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 3. SIGNAL GENERATION, FILTERING, CROSS CORRELATION, A/D, D/A, DMA

I. Introduction

In this experiment, some of the main resources of myRIO are utilized to generate and process signals. In this respect, both the CPU and FPGA are used to implement real-time signal processing tasks. The programing structure constitutes the fundamental part of this course. Hence, some or the whole of the VI's are used in the following experiments.

First a signal is generated in accordance with the user selected parameters. This signal is transferred from FPGA to myRIO CPU through DMA operation. In the FPGA, signal samples are fed to the D/A converter. The analog signal at the analog output AO0 pin is connected to the analog input AI0 pin through a cable. In the FPGA, analog input is sampled with A/D converter and the samples are transferred to myRIO CPU through DMA. These samples are then processed in myRIO to observe the response in the front panel.

II. Preliminary Work

1)

a) Generate the following **10 point sequence x** and **5 point sequence h** in MATLAB.

x= [1 2 3 4 5 6 7 8 9 10],

h= [10 -8 6 -4 2].

b) Using ***"conv"*** command, find the convolution of x and h. Plot the result with ***"stem"*** command. **Attach this plot** to your preliminary work.

c) Using ***"xcorr"*** command, find the cross correlation of x and h. Plot the result with ***"stem"*** command. **Attach this plot** to your preliminary work.

d) Now, **plot** the convolution of x with **flipped version of h** using **"stem"** command. **Attach this plot** to your preliminary work. You can use ***"fliplr"*** command in order to obtain flipped version of h.

e) Is there any **similarity** between the results obtained in **b, c, and d**? **Comment** on the **similarity and differences** between these three results.

2)

a) Generate **200 Hz sine sequence** with sampling rate **10000 samples/s in 0-80 ms** range. **Plot** it with **appropriate time axis**, i.e., **0-80 ms**. **Attach this plot** to your preliminary work.

b) **Cross correlate the sine sequence** you generated in part a) with **itself**. **Plot** the result and **attach** it to your preliminary work.

c) Now, generate a **chirp signal** with sampling rate **10000 samples/s in 0-80 ms** range with **start frequency 0** and **end frequency 200 Hz** using **"chirp"** command. **Plot** it with **appropriate time axis, i.e., 0-80 ms**. **Attach this plot** to your preliminary work.

d) **Cross correlate the chirp sequence** you generated in part c) with **itself. Plot** the result and **attach** it to your preliminary work.

e) **Compare** the cross correlated sequences you obtained in part b) and d).

f) **Cross correlate the chirp signal** in part c) with the **sine** signal and **plot** the result. **Attach** it to your preliminary work.

g) Generate **another chirp signal** with sampling rate **10000 samples/s in 0-80 ms** range with **start frequency 200 Hz** and **end frequency 400 Hz**. **Cross correlate this chirp signal with sine signal** and **plot** the result. **Attach this plot** to your preliminary work.

h) Generate **another chirp signal** with sampling rate **10000 samples/s in 0-80 ms range with start frequency 400 Hz** and **end frequency 600 Hz**. **Cross correlate this chirp signal with sine signal** and **plot** the result. **Attach this plot** to your preliminary work.

i) **Compare** the results in part **f), g), h)**. **Comment** on them.

j) Now, generate **200 Hz square wave** with sampling rate **10000 samples/s in 0-80 ms** range. **Plot** it with **appropriate time axis**, i.e., **0-80 ms**. **Attach this plot** to your preliminary work. (You can use **"square**" command in MATLAB)

k) **Cross correlate the square wave** in part **j)** with the chirp signal in **h)** and **plot** the result. **Attach this plot** to your preliminary work. **Compare** the result with the plot in **h)** and **comment on the differences** considering frequency content of the square wave (harmonics).

III. Experimental Work

In this experiment, myRIO CPU and FPGA are programmed while the interaction between two is established through **DMA** operations.
In the following part, real-time programming tasks are described in a step-by-step manner.

PART 1

In this part, a **square & sine wave** will be generated and then converted to **analog output** by the **digital to analog converter (D/A) in myRIO**. This analog output will be observed both in the **front panel** and **oscilloscope** screen.

1) Create a myRIO project by following the same steps in Experiment 2.

2) **Right click on NI-myRIO** in the **Project Explorer** window and select **New→Targets and Devices** to add the **Chassis** as in Experiment 2. Now you should see the project as in Fig. 1.

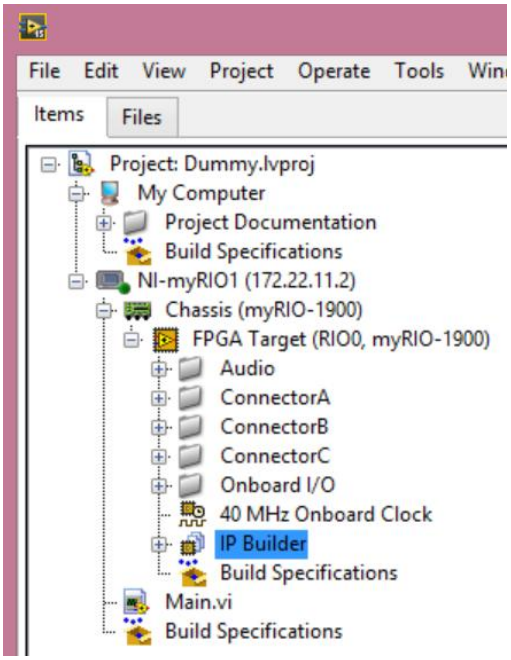

Fig. 1. Project Outline.

- In your project, there are three mediums where you can do programming. **My Computer** is the computer platform where you can run any program but this is not a real-time programming environment. NI-myRIO has its own **CPU** and **FPGA** and hence you can program both for real-time operation. In order to program any medium, you need to place a ".vi" program under that medium. You can expand each item to see the available ports and devices.

- In this experiment, you need to program both the FPGA and CPU present in myRIO.

FPGA Programming

3) Right click on FPGA Target and **New→VI**.

4) Select the Block Diagram. From the **Functions** select the **Flat Sequence Structure** and place on the block diagram. Inside the **first cell**, place the following timing loop in Fig. 2.

- This frame sets the execution rate of the loop. The actual loop rate is also measured and checked to make sure the desired loop rate is achieved. Actual loop rate will also be used in CPU VI to determine the time axis.

5) Check that the ***"Counter units"*** is ***"Ticks"*** as shown in Fig. 3 by **double clicking on Loop Timer**.

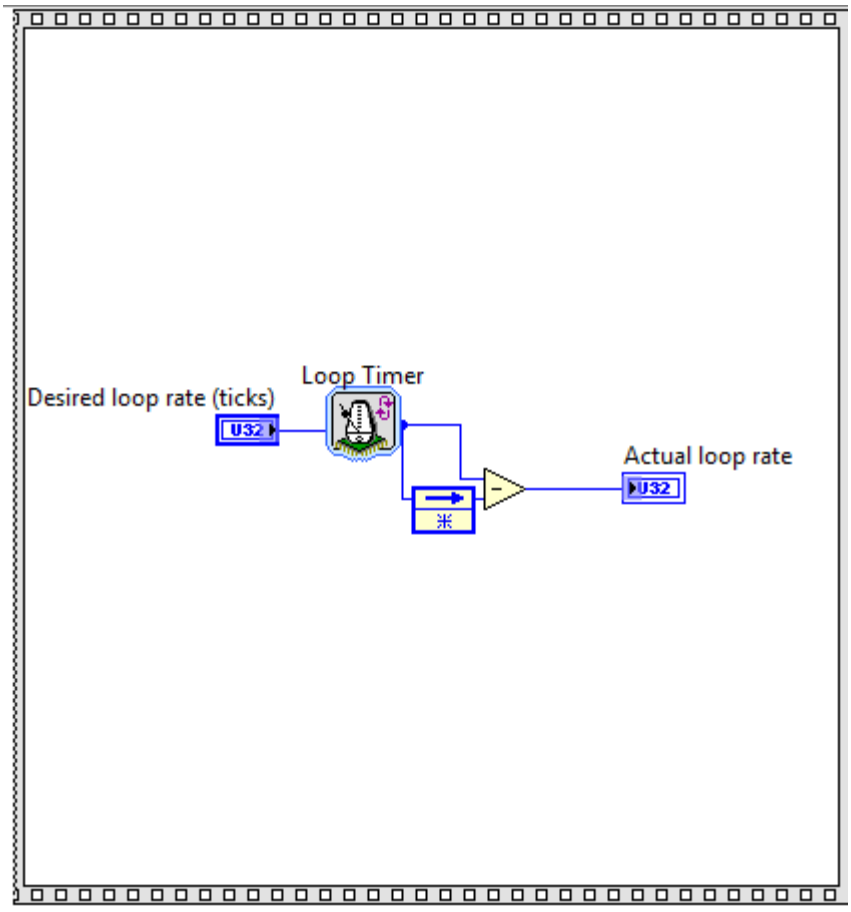

Fig. 2. FPGA Timing Loop.

