MÜHENDİSLİK FAKÜLTESİ
FACULTY OF ENGINEERING
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ
DEPARTMENT OF ELECTRICAL AND
ELECTRONICS ENGINEERING

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

DUMLUPINAR BULVARI 06800
ÇANKAYA ANKARA/TURKEY
T: +90 312 210 23 02
F: +90 312 210 23 04
ee@metu.edu.tr
www.eee.metu.edu.tr

EXPERIMENT 4. DECIMATION, INTERPOLATION AND PHASE-LOCKED LOOP

I. Introduction

This experiment is built upon the programming environment used in Experiment 3. Hence both MyRIO CPU and FPGA are used to generate signals, sample them and transfer them through DMA operation. However the basic signal processing tasks considered are decimation, interpolation and phase-locked loop design. In this respect, a rate conversion system is implemented to increase and decrease the frequency of a sinusoid generated digitally. In fact, there are two sinusoids that are considered in this experiment. The first one is a sinusoid which is obtained after A/D sampling. While it is assumed that the frequency of this sinusoid is known, it may not be precise and the aim is to generate an internally produced sinusoid such that it has exactly the same phase and frequency. Hence the frequency and phase of this digital sinusoid is adjusted such that it locks onto the sampled sinusoid.
The VI that you need to program uses the same functions of the Experiment 3. Hence it is easier to modify the VI's of Experiment 3 to obtain the desired functionality.

II. Preliminary Work

1) Read the following information on decimation, interpolation and digital phase-locked loop.

A. Sampling Rate Change

Sampling rate change is an operation to increase or decrease the sampling rate of a signal to match with the sampling rate of another signal. Sampling rate change is a frequently used since signal processing produces meaningful results only when two signals are at the same sampling rate. Sampling rate change can be easily done in digital domain by using decimation and interpolation. The rate change can be integer or a rational number but it cannot be a real number in general.

A.1. Decimation

Decimation reduces the sampling rate of the input signal by a factor of M where M is an integer. Hence decimation discards samples uniformly resulting compression in time. This

generates expansion in frequency which in turn may cause “aliasing”. Aliasing is a loss of information and is irreversible.

Decimation has two components, namely the decimation filter and the compressor. The function of the decimation filter is to bandlimit the input signal before the compressor such that there is no aliasing. Fig. 1 shows the block diagram of decimation.

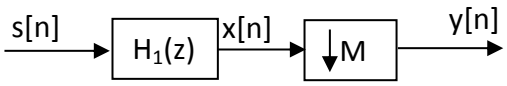

Fig. 1. Decimator composed of a filter and a compressor.

The relation between x[n] and s[n] is through time-domain convolution which is a multiplication in frequency given as below,

$$X\left(e^{j\omega}\right)=S\left(e^{j\omega}\right)H_1\left(e^{j\omega}\right)$$

$H_1(z)$ is a lowpass filter whose gain is 1 and cutoff frequency is $\frac{\pi}{M}$. The relation between y[n] and x[n] can be written as,

$$y[n]=x[Mn],\ n \text{ integer}$$

The above equation in frequency domain is,

$$Y\left(e^{j\omega}\right)=\frac{1}{M}\sum_{k=0}^{M-1}X\left(e^{j\frac{(w+2\pi k)}{M}}\right)$$

A.2. Interpolation

Interpolation increases the sampling rate by a factor of L and obtains the samples between data points. This is effectively expansion in time and compression in frequency. Hence an image spectrum is observed after compression which should be removed through a filter. Therefore interpolation is composed of an expander followed by a filter. The block diagram of interpolation is given in Fig. 2.

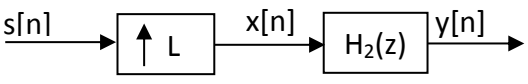

Fig. 2. Interpolation by a factor of L.

The relation between s[n] and x[n] can be given as below,

$$x[n]=\begin{cases} s\left[\frac{n}{L}\right], & n=0,\pm L,\pm 2L,\ldots \\ 0, & otherwise \end{cases}$$

Above expression in frequency domain can be expressed as,

$$X\left(e^{j\omega}\right)= S\left(e^{j\omega L}\right)$$

The relation between x[n] and y[n] can be established easily in frequency as,

$$Y\left(e^{j\omega}\right)= X\left(e^{j\omega}\right)H_2(e^{j\omega})$$

$H_2(z)$ is a lowpass filter whose gain is L and cutoff frequency is $\frac{\pi}{L}$.

A.3. Sampling Rate Change by a Rational Number

Usually, sampling rate change by a factor of a rational number is desired. In this case, decimation and interpolation in Fig. 1 and Fig. 2 should be combined. Since cascaded two filters is equivalent to a single filter, the block diagram for this system is given as in Fig.3.

