Project: **Time Division Multiplexing**

Student Name: **Abhishek Barla**

Student ID:  **900-55-9822**

Course: **ECE 4342 Virtual Instrumentation Lab**

Submission Date: **April 6, 2015**

Instructor: **Gregory Lovell**

**Purpose:**

The purpose of this project is to generate 3 different kinds of waveforms in the LabVIEW software and input 2 sound files and multiplex these 5 signals using time division multiplexing and de-multiplex the signals to retrieve the same waveforms again. In the second stage of the project, the 3 different waveforms are generated using circuits on the breadboard, which are then multiplexed using Time Division Multiplexing and then retrieved by de-multiplexing them.

**Components:**

* 1µF Capacitors
* 1kΩ Resistors

**Procedure:**

Software Implementation

As shown in Figure 1, the first step of this part of the project is to generate the 3 different

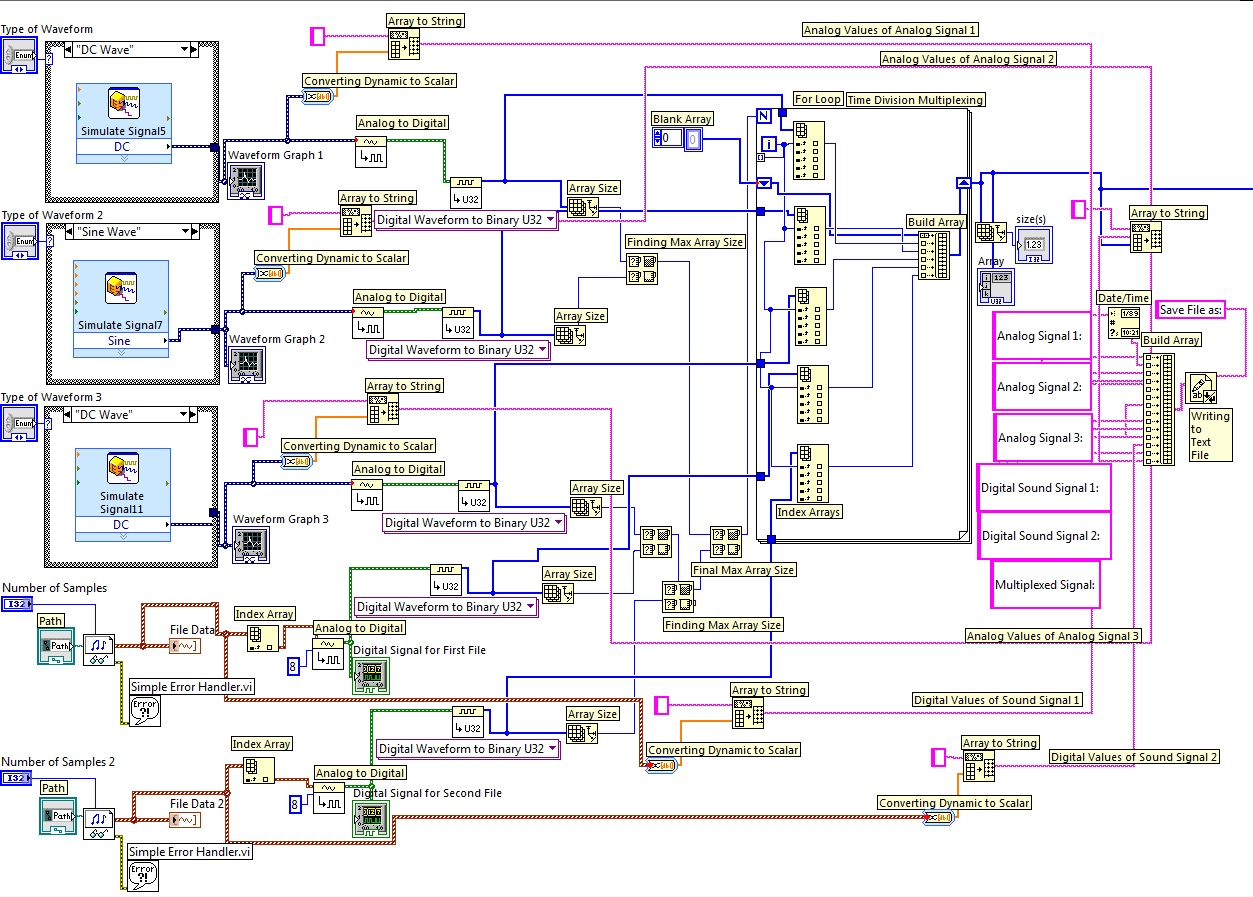


Figure 1: Multiplexing using the Time Division Multiplexing

Waveforms using the simulate Signal Express VI. The user is given the option of selecting from 5 different waveforms (Sine, Square, Triangular, Sawtooth, DC Wave) using the Enumerated List control on the Soft Front Panel (SFP). Next, for the user input sound files, the Sound File Read Simple VI is used on the Block Diagram. For the user to enter a path, a Path control is placed on the SFP. This way, the user can select the two sound files to be used. Now, these 5 signals are displayed using Waveform Graphs.

Next, these signals are converted to Digital using the Analog to Digital VI on the block diagram. The two sound signals are converted using 8 bits. Next, the signals are converted to Binary form using the Digital to Binary VI on the block diagram. Now, since the outputs