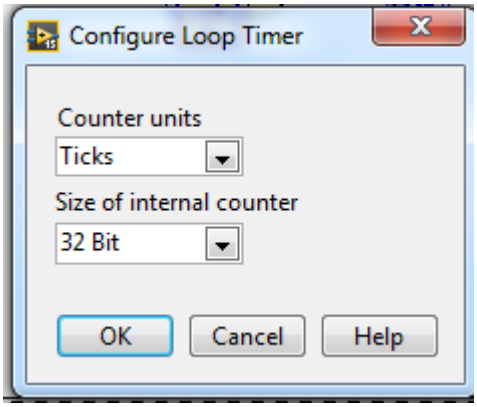

Fig. 3. Configure Loop Timer.

6) Add a **new cell to flat sequence**. The inside of this cell is shown in Fig. 4. You will get the input parameters from the CPU VI.

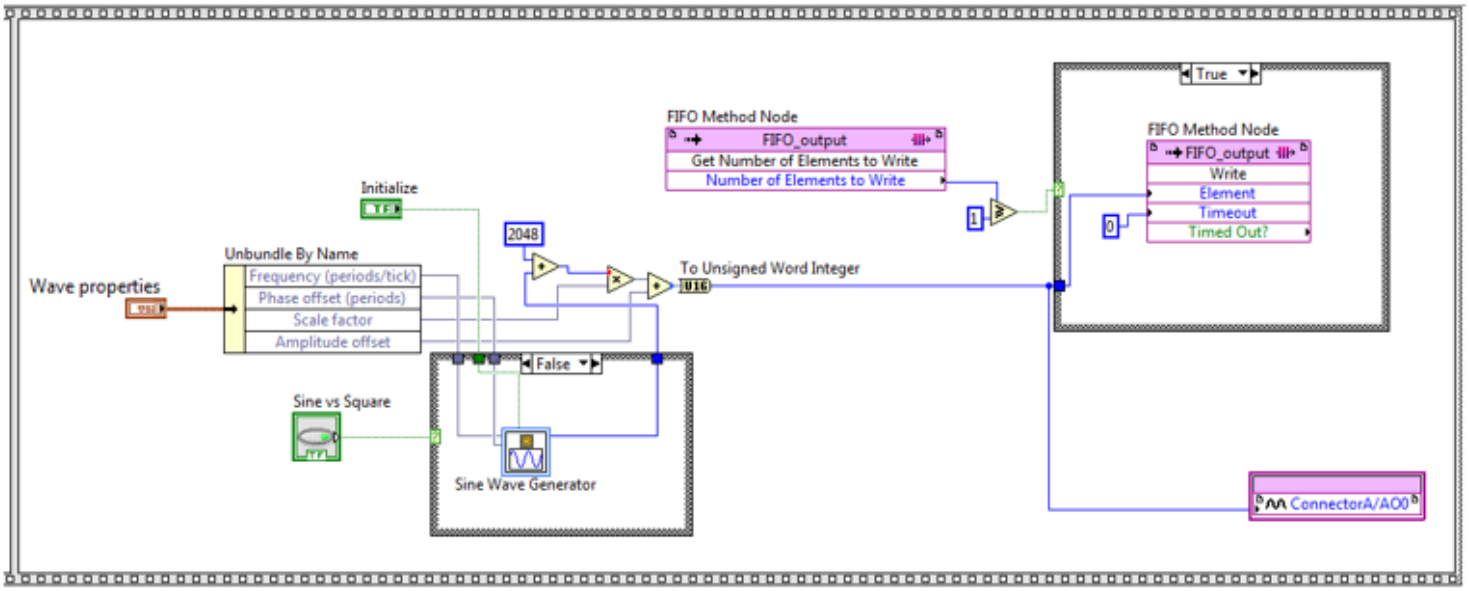

Fig. 4. Second cell of flat sequence.

7) ***"Wave properties"*** is a **cluster control of 4 fixed point elements** as shown in the Context Help window in Fig. 5a.

- Adjust the data type of these 4 fixed point controls as shown in Fig. 5b by right clicking on each control in the front panel and selecting **Properties → Data Type**.

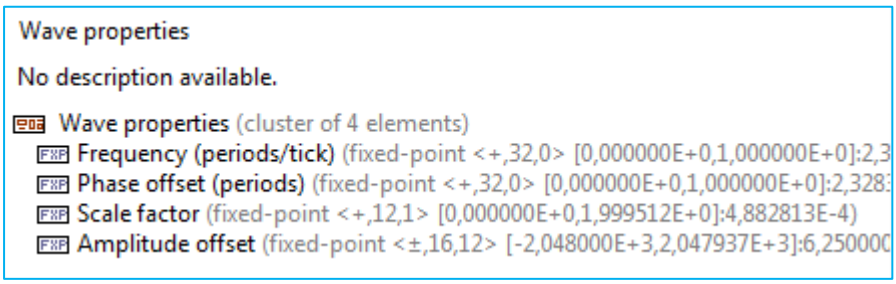

Fig. 5a. Wave properties.

