DEVELOPMENT OF AN FPGA SYSTEM FOR PARALLEL PROCESSING OF RAILWAY NON-DESTRUCTIVE TESTING DATA

by

Zhenhe Zhang

A thesis submitted to

The University of Birmingham

for the degree of

MASTER OF RESEARCH

School of Electronic, Electrical and Computer Engineering
College of Engineering and Physical Sciences
University of Birmingham
September 2014

Abstract

Cracks in rails are bad news; they cause accidents and cost money due to delays, as well as incurring repair costs. Inspection of tracks is required in order to find small cracks before they become dangerous. Early detection could also allow repair work which needs maintenance possession on railways to be planned. Non-destructive testing (NDT) is commonly used in rail crack inspection. Alternating Current Field Measurement (ACFM) is one of the latest NDT techniques to be used in crack measurement. This technique is able to detect surface breaking cracks in metals and measure them with proper processing of the non-destructive testing data. In the first part of this dissertation, the current limitations of inspection using ACFM techniques will be laid out. The content that follows describes a high-speed data processing chain for non-destructive testing data, as implemented using an FPGA development board. Multiple ACFM probes are used in practice to cover the surface of the track. Meanwhile, the data collected are parallel processed within the FPGA device. Here, the latest progress and the achievements of this project will be shown using proposed structure diagrams and initial results.

Keywords: non-destructive inspection, NDT, ACFM, high-speed inspection, maintenance, FPGA, railway, crack detection

Acknowledgement

The provision of facilities and funding by the Centre for Rail Research and Education and the School of Engineering at the University of Birmingham is gratefully acknowledged. The author also wishes to express his gratitude to TSC Inspections Systems Ltd for the provision of the ACFM example data and equipment. Finally, the author also wishes to thank his supervisors, Dr. Edward Stewart and Prof. Clive Roberts, for their patient supervision.

Table of Contents

List of Figures

List of Tables

List of Equations

1 Introduction

Rail tracks are subjected to intense bending and shear stresses, plastic deformation and wear, leading to degradation of their structural integrity with time (Ferreira and Murray, 1997). At the same time, tracks suffer rail failures, which have been a significant problem for more than 150 years. Maintenance procedures have been developed over the years, but broken rails are still found in the rail network from time to time (Papaelias, Robert and Davis, 2008). In October 2000, the derailment at Hatfield highlighted the importance of the early detection of cracks in rails, which could help find potential safety hazards in order to prevent the accidents happening. In the final report relating to this accident, the Investigation Board claimed that the derailment began with a transverse fatigue crack and was followed with more failures which occurred as a reaction to the stresses induced in the unsupported rail(Office of rail regulation;, 2006). Early detection of cracks in rails is of significant importance, since a single crack can potentially lead to fatigue failure of a rail and, as a result, the derailment of a train. The stresses sustained by the rails have increased in the last few years due to the use of higher train speeds and heavier axle loads (Papaelias et al., 2009). To combine with the substantial increase in axle loads and travel speeds, new harder steel grades, which show significantly higher wear resistance than steel grades used in the past are introduced in rails (Papaelias et al., 2010). For this reason, surface and near-surface defects due to rolling contact fatigue (RCF) have become one of the most common fatigue defects in rails. RCF is a rail head and near-surface defect caused by wheel and rail interaction. *Figure 1* shows a rail section containing typical rolling contact fatigue (RCF) cracking.

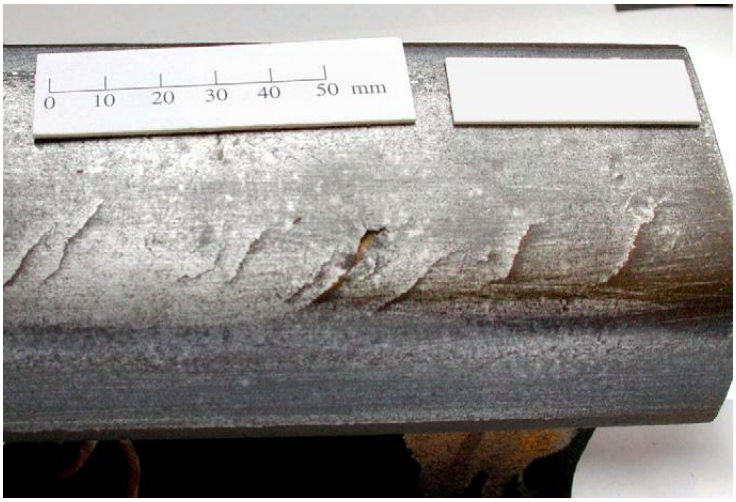


Figure 1 –Typical RCF cracking found in rails (photograph is courtesy of Dr Garnham)

The detection and quantification of RCF defects at the earliest possible stage is important, if maintenance costs are to be reduced followed by an improvement in existing safety standards. In order to reduce the extent of disturbance to railway system operations, non-destructive testing (NDT) is necessary. NDT is commonly used in infrastructure management by railway operators, such as Network Rail in Great Britain. Research on the application of NDT methods has been ongoing since 1877(Herring, 1877).

In the 1900s, Dr Sperry developed the first rail inspection vehicle using magnetic inducing sensors (Kube, 2005)(Bray, 2000). Since ultrasonic transducers were added to the Sperry test vehicles for the first time (Bray, 2000), the NDT concept for the high speed inspection of rails has been widely known. The rail industry commonly employs ultrasonic probes mounted on special test trains in order to inspect rails rapidly. But the inadequate detection on small (<4mm) surface defects of these ultrasonic systems these has been criticised for several years, and the potential application of other NDT technologies, including Alternating Current Field Measurement (ACFM) sensors(Lugg and Topp, 2006)(Topp and Smith, 2005)(Howitt, 2002), has been under investigation.

The NDT technique used in the project is the ACFM technique. The ACFM technique is based on the principle that an alternating current (AC) can be induced to flow in a thin skin

near the surface of any conductor carried by high frequency ultrasonic carrier wave (Papaelias et al., 2010). If there are no defects present, the introduced remote uniform current will be undisturbed, otherwise, the uniform current is disturbed and the current flow around the ends and down the faces of the crack. TSC Inspection Systems (UK), who is the patent holder of ACFM, invented this concept in the 1970s. The ACFM probe integrated the transmitting and receiving of the high frequency ultrasonic carrier wave, which is normally 50kHz. Then all the information of the rail surface is contained on the wave. Then the signal on the reflected wave is needed to be processed by using appropriate signal data processing technique. In 2000, with the support of Bombardier Transportation, TSC began the development of ACFM applications in the rail industry. The objectives were to develop an ACFM system with a friendly user interface, capable of automatically detecting, sizing and thresholding train wheelsets. In the initial tests on previously rejected train axles, the developed ACFM system achieved an 84% Probability of Detection (PoD) in comparison to 44% PoD for magnetic particle inspection (MPI)(Howitt, 2002), either due to failure on MPI or because of excessive surface corrosion, since the ACFM sensor requires no contact with component and can therefore be applied without the removal of surface coatings or grime. Following the experimental work on the train axles, it became distinct that an ACFM system could be applied to both detect and size RCF cracking on rails. This led to a new era of ACFM system.

Figure 2(a) shows an ACFM Probe, AMIGO interface electronics and custom software running on a supporting laptop. The ACFM probes scan the rail, then data is sent for processing in the yellow box called the AMIGO box. After the ACFM algorithm calculation, the ACFM results are then sent to be visualised on a PC. *Figure 2(b)* shows the ACFM walking stick. The incorporated 8-ACFM sensor array installed at the bottom of the stick, has been shaped to conform to the shape of the rail head surface, which allows the ACFM system

applying in both new and worn rails. The technician can perform inspections in real time by reading the curves on the screen.

(a)

(b)

Figure 2(a) -The ACFM Probe and the AMIGO box

Figure 2(b) -The ACFM walking stick

One of the limitations of current ACFM applications is the processing speed. Taking the AMIGO box and the ACFM walking stick as an example, they are electronics based system in which the analogue circuits are using to process data.

In order to allow higher speed operation, the system could be realized in digital circuits, which is a more efficient and economic method.

Digital signal processors (DSP) and field programmable gate arrays (FPGA) are the most popular conventional microprocessors that are used to implement digital signal processing operations. A DSP performs very efficiently compared to other off-the-shelf processors, especially for DSP-oriented tasks, such as complicated mathematical calculations due its internal structure. However, it may not be adequate for demanding tasks in some cases(Berkeley Design Technology, October 2002).

TSC Inspection Systems (UK) successfully implemented the one single ACFM processing chain on a DSP platform. It achieved a significant increase in the processing speed. However, an 8-ACFM sensor array is needed to cover the rail head surface. Also the efficiency of 8-ACFM parallel processing design is much higher than the serial processing design. In this

case, an FPGA is ideal for the 8-ACFM parallel processing chain expansion, since FPGA has the potential for very high parallelism(Berkeley Design Technology, October 2002). The internal structure of an FPGA is an amorphous "sea" of reconfigurable logic with reconfigurable interconnect, giving FPGA higher internal bandwidth and capacity for highly demanding tasks. An FPGA, however, will not necessarily be optimised for specific mathematical functions in the way that a DSP would. Therefore, TSC Inspection Systems (UK) began to consider building up an 8-ACFM sensor array and parallel processing system using an FPGA. FPGA is capable for parallel, but the bottlenecks needs to be addressed, which are speeds of external interfaces.

This project is conjunction with TSC Inspection Systems (UK) to develop an FPGA system for parallel processing of ACFM data from 8-ACFM sensor array.

The latest progress before this project is that one single ACFM processing chain has been built and verified using an FPGA. The author is carrying on the development and undertaking the research in expanding the single ACFM processing chain to an eight parallel ACFM processing chains system.

2 System overview

2.1 System requirements

This project aims at building the ACFM processing chain on the FPGA platform. The ACFM algorithm is the cumulative sum of the calculation results of a series of data acquired through ACFM sensor. The calculation of each data is a complex procedure, which includes division at carrier wave frequency and sampling frequency; trigonometry operations on the sampling starting phase; multiply operations on the polynomials etc. The intermediate variables are not all integer but decimals also. Therefore, to realise the ACFM algorithm on an FPGA, the challenge is the data format that should be used to represent non-integer values in FPGA.

Ideally fully floating point representation would be used, but that is computationally expensive. Therefore, fixed point representation is considered. Fixed point is a data type between integer and floating point. Fixed-point arithmetic is based on integer mathematics, so it can operate efficiently and fast.

In this design, the fixed point data format that is defined in VHDL package is used to perform the fixed-point arithmetic. The package defines two new data types: "ufixed" is the unsigned fixed point and "sfixed" is the signed fixed point.

In order to make the ACFM processing chain on the FPGA capable of verification and simulation, the system on the FPGA should be able to perform external communication.

The target system is generally divided into three parts. The first part is realised with a DSP Digilent board to perform real-time data acquisition.The second part is the main algorithm part, which is realised by using an FPGA development board to carry out mathematical calculations of the ACFM algorithm and get the data for visualisation. The third part is the terminal end - PC, receiving the data through a serial communication port from FPGA outputs and displaying the data on the screen. The system description will be laid out in Section 2.3.

The design has been integrated into an Xilinx Spartan-6 (XC6SLX45) FPGA chip (Figure **8**). The internal programmable logic components, known as “logic blocks”, perform the function realisation and link components. The architecture guarantees a fast internal processing speed in the FPGA. Moreover, using a re-configurable device makes the system very flexible. An upgrade of functions or new configurations can be made easily (Brizuela, Ibanez and Fritsch, 2010). Considering the three steps of the target system design, the third part is the bottleneck of the speed issue since the serial communication speed is much lower than the FPGA internal processing speed.

2.2 High speed processing specialty

Inspection speeds of up to 121.5 km/h were carried out using a turning lathe test rig for ACFM experiments. Results from the experiments carried out on the rail rig at the Birmingham Rail Research Centre at up to 48 km/h have also been reported (M Ph Papaelias, 2009)(Papaelias et al., 2009)(Papaelias et al., 2008). This project aims to achieve high speed parallel data processing in order to realise high-speed inspection on rail tracks. Initially, the project is set as doing experiment on the rail track surface using ACFM probes. However, due to the complicated connection work of the 8-ACFM probe as well as the shortage of the probe during the research, the real data provided by TSC company is used as the input. Regardless of the frequency of the ultrasonic carrier wave, FPGA should guarantee the processing speed, which should be fast enough to finish the calculation and transmission work during the gap between the adjacent two data points.

As mentioned the bottleneck of the speed issue in the system is the serial communication part.

The internal oscillator offers a 100 MHz global clock. The FPGA can adapt to a number of different configurations. The system parameter settings are discussed in detail in the methodology section (Section 4.3).

2.3 System description

The target system of this project is a totally new research field which has never been realized before. The ACFM probes are still playing the role of real-time data acquisition. And, the FPGA takes the responsibility of processing the data of the eight paths. Unlike a single probe, the digital data will be transferred to an FPGA board through multiple paths. In order to offer a high data processing speed, the input signals of FPGA from the multi-channel acquisition will be parallel processed. In the meantime, FPGA communicates with the output interface

and the data transmission rate (baud rate) needs increase to ensure sufficient speed for the eight paths processing chain.

2.4 The system with 8 parallel processing chains

The digital system of eight parallel processing chains consists of eight ACFM probes, a digital signal processor (DSP), an FPGA device and a PC (shown in *Figure 3*). The DSP is used as an analogue to digital converter (ADC) to import the raw probe signals and takes the responsibility of sending the ultrasonic carrier wave to the probes and delivering the digital data to the FPGA inputs through its serial peripheral interface (SPI). The FPGA is used to process the data of the eight paths in order to realise ACFM algorithm. The PC is used to display the curve of the ACFM results.

