Wideband harmonic radar detection

Farrukh Aslam, S. M.

Wideband Harmonic Radar Detection

By

S. M. Farrukh Aslam

A thesis submitted to the
University of Durham, Durham, United Kingdom,
For the degree of MSc
June 2008

Abstract

Radio sites consist naturally of metallic structures. Metals are always covered by an oxide film due to the metal reacting chemically with the oxygen in air. The rate of this oxide formation depends largely on the environment. Any oxide film between metallic contacts will cause non-linearity. RF currents passing through these junctions would generate harmonics. When RF signals at two frequencies f1 and f2 pass through a non-linearity they create signals at their sum and difference frequencies. These are known as 'inter-modulation products'. This generation of inter-modulation products when radio waves interact with rusty parts is called as the 'Rusty Bolt Effect'. Radio spectrum is carefully controlled for optimal usage of the available frequencies so that different services operate in well-defined frequency channels. Ofcom has set some standards for radio site engineering. This set of standards is given in the document 'MPT 1331: Code of Practice for Radio Site Engineering'. Any transmission site which is not following these codes would likely cause interference to other users. It is important that radio engineers should check the sites for their compliance with these codes. If a particular radio site is causing interference due to the rusty-bolt effect, the corroded points must be located to minimize their effect using a Harmonic Radar.

A 'Harmonic Radar' is a device that illuminates a region of space with RF waves and receives the harmonics of the transmitted frequencies. The received data can then be processed to find the exact location and mobility of the points causing the generation of these harmonics. It works on the principle of radar transmitting a chirp signal and receiving harmonics of the transmitting frequency. Work is currently being carried out at the 'Centre for Communication Systems' in Durham University funded by HMGCC on the design and implementation of a novel Wideband Harmonic Radar system. The radar system would employ advanced sub-systems i.e. a suitable waveform and multiple antenna arrays processing super-resolution algorithms for angular information.

Declaration

No portion of the work referred to in this report has been submitted in support of an application for another degree or qualification at this or any other university, or institution of learning.

Acknowledgements

I would like to sincerely thank my supervisor, Professor Sana Salous for her help and guidance throughout the duration of my PhD. I would also like to express my gratitude to Dr. Feeney, who on many occasions provided me with constructive discussions and guidance. Furthermore, I would like to give my thanks to the HMGCC for funding my studies and giving me the opportunity to undertake research in this field.

List of Abbreviations

AOA	Angle of Arrival
DOA	Direction of Arrival
DSP	Digital Signal Processing
EIRP	Effective Isotropic Radiated Power
EM	Expectation Maximization
FFT	Fast Fourier Transform
FMCW	Frequency Modulated Continuous Wave
MIMO	Multiple Input Multiple Output
ML	Maximum Likelihood
NLJD	Non-Linear Junction Detector
PIM	Passive Intermodulation
RCS	Radar Cross Section
RF	Radio Frequency
Rx	Receiver
SAGE	Space Alternating Generalized Expectation-Maximization
SIMO	Single Input Multiple Output
Tx	Transmitter
UCA	Uniform Circular Array
ULA	Uniform Linear Array
VFO	Variable Frequency Oscillator
WRF	Waveform Repition Frequency

Table of Contents

Chapter 1

1.1 Introduction

A Radar system uses electromagnetic waves to identify the range, direction, or speed of moving and fixed objects such as aircraft, ships, vehicles, and landscape. The term RADAR is an acronym for **Ra**dio **D**etection **a**nd **R**anging. Radar can be regarded as an all-weather day/night performing sensor that can measure target range accurately and precisely.

Radar system was developed during World War II as a way to detect enemy aircrafts. Enemy airplanes could be detected because they reflected some of the transmitted energy. Crude radar images were obtained on the displays of airborne radar systems. Over the many decades radar techniques and technology have been developed. The enormous advances that have been made in radar since the 1950s are mainly due to the development of fast, high-performance digital processing hardware and algorithms [1]. Radar systems continue to find applications in diverse areas like Detection and Ranging of ground, sea and air targets, Air Traffic Control (ATC), Meteorological applications, Collision avoidance, Speed measurement and Remote sensing.

Much work has been done in developing radar systems having a high bandwidth. The distinguishing characteristic of a wideband radar is its fine range resolution, which is inversely proportional to the operating bandwidth. Wideband radar systems help to accurately pin-point the intended target. The Lincoln Laboratory in the United States has done pioneering work in the development of high-power wideband radar. Since 1970 the Laboratory has developed and fielded several wideband radars for use in ballistic-missile-defence research and space-object identification. [2]

One main advancement has been in the different types of radar transmit waveforms. The type and quality of information received by a radar depends in part on the waveform it transmits. FMCW – Frequency Modulated Continuous Waveform is one example that has been successfully employed in practical systems. [3]

An interesting class of radar systems is a harmonic radar whereby the radar system receives the harmonics of the fundamental frequency transmitted. The aim is to locate and identify the target(s) that are generating these harmonics. Harmonic radars have been put to great use in the field of Entomology where they have been instrumental in tracking the movement of insects. Such systems are now being developed to be used in other areas where non-linear or harmonic generating elements are the intended targets. One such application is to locate the junctions producing passive inter-modulation frequencies commonly at transmission sites [4]. These rusty-bolt joints can be located using a wideband harmonic detector and counter-measures can then be performed.

Figure 1.1 SIMO Channel System

Antenna technology has proved to be a very important component of radar and other communication systems. Recent research work has been on 'smart antenna' systems which make use of antenna arrays. One such system is the SIMO system as shown in Figure 1.1. SIMO (single input, multiple output) is an antenna technology for wireless communications in which multiple antennas are used at the receiver. The antennas are combined to minimize errors and optimize data speed. The data henceforth received can be processed using algorithms to extract useful information. Advancement in signal processing techniques has made it possible to employ super-resolution algorithms to precisely locate the targets.

The project undertaken aims to utilize the strengths of various systems, techniques and methodologies mentioned above to develop a highly effective 'Wideband Harmonic Radar system' with appropriate antenna arrays employing high resolution signal processing algorithms.

1.2 Review of Chapter Contents

Chapter 2 describes the FMCW Harmonic radar system in detail. It explains the theory behind an FMCW waveform and gives an overview of the historical development of harmonic radar system.

Chapter 3 provides an overview of the Rusty-bolt Effect. It describes the reason behind its occurrence and mentions the counter-measures available in literature.

Chapter 4 provides a system level overview of the harmonic radar. It also explains the work done so far and gives an understanding of its various components.

Chapter 5 deals with signal processing algorithms. Two techniques are mentioned and their underlying principle is explained.

Chapter 6 is the conclusion of the report. It describes the work accomplished so far and the requirements needed in order to carry out developing the rest of the radar system.

1.3 References

[1] H. D. Griffiths, Chris J. Baker, *"Radar Imaging for Combating Terrorism"*, Department of Electronic and Electrical Engineering, University College London, London, UK

[2] W. W. Camp, J. T. Mayhan, R. M. O'Donnell, *"Wideband Radar for Ballistic Missile Defense and Range-Doppler Imaging of Satellites"*, Lincoln Laboratory Journal volume 12, number 2, 2000

[3] M. I. Skolnik, *Introduction to Radar Systems*, 2nd ed: McGraw Hill, 1988

[4] "MPT 1331 : Code of Practice for Radio Site Engineering", June 2001, Flyde Microsystems Ltd

Chapter 2

FMCW Harmonic Radars

2.1 Radar Theory & Principles

Radar measurement is based on the principles of properties of radiated electromagnetic energy. Electromagnetic energy travels through air at approximately the speed of light. This energy is transmitted to and reflected from the reflecting object. A small portion of the reflected energy returns to the radar set. This returned energy is called an 'echo'. Radar sets use the echo to determine primarily the direction and distance of the reflecting object.

One of the most important relations used to define the working of a radar system is the radar range equation. It relates the transmit power from the radar to the received power in terms of range, antenna and target dimensions. The fundamental form of the radar range equation is

$$P_r = \frac{P_t G}{4\pi R^2} \frac{\sigma}{4\pi R^2} A_e \qquad (2.1)$$

where

- P_t - Transmit Power
- P_r - Receive Power
- G - Antenna Gain
- R - Target Range
- σ - RCS – Radar Cross Section i.e. measure of size of target as seen by the radar. Its units are that of area.
- A_e - Effective Aperture of receiving antenna

Two important parameters for radar performance is its resolution and sensitivity.

Resolution

The range resolution tells us how far apart two targets have to be to be distinguished as two separate entities. If the time delay between the reflected signals from two

objects is greater than the pulse duration, then the two objects are seen separately. [21] If the targets are closer than this duration, then the receiver will not be able to distinguish between the two.

Sensitivity

The range sensitivity or accuracy indicates uncertainty in a measurement of the absolute distance to an object. Intuitively, accuracy of a range measurement should depend on the 'sharpness' of the pulse shape. However, the crucial factor determining range accuracy is bandwidth. Consider a single wavelength being used to locate a target. The target phase measured will be ambiguous every wavelength. A second wavelength added to the transmission will aide in locating the target. It reduces ambiguities and sharpens the position of the target. Adding more wavelengths means increasing bandwidth. Therefore, adding more bandwidth leads to greater accuracy and the system becomes more sensitive [21].

2.2 FMCW / Chirp Waveform

FMCW or Frequency Modulated Continuous Wave work on the principle of 'pulse compression' or 'pulse coding'. It is a processing technique that maximises the sensitivity and resolution of radar systems. Chirp signals derive their name from the fact that if the signal was audible it would sound like the chirp or tweet of a bird. The chirp of a bird increases monotonically over a frequency interval.

Chirp waveform was first classified by Klauder et al. [1] as published in the Bell Labs Technical Journal in 1960. However, the basic idea was presented earlier by Oliver in 1951 in a Bell Telephone Laboratory internal memorandum which was interestingly titled "Not with a bang but a Chirp" [2].

Increase in radar sensitivity can be achieved by two methods.

1. By increasing the average transmitting power
2. By increasing the pulse length

Average transmitted power is the peak power multiplied by the transmitter duty cycle. The peak power can be as high as several hundred kilowatts. However, a pulse radar transmits short burst of pulses where the average transmit power might be 1% of the peak transmit power. This makes such a system very inefficient. Increasing the pulse length can increase the average transmitted power without decreasing efficiency but this has an adverse effect of degrading the range resolution. The radio pulse is too long and can not distinguish between two closely spaced targets.

This conflict in user requirements can be resolved in designing such a transmit waveform that would maximise both sensitivity and resolution. As described above, the range resolution does not necessarily depend upon the pulse length but on the pulse bandwidth. The bandwidth can be altered by manipulating the amplitude and/or phase within the pulse without changing its duration. Thus, the radar resolution can be increased without having to change its duration. This manipulation of pulse is called 'pulse coding'. The types of pulses that involve changing the frequency of the transmitted pulses are called 'chirp' pulses. This is an electronic method to boost the apparent signal strength as perceived by the radar receiver. The outgoing radar pulses are chirped, that is, the frequency of the carrier is varied within the pulse, much like the sound of a cricket chirping.

The timing mark is the changing frequency. The transit time is proportional to the difference in frequency between the echo and the transmitter signal. The greater the transmitter frequency deviation in a given time interval, the more accurate the measurement of the transit time and the greater will be the transmitted spectrum.

In FMCW radar, the transmitter frequency is changed as a function of time in a known manner. This known manner can be linear or non-linear in nature.

A linear chirp waveform modulated signal can be defined by the following equations:

$$x_T(t) = A_0 \cos[\phi_T(t)] \tag{2.2}$$

where $$\phi_T(t) = 2\pi\left(f_c t + \frac{kt^2}{2}\right) \tag{2.3}$$

and $$A_T = \begin{cases} E_o \text{ for } 0 \le t \le t' \\ 0 \quad \text{otherwise} \end{cases} \qquad (2.4)$$

Here, the subscript T stands for transmitting, f_0 is the carrier frequency at $t = 0$, k is the linear chirp rate and $\phi_T(t)$ is the instantaneous transmitted phase. The time value t' represents the length of the transmitted chirp signal. It is known that the instantaneous transmitted frequency is specified by:

$$f_T(t) = \left(\frac{1}{2\pi}\right)\frac{d\phi(t)}{dt} \qquad (2.5)$$

which implies $$f_T(t) = f_c + kt \qquad (2.6)$$

It is evident from the equation that the frequency increases linearly with time. Therefore, the transmitted signal exhibits linear frequency modulation commonly referred to as the saw-tooth signal shown in figure 2.1

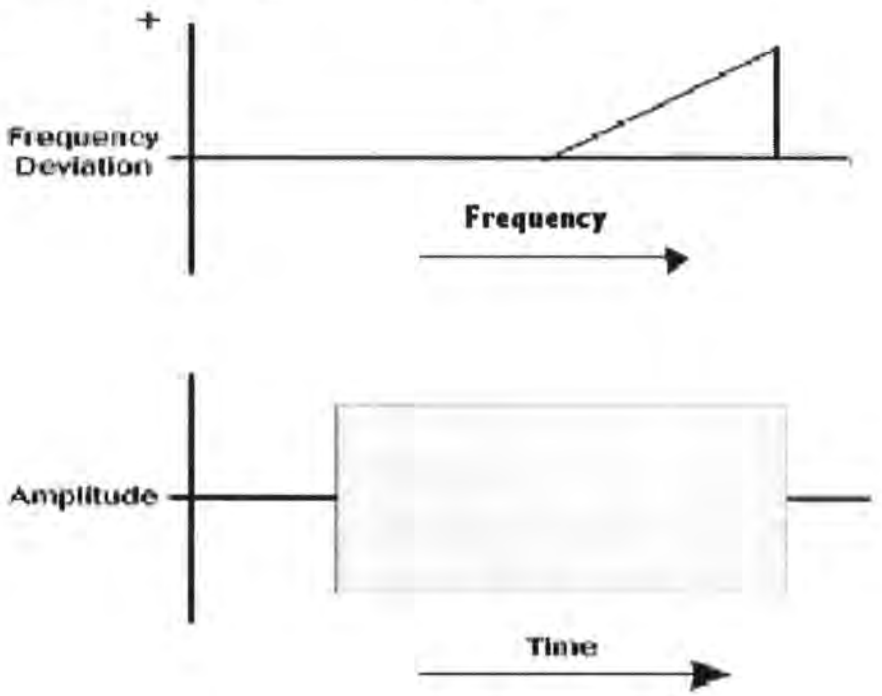
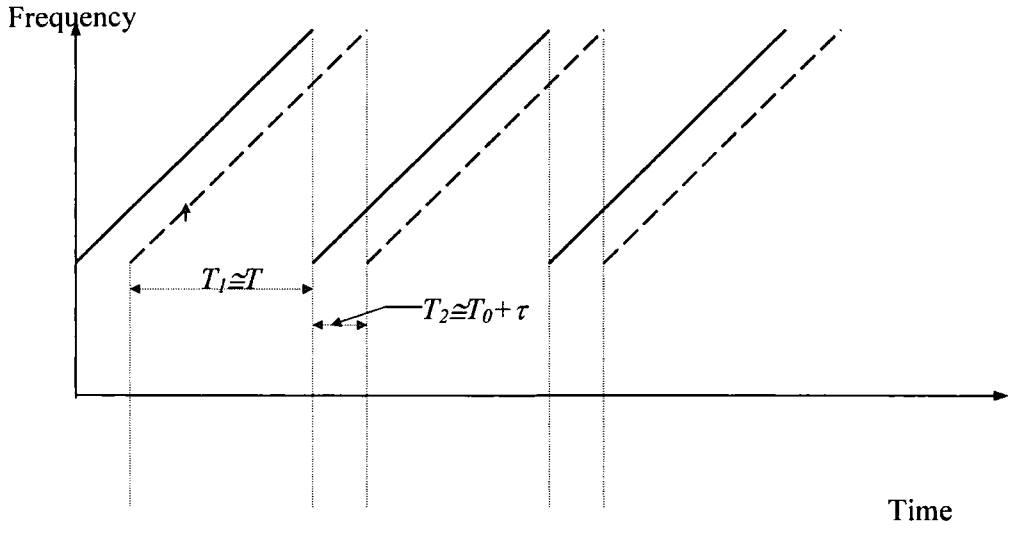


Fig 2.1 A Chirp Signal

This signal is generated repetitively a fixed quantity of times in a second. This quantity is known as the Wave Repetition Frequency (WRF).



——— Transmitted chirp ---------- Received chirp

Fig. 2.2 Resultant frequencies at the receiver [3]

Figure 2.2 shows the resultant frequency at the receiver. Here it is possible to see graphically all the parameters involved in the preceding sections. The solid line represents the transmitted signal the dotted line shows the received signal. The received signal is delayed by a time τ, with respect to the transmitted signal. If a signal that is a replica of the transmitted signal is combined with the received signal, then the resultant difference frequency will be a train pulse and the pulse width has a frequency f_D . This can be used to extract useful information using a signal processing technique. More of it is explained in Chapter 6.

For a linear chirp, the sweep rate is the slope of y-axis (frequency) to x-axis (time) over a single chirp. Therefore, the sweep rate 'k' is

$$k = \frac{B}{T} \tag{2.7}$$

and equation (2.2) becomes

$$x_T(t) = A_T \cos[\phi_T(t)] = A_T \cos\left(2\pi f_c t \pm \pi \frac{B}{T} t^2\right) \tag{2.8}$$

where f_c is the carrier frequency

A_T is the amplitude of the transmitted signal and is constant

B is the sweep bandwidth

T is the sweep period.

The instantaneous frequency of this signal is:

$$f_T(t) = \frac{1}{2\pi}\frac{d\phi_T(t)}{dt} = f_c \pm \frac{B}{T}t \tag{2.9}$$

For a linearly increasing chirp,

$$f_T(t) = f_c + \frac{B}{T}t \tag{2.10}$$

In this case, f_c is the lower frequency of the bandwidth.

2.3 Harmonic Radar – Historical Development

Harmonic Radar is a device that illuminates a region of space with RF waves and receives the harmonics of the transmitted frequencies. The received data can then be processed to find the exact locations of the points causing the generation of these harmonics. It works on the principle of radar transmitting a chirp signal and receiving harmonics of the transmitting frequency.

Radars were the earliest applications of chirp waveform. Barrick [4], Poole [5], Klauder et al. [1] presented valuable works related to the theory of the FMCW signals and its radar applications. With the development of electronic circuit techniques and signal processing algorithms, this principle found new applications - one of them being in area of mobile radio channel characterisation as presented by Salous [6-7].

An early commercial FMCW Harmonic Radar system was METRRA – Metal Target Re-Radiation developed for the US Army in the late 70's. [8] Its intended use was to detect stationary military targets e.g. tanks, vehicles, and weapon caches etc. which are hidden by foliage. The system transmitted a 400 MHz signal and received its third harmonic (120 MHz) signal. It is reported to be successfully demonstrated at a range of 1 kilometre. [8]

Another harmonic radar system was developed by the U.S. 'Strategic Environmental Research and Development Program (SERDP)' which is the Department of Defense's (DoD) environmental science and technology program. This system developed in the late 90's third harmonics and used for Unexploded Ordinance (UXO) Detection. This project built a prototype system that demonstrated capability to detect and locate buried UXO remotely. [9]

Harmonic radars have also been an area of interest in academia. One such system was developed at the Electromagnetic Laboratory at Michigan State University. Its intended use was in etymology. This radar was a bi-static CW system. The transmit frequency was 800 – 900 MHz and the receiver could be tuned to from 1232 MHz to 1862 MHz. The system was intended to detect harmonic radar tags in high-clutter environments. [10]

Recent developments in non-linear junction detectors (NLJD) have shown the advancement and sophistication of new devices coming up. A number of patents have been issued in the past few years.

Jones et al. [11] developed a non-linear junction detector for counter surveillance measures. Its main feature is the use of a circularly polarized Tx/Rx antenna. Using linearly polarized antenna, one has to scan the target area twice in a horizontal and vertical position. This ensures that the surveillance device returning polarized harmonic return do not go un-detected. Using a NLJD with circularly polarized antenna allows successful detection regardless of which angle the scan is made. Barsumian et al. [12] have developed an interesting NLJD that transmits a series of pulses as the transmit signal. The transmit power of pulses is varied. The received harmonics of all pulses are compared to each other as well as to a set of standard data. The harmonic signals are analysed if they correspond to a known set of non-linear device. The receive harmonics are also demodulated to the audible frequency range where they are detected by an audio circuit. Some NLJDs have a feed back control system in them such as developed by Holmes et al. [13]. The feed back control maintains a pre-determined minimum threshold value of the received signals. So the control system has two parts – one to determine the signal strength and other to vary

the power output level of the transmitter. Advantage of this invention is improved system efficiency.

All of the systems mentioned above transmit at a single frequency and receive at multiple harmonics. An innovative method is to transmit at multiple frequencies and then try to receive different combination of its harmonics. One such invention is the CWER (Concealed Weapon and Electronics Radar) system developed at John Hopkins University by Jablonski et al. [14]. This system transmits two frequencies at f_1 and f_2 from separate antenna systems. This creates received signals of the order of $nf_1 \pm mf_2$, where n and m are integers. A single frequency transmit has adequate detection capability but it has very limited capability with respect to classifying different types of targets. This Dual Frequency Scanning Harmonic Radar has the ability to distinguish objects of different types and to distinguish them from nearby clutter. Another recently patented invention was by Rafael-Armament Development Authority Ltd. [15] This NLJD transmits more than two frequencies f1, f2, f3 .... towards the area of interest. The receiving unit then tunes to receive at intermodulation products nf1 + mf2 + qf3 ...., wherein n, m, q are non-zero integers.

2.3.1 Non-Linear Junction Detector

The use of a Harmonic Detector, also called as a 'Non-Linear Junction Detector' is dependent on the fact that electronic devices and metallic objects that come in contact with one another create a non-linear junction. What a non-linear junction detector does is detect these non-linear junctions. In doing so, active or inactive junctions can be detected. One should keep in mind that these non-linear junctions can be anything from electronic circuits like eavesdropping equipment (bugs, microphones etc) to various corroded metal junctions. Various methods of analysis and algorithms can be used to distinguish between the targets.