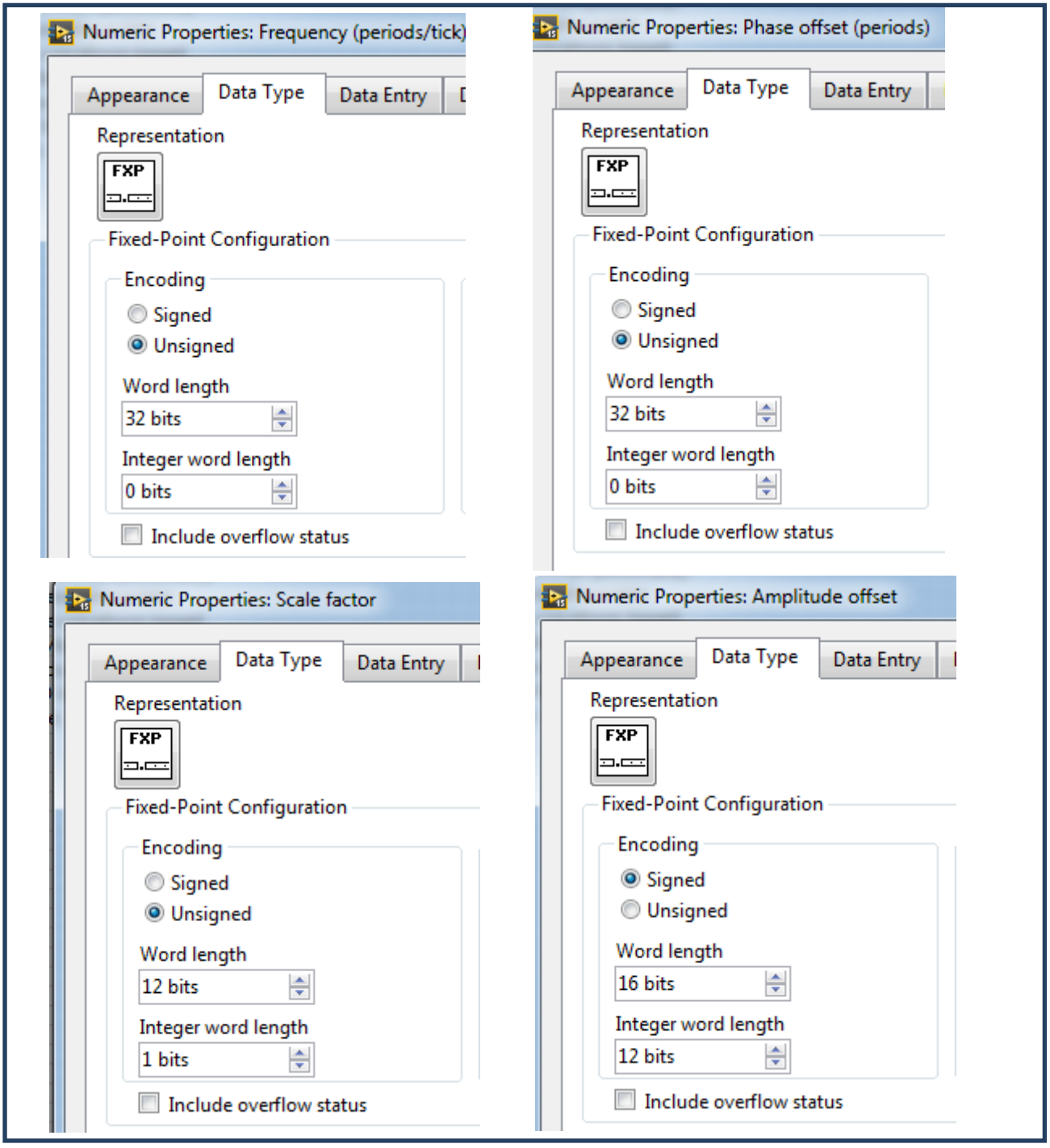

Fig. 5b. Data type of fixed point controls.

8) For the Case structure in the leftmost side, there are two cases. If **Sine vs Square** is **false**, then a sine wave will be generated, otherwise a square wave is generated.

9) The Context Help window for **Sine Wave Generator** is shown in Fig. 6. When you first place this function on the Block Diagram, frequency and phase offset are not adjusted as inputs. In order to view these terminals click **Show frequency terminal** and **Show offset terminal** as in Fig. 7 from Configuration window.

- In addition, set the **amplitude** as **2048** as shown in Fig. 7 from Configuration window.

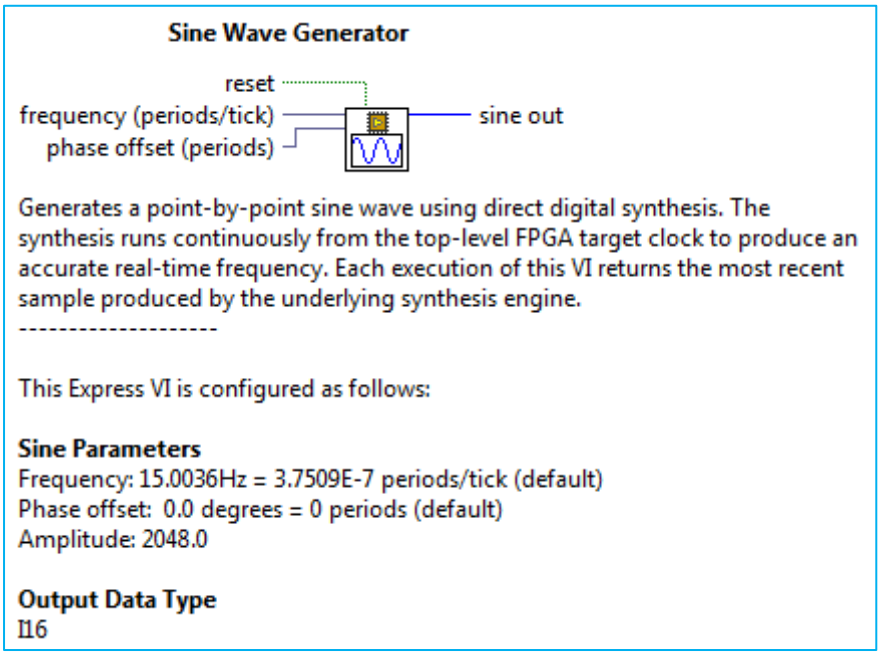

Fig. 6. Sine Wave Generator.

- Note that the **data type of output of Sine Wave Generator** is **I16**. Since the analog output 0 (AO0) of myRIO gives output in **0-5 V range and D/A resolution is 12 bits**, the **data type of the input terminal AO0 in FGPA VI** should be **U16**. For this purpose, the result is first brought to the range **(0, 4096)**, scaled and then converted into **U16** data.

10) In the Fig. 4, the False case corresponding to sinusoidal output is shown. Using **another Sine Wave Generator**, complete the inside of the **True case** by generating a **square wave with -2048 and 2048 levels**. As a hint, you can use ***Select*** function in order to decide whether 2048 or -2048 will be generated by looking at the sine value.

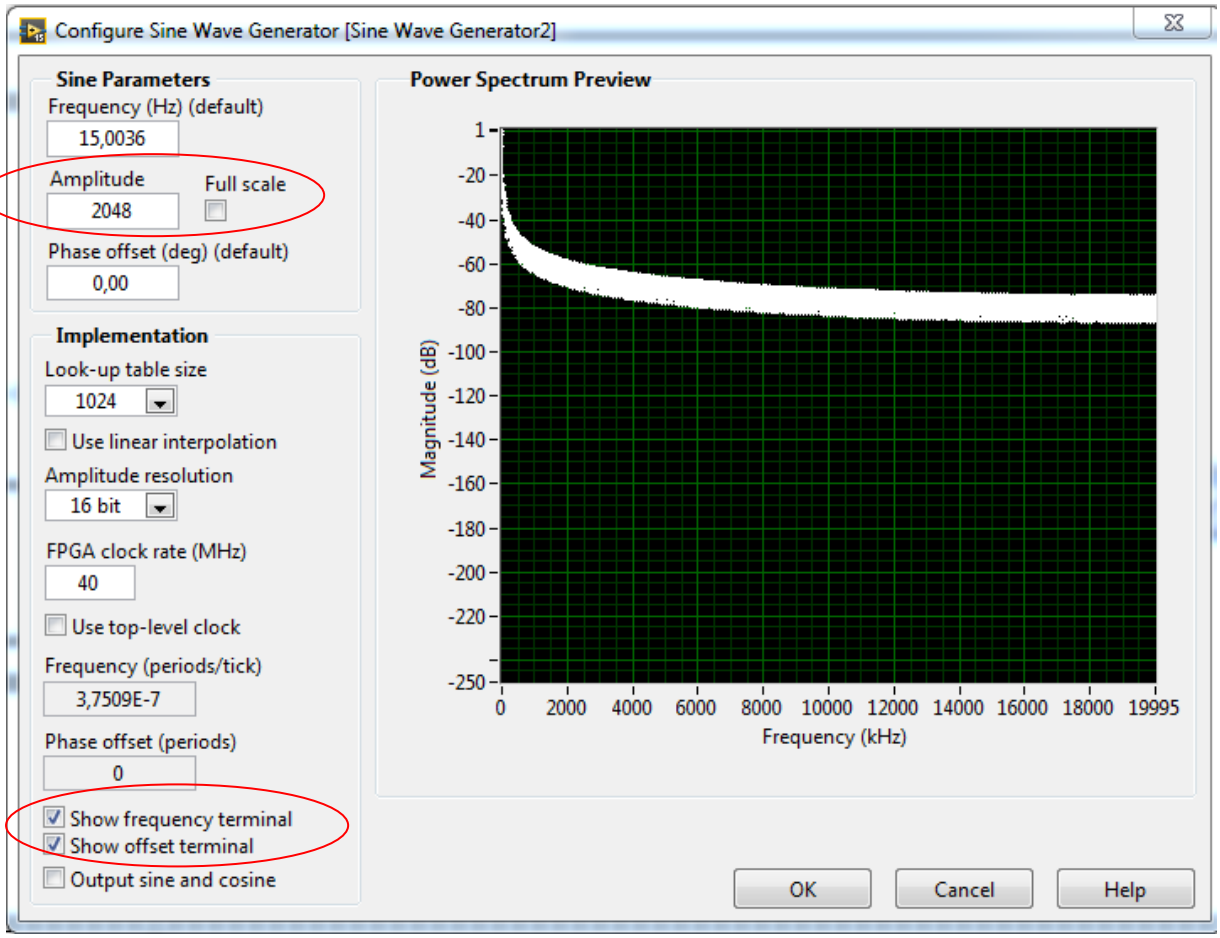

Fig. 7. Configure Sine Wave Generator.

