

Thesis ID : IJERTTH0008

NI based Time Dissemination system over UHF



Ms. Pranita Behera

**Sikkim Manipal Institute of Technology
India.**

Published By

**International Journal of
Engineering Research and Technology
(www.ijert.org)**

Chapter 1

Introduction

1.1. Introduction

Reliable Time dissemination is a primary requirement for synchronization and event management of remotely located tracking sensors and distributed command control systems in a test range.

For dynamic testing of Flight Vehicle several predefined tasks are needed to be executed before the lift-off and subsequently after also. Depending on the complexity of the vehicle and the involved supporting systems, these timed jobs are sequenced in a manner to fulfill activities which are interlinked to each other.

For the purpose of proper management of these events a programmed time also referred as count down time (CDT) is generated in IRIG-B [1] format and disseminated to various remotely located tracking sensors and distributed command control systems in a test range through wired links via communication Multiplexers.

For time Synchronization of various distributed, remotely located sensor devices at very inaccessible locations like hills, sea or ill networked land sites there is a strong need for CDT dissemination over wireless telecommunication link as shown in Fig.1.1.

Now-a-days communication plays a vital role in transmitting and receiving of information through various channels. These channels may include underground or sea level. The main problem arises when we find difficulties in transmitting our desired signal in cables at and under the sea level. Providing cables for the larger distance is quite impossible and is too

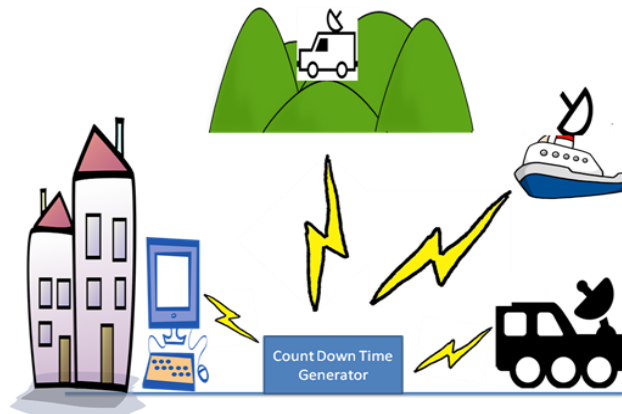


Fig.1.1 CDT dissemination over wireless telecommunication link

costly if proposed. Maintenance under the sea level at a greater depth is not possible. So reception of the signal becomes quite difficult as well. Thus it's necessary to set up a low cost receiver system that can easily receive and we can observe our required signal wirelessly.

1.2 Objective

The main objective of the project is to disseminate timing information (CDT) to base stations over RF channel and displaying one of its outputs on Raspberry Pi. The transmitter consists of time generator which generates information. It is further encoded with Manchester encoding which consists of 32 bit (1 frame) of data in terms of time. A data format consists of 3 bits of start bit, 28 bits of data bit and 1 bit of stop bit as shown in Fig.1.1 and the frame is sent at a stipulated time. The basic time format is shown in Fig. 1.2.

3 bits Start bit		28 bits Data bit	1 bit Stop bit
Hour (HH)	Minute (MM)	Second (SS)	Status (D/U/H)

Fig. 1.3 Time Format

The receiver consists of an antenna where the timing information is received. The received BFSK signal is further decoded and displayed on Raspberry Pi. Raspberry Pi as the receiver which is an ultra low cost credit card sized Linux computer which has low power and which do not require any Ethernet cable or USB ports.

The hidden objective is to get familiarized with the development in the NI environment. The NI platform involves working in the field of NI Flex RIO hardware and LabVIEW software. The NI Flex RIO hardware includes various chassis which performs various functions like embedded controller, Up-converter, Down-converter, various ports for serial and parallel communications, RF pre-amplifier which acts as low noise amplifier and baseband transceiver system. Secondly to get familiarized with the Raspberry Pi related embedded system. Raspberry Pi being a low cost receiver is easier to implement in inaccessible remote area where communication through cables are not possible. Embedded system which involves NI hardware and LabVIEW software interfaced with the Raspberry Pi board to display the timing information on R Pi display.

Chapter 2

Motivation

2.1 Motivation

The purpose is to serve a good communication facility to the areas where we provide cables to a larger distance are quite impossible and cost effective. Transmission of signal through cables over the sea level for a larger distance is not possible. So we need to implement a type of receiver where we can receive our desired timing information within a stipulated time wirelessly. This installation can be helpful if implemented in inaccessible remote areas.

While conducting missile test inside the deep sea, various sea platforms are used in ships, boats, floating platform, etc. It is not practicable to lay cables on the sea surfaces and disseminate any information from the mother ship to all platforms. The scenario, RF dissemination shall be the means to disseminate the information.

The main motivation aims at working in the field of FPGA which provides flexibility and rapid prototyping with low power consumption in an optimized area. It also provides reliability in hard-wired implementation. The receiver Raspberry Pi is a low cost credit card sized PC which provides raw data through the GPIO pins. It basically runs on Linux environment which provides a windowing system and text based interface for controlling of Pi.

Further interest to work in the area of embedded system led me to work in the area of NI based system and LabVIEW software.

Chapter 3

Literature Survey

3.1 Introduction to Timing Division

Timing division mainly comprises of two primary parts such as Count Down Time and Accumulated Time of Day. The CDT stands for Count Down Time comprises of some basic information for any event occurred within that particular time. The ATD stands for Accumulated Time of Day which indicates normal IST time. This includes year (YYYY), day (DD), hour (HH), minute (MM) and second (SS)

3.2 Frame format of Count Down Time

Hour (HH)	Minute (MM)	Second (SS)	Status (D/U/H)
-----------	-------------	-------------	----------------

Fig. 3.1 Time Frame format

The format for Count Down Time is as shown in Fig. 3.1. The format consists of Hour (HH), Minute (MM), and Second (SS). The event carried out in a particular event is performed within that Count Down Time.

3.3 LabVIEW

3.3.1 Introduction

LabVIEW stands for Laboratory Virtual Instrumentation Engineering Workbench. By the use of this software we code the program in terms of VIs (Virtual Instrumentations) and dump the program into the hardware for several applications [1]. There are basically three modules in LabVIEW:

- Basic module
- FPGA module
- Real time module

3.3.2 Programming Methods

Basic module in LabVIEW includes VIs for signal generation, signal processing, filtrations, windowing, etc. Basic LabVIEW consists of VI comprises of Front Panel and Block Diagram.

3.3.2.1 Front Panel: The front panel is the user interface of a VI. We build the front panel by using controls and indicators, which are the interactive input and output terminals of the VI, respectively. Controls and indicators are located on the Controls palette. Controls are knobs, push buttons, dials, and other input mechanisms. Controls simulate instrument input mechanisms and supply data to the block diagram of the VI. Indicators are graphs, LEDs, and other displays. Indicators simulate instrument output mechanisms and display data the block diagram acquires or generates.

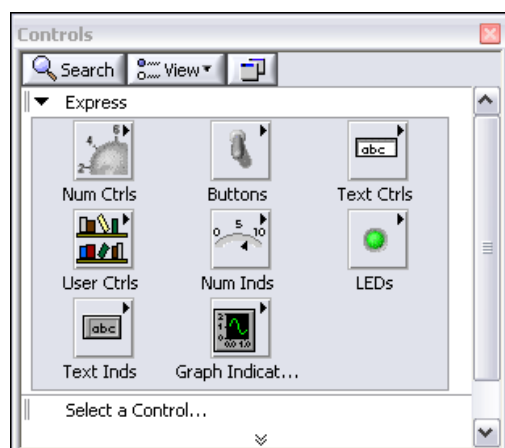


Fig. 3.2 Control Palette

3.3.2.2 Block Diagram: The block diagram contains the graphical source code, also known as G code or block diagram code, for how the VI runs. The block diagram code uses graphical representations of functions to control the front panel objects. Front panel objects appear as icon terminals on the block diagram. Wires connect control and indicator terminals to Express VIs, VIs, and functions. Data flows through the wires from controls to VIs and functions, from VIs and functions to other VIs and functions, and from VIs and functions to indicators. The movement of data through the nodes on the block diagram determines the execution order of the VIs and functions. This movement of data is known as dataflow programming.

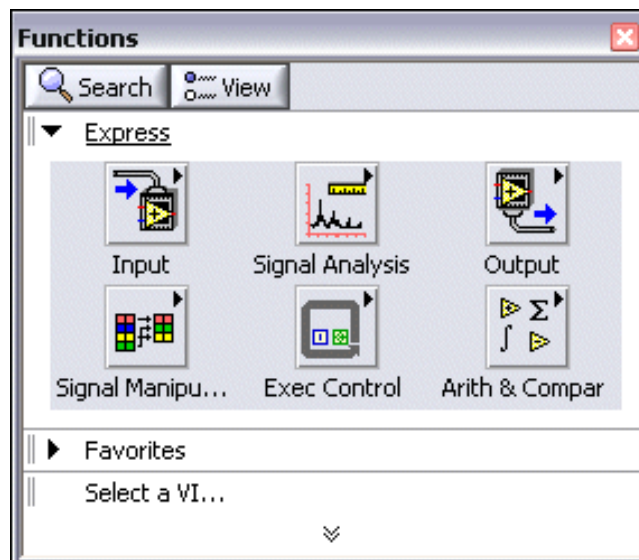


Fig. 3.3 Function Palette

3.3.2.3 Front Panel and Block Diagram Tools

The Positioning tool appears when we move the cursor over an object in the front panel window or on the block diagram. The cursor becomes an arrow that you can use to select, position, and resize objects. The Wiring tool appears when you move the cursor over a terminal of a block diagram object. The cursor becomes a spool that you can use to connect objects on the block diagram through which you want data to flow.

FPGA module includes two types of VIs such as LabVIEW FPGA VI and the Host VI. The LabVIEW FPGA VI executes on the FPGA of the hardware and Host VI executes on the host PC, controls the FPGA and provides user interface to the system [2]. The real-time module includes deterministic floating point processing and control algorithm for the ongoing process.

Basic components in target consist of LabVIEW FPGA module, the NI-RIO Driver, the NI-RIO Device (RIO Target).