The two main functions of a sweep using Non Linear Junction Detectors are:

1. The Detection of non-linear junctions
2. The Discrimination between junctions so as to differentiate between electronics and other forms of junctions

The RF sweep would be useless without the second step as there would be many false alarms.

The junctions in electronic devices are different than those in other junctions. The harmonic returns of electronics are well defined but that of non-electronics are not. Electronic components will have a strong second harmonic signal and a weak third harmonic signal. Other junctions will have a weak second harmonic signal and a strong third harmonic signal [16]. The 3rd, 5th and other odd harmonic reflections are reflected by any conductive or metallic surface. Fig. 2.3 shows graph of current and voltage characteristics of a semi-conductor junction and a false junction.

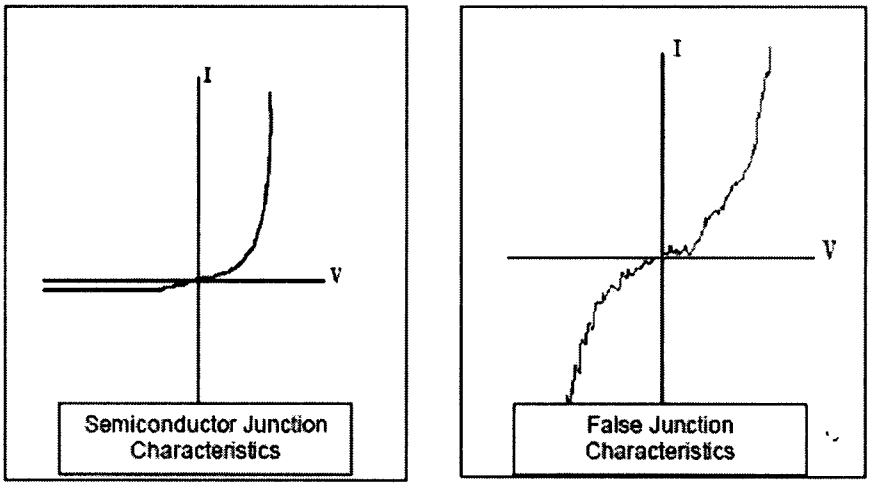


Fig. 2.3 IV- graphs for non-linear junctions [17]

The non-linear characteristics of semiconductor junctions differ from false junctions: the 2nd and 3rd harmonic signals will have different intensities. When a NLJD radiates a semiconductor junction, it results in a 2nd harmonic stronger than the 3rd harmonic. A false junction returns a 3rd harmonic that is stronger than the 2nd harmonic.

Fig. 2.4 User Interface Graph for a commercial NLJD [17]

NLJD's should have the capability to compare the received signal strength of both the 2nd and 3rd harmonic which will enable to discriminate between true semiconductor junctions and false junctions. It is also very important that NLJD's with both 2nd and 3rd harmonic receiving capabilities provide good RF isolation because the receiving functions must not interfere with each other. If it is not the case then a pure semiconductor junction may still appear to have a fairly strong 3rd harmonic and a pure false junction may appear to have a fairly strong 2nd harmonic which will lead to erroneous results. Fig. 2.4 shows the display of ORION – a commercially available Non-Linear Junction detector which displays the relative harmonic levels of both a semi-conductor and a false junction.

2.3.2 Issues related to NLJD Operation

Most NLJDs on the market today transmit on a single frequency or are limited to a small frequency range. This creates three problems.

2.3.2.1 Null Range Effect

If the distance between the NLJD and the target is equal to ½ the wavelength of the transmit frequency then there is a null effect in the RF transmit signal. This reduces the detection sensitivity associated with that specific range and frequency. Usually this effect is not a problem because the user is constantly moving the NLJD and therefore the range to the target is constantly changing.

2.3.2.2 Target Dependence on Frequency

It is often observed that NLJD's perform differently for different targets. This is because detection range is dependent on frequency. Consider a cellular phone as a potential target. If an NLJD operates at a frequency that is within the operational band of the cellular phone, then the detection range of the phone will be large, however, if the NLJD operates at a frequency range that is outside of the operational band of the cellular phone, then the built-in filters within the phone will attenuate the NLJD signal and the detection range will be greatly reduced.

2.3.2.3 Frequency Interference

If the NLJD is operating on a frequency that may also be occupied by another transmitter, the NLJD may have very erratic and unreliable readings. As more wireless devices are being assigned to more frequencies, the performance of these limited NLJD units can suffer. NLJD should be frequency agile and automatically search for quiet channels on which to operate to avoid frequency interference from other devices.

One commercial NLJD ORION™ [20] addresses the above mentioned problems using two methods: Quiet Channel Search and Frequency Hopping. In normal search mode, the ORION™ automatically searches for the quietest channels on which to operate in the ambient environment. The new frequency hopping search method employs an algorithm that constantly changes the transmit frequency over the full legal range to increase the target hit rate.

A list of popular NLJD's used in the commercial market is given in a tabular format below listing their main parameters:-

Company	**Product**	**Frequency (MHz)**	**Power**	**Antenna Polarization**
AudioTel, U.K.	SuperBroom	**Tx** – 888.5 **Rx1** – 1777 **Rx2** – 2665.5	EIRP - +40dBm	**Tx** – Linear **Rx** - Circular
Research Electronics, U.K.	ORION	**Tx** – 850 – 1005 **Rx1** – 1700-	1.4W	**Tx** – Circular

		2010 **Rx2** – 2550-3015		**Rx** – Circular
Surveillance Consulting Group, Switzerland	Eclipse	**Tx** – 890 – 895 **Rx1** – 1780 – 1790 **Rx2** – 2670 - 5370	2 W	**Tx** – Circular **Rx** – Circular
Information Security Associates, U.S.A.	Boomerang	**Tx** - 915 **Rx1** – 1830 **Rx2** – 2745	100mW, 500mW	**Tx** – Linear **Rx** - Linear

2.3.3 Further Applications

FMCW Harmonic Radars can also be used in many other different applications such as:-

1. Radar Entomology i.e. tracking of insects' movements e.g. Butterflies, Bees, and Snails etc.
2. Unexploded Ordinance Detection (Third Harmonic Detection)
3. Electronic counter surveillance i.e. Detection of 'bugs' implanted by enemy agents. Professional investigators or "spies" sometimes use many electronic devices that do not utilize radio frequency transmissions. The NLJD will detect and locate any electronic device regardless of whether or not the device is powered.
4. Measurement of thickness of ionizing layer in space and upper atmosphere

2.4 Harmonic Radar Range Equation

The basic radar range equations can be used to develop the range equation for harmonic radar. This can be subsequently used to calculate the power link budget.

Colpitts & Boiteau [18] developed the range equation for harmonic radar to be used for insect tracking. The target was a harmonic 'tag' (a dipole and an inductive loop) which would be planted on insects to track them.

The Friis Transmission equation is given as

$$P_r = P_t \frac{G_t G_r \lambda^2}{(4\pi R)^2} \tag{2.11}$$

The Harmonic Radar Range Equation can be developed in a similar manner as the conventional radar except that a harmonic cross-section should be replaced for radar cross-section. The harmonic cross-sectional area of the tag is described as follows:

$$\sigma_h = \frac{P_{dth} G_{dh}}{W_{df}} \tag{2.12}$$

where $P_{dth}G_{dh}$ - Effective isotropic radiated power (EIRP) from the tag at the second harmonic. Its units are Watts.

W_{df} - Fundamental frequency power density incident upon the tag in units of W / m^2

The power density incident upon the tag is given by

$$W_{df} = \frac{P_{stf} G_{sf}}{4\pi R^2} \tag{2.13}$$

where P_{stf} - Radar transmit power

G_{sf} - Gain of the transmit antenna

R - Distance between radar and tag

The EIRP of the illuminated tag is the power transmitted from the tag at the second harmonic multiplied by the antenna gain of the tag at the second harmonic.

$$(EIRP)_{tag} = P_{dth} G_{dth} = \frac{\sigma_h P_{stf} G_{sf}}{4\pi R^2} \tag{2.14}$$

The received second harmonic power from the tag back to the radar can be calculated as

$$P_{srh} = P_{dth} \frac{G_{dh} G_{sh} \lambda_h^{\,2}}{(4\pi R^2)} \tag{2.15}$$

where G_{sh} - Receive radar antenna gain at the second harmonic frequency

λ_h - Harmonic wavelength

The ratio of the received second harmonic power to the radar transmit power at the fundamental frequency is given as

$$\frac{P_{srh}}{P_{stf}} = \frac{\sigma_h G_{sf} G_{sh}}{4\pi} \left[\frac{\lambda_h}{4\pi R^2} \right]^2 \tag{2.16}$$

Jenn [19] developed a generalised harmonic radar range equation for the third harmonic given as

$$\mathrm{P_r} = \frac{(P_t G_t)^{\alpha} G_r \lambda_3^{\,2} \sigma_h}{(4\pi)^{\alpha+2} R^{2\alpha+2}} \tag{2.17}$$

where $\alpha \approx 2.5$ is a non linear parameter determined experimentally. It varies slightly from target to target.

2.5 References

[1] Klauder J. R., Price A. C., Darlington S., and Albersheim W. J., "The Theory and Design of Chirp Radars", *The Bell System Technical Journal*, July 1960, Vol. 39, No. 4, pp 745-815

[2] B.M. Oliver, Bell Telephone Laboratories Technical Memorandum, MM51-150-10, Case 33089, March 8, 1951

[3] V. M. Hinostroza, "Indoor Wideband Mobile Radio Channel Characterisation System", UMIST PhD Thesis, 2002

[4] Barrick D. E. " FMCW radar signals and digital processing". *NOAA technical report,* ERL 283-wpl26

[5] Poole A. W. V., " Advanced sounding. The FMCW alternative", *Radio Science*, December 1985, Vol. 20, No. 6, pp 1609-1619.

[6] Salous S., and Nickandrou N. "Architecture for Advanced FMCW Sounding", *International Journal in Electronics*, 1998, Vol. 84, No. 5, pp. 429-436.

[7] Salous S., Nikandrou N., and Bajj N.F., "Digital techniques for mobile radio chirp sounders ", *IEE proceedings Communications*, June 1998, Vol. 145, No. 3, pp. 191-196.

[8] *The Field Artillery Journal*, March-April 1977 pp. 50-51, http://sill-www.army.mil/famag/1977/ [accessed 17th Feb. 2008]

[9] J. Kositsky, (2001), *Unexploded Ordnance (UXO) Detection by Enhanced Harmonic Radar*, http://www.p2pays.org/ref/20/19401.pdf [accessed 17th Feb. 2008]

[10] Gregory L. Charvat, Edward J. Rothwell, Leo C. Kempel, "Harmonic Radar Tag measurement", Michigan State University, (2003)

[11] THOMAS H. JONES. 2000. *Non-Linear Junction Detector.* US006057765A.

[12] BRUCE R. BARSUMIAN. 2000. *Pulse Transmitting Non-Linear Junction Detector.* US006163259A

[13] STEVEN JOHN HOMES. 2005. *Non-linear junction detector.* US006897777B2

[14] DANIEL G. JABLONSKI. 2004. *System and method of radar detection of non-linear interfaces.* US006765527B2

[15] ITZHAK SCHNITZER, 2007. *Radar system and method for locating and identifying objects by their non-linear echo signals*

[16] Ralph D. Thomas, (1999), *The Detection of Active and Non-Active Eavesdropping Devices Buried In Walls and Furniture*, http://www.pimall.com/nais/n.junction.html [accessed 17th Feb. 2008]

[17] Thomas H. Jones, (1999), An Overview of Non-Linear Junction Detection Technology for Countersurveillance, www.reiusa.net/system/products/NJE-4000/NLJDTech.pdf [accessed 17th Feb. 2008]

[18] Bruce G. Colpitts and Gilles Boiteau, "Harmonic Radar Transceiver Design: miniature Tags for Insect Tracking", IEEE Transactions on Antennas and Propagation, Nov. 2004, Vol. 52, No. 11, pp 2825 – 2831

[19] Professor David Jenn, "*Microwave Devices & Radar*", Lecture Notes, Volume IV, Naval Postgraduate School, www.dcjenn.com/EC4610/VolIV(v4.7.2).pdf [accessed 17th Feb. 2008]

[20] "ORION™, Non-Linear Junction Evaluator: Operating Manual" Version 2.1, (2005), Research Electronics International

[21] S. Kingsley, S. Quegan, "Understanding Radar Systems", McGraw-Hill, 1992

Chapter 3

Rusty Bolt Effect

3.1 Introduction

Radio transmission sites are vulnerable to corrosion and rusting with the passage of time. These deteriorations can appear randomly at different locations which can be scattered over the entire site. They can result in erroneous frequency transmissions. The main reason of the appearance of these impairments is a phenomenon called as the 'Rusty Bolt Effect'.

Radio sites consist naturally of metallic structures. Metals are always covered by an oxide film due to the metal reacting chemically with the oxygen in air. The rate of this oxide formation depends largely on the environment. Any oxide film between metallic contacts will cause non-linearity.

RF currents passing through these junctions would generate harmonics. When RF signals at two frequencies f_1 and f_2 pass through a non-linearity they create signals at their sum and difference frequencies ($f_1 - f_2$) and ($f_1 + f_2$). These are known as 'inter-modulation products'. This generation of inter-modulation products when radio waves interact with rusty parts is called as the **'Rusty Bolt Effect'**. It is also known as passive inter-modulation abbreviated as PIM.

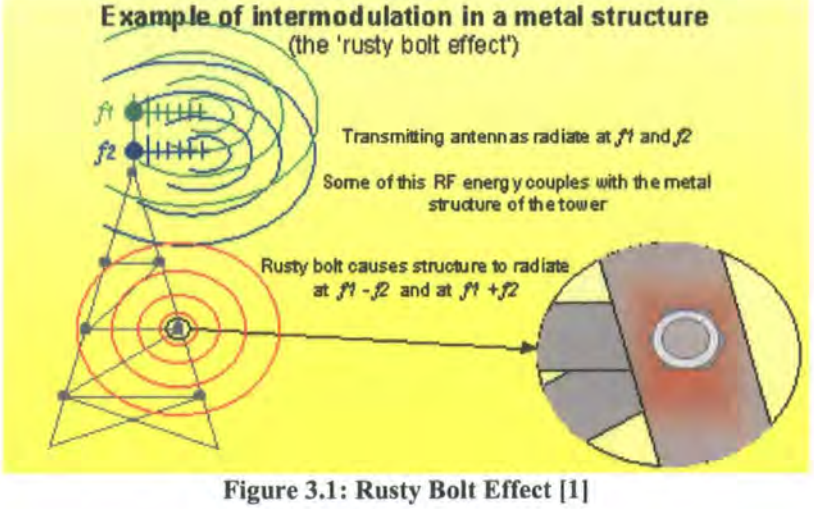


Figure 3.1: Rusty Bolt Effect [1]

Radio spectrum is carefully controlled for optimal usage of the available frequencies so that different services operate in well-defined frequency channels. Inter-modulation creates frequencies that can be difficult to control and would cause interference in channels reserved for other uses. Studies have shown that PIM distortion has caused problems in naval and land communication systems. [2]

PIM distortion was first identified during the early 1940's [3]. Studies were done on systems with antennas containing rusty-bolt parts. The detailed understanding of the phenomenon was not present at that time but the idea of rust prevention to reduce PIM was known.

3.2 Passive Intermodulation

3.2.1 Mathematical Representation

PIM components can be represented using mathematical equations. Consider current passing through a rusty point. The relationship between current and voltage will not be completely linear. It will contain harmonic frequencies and their linear combinations.

The voltage transfer characteristic of a non-linear system can be described as

$$v_i(t) = a_1 v_i(t) + a_2 v_i^2(t) + a_3 v_i^3(t) + \ldots\ldots \quad (3.1)$$

Consider an input signal consisting of two sinusoidal frequencies ω_1 and ω_2 represented as

$$V_i(t) = A_1 \cos(\omega_1 t) + A_2 \cos(\omega_2 t) \quad (3.2)$$

where $\omega = 2\pi f$

The contribution from the second term is given as

$$a_2 v_i^2(t) = a_2[A_1 cos(\omega_1 t) + A_2 cos(\omega_2 t)]^2$$

$$= a_2[A_2 \cos^2(\omega_1 t) + B_2 \cos^2(\omega_2 t)] + 2A_1 A_2 \cos(\omega_1 t)\cos(\omega_2 t)] \quad (3.3)$$

Using the trigonometric relation

$$\cos^2\theta = (1+\cos 2\theta)/2$$

we can reduce Eq. (3.3) to

$$a_2 v_i^2(t) = a_2[A_1^2/2 + A_2^2/2 + 1/2A_1^2\cos 2\omega_1 t + 1/2A_2^2\cos 2\omega_2 t + 2A_1A_2\cos\omega_1 t\cos\omega_2 t] \quad (3.4)$$

Using the trigonometric relation,

$$\cos(\theta_1+\theta_2)+\cos(\theta_1-\theta_2) = 2\cos\theta_1\cos\theta_2$$

Eq. (3.4) can be converted to

$$a_2 v_i^2(t) = a_2[1/2(A_1^2 + A_2^2) + 1/2(A_1^2\cos 2\omega_1 t + A_2^2\cos 2\omega_2 t) + A_1A_2\{\cos(\omega_1+\omega_2)t\cos(\omega_1-\omega_2)t\}] \quad (3.5)$$

We can see the output contains dc component, second harmonics of the input frequencies and intermodulation products at the sum and difference frequencies. Consequently, from Equation (3.5) third, fourth and higher harmonics will be present depending on the non-linear characteristic.

3.2.2 MPT 1331: Code of Practice for Radio Site Engineering

In order to assist radio system designers to minimise such interferences, Ofcom has set some standards for radio site engineering. This set of standards is given in the document 'MPT 1331: Code of Practice for Radio Site Engineering'.

Sections of the code emphasize on different aspects of radio engineering to make the sites operate effectively. Section (3.1) deals with the sources of generation of unwanted inter-modulation products, Section (3.4) describes the Corrosion & Climatic effects on the radio sites and Section (5) describes some recommendations to control inter-modulation and unwanted products. [6]

Any transmission site which is not following these codes would likely cause interference to other users. It is important that radio engineers should check the sites for their compliance with these codes.

If a particular radio site is causing interference due to the rusty-bolt effect, the corroded points must be located to minimize their effect.

3.3 Localization Techniques

3.3.1 Audio Detector – Laboratory Model

An audio detection technique can be used to demonstrate the presence of PIM products. One such system was developed by Bailey [4] as a 'rusty-bolt simulator'.

The system consists of two oscillators of known frequencies representing two signals. These are then mixed using a diode. The resultant frequency is then mixed with another frequency provided by a variable frequency oscillator (VFO). The VFO can be tuned so that the output of the mixer remains between 2 and 8 kHz. This can then be detected using a speaker.

In the lab demonstrating kit mentioned above, two oscillator frequencies were 4 and 6 MHz, the intermodulation frequency was 10 MHz and the VFO was 9.992-9.998 MHz.

3.3.2 Microwave Holographic Imaging

Microwave Holographic Imaging is a well established and an efficient technique to locate PIM sources.

Aspden et al. [5] have used this technique to image intermodulation product sources on reflector antennas. Two signal sources at 7.2GHz and 7.7GHz illuminate a parabolic reflector. It is then scanned for the third order IMP at 8.2GHz. This was done by trying to lock the received signal with a reference signal. The reference IMP signal is generated by a non-linear diode.

3.4 References

[1] Preventing Intermodulation, Radio communications Agency: EMC Awareness, www.ofcom.org.uk/static/archive/ra/topics/research/RAwebPages/Radiocomms/pages/mittech/install/intermod.htm [accessed 2nd May, 2007]

[2] P. L. Lui, "Passive intermodulation interference in communication systems," *Electron. Commun. Eng. J.*, vol. 2, pp. 109–118, June 1990.

[3] *Reuven Shavit*, (2003) "Intermodulation Distortion in Metal Space Frame Radomes", www.l-3com.com/essco/resources/IMP-MSF.pdf [accessed 20th Feb. 2008]

[4] Brandt Bailey, John Schneider (1999), Rusty Bolt Simulator, www.ece.ndsu.nodak.edu/~glower/design/projects/sp99/90_rust.pdf, [accessed 20th Feb. 2008]

[5] Aspden, P.L. Anderson, A.P. Bennett, J.C. , "Microwave holographic imaging of intermodulation product sourcesapplied to reflector antennas", *Antennas and Propagation, 1989. ICAP 89., Sixth International Conference on* , 463-467 vol., Apr 1989

[6] "MPT 1331 : Code of Practice for Radio Site Engineering", June 2001, Flyde Microsystems Ltd.

Chapter 4

System Description

4.1 Introduction

The Harmonic Radar envisioned for the project is a mono-static wideband FMCW radar system with a suitable receive antenna array. The received waves would then be processed using super-resolution algorithms to accurately locate the target. A conceptual diagram of such a system is shown in Fig 4.1.

