuration Window for Filter Express VI

6. Run VI. Note that the signal is attenuated as you increase the frequency above 1kHz.
7. Save and close Exercise2.vi

Exercise 3: Acquire and Analyze Sound in Real Time

1. Open Exercise2.vi that you built in previous example and save it as Exercise3.vi.
2. Add a **Horizontal Pointer Slide** with a range of 1000 and 2500Hz and label it as **Cut-off Frequency.**

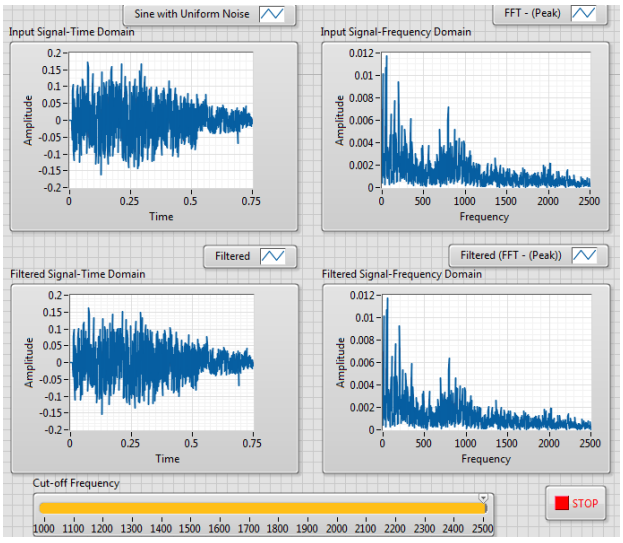

Figure 3.1: Front Panel of Exercise 3

3. Go to the block diagram and replace the **Simulate Signal** VI with **Acquire Sound VI** located on **Functions»Programming>>Graphics&Sound>>Sound>>Input** palette. Remove the **Amplitude** and **Frequency** sliders and the broken wires.
4. Wire the **Cut-off Frequency Slider** to the **Lower Cut-off node** of the Filter VI.
5. Then wire the output of the **Play Waveform VI** to the **Input Signal** of the Filer VI.
6. Make sure all the data are Dynamic Data (Do not use **Convert From Dynamic Data** Pallete).

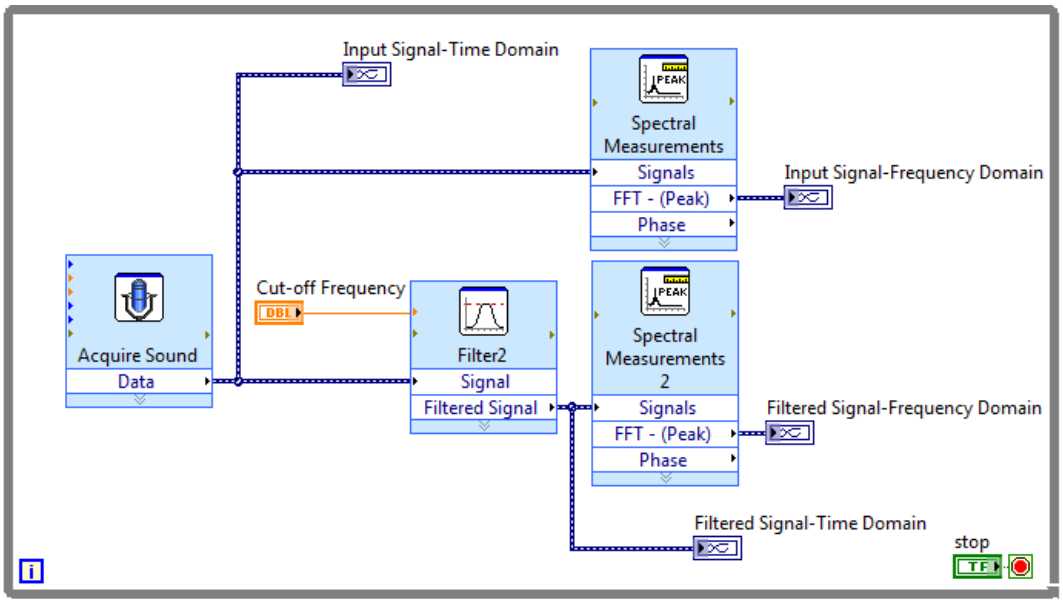

Figure 3.2: Block Diagram of Exercise 3

7. Run the VI. Whistle to microphone and see what the frequency of the sound is. If you cannot see the data, enable the **Auto-scale.** You can do it by moving your cursor to the graph axis, right click and check **Auto-Scale**.
8. Play around with the **Cut-off Frequency** and see the results of the filtered signal.
9. Repeat the Exercise with various filter modes (High Pass, Bandpass, Butterworth, Elliptic, etc).
10. Save and close Exercise3.vi.