of these signals are 1D arrays, the array sizes of each of them are found out using the Array Size element on the block diagram. Using the Max and Min element, the maximum array size of these 5 arrays. Next, a for-loop is used to build an array which contains the first element of the 5 arrays as the first 5 elements and the second elements of the 5 arrays as the next 5 elements and so on. The size of this new array is the Max array size found out previously.

Now, once the stream of elements from an array are placed in the new array, zeros are placed as place holders to fill their place in the new array. This way the new array is populated. To do this, in the for-loop, an Index Array element is placed to find each element of the 5 arrays. Then the Build Array element is used to build the new array. Elements of this new array are populated from the Index Arrays. Once, the population of the new array is called multiplexing using Time Division Multiplexing.

Once the Multiplexing is done, the next task is to save the contents of the new array to a text file. To do this, first each array is converted to a scalar form using a Dynamic to Scalar element on the block diagram. Next, this array is converted to String using the Array to String element on the block diagram. Next, these strings are used to build an output array using the Build Array element on the Block Diagram. The appropriate spacing elements are inputted using blank string to make the text file readable. The output array is then passed to the Write to Text File element on the block diagram. This completes the Multiplexing through the Time Division Multiplexing and writing the contents to a text file. Now, the output of the final build array in the previous section is the Multiplexed array.