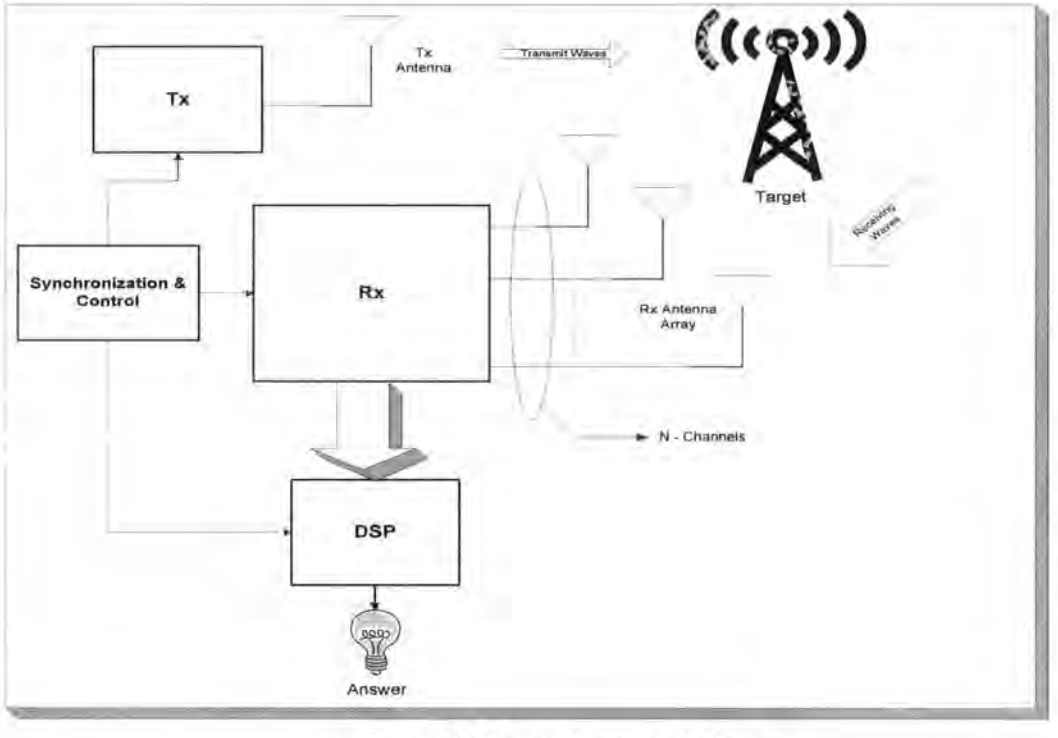


Figure 4.1 System Conceptual Diagram

The 'Tx' block represents the transmitter which transmits a frequency at 1GHz. The 'Rx' block represents the receiver. It can receive the harmonic frequencies at 2GHz, 3GHz and so on through an antenna array. The DSP represents the digital signal processing block which processes the data. There is also a need for a separate sub-system that would control and synchronize between the three distinct blocks.

4.1 System Design

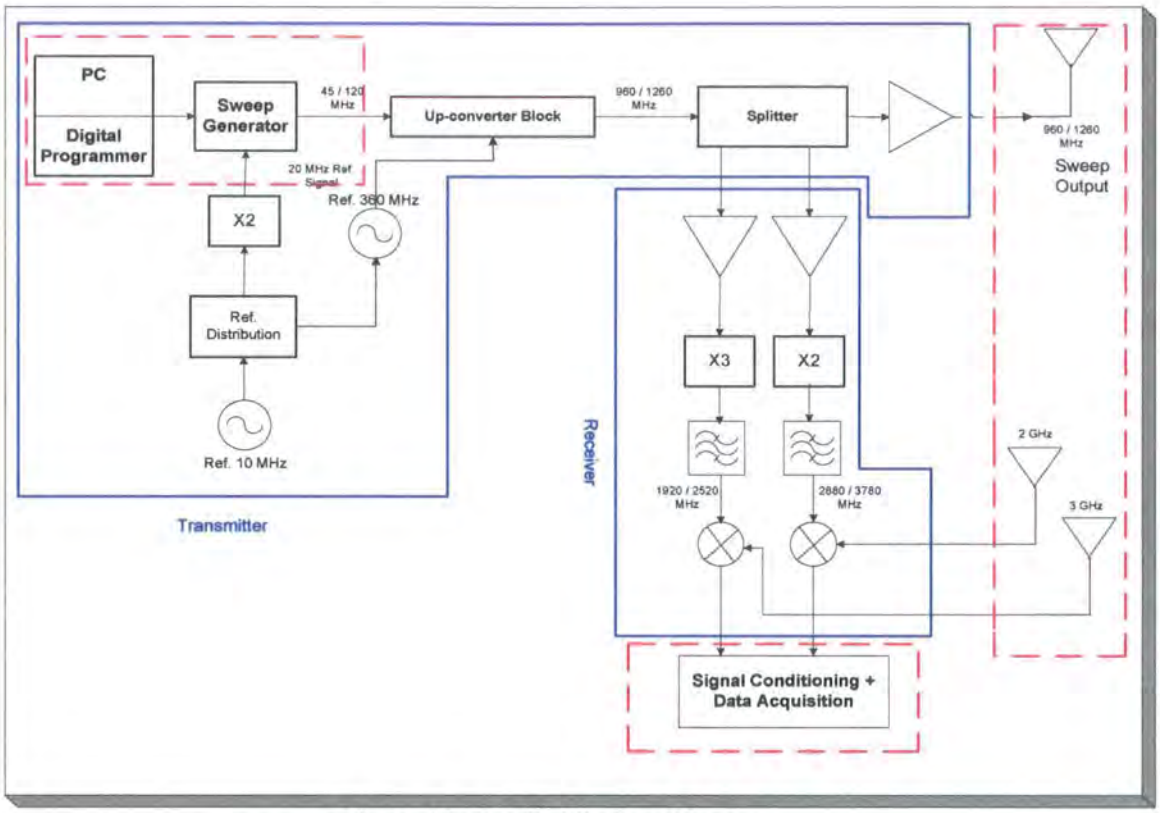


Figure 4.1: Detailed System Diagram

Figure 4.2 shows a detailed system diagram of the transmitter and receiver described in the previous section. The blue lines clearly mark their respective outlines.

4.1.1 Transmitter

The transmitter can be broadly described in two main sub-systems.

1. Chirp Generator
2. Up conversion Block

The chirp generator is the block that produces the sweep wave output. It works on a reference signal of 20MHz and produces an output frequency sweep of 45 – 120 MHz. More on this part is described in section 4.3. The sweep generator works on a reference frequency of 20 MHz. The system reference signal is produced by a 10 MHz oscillator. This signal is then fed to a reference distribution board that provides this reference signal to different components. Two of these signals are shown in the

diagram. The first is multiplied with a 'times two' multiplier to provide the reference signal for the sweep generator. The second provides the reference signal for the PLL synthesizer that produces a signal of 360 MHz which is then used in the up-conversion block.

The up conversion block is a complex sub-system required to up-convert the sweep frequencies coming from the chirp generator to the required transmit sweep centring at 1 GHz. The required output sweep from the up-converter block is 960 – 1260 MHz. This would realistically require stages of up-conversion from the lower frequency to intermediate levels and then to the RF level. This would translate to the use of components of mixers, LO's (local oscillators), multipliers, amplifiers and band-pass filters to remove the spurious frequencies. The RF frequency is then split into three parts. One is transmitted after amplifying through a high power amplifier. The other two are then passed onto to the receiver to be used in the channels for super-heterodyne down-conversion.

4.2.2 Harmonic Receiver

The harmonic receiver primarily consists of two main channels. It will receive the harmonics of the transmit frequency at 2GHz and 3GHz through a receive antenna array. The two down-conversion channels coming from the transmitter are amplified and then up-converted using 'times two' and 'times three' multipliers. The output is then filtered to give a clean signal at 1920 – 2520 MHz and 2880 – 3780 MHz respectively. These two channels are needed to perform super heterodyne down conversion on the receiving harmonic waves. This down-conversion is performed using two mixers.

The output of the mixers would be converted to digital data. It can then be digitally processed using advanced algorithms to obtain useful information. A detailed description on the signal processing can be found in Chapter 5.

4.2.3 Antenna Array

The radar antenna array is an important part of the system. The system requires one transmit antenna and two or more receive antennas. The important factors deciding the selection of antenna are:-

- Bandwidth, and
- Directivity

Bandwidth of any antenna can be defined as the maximum frequency range over which the antenna meets a defined specification. [1] Therefore, the system requirements are a directional antenna covering a wide bandwidth for transmitter (960-1260 MHz) and receiver array (1920-2520 MHz and 2880-3780MHz). Different antennas were studied in literature that would satisfy these two main conditions. A good option would be horn antennas which have been mentioned in literature. [2]

Figure 4.3: Log-Periodic Antennas (900-2600 MHz) [4]

The candidate wideband directional antennas are:-

1. Horn
2. Log-Periodic
3. Discone
4. Modified Patch Antennas
5. Conical Spiral with cavity
6. Log Spiral with cavity

Log-periodic antennas were ordered as shown in Figure 4.3 depending on the availability and price.

4.3 Chirp Generator

The chirp or sweep generator produces a signal that varies in frequency from 45 MHz to 120 MHz and remains constant in amplitude. It consists of two main parts:-

1. DDFS (Direct Digital Frequency Synthesizer) Board
2. Digital Programmer

A block diagram of the digital programmer with the DDFS board is shown in Fig. 4.4

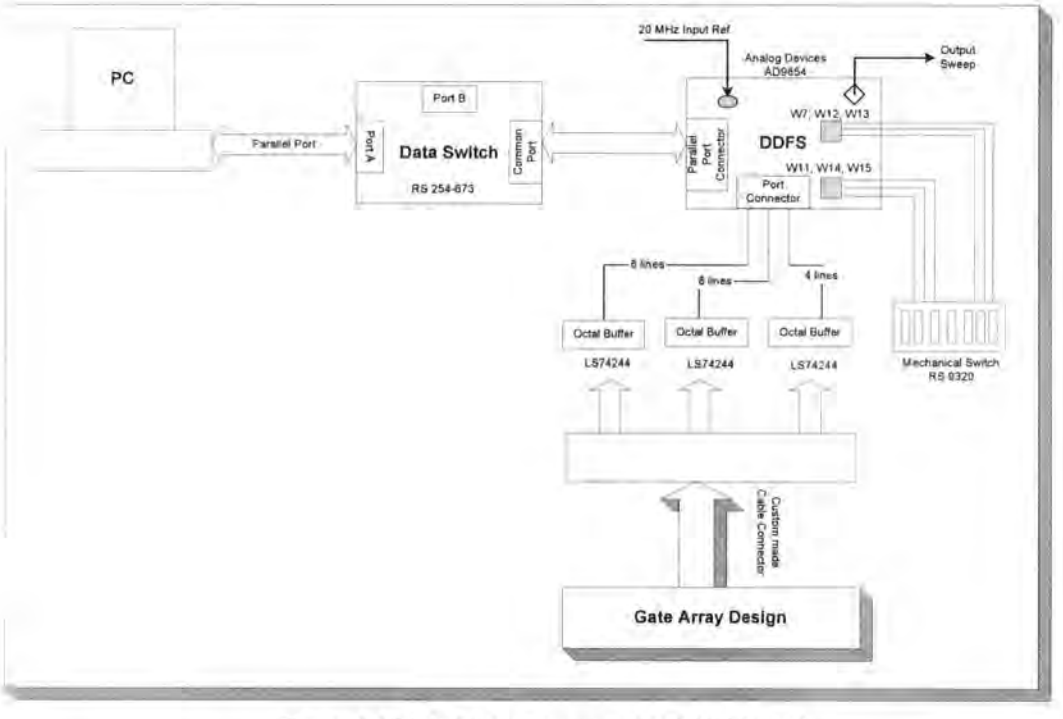


Figure 4.4 Digital Programmer with DDFS board

4.3.1 DDFS Board - AD9854

DDFS - Direct Digital Frequency synthesis is a method to create arbitrary waveforms and frequencies from a single, fixed source frequency using digital processing blocks. The DDS board used in the chirp generator is Analog Devices AD 9854. The AD9854 can generate highly stable, frequency-phase, amplitude-programmable sine and cosine

outputs that can be used in many applications. Some of the available features of this board are:-

Programmable Ref. CLK Multiplier	-	4 to 20
Control Register Data	-	8 bits
Control Register Address	-	6 bits
Frequency Resolution	-	48 bits
Frequency Step Resolution	-	48 bits

The AD9854 has five programmable operational modes. Three bits in the control register called as Mode 0, Mode 1 and Mode 2 must be programmed to select the type of operation. The different mode bits selections and their result are shown in table below:

Mode 2	**Mode 1**	**Mode 0**	**Result**
0	0	0	Single Tone
0	0	1	FSK
0	1	0	Ramped FSK
0	1	1	Chirp
1	0	0	BPSK

The AD9854 supports linear as well as nonlinear FM sweep patterns. The different parameters needed to program the chirp mode are:-

- Start Frequency Tuning Word (FTW 1)
- Time steps (Ramp rate)
- Frequency steps (Delta frequency)
- I/O UD Clock
- Clear Accumulator 1 Bit (CLR ACC1)

Other controlling parameters present on the DDS board are:-

- Clear Accumulator 2 Bit (CLR ACC2)
- FSK bit (Frequency Shift Keying)
- BPSK bit (Binary Phase Shift Keying)

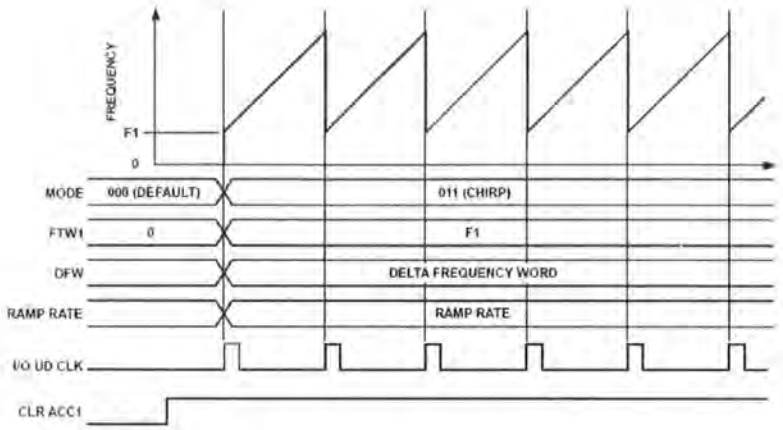


Figure 4.5 Chirp Mode Operation [3]

Fig. 4.5 illustrates the functioning of a simple chirp mode. It is important to note that FTW1 is only a starting point for FM chirp. There is no built-in restraint requiring a return to FTW1. However, instant return to FTW1 can be achieved by interrupting the current chirp, reset the frequency back to FTW1, and continue the chirp at the previously programmed rate and direction.

A program was developed in Visual Basic that would calculate all the required Chirp mode parameters. The user interface of this program is shown in Figure 4.6

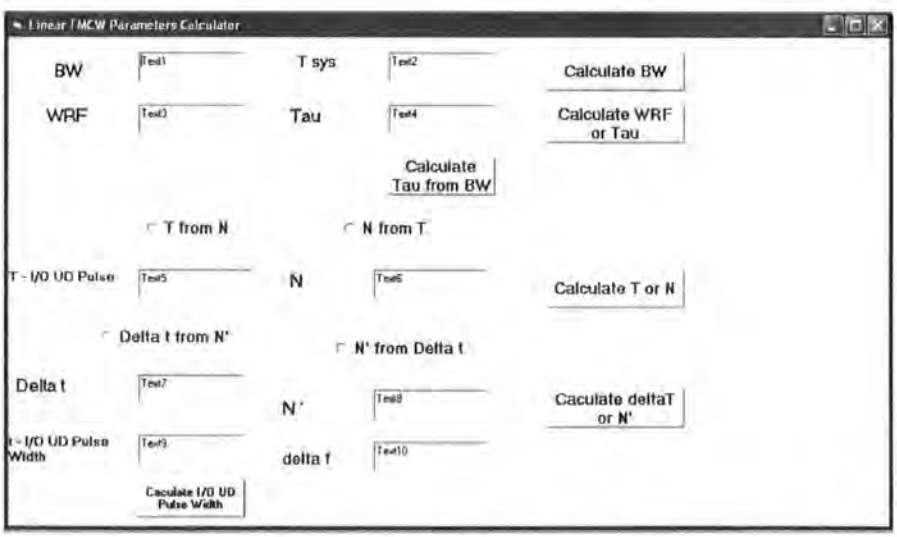


Figure 4.6: User Interface of Chirp Mode Parameter Calculator

This program uses a set of standard equations available in literature to calculate the required parameters. The equations are described below:

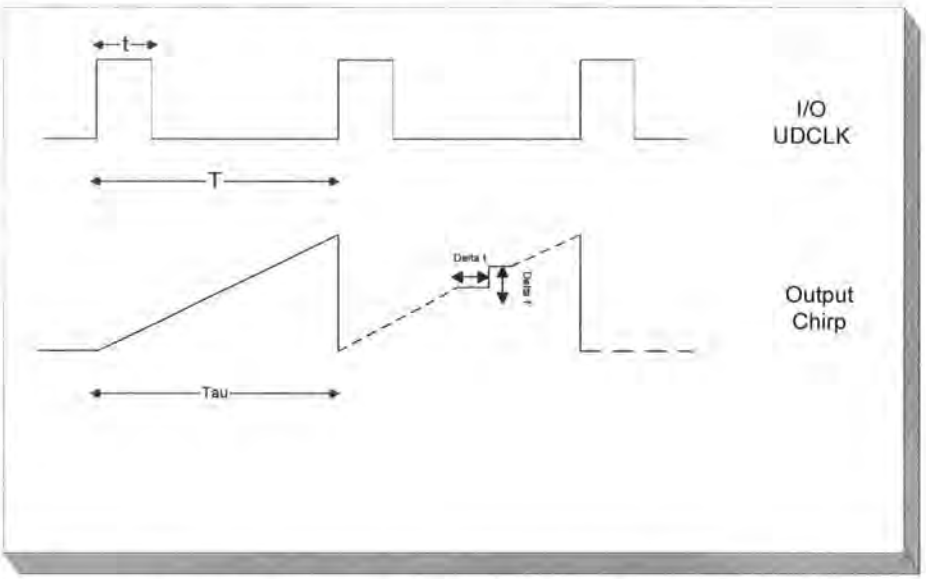


Figure 4.7: Relationship of I/OUDCLK and Output Chirp

All the equations that follow refer to figure 4.7 above.

The time period T of the chirp is related to the system clock by the relation

$$T = (N+1)(T_{sys} \times 2) \tag{4.1}$$

where T_{sys} is the DDFS system clock.

The time step value in Fig. 4.5 is Δt. It is related to the system clock with a counter N'. Their relation is given as

$$\Delta t = (N'+1)(T_{sys}) \tag{4.2}$$

The width of the I/O UDCLK pulse is related to the system clock by

$$t = 8T_{sys} \tag{4.3}$$

By definition of BW,

$$BW = n\Delta f \tag{4.4}$$

The relation of time period and step-time for the chirp is given as

$$n = \frac{\tau}{\Delta t} \tag{4.5}$$

Replacing Eq. (4.5) into Eq. (4.4) gives us

$$BW = \tau \frac{\Delta f}{\Delta t} \tag{4.6}$$

But since

$$\tau = T \tag{4.7}$$

Therefore we have,

$$\frac{BW}{T} = \frac{\Delta f}{\Delta t} \tag{4.8}$$

These equations have been used in the VB program to calculate the required parameters to generate a chirp depending on the parameters entered by the user.

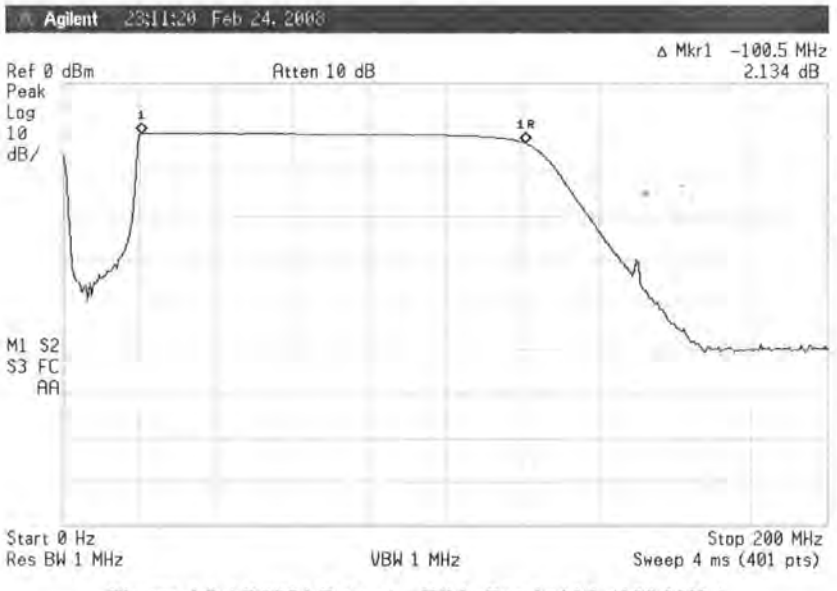


Figure 4.8 AD9854 Output – Wide Bandwidth (100 MHz)

AD9854 is capable of producing a wide bandwidth clean signal. Fig 4.8 shows the output from the DDS board on a spectrum analyzer. It has constant amplitude from 10 to 110 MHz giving a clean bandwidth of around 100 MHz.

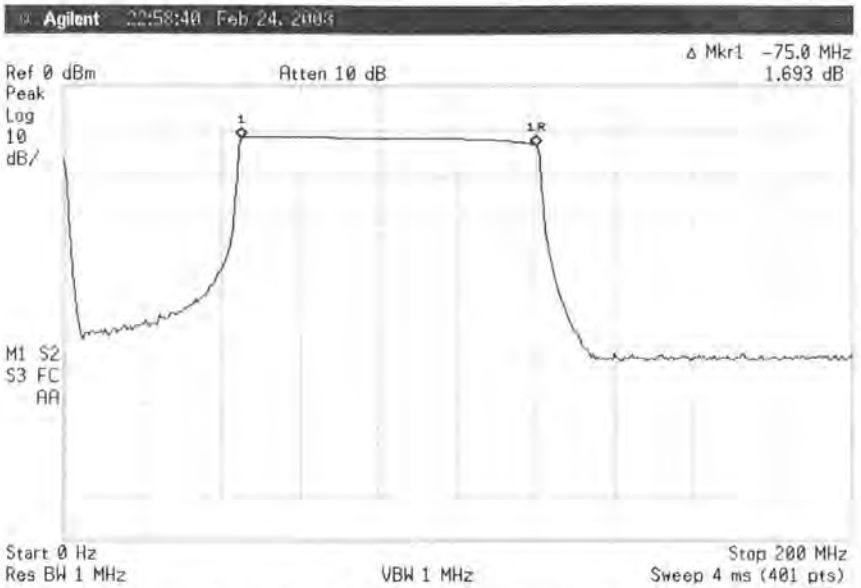


Figure 4.9 AD9854 Output – 45/120 MHz

Fig 4.9 shows the output from AD9854 when operating in the sweep generator. It produces a sweep from 45 MHz to 120 MHz with almost constant amplitude. The bandwidth thus achieved is 75 MHz.