3.4 Introduction to Raspberry Pi and Python

3.4.1 Introduction to Raspberry Pi

The Raspberry Pi has a HDMI (high-definition multimedia interface) video output, but most monitors have VGA or DVI input. If at all possible, we use a monitor that has DVI or HDMI input. A HDMI-to-DVI converter should cost only a few pounds/dollars and shouldn't detract from the image quality. HDMI-to-VGA converters are available, but they're more expensive and can cause problems, so use them only if you have no other option [5].

Most micro USB power supplies from reputable manufacturers should work; however, some cheap ones from no-name companies have caused problems. We could use a USB cable from a normal computer to

power your Pi [6]. Powered USB hubs are recommended for the power-related problems. Not all USB hubs are powered, so make sure that whatever one you get plugs into the mains electricity to get extra power [7].

To set up a raspberry pi, we need some equipment such as;

- i. Raspberry Pi
- ii. USB keyboard
- iii. USB mouse
- iv. SD card
- v. Monitor
- vi. Power supply

And some of the optional equipments as;

- i. USB
- ii. Wi-Fi dongle
- iii. Powered USB hub (highly recommended)
- iv. Camera module
- v. USB webcam

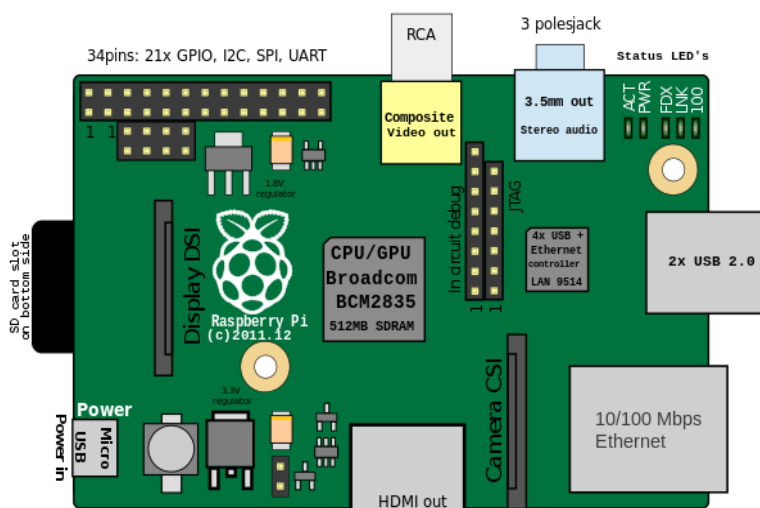


Fig. 3.4 Raspberry Pi

3.4.2 Introduction to Python

Python is a widely used high level, general purpose, interpreted, dynamic programming language. Python interpreters are available for many operating systems, allowing Python code to run a wide variety of systems. There are two ways of using Python, from the shell and saved programs. The shell executes each instruction as you type it, which means it's a really good way of trying out things and doing experiments. Saved programs are bits of Python code that are saved in a text file and run all at once [8]. There are two different ways we can write programs for Python. We can create text files that contain the code, and then we run these files with Python, or we can use an Integrated Development Environment (IDE) such as IDLE 3.

Chapter 4

Problem Discussion and Proposed Solution

4.1 Problem Discussion

Now-a-days communication plays a vital role in transmitting and receiving of information through various channels. These channels may also include over the sea level. The main problem arises when we find difficulties in transmitting our desired signal through cables at inaccessible remote areas. And providing cables for the larger distance is quite impossible and is too costly if proposed. Maintenance for the same could rather be not possible. Reception of the signal becomes quite difficult as well. Thus it's necessary to set up a low cost receiver system at those inaccessible remote areas which can easily receive the information and we can observe our required signal wirelessly.

4.2 Proposed Solution

The problem can be solved by setting up a low cost receiver at any inaccessible remote places to receive our desired information. The set up can use directional or the Omni-directional antenna for the purpose of reception of timing information. The process involves transmission from baseband to ultra high frequency band to transmit our information. The receiver includes reception from ultra high frequency band to baseband. Thus the receiver will decode the information and we can obtain the CDT on Raspberry Pi. The system performs at NI embedded platform and Raspberry Pi kit.

The methodology includes incoming of Count Down Time by a Time code generator. The timing information is encoded by Manchester Encoding. The encoded information is then modulated by BFSK modulation. The signal is then digitally and analogically up-converted at NI platform and is transmitted in UHF band. The receiver receives the signal and the signal is pre-amplified by RF pre-amplifier which acts as a low noise amplifier. The signal is analogically and digitally down-converted by the NI hardware. The decoding of BFSK is carried out followed by Manchester decoding. Finally the timing data is displayed on the Raspberry Pi screen.

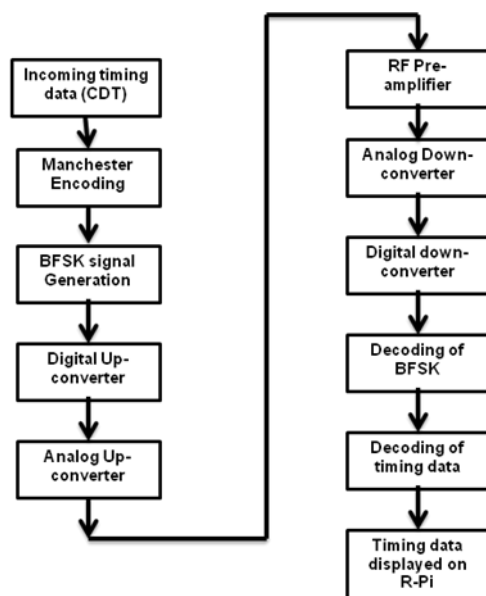


Fig. 4.1 Methodology for Time Dissemination System

Chapter 5

Design and development of NI based Transmitter system for Timing dissemination System

5.1 Introduction

The system was designed and developed on NI platform. NI hardware consists of various chassis which performs some basic functions as controlling of embedded system, Up-conversion and Down-conversion, connector ports for both serial and parallel communications, base band transceiver system which consists of I/O modules and a RF pre-amplifier which works as low noise amplification. NI PXIe-1075 (Fig. 5.1) consists of various modules as NI PXIe-8135 (Embedded Controller), NI PXIe-8431/8 (Connector)-RS 485/RS 422, NI PXI-5690 (RF Pre-amplifier), NI PXI-5600 (RF Down converter), NI PXI-5781 (Baseband Transceiver), NI PXI-5610 (RF Up converter), and NI PXIe-8430/8 (Connector)-RS 232. The brief description is described below:



Fig. 5.1 NI PXIe-1075

5.1.1 NI PXIe-8135 (Embedded Controller)

NI PXIe-8135 is a high performance and integrates on Intel Core i7 processor-based embedded controller. It has a base frequency of 2.3 GHz and single core turbo boost as 3.3 GHz frequency which is ideal for processor-intensive, modular instrumentation and data acquisition applications.

Requirement and compatibility includes OS information working on the platform of Windows 7 and Windows XP. The system bandwidth extends upto 8GB/s and slot bandwidth extends upto 4GB/s. It also consists of GPIB controller which provides control to the external instrumentation.



5.1.2 NI PXIe-8431/8 (Connector) and NI PXIe-8430/8 (Connector)

Table no. 5.1 Specification of NI PXIe-8431/8 (Connector) and NI PXIe-8430/8 (Connector)

Features	RS-232	RS-424	RS-485
Transmission Lines	Single-ended	Differential	Differential
Number of Drivers	1	1	32
Numbers of Receivers	1	10	32
Driver Output	5 to 25 V	2 to 6 V	1.5 to 6 V
Driver Load	Less than 3 k Ω	100 Ω	60 Ω



5.1.3 NI PXI-5690 (RF Pre-amplifier)

NI PXI-5690 (RF Pre-amplifier) works at the operating frequency of 100 kHz to 3 GHz and operating system on Windows 2000/XP with the recommended software as LabVIEW, LabWindows/CVI. It includes driver software as NI 5690.

5.1.4 NI PXI-5600 (RF Down converter)

NI PXI-5600 (RF Down converter) working at operating frequency of 5 MHz to 25 MHz is a modular, broadband down converter capable of RF measurement in

PXI module. It provides 20 MHz as real-time bandwidth and 10 MHz as the high stability time base oscillation. It generally works on the platform of Windows 2000/XP with recommended software as LabVIEW or LabWindow and including driver software as NI-TUNER. The noise and distortion are stable and repeatable over a wide range of time and temperature.



5.1.5 NI PXI-5781 (Baseband Transceiver)

The NI 5781 is an analog dual-input, dual-output NI FlexRIO adapter module optimized for interfacing with baseband to RF up converters and down converters. RF signals can be measured, generated, modulated, demodulated, filtered, decoded, and more. The NI 5781 is useful in frequency-based control. The NI 5781 samples and generates at 100 MS/s, or with a period of 10 ns, pipelining on the analog-to-digital converter (ADC), DAC, and FPGA, along with the algorithm running on the FPGA, introduce latency.



Table no. 5.2 Specification of analog I/O

Specification	Analog Input	Analog Output
Sample Rate	100 MS/s	100 MS/s
Resolution	14 bits	16 bits
Number of channels	2	2
Bandwidth (-3dB)	40 MHz	40 MHz
Range (Differential)	2 V _{pp}	2 V _{pp}
Coupling	DC	DC
Impedance	50 Ohm	50 ohm

5.1.6 NI PXI-5610 (RF Up converter)

It works on the operating frequency of 250 kHz to 2.7GHz with a real time bandwidth of 20MHz. it provides an adjustable gain of 130 dB. It also provides high stability time base oscillation with 10MHz.



5.2 FPGA in LabVIEW

5.2.1 Working of LabVIEW FPGA

The working principle (Fig. 5.3) states that the LabVIEW VI consists of graphical code which is translated to VHDL code. The Xilinx ISE Compiler compiles the VHDL code into the hardware circuit realization. The end result is the bit stream that contains information which in turn is downloaded to the FPGA target. When the application is run the bit stream is loaded into the FPGA chip.