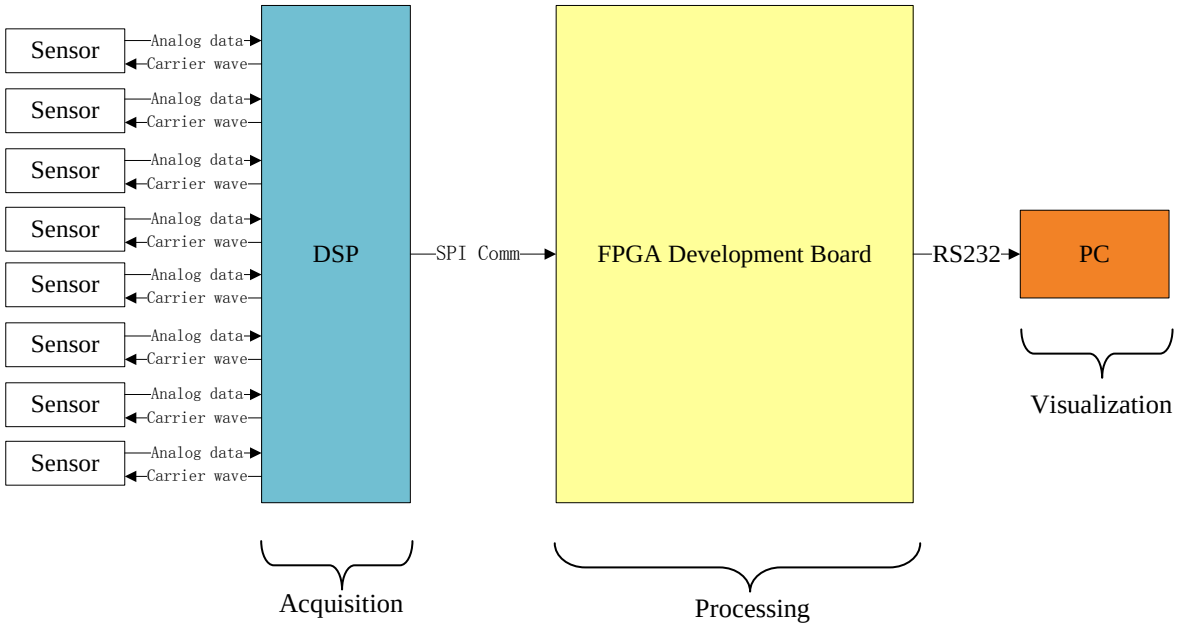


Figure 3 –The general FPGA based ACFM system structure for 8-ACFM array probes

The digital data converted in the DSP is sent to the FPGA through the SPI. So, the external communication between the DSP and the FPGA is serial data transmission. In order to guarantee the high speed specialty of the system, the communication speed of the SPI is one of the key parameters. To allow the communication speed of the SPI setting to be high, the

FPGA parallel processing structure is beneficial, since the data sent through the SPI do not need to be latched until the whole FPGA processing chain finishes working on the last round. Furthermore, the FPGA parallel processing structure is not simply eight copies of one single ACFM processing chain, but aims at achieving short time consuming of the processing with the least usage of FPGA internal components.

As mentioned above, the FPGA operates in a more sophisticated way than a DSP when facing DSP-oriented architecture features tasks, such as complicated mathematic calculations. So during the unpacking work of the single ACFM processing chain, the mathematical modules should be avoided to repeating in eight chains.

The serial communication through the RS-232 between the FPGA and PC is another key point for the realisation of high speed specialty. The baud rate setting should be considered, in order to meet the requirement of both high precision and fast bit flow.

2.5 Dissertation structure

In the dissertation, the development history of the NDT technique for rails, the development of digital signal processing and FPGA applications now in use are introduced in Section 3, the literature review. The tools used in the project, including software and hardware, and the system design are described in detail in Section 4, the methodology section. Simulation results and synthesis results of the eight ACFM channels parallel processing system are demonstrated in Section 5 and 6. The visualization on a PC is discussed in Section 7 and the DSP programme is included in Section 8. The conclusion of this project and the future work are discussed in Section 9 and references are in Section 10.

3 Literature Review

3.1 Overview

Railway services play an important role in transport networks in the UK and Europe. With the existing rail network, passengers can easily travel cross-border and intercity between the UK and Europe. Total passenger kilometres in 2007-08 were 49 billion, an increase of 6.0 percent in the United Kingdom on 2006-07 (Board, 2007-2008). This compares with an increase of 7.0 percent between 2005-06 and 2006-07. From these increasing figures in national railway development, it can be seen that the rail service is becoming a more popular transportation option.

Railway tracks are expected to withstand bending and shear stresses, plastic deformation and wear, but these still result in slow degradation of their structural integrity over time.

Under normal circumstances, rail failures can be divided into three types (Cannon, 2003):

I. Manufacturing defects;

II. Improper usage, handling or installation;

III. Structural degradation due to fatigue or corrosion of the rail.

Rolling Contact Fatigue (RCF) is considered to be the third type of failure mentioned above.

RCF is the rail head and near-surface defect caused by wheel and rail interaction. Defects in in-service rails can be present in the rail head, web or foot. This project concentrates on high-speed detection of rail head defects, so the ACFM technique which is capable to detect the rail head surface cracking could meet this requirement. Ideally the system used for identifying such defects should be a high speed train based automatic inspection. The equipment integrating all these features is still under research, the progress of which is good. Inspection speeds up to 121.5 km/h were carried out using a turning lathe test rig in ACFM

experiments. Results from the experiments carried out on the rail rig at the Birmingham Rail Research Centre up to 48 km/h have also been reported (M Ph Papaelias, 2009)(Papaelias et al., 2009)(Papaelias et al., 2008). Currently, in general, the ACFM technique is limited by various factors in addition to the speed of the data processing systems. Researchers are therefore currently seeking a more comprehensive and accurate way to enhance ACFM performance in order to meet the requirements of high speed detection. So far, some progress has been achieved. Researchers have verified the feasibility of using a digital processing system to detect cracks. However, the inspection "walking stick" (*Figure 2(b)*) is manually operated so far. The limitations mainly concern the coverage and the digital data processing system. These problems will be focused on in this project by developing a parallel processing system using an FPGA device connecting with an eight ACFM probes array.

3.2 Non-destructive testing (NDT)

To provide a good and reliable service, the railway companies check and maintain the rails on a regular basis. Railway safety has been a core issue in need of attention. Therefore, various Non-destructive Testing (NDT) technologies have been used to find cracks on rails. As discussed above, research on the application of NDT methods has been ongoing since 1877 (Herring, 1877), and the first rail inspection vehicle using magnetic inducing sensors was developed after 50 years (Kube, 2005)(Bray, 2000). Magnetic induction was the only technique available for the high speed inspection of rails until 1953, when ultrasonic transducers were added to the test vehicles for the first time (Bray, 2000). Since then, the NDE concept for the high speed inspection of rails has remained largely unaltered (Papaelias, Robert and Davis, 2008).

Ultrasonic inspection is the most common technology for checking rails and put into rail industry, the inspection speed of which can reach 75 km/h(Thomas, Heckel and Hanspach, 2006). In ultrasound, normally the ultrasonic energy generated by a piezoelectric element

beam is propagated to the rail, and a detection sensor collects the energy of the reflected transmission beam and the scattered transmission beam. This inspection is carried out by a variety of different instruments ranging from manual hand-held devices, through automatic dual-purpose road/track vehicles, to test fixtures that are towed or carried by dedicated rail cars. More recently, considering the ability to propagate around curved surfaces with little energy loss of Shear horizontal ultrasound, Steve Dixon in University of Warwick has used Shear horizontal (SH) ultrasound guided waves in measuring a 0.83 mm thick aluminium sheet with a shallow bend introduced and got good experimental agreement with an analysis model elucidates the reflection process. (Fan, 2007)

However, the performance of existing conventional ultrasonic probes in detecting small (<4 mm) surface defects such as head checks and gauge corner cracking is inadequate during high speed inspection (Papaelias, Robert and Davis, 2008). In addition, the presence of larger and more critical internal defects can be shadowed by smaller surface cracks during inspection. Because of these problems, the current international practice is to combine non-destructive evaluation of the rail network with preventative maintenance procedures, such as rail head grinding, in order to optimise the trade-off between maintenance cost and structural reliability (Schöch, 2004)(Anami, 2004)(Grassie, 2005)(Schöch and Heyder, 2003)(Schöch, Heyder and Grohmann, 2006)(Grassie, 2005)(Pohl, Krull and Meierhofer, 2006).

In order to overcome the problems with ultrasound inspection, conventional ultrasound transducers and Magnetic Flux Leakage (MFL) sensors are combined. In MFL, permanent magnets or direct current (DC) electromagnets are deployed to generate a magnetic field in order to magnetize the ferromagnetic specimen under inspection to saturation. The magnetic flux lines are coupled into the specimen using metal 'brushes' or air coupling. Once there are any anomalies or inclusions, the magnetic flux lines will leak outside the specimen in proximity of the anomalies and then the sensor or sensor array will detect the leakage

magnetic field, which corresponds to anomalies or inclusions such as corrosions, cracks, groovings, etc(Burd, April 2005)(Drury and Pearson, April 2005). The search coil, which is placed a fixed distance from the rail, is used to detect the changes in the magnetic field near the rail head generated by the DC solenoids (Clark, 2004). The method can detect smaller surface defects (cracks in the head and corner cracking) than normal ultrasonic. However, as the speed of inspection increases, the magnetic flux leakage sensor performance quickly deteriorates due to the decrease in magnetic flux density (Ireland and Torres, 2005) (Mandayam et al., 1996) (Li, Tian and Ward, 2006).

More recently, Pulsed Eddy Current (PEC) probes have been added on certain ultrasonic test trains to offer increased sensitivity in the detection of surface defects at high inspection speed (Clark, 2004)(Thomas et al., 2000)(Junger et al., 2004)(Thomas, Heckel and Hanspach, 2006)

The eddy current test method for rails was adopted by industry in 1999 by the German Rail plc. It is sinusoidal alternating electrical current of a particular frequency used to excite the probe, which is especially for checking the rail surfaces for RCF. The PEC technique improves from the eddy current method by using a step function voltage to excite the probe, the advantage of which is that it contains a continuum of frequencies, then the electromagnetic response to several different frequencies can be measured with a single step. On the other hand, the ultrasound technique aims at measurement in the rail bulk volume, which is not feasible using the PEC technique. However, experience gained from application has shown that clear improvement on rail inspection can be achieved. The defects that have been classified using ultrasound test can be further labelled as 'distinguished positions' using the PEC technique(Thomas, Heckel and Hanspach, 2006). The ultrasound technique combined with the PEC probes perform far better than MFL sensors at higher inspection speeds but are affected more by lift-off variation. The signal strength for a given defect decreases as lift-off increases with a cubic index (Papaelias, Robert and Davis, 2008).

Alternating Current Field Measurement (ACFM) probes ultrasonic test system is another system for the detection of orbital cracks. It has a high ability to detect the smaller near-surface and surface ruptures of the rail head. When lift-off is constant and while increasing speed, the ACFM signal is largely unaffected. Moreover, the signal strength for a given defect decreases as lift-off increases with a square index, which is an improvement compared with PEC probes (Papaelias et al., 2009).

The following sections describe, in more detail, how the ACFM technique works.

3.3 ACFM principle

Alternating Current Field Measurement (ACFM) is an electromagnetic detection method; it is able to detect surface breaking cracks in metal and measure the size (length and depth) of them(Howitt, 2002). The ACFM method is selected because it has high reliability for small near-surface and rail head surface cracks.

In contrast with PEC sensors, that requires to be placed at a close (<2mm) and constant distance from the inspected surface, when using ACFM probes the maximum lift-off can reach 5 mm. For larger threshold defects, several millimetres deep, a higher operation lift-off (>5mm) is possible(Howitt, 2002). In the ACFM method, a locally uniform, unidirectional current is induced into the rail, resulting in a magnetic field above the rail that can be measured. The constant current source and a constant magnetic field sensor-driven incident field are used to monitor changes in the surface magnetic field (Lewis et al., 1988). If the direction or strength of the current changes, the effect of the incident field on the flowing anion in the current would result in the magnetic field change based on the theory of electromagnetic effects. Based on these changes, it is possible to estimate the size of the cracks. All of these results can be obtained just through appropriate monitoring and analysis of the changes in the magnetic field. Current is undisturbed if there is no crack, however, if

there are cracks, the uniform current is interrupted. The presence of cracks disrupts the electric field distribution. Through appropriate monitoring of the magnetic field, the crack can easily be detected.

The magnetic field described above is a complex three-dimensional field. However, the magnetic field parameters can be measured by selecting the appropriate orthogonal axes (Dover et al., 1981).

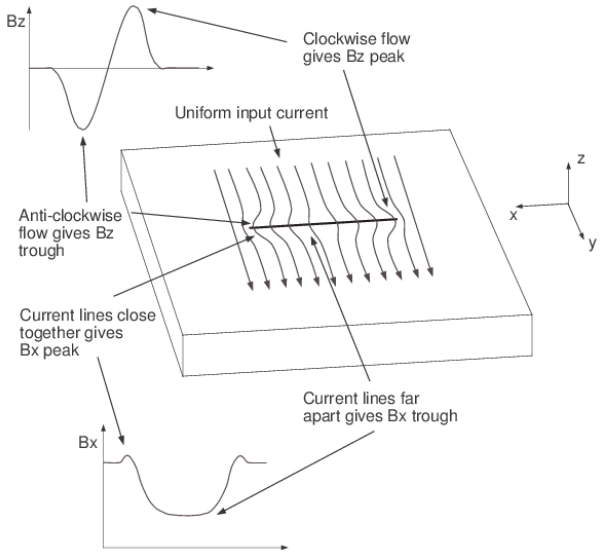


Figure 4 – Definition of field directions and co-ordinate system used in ACFM (Taken from reference(Nicholson and Davis, 2012))

Figure 4 shows the definitions of field direction and coordinate systems used in the ACFM. The rail section is lying in the x-y plane. The current flows in the y-direction which induces a uniform magnetic field above the rail section. The presence of a defect leads to changes to the current density around the crack since some of the current flows along the crack faces and around the crack ends. Hence the x-component of the associated magnetic field, B_x, is reduced over the majority of the crack length, but is increased at the crack ends. The severity of the defect can be assessed by measuring the maximum reduction in the x-component of the magnetic field from the background level. So the B_x component corresponds to the reduction in current surface density as the current flows down the crack and is indicative of the depth of

the defects (Papaelias et al., 2009). The B_z component is used in the analysis to give the length of the crack. The crack length can be obtained based on the distance between the trough and peak in the z-component of the magnetic field (Nicholson and Davis, 2012). The author is working on inspecting the depth of cracking. A standard depth indication figure is shown in *Figure 5*.