11) Generate a FIFO called **"FIFO_output"** for **transferring the samples of waveform to the CPU memory** through a DMA operation as shown in Fig. 4 by following the same steps in Experiment 2. The type of FIFO_output is **"Target to Host – DMA"** and data type is **U16**.

12) For the other Case structure, do nothing when the **Number of Elements to Write** is less than 1.

13) You can drag and drop the analog output **ConnectorA/AO0** for **D/A conversion** from the **project FPGA ports**.

14) The complete FPGA programming block should be placed inside a while loop to run in real-time continuously as shown in Fig. 8. Moreover, create a stop button.

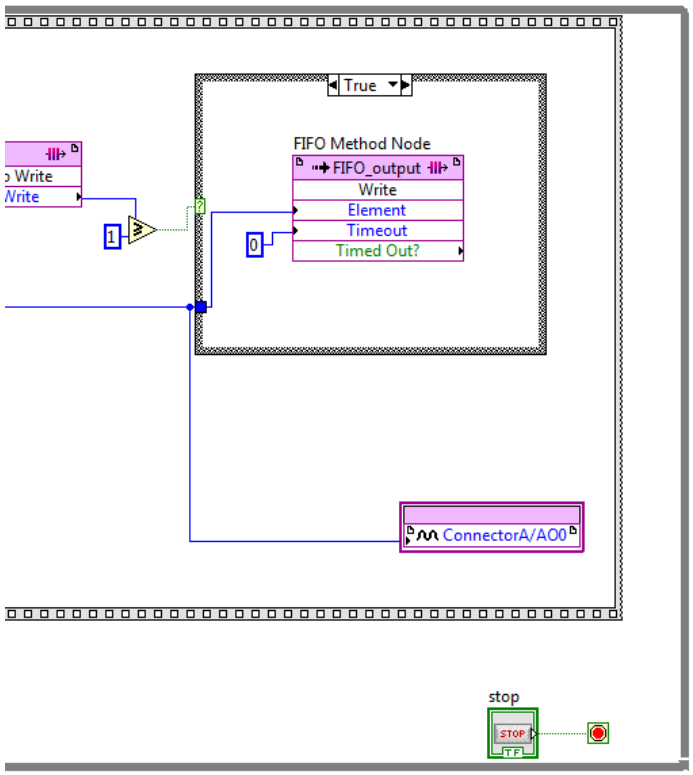

Fig. 8. Placing the flat sequence inside while loop and stop control.

15) Fig. 9 shows the description of **analog inputs and outputs** on **MXP Connectors A and B**. The generated output is given from AO0 output of Connector A. In order to observe the analog output voltage in the oscilloscope, **connect the AO0 and AGND terminals of Connector A to the breadboard** using the supplied cables as shown in Fig. 10. Fig. 11 denotes the **pin diagram of Connector A and B** of myRIO. **Pin 2 and 6** correspond to **AO0 and AGND**, respectively.

Signal Name	Reference	Direction	Description
+5V	DGND	Output	+5 V power output.
AI <0..3>	AGND	Input	0-5 V, referenced, single-ended analog input channels. Refer to the *Analog Input Channels* section for more information.
AO <0..1>	AGND	Output	0-5 V referenced, single-ended analog output. Refer to the *Analog Output Channels* section for more information.
AGND	N/A	N/A	Reference for analog input and output.

Fig. 9. Signal descriptions for Analog Input & Outputs on Connector A and B.

Fig. 10. AO0 and AGND on Connector A.

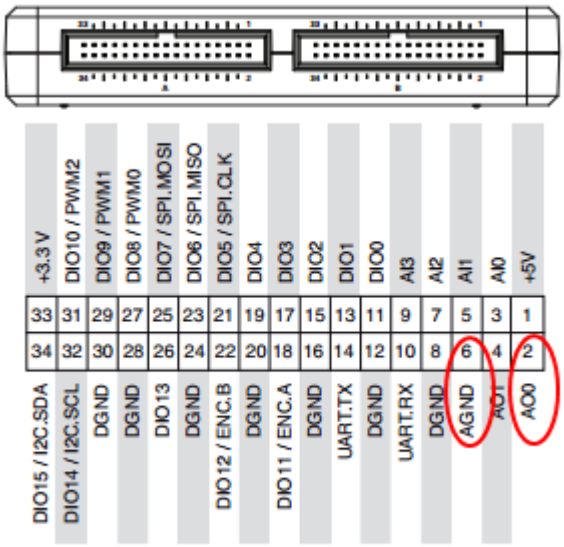

Fig. 11. Pin diagram for Connector A and B of myRIO.

16) Now, compile the FPGA code by running the FPGA VI.

myRIO CPU Programming

- **FPGA** is used to **generate the analog waveform**. Major signal processing routines are performed using the myRIO CPU. A VI will be placed under myRIO-1900 and used to implement the signal processing units. In this part, we will control the FPGA by adjusting waveform parameters from the user interface and read the waveform through DMA from **FIFO_output**. We will **plot the output waveform in time and frequency**. In the second part, an analog input is connected to the AO0 and the analog waveform will be sampled and more signal processing tasks will be implemented on the sampled signal.

17) Create a **new VI under myRIO CPU** by **right clicking on NI myRIO** in the project explorer window and select **New → VI**.

18) Place a **flat sequence** on the block diagram. **Inside of the first cell** should be seen like Fig. 12. Please remember the steps in Experiment 2 to construct this cell.

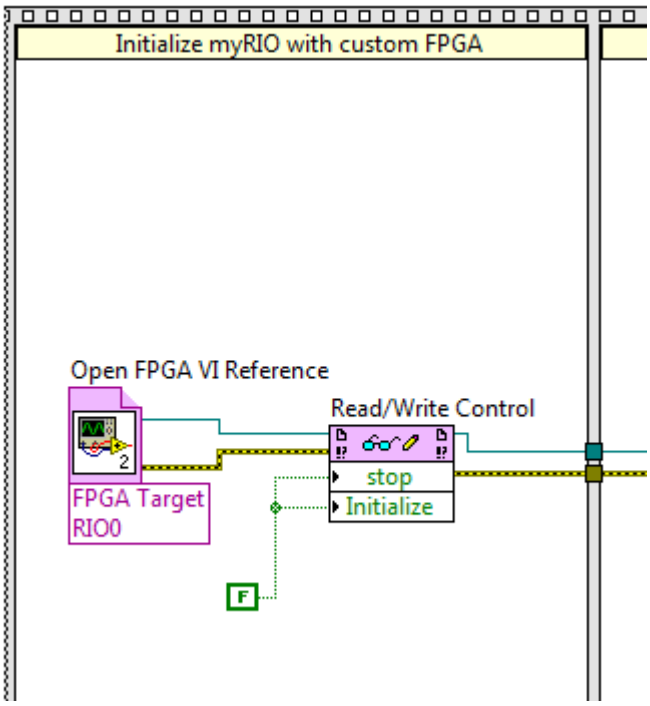

Fig. 12. Initializing myRIO with custom FPGA.

19) Add a **new cell to the flat sequence** and construct the code in Fig. 13 and 14 inside this cell.