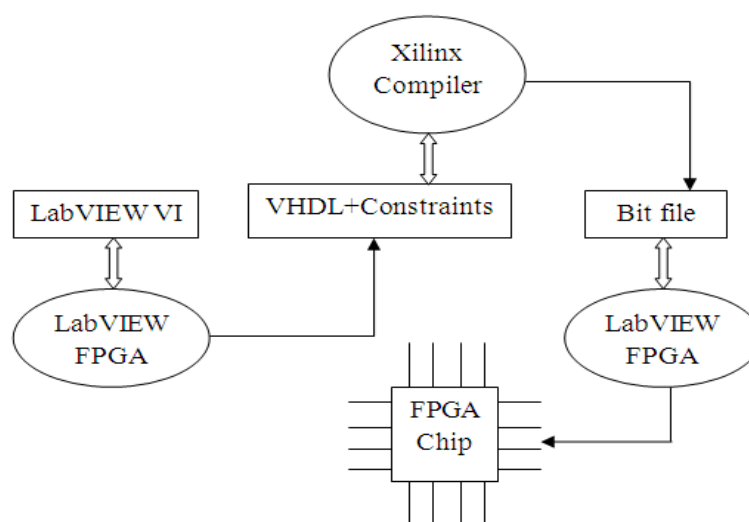


Fig. 5.2 Working of LabVIEW FPGA

5.2.2 DDS in LabVIEW

The integration of a D/A converter and a DDS onto a single chip is commonly known as a complete DDS solution, a property common to all DDS devices from ADI. DDS (Direct Digital Synthesizer) Compiler core sources sinusoidal waveforms for use in many applications. A DDS consists of a Phase Generator and a SIN/COS Lookup Table. Phase Generator and SIN/COS Lookup Table can be generated independently or combined together with optional Dither circuit to provide complete DDS solution.

Direct digital synthesizers (DDS), or numerically controlled oscillators (NCO), are important components in many digital communication systems. Quadrature synthesizers are used for constructing digital down and up converters, demodulators, and implementing various types of modulation schemes, including PSK (phase shift keying), FSK (frequency shift keying), and MSK (minimum shift keying). A common method for digitally generating a complex or real valued sinusoid employs a lookup table scheme. The lookup table stores samples of a sinusoid.

A digital integrator is used to generate a suitable phase argument that is mapped by the lookup table to the desired output waveform. A simple user interface accepts system-level parameters such as the desired output frequency and spurs suppression of the generated waveforms.

Direct digital synthesizers use an addressing scheme with an appropriate lookup table to form samples of an arbitrary frequency sinusoid. If an analog output is required, the DDS presents these samples to a digital-to-analog converter (DAC) and a low-pass filter to obtain an analog waveform with the specific frequency structure. The samples are also commonly used directly in the digital domain. The lookup table traditionally stores uniformly spaced samples of a cosine and a sine wave.

5.2.2.1 Core Architecture Overview

The core consists of two main parts, a Phase Generator and SIN/COS LUT, which can be used independently or together with an optional dither generator to create a DDS capability. A time-division multi-channel capability is supported, with independently configurable phase increment and offset parameters. Fig. 5.3 provides a block diagram of the DDS Compiler core.

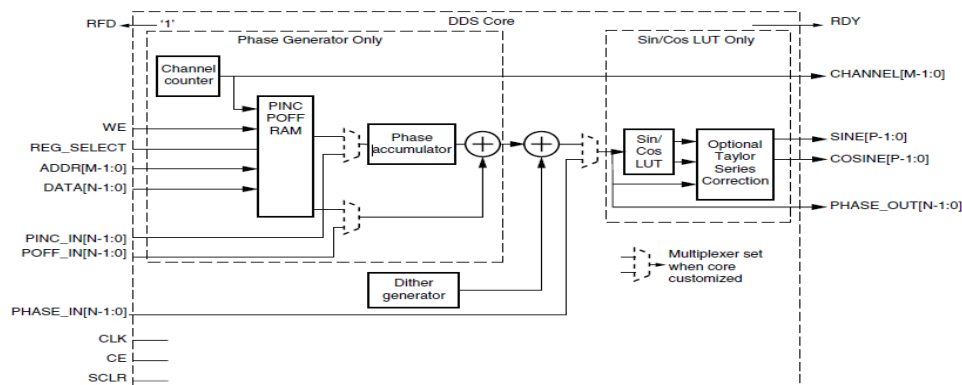


Fig. 5.3 DDS Core Architecture

5.2.2.2 Phase Generator

The Phase Generator consists of an accumulator followed by an optional adder to provide addition of phase offset. When the core is customized the phase increment and offset can be independently configured to be either fixed, programmable or supplied by the `PINC_IN` and `POFF_IN` input ports respectively.

When set to programmable, registers are implemented with a bus interface, consisting of `ADDR`, `REG_SELECT`, `WE`, and `DATA` signals. The address input, `ADDR`, specifies the channel for which `DATA` is to be written when multi-channel, with `REG_SELECT` specifying whether `DATA` is phase increment or offset. When set to fix the DDS output frequency is set when the core is customized and cannot be adjusted once the core is embedded in a design.

5.2.2.3 SIN/COS LUT

When configured as a SIN/COS LUT, the Phase Generator is not implemented, and the phase is input via the PHASE_IN port, and transformed into the sine and cosine outputs using a look-up table. Efficient memory usage is achieved using half wave and quarter wave storage schemes. The presence of both outputs and their negation are configurable when the core is customized. Precision can be increased using optional Taylor Series Correction. This exploits XtremeDSP slices on FPGA families that support them to achieve high SFDR with high speed operation. The generated DDS compiler and spectrum is as shown in Fig. 5.4 and Fig. 5.5.

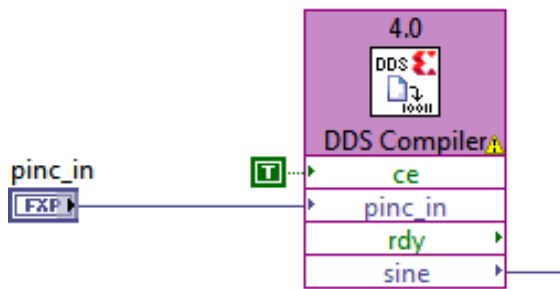


Fig. 5.4 Generated DDS compiler

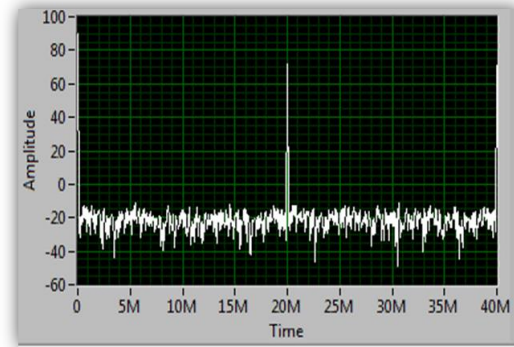


Fig. 5.5 Spectrum of 20MHz sine wave

5.2.3 FIR Compiler

FIR Compiler core provides a common interface for users to generate highly parameterizable, area-efficient high-performance FIR filters utilizing either Multiply-Accumulate (MAC) or Distributed Arithmetic (DA) architectures. A wide range of filter types can be implemented in the Xilinx CORE Generator: single-rate, half-band, Hilbert transform and interpolated filters, in addition to multi-rate filters such as polyphase decimators and interpolators and half-band decimators and interpolators. Structure in the coefficient set is exploited to produce area-efficient FPGA implementations. Sufficient arithmetic precision is employed in the internal data path to avoid the possibility of overflow.

Fig 5.6 illustrates the conventional tapped delay line realization of this inner-product calculation, and although the illustration is a useful conceptualization of the computation performed by the core, the actual FPGA realization is quite different. Where a MAC realization is selected, one or more time-shared multiply-accumulate (MAC) functional units are used to service the N sum-of-product calculations in the filter. The core automatically determines the minimum number of MAC engines required to meet user-specified throughput. Where a Distributed Arithmetic (DA) realization is selected, no explicit multipliers are employed in the design; only look-up tables (LUTs), shift registers, and a scaling accumulator are required.

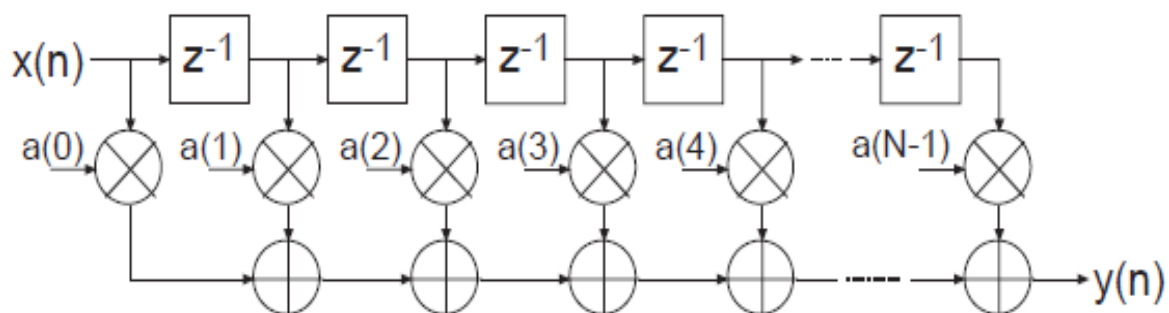


Fig. 5.6 Conventional Tapped Delay Line FIR Filter Representation

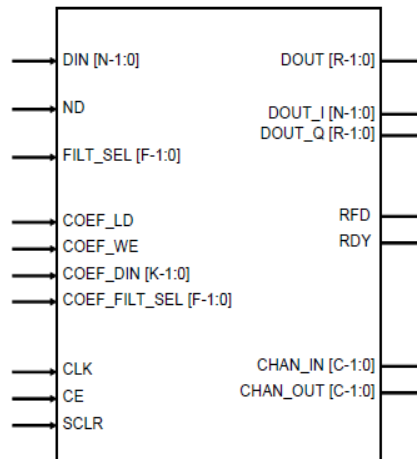


Fig. 5.7 FIR Compiler Core Pin out

Filter input data is supplied on the DIN port (N bits wide), and filter output samples are presented on the DOUT port (R bits wide). The maximum output width R is the sum of the data bit width N and the bit growth of the filter; see the Output Width and Bit Growth section for more details. The output width may also be reduced further under user control by truncation or rounding. The CLK signal is the system clock for the core, where the clock rate may be greater than or equal to the input signal sample frequency. The ND, RDY, and RFD signals are filter interface/control signals that permit a simple and efficient data-flow style interface for supplying input samples and reading output samples from the filter. These core interface signals are detailed in "Interface, Control, and Timing."