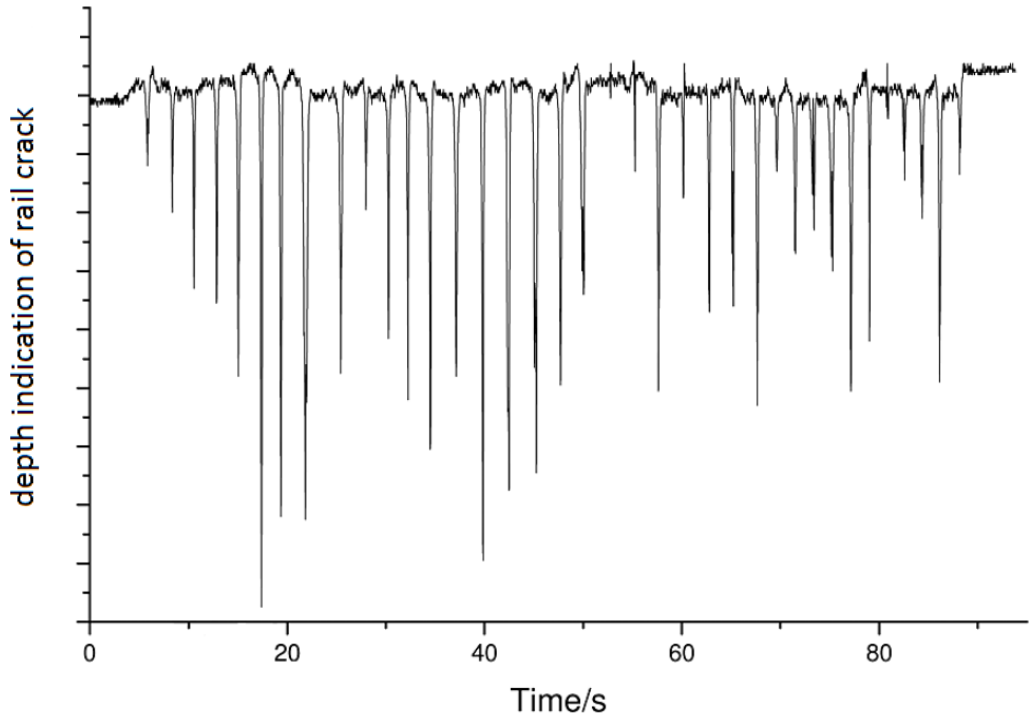


Figure 5 –Standard waveform of the Proportion of ACFM results on a rail section with spark-eroded slots(Papaelias et al., 2010)

The experiment is the ACFM inspection on rail surface breaking crack using a high-speed single-channel micro-pencil probe manufactured by TSC Inspection Systems with the lift off at 2mm. The ACFM response shown in *Figure 5* is the depth indication of rail cracks, which is B_x waveform. For a healthy rail section, the waveform should be a steady line always staying on the top. Also, the trough in the ACFM response increases with increasing defect depth. So there is more than one depth of cracks in the figure.

The author is working on inspecting the variation of the waveform, where is the indication the crack. And the working on getting from the proportion to measurement is undertaking by Dr Nicholson (Nicholson et al., 2013).

Therefore, by proper analysis, the approximation of the size of the cracks could be estimated.

ACFM probes can be used as standard pencil probes and array probes. These probes can be customised in order to optimise the inspection of particular structural component under certain situations. To make best use of an array probes, it is necessary to switch the sensors as quickly as possible in order to allow rapid inspection, and needs to balance the limitations, which includes switching settling times, data transfer rates and limitations in the signal sampling. With conventional analogue electronics, these factors limit the speed of scan for array probes to around 0.15m/s for a single field 16-channel array(Papaelias et al., 2009). In order to improve the performance of the ACFM array probes a high-speed instrument, digital construction device, has been developed which allows scanning speeds 4-5 times faster than the conventional ACFM instrument, which is achieved by increasing the energising frequency from 5 kHz to 50 kHz together with modifications to the signal processing electronics(Lugg and Topp, 2006)(Topp and Smith, 2005).

The ACFM experiments were carried out at inspection speeds up to around 100km/h using a turning lathe-based test rig. A high-speed single-channel micro-pencil probe manufactured by TSC Inspection Systems was utilised during testing. The pencil probe operates at a frequency of 50 kHz, which in digital processing point of view is 50 kHz carrier wave frequency. The data acquisition rate during tests is 2 MHz.

3.4 FPGA in digital signal processing application

Digital signal processing can be defined as representing a signal by a sequence of numbers or symbols and processes which are performed on the signals to enhance the desired energy

components and discriminate against noise and interference corrupting the signal (Grant, 1993). Digital signal processing has become increasingly common in our daily lives. It is used in a wide range of applications such as data communications, voice, audio, and biomedical signal processing (Jones and Watson, 1990). People can use conventional microprocessors, DSPs or FPGA boards to implement digital signal processing operations. DSP processors are specially designed for high computational performance and signal calculation.

In recent years, FPGAs have become increasingly attractive signal processing engines due to their large internal broadband and economical efficiency. FPGAs have now been used in various applications, which were generally previously implemented using DSPs. Currently FPGAs are used in various applications such as aerospace, designed with radiation tolerant FPGA (Gao and Teng, 2008); the education system, where implementation of a DSP trainer served as an educational tool to effectively teach the fundamental principles of digital signal processing (Rosula et al., 2008); and medical applications, serving in video and image processing systems for surgical and video recording and image enhancement for patient monitoring and diagnostic applications (Nair, 2008).

The Digital Signal Processor (DSP) was used for signal analysis in the previous work of TSC Inspection Systems (UK). However, as mentioned above, Field Programmable Gate Array (FPGA) technology is gradually replacing DSPs to do signal processing because of its advantage in flexible internal structure and the potential for very high parallelism(Berkeley Design Technology, October 2002). The speed requirements of non-destructive inspection on the railway system are very exacting. FPGA processing speed could meet these requirements, especially in the case of parallel data processing. In this project, an FPGA development board is used to perform the signal processing.

In the FPGA, each design can be summarised as a module. These ready-made modules can be easily ported to any other design, and perform the same function. Such a module can be applied in any design; this is called modularity.

In the past, ACFM systems were purely based on using the analogue circuits to do signal processing. The AMIGO box is the instrument developed by TSC to realise the ACFM inspection in an analogue circuit. The limits were that they could not process data at high speeds.

The project in conjunction with TSC Inspection System (UK) is to develop an FPGA system for parallel processing of ACFM data. The latest progress before this project is the real-time acquisition data processing chain of one single probe have been realised on FPGA. The results are encouraging and a good starting point for this work. However, the one single ACFM probe system cannot meet the requirements of automatic crack inspection. When the ACFM probes are fixed in position on a measurement train, the requirements should include covering the whole rail head because cracks can appear in different locations on the surface of the rail head. To meet this requirement, the multiple ACFM probes should be oriented in an array configuration across the rail head; in this way more information of the rail head surface could be collected. The information from multiple probes could also help with crack sizing.

The author undertakes the further research to expand the single ACFM system to eight parallel ACFM system. The objective of the project is to develop a digital system on one FPGA board to process the ACFM data acquired by the multiple ACFM probes array which connects with the DSP board (analogue to digital conversion). In order to ensure the data processing speed, the input signals to the FPGA from the multi-channel acquisition are chosen to parallel processed. At the same time, the FPGA output data transmission rate (baud rate) has to increase to ensure high-speed data flow.

3.5 FPGA for ACFM

The digital intelligent device chosen to implement the NDT method, the ACFM technique, is the FPGA development board in this project.

The economical efficiency and high speed processing specialties of FPGA have brought it into researchers' field of vision. FPGAs have been applied in detecting railway wheelflats. The system realised on FPGA operates with the train moving at low speed over a measuring rail. Ultrasonic surface wave pulses are sent at regular intervals and echoes are acquired and processed by the system (Brizuela, Ibanez and Fritsch, 2010). Applying FPGAs to rail inspection, the digital method has been validated and realized by researchers with real-time acquisition at low speed in manual inspection.

The progress having achieved in this project is that the ACFM system has been connected with one single ACFM probe, which has been realised on the FPGA(Ma, 2010-2011).

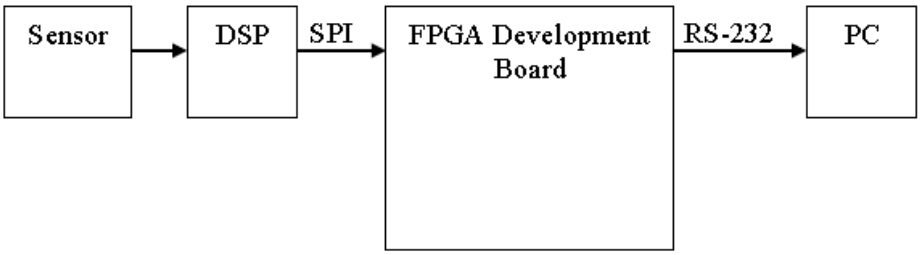


Figure 6 –The general ACFM system structure with single probe

Figure 6 shown above is the general ACFM system structure with one single ACFM probe. The real-time acquisition is realised by the ACFM probe. The conversion of analogue signal to digital signal is implemented by the DSP chip. After the FPGA board finishes the data processing, the processed data is passed to the computer via the RS-232 protocol.

4 Methodology

From *Figure 3*, the general system structure diagram of the multiple paths ACFM system, it can be seen that the system is comprised of a DSP development board, an FPGA device and a PC. In this section, each of these three parts will be discussed.

Before moving to the description of each part in the system, the working environment and tools will be introduced first.

4.1 Design tools

4.1.1 Software tools

1) ISE Design Suite 13.2

The integrated system environment (ISE) design suite is developed by Xilinx Company. It is a tool used to provide users with a programming design, synthesis and simulation environment, and it is compatible with third-party simulation software. This design tool achieves a perfect combination of design and productivity. In this project, the designer has chosen the FPGA chip from Xilinx Company. Xilinx ISE Suite13.2 is used to program the FPGA development board.

Specifically, with Xilinx ISE Suite13.2, users can program in HDL language, compile code and synthesis. A simulation tool is also integrated. The compiled program with the design of the user can be downloaded into the hardware through the JTAG (Joint Test Action Group) cable, which was initially devised by electronic engineers for testing printed circuit boards using boundary scan and is still widely used for this application.

2) Xilinx Core Generator System (IP Core Generator)

The Xilinx Core generator system is a tool used to deliver the parameterised functional modules which have already been pre-designed and optimised. The aim of this system is to accelerate the FPGA designing procedure and avoid re-inventing the existing modules which are frequently used by designers. Xilinx Intellectual Property (IP) is the key building block of Xilinx Targeted Design Platforms. It helps in maximising design flexibility and reducing risk, also reducing project duration. The cores are delivered through the Xilinx CORE Generator System and integrate seamlessly with the Xilinx design flow.

In this project, the Xilinx CORDIC v4.0 is used to perform the calculation on the sine and cosine functions of the input phase angle. The inputs and outputs are represented using fixed point numbers in order to interface with the other cores. Floating point elements are used within this core.

Having introduced the software tools used in the project, an introduction to the hardware will follow.

3) ModelSim 10.0 (Simulation)

In this project, ModelSim is used to perform the simulation on each module from low level entity to top-level entity. The simulation is considered as an important part of the design verification process. Considering the compatibility with Xilinx ISE Suite 13.2, ModelSim 10.0 is the suitable version to use.

Figure 7 shows the simulation flow that used in this project.

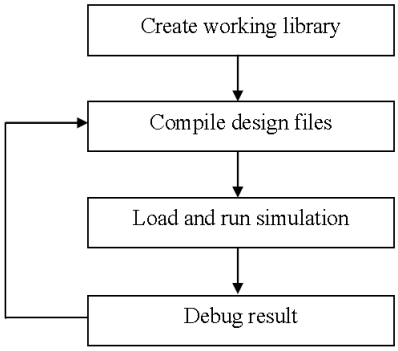


Figure 7 – Basic Simulation Flow

In this simulator, there are many debugging methods. The most common method is to view the waveform of each signal with time, in which way the behaviour of each signal can be easily inspected. Once the behaviour of the module is not what is expected, ModelSim, using its robust debugging environment, could help to track down the cause of the problem.

4.1.2 Hardware - ATLYS

In this project, the Atlys FPGA board is used to perform prototyping. The picture is shown below in *Figure 8*.

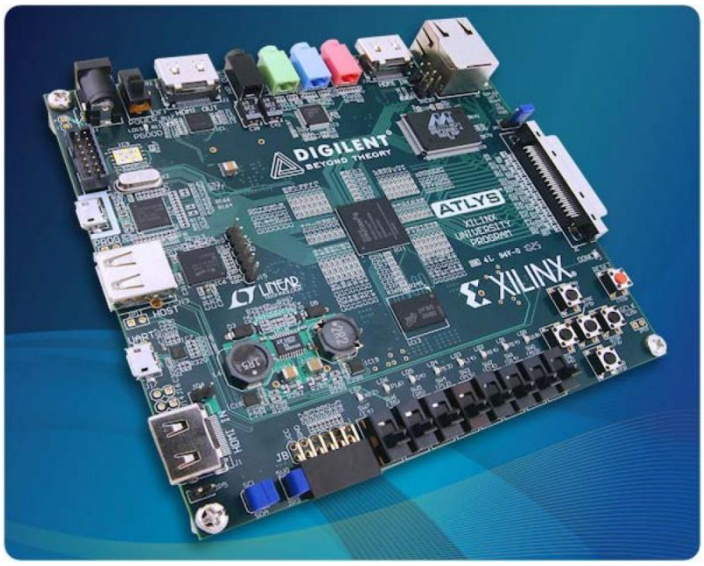


Figure 8 –Picture of ATLYS FPGA Board

The Atlys board combines a high-capacity Spartan-6 FPGA with the circuits and devices needed to create the comprehensive design platform. The board has On-board USB-based FPGA configuration circuitry, which makes downloading the project file (.bit file) to the chip much more convenient than download through the JTAG cable. Moreover, On-board, real-time power supply monitors provide the board a safe power environment and also provide the users with a safe working environment. The board also has advanced clocking circuits and the crystal oscillator on-board is 100 MHz.

The Atlys board has a 68-pin VHDC connector for high-speed/parallel I/O and an 8-pin Pmod connector for lower speed and lower pin-count I/O. Two pins from the 8-pin Pmod connector are used to send data to PC. In this project, the other 6 IO are used to perform the SPI communication with the DSP board.

In FPGA development, it is common practice for people to start their prototyping on an FPGA development board before moving on to produce the final product. The same practice is applied in this project. The important thing is that the chosen development board must utilise an FPGA chip that consists of large enough number of logic gates, cores, multipliers, etc. to support the implementation. At the beginning of the project, the Xilinx Spartan-3AN FPGA Starter Kit Board, which consists of Spartan 3AN (XA3S700A) was used. It is sufficient to be used in a one-single path system, but not enough for expanding into eight. The Atlys FPGA board which comes with Spartan-6 (XC6SLX45) is suitable for this project. The development board helps to shorten the development time and also the time to market the product.