Exercise 4: Generate Audio Output Using LabVIEW's "Play Waveform" Express VI

1. Open a new VI.
2. On the **Front Panel,** place a **Waveform Graph** and three **Numeric Controls**from **Control>>Modern>>Numeric**. Name each control as Duration(s), f(Hz) and fs(Hz).

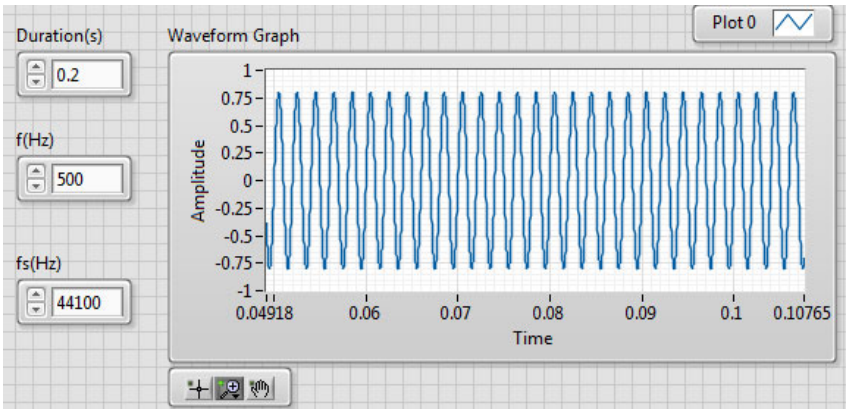

Figure 4.1: Front Panel of Exercise 4

3. Right click on **Waveform Graph** and go to **Visible Items** and enable **Graph Palette.**
4. On block diagram, add a **Play Waveform Express VI** from **Functions»Programming>> Graphics&Sound>>Sound>>Output.**

5. The sound can be generated from a simple sine-based signal. Thus add **Signal Processing>> Signal Generation>>Sine Wave.vi** into the block diagram. Build the block diagram as illustrated in Figure 4.2.

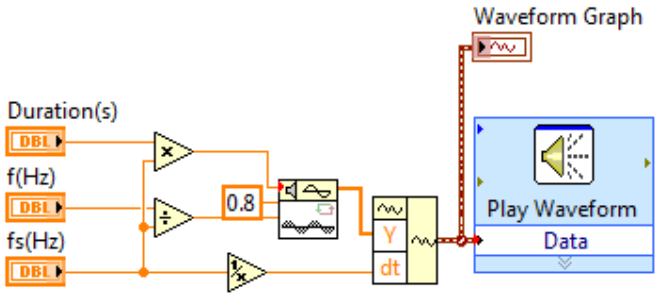

Figure 4.2: Block Diagram of Exercise 4

6. By multiplying the duration with sampling frequency, we will get the **Number of Samples** for **Sine Wave.vi.**
7. **Amplitude** will be a constant number between 0 and 1. Assign 0.8 as the amplitude.
8. **Frequency input** of the **Sine Wave.vi** is the normalized frequency. Desired frequency divided by sampling frequency will result the normalized frequency.
9. Set the **Duration** to 0.2, **Desired Frequency** to 500 and **Sampling Frequency** to 44100(typical CD sampling frequency).
10. The output of the **Sine Wave.VI** is in array while the input of **the Play Waveform.vi** is in waveform. Thus place a **Build Waveform Palette** from **Programming>>Waveform** to convert the array data into waveform.
11. Wire the **Sine Wave data** with **Y**, while the **dt** will be sampling interval(1/sampling frequency).
12. Wire the output of **Build Waveform Palette** with **Waveform Graph** and **Play Waveform.VI**.
13. Run the VI with various values of **Desired Frequency.**
14. Save and close Exercise4.vi.

Exercise 5: Generate and Acquire/Analyze Sound in Real-time

1. Open both Exercise3.vi and Exercise4.vi. Place **Front Panel** of both VI side by side.
2. Run Exercise3.vi continuously.
3. Run Exercise4.vi simultaneously and play around with desired frequency.
4. Exercise4.vi will generate the sound of assigned frequency and Exercise3.vi will capture the sound and will calculate the frequency of the generated sound. Both frequency of generated and acquired sound should be the same.

QUESTIONS: - PLEASE CHANGE TO OPEN-ENDED QUESTION

1. What are the effects/differences between the filtered signals when you use different filter topology (Butterworth, Chebyshev, Elliptic, etc)? Answer with at least three types of topology.

END OF EXPERIMENT