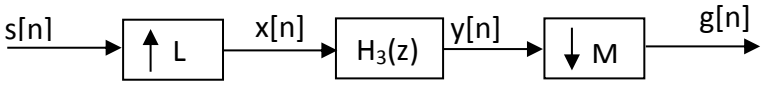

Fig. 3. Sampling rate change by a rational factor.

In the above system, $H_3(z)$ is a lowpass filter whose gain is L, cutoff frequency, ω_c, is the minimum of the both filters, i.e.,

$$\omega_c = \min\{\omega_{c1}, \omega_{c2}\}.$$

B. Digital Phase-locked Loop, DPLL

Phase-locked loops (PLLs) are widely used in a variety of fields including signal processing, communications, multimedia, etc. PLL is usually used for carrier phase and frequency tracking as well as clock recovery. While analog PLLs are still used, digital PLLs become more popular due their simplicity, robustness and cost effectiveness. PLL structure can be analyzed using both a nonlinear structures and linear approximations. In our case, linear model is preferred due to its simplicity.

PLL structure is composed of a phase detector, a loop filter and a voltage-controlled oscillator (VCO). Figure 4 shows a simple PLL structure in nonlinear and linear forms.

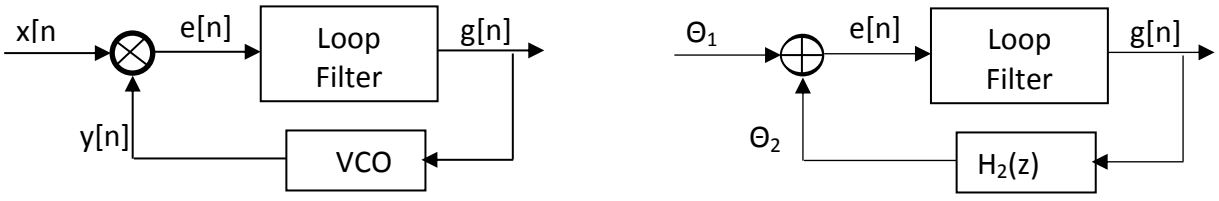

Fig. 4. A simple PLL structure a. Nonlinear structure, b. Linear structure.

In Fig. 4 a, the phase detector is implemented as a multiplier and lowpass filter. e[n] is the error signal representing the phase difference. The above structure multiplies the input signal x[n] and the generated sinusoid y[n] to obtain the phase difference which is then filtered by the loop filter to obtain a quantity which determines whether the phase/frequency of the VCO is increased or decreased.

Let x[n] and y[n] be given as follow, then e[n] can be written as,

$$x[n] = \sin(2\pi f_0 t + \theta)$$
$$y[n] = \cos(2\pi f_0 t)$$
$$e[n] = \frac{1}{2}(\sin(2\pi 2 f_0 t + \theta) + \sin(\theta))$$

Once e[n] is lowpass filtered to remove the twice the frequency component, g[n]=sin(θ)≈θ. Hence phase of the VCO is fed by g[n]. Note that if there is a small frequency offset between x[n] and y[n], it can also be tracked by the PLL as long as the error is within the loop bandwidth.

The loop filter can be an IIR filter of the following form, i.e.,

$$H(z) = \frac{az - 1}{z - 1}$$

where a is the loop gain. VCO can be represented as a digital controlled oscillator, DCO, and it can be written as,

$$H_2(z) = \frac{c}{z - 1}$$

Note that both of the IIR filters have a pole at z=1 and are unstable in strict sense. Nevertheless, it is assumed that the system is operated at a frequency other than DC and used in a narrow band. While the above information is given for completeness, implementation of the DPLL in this experiment can be performed in a different way. For example, VCO in our case is replaced by a sinusoidal waveform generator as a LabVIEW module which has both frequency and phase inputs.

The input signal, x[n] can have a time varying frequency and phase, i.e.

$$x[n] = A\cos(\omega_c(n)n + \theta(n))$$

The VCO output, y[n], should match with both frequency and phase drifts, i.e.,

$$y[n] = B\sin(\omega_d(n)n + \phi(n))$$

PLL has a lock range out of which the system can no longer follow the input signal frequency. Usually this lock range is small and it is determined by the loop filter. In our PLL

implementation, we use a first order PLL structure where the loop filter has zero order. In other words, loop filter in our case is FIR moving average filter whose coefficients are all 1/M where M is the length of the filter. This filter is equivalent to mean operator in LabVIEW or MATLAB.

2) Read the section **"Changing Sampling Rate Using Discrete-Time Processing"** in ***Discrete Time Signal Processing by Oppenheim, Schafer***.