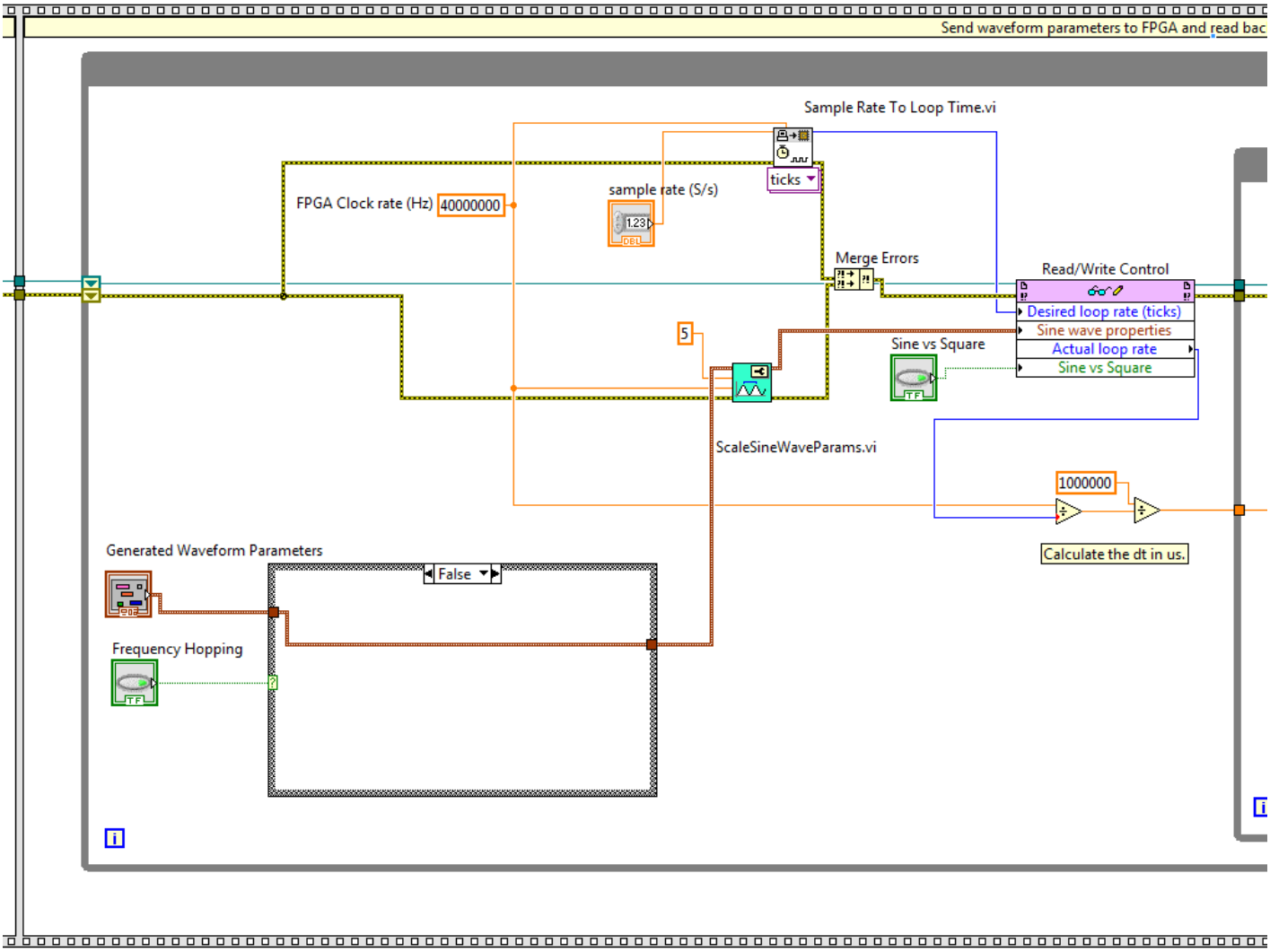

Fig. 13. Left side of the second cell in CPU VI.

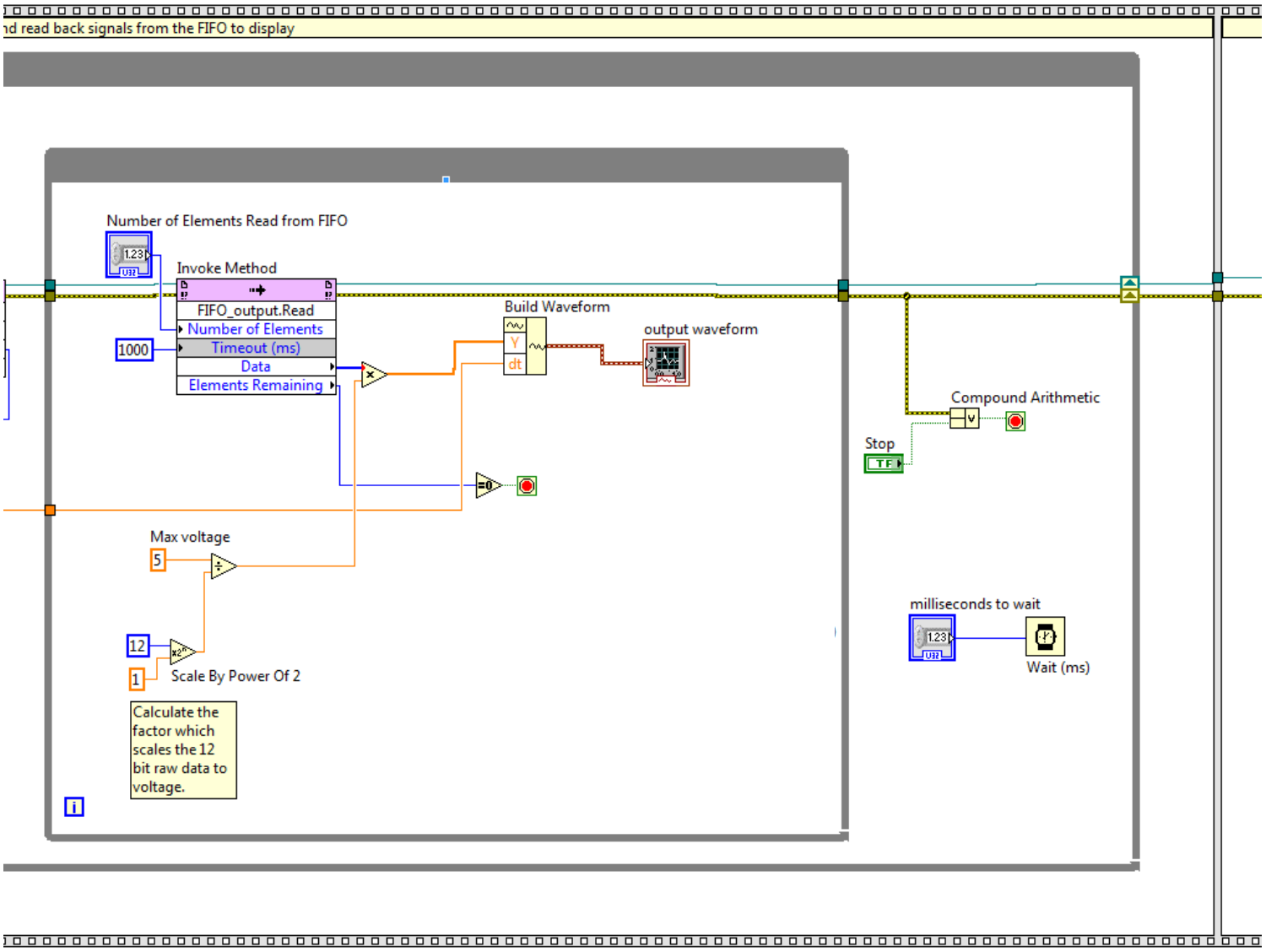

Fig. 14. Right side of the second cell in CPU VI.

- Note that **ScaleSineWaveParams.vi** will be supplied inside your project directory by the assistant. This VI is used to scale wave parameters entered form user interface such that they are compatible with **Sine Wave Generator in the FPGA VI**.

20) In the case structure of Fig. 13, the **False** case is viewed. Nothing is done when **Frequency Hopping** control is false. When this control is **True**, we want that the frequency of the wave changes randomly in some neighborhood of the selected frequency. **Write your own code inside the True case** for this purpose. You will use **random number generator** and some arithmetic operations. You will also define a **control input called "frequency deviation" in Hz**.

21) As you see in Fig. 14, the waveform given from analog output is read from **FIFO_output** and plotted in time domain. You are required to add some blocks in order to **plot magnitude spectrum of the waveform using fft as in Experiment 1**.

22) Finally, **add a third cell to the flat sequence** and construct in as I shown in Fig. 15.