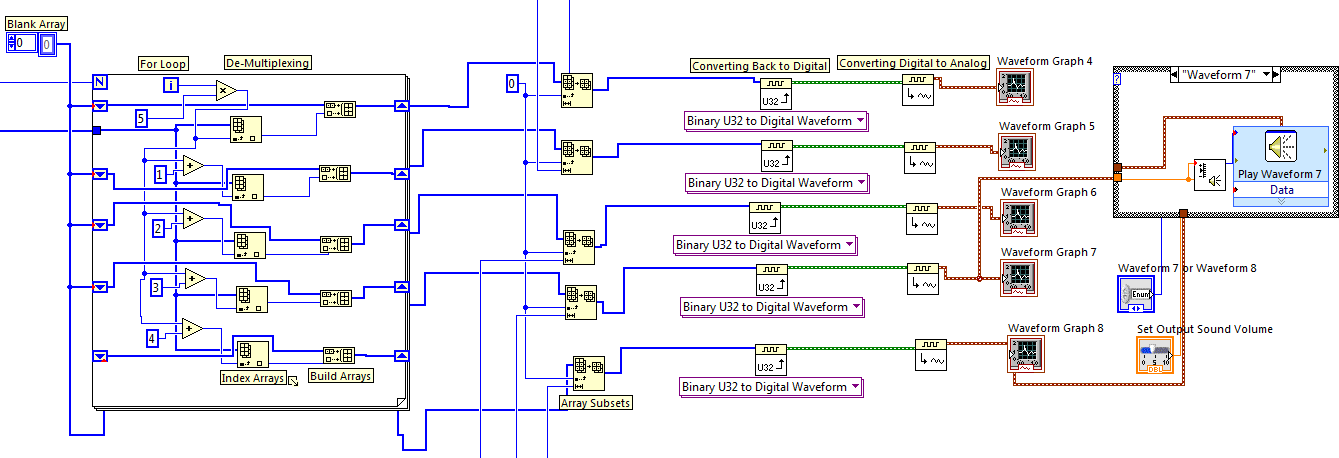


Figure 2: De-Multiplexing the multiplexed signal

This multiplexed signal is passed through a for-loop for de-multiplexing as shown in Figure 2. The number of iterations of the loop is the Max Array Size found out in the previous section. The loop contains Index Arrays, Build Arrays and Shift Registers to individualize the 5 arrays from the input array, according to the order in which they were multiplexed.

Now, as stated before, these individual arrays may contain zeros as all the arrays would be of size Max Array Size (as the loop runs those many times). To remove these zeros, the Array Subset element is used on the block diagram with the array sizes of each array to get arrays without zeros. Then these arrays are passed through a Binary to Digital Waveform Convertor on the Block Diagram to be converted into the Digital form. Once this is done, the digital signals are passed through the Digital to Analog Waveform Convertor element on the block diagram to convert them to Analog form. Next, these analog waveforms are displayed on the SFP using the Waveform Graphs as shown in Figure 2. Thus the 5 signals have been de-multiplexed from the single input array from the previous section.

Now, to output the sound file, an enumerated list control is placed on the SFP to select between the sound files. A switch case is used to switch between the two selected enumerated inputs. To set the volume of the output, an indicator is placed on the SFP which gives the user the option to set the sound volume. This indicator is connected to the Sound Output Set Volume VI element on the Block Diagram to set the volume. The signal is then connected to the Play Waveform Data element on the block diagram which plays the sound.

Hardware Implementation

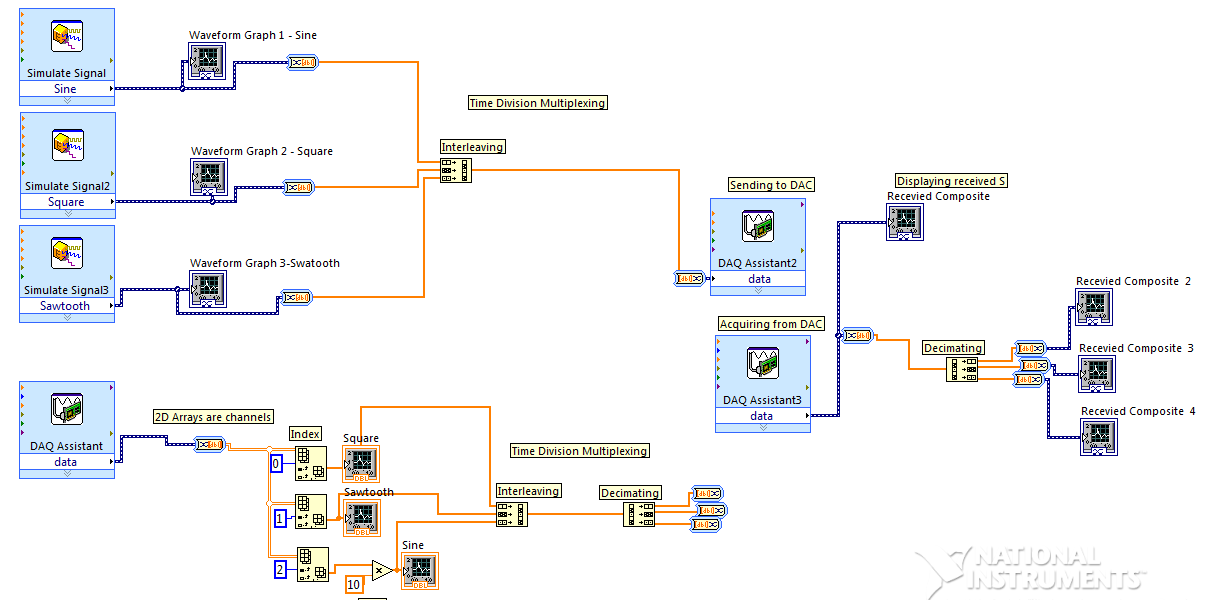


Figure 3: Figure showing the Hardware implementation of Time Division Multiplexing