3) In this question, you will observe the **change in sampling rate in MATLAB** by listening to the sound of a chirp signal with different sampling rates. **Don't forget to attach all the codes in MATLAB to your preliminary work**.

a) Generate a **chirp signal in 0-4 s** range with **sampling rate 10000 samples/s**, **start frequency 800 Hz** and **end frequency 1600 Hz**. Use ***"sound"*** command in MATLAB to **listen to** this chirp signal with **sampling frequency 10000 Hz (it is the sampling frequency used by "sound" command for sending the signal out to the speaker)**. Please be aware that the name of ***"sound"*** command may differ in different versions of MATLAB. **Write down** the **duration** of the sound you heard.

b) Now, use ***"sound"*** command to **listen to** the chirp signal you generated in **part a)** with **sampling frequency 5000 Hz (it is the sampling frequency used by "sound" command for sending the signal out to the speaker)**. Notice the **pitch** of the chirp sound has been **lowered** compared to part a). **Explain the reason** for this. Furthermore, **write down** the **duration** of the sound you heard**. Is there any change in comparison to part a)? Explain the reason for your answer.**

c) Use ***"sound"*** command to **listen to** the chirp signal you generated in **part a)** with **sampling frequency 20000 Hz (it is the sampling frequency used by "sound" command for sending the signal out to the speaker)**. Notice the **pitch** of the chirp sound has been **increased** compared to part a). **Explain the reason** for this. Furthermore, **write down** the **duration** of the sound you heard. **Is there any change in comparison to part a)? Explain the reason for your answer.**

d) Now, we will explore **sampling rate change on the chirp signal** generated in part a) using ***"resample"*** command. Use this function to **increase the sampling rate** of the chirp sequence in part a) **by 2**. Call this new sequence **"y"**. Use ***"sound"*** function to **listen to y** with **sampling frequency 10000 Hz (it is the sampling frequency used by "sound" command for sending the signal out to the speaker)**. Which sound in **part a), b), and c),** is this sound **similar to**? **Explain the reason**. Also, **write down** the **duration** of the signal and **comment on it**.

e) Use ***"sound"*** function to **listen to y** with **sampling frequency 20000 Hz (it is the sampling frequency used by "sound" command for sending the signal out to the speaker)**. Which sound in **part a), b), and c)**, is this sound **similar to**? **Explain the reason**. Also, **write down** the **duration** of the signal and **comment on it**.

f) Now, use **"resample"** function to **decrease the sampling rate** of the chirp sequence in part a) **by 2**. Call this new sequence **"z"**. Use ***"sound"*** function to **listen to z** with **sampling frequency 5000 Hz (it is the sampling frequency used by "sound" command for sending the signal out to the speaker)**. Which sound in **part a), b), and c)**, is this sound **similar to**? **Explain the reason**. Also, **write down** the **duration** of the signal and **comment on it**.

4) In this question, you will explore **decimation** and **upsampling** functions in **LabVIEW**.

a) Generate a constant array consisting of 7 digits of your student I.D. number.

b) Construct the following block diagram. Note that in Fig. 5, the constant array consists of the digits of 1234567. This is the example I.D. number. In your own code, you should replace this array with the one consisting of your I.D. number digits.

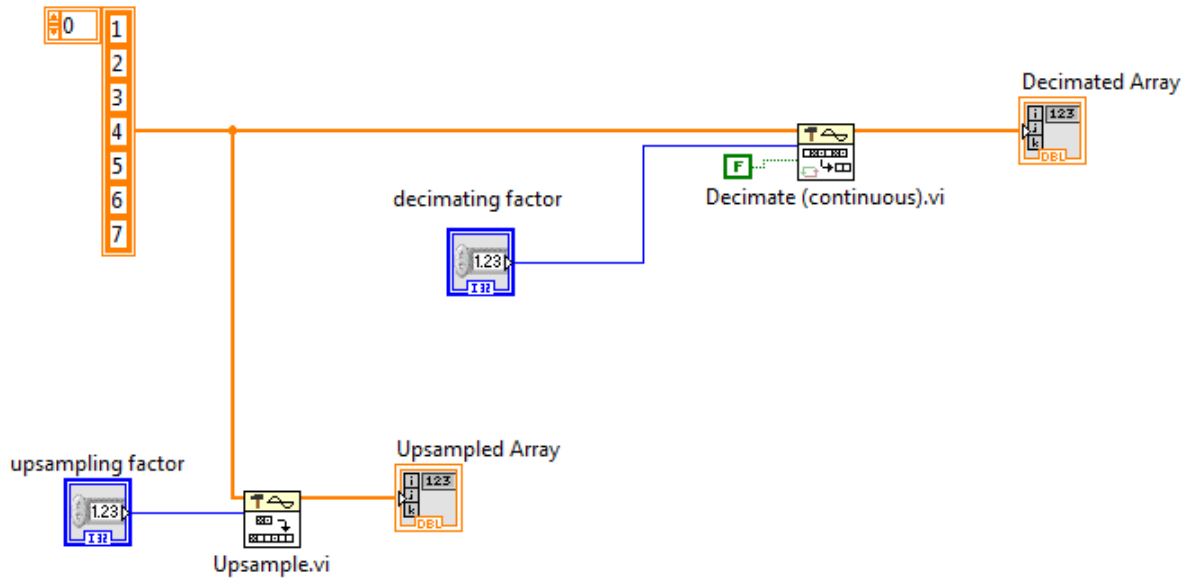

Fig. 5. Decimating and upsampling.