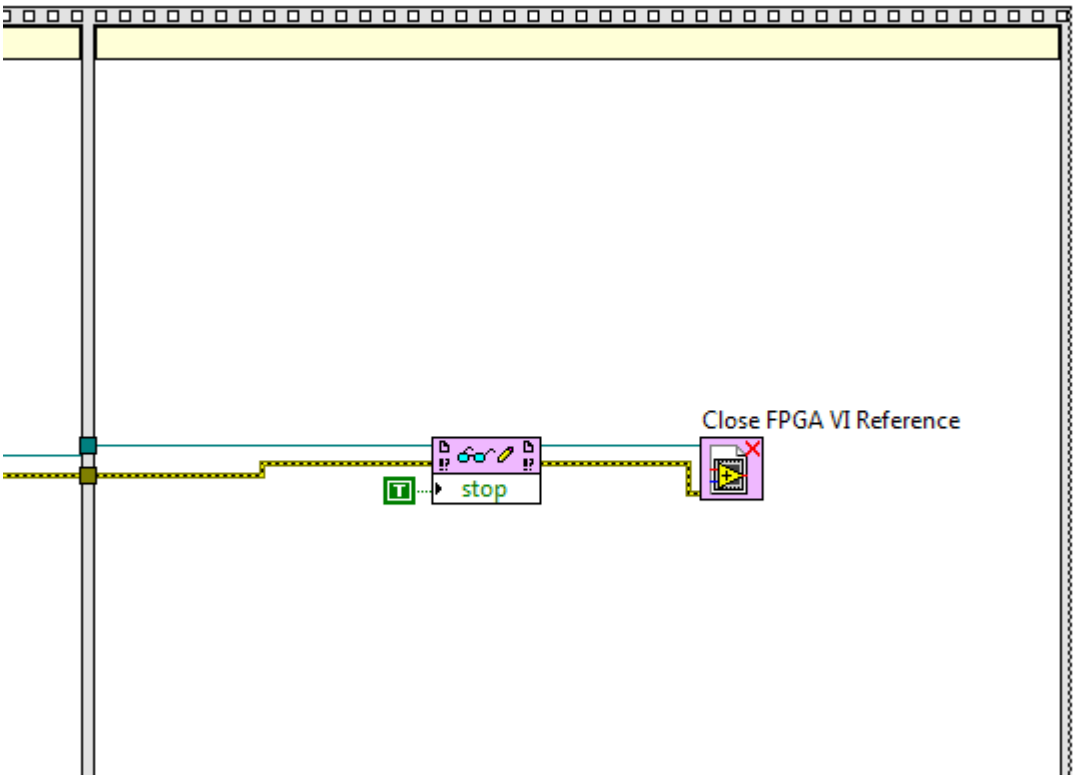

Fig. 15. Third cell of flat sequence in CPU VI.

23) Now, you can run your CPU VI and do the following task. An example front panel is shown in Fig. 16.

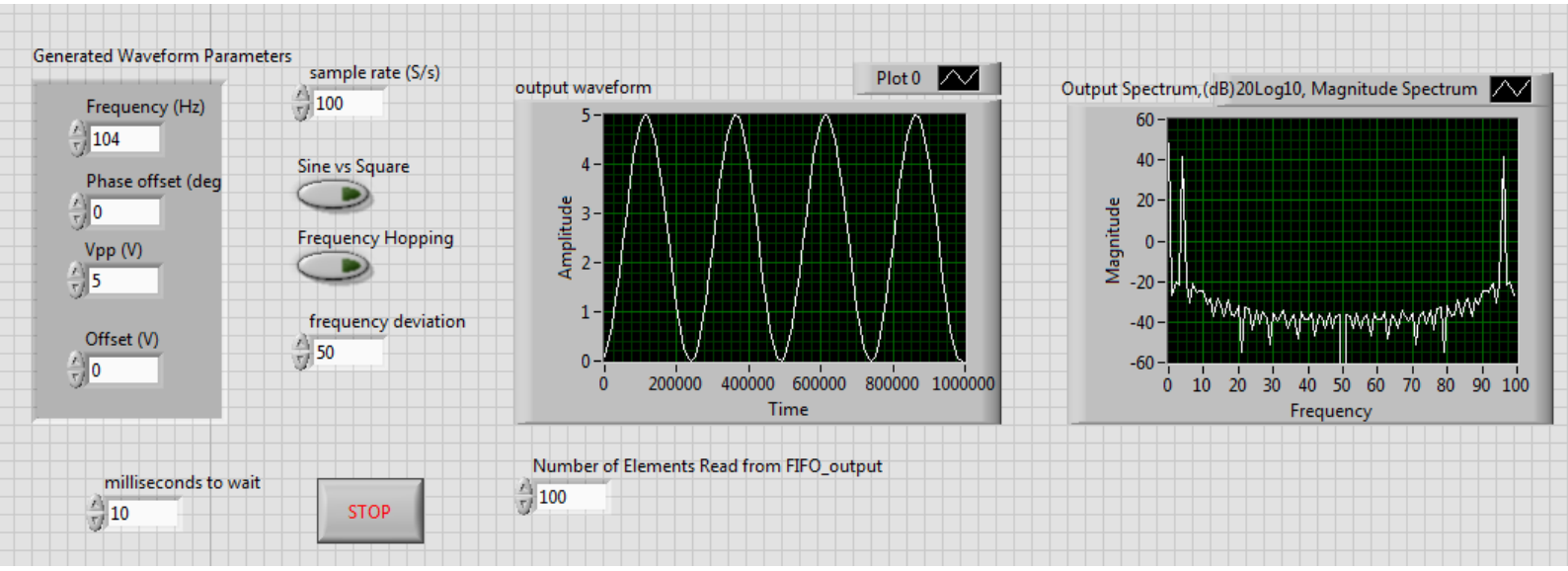

Fig. 16. Front panel of CPU VI for the Part 1.

Task 1

1. Generate a **sine wave without frequency hopping** where the waveform parameters are as follows. **Sampling rate** is **50000 Samples/s**, waveform **frequency** is **100 Hz**, **Phase offset** is **0 degree**, **Vpp** is **5 V** and **Offset** is **0 V**. **Observe the output waveform** both in the **oscilloscope and front panel**. Now, increase the frequency of sine wave **from 500 Hz until 3 kHz in 500 Hz steps**. **Comment on the change** in the waveforms.
2. Keep the **frequency of sine wave** at **3 kHz** and increase the sample rate from **50000 Samples/s to 500000 Samples/s in 50000 Samples/s steps**. **Comment on the change**.
3. Set the **frequency of sine wave** to **100 Hz** and decrease the sample rate from **45000 Samples/s to 5000 Samples/s in 10000 Samples/s steps**. **Comment on the change**.
4. Now, set the frequency of sine wave and sample rate to **1 kHz** and **50000 Samples/s**. **Turn on Frequency Hopping** option and adjust the **frequency deviation** to **50 Hz**. Observe the waveforms. Increase the frequency deviation **from 100 Hz until 500 Hz in 100 Hz steps**. **Observe the changes** in the waveforms and **comment on the results**.
5. **Repeat the steps 1, 2, 3 and 4 for square wave**.
6. Generate a **sine wave without frequency hopping** with sampling rate **100 Samples/s**. Set the **number of elements read from FIFO_output** to **100**. Adjust the **frequency of sine wave** to **101 Hz**. **What do you observe in the Front Panel? What is the frequency of observed signal**? **Comment** on the result.
7. Now, **increase the frequency to 105 Hz in 1 Hz steps**. **Comment on the frequencies** of the waveforms.
8. Increase the frequency of sine wave **from 1001 Hz to 1005 Hz in 1 Hz steps** while **sample rate is still 100 Samples/s**. Are the results **similar** to the ones in part 7 ? **Why?**

PART 2

In this part, we will **connect** the **analog output pin AO0** to the **analog input pin AI0**. We will **add some blocks to both FPGA and CPU VI's** in order to sample the analog input and implement the signal processing routines on the sampled signal. FFT, Butterworth filtering and cross correlation of the sampled signal with a chirp signal are the additional tasks we will implement in the CPU.

FPGA Programming

1) As you remember, there was a flat sequence inside a while loop in our FPGA VI in order to generate analog waveform at AO0. Our aim is to **sample the analog waveform at the input pin AI0**. For **A/D conversion**, add the following **flat sequence** in Fig. 17 to your **FPGA VI** inside **another while loop** with **stop 2 control**.