5.3.1.1 Waveform Designing

Waveform designing is an important area of research in radar signals. The type and quality of information received by radar depend in part on the waveform it transmits. Efficient waveforms minimize the ambiguities present in FMCW radar system. One such practical solution has been presented by Salous [5] where a multiple WRF signal was introduced as a solution to range/Doppler ambiguity. This goes back to the pioneering work by Poole [6] where he introduced a multi-cell waveform.

More advanced waveforms than the basic linear chirp signal were studied to be implemented with the DDS AD9854. The result has been summarized in the table below:-

Waveforms	AD9854 Signals Used
1. Linear Chirp	I/O UDCLK, CLRACC1
2. Triangular Modulation	I/O UDCLK, FSK/BPSK bit (Ramped

	FSK Mode)
3. Linear FM Ranging	I/O UDCLK, CLRACC2 (A variant of Chirp Mode)
4. Segmented Linear FM	I/O UDCLK, FSK/BPSK bit, CLRACC2
5. Multi-cell Chirp	Not possible without adding external hardware circuit
6. Staggered WRF	This can be programmed in the Chirp mode using the signals I/O UDCLK, CLRACC1. The critical part would be programming new Δf and/or Δt values after every single chirp.
7. Multiple WRF	This is the same as above except that new Δf and/or Δt values must be programmed after a number of individual chirps as defined by the WRF.

5.3.1.2 Phase Coherency

An important consideration is the phase component of the transmitting signal. The Chirp mode by default has an internal problem for our applications. The CLR ACC1 signal only clears the Frequency Accumulator. It does not affect the Phase Accumulator value. The frequency changes of the DDS become phase continuous, but not phase coherent. When a new frequency is programmed into the DDS, the next phase will simply be incremental with respect to the last phase value in the phase accumulator, and therefore the output sine wave will be phase continuous.

This phase continuity will cause a problem in the harmonic detector. There will always be a phase component present in the transmitting signal and therefore in the receiving signal. This phase component will appear as a Doppler or a movement / velocity in the end data. Thus, if the DDS is used as such, it will show a Doppler component of an object without any actual movement present.

Some modifications either in the source code of the program or with some additional external hardware circuitry is needed to clear the phase accumulator along with the frequency accumulator.

The CLR ACC2 control bit (Register Address 1F hex) is available to clear both the frequency accumulator (ACC1) and the phase accumulator (ACC2). When this bit is set high, the output of the phase accumulator results in 0 Hz output from the DDS. As long as this bit is set high, the frequency and phase accumulators are cleared, resulting in 0 Hz output. To return to the previous DDS operation, CLR ACC2 must be set to logic low. This bit can be used to clear the phase of the signal along with the amplitude.

However, we cannot afford to keep the CLR ACC2 bit high for a long duration. Otherwise it will result in an ON/OFF Keying pattern.

Solutions were designed so as to clear the ACC2 bit before the start of the next chirp. This would require a number of steps as follows:

- The DDS board should be programmed from a circuitry controlled by the user. For this, a cable was built to connect the junction pins on the board with a test board. These pins can then be used to control the DDS by sending in appropriate control commands at the correct addresses.
- The CLR ACC2 bit requires a signal that would clear the output in between the end of the first chirp and start of the next chirp. This 'high' time should be as low as possible so as not to severely distort the shape of the end waveform.

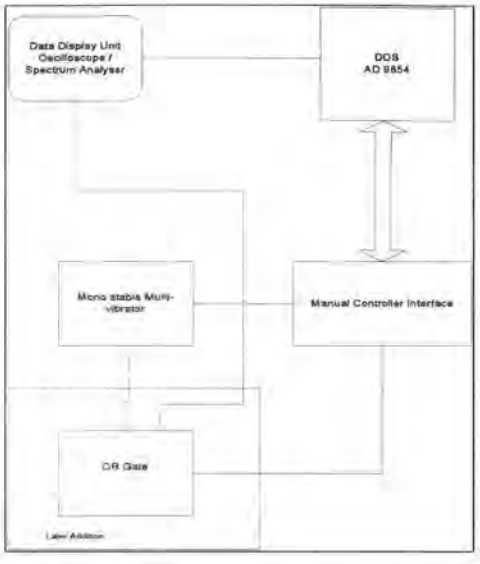


Figure 4.10: Block diagram of external hardware to generate the CLR ACC2 signal

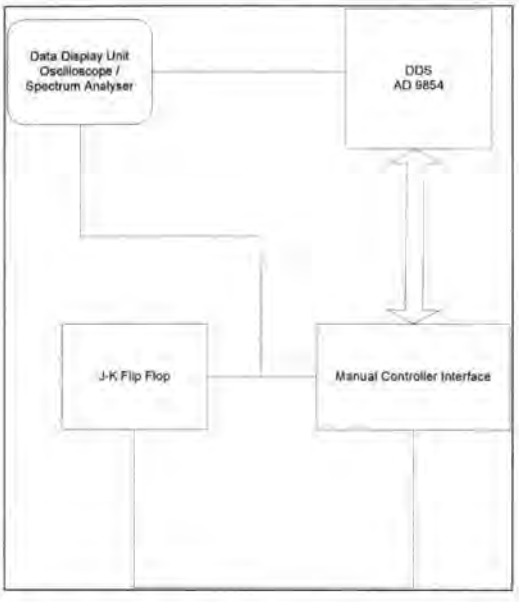


Figure 4.11: Block diagram of external hardware to generate the CLR ACC2 signal

The circuit performs the following functions:

1. All the signals provided in the junction J10 on the DDS board are connected to a test board via a cable.
2. The Manual Controller allows the user to send control words at appropriate addresses.
3. The I/O Update Clock is then sent to a mono stable multi-vibrator (74121) which generates a longer version of the UD CLK pulse width depending on the resistor and capacitor values selected.
4. This longer pulse was then sent as CLR ACC2 signal back to the DDS board through the manual controller.

This solution did not work. The reason for this was that the output of the multi-vibrator occurs after a certain amount of delay from the input UD CLK signal. For the phase accumulator to be cleared, the two signals I/O UD CLK and CLR ACC2 must start at exactly the same time.

A slight modification in the circuit can achieve this quite easily. An additional OR gate was added to the circuit as shown in Fig. 4.10

1. One input to the OR gate is the I/O UD CLK signal.
2. The other input is the output of the multi vibrator signal.
3. The output signal from the OR gate would be a wider pulse that starts at exactly the same time as the I/O UD CLK. This signal was then sent as the CLR ACC2 signal to the DDS board through the manual controller.

This solution also failed to clear the phase accumulator.

In order to diagnose the problem, an attempt was made to provide such signals which are the exact replica as that in Fig 4.11. The CLR ACC2 signal has a period that is the exact twice of the I/O UD CLK period. Such a signal can be generated using a J-K Flip Flop. The block diagram of such a circuit is shown in Fig. 4.11.

1. The I/O UD CLK signal is sent from the manual controller to a J-K Flip Flop.
2. The J-K Flip Flop is set in a toggle mode.
3. It then generates a signal that has a period exact twice that of the I/O UD CLK.
4. This signal is then sent as the CLR ACC2 signal to the DDS board via the manual controller.

This design does clear the phase at the start of every chirp. However, it also clears the amplitude for the whole pulse duration and there is no chirp during this time. A gate array design has been envisioned to solve this problem. Further details of it are given in section 4.3.2.

4.3.2 Digital Programmer

The digital programmer is supposed to program the DDFS board according to the parameters selected by the user. A further task in this sweep generator is to reset the phase after every sweep as identified in section 4.3.1.

There can be two variants to this digital programmer. One based on a 'C' program being controlled through a PC or the other being a gate array design. Efforts were made on both variants of the design. Both designs are explained below:-

5.3.2.1 C based Design

The 'C' code should emulate the entire program execution of the DDS AD9854. A flowchart of the steps carried out to program the DDS board is shown in Figure 4.12. There are three main parts to program the DDFS

- Accessing the parallel port
- Calculating the chirp parameters
- Sending appropriate data with address and control signals to program the DDFS.

Such a 'C' language code was written using the DOS based C-compiler TC++v3. The code listing of this is provided in A 3.1

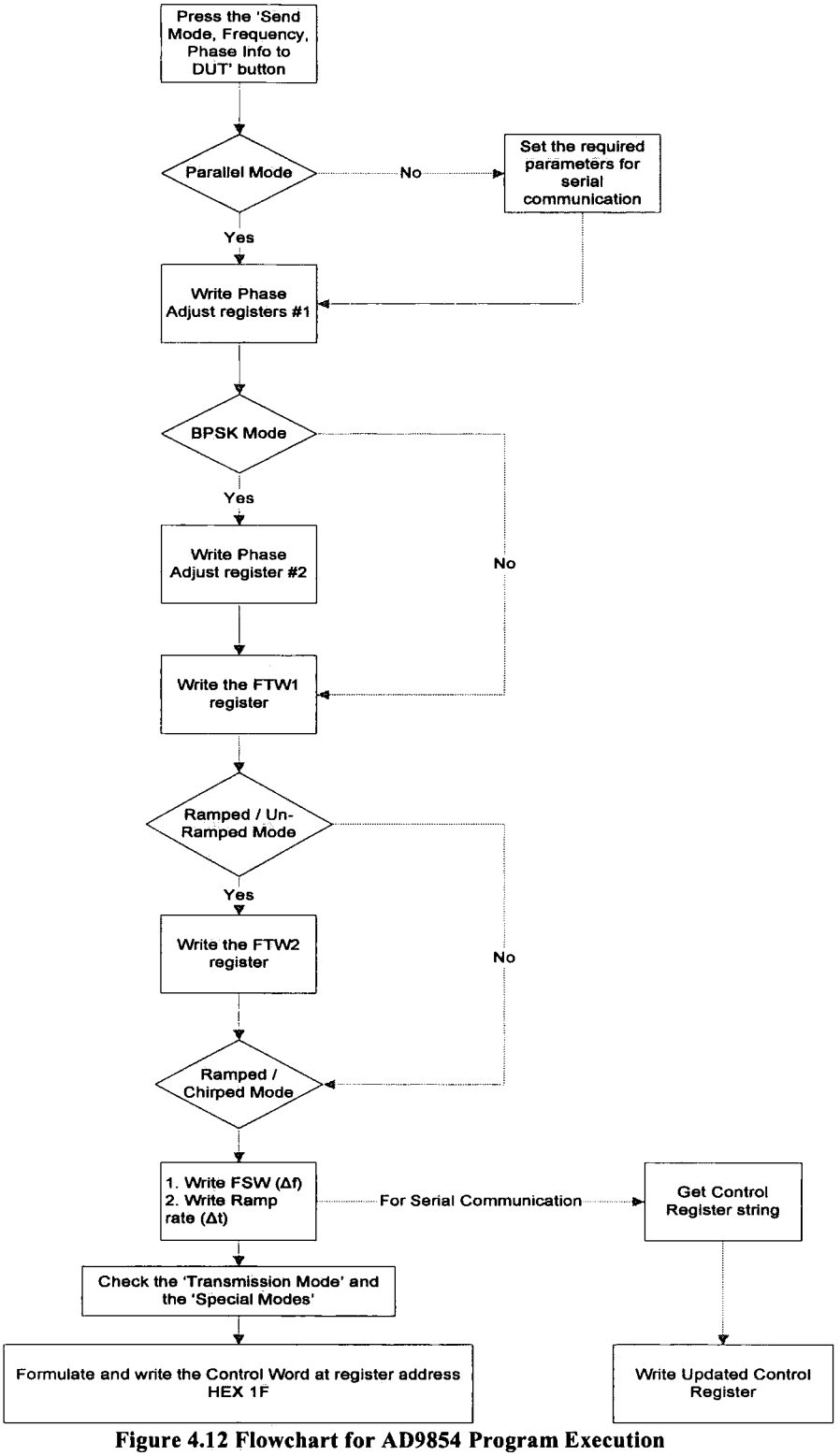


Figure 4.12 Flowchart for AD9854 Program Execution

The code has limited success in its limitation. The data send by the programmer (laptop) can be traced on the registers and buffers on the DDFS board but it is not

being read by the AD9854 chip. One possible reason might be the RD (read) and WR (write) signals. Their timing is crucial in writing any data to the chip. The code controls these signals and is using an arbitrary 'delay' value. It has been decided that the gate array design would be pursued to completion.

5.3.2.2 Gate Array Design

There are two variants of the Gate Array Design.

- Full Extensive VHDL code design

The VHDL code was designed by another worker. The design consists of 7 functional modules, each concerned with a specific task. Each module carries out its task in consequence to external inputs or from one or more of the other modules. Many of the individual block modules work but some modules and their integrated simulation are not entirely correct.

Some reasons attributed to this are as follows:-

a. Code was previously developed for Xilinx compiler tools
b. Some commands were specific for the Xilinx FPGAs which are not valid in Altera's Quartus compiler
c. Some of the numerical factors used in the design are specific to the board and the clock frequency being tested for. It was mistakenly taken as absolute values.

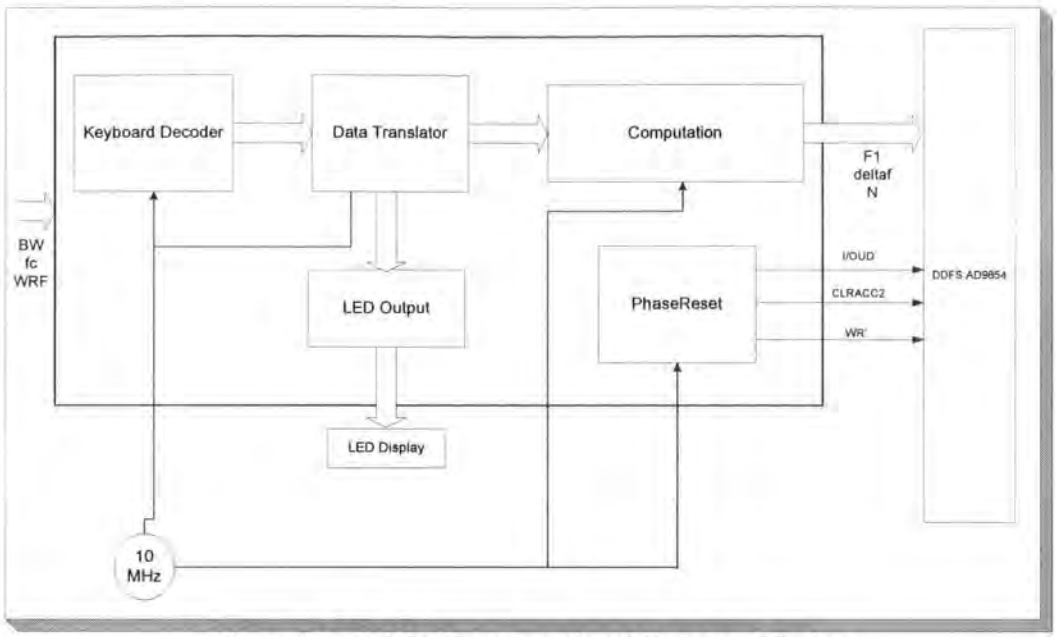


Figure 4.13 Modular Representation of Gate Array Design

- Simpler Design (VHDL + Schematics)

Figure 4.13 shows a design that is almost in state of completion at the time of writing this document. This gate array design consists of a mixture of VHDL code and hardware schematics. They can be viewed in Appendices 1-2.

The modules 'keyboard decoder', 'data translator' and 'LED Output' were designed by another worker. They were tested successfully with the new compiler and simulated. The two blocks 'PhaseReset' and 'Computation' were designed. They are described as follows:-

Phase Reset Module

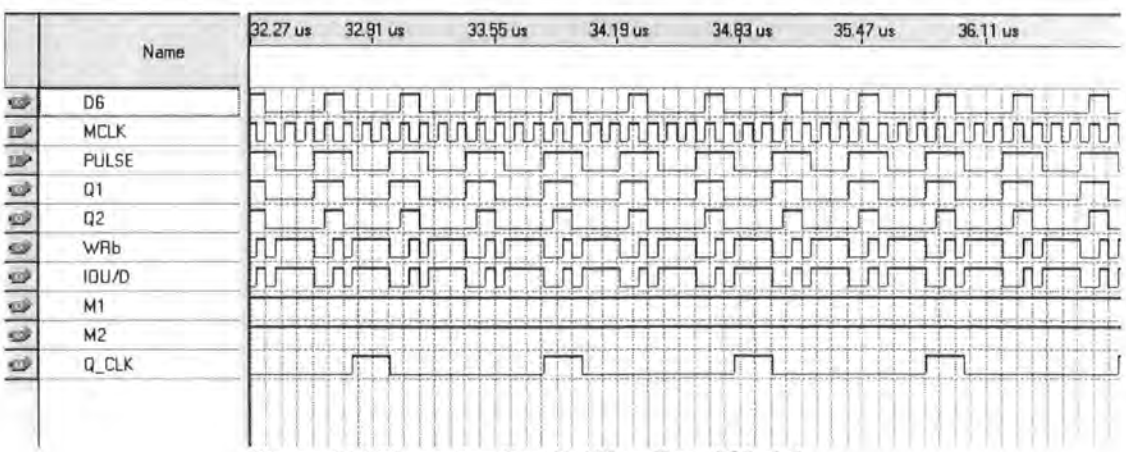


Figure 4.14 Simulation Result: 'PhaseReset' Module

The 'PhaseReset' module is designed using a hardware schematic as shown in Appendix A1.1. Its simulation result is shown in Figure 4.14 above. The three output signals going to the DDFS are I/OUD, WRb and D6 (CLRACC2). The other outputs shown in the simulation were tapped to observe the behaviour of the circuit.

Computation Module

The computation module is supposed to calculate the values required to program the DDFS. It takes the values coming from the Data Translator module and is supposed to output to the DDFS. The schematic is shown in A1.2

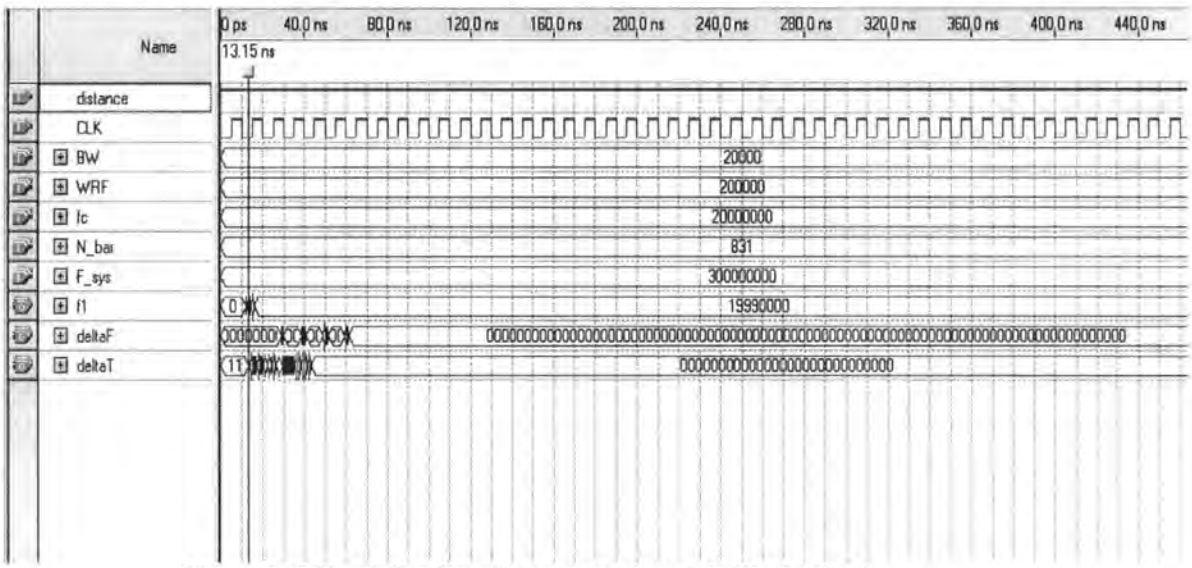


Figure 4.15 Simulation Result: 'compnew' module (Real values)

Figure 4.15 shows the simulation result of this module when the input numerals are actual values for the board. The zero result shows either the circuit is not working or the values are going out of range for the present design. To check the validity of the circuit, it was simulated for simpler values. The simulation result is shown in Fig. 4.16 below.