c) Try different factors for decimating and upsampling and observe the decimated and upsampled arrays. **Attach the Front Panel screenshot** for **decimating and upsampling factors as 3**. **All the elements of output arrays should be visible.**

d) Which blocks in Fig. 1 and 2 do the functions “Decimate (continuous).vi” and “Upsample.vi” correspond to?

e) Now, change the block diagram in Fig. 5 as shown in Fig. 6 by adding lowpass filters. Again, your constant input array should be the digits of your I.D. number.

f) Try different factors for decimating and upsampling and observe the decimated and interpolated arrays. **Attach the Front Panel screenshot** for **decimating and upsampling factors as 3**. **All the elements of output arrays should be visible.**

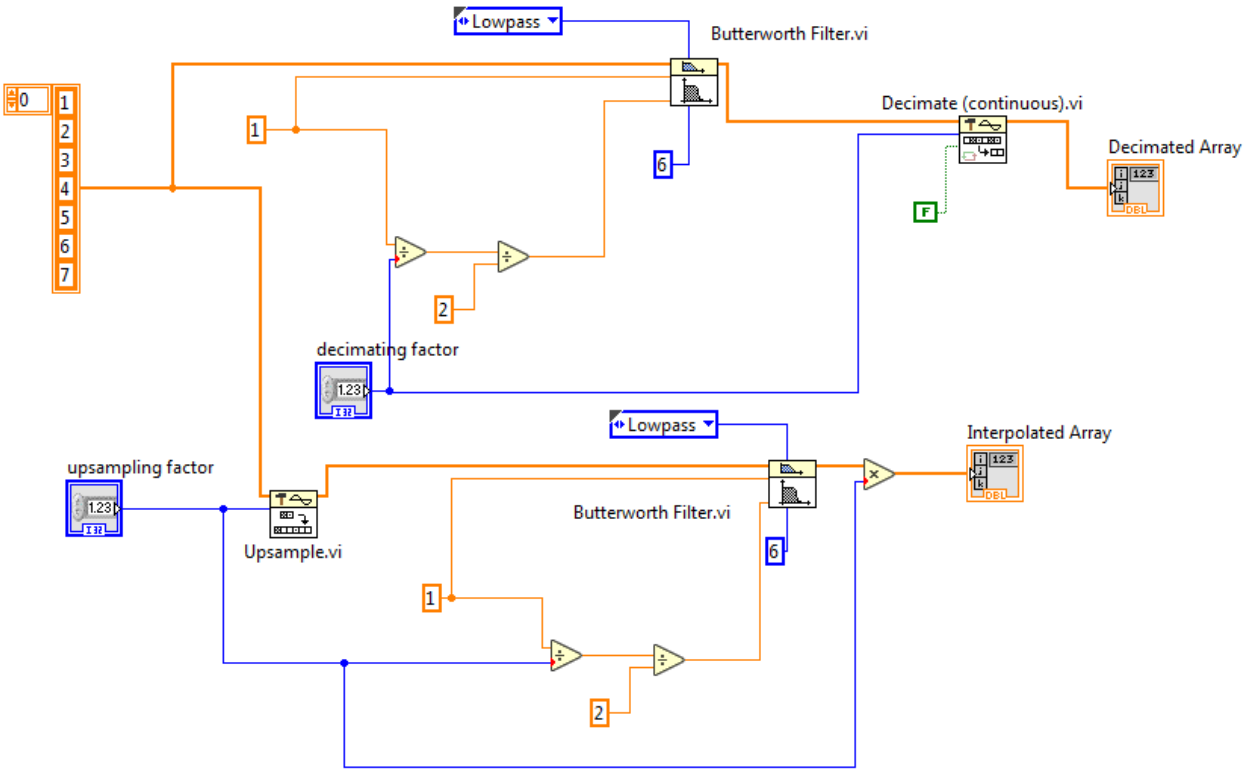

Fig. 6. Decimating with prefiltering and interpolation.

g) Why do we give the low cut-off frequencies of the filters as ***“1/(2*decimating factor)”*** and ***“1/(2*upsampling factor)”*** where **sampling frequency** is **1** ?