The Freq. Response tab (Fig 5.8), the default tab when the CORE Generator is started, displays the filter frequency response (magnitude only). The content of the tab can be adjusted to fit the entire window or undocked (as shown) into a separate window.

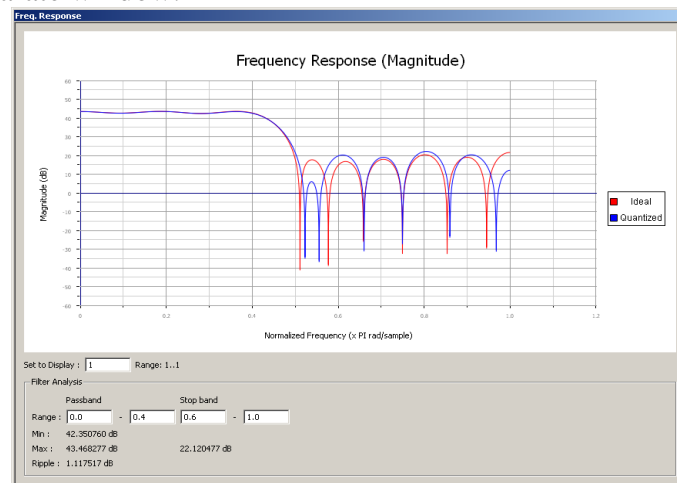


Fig. 5.8 Freq. Response Tab

5.3 Transmitter Design

5.3.1 Introduction

The transmitter design includes incoming of Timing data. The data is encoded by Manchester Encoding and it generates BFSK signal. Then the corresponding signal is digitally up-converted and later in analog form which was carried out in Up-converter of the NI system. After the process of up-conversion the signal was transmitted in UHF as shown in Fig. 5.9

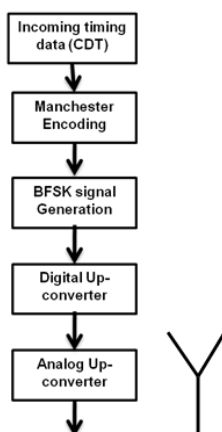


Fig. 5.9 Transmitter Design

5.3.2 Manchester Encoding

The Manchester encoding is carried out by the LabVIEW software. The coding is performed on the State Machine Concept where a single cycle timed-loop running at 5MHz generates 3 bits of Start bit, 28 bits of Stop bit and 1 bit of Stop bit as shown in Fig. 5.10. Then generated bit is passed through DDS compiler by selecting high and low frequency and data is sent through FIFO. Thus generating BFSK signal.

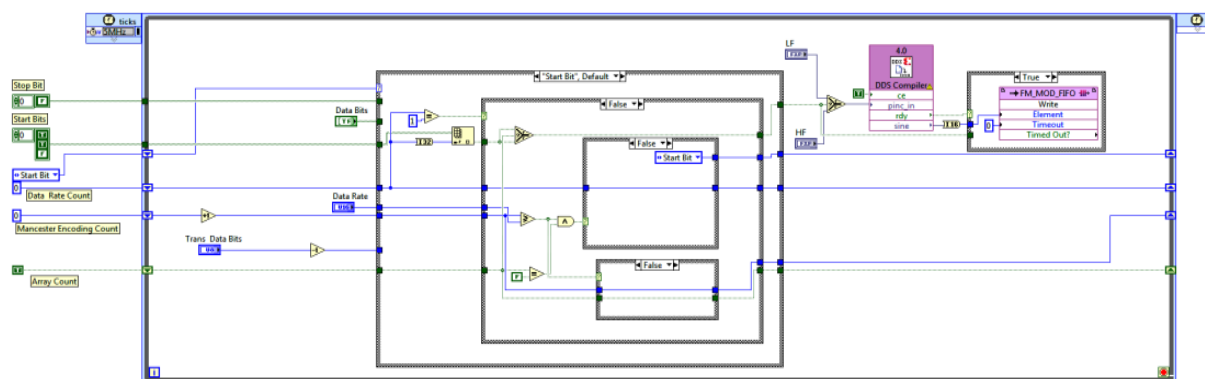


Fig. 5.10 Manchester Encoding

Three bits of start bits are stored in a Start Bit. Similarly data bits and stop bits are stored in Data Bit and Stop Bit respectively. Here Start Bit and Stop Bit are fixed, but Data Bit. These are varying according to the command code which is set at configuration settings. Binary 1 and 0 are taken as Boolean True and False respectively. One time-loop is taken which runs at 5MHz clock rate. Each iteration of the time-loop, True/False is being generated. After Manchester encoding, a 2Kbps data is changed to 4 Kbps data. The corresponding bit duration of this data rate is 250 microsecond. For a 5MHz clock rate total 1250 number Boolean True or False are generated during one bit duration as shown in Fig. 5.11, Fig. 5.12, Fig. 5.13.