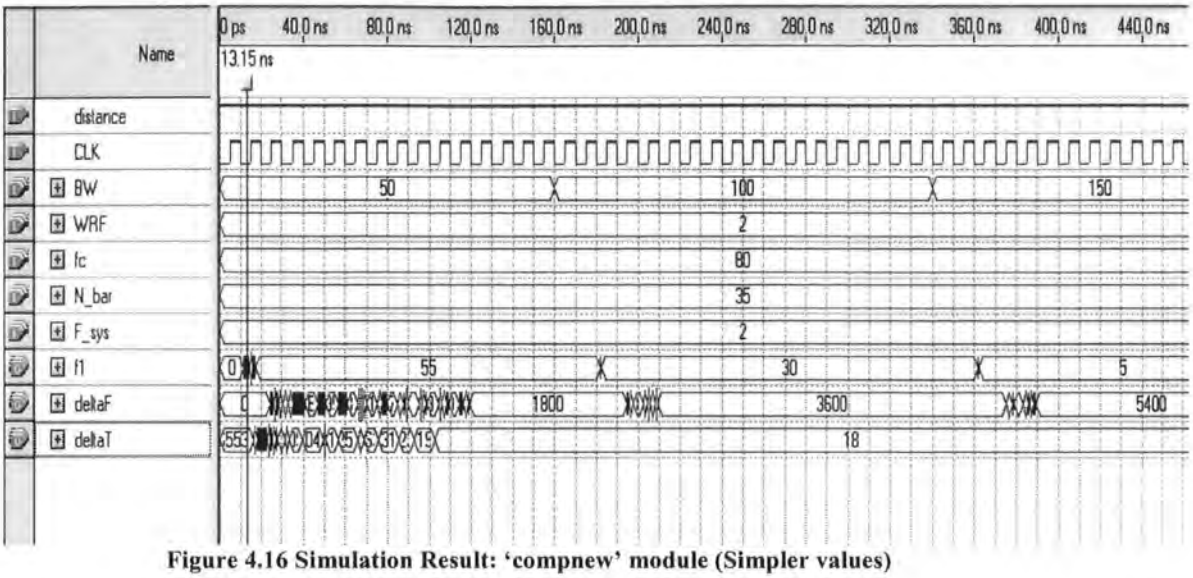
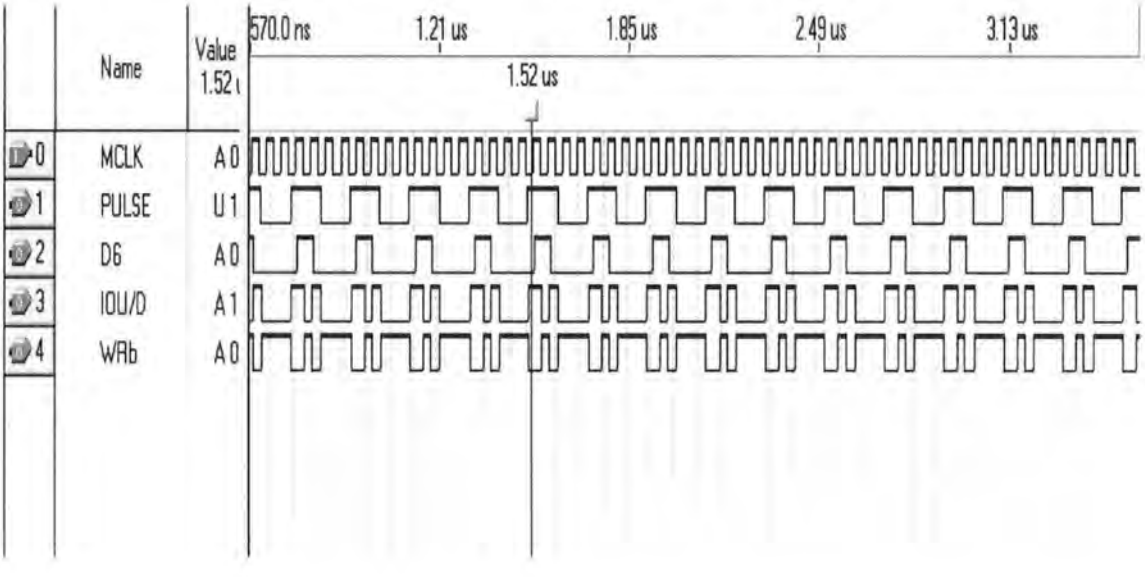


Figure 4.16 Simulation Result: 'compnew' module (Simpler values)

The output signals are correct which validates the working of the circuit. An attempt was also made to design this module using VHDL code given in Appendix A2.1. There is a numerical factor that goes outside the 'integer' range as defined by the compiler. It is a probable reason for a zero output.

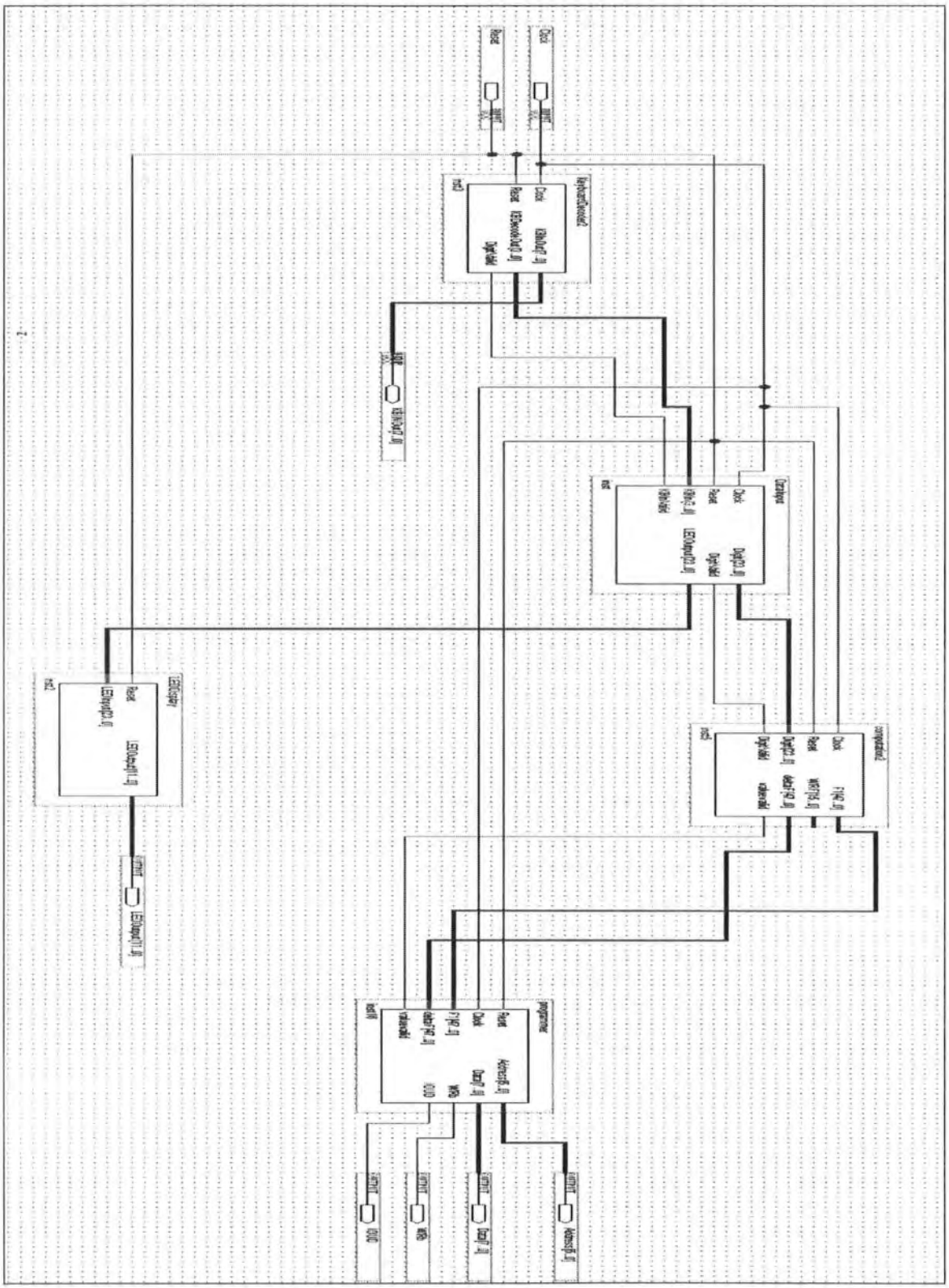
'PULSE' Control Signal Generation for Phase-Reset Module



Comments:

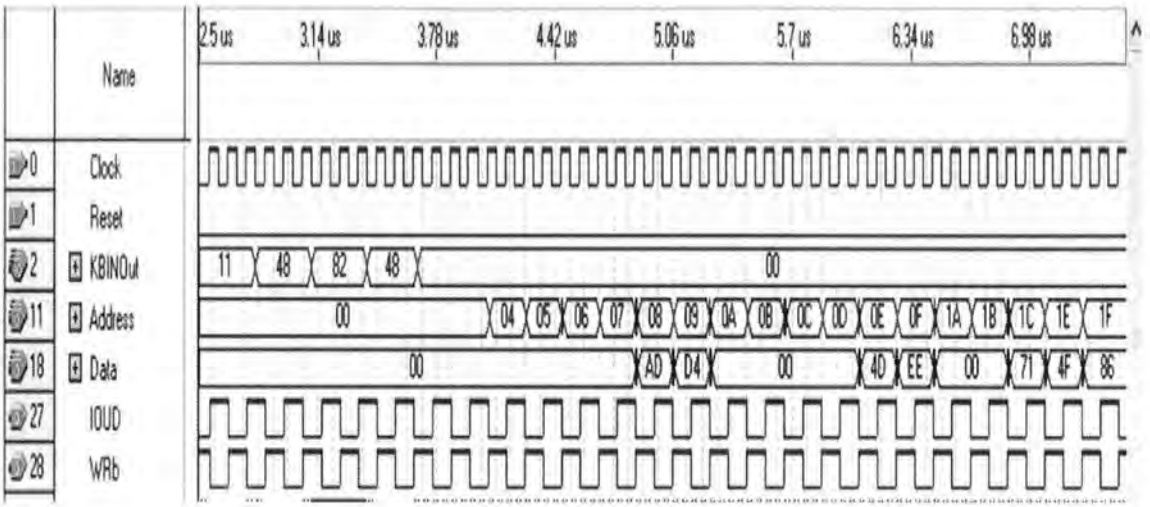
1. The simulation is for 10 MHz clock.

2. A MAX7000 family device is sufficient to compile this design.
3. The 'Pulse' generator block has been added which is a VHDL code for a 'divide-by-4 clock frequency' design.

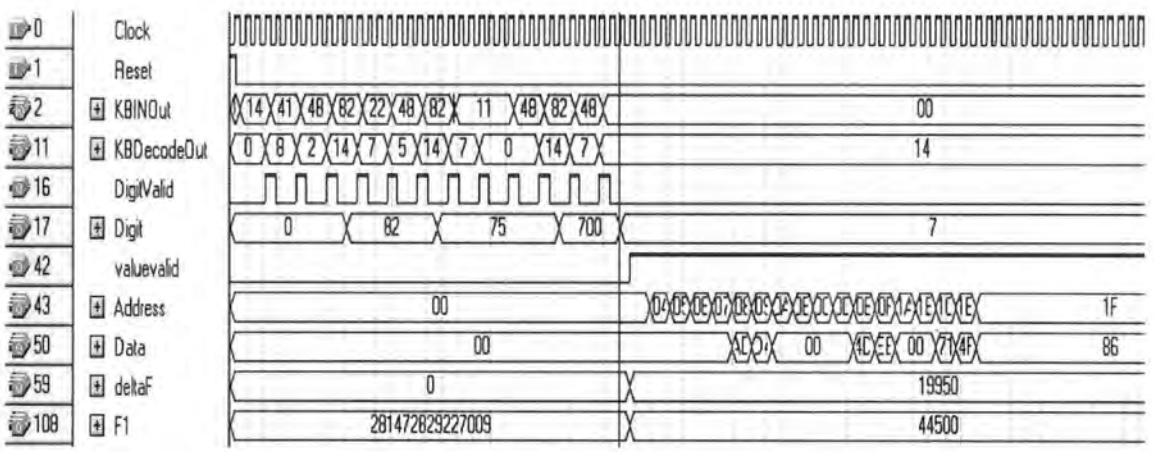


Simulation Results

The final simulation result for the design is shown below.



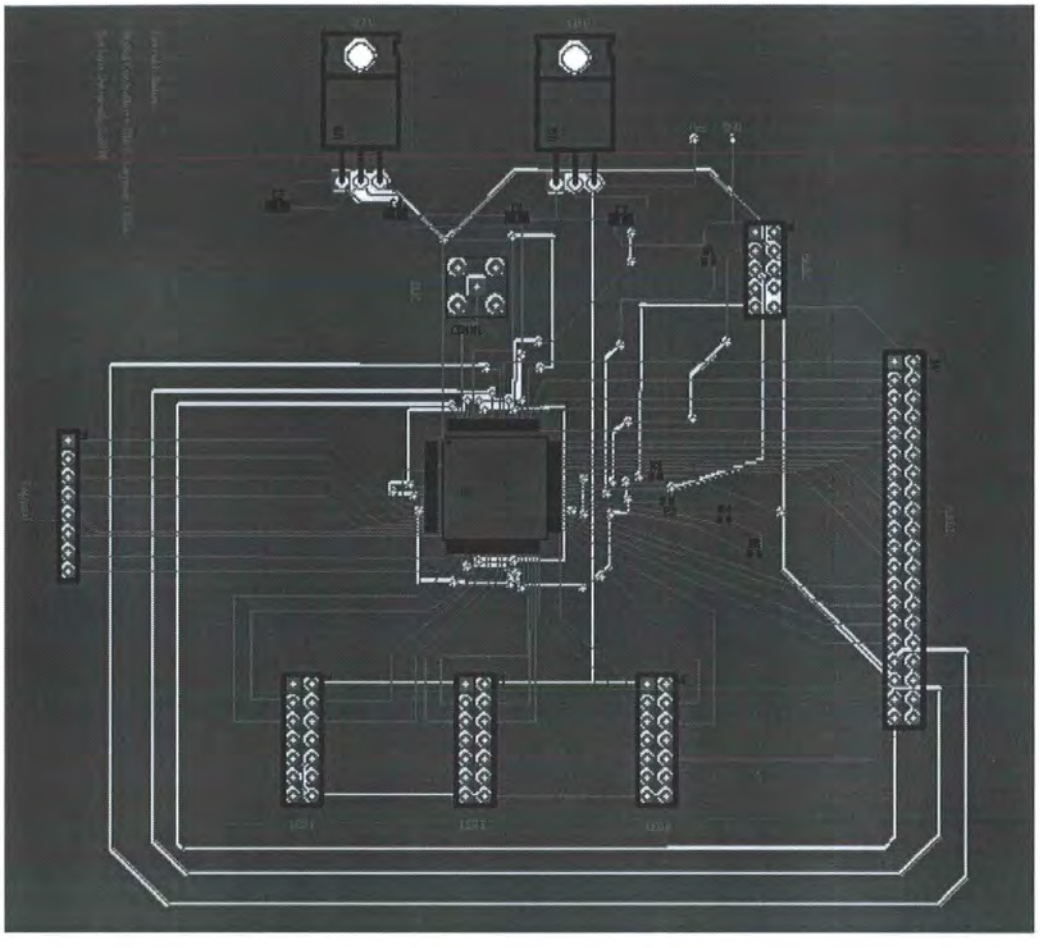
To clarify the inputs and outputs, a test output "KBDecodeOut" is added to the result. The output is shown in the following figures:-



Comments:

1. The CLK is 10 MHz.
2. The simulated sequence from the keypad is (8,2,E),(7,5,E),(7,0,0,E) and (7,E). This results in fc = 82 MHz, BW = 75 MHz and WRF = 700 Hz.
3. The Device selected is Cyclone EP1C3T100C8

PCB Layout



Parts / Components List:

No.	Device	Package	Value	Quantity
1	Altera Cyclone	TQFP100-14	EP1C3T100C8	1
2	10 Pin Connector		2x5	2
3	14 Pin Connector		2x7	3
4	8 Pin Connector		1x8	1
5	Tri State Buffer	DIP20	74244	3
6	40 Pin-Connector	HDR40_2	2x20	1
7	Voltage Regulator	TO220	LM317-1.8	1
8	Voltage Regulator	TO220	LM317-3.3	1
9	R1	0603	1K	1

10	R2, R3, R4, R5	0603	10K	1 each
11	C1, C2, C3, C4	0805	10uF	1 each

4.4 References

[1] S. Davies, H. R. Holliday, "Wideband Antennas – a Historical Perspective", *Ideas Engineered*, Q-par Angus Ltd.

[2] Schantz, "Introduction to Ultra-wideband antennas", IEEE 2003.

[3] Analog Devices AD9854 Datasheet, pg. 26

[4] Printed Circuit Board Antennas - Log Periodic, Kent Electronics, http://www.wa5vjb.com/products1.html [Last accessed: 2nd March, 2008]

[5] Musa M., Salous S., "Ambiguity Elimination in HF Radar Systems", *IEE Proceedings – Radar, Sonar Navigation*, Aug. 2000, Vol. 147, No. 4, pp. 182 -188.

[6] Poole A. W. V., " Advanced sounding. The FMCW alternative", *Radio Science*, December 1985, Vol. 20, No. 6, pp 1609-1619.

Chapter 5

Data Processing

An important part of the harmonic detector is the data processing part after the receiver. The objective of processing the data is to be able to accurately locate the points generating harmonic frequencies. The accuracy and efficiency of the result depends upon the technique used.

Two such methods studies so far are being described below.

5.1 Heterodyne detection

Barrick [1] has presented the technique of heterodyne detection FFT processing in his 1973 article "FM/CW Radar Signals and Digital Processing". In the 'heterodyne detection' method, RF channel is generated locally having the same band and the same repetition rate as that of the transmitted signal. Mixing this replica of the transmitted signal with the received signal gives difference and sum frequencies. The sum frequencies can be filtered out using a low pass filter. The difference frequencies, also called as *beat notes*, can be used to extract the time delay and Doppler information. The time delay will give the range and Doppler will lead to the velocity information. The output signal is compressed and the frequency of the beat spectrum can be on the order of a few tens of *kHz*. [2]

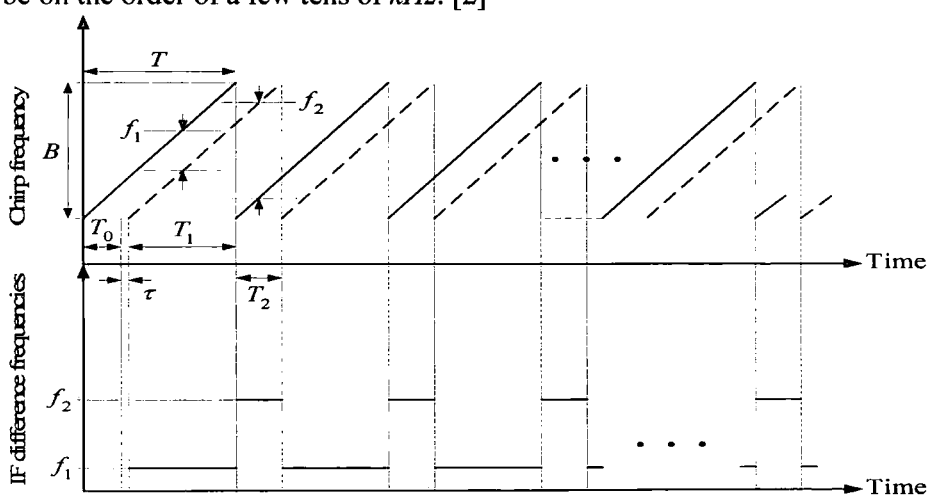


Figure 5.1 : Swept frequency versus time and the frequency of beat notes at the output of the detector.

The mathematical expression for the transmitted chirp signal was defined in equations (3.1) and (3.2). In a modified form, the transmitted and received waveform can be represented as

$$V_T(t) = \cos[w_c t + \pi B f_r t^2] = \cos\phi_T(t) \tag{5.1}$$

$$\begin{aligned} V_R(t) &= A V_T(t - t_d) \\ &= A\cos[w_c(t - t_d) + \pi B f_r (t - t_d)^2] \\ &= \cos\phi_T(t - t_d) \end{aligned} \tag{5.2}$$

The heterodyne mixing results in a mathematical multiplication of the two signals. The higher frequency terms are filtered out. The beat note phase can be written as

$$\phi_I(t_i) = \phi_T(t_i - t_d) - \phi_T(t_i) \tag{5.3}$$

where ϕ_I is beat or intermediate phase

t_i is the internal time within the pulse

t_d is the delay time

After some long derivations and calculations, this beat note phase can be written as

$$\phi_I(t_i) = \phi_o - 2\pi[2\frac{v}{c} f_c + B f_r t_o]t_i \tag{5.4}$$

where ϕ_o is the initial phase delay corresponding to initial time delay t_o. This is within a pulse i.e. the time period is

$$-\frac{T_r}{2} < t < \frac{T_r}{2}$$

The beat frequency is

$$f_I = \frac{2v}{c} f_c + B f_r t_o \tag{5.5}$$

The first term in the above equation is due to target velocity (Doppler) and the second term is due to delay (range) of the target.

The above equation is valid within a pulse. For n-pulses the frequency can be represented as

$$f_{I_n} = \frac{2v}{c} f_c + B f_r t_o + \frac{2v}{c} Bn \tag{5.6}$$

5.2 Double FFT Digital Processing

Essentially two methods exist for extracting time delay (range) and Doppler shift (velocity) channel information [3]. The first technique employs double FFT processing; where an initial FFT is carried out over one sweep in order to extract time delay information, and another FFT is carried out over N sweeps for each single time delay bin in order to determine Doppler shift information. The second technique employs a long FFT process over N sweeps in order to provide simultaneous delay/Doppler channel information. Both techniques are identical and have the same number of computational steps [1]. Here only the first technique is described in detail as that has been studied so far. It may also be noted that for the harmonic radar Doppler processing is not needed as the target (rusty-bolts) are not moving.

For a single transmission path, the Fourier transform over a single sweep n is given by

$$V_{1,n}(f) = \int_{-T/2}^{T/2} \left\{ A\cos\left[\phi_{1,n}(t)\right] \right\} e^{-j2\pi ft} dt \tag{5.7}$$

using Euler's definitions: $\cos x = (e^{jx} + e^{-jx})/2$, $\sin x = (e^{jx} - e^{-jx})/2j$, and $\int e^{ax} dx = e^{ax}/a$, equation (5.30) becomes

$$V_{1,n}(f)=\frac{AT}{2}\left\{\frac{\sin\left(2\pi\left(f-f_{1,n}\right)T/2\right)}{\left(2\pi\left(f-f_{1,n}\right)T/2\right)}e^{-j\phi_0+j2\pi f_0\frac{v}{c}nT}\right\}$$
$$+\frac{AT}{2}\left\{\frac{\sin\left(2\pi\left(f+f_{1,n}\right)T/2\right)}{\left(2\pi\left(f+f_{1,n}\right)T/2\right)}e^{+j\phi_0-j2\pi f_0\frac{v}{c}nT}\right\} \qquad (5.8)$$

Therefore the detected signal spectrum is a $\sin(x)/x$ pulse with a width of T.