h) Now, change the ***constant 1*** **(sampling frequency) to 1000 for both filters**. **Do the output arrays change? Why?**

III. Experimental Work

PART 1

MATLAB Programming Tasks

a) Write a code in MATLAB to implement **decimation** operation. Given the input signal, and decimation factor, M, the code obtains the output. The **input** and **output** signals are **plotted** both in **time** and **frequency**.

b) Repeat **a)** for **interpolation**.

c) Write a code for sampling rate change with a **rational factor**. **Plot** the **input** and **output** both in **time** and **frequency**.

d) Implement a **digital PLL** structure. In this case, assume that there is a **sinusoid** where **you would lock onto its frequency and phase.** You should be able to **change the frequency and phase of this sinusoid.** The second sinusoid is an **internally produced** one. It may have a constant frequency and you can use decimation and interpolation to match its frequency with the first sinusoid. For this purpose, you need to **detect the frequency of the first sinusoid by using the FFT magnitude.** Then **determine the decimation and interpolation factors.** After the frequency of the second sinusoid is brought to the lock range of the PLL, PLL works on to lock onto the fine frequency and phase. You should **plot all the critical signals in time and frequency**.

PART 2

Real-time Programming Task

The programming task done in MATLAB is to be repeated in real-time using MyRIO. In this case, some blocks will be added to the code developed in Experiment 3. In this respect, a sinusoid which is assumed to be the input of the PLL for locking to frequency and phase is generated in MyRIO FPGA, then is input to the analog input for A/D sampling. The samples are transferred to MyRIO CPU through DMA operation. The second sinusoid representing the VCO in PLL is obtained by using the LabVIEW sinusoidal waveform generator, **Sine Wave.vi.** Since the programming structure in Experiment 3 is used accordingly, the project looks similar as shown in Fig. 7.

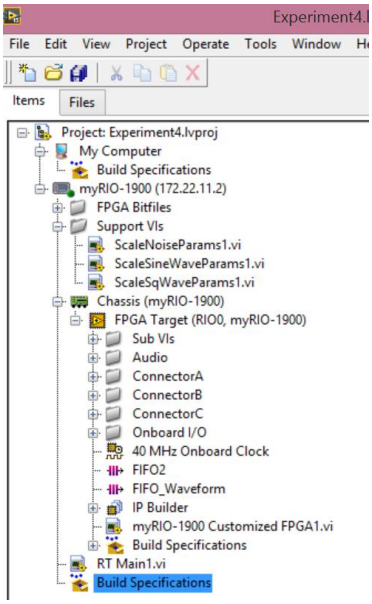

Fig. 7. Project structure for Experiment 4.

Fig. 8 shows the front panel for Experiment 4.