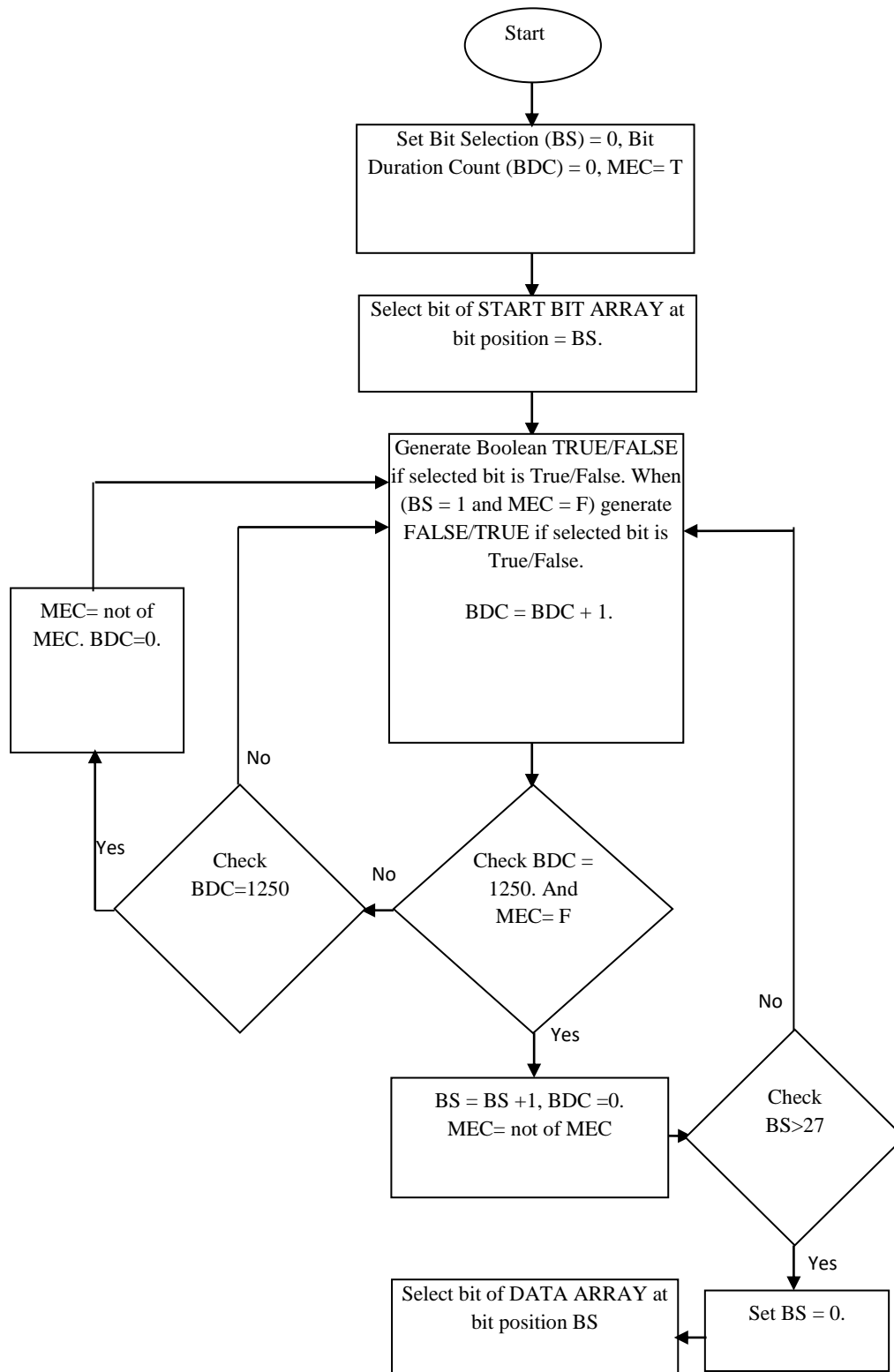


Fig. 5.11 Flowchart for Start bit generation

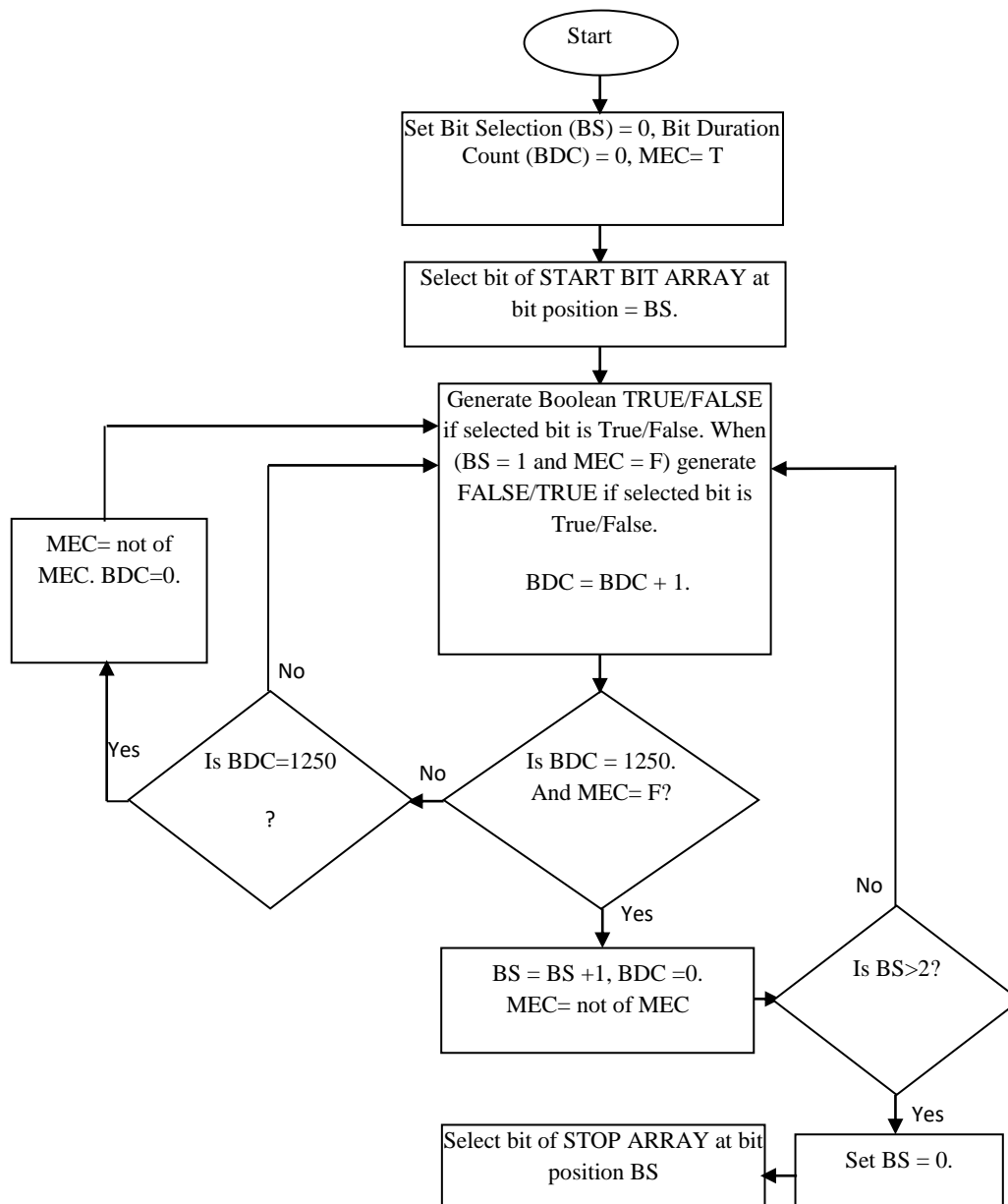


Fig. 5.12 Flowchart for Data bit generation

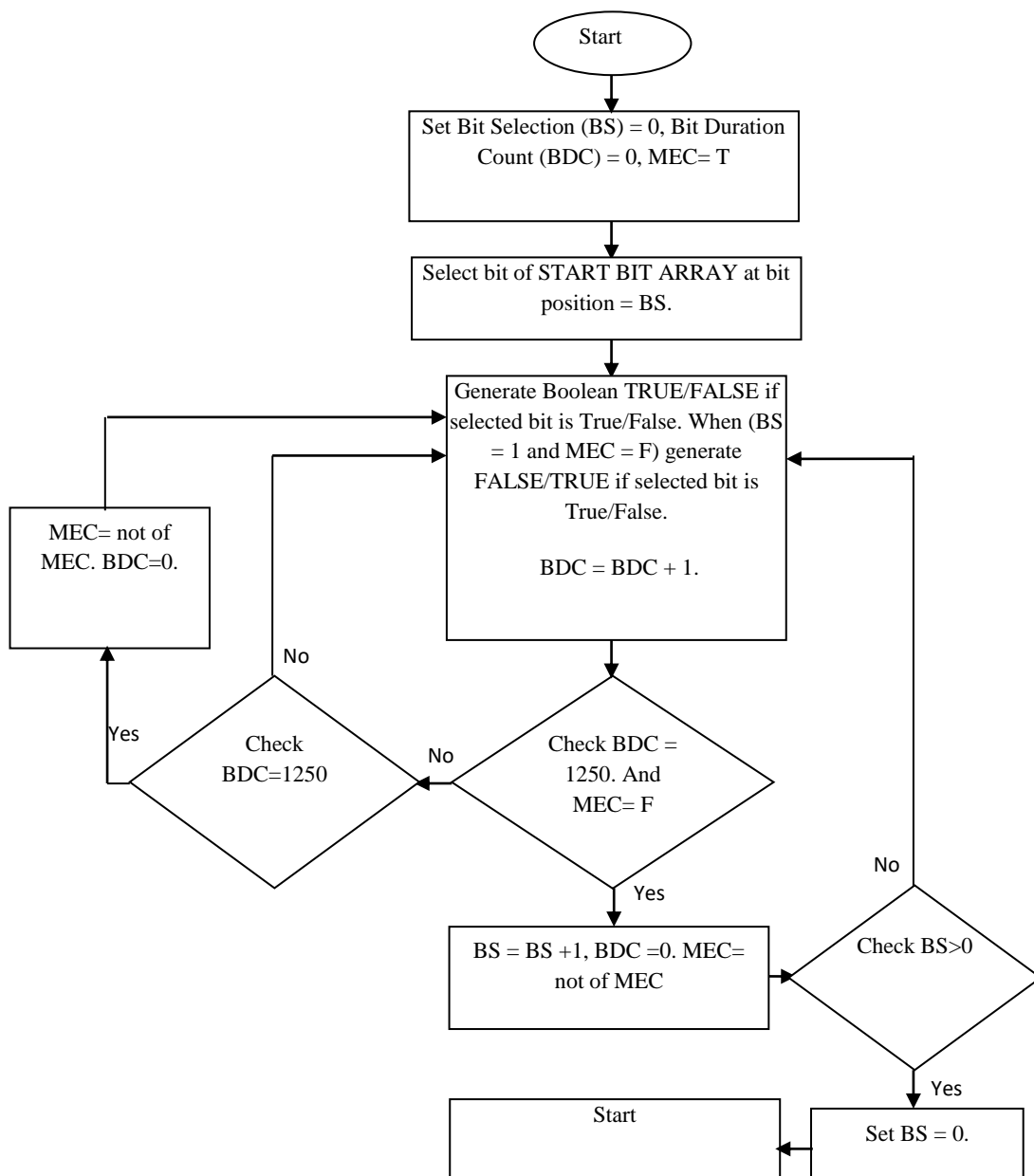


Fig. 5.13 Flowchart for Stop bit generation

5.3.3 BFSK Modulation

The BFSK data is performed after the Manchester encoding inside the same single cycled timed loop running at 5MHz clock frequency. After encoding process the BFSK signal is generated by using DDS compiler for generating higher frequency and lower frequency using two controls of fixed point at pinc_in (phase increment in). The higher and lower frequency is selected accordingly by using a selector option. Thus the generated BFSK signal is transferred to the Host by using a case structured loop by providing Target to Host FIFO.

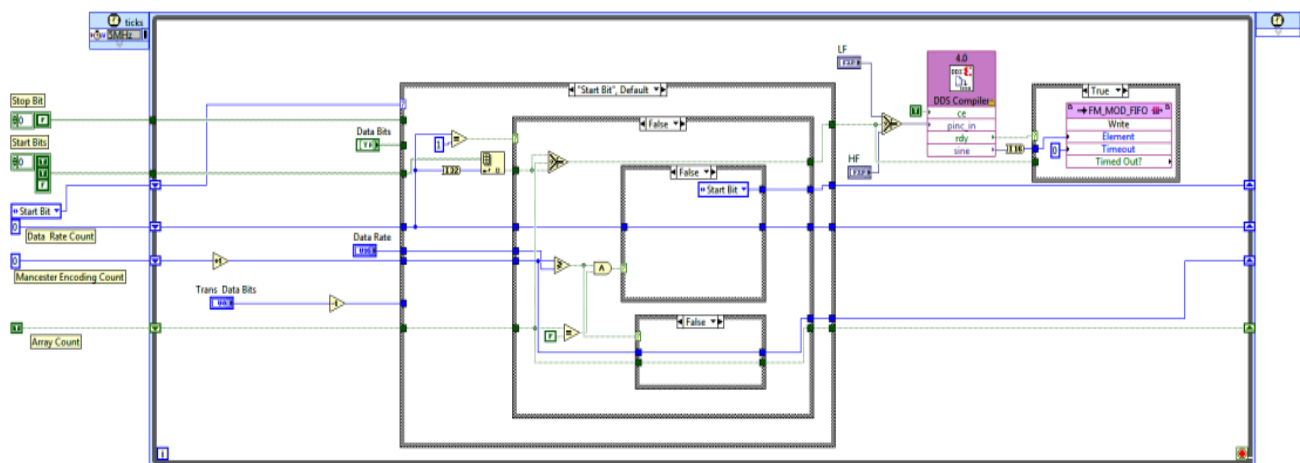


Fig. 5.14 BFSK Modulation

5.4 Baseband to UHF Up-conversion

An Up-Converter (UC) is a key component of front-end circuits for RF systems in communications, sensing, and imaging. The function of the UC is to translate one or more channels of data from baseband to a passband signal comprising modulated carriers at a set of one or more specified radio or intermediate frequencies (RF or IF).

A modulator feeds into a set of filters for pulse-shaping and interpolation, and the filter output is then mixed with a vector of one or more carrier frequencies. Further filtering, RF processing and frequency translation may be performed. Finally, the up-converted signal is converted to an analog signal for further processing and up-conversion to the RF band as in Fig. 5.15.