Any practical system measures the data as samples of the received signal (obtained by employing an ADC – Analogue to Digital Converter). The FFT is required to obtain the discrete spectrum of the $\sin(x)/x$ pulses. If M denotes the number of points used in the FFT, then the output of the FFT performed on M samples over a sweep will result in M discrete frequency points whose values range from $-F_S/2$ to $F_S/2$, where F_S denotes the sampling rate. By symmetry the negative frequency components can be ignored, leaving $M/2$ beat frequency bins. [3]

The first FFT process on M samples within a sweep gives $M/2$ time delay bins per sweep. The FFT process carried out over N sweeps and digitally storing the values for each sweep in a row of a data matrix gives an $M/2 \times N$ matrix containing both delay and Doppler information. This is illustrated in the following figure.

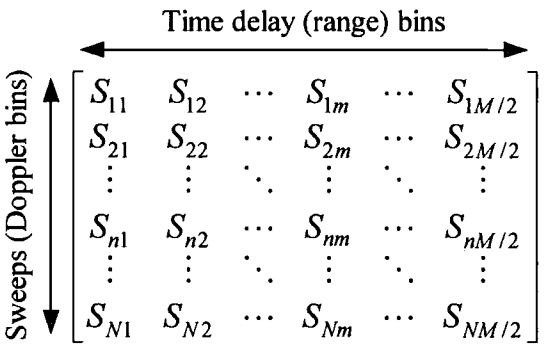


Figure 5.2 : Matrix containing range-Doppler numbers obtained with double-FFT process

The columns of the matrix represent the time delay bins (after the first FFT). The elements of each column represent the digital samples of a beat frequency component obtained every T seconds. As noted previously, for each successive sweep the elements of the matrix columns will undoubtedly change because the beat frequency, $f_{l,n}$, and the phase term, $2\pi f_c(v/c)nT$, are changing from sweep to sweep. The amplitude also changes from sweep to sweep due to the minor shift in the position of the beat note. However this change is small enough to be neglected [2]. Every sweep takes T seconds, therefore NT seconds are required to fill the matrix shown in figure 5.2.

It can be concluded from the above that the main variations within a column are due to the phase factor, thus

$$S_{nm} = K(f)e^{j2\pi f_0 \frac{v}{c} nT} = K(f)e^{j2\pi f_0 \frac{v}{c} t_n} \tag{5.9}$$

where $t_n = NT$ represents the discrete time flow from sweep to sweep and $K(f)$ is the amplitude variation.

The Fourier transform of equation (5.9) over t_n from 0 to NT is given by

$$S_m = K(f)NT \frac{\sin\left(2\pi\left(f - \frac{v}{c}f_0\right)NT/2\right)}{\left(2\pi\left(f - \frac{v}{c}f_0\right)NT/2\right)} \tag{5.10}$$

The above equation shows that the movement of the mobile results in a displacement of the $\sin(x)/x$ pulse. This is known as the Doppler shift $f_D = (v/c)f_0 = v/\lambda$. [1]

5.3 SAGE Algorithm

Spatial array signal processing has emerged as an active area of research concerned with estimating parameters from the data collected at array sensors. The estimation

problem depends on parameters such as array geometry, sensor characteristics and signal properties, etc. [3]. For the harmonic radar a high resolution estimation of the DOA of received waves would give a highly accurate estimation of the location of targets.

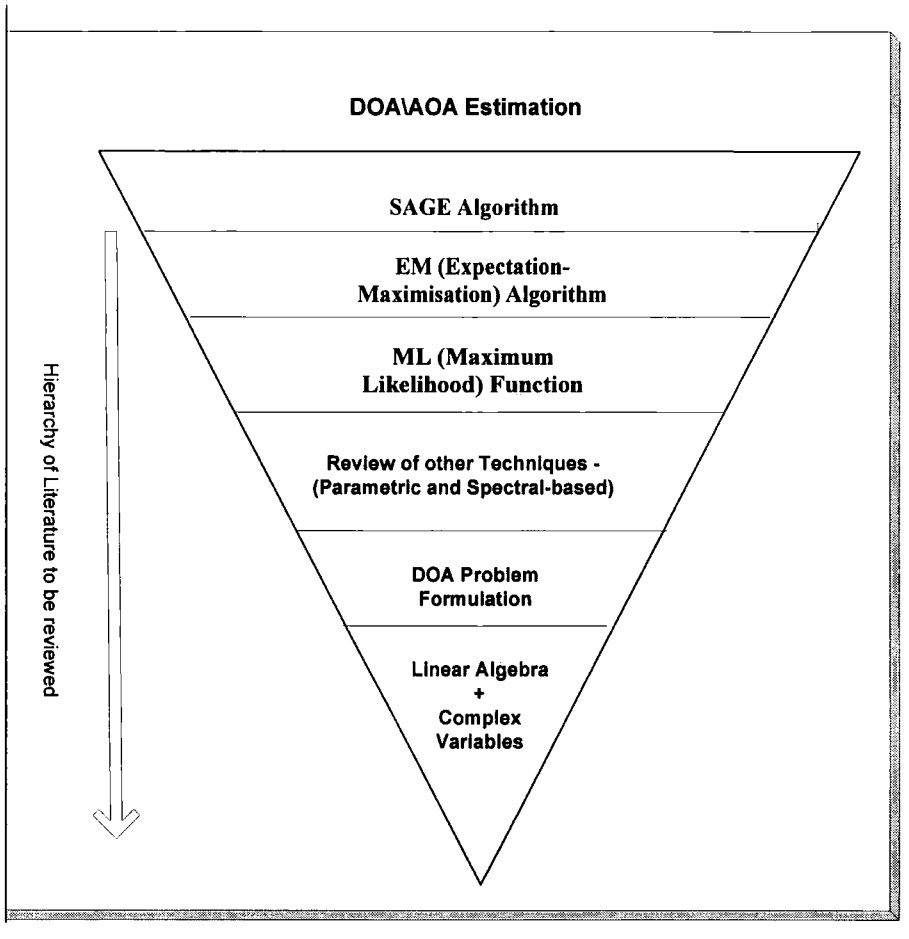


Figure 5.3: Hierarchy of Literature needed to be reviewed for SAGE Algorithm

One such high resolution and computationally effective algorithm is the SAGE (Space Alternating Generalized Expectation-Maximization) algorithm. This algorithm requires a deep and thorough understanding of a variety of statistical algorithms and a background in linear algebra. Figure (5.3) shows the kind of background required to gain an understanding of SAGE algorithm with the help of an inverted pyramid. A thorough understanding of the EM (Expectation-Maximisation) and ML (Maximum-

Likelihood) techniques is needed to build upon the implementation of SAGE algorithm. That should be the first step in SAGE Algorithm.

5.4 References

[1] D. E. Barrick, "FM/CW Radar Signals and Digital Processing," *U.S. Department of Commerce, National Oceanic and Atmospheric Administration (NOAA) Technical Report ERL 283-WPL 26*, 1973.

[2] Gokalp. H, "Characterisation of UMTS FDD Channels," Ph.D. Thesis, Dept. of Electrical Engineering and Electronics, UMIST, Manchester, UK, 2001

[3] Razavi-Ghods. N, "Characterisation of MIMO Propagation Channels", PhD. Thesis, Dept. of Engineering, University of Durham, 2007

Chapter 6

Conclusions & Further Work

It can be inferred from the contents of this report that developing an advanced radar system depends upon the knowledge of many disciplines. Several areas of research must be integrated together to have a harmonic radar system that can effectively and accurately locate rusty-bolt junctions. The project is multifaceted in nature and requires a thorough understanding of various dimensions ranging from having hands-on experience to design RF hardware to having a grasp of advanced linear algebra to implement processing algorithms. All in all, there is a real effort by the research community to investigate and develop more efficient and accurate harmonic radar systems as its applications find new frontiers. This is quite evident from the patents issued in recent years as referred in the report.

The chapters in the report covered many aspects of the project. Chapter 1 introduced the developments in radar systems and proposed a system utilizing all of these advancements. Chapter 2 gave an overview of FMCW Harmonic radars together with a research of NLJD (Non-Linear Junction Detectors) available in the commercial market. Comparison with commercial products can provide valuable insight to different aspects of this project. The radar range equation for this system needs to be developed further. This requires the knowledge of target RCS. For the project under discussion, some models of rusty-joints made of diodes would be used. More information on their dimensions is required. Chapter 3 described in detail the phenomenon of rusty-bolt effect focusing on the techniques used to identify them. Comparing the well-known techniques with the one proposed in this project highlights the novelty of this method. However, their performance should be compared in terms of some measurable quantities. Chapter 4 provided a system overview of the project and discussed its various parts focussing mainly on the digital programmer. A significant outcome of Chapter 4 was a critical study of the work done so far and some of the probable reasons the apparent slow progress. Chapter 5 described primarily the theoretical background of the signal processing techniques to

be used in the DSP block of the receiver. It highlighted the background one needs to fully comprehend the super-resolution SAGE processing algorithm.

It should be noted that the chapters of this report only provide an elementary description of the various systems and issues involved in developing a wideband radar system. More detailed understanding needs to be presented for forthcoming research. Following is a listing of possible areas of intuitive research during the course of the project. Some of them have been indicated in the Chapter review above.

- Development of a harmonic radar range equation similar to the one being used in entomological applications.

- RF Isolation between two receive channels which is a problem in many commercial NLJD units.

- Comparison with other localization techniques mentioned in literature in terms of performance and accuracy.

- Novel transmit waveform design apart from the linear chirp

- Design and development of the receiver DSP block comprising of SAGE Algorithm. The number of antenna array 'N' in relation to the angular resolution needs to be worked out.

- System calibration and evaluation after measurements in different scenarios.

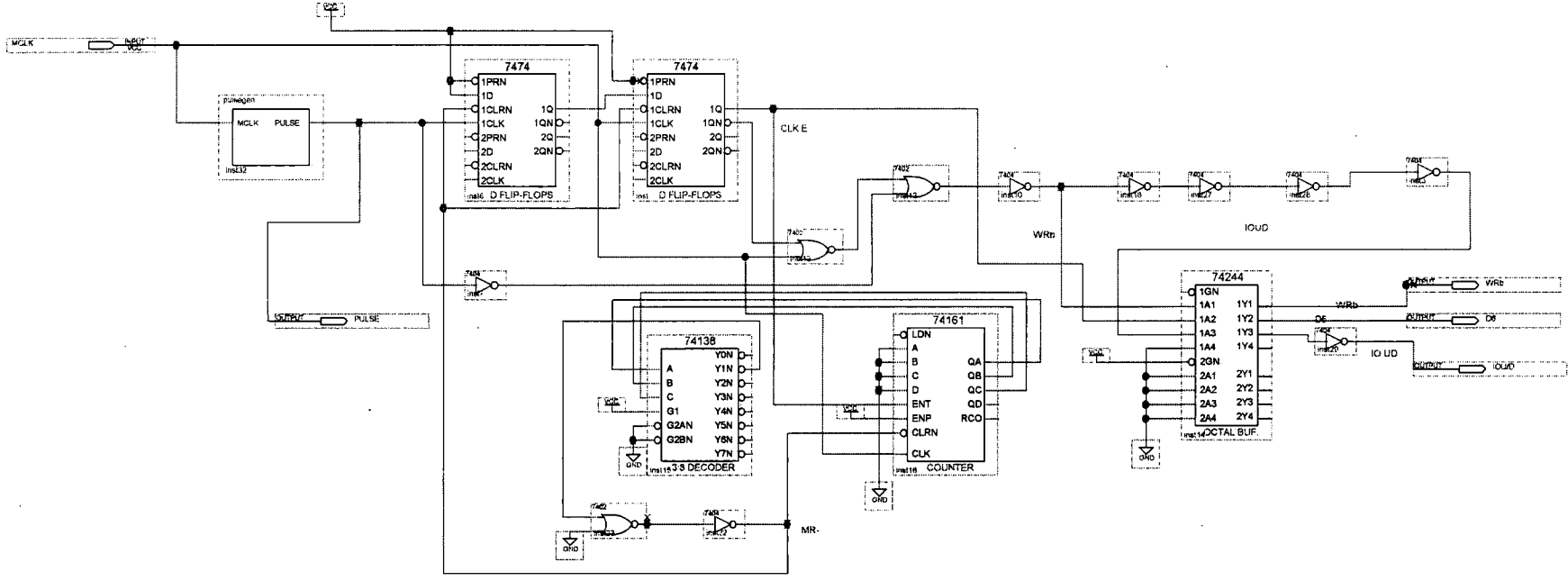
So far the approach had been to concentrate more on literature review, background study and to develop possible hypotheses that would provide novelty to the project. While this approach resulted in the development of some valuable premise for doing research, it greatly effected the overall practical hardware implementation of the project.

Appendix – 1

Gate Array Design – Schematics

A1.1 Phase Reset Circuit

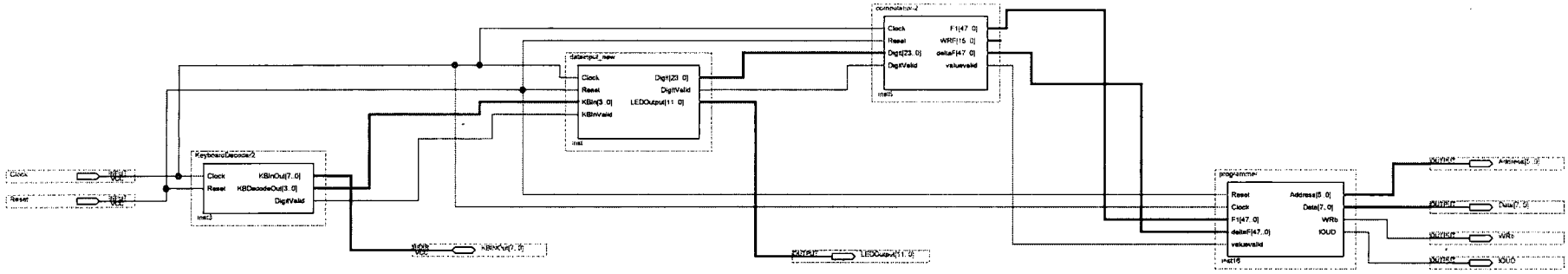
A digital circuit was designed that would reset the accumulated at the start of every chirp. The schematic is shown.



TITLE	Phase Reset Circuit
COMPANY	Durham University
DESIGNER	Farrukh Aslam
NUMBER	N/A
REV	A
DATE	Sun Apr 13 22:06:59 2008
SHEET	1 OF 1

A1.2 Digital Controller

The digital controller schematic consists of four blocks, each written in VHDL. The controller schematic is shown below:-


Clock
Reset
KeyboardDecoder2
Clock
Reset
KBInOut[7..0]
KBDecodeOut[3..0]
DigitValid
inst3
dataInput_new
Clock
Reset
KBin[3..0]
KBInValid
Digit[23..0]
DigitValid
LEDOutput[11..0]
inst4
Clock
Reset
Digit[23..0]
DigitValid
F1[47..0]
WRF[15..0]
deltaF[47..0]
valuesvalid
inst5
programmer
Reset
Clock
F1[47..0]
deltaF[47..0]
valuesvalid
Address[5..0]
Data[7..0]
WRb
IOUD
inst16