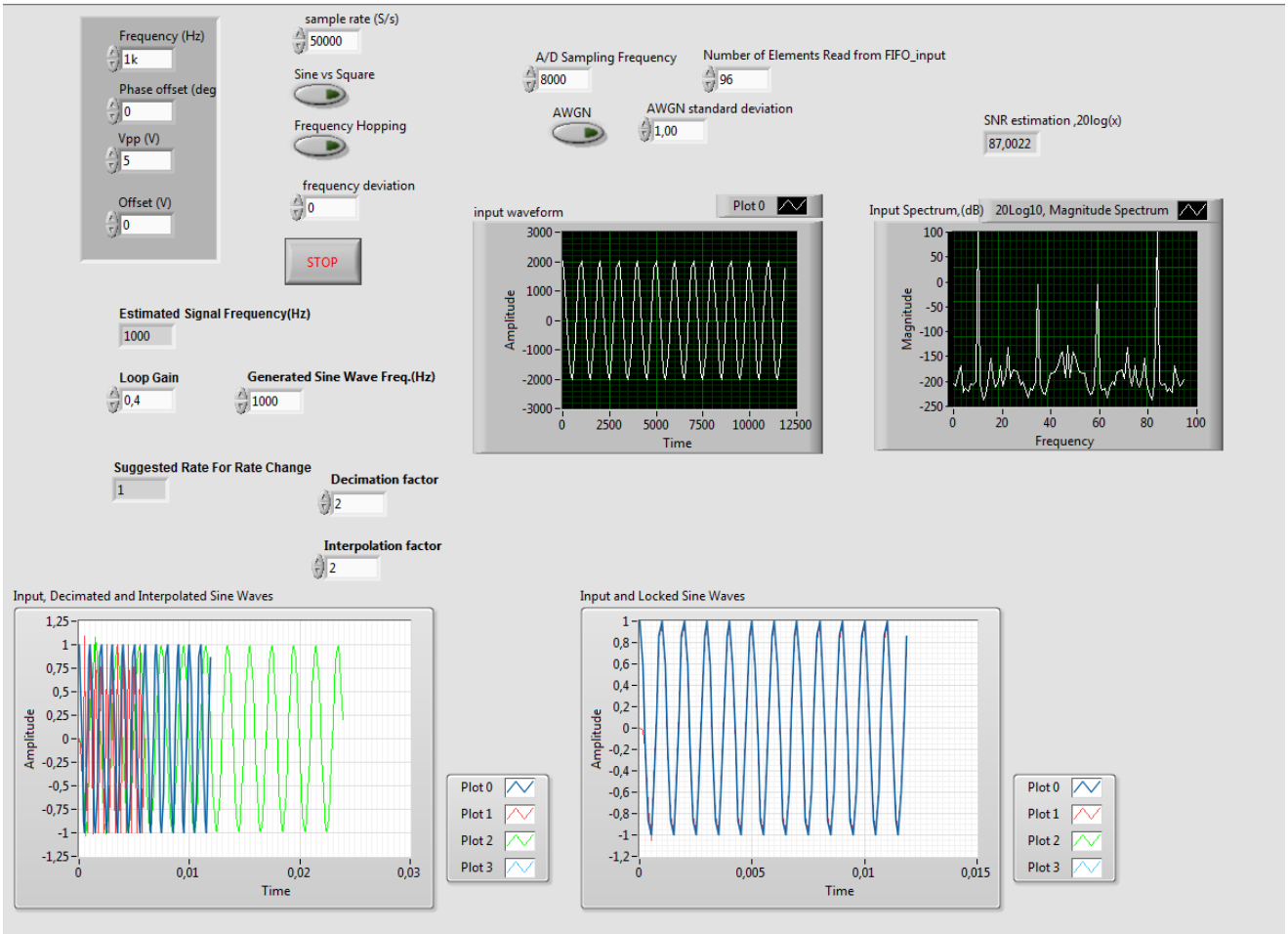

Fig. 8. Front Panel for Experiment 4.

The upper part belongs to the parameters of the desired sinusoid or test signal. This sinusoid is obtained by A/D sampling as in Experiment 3. The upper plots are the time and frequency characteristics of this sinusoid sampled at 8kHz. The estimated frequency of this signal is presented through an indicator below **"Estimated Signal Frequency"**. The second sinusoid frequency is the input **"Generated Sine Wave Freq. (Hz)"**. Since the two sine wave frequencies are known, rate change factor can be found. This is given under the **"Suggested Rate For Rate Change"** heading. Hence both the **decimation and interpolation factors** are selected as **2**. **PLL loop gain** is the input and selected as **0,4**. The user can add white noise with a certain variance to the input signal when the AWGN button is pressed. The lower left plot shows the input, decimated signal and the interpolated signal respectively. The lower right plot shows the input and the VCO output signal in time. In this case, PLL works appropriately and the VCO output locks to the input signal.

In the following part, the programming steps for Experiment 4 are outlined. Note that the **same FPGA VI as in Experiment 3 is used** and hence the programming in Experiment 4 involves only the **MyRIO CPU**.

1) Run the FPGA VI you generated in Experiment 3 and compile it.

2) Open the myRIO CPU VI you generated in Experiment 3. You will not delete anything you previously added. But, you will not need all the plots from the Experiment 3 in the Front Panel of this experiment. Hence, you can change the places of plots as shown in Fig. 8.

3) Add the following blocks in Fig. 9 to the **second while loop inside the second cell** in **myRIO CPU**. This while loop is the one where you read the **samples of A/D converter** through **FIFO_input in Experiment 3**.