The FPGA development board is used to verify the processing algorithm in FPGA. The FPGA development board helps in speeding up the development process by eliminating the need to construct the hardware connection with an FPGA silicon device. The FPGA development board also provides existing various interfaces which can be used to

communicate with other devices, such as the 68-pin VHDC connector and 8-pin Pmod connector in the ATLYs board.

Having introduced the working tools and devices, the desired system designs are now considered.

4.2 System parameter setting requirements

The project is defined in conjunction with TSC Inspections Systems Ltd. The requirements they expect are described below as mandatory features and preferred features.

Mandatory features

1) Process NDT data from 8 channels
2) The analogue to digital convertor (ADC) in each channel runs at 1 MSPS (Mega Samples Per Second)
3) 12-bit quantisation for the ADC and 16-bit data processing in FPGA
4) An eight channel parallel ACFM algorithm implementation in FPGA
5) The baud rate can reach 1 Mbps
6) The input interface of FPGA could fit into an eight DSP chip output interface
7) Synchronization of 8 probe data processing chain
8) Visualisation of 8 channels' results

Preferred features

1) The ADC in each channel runs at 2 MSPS.
2) The structure of the DSP system could be simplified from 8 DSPs to 1 DSP.
3) The system could accommodate other parameter configurations, such as carrier frequencies, sample frequency and sampling starting phase, etc.

The first mandatory feature (MF1), fourth mandatory feature (MF4) and seventh mandatory feature (MF7) refer to the capability of the processing chain on FPGA, which should be able to operate synchronized parallel ACFM algorithm and perform external communication with ACFM data from 8 channels.

The second mandatory feature (MF2) and first preferred feature (PF1) are two corresponding requirements, and PF1 is the enhancement of MF2. By increasing the sampling rate of the ADC, a higher resolution could be achieved and then higher speed detection could be performed.

MF3 decides the quantisation resolution of ADC and data width of ACFM data. Assume the normalized analogue voltage is 0 to 5V, the resolution of 12-bit quantisation is 1 mV.

MF5 decides the baud rate of SCI communication between FPGA and PC terminal. As mentioned before, the SCI communication is the bottleneck of high speed realization. The front of the system design is based on 1 Mbps baud rate. The details are discussed in Section 4.3.

MF6 and PF2 are two corresponding requirements and PF2 is the enhancement of MF6. The first part of system is real-time data acquisition, which could be realized by using eight DSPs or one DSP connecting to eight ACFM probes. The eight DSPs connection saves work on DSP programming, but the structure of which is complicated. In contrast, the one DSP connection demands complicated programming, but the structure is simple and neat.

MF8 is the third part of system requirement, and the visualization tool is optional.

PF3 is the requirement of the scalability of system, which allows different parameter settings into the system for the future possible extension.

The system design is strictly based on the principles listed. The detailed system design is discussed in the methodology section (Section 4).

4.3 Desired system design

The system level design will be discussed in this section.

In the system level design, the most important issue is how to transmit the bit flow acquired from the top to the end. The transmission inside each part and between each part both need to be taken into consideration.

Table 1 shows the input combination that is used in the simulation. These example data are provided by TSC Company, which are collected using an ACFM probe of the designed settings (discussed in Section 4.2).

Table 1 –Two frames of the real data from TSC Company

	Frame 1	Frame 2
Element	**Clip 1**	**Clip 2**
N	40	40
β	260	260
Fc, LSB	50000	50000
Fc,MSB	0	0
Fs,LSB	33950	33950
Fs,MSB	30	30
a[0]	834	832
a[1]	770	763
a[2]	736	714
a[3]	758	761
a[4]	788	754
a[5]	888	867
a[6]	998	974
a[7]	1162	1129
a[8]	1361	1325
a[9]	1540	1510
a[10]	1765	1726
a[11]	1988	1969
a[12]	2234	2167
a[13]	2495	2457
a[14]	2692	2659
a[15]	2939	2935
a[16]	3163	3132

	Frame 1	Frame 2
Element	**Clip 1**	**Clip 2**
a[17]	3343	3294
a[18]	3522	3492
a[19]	3612	3598
a[20]	3745	3741
a[21]	3777	3794
a[22]	3832	3801
a[23]	3824	3864
a[24]	3750	3760
a[25]	3697	3704
a[26]	3547	3578
a[27]	3415	3417
a[28]	3235	3290
a[29]	3021	3036
a[30]	2813	2841
a[31]	2556	2601
a[32]	2332	2346
a[33]	2092	2092
a[34]	1824	1916
a[35]	1628	1632
a[36]	1430	1443
a[37]	1208	1245
a[38]	1071	1104
a[39]	942	920

From the table it can be seen that two sets of data are provided, and each of them is one frame of converted digital ACFM data and has 46 elements. The frame consists of 6 header data (n_h = 6) and 40 samples (N = 40).

In the frame, Fs (ADC sampling frequency) is represented as a combination of MSB (Most Significant Bit) and LSB (Least Significant Bit), since the data width is 16-bit, the maximum value of which is $2^{16} = 65536$. So the Fs value in the frame is $(30 \times 2^{16} + 33950)\text{Hz} = 2$ MHz.

The parameters in the table are the carrier wave frequency "Fc", sampling frequency "Fs", number of sampling points in each clip "N" and sampling starting phase " β " (between zero to two pi). These parameter settings are in the header of each frame ("clip"). Only a[n] changes according to the condition on the rail tracks.

The front part, the DSP, sends a 50 kHz carrier wave to the ACFM probes. The probes send the wave to the inspected track surface and get the reflected wave back. At the same time, the 2 MHz sampling rate ADCs in the DSP, which are working all the time, now sample the carrier wave to get 40 points to store in a[n]. The whole process repeats eight times, and then it goes back to the beginning. The data will be stored in the DSP memories before it is sent to the FPGA through a SPI by a 6.25 MHz communication clock.

In order to guarantee the high speed bit stream, the communication speed is set at 6.25 Mbps, since with this speed, the time gap between each time sampling is less than 0.1 milliseconds. And even if the train is running at 400 km/h, the distance between the two adjacent sampling times is shorter than 0.6 mm, which quite satisfies the cracking size of ACFM equipments detection.

A summary of the system settings is set out below:

1) Number of paths N_P = 8

2) Carrier wave frequency Fc = 50 KHz

3) ADC sampling frequency Fs = 2 MHz

4) ADC sampling data bit width W_b = 16 bits

5) Sampling points in each period N = 40

6) Header length in each frame n_h = 6 (detail discussed in Sections 3.3 & 5.2)

7) SPI communication clock F_{SPI} = 6.25 MHz

8) SCI transmission speed baud rate = 460800 bps

9) Number of serial ports used = 2

In *Figure 9* is the desired system flow chart with settings.

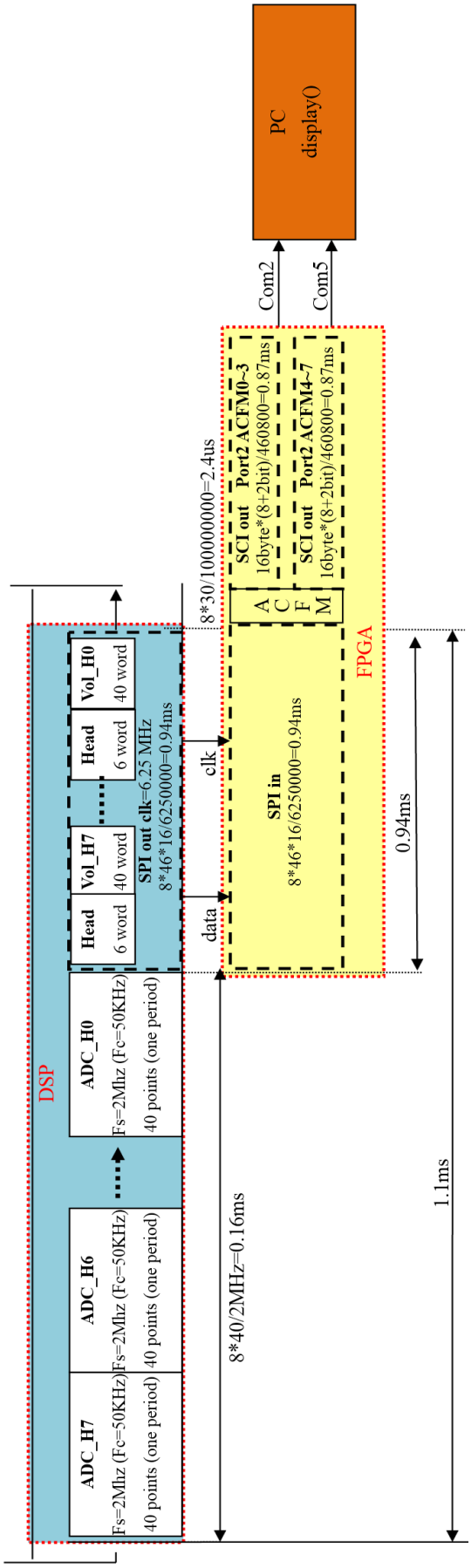


Figure 9 –Desired system flow chart

With these system parameter settings, the approximate time consumption of each part of the system can be viewed. They are listed below:

1) The time consumption of eight ADCs sampling one frame is the time spent on sampling 40 samples in eight channels; the ADC sampling frequency is 2 MHz.

$$t_{ADC} = 8 * \frac{40}{2MHz} = 0.16 \text{ ms} \quad (1)$$

Equation 1: Time consumption of eight ADCs sampling one frame

2) The time consumption of the SPI transmitting one frame in eight paths is the time spent on the 46 16-bit elements (6 elements in the header and 40 samples) in one frame in eight channels; the SPI communication bandwidth is 6.25 MHz.

$$t_{SPI} = 8 * \frac{(40+6)*16}{6.25MHz} = 0.94 \text{ ms} \quad (2)$$

Equation 2: Time consumption of SPI transmitting one frame in eight paths

3) Two serial ports are used for SCI transmission, and each port transmits the final results of 4 channels. As stated in RS-232 protocol, the 16-bit data needs to be divided into 8-bit MSB and 8-bit LSB to transmit, and 1 bit staring bit and 1 bit stop bit need to add into the 8-bit data.

$$t_{SCI} = 4 * \frac{16*(8+2)}{460800} = 0.18 \text{ ms} \quad (3)$$

Equation 3: Time consumption of SCI transmission

4) So the time consumption allowance of the ACFM algorithm in FPGA:

$$t_{ACFM} < t_{SPI} \text{ and } t_{ACFM} < t_{SCI}$$
$$\Rightarrow t_{ACFM} < 0.18\, ms \quad (4)$$

Equation 4: Allowed time consumption of ACFM algorithm realisation in FPGA

The settings are aimed at meeting the system requirements of TSC Inspection Systems (UK). From the timing issue analysed above, it can be seen that the time consumption inside the

FPGA processing is small compared with the data transmission speed between each device. Hence, it indicates that the FPGA device is suitable for this system, but the bottle neck is still in the non-internal data transmission between each part. More specific is the SCI transmission speed between FPGA and PC, and the transmission speed between DSP and FPGA.

For the SCI transmission part, the baud rate which could be achieved is the speed indicator. In the following section, the speed that the baud rate could reach will be tested.

4.4 Serial communication performance test

As the speed issue is important in this project, and the internal high speed data processing specialty of FPGA is admitted, before building the multiple paths system, the transmission speed should be the first factor taken into consideration. The standard baud rate of serial transmission is from 9600 bps (bit per second), and including 19200 bps, high value 115200 bps, no specific maximum value. The previous digital processing chain suffered from low SCI bandwidth (19200). The performance needs to be enhanced. Testing is executed by using Matlab to catch the programmed bit stream and calculate the bit error rates, then draw the transfer precision curve in the graph. The serial transfer performance testing results are shown in *Figure 10* below.

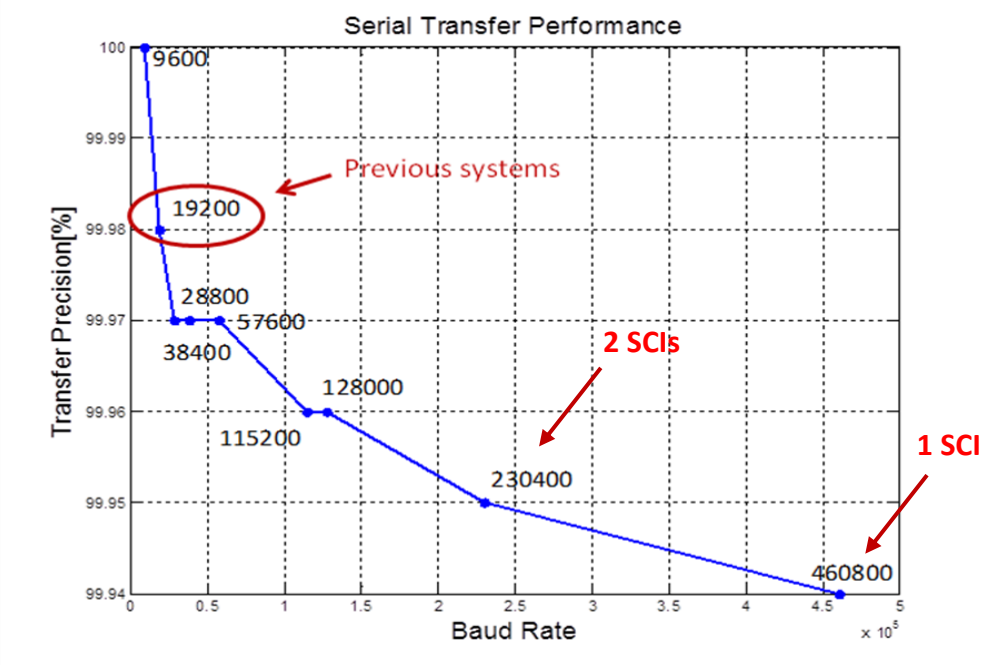


Figure 10 –The serial communications performance testing curve

It illustrates that a baud rate higher than 0.5 Mbits could be achieved. It can also be seen that the transfer precision does not vary much. However, the transfer precision tolerance should be set as high for the high speed processing and transmission. Considering this factor, two SCIs are used in this system. The structure of two SCIs is more complex in the implementation of FPGA, but higher transfer precision could be achieved in comparison with the one SCI structure. Therefore the higher transfer precision and more complex structure method is chosen in this project.