Appendix – 2

Gate Array Design – VHDL Code

```
-- Created by Farrukh Aslam
-- Last Update: 05/05/08

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

-- Keypad Debounce : Filters out mechanical switch bounces. Output is a clean signal
typically 40ms
-- Debounce clock should be approximately 10ms or 100Hz
entity debouncer is
  port(
              kpb                    : inout std_logic_vector (7 downto 0);
              clock_100Hz            : in std_logic;
     kpb_debounced    : inout std_logic_vector (7 downto 0)
              );
end debouncer;

architecture debouncer of debouncer is

signal shift_kpb                                          : std_logic_vector(31
downto 0);
signal flag0, flag1, flag2, flag3              : integer;
signal flag4, flag5, flag6, flag7              : integer;

begin

  process (clock_100Hz)
  begin

-- Use of a shift register to filter switch contact bounce

------------------------------ For kpb(0) -----------------------
        if (clock_100Hz'event and clock_100Hz='1') then
-- detect '1'
                        shift_kpb(3 downto 1) <= shift_kpb(2 downto 0);
                        shift_kpb(0) <= kpb (0);
                        if shift_kpb(3 downto 0)= "1111" then
                                kpb_debounced (0) <= '1';
                                flag0 <= 1;
                        else
                                kpb_debounced (0) <= '0';
                        end if;
```

```
-- detect '0'
                if (flag0 = 1) then
                        shift_kpb(3 downto 1) <= shift_kpb(2 downto 0);
                        shift_kpb(0) <= kpb (0);
                        if shift_kpb(3 downto 0)= "0000" then
                                kpb_debounced (0) <= '0';
                        else
                                kpb_debounced (0) <= '1';
                        end if;
                end if;

--      end if;

------------------------------ For kpb(1) -----------------------

-- detect '1'
                        shift_kpb(7 downto 5) <= shift_kpb(6 downto 4);
                        shift_kpb(4) <= kpb (1);
                        if shift_kpb(7 downto 4)= "1111" then
                                kpb_debounced (1) <= '1';
                                flag1 <= 1;
                        else
                                kpb_debounced (1) <= '0';
                        end if;

-- detect '0'
                if (flag1 = 1) then
                        shift_kpb(7 downto 5) <= shift_kpb(6 downto 4);
                        shift_kpb(4) <= kpb (1);
                        if shift_kpb(7 downto 4)= "0000" then
                                kpb_debounced (1) <= '0';
                        else
                                kpb_debounced (1) <= '1';
                        end if;
                end if;

------------------------------ For kpb(2) -----------------------
-- detect '1'
                        shift_kpb(11 downto 9) <= shift_kpb(10 downto 8);
                        shift_kpb(8) <= kpb (2);
                        if shift_kpb(11 downto 8)= "1111" then
                                kpb_debounced (2) <= '1';
                                flag2 <= 1;
                        else
                                kpb_debounced (2) <= '0';
                        end if;

-- detect '0'
                if (flag2 = 1) then
                        shift_kpb(11 downto 9) <= shift_kpb(10 downto 8);
```

```
                shift_kpb(8) <= kpb (2);
                if shift_kpb(11 downto 8)= "0000" then
                        kpb_debounced (2) <= '0';
                else
                        kpb_debounced (2) <= '1';
                end if;
        end if;

------------------------------ For kpb(3) ------------------------
-- detect '1'
                shift_kpb(15 downto 13) <= shift_kpb(14 downto 12);
                shift_kpb(12) <= kpb (3);
                if shift_kpb(15 downto 12)= "1111" then
                        kpb_debounced (3) <= '1';
                        flag3 <= 1;
                else
                        kpb_debounced (3) <= '0';
                end if;

-- detect '0'
        if (flag3 = 1) then
                shift_kpb(15 downto 13) <= shift_kpb(14 downto 12);
                shift_kpb(12) <= kpb (3);
                if shift_kpb(15 downto 12)= "0000" then
                        kpb_debounced (3) <= '0';
                else
                        kpb_debounced (3) <= '1';
                end if;
        end if;

------------------------------ For kpb(4) ------------------------
-- detect '1'
                shift_kpb(19 downto 17) <= shift_kpb(18 downto 16);
                shift_kpb(16) <= kpb (4);
                if shift_kpb(19 downto 16)= "1111" then
                        kpb_debounced (4) <= '1';
                        flag4 <= 1;
                else
                        kpb_debounced (4) <= '0';
                end if;

-- detect '0'
        if (flag4 = 1) then
                shift_kpb(19 downto 17) <= shift_kpb(18 downto 16);
                shift_kpb(16) <= kpb (4);
                if shift_kpb(19 downto 16)= "0000" then
                        kpb_debounced (4) <= '0';
                else
                        kpb_debounced (4) <= '1';
                end if;
```

```
                end if;
------------------------------ For kpb(5) -------------------------
-- detect '1'
                        shift_kpb(23 downto 21) <= shift_kpb(22 downto 20);
                        shift_kpb(20) <= kpb (5);
                        if shift_kpb(23 downto 20)= "1111" then
                                kpb_debounced (5) <= '1';
                                flag5 <= 1;
                        else
                                kpb_debounced (5) <= '0';
                        end if;

-- detect '0'
                if (flag5 = 1) then
                        shift_kpb(23 downto 21) <= shift_kpb(22 downto 20);
                        shift_kpb(20) <= kpb (5);
                        if shift_kpb(23 downto 20)= "0000" then
                                kpb_debounced (5) <= '0';
                        else
                                kpb_debounced (5) <= '1';
                        end if;
                end if;
------------------------------ For kpb(6) -------------------------
-- detect '1'
                        shift_kpb(27 downto 25) <= shift_kpb(26 downto 24);
                        shift_kpb(24) <= kpb (6);
                        if shift_kpb(27 downto 24)= "1111" then
                                kpb_debounced (6) <= '1';
                                flag6 <= 1;
                        else
                                kpb_debounced (6) <= '0';
                        end if;

-- detect '0'
                if (flag6 = 1) then
                        shift_kpb(27 downto 25) <= shift_kpb(26 downto 24);
                        shift_kpb(24) <= kpb (6);
                        if shift_kpb(27 downto 24)= "0000" then
                                kpb_debounced (6) <= '0';
                        else
                                kpb_debounced (6) <= '1';
                        end if;
                end if;
------------------------------ For kpb(7) -------------------------
-- detect '1'
                        shift_kpb(31 downto 29) <= shift_kpb(30 downto 28);
                        shift_kpb(28) <= kpb (7);
                        if shift_kpb(31 downto 28)= "1111" then
                                kpb_debounced (7) <= '1';
                                flag7 <= 1;
```

```
                    else
                        kpb_debounced (7) <= '0';
                    end if;

-- detect '0'
            if (flag7 = 1) then
                    shift_kpb(31 downto 29) <= shift_kpb(30 downto 28);
                    shift_kpb(28) <= kpb (7);
                    if shift_kpb(31 downto 28)= "0000" then
                        kpb_debounced (7) <= '0';
                    else
                        kpb_debounced (7) <= '1';
                    end if;
            end if;


        end if;


  end process;

end debouncer;

-- Created/Amended by Farrukh Aslam
-- Last Update : 29/04/08

-- The keyboard decoder works as follows:
-- A high signal is sent to all columns.
-- When a key is pressed, one of the row lines becomes high
-- This line is registered and is kept presistent so that is propagates back to the
columns
-- when the column line is identified, the combo of both lines is deciphered to give the
ouptut


library IEEE;
use IEEE.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity KeyboardDecoder2 is
        port (
          Clock: in STD_LOGIC;
                Reset: in STD_LOGIC;
                KBInOut: inout STD_LOGIC_VECTOR (7 downto 0);
                KBDecodeOut: out STD_LOGIC_VECTOR (3 downto 0);
                DigitValid: out STD_LOGIC

        );
end KeyboardDecoder2;
architecture KeyboardDecoder2 of KeyboardDecoder2 is
signal S_ScanRow: STD_LOGIC;
```

```
signal S_ScanCol: STD_LOGIC;
signal S_PulseCounter: INTEGER;
signal S_ResetFlag: INTEGER;
signal S_KBInOut: STD_LOGIC_VECTOR(7 downto 0);
signal S_KBInRow: STD_LOGIC_VECTOR(3 downto 0);
signal S_KBInCol: STD_LOGIC_VECTOR(3 downto 0);
signal S_KBOutRow: STD_LOGIC_VECTOR(3 downto 0);
signal S_KBOutCol: STD_LOGIC_VECTOR(3 downto 0);

begin

-- First four lines of KBInOut are connected as rows on the keyboard and the last four
lines
-- as columns. The condition is a default condition set by the reset line and when no
key is
-- pressed.

S_KBInRow<=KBInOut(3 downto 0)  when (S_ScanRow='1' and S_ScanCol='0')
else (others=>'Z');
KBInOut(7 downto 4)<=S_KBOutCol when (S_ScanRow='1' and S_ScanCol='0')
else (others=>'Z');
KBInOut(3 downto 0)<=S_KBOutRow when (S_ScanCol='1' and S_ScanRow='0')
else (others=>'Z');
S_KBInCol<=KBInOut(7 downto 4)  when (S_ScanCol='1' and S_ScanRow='0') else
(others=>'Z');


process(Reset,Clock)
begin
        if(Reset='1') then
                S_KBOutCol<=(others=>'1');
                S_ScanRow<='1';
                S_ScanCol<='0';
                S_ResetFlag<=1;
                S_PulseCounter<=0;
        elsif(Clock'Event and Clock='1') then
                if(S_ScanRow='1' and S_ScanCol='0') then
                        if(S_ResetFlag=0) then
                                S_PulseCounter<=S_PulseCounter+1;
                        end if;
                DigitValid<='0';
                --      if(S_PulseCounter>3 and S_PulseCounter<5) then
                --      if(S_PulseCounter=1) then
                --              DigitValid<='1';
                --      else
                --              DigitValid<='0';
                --      end if;
                        if(S_PulseCounter>5) then               S_PulseCounter<=6;  end
if;
                        if((S_KBInRow/="ZZZZ") and (S_KBInRow>"0000")) then
```

```
                    S_KBInOut(3 downto 0)<=S_KBInRow;
                    S_KBOutRow<=S_KBInRow;
                    S_ScanRow<='0';
                    S_ScanCol<='1';
                end if;
            elsif(S_ScanRow='0' and S_ScanCol='1') then
                S_PulseCounter<=0;
                if((S_KBInCol/="ZZZZ") and (S_KBInCol>"0000")) then --
remember to set KBIn pulled down
                    S_KBInOut(7 downto 4)<=S_KBInCol;
                    S_ScanRow<='0';
                    S_ScanCol<='0';
                    S_ResetFlag<=0;
                end if;
            elsif(S_ScanRow='0' and S_ScanCol='0') then
                S_PulseCounter<=0;

-- The 1's in the KBInOut are basically shorted out pins coming from the keyboard
decoder after
-- pressing of any key.
                if(S_KBInOut="00010001") then
        KBDecodeOut<="0000"; --0
                elsif(S_KBInOut="00100001") then
        KBDecodeOut<="0001"; --1
                elsif(S_KBInOut="01000001") then
        KBDecodeOut<="0010"; --2
                elsif(S_KBInOut="10000001") then
        KBDecodeOut<="0011"; --3
                elsif(S_KBInOut="00010010") then
        KBDecodeOut<="0100"; --4
                elsif(S_KBInOut="00100010") then
        KBDecodeOut<="0101"; --5
                elsif(S_KBInOut="01000010") then
        KBDecodeOut<="0110"; --6
                elsif(S_KBInOut="10000010") then
        KBDecodeOut<="0111"; --7
                elsif(S_KBInOut="00010100") then
        KBDecodeOut<="1000"; --8
                elsif(S_KBInOut="00100100") then
        KBDecodeOut<="1001"; --9
                elsif(S_KBInOut="01000100") then
        KBDecodeOut<="1010"; --a
                elsif(S_KBInOut="10000100") then
        KBDecodeOut<="1011"; --b
                elsif(S_KBInOut="00011000") then
        KBDecodeOut<="1100"; --c
                elsif(S_KBInOut="00101000") then
        KBDecodeOut<="1101"; --d
                elsif(S_KBInOut="01001000") then
        KBDecodeOut<="1110"; --e
```

```
                elsif(S_KBInOut="10001000") then
        KBDecodeOut<="1111"; --f
                end if;

                DigitValid<='1';

                S_KBOutCol<=(others=>'1');
                S_ScanRow<='1';
                S_ScanCol<='0';
            end if;
        end if;
end process;
end KeyboardDecoder2;

--      Created/Amended by Farrukh Aslam
--      Last Update: 06/05/08

-- This block formulates a number for the computation block. It detects if a number or
a letter
-- key has been pressed. So far actions are only taken on a number key and the 'E' and
'C' keys.
-- It also ouptuts to the LEDs displaying the keys that have been pressed.

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity datainput_new is
        port (
                Clock: in STD_LOGIC;
                Reset: in STD_LOGIC;
                KBIn: in STD_LOGIC_VECTOR (3 downto 0);
                KBInValid: in STD_LOGIC;
                Digit: out STD_LOGIC_VECTOR (23 downto 0);
                DigitValid: out STD_LOGIC ;
                LEDOutput: out STD_LOGIC_VECTOR (11 downto 0)
                );
end datainput_new;

architecture dataInput_new of dataInput_new is

signal S_Digit: STD_LOGIC_VECTOR (23 downto 0);
signal I_Chunk1: INTEGER;
signal I_Chunk2: INTEGER;
signal I_Chunk3: INTEGER;
signal I_Chunk4: INTEGER;
signal I_Chunk5: INTEGER;
signal I_Chunk6: INTEGER;
signal I_Total: INTEGER;
```

```
signal KBIn0, KBIn1, KBIn2 : std_logic_vector (3 downto 0) ;
signal flag0, flag1, flag2 : integer;

begin

process (Reset,Clock,KBInValid)
begin
        if(Reset='1') then
                DigitValid<='0';
                flag0 <= 1;     flag1 <= 0;     flag2 <= 0;

                Digit<=(Others=>'0');

                S_Digit<=(Others=>'1');
                I_Chunk1<=0;                    I_Chunk2<=0;
                I_Chunk3<=0;                    I_Chunk4<=0;
                I_Chunk5<=0;                    I_Chunk6<=0;
                I_Total<=0;

        elsif(Clock'Event and Clock='1') then
                DigitValid<='0';

                if(KBInValid='1') then
                        if(KBIn<"1010") then
                                I_Chunk6<=I_Chunk5;
        I_Chunk5<=I_Chunk4;
                                I_Chunk4<=I_Chunk3;
        I_Chunk3<=I_Chunk2;
                                I_Chunk2<=I_Chunk1;
        I_Chunk1<=CONV_INTEGER(KBIn);
                                S_Digit(23 downto 20)<=S_Digit(19 downto 16);

                                S_Digit(19 downto 16)<=S_Digit(15 downto 12);
                                S_Digit(15 downto 12)<=S_Digit(11 downto 8);

                                S_Digit(11 downto 8)<=S_Digit(7 downto 4);
                                S_Digit(7 downto 4)<=S_Digit(3 downto 0);
                                S_Digit(3 downto 0)<=KBIn;

-------------------------- LEDOutput ----------------------------
                                if (flag0 = 1) then
                                KBIn0 <= KBIn;
                                LEDOutput (11 downto 8) <= KBIn0 ;
                                LEDOutput (7 downto 4) <= x"0";
                                LEDOutput (3 downto 0) <= x"0";
                                flag0 <= 0;
                                end if;

                                if (flag0 = 0) then
```

```
                    KBIn1 <= KBIn;
                    LEDOutput (11 downto 8) <= KBIn0 ;
                    LEDOutput (7 downto 4) <= KBIn1 ;
                    LEDOutput (3 downto 0) <= x"0";
                    flag1 <= 1;
                    end if;

-----------------------------------------------------------------


                elsif(KBIn="1100") then --- clear the input digit
                    S_Digit(23 downto 0)<=(Others=>'1');
                    I_Chunk1<=0;                    I_Chunk2<=0;
                    I_Chunk3<=0;                    I_Chunk4<=0;
                    I_Chunk5<=0;                    I_Chunk6<=0;

                elsif(KBIn="1110") then --- enterkey
                    S_Digit(23 downto 0)<=(Others=>'1');

                    I_Total<=I_Chunk1+I_Chunk2*10+I_Chunk3*100+

        I_Chunk4*1000+I_Chunk5*10000+I_Chunk6*100000;
                    I_Chunk1<=0;                    I_Chunk2<=0;
                    I_Chunk3<=0;                    I_Chunk4<=0;
                    I_Chunk5<=0;                    I_Chunk6<=0;

-- LEDOutput after  'Enter' key
                    if (flag1 = 1) then
                    KBIn2 <= KBIn;
                    LEDOutput (11 downto 8) <= KBIn0 ;
                    LEDOutput (7 downto 4) <= KBIn1 ;
                    LEDOutput (3 downto 0) <= KBIn2 ;
                    flag0 <= 1;
                    end if;


                end if;
            elsif(I_Total>0) then
                Digit<=CONV_STD_LOGIC_VECTOR(I_Total,24);
                I_Total<=0;
                DigitValid<='1';

                end if;

        end if;
end process;
End datainput_new;
```

```
-- created by Farrukh Aslam
-- Last Update: 29th April 2008

library IEEE;
use IEEE.std_logic_1164.all;

entity LEDDisplay is
        port (
                Reset: in STD_LOGIC;
                LEDInput: in STD_LOGIC_VECTOR (23 downto 0);
                LEDOutput: out STD_LOGIC_VECTOR (11 downto 0)
                );
end LEDDisplay;
architecture LEDDisplay of LEDDisplay is
begin

process (LEDInput)
begin
                LEDOutput<=(Others=>'0');
                if(LEDInput(3 downto 0)="0000")   then   LEDOutput(3 downto
0)<="0111";
                elsif(LEDInput(3 downto 0)="0001") then   LEDOutput(3 downto
0)<="0100";
                elsif(LEDInput(3 downto 0)="0010") then   LEDOutput(3 downto
0)<="1101";
                elsif(LEDInput(3 downto 0)="0011") then   LEDOutput(3 downto
0)<="1101";
                elsif(LEDInput(3 downto 0)="0100") then   LEDOutput(3 downto
0)<="1110";
                elsif(LEDInput(3 downto 0)="0101") then   LEDOutput(3 downto
0)<="1011";
                elsif(LEDInput(3 downto 0)="0110") then   LEDOutput(3 downto
0)<="1011";
                elsif(LEDInput(3 downto 0)="0111") then   LEDOutput(3 downto
0)<="0101";
                elsif(LEDInput(3 downto 0)="1000") then   LEDOutput(3 downto
0)<="1111";
                elsif(LEDInput(3 downto 0)="1001") then   LEDOutput(3 downto
0)<="1111";
                elsif(LEDInput(3 downto 0)="1101") then   LEDOutput(3 downto
0)<="1100";
                elsif(LEDInput(3 downto 0)="1100") then   LEDOutput(3 downto
0)<="0011";
                elsif(LEDInput(3 downto 0)="1001") then   LEDOutput(3 downto
0)<="1111";
                else LEDOutput(11 downto 0)<="000000000000";
                end if;
                if(LEDInput(7 downto 4)="0000")   then   LEDOutput(7 downto
4)<="0111";
```

```
                elsif(LEDInput(7 downto 4)="0001") then   LEDOutput(7 downto
4)<="0100";
                elsif(LEDInput(7 downto 4)="0010") then   LEDOutput(7 downto
4)<="1101";
                elsif(LEDInput(7 downto 4)="0011") then   LEDOutput(7 downto
4)<="1101";
                elsif(LEDInput(7 downto 4)="0100") then   LEDOutput(7 downto
4)<="1110";
                elsif(LEDInput(7 downto 4)="0101") then   LEDOutput(7 downto
4)<="1011";
                elsif(LEDInput(7 downto 4)="0110") then   LEDOutput(7 downto
4)<="1011";
                elsif(LEDInput(7 downto 4)="0111") then   LEDOutput(7 downto
4)<="0101";
                elsif(LEDInput(7 downto 4)="1000") then   LEDOutput(7 downto
4)<="1111";
                elsif(LEDInput(7 downto 4)="1001") then   LEDOutput(7 downto
4)<="1111";
                else LEDOutput(11 downto 0)<="000000000000";
                end if;
                if(LEDInput(11 downto 8)="0000") then            LEDOutput(11
downto 8)<="0111";
                elsif(LEDInput(11 downto 8)="0001") then LEDOutput(11 downto
8)<="0100";
                elsif(LEDInput(11 downto 8)="0010") then LEDOutput(11 downto
8)<="1101";
                elsif(LEDInput(11 downto 8)="0011") then LEDOutput(11 downto
8)<="1101";
                elsif(LEDInput(11 downto 8)="0100") then LEDOutput(11 downto
8)<="1110";
                elsif(LEDInput(11 downto 8)="0101") then LEDOutput(11 downto
8)<="1011";
                elsif(LEDInput(11 downto 8)="0110") then LEDOutput(11 downto
8)<="1011";
                elsif(LEDInput(11 downto 8)="0111") then LEDOutput(11 downto
8)<="0101";
                elsif(LEDInput(11 downto 8)="1000") then LEDOutput(11 downto
8)<="1111";
                elsif(LEDInput(11 downto 8)="1001") then LEDOutput(11 downto
8)<="1111";
                else LEDOutput(11 downto 0)<="000000000000";
                end if;
end process;
end LEDDisplay;
```

```
-- created by Farrukh Aslam
-- Last Update: 14/04/08

-- This block computes the values needed to be programmed in the DDFS synthesizer.
-- The order of values being entered is fc, BW, WRF. The factor (2^48/SysClk)
remains
-- to be added.

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
--use IEEE.math_real.all;
--use IEEE.numeric_std.all;
entity computation2 is
        port (
                Clock: in STD_LOGIC;

                Reset: in STD_LOGIC;
                Digit: in STD_LOGIC_VECTOR (23 downto 0);
                DigitValid: in STD_LOGIC;

                F1: out STD_LOGIC_VECTOR (47 downto 0);
        --      FC: out STD_LOGIC_VECTOR (47 downto 0);
        --      WRF: out STD_LOGIC_VECTOR (15 downto 0);
        --      BW: out STD_LOGIC_VECTOR (47 downto 0);
                deltaF: out STD_LOGIC_VECTOR(47 downto 0) ;

                valuevalid: out std_logic ;
        --      F1_factor: out STD_LOGIC_VECTOR (47 downto 0)

                factor_bin : out std_logic_vector (19 downto 0)

                );
end computation2;

architecture computation2 of computation2 is
signal I_WordNumber:INTEGER;

signal I_FC: INTEGER ;
signal I_BW:INTEGER;
signal I_WRF:INTEGER;
signal I_F1:INTEGER;
signal I_DF:INTEGER;

--signal counter:INTEGER;


--signal I_FC: unsigned (1 downto 0) ;
--signal I_BW: unsigned (1 downto 0);
```

```
--signal I_WRF:unsigned (1 downto 0);
--signal I_F1:unsigned (1 downto 0);
--signal I_DF:unsigned (1 downto 0);



--signal I_F11: unsigned;
--signal F1_factor: real;

--constant x: unsigned (1 downto 0) := b"01" ;
--constant y: unsigned (1 downto 0) := b"00";
--constant z: unsigned (11 downto 0) := X"3E8";

--constant a: unsigned (1 downto 0) := b"10";

signal factor : real ;


begin

factor <= (2.0E10) / 3.0E3 ;
factor_bin <= conv_std_logic_vector (factor,20);

process (DigitValid,Reset)
begin

        if (reset='1') then
                I_WordNumber <=1;
                I_BW  <= 0;
                I_WRF <= 0;
                I_FC <= 0;
                I_DF<= 0;
                valuevalid<= '0';

                --I_F11 <= to_unsigned (I_F1,48 );

                elsif (DigitValid'Event and DigitValid='0') then
                                if(I_WordNumber=1) then
                        I_FC <= CONV_INTEGER (Digit);
                                --I_FC <=  unsigned (Digit) ;
                                I_WordNumber<=2;

                                elsif(I_WordNumber=2) then
                                I_FC <= I_FC * 1000;
                                --I_FC <= I_FC * z ;
                                I_BW <= CONV_INTEGER (Digit);
                                I_WordNumber<=3;

                                elsif (I_WordNumber=3) then
                                --I_BW <= I_BW * z;
```

```
                    I_BW <= I_BW * 1000;
                    I_WRF<= CONV_INTEGER (Digit);
                I_WordNumber<=4;

                  elsif (I_WordNumber<=4) then
                    I_F1 <= I_FC - (I_BW / 2);
                I_DF <= ((I_BW * I_WRF) / 300000) * 114 ;


                    valuevalid <= '1';
                    --x <=  round (2.0E1) ;
                end if;


        end if;
        F1 <= CONV_STD_LOGIC_VECTOR (I_F1, 48);
        --FC <= CONV_STD_LOGIC_VECTOR (I_FC, 48);
        --BW <= CONV_STD_LOGIC_VECTOR (I_BW, 48);
--      WRF<= CONV_STD_LOGIC_VECTOR (I_WRF, 16);
        deltaF <= CONV_STD_LOGIC_VECTOR (I_DF, 48);

        --F1_factor <= CONV_STD_LOGIC_VECTOR (x, 48);

end process;

end computation2;

-- DDFS AD9854 Programmer
-- created by Farrukh Aslam
-- Last Update : 14/04/08

-- This block transmits the values of F1, deltaF, ramp rate, RefMult and the control
word
-- to AD9854. It also generates signals for WRb and the IOUD Clk.

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity programmer is
        port (

                Reset: in STD_LOGIC;
                Clock: in STD_LOGIC ;
                F1 : in STD_LOGIC_VECTOR (47 downto 0);
                deltaF: in STD_LOGIC_VECTOR (47 downto 0);
                valuevalid : in std_logic;

                Address : out STD_LOGIC_VECTOR (5 downto 0);
```

```
            Data : out STD_LOGIC_VECTOR (7 downto 0);
            WRb : out std_logic ;
            IOUD : out std_logic

            --dfcount : out std_logic
            );

end programmer;

architecture programmer of programmer is
signal clk_s : std_logic;
signal syncon : std_logic;
signal counter: integer;
signal count : integer;
signal dfcounter: integer;
signal start1: integer;
signal rrcounter: integer;
signal rcount: integer;
signal rmcounter: integer;
signal cwcounter: integer;

signal buff0: std_logic;
signal buff1: std_logic;
signal WRs: std_logic;

signal freg0 : STD_LOGIC_VECTOR (7 downto 0);
signal freg1 : STD_LOGIC_VECTOR (7 downto 0);
signal freg2 : STD_LOGIC_VECTOR (7 downto 0);
signal freg3 : STD_LOGIC_VECTOR (7 downto 0);
signal freg4 : STD_LOGIC_VECTOR (7 downto 0);
signal freg5 : STD_LOGIC_VECTOR (7 downto 0);

signal dfreg0 : STD_LOGIC_VECTOR (7 downto 0);
signal dfreg1 : STD_LOGIC_VECTOR (7 downto 0);
signal dfreg2 : STD_LOGIC_VECTOR (7 downto 0);
signal dfreg3 : STD_LOGIC_VECTOR (7 downto 0);
signal dfreg4 : STD_LOGIC_VECTOR (7 downto 0);
signal dfreg5 : STD_LOGIC_VECTOR (7 downto 0);



begin

freg0 <= F1 (47 downto 40);
freg1 <= F1 (39 downto 32);
freg2 <= F1 (31 downto 24);
freg3 <= F1 (23 downto 16);
freg4 <= F1 (15 downto 8);
freg5 <= F1 (7 downto 0);
```

```
dfreg0 <= deltaF (47 downto 40);
dfreg1 <= deltaF (39 downto 32);
dfreg2 <= deltaF (31 downto 24);
dfreg3 <= deltaF (23 downto 16);
dfreg4 <= deltaF (15 downto 8);
dfreg5 <= deltaF (7 downto 0);


syncon_gen: process(Clock)
begin
        if (Clock'event and Clock = '1') then
                clk_s <= not (clk_s);
                syncon<=clk_s;
        end if;
end process syncon_gen ;

-- Reset Initialization
init: process (Clock, Reset )
begin
        if (Reset ='1') then
                start1 <=0;

        elsif (valuevalid = '1') then

                start1<=1;

        end if;

end process init;

--Tranmission of WRb_gen
wrbgen: process (Clock)
begin
        if (Clock'Event and Clock='1') then
                --buff0 <=not (Clock);
                --buff1 <=buff0;
                WRs <= not (WRs);
                WRb  <= WRs;
        end if;
end process wrbgen;

-- Transmission of IOUD Clock
IOUD<=WRs ;

-- Process to send data to AD9854
prog: process (syncon, Reset)
begin
        if (Reset='1') then
                counter<=0;
                count<=0;
```

```
            Address<=b"000000";
            Data <=b"00000000";
        --  WRb<='1';
        --  IOUD<='0';
            dfcounter<=0;
            rrcounter<=0;
            rcount<=0;
            rmcounter<=0;
            cwcounter<=0;

---- Transmission of F1 words
        elsif (syncon'Event and syncon='1') then

                    if(counter=0 and start1=1) then

                        Address  <= b"000100";
                        Data <=  freg0;
                        counter<=1;

                    elsif(counter=1 and start1=1) then

                        Address  <= b"000101";
                        Data <= freg1;
                        counter <=2;

                    elsif(counter=2 and start1=1) then

                        Address  <= b"000110";
                        Data <= freg2;
                        counter <=3;

                    elsif(counter=3 and start1=1) then

                        Address  <= b"000111";
                        Data <= freg3;
                        counter<=4;

                    elsif(counter=4 and start1=1) then

                        Address  <= b"001000";
                        Data <= freg4;
                        counter<=5;

                    elsif(counter=5 and start1=1) then

                        Address  <= b"001001";
                        Data <= freg5;
                        dfcounter<=1;

                    end if;
```

```
---- Transmission of detlaF words
                    if(count=0 and dfcounter=1) then
                        Address <= b"001010";
                        Data <= dfreg0;
                        count<=1;

                    elsif(count=1 and dfcounter=1) then
                        Address <= b"001011";
                        Data <= dfreg1;
                        count <=2;

                    elsif(count=2 and dfcounter=1) then

                        Address <= b"001100";
                        Data <= dfreg2;
                        count <=3;

                    elsif(count=3 and dfcounter=1) then

                        Address <= b"001101";
                        Data <= dfreg3;
                        count<=4;

                    elsif(count=4 and dfcounter=1) then

                        Address <= b"001110";
                        Data <= dfreg4;
                        count<=5;

                    elsif(count=5 and dfcounter=1) then

                        Address <= b"001111";
                        Data <= dfreg5;
                        rrcounter<=1;

                    end if;
-- Tranmission of ramp rate clock

                    if(rcount=0 and rrcounter=1) then
                        Address <= b"011010";
                        Data <= b"00000000";
                        rcount<=1;

                    elsif(rcount=1 and rrcounter=1) then

                        Address <= b"011011";
                        Data <= b"00000000";
                        rcount <=2;
```

```
                elsif(rcount=2 and rrcounter=1) then

                        Address  <= b"011100";
                        Data <= b"01110001";
                        rmcounter<=1;

                end if;
--Tranmission of RefMult values
                if(rmcounter=1) then
                        Address  <= b"011110";
                        Data <=  b"01001111";
                        cwcounter<=1;
                end if;

--Tranmission of Control Word
                if(cwcounter=1) then
                        Address  <= b"011111";
                        Data <=  b"10000110";
                end if;

        end if;

end process prog;


end programmer;
```