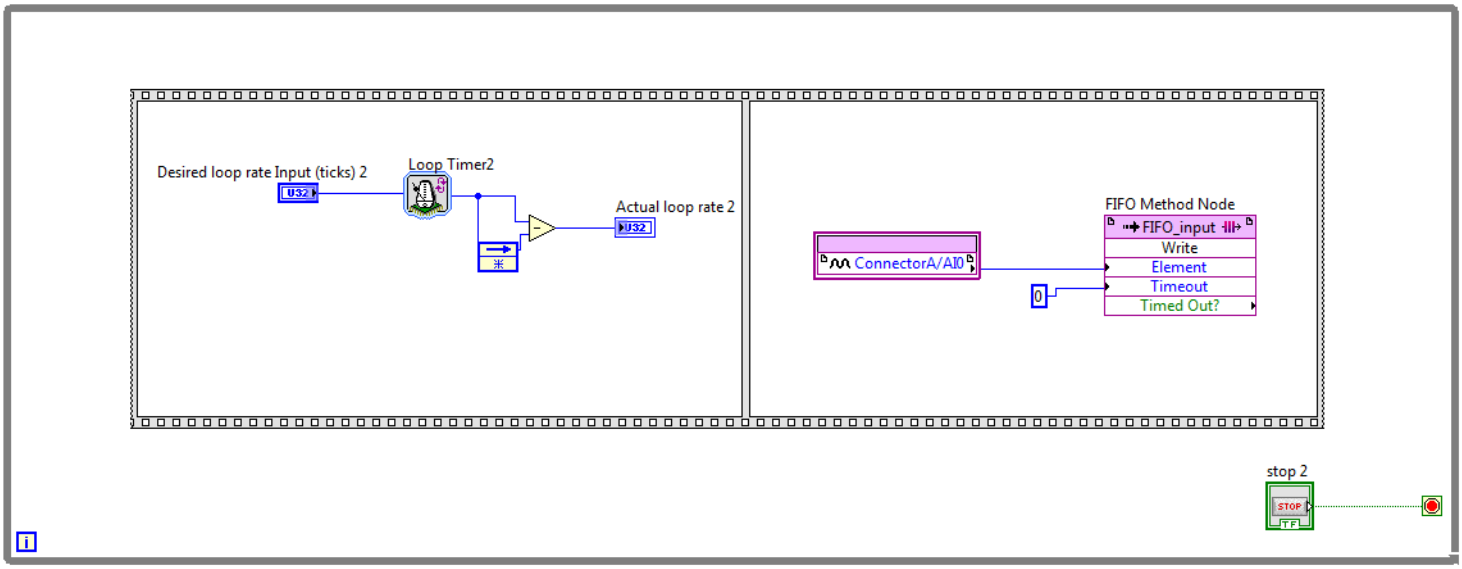

Fig. 17. Flat sequence for sampling analog input waveform.

2) Generate a FIFO called **"FIFO_input"** for **transferring the samples of analog input waveform to the CPU memory** through a DMA operation as shown in Fig. 17 by following the same steps in Experiment 2. The type of FIFO_input is **"Target to Host – DMA"** and data type is **U16**.

3) Check that the ***"Counter units"*** is **"Ticks"** as shown in Fig. 3 by **double clicking on Loop Timer 2.**

4) **Connect** the **analog output pin AO0 (pin No.2) to analog input AI0 (pin No.3) in Connector A** through the cable supplied.

5) Now, compile the FPGA code by running the FPGA VI.

6) We will add some blocks to CPU VI constructed in Part 1 in order to implement signal processing tasks on the sampled signal. First, generate **additional wires from the output terminals of Open FPGA VI Reference** as shown in Fig. 18. We will use the same FPGA reference for **reading the samples through FIFO_input**.

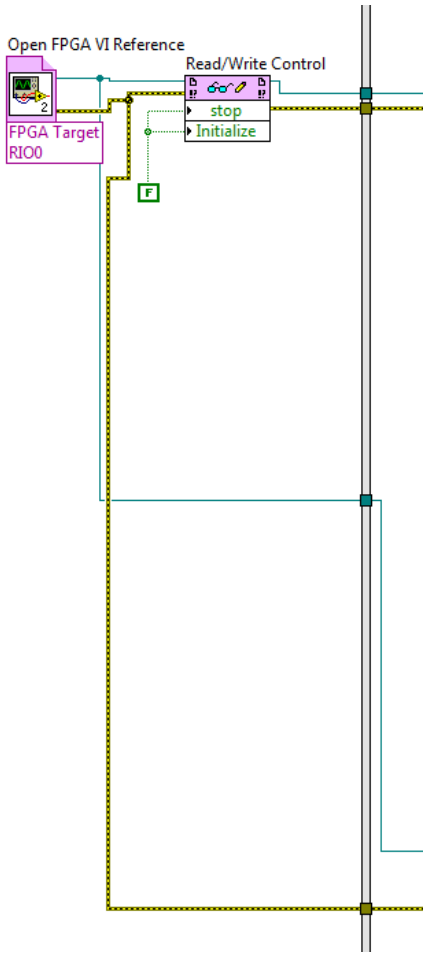

Fig. 18. The first cell of flat sequence in CPU VI.

7) Now, add **another while loop inside the second cell of flat sequence**. The code inside this loop is shown in Fig. 19.

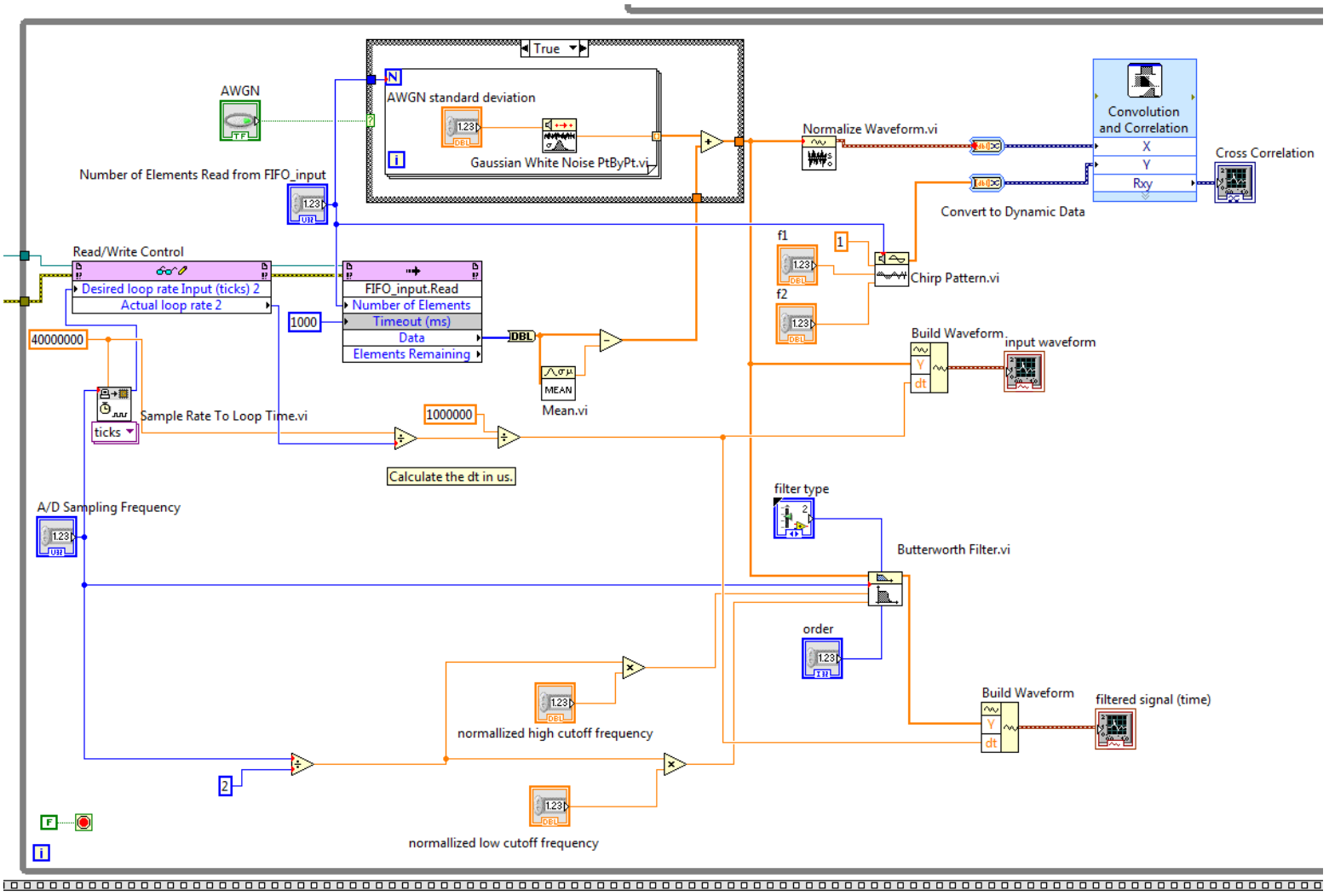

Fig. 19. Added while loop inside the second cell of CPU VI.