The baud rate is set at 460800 bps for the two SCIs. Each of them is in charge of four paths' data transmission. Therefore the baud rate of the eight paths bit stream is approximately 1 Mbps. There is already a very high speed transmission in RS-232. However, further speed tests will be carried out after the top system work.

We have now considered and selected the system design. The ACFM algorithm which needs to be implemented in the system next needs to be considered.

4.5 ACFM algorithm

The processing algorithm that needs to be implemented in the FPGA device is shown in *Equation 5*.

$$A = \sum_{n=1}^{N} a[n]\{P\sin(R[n-1]) + Q\cos(R[n-1])\} \quad (5)$$

Equation 5: ACFM Algorithm

A in this equation is the ACFM result. From the equation, it can be seen that the ACFM result is summation of a series from n equals to 1 to N, in which N is the number of sampling points in one "clip". The P, Q, R parameters are constants in this formula, which are only related to the carrier wave frequency "Fc", sampling frequency "Fs", number of sampling points in each clip "N" and sampling starting phase " β " (between zero to two pi). These constant settings are in the header of each frame ("clip"). Only a[n] are changing according to the condition on the rail tracks. The equation has been realised in Matlab. The Matlab implementation is the preparation for the following simulation and verification of FPGA behaviour. The results calculated in Matlab could be used as the criterion.

Based on the specialty of the ACFM algorithm, it is possible to make a master path to calculate the constants in the formula, such as P, Q, R parameters, only once, and deliver them to the other seven paths, which are named the slave paths in this project, in every "clip". Using this method could avoid wasting plenty of spaces in the FPGA chip, making the system more productive and efficient.

Having understood the ACFM algorithm, the following task is to see how the algorithm could be realised in the FPGA device. The progress of one-single ACFM path system has been achieved. The following section is to briefly describe the working principle of the system.

4.6 The one-single ACFM path system

The one path calculation chain has debugged through at the beginning of this project. How this system works needs to be clear, since the processing chain in the master path is quite similar. Furthermore, the bits which are necessary in both the master path and the slave paths need to be built eight times, whereas the bits which are only needed in the master path or which are needed to modify the slave paths, are only going to be built once.

The block diagram of a one-single path system is shown below in *Figure 11*, where "θ" represents R[n-1], "An" represents a[n] in *Equation 5*.

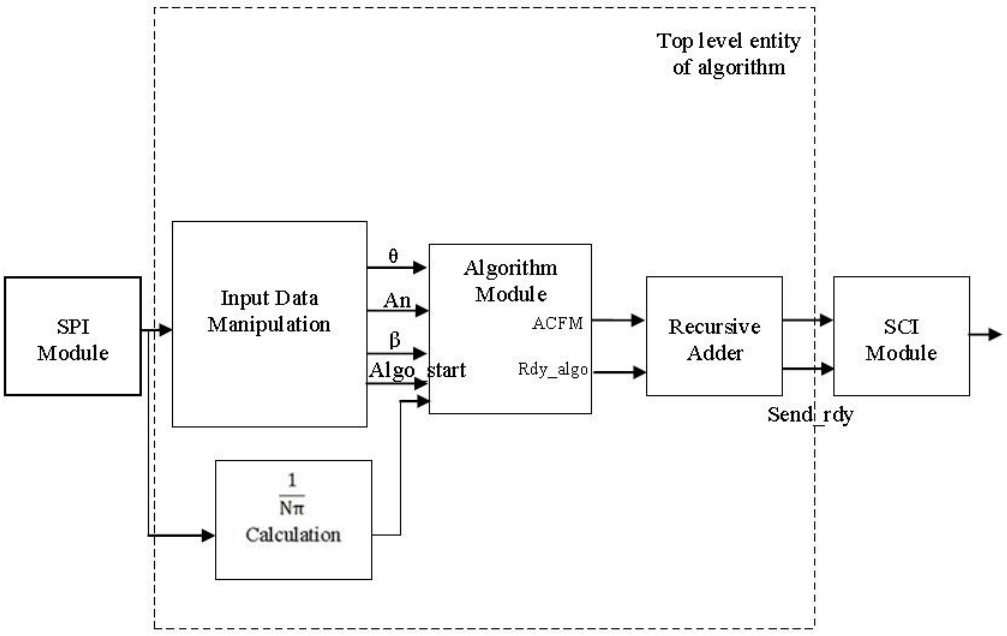


Figure 11 –Block diagram of one-single path system(Ma, 2010-2011)

The functions of each module are listed below:

1) SPI Module: Receive the data stream and control signals from the SPI module functioning in SPI mater mode in DSP, performing in SPI slave mode itself.
2) Input Data Manipulation: Manipulate the data contained in each frame. Separate the parameters N, β（sampling starting phase）, Fc (carrier wave frequency), Fs (sampling

frequency) and a[n] (data sampled from the reflex carrier wave), then send them to the following algorithm module.

3) $\frac{1}{N\pi}$ Calculation: Receive the data from the SPI module in front and calculate $\frac{1}{N\pi}$ for preparation. The calculation is in sfixed format, the process is complex.
4) Algorithm Module: Receive the separated data from the front modules, realise ACFM formula, and calculate the ACFM value for each a[n]. When finished, the result and control signal are sent to the following recursive adder module.
5) Recursive Adder: Receive the ACFM value of each a[n] in one frame, and repeatedly add them to get the final ACFM result for one frame. When finished, the final result and control signal are sent to the SCI module.
6) SCI Module: Get the 16-bit final ACFM result parallel from the front module, and then transmit it to the PCI through a serial port by RS-232 protocol.

As mentioned above, it is important to understand how this system works. The first step of extending the one-single path to the eight paths is unpacking the former system to find out the bits which need to be built eight times in all the eight paths and the ones which need not extend but are just built in the master path.

The following section explains how the unpacking work is executed.

4.7 Unpacking the one-single ACFM path system

The unpacking work is not only unpacking the structure, but finding out the useful bits and repacking them in the master path and the slave paths.

- Unpacking work in the top-level entity implementation of the master path

In the view of the master path, all four modules in the top-level entity of the algorithm in the one-single path system are needed. The structure is also the same.

The top-level block diagram of the master path is shown below in *Figure 12*, where "θ" represents R[n-1], "An" represents a[n] in *Equation 5*.

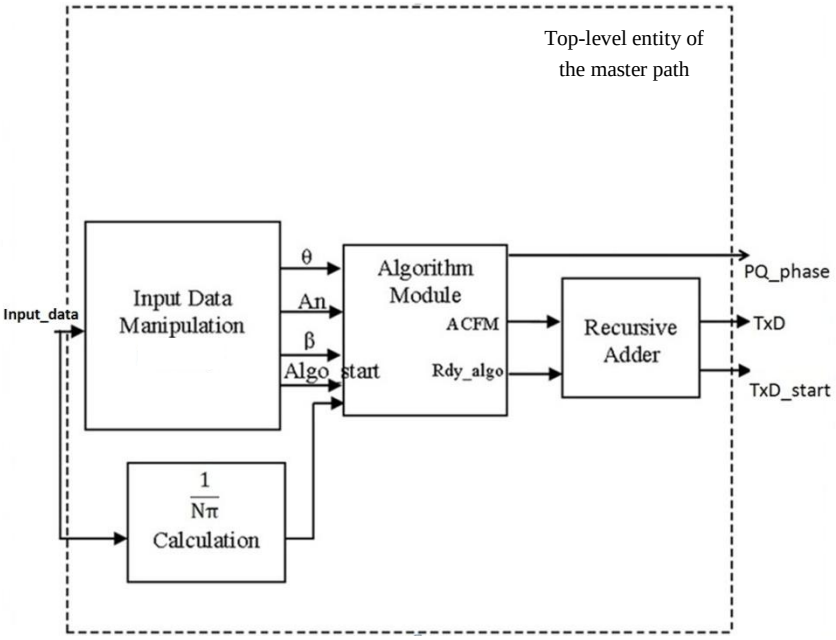


Figure 12 – Block diagram of the master path

From the diagram above, it can be observed that the master path top-level module consists of an input data manipulation module, $\frac{1}{N\pi}$ calculation module, algorithm module, and recursive adder module.

1) Input Data Manipulation: Manipulate the data contained in each frame for the master path. Firstly, separate the parameters N, β, Fc (carries wave frequency), Fs (sampling frequency) and a[n] (data sampled from the reflex carrier wave), then calculate $\theta = R(n-1) = \frac{2\times\pi\times Fc}{Fs}$. After the calculation is done, N, β and θ are sent to the following algorithm module.
2) $\frac{1}{N\pi}$ Calculation: Receive the data from the SPI module in front and calculate $\frac{1}{N\pi}$ for preparation. The calculation is in sfixed format, the process is complex.
3) Algorithm Module: Receive the separated data from the front modules, realise ACFM formula. Firstly, calculate

$$P = 4\cos\beta \times \frac{1}{N\pi} \tag{6}$$

Equation 6:'P' parameter representation

And

$$Q = 4\sin\beta \times \frac{1}{N\pi} \tag{7}$$

Equation 7:'Q' parameter representation

Then, with P, Q, and

$$\theta = R(n-1) \tag{8}$$

Equation 8:' θ *' parameter representation*

the ACFM value for each a[n] could be calculated. When finished, the result and control signal are sent to the following recursive adder module.

4) Recursive Adder: Receive the ACFM value of each a[n] in one frame, and repeatedly add them to get the final ACFM result for one frame. When finished, the final result and control signal are sent to the SCI module.

After the input data manipulation, the constant parameters and changing a[n] have been separated and delivered to the algorithm module. The " θ " here still represents $R[n-1]$ in the ACFM formula mentioned above. What needs to be declared is the "PQ_phase", the output of the master path. It is the combination of all the constants, which is

$$PQ_{phase} = P\sin(R[n-1]) + Q\cos(R[n-1]) \tag{9}$$

Equation 9: PQ_{phase} *representation*

- Unpacking work in the top-level entity implementation of the slave paths

The design of the slave paths is based on the specialty of the ACFM algorithm, it is possible to make the master path to calculate the constants in the formula, such as P, Q, R parameters

only once, and deliver them to the other seven paths. So the signal "PQ_phase", the interim signal inside the master path, is led out to the seven slave paths. "PQ_phase" is the combination of all the constants, which is demonstrated in *Equation 9*. With it, most mathematic modules could be removed.

In this project, all the other seven paths except the master path are the slave paths. The main reason for the slave paths depending on the master path is the constant parameter combination, "PQ_phase" as discussed above. The constant parameter combination has been calculated by the master path, and the signal "PQ_phase" contains the value of it. With the "PQ_phase" value, in the slave paths, $\frac{1}{N\pi}$ calculation module is not needed, and θ, β need not be separated from input data. Moreover, the internal structure in the algorithm module is much simpler. *Figure 13* below shows the top-level block diagram of the slave paths.

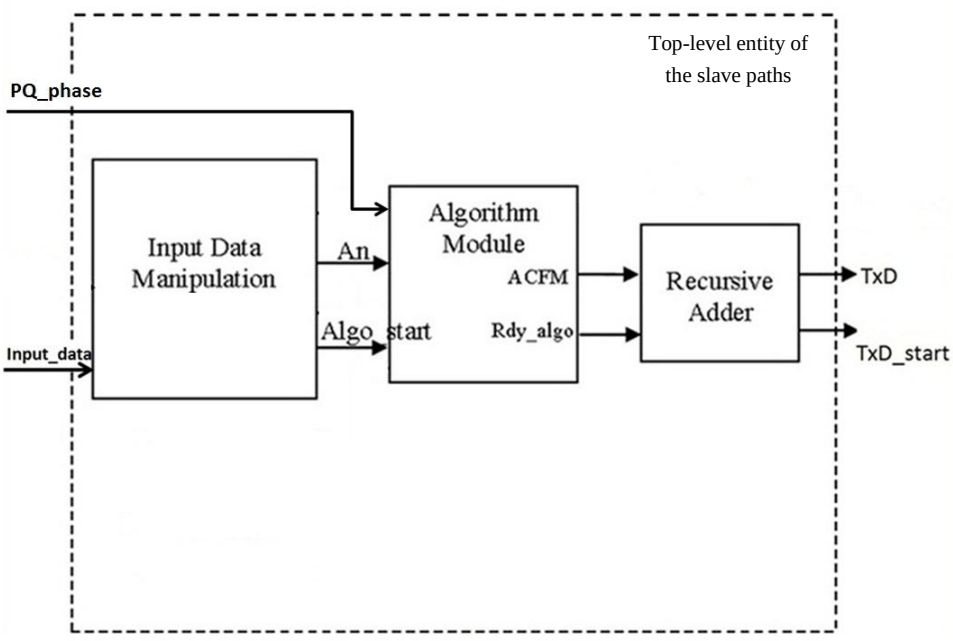


Figure 13 – Block diagram of the slave paths

From the diagram above, it can be observed that the slave path top-level module consists of input data manipulation module, algorithm module, and recursive adder module.

The top-level entity of the system still consists of input data manipulation module, algorithm module, recursive adder module, but no $\frac{1}{N\pi}$ calculation module. The function of each module is the same as the master path. The simplification is much greater than reducing the one module. The lower-level structure is simpler and less time consuming. This can be seen in the following lower level module description.

Having decided the top-level structure of the master path and the seven slave paths, the top-level structure of the eight paths system should be considered. The following section is to describe the eight paths system structure.

4.8 The eight paths system structure

The eight paths system consists of the master path and the seven slave paths. The structure of each path has been discussed in the previous section. In order to put them together, a higher level entity is needed, which is the eight paths system. The example input system diagram of the eight paths is shown below in *Figure 14*.