Appendix – 3

DDS Programmer – C Code

A3.1 Decimal to binary Converter

A decimal to binary converter is needed when calculating the chirp parameters. It is given below.

```
#include <iostream.h>

void binary(int);

void main(void) {
        int number;

        cout << "Please enter a positive integer: ";
        cin >> number;
        if (number < 0)
                cout << "That is not a positive integer.\n";
        else {
                cout << number << " converted to binary is: ";
                binary(number);
                cout << endl;
        }
}

void binary(int number) {
        int remainder;

        if(number <= 1) {
                cout << number;
                return;
        }

        remainder = number%2;
        binary(number >> 1);
        cout << remainder;
}
```

A3.2 Programming the DDFS

A C code was written that would take values from user, calculate the required parameters and then send data to the DDFS board.

```
// chirp.cpp : Defines the entry point for the console application.

#include <stdio.h>
#include <dos.h>
#include <iostream.h>
#include <conio.h>

#define DATAPORT 0X3BC
#define STATUSPORT DATAPORT + 1
#define CTRLPORT DATAPORT + 2

void cmdloadPLL_click ( unsigned long refmul);
void cmdloadUDCLK_click (unsigned long UDCLK);
void cmdAmpLoad_click();
void cmdLoad_click (unsigned long FTW, unsigned long FSW, unsigned long
Ramprate);
void cmdLoad_click_test();

void write (int address , int data);
void writedata (int data);
void writeaddr (int addr);
void WRBAR_lo ();
void WRBAR_hi ();
void latchdata ();
void latchaddr ();
void latchctrl ();


unsigned long extclk, refmul, FTW, FSW, Ramprate;
unsigned long F_sys, UDCLK;

void main ()
{
        clrscr();
/*
        printf ("Enter the external clock frequency: ");
        scanf ("%d", &extclk);

        printf ("\nEnter the system reference multiplier value:");
        scanf ("%d", &refmul);

        printf ("\nEnter the system I/O UD CLK frequency:");
        scanf ("%d", &UDCLK);

        F_sys = refmul * extclk;
```

```
        cout << "\n Value of F_sys:" << F_sys;
*/
        cout << "Enter to start:";
        cmdloadPLL_click(refmul);          // Load the Reference multiplier value in
the register value 1E

        cmdloadUDCLK_click (UDCLK);        // Load the UDCLK value in the
registers 16,17,18,19

//      cmdAmpLoad_click();                // Load 0 value at 1F

        /*
        printf ("Enter the Start frequency :");
        scanf ("%d", &FTW);

        printf ("\nEnter Frequency Step Word:");
        scanf ("%d", &FSW);

        printf ("\nEnter the Ramp rate:" , "\n");
        scanf ("%d", &Ramprate);
        */
        //cmdLoad_click (FTW, FSW, Ramprate);  // Load FTW, FSW , Ramprate
and Control word register

        cmdLoad_click_test ();
        cout << "Finished!";
        getch();
}

void cmdloadPLL_click ( unsigned long refmul)
{

/*      int Intval = 64 + refmul;
        write (30, Intval);
*/
        write (30,0x46);
}

void cmdloadUDCLK_click (unsigned long UDCLK)
{
        unsigned long N;
        int UD1, UD2, UD3, UD4;
        /*N = (F_sys / (UDCLK*2)) - 1 ;
        //printf ("\n Value of N is: ", '%d', 'N');
        //cout << "Value of N is: " << N;

        UD1 = N & 0xFF000000;
        UD2 = N & 0xFF0000;
        UD3 = N & 0xFF00;
        UD4 = N & 0xFF;
```

```
        write (16, UD1);
        write (17, UD2);
        write (18, UD3);
        write (19, UD4);
        */
        //int Intval = 3; // For Chirp mode
        //Intval = Intval + 1; // For Internal Clock

        //write (0x1F,Intval);
        write (0x16,0x00);
        write (0x17,0x0B);
        write (0x18,0x71);
        write (0x19,0xAF);

}


void cmdAmpLoad_click()
{

        int Intval = 0;
        write (Intval, 0x1F);


}
/*
void cmdLoad_click (unsigned long FTW, unsigned long FSW, unsigned long
Ramprate)
{
        int FTW1, FTW2, FTW3, FTW4, FTW5, FTW6;
        int FSW1, FSW2, FSW3, FSW4, FSW5, FSW6;
        int Ramprate1, Ramprate2, Ramprate3;

        //Use the default values of phase registers is 00.
        // Separate FTW bytes and write in corresponding registers
        FTW1 = FTW & 0xFF0000000000;
        FTW2 = FTW & 0xFF00000000;
        FTW3 = FTW & 0xFF000000;
        FTW4 = FTW & 0xFF0000;
        FTW5 = FTW & 0xFF00;
        FTW6 = FTW & 0xFF;

        write (0x04, FTW1);
        write (0x05, FTW2);
        write (0x06, FTW3);
        write (0x07, FTW4);
        write (0x08, FTW5);
        write (0x09, FTW6);
```

```
        // Separate FSW bytes and write in corresponding registers
        FSW1 = FSW & 0xFF0000000000;
        FSW2 = FSW & 0xFF00000000;
        FSW3 = FSW & 0xFF000000;
        FSW4 = FSW & 0xFF0000;
        FSW5 = FSW & 0xFF00;
        FSW6 = FSW & 0xFF;

        write (0x10, FSW1);
        write (0x11, FSW2);
        write (0x12, FSW3);
        write (0x13, FSW4);
        write (0x14, FSW5);
        write (0x15, FSW6);

        // Separate Ramprate registers and write in corresponding registers

        Ramprate1 = Ramprate & 0xFF0000;
        Ramprate2 = Ramprate & 0xFF00;
        Ramprate3 = Ramprate & 0xFF;

        write (0x1A, Ramprate1);
        write (0x1B, Ramprate2);
        write (0x1C, Ramprate3);

        write (0x1F , 0x87);    // Write the control word for chip generation at 1F


} */

void cmdLoad_click_test ()
{
        //write FTW1
        write (0x04, 0x08);
        write (0x05, 0x88);
        write (0x06, 0x88);
        write (0x07, 0x88);
        write (0x08, 0x88);
        write (0x09, 0x88);

        //write FSW
        write (0x10, 0x00);
        write (0x11, 0x02);
        write (0x12, 0x2F);
        write (0x13, 0x3E);
        write (0x14, 0x93);
        write (0x15, 0x97);

        //write Ramprate
        write (0x1A, 0x00);
```

```
        write (0x1B, 0x0B);
        write (0x1C, 0x3F);

        write (0x1F , 0x87);   // Write the control word for chip generation at 1F

}
void write (int addr , int data)
{

        writedata (data);
        writeaddr (addr);
        WRBAR_lo ();
        WRBAR_hi ();
}

void writedata (int data)
{

        outp (DATAPORT, 255 - data);
        delay (100);
        latchdata ();

}

void writeaddr (int addr)
{

        outp (DATAPORT, 255 - addr);
        delay (100);
        latchaddr();

}

void WRBAR_lo ()
{
        outp (DATAPORT, 255 - 18);
        delay (100);
        latchctrl ();

}

void WRBAR_hi ()
{

        outp (DATAPORT, 255 - 19);
        delay (100);
        latchctrl ();

}
```

```
        write (0x1B, 0x0B);
        write (0x1C, 0x3F);

        write (0x1F , 0x87);   // Write the control word for chip generation at 1F

}
void write (int addr , int data)
{

        writedata (data);
        writeaddr (addr);
        WRBAR_lo ();
        WRBAR_hi ();
}

void writedata (int data)
{

        outp (DATAPORT, 255 - data);
        delay (100);
        latchdata ();

}

void writeaddr (int addr)
{

        outp (DATAPORT, 255 - addr);
        delay (100);
        latchaddr();

}

void WRBAR_lo ()
{
        outp (DATAPORT, 255 - 18);
        delay (100);
        latchctrl ();

}

void WRBAR_hi ()
{

        outp (DATAPORT, 255 - 19);
        delay (100);
        latchctrl ();

}
```

```
void latchdata ()
{
        outp (CTRLPORT, 10);
        delay (100);
        outp (CTRLPORT, 11);
        delay (100);
}

void latchaddr ()
{
        outp (CTRLPORT, 9);
        delay (100);
        outp (CTRLPORT, 11);
        delay (100);
}

void latchctrl ()
{
        outp (CTRLPORT, 3);
        delay (100);
        outp (CTRLPORT, 11);
        delay (100);
}
```

Appendix 4

Chirp Parameter Calculator

A4.1 Visual Basic Code

The following code was written to calculate chirp parameters used to program the DDFS. It was developed in Visual Basic.

```
Private Sub Command1_Click()

' Dim BW As Single

' Calculate BW
Text1.Text = Val(Val(Text4.Text) * (Val(Text10.Text) / Val(Text7.Text)))

End Sub

Private Sub Command2_Click()
' WRF is the inverse of Tau and vice versa

If (Val(Text4.Text) > 0) Then
' Calculate WRF
Text3.Text = Val(1 / Val(Text4.Text))
Else
' Calculate Tau
Text4.Text = Val(1 / Val(Text3.Text))
End If

End Sub

Private Sub Command3_Click()

If Option1.Value = True Then
' Calculating T from N
N_Hex = CLng("&h" & Text6.Text)
Text5.Text = Val((N_Hex + 1) * (Val(Text2.Text) * 2))

ElseIf Option2.Value = True Then
' Calculating N from T
N_Dec = (Text5.Text / (Val(Text2.Text) * 2)) - 1
Text6.Text = Hex(N_Dec)

End If
```

```
End Sub

Private Sub Command4_Click()

If Option3.Value = True Then
' Calculating deltaT from N'
Nbar_Hex = CLng("&h" & Text8.Text)
Text7.Text = Val((Nbar_Hex + 1) * (Val(Text2.Text)))

ElseIf Option4.Value = True Then
' Calculating N' from deltaT
Nbar_Dec = (Val(Text7.Text) / Val(Text2.Text)) - 1
Text8.Text = Hex(Nbar_Dec)

End If

End Sub

Private Sub Command5_Click()
' Calculating t
Text9.Text = 8 * (Val(Text2.Text))

End Sub

Private Sub Command6_Click()
'Calculate Tau from BW

 Text4.Text = Val(Text1.Text) * (Val(Text7.Text) / Val(Text10.Text))

End Sub
```

Appendix – 5

Publications and Outputs

A5.1 Conference Publication

The following is an abstract submitted to commission C of the National USRI Symposium in Portsmouth, 2nd-3rd July 2007.

Imaging with 1 GHz Harmonic Radar

Prof. Sana Salous sana.salous@dur.ac.uk
S. M. Farrukh Aslam s.m.farrukh-aslam@dur.ac.uk

A 'Harmonic Radar' is a device that illuminates a region of space with RF waves and receives the harmonics of the transmitted frequencies. The received data can then be processed to find the exact location and mobility of the points causing the generation of these harmonics. It works on the principle of radar transmitting a chirp signal and receiving harmonics of the transmitting frequency. Work is currently being carried out at the 'Centre for Communication Systems' in Durham University funded by HMGCC on the design and implementation of a novel Wideband Harmonic Radar system with a suitable waveform and multiple antenna arrays with algorithms for angular information.

Radio sites consist naturally of metallic structures. Metals are always covered by an oxide film due to the metal reacting chemically with the oxygen in air. The rate of this oxide formation depends largely on the environment. Any oxide film between metallic contacts will cause non-linearity. RF currents passing through these junctions would generate harmonics. When RF signals at two frequencies f1 and f2 pass through a non-linearity they create signals at their sum and difference frequencies. These are known as 'inter-modulation products'. This generation of inter-modulation products when radio waves interact with rusty parts is called as the 'Rusty Bolt Effect'. Radio spectrum is carefully controlled for optimal usage of the available frequencies so that different services operate in well-defined frequency channels. Ofcom has set some standards for radio site engineering. This set of standards is given in the document 'MPT 1331: Code of Practice for Radio Site Engineering'[1]. Any transmission site which is not following these codes would likely cause interference to other users. It is important that radio engineers should check the sites for their compliance with these codes. If a particular radio site is causing interference due to the rusty-bolt effect, the corroded points must be located to minimize their effect using the Harmonic Radar.

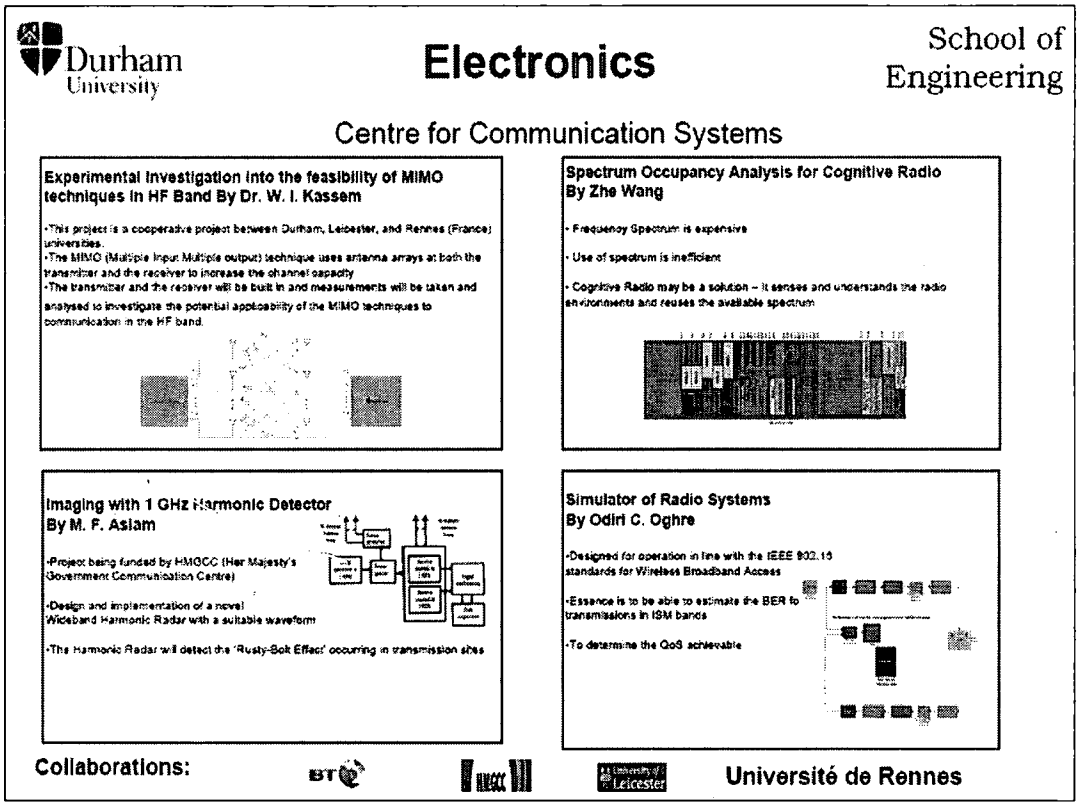
This Harmonic Radar will transmit at 1GHz and receive at 2/3 GHz channels. The first part of the ... being carried out is on the chirp generator which is the heart of the transmitter. It consists of a Direct Digital Frequency Synthesizer (DDFS), mixers, Programmable Phase Locked Loops (PLLs), Band Pass Filters, power amplifier and a digital controller. DDFS AD9854 has been used in the chirp generator. Future work will focus on the remaining part of the transmitter and the receiver, primarily on the antenna arrays and the accompanied signal processing.

Reference:

1. MPT 1331: Code of Practice for Radio Site Engineering

A5.2 Other Outputs

Following is a poster presented at the Annual School Research Day, University of Durham, June 2007



Following is a poster accepted to be presented at the GRADUK Northeast 2008 Poster Competition being held at University of York, 7th March 2008.

Wideband Harmonic Radar Detection

Can we detect very small bugs ?

Prof. Sana Salous, Dr. S.M. Feeney, Farrukh Aslam
Department of Engineering, Durham University
s.m.farrukh-aslam@dur.ac.uk

Durham University

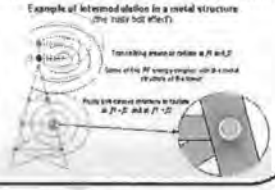
HMGCC

1. Introduction

Radio sites consist naturally of metallic structures. Any oxide film between metallic contacts will cause non-linearity. Currents passing through these junctions would generate harmonics. This generation of inter-modulation products when radio waves interact with rusty parts is called as the 'Rusty Bolt Effect'. These rusty bolts cause the transmission sites to operate beyond well-defined frequency bands. Ofcom has set some standards for radio site engineering which should be abided by all transmission sites.'

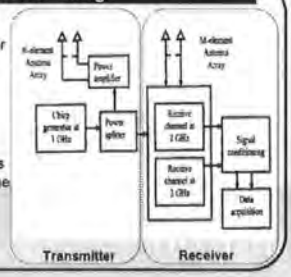
2. Objectives

To develop a Radar system that would accurately detect the 'rusty bolt' joints (bugs) causing the harmonic emissions.

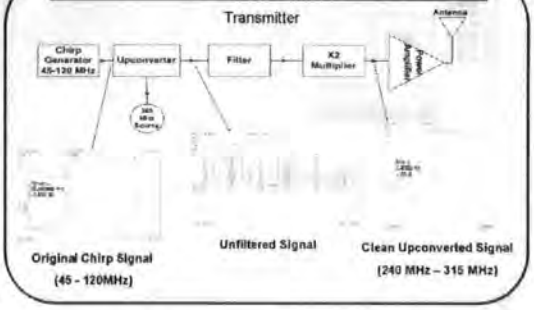


3. Transceiver Design

- **Bi static System** – Separate blocks for transmitter and receiver
- **Chirp** type Transmit Signal at 960 MHz - 1920 MHz
- **Antenna Arrays:** 'N' Transmitter and 'M' Receiver
- **Signal Processing Algorithms** to find the direction of arrival of the receiving waves (e.g. SAGE Algorithm)



4. Results



5. Conclusions & Further Work

Accuracy and resolution of this harmonic radar depends upon the bandwidth and type of the transmit signal. The frequency synthesizer board produces a clean chirp signal at 45 – 120 MHz. It is then successfully up converted and then filtered to produce a signal at 240 – 315 MHZ. The resulting frequency response graphs on a spectrum analyser are shown. It would then be up converted to a 1GHz transmit signal.

Further work needs to be done to complete the receiver part of the project. Signal Processing Algorithms would then be developed that would use an appropriate number of receiving antennas to collect data and then process them to retrieve useful information.

6. Reference

1. MPT 1331: Code of Practice for Radio Site Engineering