The Hardware Implementation of Time Division Multiplexing is done as shown in Figure 3. Firstly, three signals are generated using the Simulate Signal element on the Block Diagram. These Signals are then graphed on the SFP using the Waveform Graph element on the Block Diagram. Then, the signals are converted to scalars using the Dynamic to Scalar Convertor on the Block Diagram. Now that the signals are 1D arrays, the three arrays are interleaved using the Interleaving Arrays element on the Block Diagram. Now, the arrays elements are placed alternately in the interleaved array. This array is converted to the Dynamic form using the Scalar to Dynamic Convertor on the Block Diagram. The signal is now multiplexed using Time Division Multiplexing.

Now, the signal is sent to the DAQ assistant to send it to the Bread board. This signal is retrieved from the board using another DAQ assistant. The received composite is displayed on the SFP using the Waveform Graph. Then, this signal is converted back to the Scalar form using the Dynamic to Scalar Convertor. The signal is now a 1D array. This 1D array is sent to a Decimated Array element to decimate the elements of the 3 arrays to separate them into different arrays. Then these separate arrays are then converted to the Dynamic form using the Scalar to Dynamic element on the Block Diagram. Then, these signals are displayed on the SFP using the Waveform Graph. Thus, the signal has been de-multiplexed and displayed on the SFP.

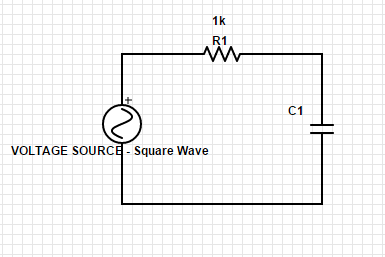
Now, to generate the same signals using the hardware, a square wave is generated using the Function Generator. On the Block Diagram, a DAQ assistant is used so that one of the channels is used to retrieve the voltage signals from the Function Generator. Thus, a square wave signal is generated. Now, to generate a Saw Tooth wave, the voltage from the Function Generator is passed across a Series RC Circuit with a resistor of 1kΩ and a capacitor of 1µF, as shown in Figure 4. This converts the Square Wave into a Saw tooth wave. The voltage from across the Capacitor C1 in Fig 4 is

Figure 4: RC Circuit to Convert Square Wave into Saw Tooth wave

the voltage that is measured by the DAQ assistant on the Block Diagram. Another channel is used to measure this voltage. This is the Saw tooth wave signal. Next, to generate a sine wave, the voltage from the Function Generator is passed in a 2nd order series RC Circuit as shown in Figure 5. The Resistors used here are of 1kΩ and the capacitors used here are of 1µF. The voltage across the Capacitor C2 is the voltage that is recorded using the DAQ assistant on the Block Diagram. Another channel is used to measure this voltage. This is the Sine wave signal generated from the hardware. Now, since the DAQ assistant outputs only dynamic stream of data, we should separate the data to get these three signals. To do this, the data is converted to a 2D array of rows where each row depicts a signal. This is done using a Dynamic to Scalar Convertor on the Block Diagram. There are three such rows now. Each row is separated using an Index Array element on the Block Diagram and then are displayed on the SFP using a Waveform Graph. Now, the 3 arrays are 1D arrays. Now, these 3 arrays are Interleaved using the Interleaving Arrays element on the Block Diagram. Now, the arrays elements are placed alternately in the interleaved array. The signal is now multiplexed using Time Division Multiplexing.