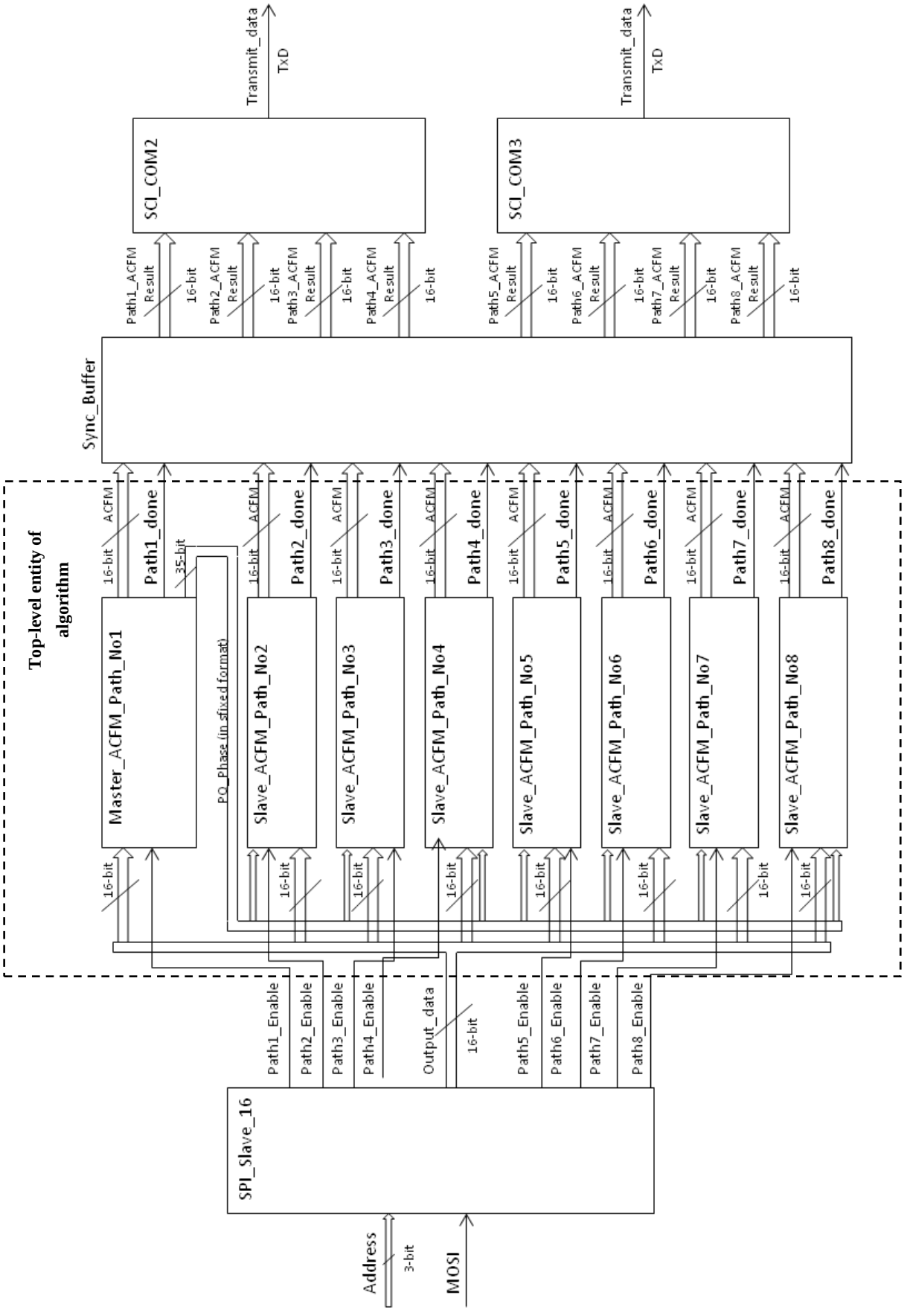


Figure 14 – System diagram of the eight paths

It can be seen from the diagram that the data processing chain in the system is generally divided into four parts: SPI slave, algorithm realisation in the eight paths, synchronisation and two SCI modules.

The top-level entity of the algorithm realisation part in the system is the data processing chain in the eight paths, which has already been discussed in the previous section. Let us then focus on the four parts in the system.

1) SPI Module: Receive the eight-path data stream and communication clock (SCK), chip select signal (SSEL) and three path address signals (Address2, Address1, Address0) from DSP SPI Master. The SPI Module will generate eight paths to enable the signal to tell which of them should get the output 16-bit data. To ensure high-speed communication, the communication clock is set at 6.25 MHz
2) 8-paths algorithm module: This is the top level entity of the eight paths algorithm structure. This top level entity is divided into the master path processing chain and the seven slave paths processing chain. The structure of each of them is discussed in the previous section.
3) Synchronize buffer: This module buffers the ACFM results for all eight paths, and guarantees the results can arrive at the SCI module at the same time.
4) SCI Modules: These two SCI modules are not in the diagram as well, but are very important. They provide the data transmission between the FPGA board and the PC through a serial port by RS-232 protocol. Two SCI modules are used, each of which transmits the data from four paths. The SCI module gets the 16-bit final ACFM result parallel from the front module and then transmits it to the PCI through a serial port by RS-232 protocol. The baud rate is set at 460800 bps.

The SPI slave part is to receive the bit stream and control signals from the front DSP SPI master module. The data transmission speed between DSP and FPGA should be high in order

to achieve high-speed testing on rail tracks. The communication clock between is therefore set at 6.25 MHz in this project. The bottleneck also occurs in this part, as well as the SCI part in the baud rate issue.

Having discussed the top-level structure of the eight paths system, the lower level structure implementation should be considered. Considering the algorithm part of the system is the core part to realise the ACFM technique in the FPGA device, the lower-level structure of the algorithm path will be described in the following section. The master path and the slave paths will be discussed separately.

4.9 The low-level entity implementation in the master path

The master path top-level module consists of input data manipulation module, $\frac{1}{N\pi}$ calculation module, algorithm module and recursive adder module. Among the four modules, the algorithm module can be considered as the core. Most of the calculation is done within this module. Then, let us focus on the low-level structure of the algorithm module. *Figure 15* shows the block diagram of the algorithm module in the master path, where "θ" represents R[n-1], "An" represents a[n] in *Equation 5*.

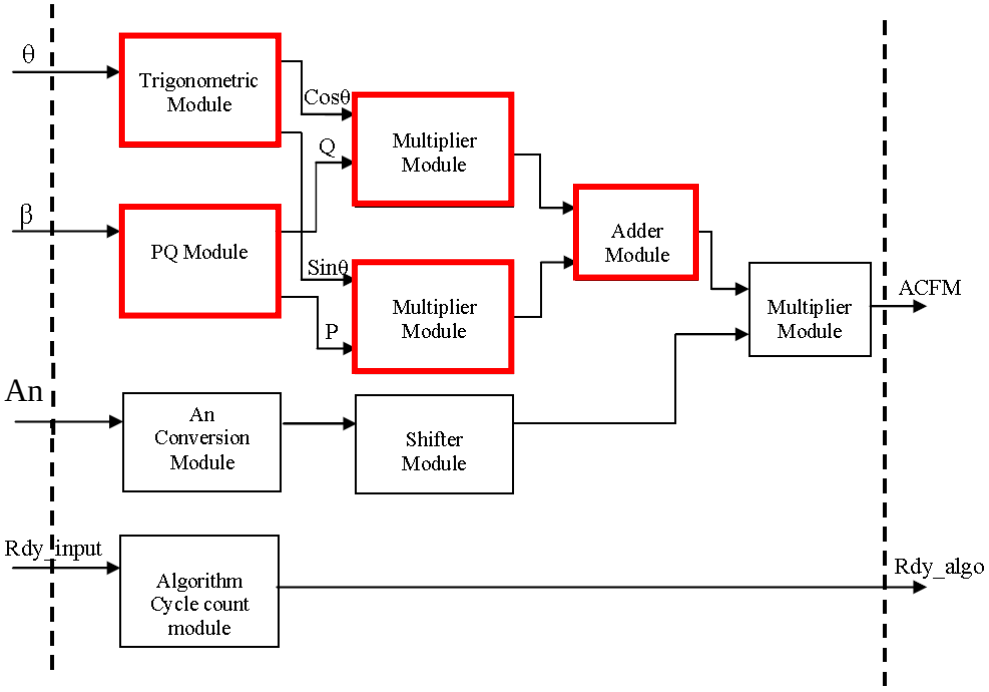


Figure 15 – Block diagram of algorithm module in the master path

From the diagram, it can be observed that the algorithm module consists of trigonometric module, PQ module, shifter, algorithm cycle count module, multiplier module and adder module. The master path only modules are highlighted by a thick red border. This module takes input of θ, β, An and rdy_input signal and produces a single ACFM value for each of these input combinations and also a control signal (rdy_algo). The control signal, rdy_algo is used to trigger the recursive adder module and indicate that the ACFM value is ready at the output port of the algorithm module.

The algorithm cycle count module is used to count the clock cycle and generate the control signal rdy_algo. This module outputs a high on rdy_algo signal when the ACFM value is ready.

The algorithm module in the master path calculation section consists of a fixed-point multiplier module, fixed-point adder module, shifter and a trigonometric calculation module. The fixed-point adder and multiplier are used because the implementation of these two modules is straightforward and faster if compared to a floating-point adder and multiplier.

Considering the structure of the master path is quite similar to the structure of the algorithm realisation part in the one-single path system, the other three modules in the top-level entity will not be discussed in this dissertation. All the of details could reference the paper of Ma Kwang Yew, a masters student who graduated from the University of Birmingham in 2011. (Ma, 2010-2011)

4.10 The low-level entity implementation in the slave paths

The slave paths' top-level module consists of three modules, which are the input data manipulation module, algorithm module and recursive adder module. As mentioned above, the algorithm module is the core, because most of the calculation is done within this module. Let us focus on the low-level structure of the algorithm module in the slave paths system.

4.10.1 Algorithm module in the slave paths processing chain

Figure 16 shows the block diagram of the algorithm module in the slave paths, where "An" represents a[n] in *Equation 5*.

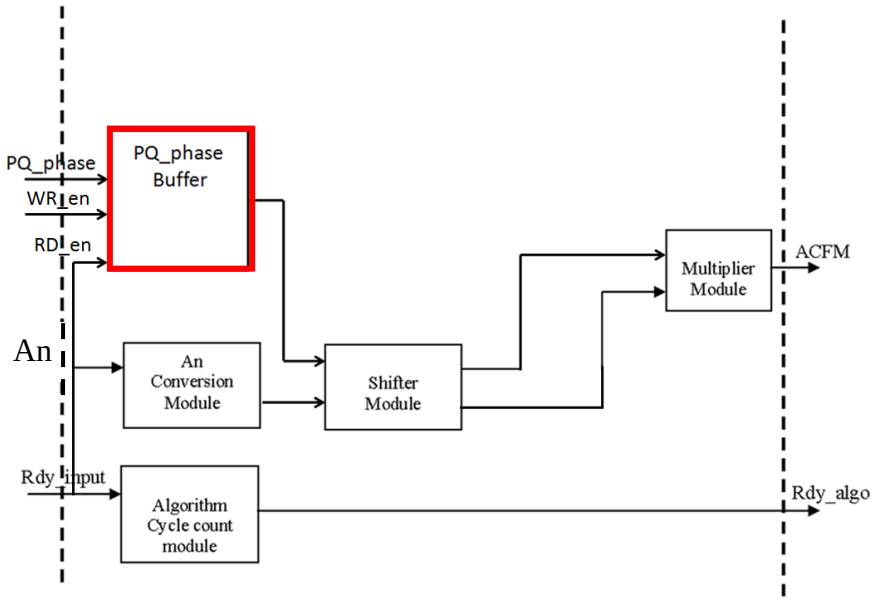


Figure 16 –Block diagram of algorithm module in the slave paths

In the algorithm module in the slave paths, only An Conversion Module, which is due to convert the format of An into sfixed format to do the fixed calculation, is kept. The algorithm cycle count and Shift Module, which are designed to solve synchronisation problems, are modified to fit into the slave paths processing chain. The Multiplier Module, which is due to multiply An and "PQ_phase", is kept to produce the ACFM value of each An in one frame ("clip").

The newly built module, which is highlighted by the thick red border, is a PQ_phase Buffer. It is a FIFO (First In First Out) here, functioning to store a series of PQ_phase calculated from the master path before the slave paths calculation starts. Because the data for the eight paths arrives in a serial way, the slave paths have to wait for the master path work to finish, and moreover, the seven slave paths internal is in ascending order.

In the following section, the newly built module, PQ_phase Buffer, and the modified modules, algorithm cycle count and Shift Module, will be described, beginning with the PQ_phase Buffer.

4.10.1.1 PQ_phase Buffer

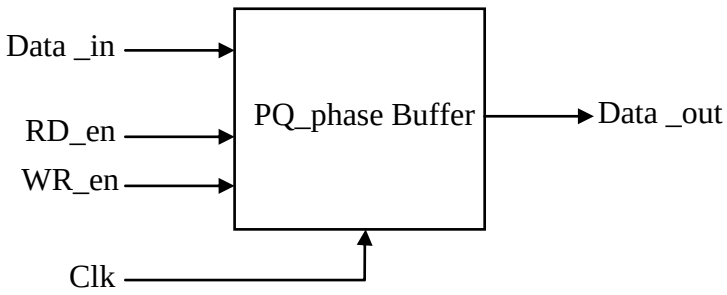


Figure 17 –Block diagram of PQ_phase Buffer

Figure 17 shows the block diagram of the PQ_phase Buffer in the slave paths. The PQ_phase Buffer is a FIFO (First In First Out), functioning to store a series of PQ_phase calculated from the master path and then output them in the right order for the slave paths' calculation.

The FIFO is generated by using a Xilinx IP Core generator. It is realised by using memory functional IP Core, generating a 128 depth FIFO (First In First Out).

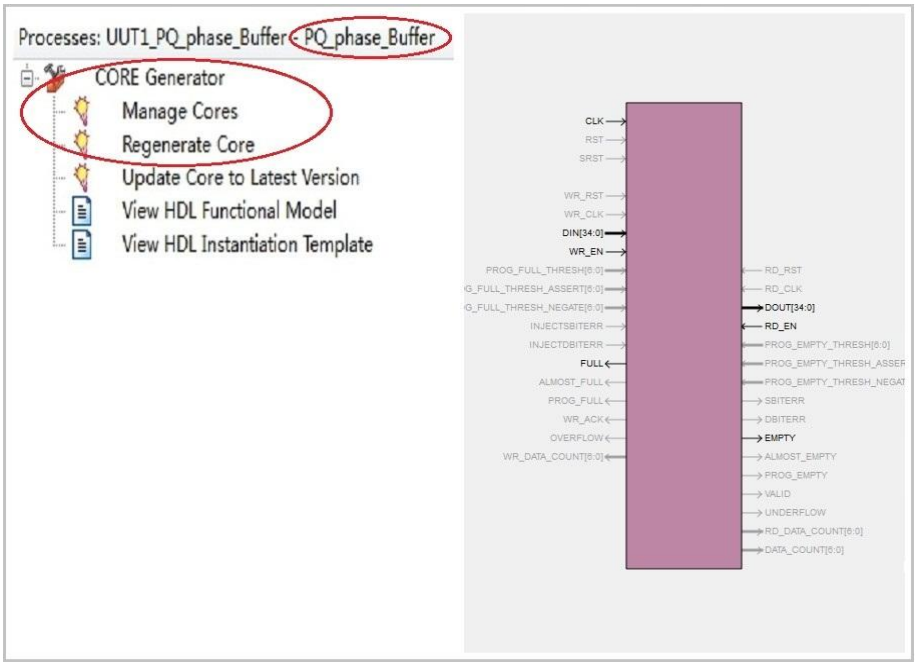


Figure 18 –IP Core generator user interface and the core black box

On the left side in *Figure 18*, it shows the user interface where to get access to the IP Core generator. The setting of the core needs to be modified frequently, and once the pins changes, the core needs to be regenerated. The most used options are therefore manage cores and regenerate core. On the right hand side is the core black box of the PQ_phase buffer. The internal structure cannot be seen, but all details are explained in the system provided user guide.