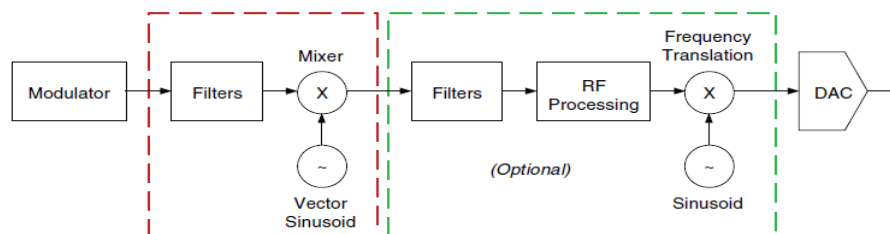


Fig. 5.15 Up-conversion

The transmitter design involved design and development of transmitter system for disseminating timing information (CDT) based on NI hardware. The whole process involves encoding of timing information followed by BFSK modulation. The generated BFSK signal is obtained on the Front Panel of the LabVIEW software. The input was provided in terms of Boolean array (True/False). The waveform as shown in Fig. 5.16 indicates the generated BFSK signal by the LabVIEW software.

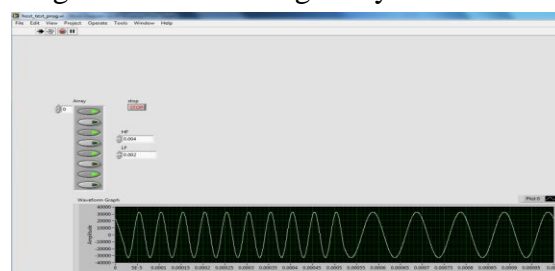


Fig. 5.16 Generation of BFSK signal by LabVIEW software

Chapter 6

Design and development of Raspberry Pi based Timing receiver system

6.1 Introduction to Raspberry Pi

There are a few things that make the Raspberry Pi a great device on which to learn programming. Firstly it's cheap. At around a tenth of the price of a low-end PC, it's cheap enough to have in addition to our main computer. This is useful because programmers tend to tinker with their development machine, and tinkering can break things. Generally this doesn't damage the machine itself, but it can require reinstalling the system, which can mean a bit of lost data, and it can put the machine out of action for a few hours.

Secondly, the Pi is raw. We have access to GPIOs that many machines don't have. Most computers come pre-packaged for a particular purpose (a tablet for surfing the web or playing games, a games console for watching movies or playing games, a laptop for working or playing games, and so on).

Thirdly, the Raspberry Pi runs Linux. This is an operating system a bit like Windows or Mac OS X. It provides a windowing system and a text-based interface for controlling the Pi.

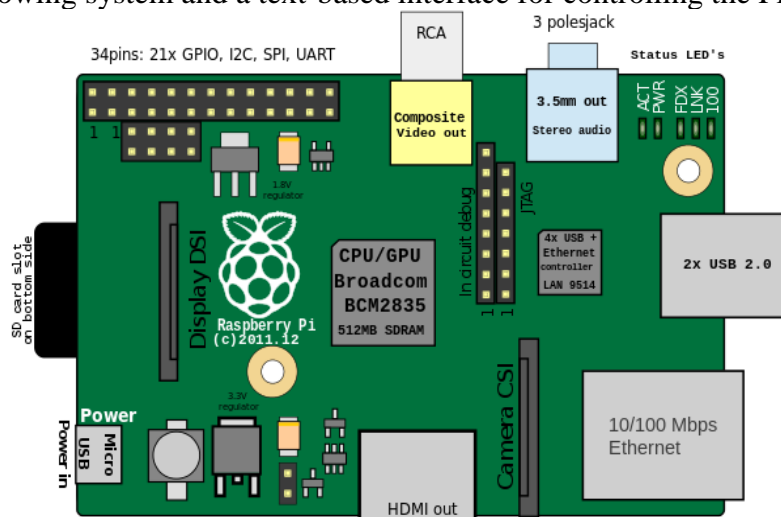


Fig. 6.1 Raspberry Pi board

6.1.1 Solving Problems in R Pi

The most common problems with the Raspberry Pi are power-related issues. Not all micro USB power sources can provide enough power, and it becomes more of a problem as we connect peripherals to your Pi, or when we over clock it. Power related problems will usually manifest themselves as the computer crashing. A good way to get around such issues is to connect your Pi to one power source and connect all the peripherals (keyboard, mouse, and so on) via a powered USB hub.

The second most common cause of problems with Pi is the SD card. These issues can be caused by power supply problems, or they can be problems with the cards themselves. It's important to take preventative measures here to ensure that your data is safe, and that means backups. We can use a service such as Google Drive (although this runs slowly on the Pi), or we can simply keep extra copies of any work on a USB memory stick. SD card issues will usually manifest themselves by the Pi displaying error messages when we try to start it. Most of the time we can solve the problem by reinstalling Raspbian, but if this doesn't work, we will need to get a new SD card.

6.1.2 Using LXDE (Lightweight X11 Desktop Environment)

The Lightweight X11 Desktop Environment is the standard windowing system for Raspbian. Its basic setup is the same as most versions of Windows pre-Windows 8. There is a button in the bottom-left side of the screen that opens an applications menu, and currently running applications are displayed in the bar along the bottom.

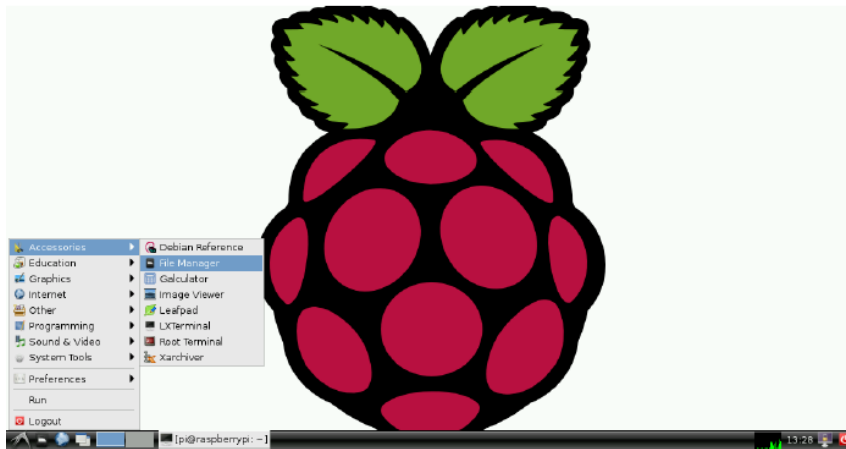


Fig. 6.2 LXDE desktop with menu open

6.1.3 The Python Interpreter

There are two ways of using Python, from the shell and saved programs. The **shell** executes each instruction as you type it, which means it's a really good way of trying out things and doing experiments. **Saved programs** are bits of Python code that are saved in a text file and run all at once. It's easy to tell which environment you're in because in the shell, all the lines will start with three chevrons:

```
>>>
```

6.1.4 Running Python Programs

There are two different ways we can write programs for Python. We can create text files that contain the code, and then we run these files with Python, or we can use an Integrated Development Environment (IDE) such as IDLE 3. If we want to write the programs as text files, we need to use a text editor such as Leafpad. A word processor such as LibreOffice's Writer is unsuitable because the various formatting it uses will confuse Python. As an example, we open Leafpad and create a new file that just has the line:

```
print("Hello World!")
```

Once we have created the file, we save it with the extension .py; for example testfile.py. Then we run it by opening LXTerminal and navigating to where the file is saved. Then we run `python <filename>`. We can use the `cd` command to move to different.

6.2 Receiver Design

6.2.1 Introduction

The receiver design includes reception of transmitted signal through antenna where it is further pre-amplified by the RF pre-amplifier. The generated signal is digitally down-converted followed by the analog down-conversion by the NI hardware. Then decoding of BFSK signal is carried out and the data is decoded (Manchester decoding) in terms of Time which in turn is displayed on the Raspberry Pi.

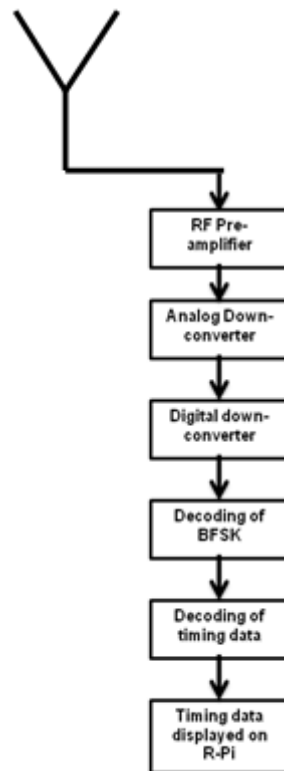


Fig. 6.3 Receiver Design

6.2.2 BFSK Demodulation

In theoretical approach the demodulation is carried out by coherent demodulation process. This involves a product modulator which multiplies BFSK with the $\cos \omega_1 t$ and integrates with respect to T_b . The value is summed to obtain the original baseband signal.

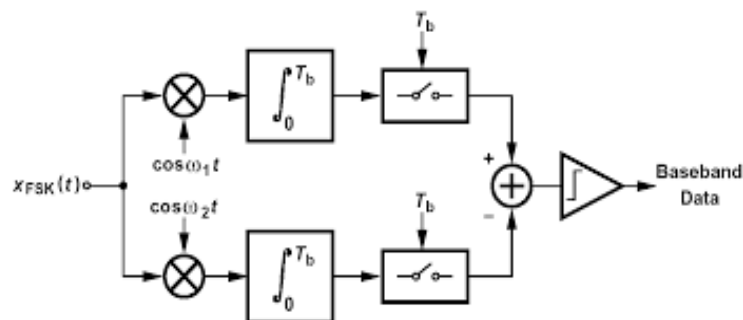


Fig. 6.4 BFSK Demodulation (Theoretical approach)

In practical approach BFSK signal is received by the target scoped FIFO and the required signal is filtered out by using FIR compiler. The demodulation is carried out taking a single cycle timed loop of 5MHz clock frequency and using a case structure for filtering out the desired signal of higher and lower frequency. The FIR compiler was configured accordingly in order to filter out higher and lower frequency.