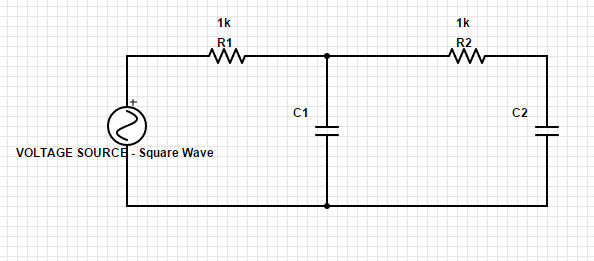


Figure 5: RC Circuit used to convert Square Wave into a Sine Wave

Now, that the signals are multiplexed, they are sent for de-multiplexing. This interleaved array is sent to a Decimated Array element to decimate the elements of the 3 arrays to separate them into different arrays. Then these separate arrays are then converted to the Dynamic form using the Scalar to Dynamic element on the Block Diagram. Then, these signals are displayed on the SFP using the Waveform Graph. Thus, the signal has been de-multiplexed and displayed on the SFP.

**Results:**

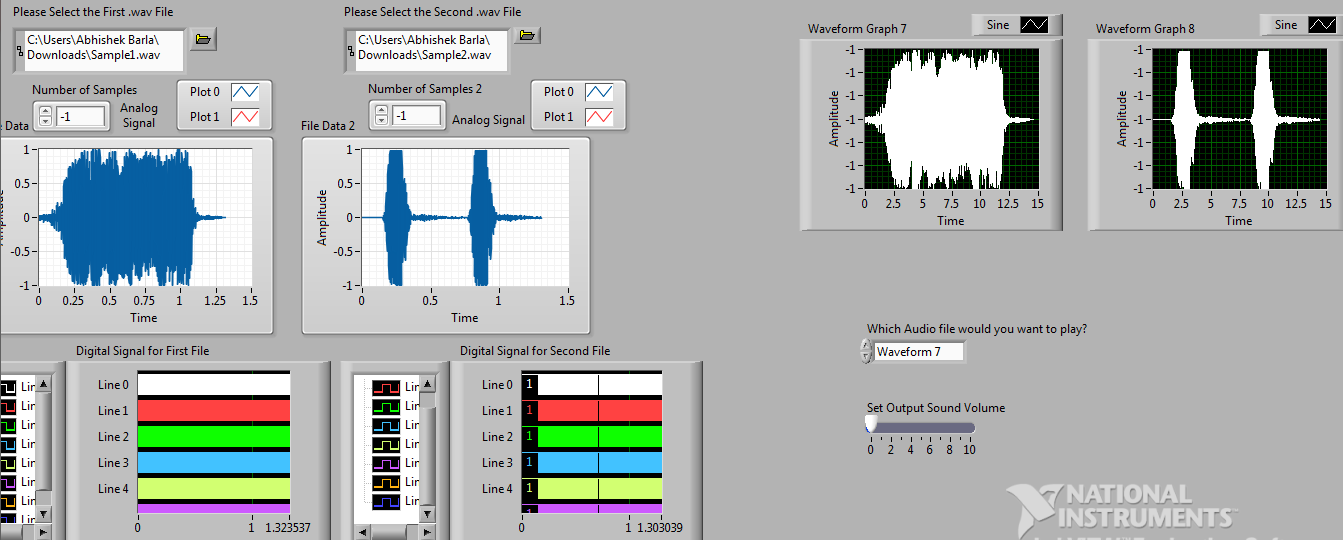


Figure 6: Figure showing the two sound signals before and after the multiplexing.