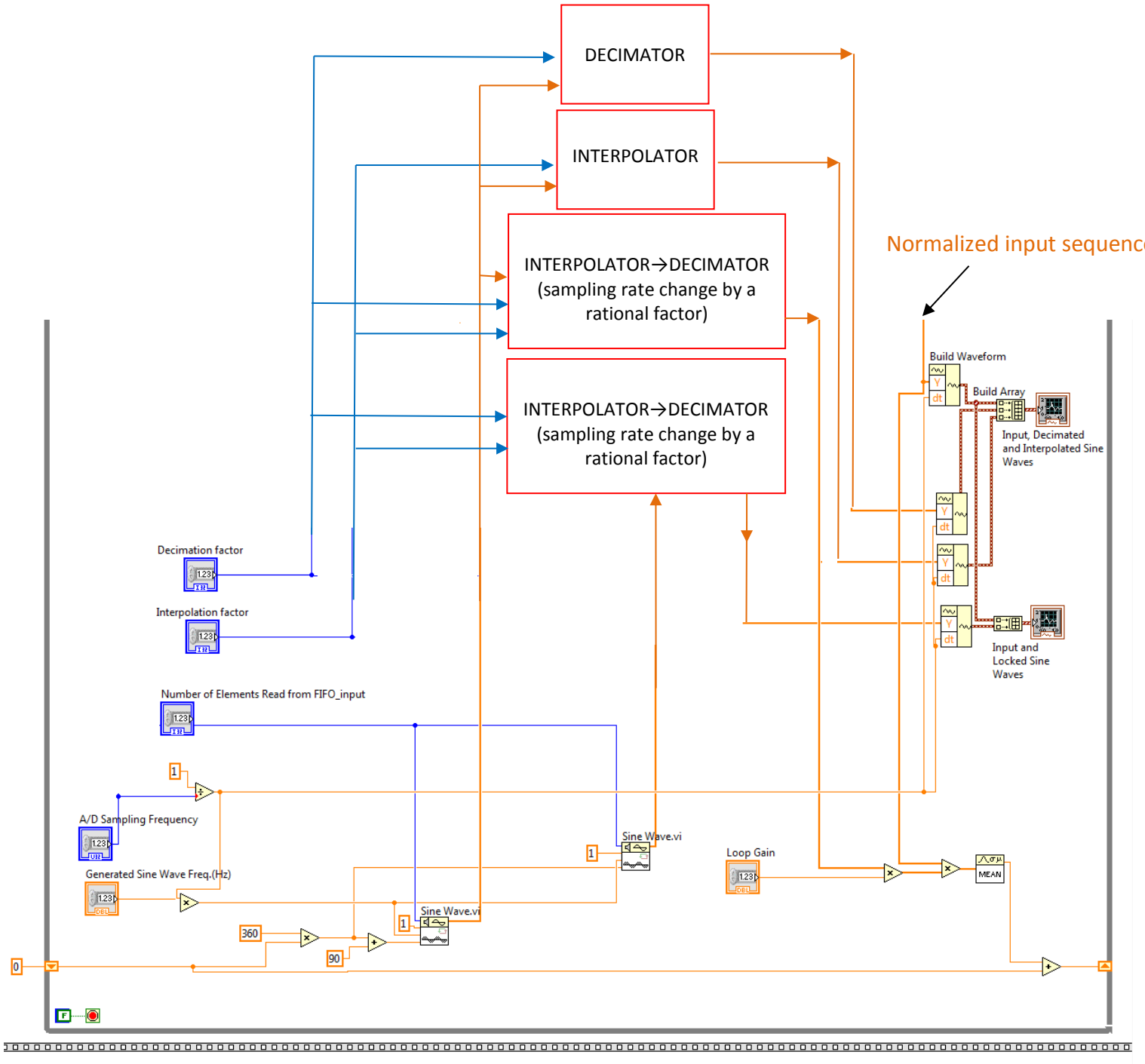

Fig. 9. Decimator, Interpolator and PLL.

- Note that the normalized input sequence in Fig. 9 is obtained as shown in Fig. 10.