4.10.1.2 Algorithm cycle count module

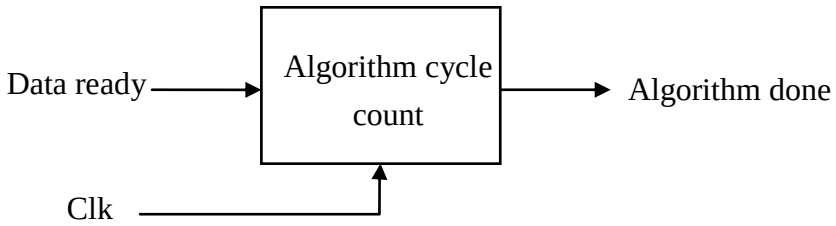


Figure 19 –Block diagram of algorithm cycle count module

Figure 19 shows the block diagram of the algorithm cycle count module in the slave paths. This module is to generate the synchronised algorithm finish signal and output it to the following recursive adder.

Shift registers are used in the module to implement the latency. *Figure 20* shows how the registers are chained up to implement this operation.

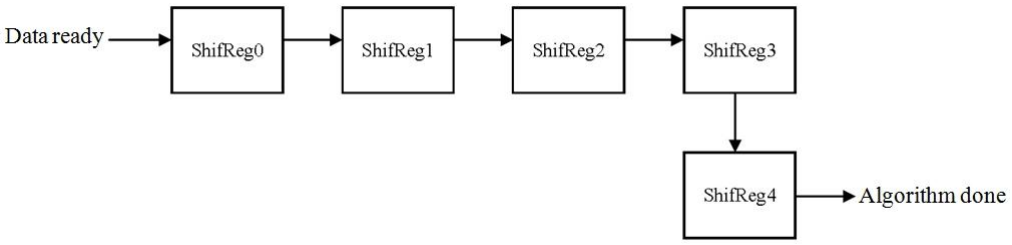


Figure 20 –Shifter latency chain in algorithm cycle count module

It can be observed that four registers are chained up to act as a shifter. When the sample An is available at the input of the algorithm module, a "data ready" signal is toggled high. This signal is used to propagate through four registers which correspond to the latency that is needed to perform the algorithm on one sample in the slave paths. When the output is ready, the control signal "algorithm done" is also ready and set high at the output port.

Having finished the description of the algorithm module in the slave paths, then the other two parts will be described in turn. The following sections begin with the input data manipulation module.

4.10.2 Input data manipulation module in the slave paths

The input data manipulation module is used to organise the data that it receives. It receives the input from the SPI module and assigns the received input to the total number of data in each frame “N” and various values sampled from the track stored in "An". Then, it transmits the data to the algorithm module to perform the signal processing.

Figure 21 shows the data format between the DSP and the FPGA device, where “θ” represents R[n-1], "An" represents a[n] in *Equation 5*.

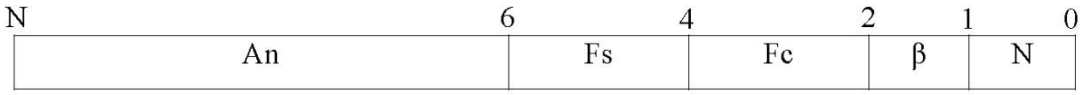


Figure 21 –SPI communication data format between DSP and FPGA device

Each element is a 16-bit data. The first data (element 0) represent the number of samples (N) in one “clip”. The second data (element 1) is the corresponding β value for that particular “clip”. The third and fourth data (element 2 and 3) represent the carrier frequency (Fc). The fifth and sixth (element 4 and 5) represent the sampling frequency (Fs). The lower 16 bits (LSB) sends first, followed by the upper 16 bits (MSB). Then following is the N number of the ADC value sampled from the track stored in "An".

The block diagram of this module is shown below in *Figure 22*.

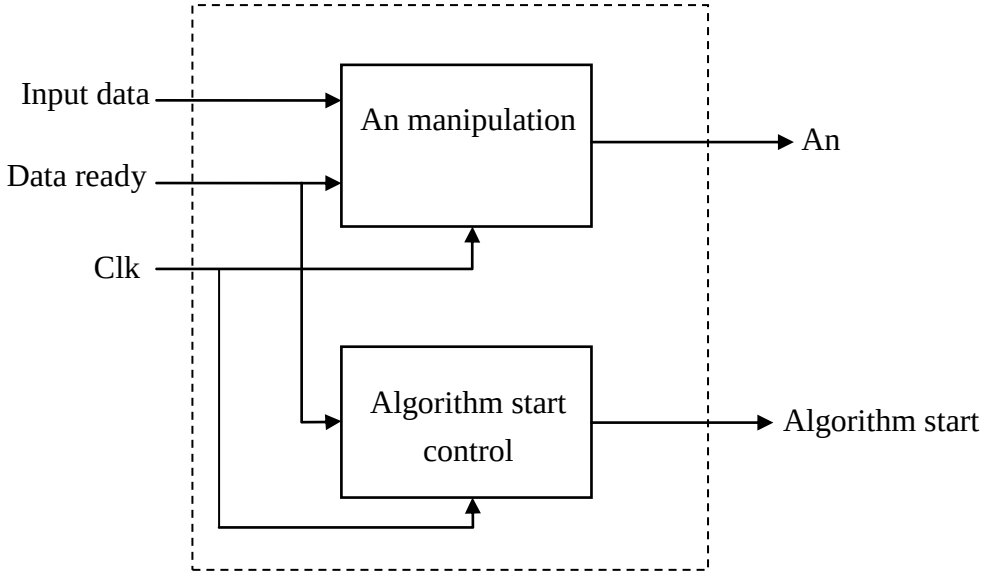


Figure 22 –Block diagram of Input data manipulation module

It can be seen from the diagram that the input data manipulation module in the slave paths can be further divided into two lower-level modules, An manipulation module and algorithm start control module, two of which are in charge of extracting An from input data and generating the algorithm start enable signal. These two modules will be discussed in the following sections.

4.10.2.1 An manipulation module

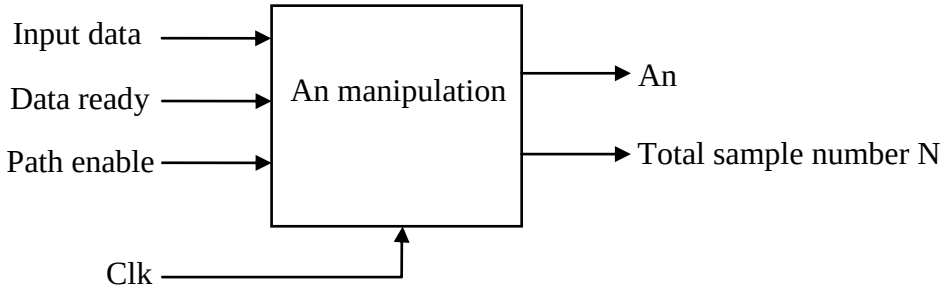


Figure 23 –Block diagram of An manipulation module

Figure 23 shows the block diagram of the An manipulation module in the slave paths. This module is used to extract An from input data, and also total sample number N, then send them to the following modules. A state machine, which is a mathematical model of

computation and normally used to design both computer programs and sequential logic circuits, is used to model the process of this module.

The state machine diagram is shown below in *Figure 24*.

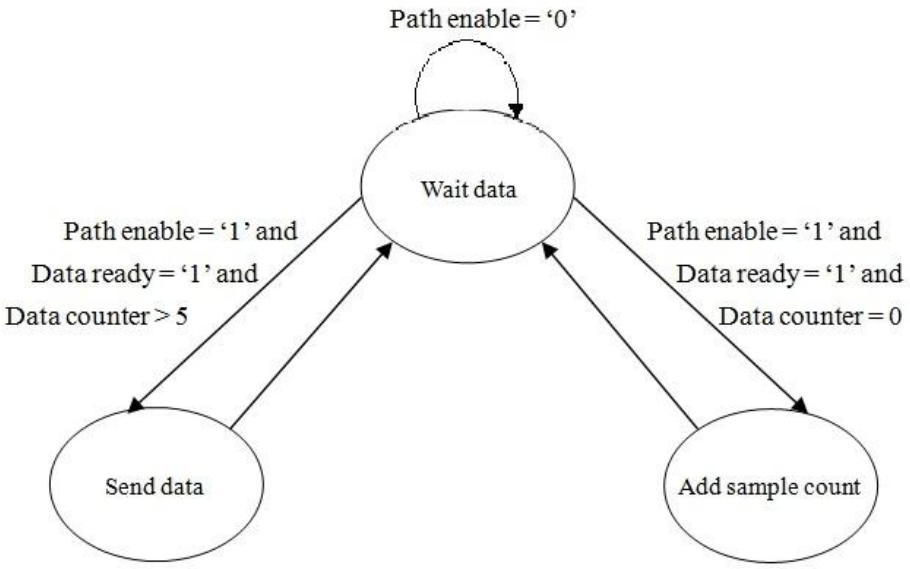


Figure 24 –An manipulation secondary module state machine diagram

It can be observed that the module starts at a "wait data" state. It waits until the "Path enable" signal changes to '1', which means that the path is selected by the SPI module. If the 16-bit data is also ready, the "Data ready" signal is set to '1'. In the meantime, if the data counter, which is the internal counter, equals zero, it means that the incoming data is N value (total number of data in one frame) and the state machine moves to the "add sample count" state. In this state, the N value is stored in the register and used for the judgement of whether all the data in one frame has been received or not. Having finished the work, the state then changes back to the "wait data" state.

In the "wait data" state, if a "Data ready" signal received is '1' and the data counter is greater than integer five, the state transits to a "send data" state. In this state, the extracted An is sent

to the following algorithm module and algorithm start enable signal is sent to the algorithm start control module to do the synchronization.

4.10.2.2 Algorithm start control module

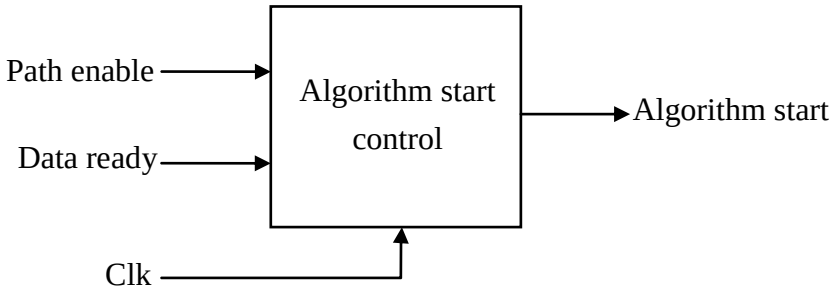


Figure 25 –Block diagram of algorithm start control module

Figure 25 shows the block diagram of the algorithm start control secondary module in the slave paths. This module is used generate the synchronised algorithm start enable signal and then send it to the following modules. A state machine is also used to model the process of this module.

The state machine diagram is shown below in *Figure 26.*

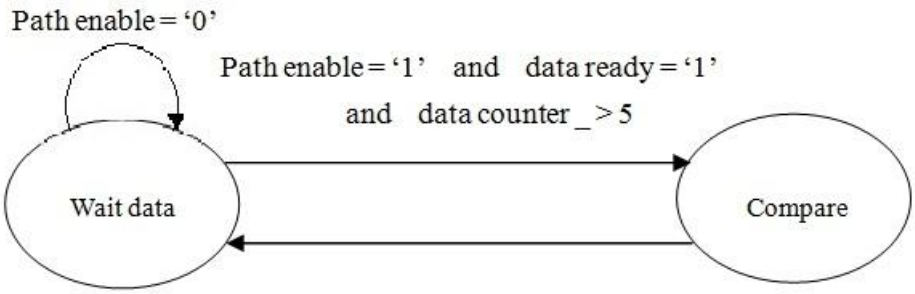


Figure 26 –Algorithm start control secondary module state machine diagram

It can be observed that the module starts at the "wait data" state. It waits until the "Path enable" signal changes to '1', which means the path is selected to be the SPI module. If the 16-bit data is ready as well, the "data ready" signal is set to '1'. In the meantime, if the data

counter, which is the internal counter, is greater than integer five, it means that the following incoming data are An, and the state machine moves to the "compare" state. In this state, the algorithm start enable signal is set to '1', which means enable. The data counter is compared with the total number of data in one frame "N". When "N" data have been received, the data counter is clear, the algorithm start enable signal is to set '0', and the state returns to the "wait data" state.

Having finished the description of the input data manipulation secondary module in the slave paths, the last part, recursive adder, will be described in the following module.

4.10.3 Recursive adder module in the slave paths processing chain

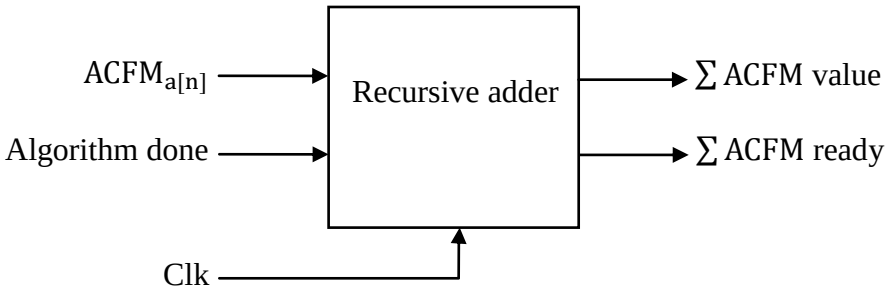


Figure 27 –Block diagram of recursive adder

The recursive adder is used to perform the addition on every single ACFM value that is produced by the algorithm module in each path. It waits to read the input and performs the calculation until the ready signal from the algorithm module (algorithm done) is received, which is due to make sure the input data is valid before the module start performs the calculation. After the adder is done with the addition for the total number of samples (N), it will then output the ∑ ACFM value and the ∑ ACFM ready signal. A state machine is used to model the behaviour of this module.