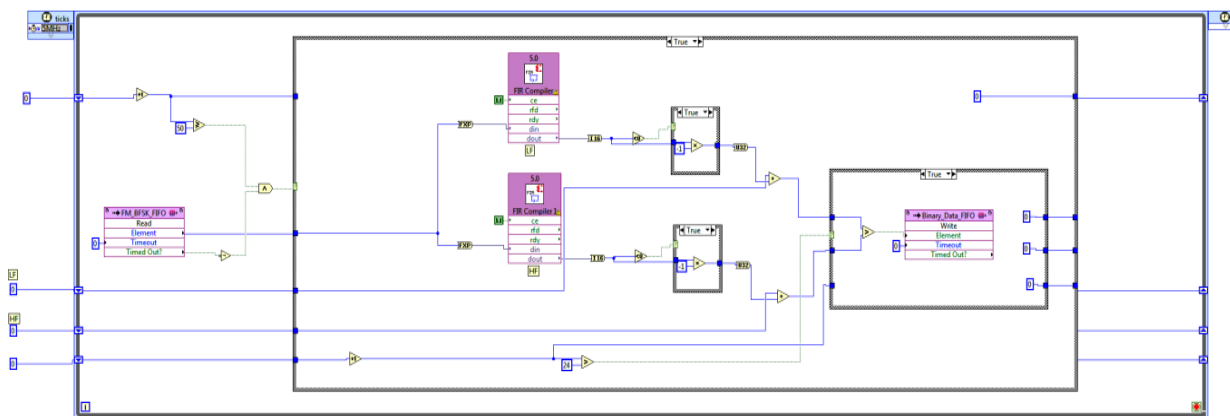


Fig. 6.5 BFSK Demodulation

6.2.3 Manchester decoding

The BFSK demodulated signal is received by the target scoped FIFO. In similar way the Manchester decoding was carried out by a single cycle timed loop of 5 MHz clock frequency. The loop was provided with appropriate constants for generating required number of bits and loop rates. The loop was also provided with an increment and a comparator for increasing the bit by number of 1 and comparing the sample to 1250 respectively. The case structure was used to perform the Manchester decoding by state machine concept which includes decoding of start bit, data bit and stop bit.

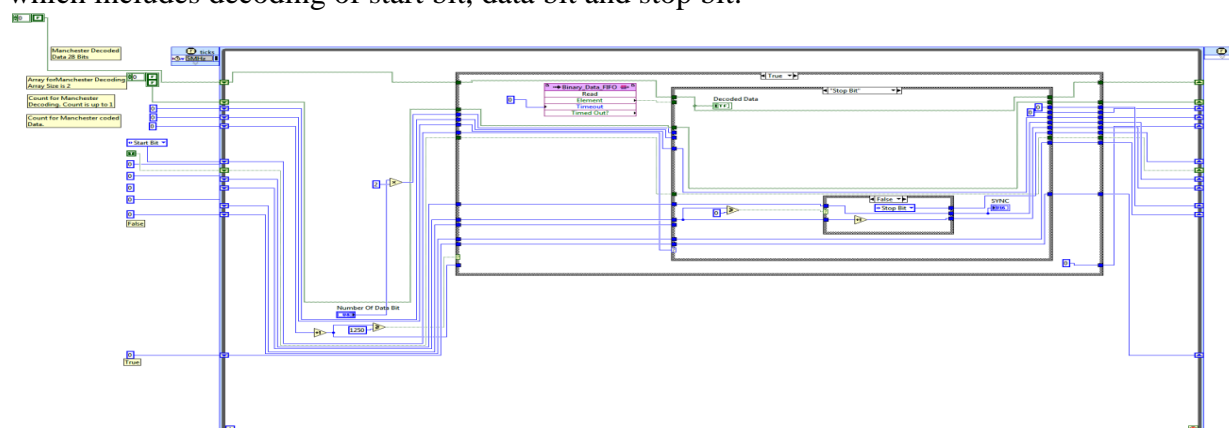


Fig. 6.6 Manchester Decoding

6.3 UHF to Baseband Down-conversion

A Down Converter is the counterpart component to the UC and is, therefore, equally important as a component in the same application systems. Its function is to translate a passband signal comprising one or more radio or intermediate frequency (RF or IF) carriers to one or more baseband channels for demodulation and interpretation.

An ADC samples the detected analog signal and feeds into the DC processing chain. An initial frequency translation, RF processing and additional filtering can be performed prior to the base down-conversion function. A mixer is used in combination with a vector of one or more sinusoids to channelize the signal, then each channel is filtered and finally demodulated.

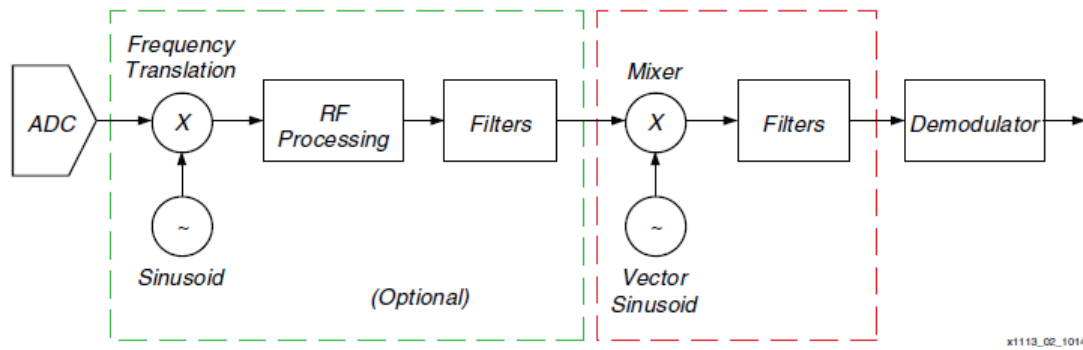


Fig. 6.7 Down-conversion

6.4 Raspberry Pi board display

6.4.1 Configuring with Raspberry Pi

The Raspberry Pi was installed accordingly with necessary equipments. The display was connected to the Raspberry Pi board. The R-Pi runs on Linux based environment. The configuration was carried out by providing login id as pi and password as raspberry as shown in Fig. 6.8.

```

pi@raspberrypi: ~
login as: pi
pi@10.111.5.92's password:
Linux raspberrypi 4.1.12+ #4 PREEMPT Tue Nov 3 14:35:30 IST 2015 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Jan 1 06:07:53 1970 from 10.111.5.238
pi@raspberrypi ~ $ startx

```

Fig. 6.8 Command window in PuTTY

After the configuration of R Pi, a Linux based window was observed where the python program was run by selecting the Run module option as shown in Fig. 6.9.

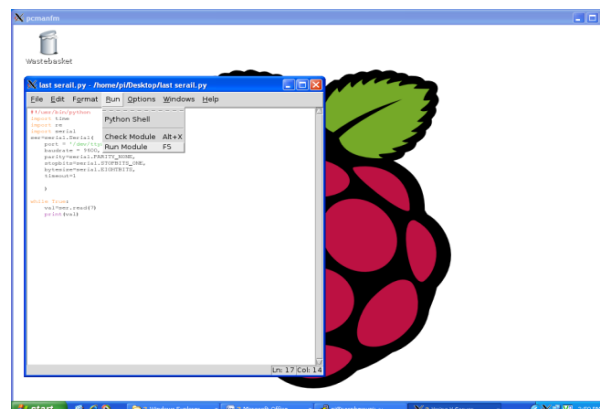


Fig. 6.9 Running the python program

The python program is mentioned as below;

```
#!/usr/bin/python
import time
import re
import serial
ser=serial.Serial(
    port = '/dev/ttyAMA0',
    baudrate = 9600,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS,
    timeout=1
)

while True:
    val=ser.read(7)
    print(val)
```

After running the program successfully the window is observed with the output on the Linux based window of the Raspberry Pi. The output of the python shell represents the Count Down Time in a downward manner as shown in Fig. 6.10.

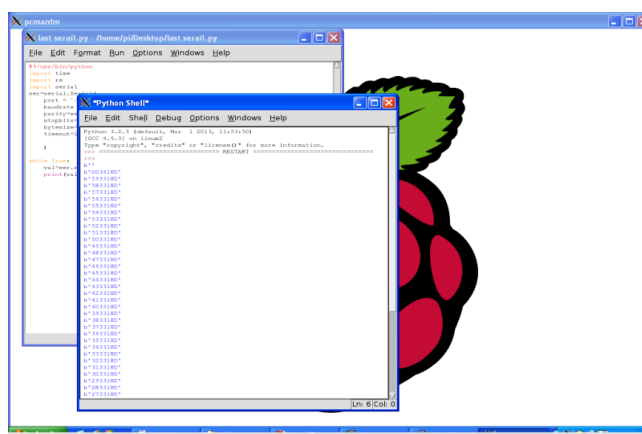


Fig. 6.10 Output Window with display of Count Down Time (CDT)

6.5 Experimental Setup

The whole system configuration of the project includes interfacing of hardware with the receiver of Raspberry Pi as shown in Fig. 6.8. The NI hardware consists of various sub-modules also known as chassis such as NI PXIe- 8135 (Embedded Controller), NI PXIe- 8431/8 (Connector RS-485/RS-422), NI PXI- 5690 (RF Preamplifier), NI PXI- 5600 (RF Down-converter), NI 5781 (Baseband Trans-receiver), NI PXI- 5600 (RF Up-converter) and NI PXIe- 8430/8 (Connector RS-232). An Ethernet cable is connected with a Host PC. The Host PC is responsible for the coding in LabVIEW software.

The receiver end consists of DB 9 Connection Port for serial communication from the NI hardware interfaced with the Raspberry Pi board. R Pi board is connected with a USB keyboard, USB mouse, and display screen.