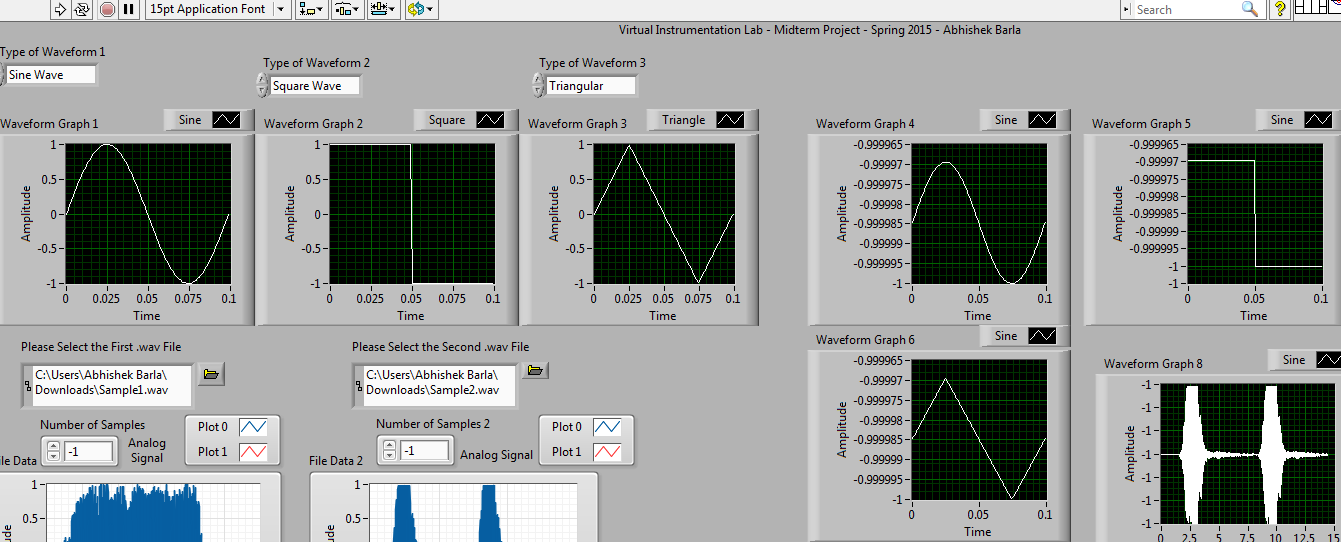


Figure 7: Figure showing the 3 waveforms before and after the multiplexing.

**Discussion:**

Time division multiplexing (TDM)1 is a communications process that transmits two or more streaming digital signals over a common channel. In TDM, incoming signals are divided into equal fixed-length time slots. After multiplexing, these signals are transmitted over a shared medium and reassembled into their original format after de-multiplexing. Time slot selection is directly proportional to overall system efficiency. Time division multiplexing (TDM) is also known as a digital circuit switched.

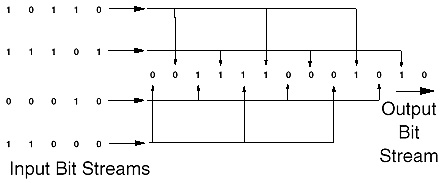


Figure 8: Time Division Multiplexing

The Time Division Multiplexing can be explained better using the Figure 8. Suppose that there were 4 bit streams as inputs which were to be multiplexed using Time Division Multiplexing and then transmitted and de-multiplexed into output bit streams. Let the four bits be 10110, 11101, 00010 and 11000. Then, the output bit stream would be 10100…. This is because the last element of the 2nd input bit stream is sent first, then the last element of the 3rd input bit stream

1Time Division Multiplexing – See Bibliography 1

is sent and then the last element of the 1st bit stream is sent and then finally the last element of the 4th input bit stream is sent. While retrieving the signal, the same is done backwards. Since each element is placed alternatively across the stream, it is easier to retrieve the stream while de-multiplexing the signal. This is Time Division Multiplexing.

**Recommendations:**

No Recommendations for this lab project.

**Conclusion:**

In conclusion, the lab project was conducted successfully and I learned a lot from this lab as I designed a fully functional VI on my own. This has been a great learning experience and I thank my GSA for helping us in learn how to design a VI. This lab also helped us think on our own and made us use some of the key concepts of Electrical Engineering learnt in the Circuits classes.

**Bibliography:**

1. Technopedia (<http://www.techopedia.com/definition/9669/time-division-multiplexing-tdm>)