- Note that we **subtract the mean from the input signal and add noise (AWGN) if AWGN is true**. We **plot** the **input signal, filtered signal by Butterworth filter and cross-correlate the normalized input with a chirp** as shown in Fig 19.

8) Now, **add necessary blocks** to your code to view the magnitude **of fft for input and filtered signals as in Experiment 1**.

9) **Implement a structure for frequency estimation of the received sampled signal for sinusoidal input** and **print the estimated frequency on the Front Panel**.

10) **Implement a structure for SNR estimation for a sinusoidal input signal** and **print the estimated SNR on Front Panel**.

11) Now, you can run your VI and do the **Task 2**. An example front panel is shown in Fig. 20.

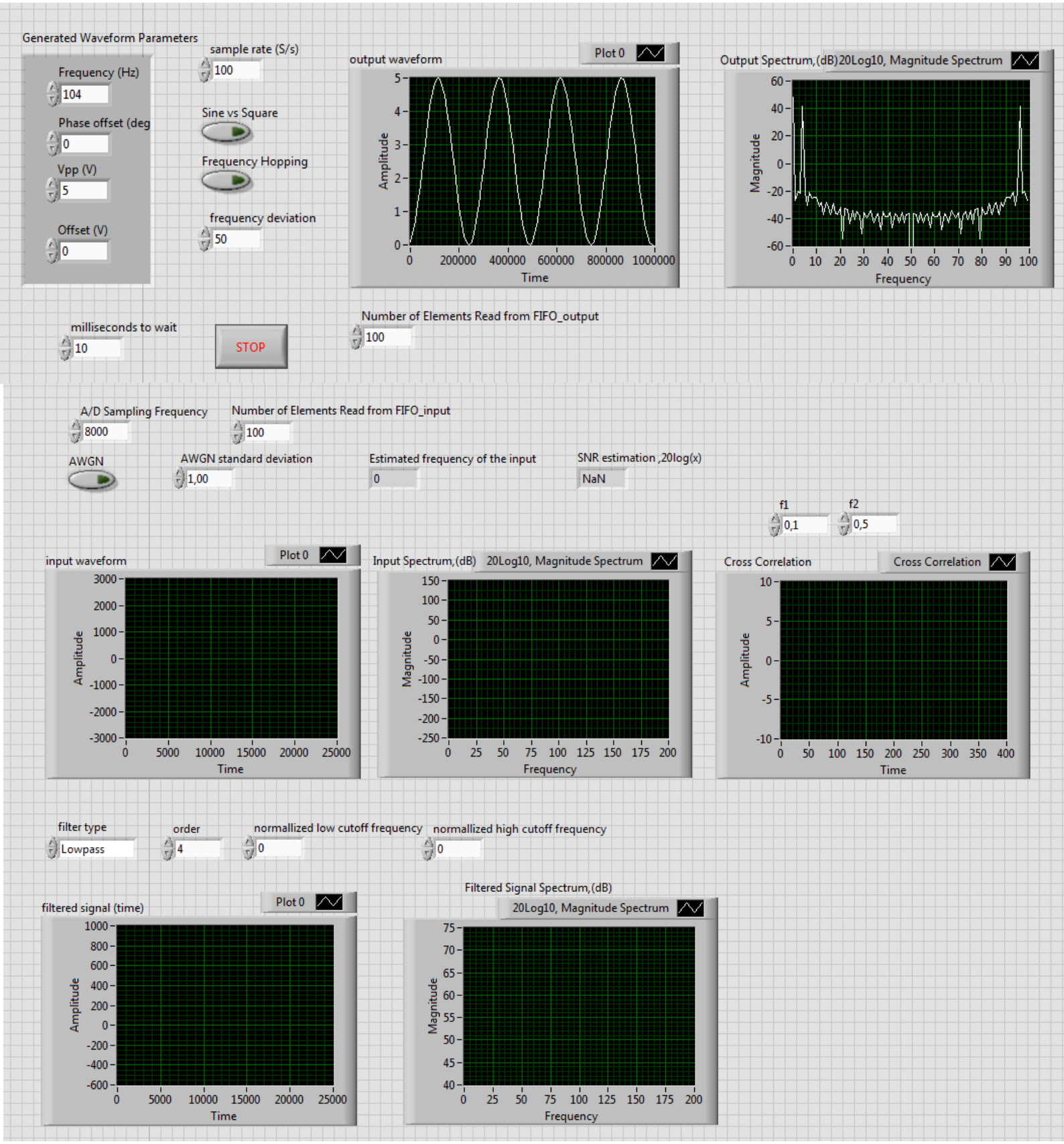

Fig.20. Complete Front Panel of CPU VI for Experiment 3.

Task 2

1. Generate a **sinusoid at 1000 Hz frequency** and **select the sampling rate as f_{s1}=16kHz for the D/A converter**. Note that **AWGN** button is **off**. **Estimate the frequency for the received signal after A/D converter while the sampling rate for A/D is selected as f_{s2}=16kHz.** Increase and decrease **f_{s1}** to **32kHz and 8kHz** respectively and note the **estimated frequency** of the received signal. Also **note the differences in time and frequency between each case**. Now select **f_{s1}=16kHz** and change **f_{s2} to 32kHz and 8 kHz** respectively. **Note the estimated frequency and differences in time and frequency.**

2. Select **f_{s1}= f_{s2}=16kHz**. Generate a **sinusoid at 1kHz**. **Add noise** to the generated signal by changing the **noise standard deviation from 0 to 1000 with an increase in 5 steps**. Note the **estimated frequency and SNR at each case**. Now **increase the noise standard deviation to a value** such that sinusoid frequency **cannot be estimated any longer**. **Note this value** and write in your report.

3. Continue **from 2** but select **noise standard deviation as 5**. Observe the **cross correlation of the input with the chirp**. Select the chirp **f1 and f2 frequencies as 0.01 and 0.4. Increase the value of f1 in 0.1 steps until 0.4. Comment on the changes in cross correlation. Attach** all the cross correlation plots.

4. **Repeat 2 using frequency hopping mode** where the **frequency deviation** is set as **50Hz**.

5. Generate a **square wave** with a **period** of **T=0.5ms**. Set **noise standard deviation** to **0**. Choose the **sampling rate as 16 kHz for the A/D converter**. Choose the **normalized low and high cutoff frequencies of Butterworth filter** as **0.2 and 0.4** respectively. **Plot** the **input and filtered signal waveforms** both **in time and in frequency**. Determine **the main spectrum width** for the square wave and **its relation to the period, T**. You can change the period to see its effect on the frequency spectrum. **Decrease T by 2 and increase it by 2** to identify the **time-frequency relation**. **Plot** the **input and filtered signal waveforms** both **in time and in frequency** for both cases. **Comment on** the results.

6. Set the frequency of the square wave to **3200 Hz**. **Plot the cross-correlation function** between **the input** and the **filtered output for the following cases**.

   i. **Normalized low and high cutoff frequencies of Butterworth filter** are **0.2 and 0.4** respectively.
   ii. **Normalized low and high cutoff frequencies of Butterworth filter** are **0.3 and 0.5** respectively.
   iii. **Normalized low and high cutoff frequencies of Butterworth filter** are **0.4 and 0.6** respectively.
   iv. **Normalized low and high cutoff frequencies of Butterworth filter** are **0.5 and 0.7** respectively.

   Comment on the results.

7. **(Bonus 20pts) Modulate the square wave generated in step 5 by a 6kHz sinusoid**. Select the **noise standard deviation as 50**. **Filter the resulting signal with a bandpass filter to remove noise as much as possible while keeping the signal characteristics**. **Determine the minimum bandwidth that you can use for this purpose**. **Demodulate** the signal down to baseband by **multiplying with another sinusoid** with the same frequency and **lowpass filtering** the resulting waveform. **Plot the input and demodulated signal time and frequency characteristics in the front panel.** Note that this structure is used to transmit and receive signals just like a standard communication system.