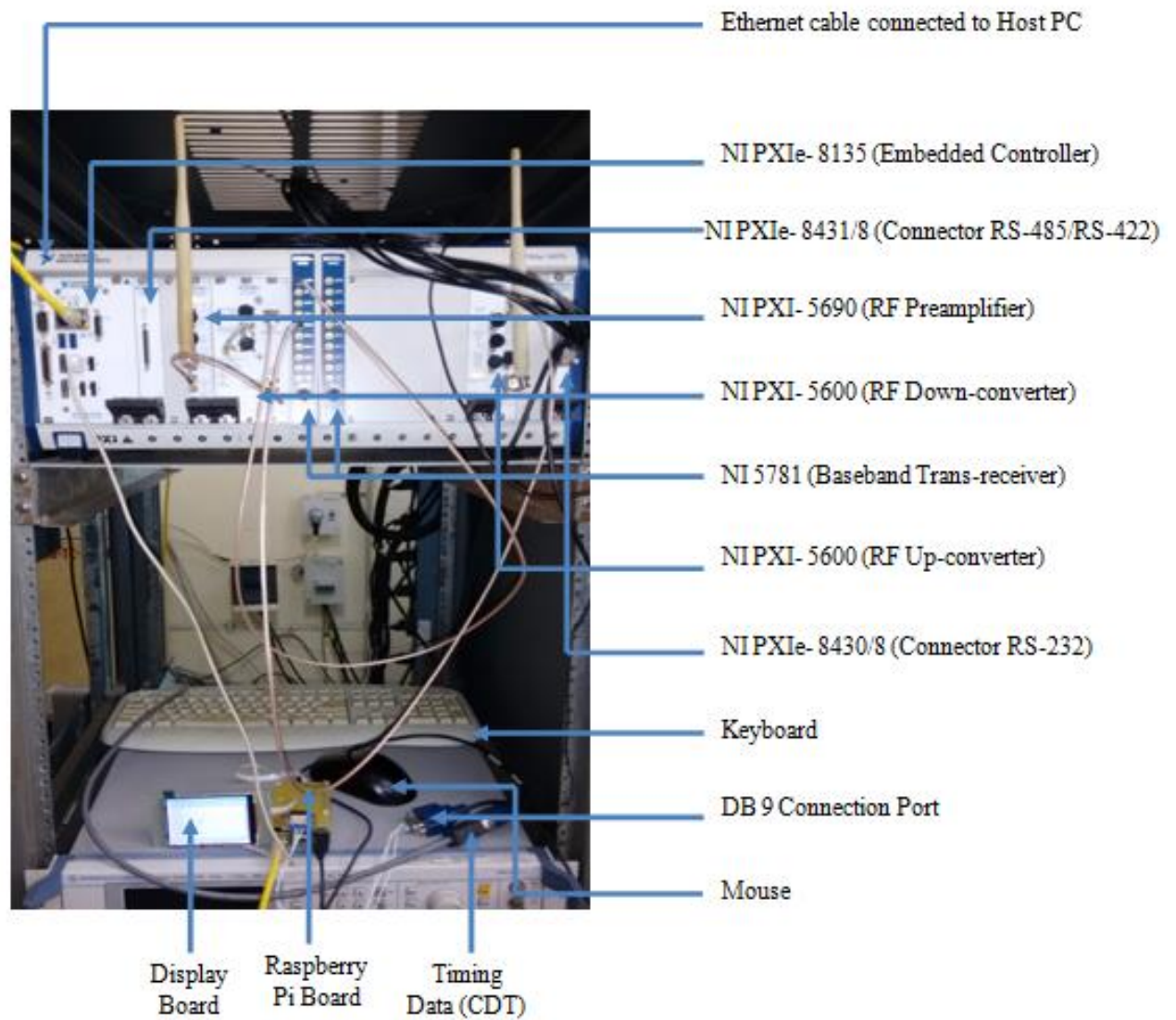


Fig. 6.11 Experimental Setup

Chapter 7

Results

7.1 Experimental Results

By various encoding, modulation and digital analog conversions the transmitter transmits up-converted BFSK of 25 MHz in UHF band and it receives the same in the down-converted form of 15 MHz. after the process. According to the Fig.7.1 the transmitted BFSK signal is mentions in blue and the received BFSK signal is shown in pink.

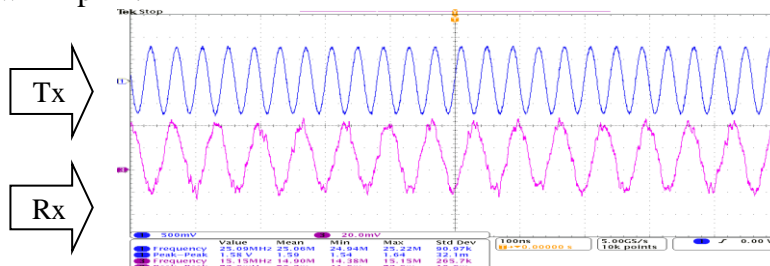


Fig. 7.1 BFSK signal generation at Transmitter (blue) and received BFSK signal at receiver (pink)

Fig. 7.2 indicates the front panel view of the Count Down Time by the LabVIEW software where the up-converter gain is increased by 15. Each value of the timing frame is mentioned by 4 bits of the Boolean LEDs as shown below.

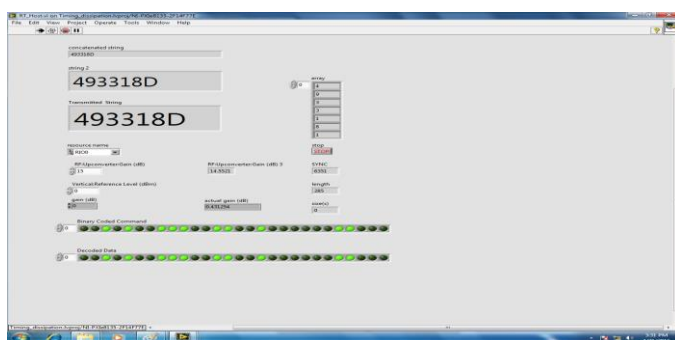


Fig. 7.2 Front panel view of CDT display

Fig. 7.3 indicates the CDT display on the Raspberry Pi after the process of demodulation and decoding. The display is displayed on a screen after the demodulation, decoding and down conversion by the NI hardware. This information of timing data is broadcasted to receiver stations.

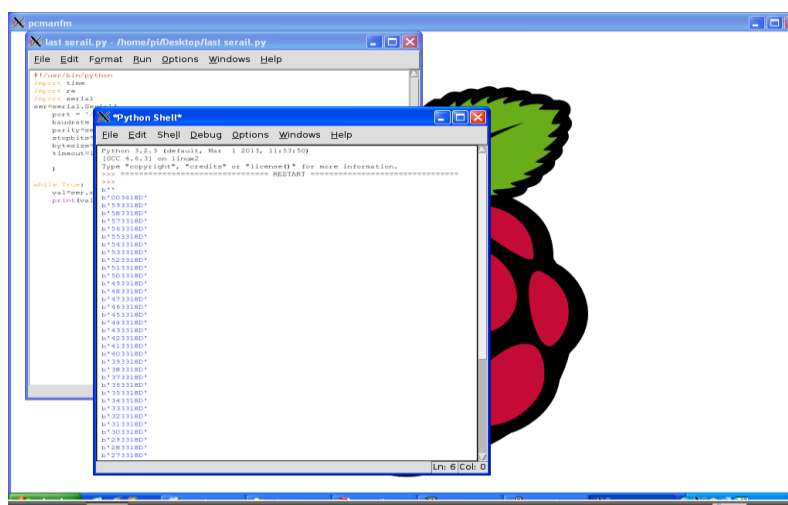


Fig. 7.3 CDT display Raspberry Pi

Chapter 8

Conclusion and Future Scope

8.1 Conclusion

The project involves functioning of the embedded system which makes it easier to replace large receivers with a small one at inaccessible remote places. This will be suitable for the users to receive any kind of information without any circumstances. Hence the thesis concludes by describing various effectiveness of using FPGA and Raspberry Pi board with the display of Count Down Time on the R-Pi. With the use of highly reliable software, FPGA are dynamic and are more flexible towards real time generation for various developmental applications. These are highly reconfigurable with rapid prototyping.

8.2 Future Scope

Further it can be implemented by using BPSK modulation technique for making bandwidth more efficient and making it suitable for future satellite communication. In the near future this project can be implemented by replacing large receivers into a small low cost receiver like Raspberry Pi which would be suitable for receiving any kind of information at any places in wireless medium. This can be useful for the communication where cabling is not possible such as deep sea testing. This can be very effective mostly at inaccessible remote places.

References

- [1] Amiya Ranjan Panda, Debahuti Mishra, Hare Krishna Ratha, "FPGA Implementation of Software Defined Radio-Based Flight Termination System," *IEEE Transactions on Industrial Informatics*, Vol.11, No.1, pp. 74-82, Feb. 2015.
- [2] N. Kehtarnavaz, S. Mahotra, "FPGA implementation made easy for applied digital signal processing courses," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) at Prague, Czech Republic*, Page(s):2892 - 2895, IEEE on 22-27 May 2011.
- [3] N. Kehtarnavaz, S. Mahotra, "Digital Signal Processing Laboratory: LabVIEW-Based FPGA Implementation", *Universal Publishers/BrownWalker Press*, 2010.
- [4] Swain, N. K. et al., "Remote data acquisition, control and analysis using labVIEW front panel & real time engine." *School of Engg. Technology and Science: South Carolina state University, IEEE*.
- [5] Sarthak Jain, Anant Vaibhav, Lovely Goyal, "Raspberry Pi based interactive home automation system through E-mail", *Optimization, Reliability, and Information Technology (ICROIT), 2014 International Conference, IEEE*, pp. 277 – 280, 6-8 Feb. 2014.
- [6] Sanjana Prasad, P.Mahalakshmi, A.John Clement Sunder, R.Swathi, "Smart Surveillance Monitoring System Using Raspberry PI and PIR Sensor", *International Journal of Computer Science and Information Technologies*, Vol. 5 (6), pp. 7107-7109, 2014.
- [7] M. Mahadi Abdul Jamil, M. Shukri Ahmad, "A pilot study: Development of home automation system via raspberry Pi", *Biomedical Engineering (ICoBE), 2015 2nd International Conference*, pp. 1 – 4, IEEE, 30-31 March 2015
- [8] Federico Milano, "A python-based software tool for power system analysis", *2013 IEEE Power & Energy Society General Meeting*, IEEE, pp. 1 – 5, 21-25 July 2013
- [9] C. C. W. Robson, A. Bousselham, and C. Bohm, "An FPGA-based general-purpose data acquisition controller," *IEEE Trans. Nucl. Sci.*, vol. 53, no. 4, pp. 2092–2096, Aug. 2006.
- [10] V. B. Alluri, J. R. Heath, and M. Lhamon, "A new multichannel, coherent amplitude modulated, time-division multiplexed, software-defined radio receiver architecture, and field-programmable-gate-array technology implementation," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5369–5384, Oct. 2010.