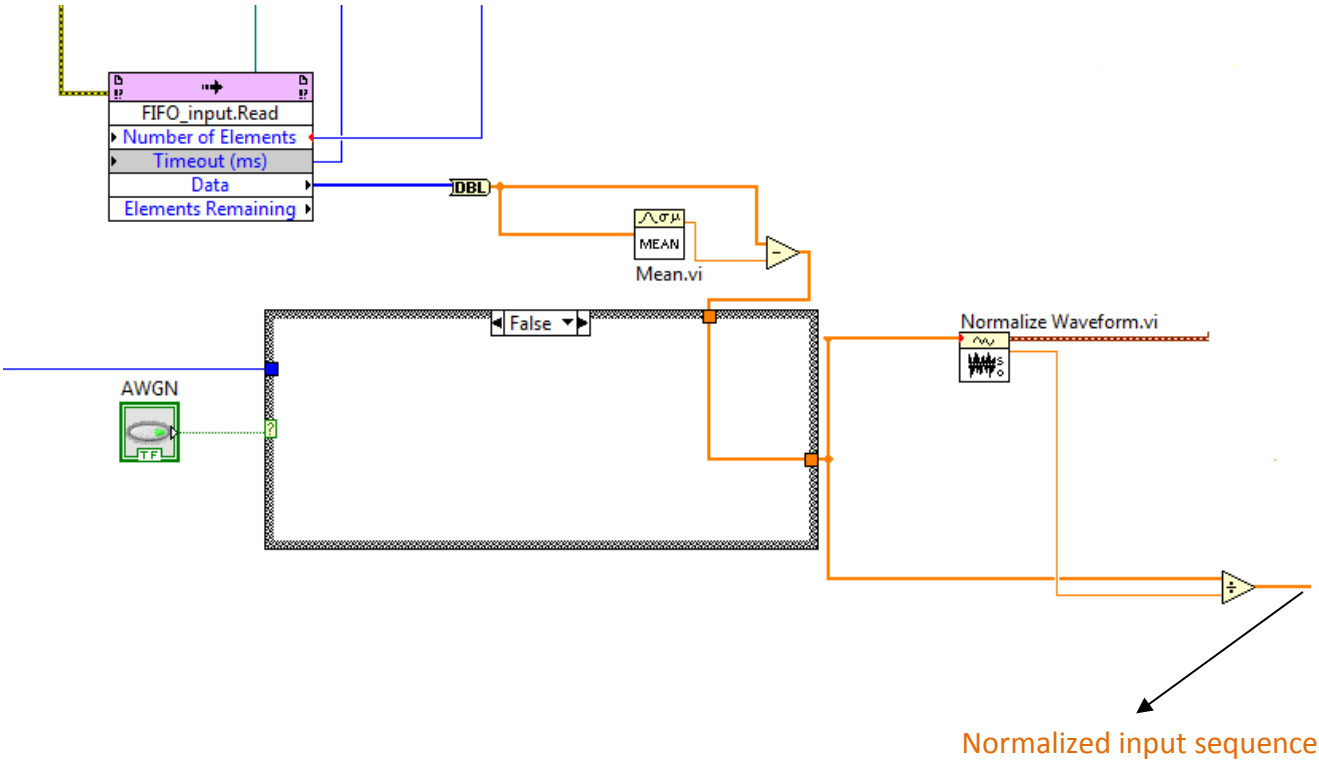

Fig. 10. Normalized Input Sequence.

4) Implement the **decimation by a given factor** as shown in **Fig. 1** and **Fig. 6**. You can use **Butterworth filter** and **Decimate (continuous).vi** in LabVIEW.

5) Implement the **interpolation by a given factor** as shown in **Fig. 2** and **Fig. 6** respectively. You can use **Upsample.vi** and **Butterworth filter** in LabVIEW.

6) Implement the **sampling rate change by a rational factor** by combining the codes in 4) and 5). You can use **Fig. 3** as a reference.

- Note that you will implement this block **(*sampling rate change by a rational factor*) twice** as shown in Fig. 9. The input of the upper block is cosine sequence whereas the lower one uses sine sequence. The output of the upper block is used as the output of VCO due to cosine convention in implementation of PLL. The other one is plotted together with input sequence in order to show locking for better visualization.

7) Obtain the "***Suggested Rate for Rate Change"*** by using the ***estimated frequency of the input*** and ***"Generated Sine Wave Freq.(Hz)"*** control in Fig. 9. During the

experiment, you will use this value in determining decimation and interpolation factor for PLL.

In the following part, it is assumed that you finished your programming tasks and the program output is evaluated and different parameters are changed to observe and explain the resulting waveforms.

Task

1) Adjust your program parameters as shown in Fig. 8 and run your program. Observe if the PLL locks onto the input frequency/phase or not. Change the **"Phase offset"** on the upper left part of the front panel to see if the PLL correctly follows the input signal.

2) Gradually **increase** the ***input frequency*** **from 1000Hz in 1Hz steps**. Determine the frequency where the PLL is **not locked any longer**. Note the maximum offset frequency which determines the lock-in range. Report this value. You can play with the **loop gain** in order to have the PLL lock. Also you can change the **number of elements read from the FIFO** in order to stabilize the waveform to see if the PLL is in lock.

3) Now you need to observe the effect of the loop gain. Choose the **input signal frequency** as **1kHz**. Starting **from 0.05 increase the loop gain to 2 in 0.2 steps**. Determine the value of loop gain where the PLL does not lock on the input signal. **Comment** on the effect of small and large loop gain.

4) Return to the **original settings in Fig. 8**. Now change the **input frequency** to **2008 Hz**. Also adjust the Decimation and Interpolation factors. What happens to the PLL output? What should be the value for the **decimation and interpolation factors** to observe PLL lock? **Repeat the same steps for 505Hz.**

5) Return to the **original settings in Fig. 8**. **Press the AWGN button** to add white noise to the input signal. Observe the **"Estimated Signal Frequency" while you increase the noise standard deviation.** At which value, estimated frequency starts to change? What can you say about the detection method optimality?

6) Return to the **original settings in Fig. 8**. Choose the **frequency deviation as 1Hz**. Apply **frequency hopping mode**. Observe if the PLL locks onto the frequency hopping signal. Now start to **increase the frequency deviation in 1Hz steps**. Determine the value for frequency deviation where the PLL **does not lock onto the signal any longer**. Report this value. **Comment on the differences** between this step and step 2.