Figure 28 shows the state machine.

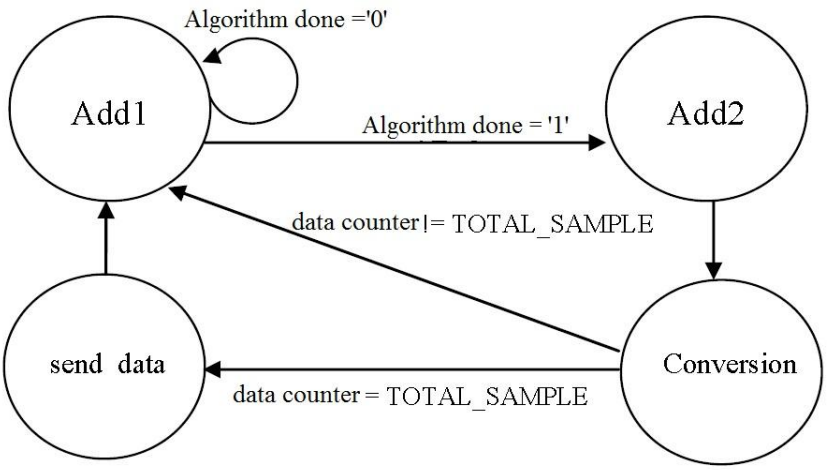


Figure 28 –Recursive adder module state machine diagram

The module starts at the "add1" state and it stays at this state until the input signal "Algorithm done" is set to '1'. It performs the addition once when the input "Algorithm done" is high and the state transits to the "add2" state. In the "add2" state, it moves the current value to the internal variable so that it can be used in the following calculation. Then, the state moves to the "conversion" state. In this state, the data counter is compared with the total sample number (N). If the data counter does equal the total sample number (N), then it will transit to the "send data" state. Otherwise it will transit back to the "add1" state and wait for the next "Algorithm done" signal to go high. In the "send data" state, the "$\sum$ ACFM ready" signal is toggled high to indicate that the summation of the series ACFM value is ready at the output port of the recursive adder.

Figure 29 shows the flow chart of this module.

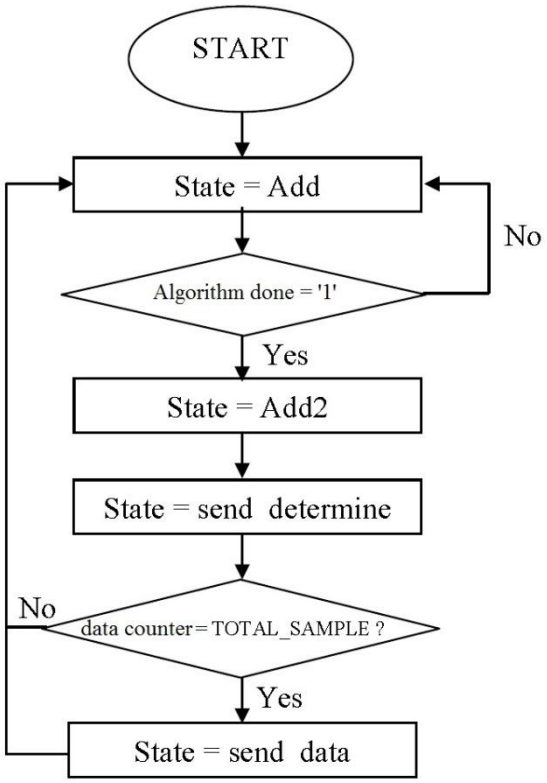


Figure 29 –Flow chart of recursive adder module

This module will sum all the ACFM output. When the data counter equals the total sample number (N), the ACFM result will be sent to the next module.

Having finished describing the last part in the slave paths data processing chain, the structure and the working principle of the eight paths data processing chain should be clear at this stage. However, not only these modules are included in the eight paths system. As demonstrated in the system diagram of the eight paths in *Figure 14*, the data processing chain in the top-level system is divided into four parts, which are SPI slave, algorithm realisation in the eight paths, synchronisation and two SCI modules.

In the sections above the algorithm realisation part in the eight paths has been discussed. The other three parts will be discussed in turn. The following sections begin with the input synchronisation part.

4.11 Synchronisation part

Synchronisation is always a difficult problem, but it is the most important part in the system design, especially in the fast signal parallel processing system. The synchronisation work in this system is generally divided into two parts. One is the synchronisation between the eight paths data flow. This work is sorted by the synchronise buffer. The other part is the synchronisation within the data flow of each 16-bit element in one "clip". This work is more complex because pipeline design concepts are introduced into this system. The work involves many synchronisation modules in the eight paths processing chain. In the following sections, these two synchronisation works will be discussed. Firstly, the synchronise buffer will be introduced.

4.11.1 Synchronise buffer

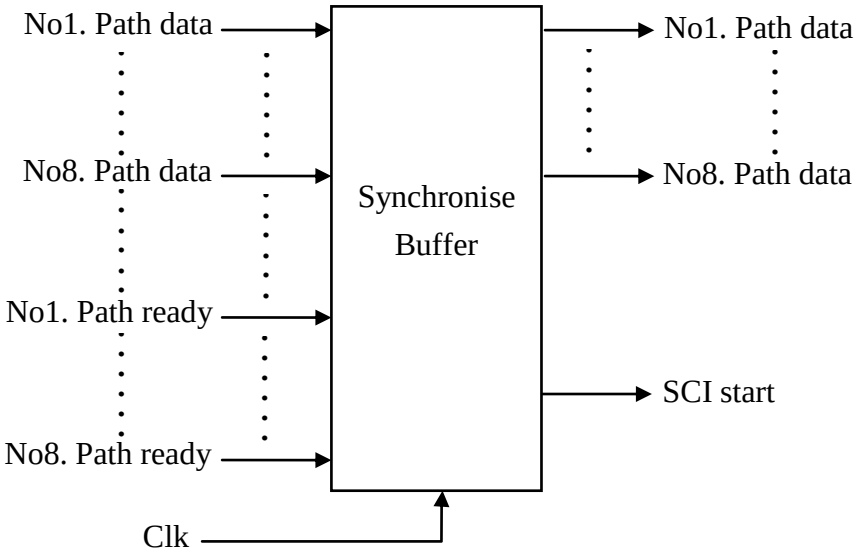


Figure 30 –Block diagram of synchronise buffer

Figure 30 shows the block diagram of the synchronize buffer. The eight paths data just pass the module, but the SCI starting signal is changed with the eight paths ready signals.

The synchronize buffer works to buffer the ACFM results for all the eight paths and guarantees the results can arrive at the SCI module at the same time. The internal registers toggle the algorithm finish signal of each path. It waits until the calculation work for all the eight paths is finished, then generates a SCI start working signal and sends the result of the eight paths to the following two SCIs. Once the disturbing impulse occurs at one of the path ready inputs, the module still needs to wait for the other seven paths' ready signals. This method could effectively prevent noise disturbance hazards. The design details will be discussed in the simulation section.

Having set out the working principle of the synchronise buffer, then the other part of the synchronisation work should be introduced. In the following section, the synchronisation within the data flow in one frame (“clip”) will be discussed.

4.11.2 Timing synchronisation design

Pipeline design concepts are introduced and implemented in the ACFM algorithm design. This provides a high quality performance (in terms of speed) for the design. Latency and throughput are the two most fundamental measures of the efficiency performance. They are closely related, but whereas latency measures the amount of time between the start of an action and its completion, throughput is the total number of such actions that occur in a given amount of time. The pipelining could allow the system to receive and process the data every clock cycle in order to achieve less latency and more efficient throughput.

Based on this idea of design, the timing synchronisation is an important issue to guarantee the fast pipeline data process will not meet data hazards, in which case the confliction would occur among the intermediate variables resulting in unexpected results. Synchronisation control modules are introduced and implemented into the system:

1) algorithm_start_control_Primary: This module is built in the Input Data Manipulation Primary module. The input data manipulation process takes seven clock cycles to finish, so when the module receives input data, the algorithm start signal is delayed for seven cycles and then generated.

2) npi_control_module: This module is built in $\frac{1}{N\pi}$ calculation module. Its work is to synchronise the calculation process internal. It generates three enable signals CE0, CE1, CE2 for fixed_to_float, floating_Div, and floating_to_fixed module separately. Firstly, the received N, total number of data per frame, should be converted to sfixed format to multiply with π which is also represented in sfixed format. However, the Division Ip Core is represented in floating format, so CE0 signal is sent to fixed_to_float module when the sfixed format converting to floating format process could begin. Then, the floating_Div module receives CE1, starting division. Finally, the floating_to_fixed module receives a CE2 signal to convert the result $\frac{1}{N\pi}$ back to sfixed format and transmits it to the following algorithm module.

3) Shifter_Primary: This module is built in primary algorithm module. The aim of this module is to insert delay for An, because it has to wait for the Sine function and Cosine function which are in the calculation of P, Q parameters. The trigonometric function related process takes twenty-four clock cycles to finish, so this module inserts twenty-four cycles of delay for a[n] in order synchronise it with P, Q combination phase, $P\sin(R[n-1]) + Q\cos(R[n-1])$ in the formula.

4) algorithm_cycle_count_Primary: This module is built in primary algorithm module, and its work is to synchronise the algorithm finish signal, which is sent to the recursive adder module to tell it start working, with the internal algorithm calculation process. From simulation, it can be seen that the algorithm process takes twenty-nine clock cycles to

finish. The signal will therefore be generated in twenty-nine cycles after it receives the input data.

5) algorithm_start_control_Secondary: This module is built in Input Data Manipulation Secondary module. The time consumption is the same as in the master path, because the work load in data manipulation is the same. The algorithm start signal is therefore generated in seven cycles after it receives the input data.
6) Shifter_Secondary: This module is built in secondary algorithm module. The aim of this module is to insert delay for the PQ_phase, because it arrives one clock cycle earlier than An. Moreover, there is no trigonometric function related process, so this module inserts one cycle delay for the PQ_phase.
7) algorithm_cycle_count_Secondary: This module is built in secondary algorithm module, and it does the same work as the module in the master path. However, from simulation, the algorithm process in the slave paths takes four cycles to finish, so the signal could be generated in four cycles after it receives the input data.
8) Sync_Buffer: This module is built in the top-level of the system. The front work of the eight paths is not finished at the same time, but in ascending order. The aim of Sync_Buffer module is to buffer the ACFM results for all eight paths and guarantee the results can arrive at the SCI module at the same time. This module is important and it also helps to prevents the noise disturbance hazards.

These modules are built in the eight paths data processing chain, which is in the algorithm part of the system. With them, high speed processing is guaranteed. In another way, the synchronisation control signal modules are independent of the calculation modules rather than integrated in them. The reason is still to achieve a high speed processing chain.

The performance is slowed down if a counter is used to count the number of latency for each sample stored in An. This is because the counter needs to hold the count value for each

sample. The counter could not hold the count for the next arriving input if the previous sample is still under processing.

In order to solve this problem, the parallel processing method is introduced in this system and demonstrated above. The parallel processing method is implemented by propagating the control signal stage by stage. This is because the control signal is independent of the input signal. The input that connects to the first shift register will only toggle high when the "Data ready" signal is high and this particular control signal is propagated along with the data while it is processed in another module. When the control signal shifts out from the last shift register, the expected output data is ready at the output port. While a sample is being processed within the module, another sample can be taken in at any time because the control signal does not hold by the previous sample. Taking the algorithm module in the master path as an example, by using this method, the latency cycles that are needed to process one sample are decreased from 30 cycles to 5 cycles, which saves about 70% of the processing time.

Having discussed the synchronisation part in the eight paths system, the two communication parts will be considered, which are the SPI slave module, and two SCI modules.

The following sections begin with the SPI slave module.

4.12 Serial Peripheral Interface (SPI) slave module

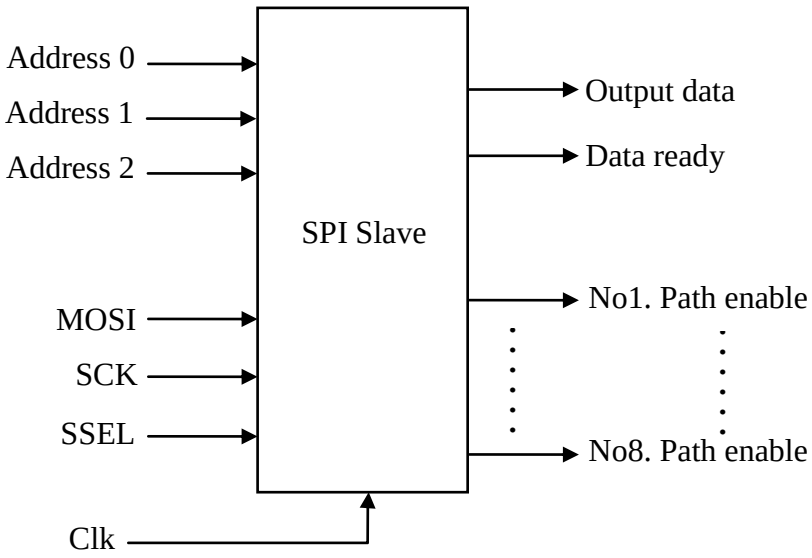


Figure 31 –Block diagram of SPI

The SPI module acts as a slave mode and is used to receive data from the DSP processor. It consists of seven input signals and ten output signals. The inputs are eight-path data stream (MOSI), SPI communication clock (SCK), chip select signal (SSEL), system clock (Clk) and three path address lines (Address2, Address1, Address0). The outputs are control signal (Data ready), 16-bit output data (output data) and eight paths enable signals (from No1. Path enable signal to No8. Path enable signal).

Normally, the SPI bus is operated much slower than the FPGA operating clock. Hence, the SPI bus is over-sampled by using the FPGA clock. It is always good practice to have SPI logic run in the FPGA clock domain. In this module, the SPI signals (SCK, CS, and MOSI) are synchronised by using the FPGA clock and shift registers.

Figure 32 shows the flow chart that is used to design the receipt of the data from the SPI master (DSP).

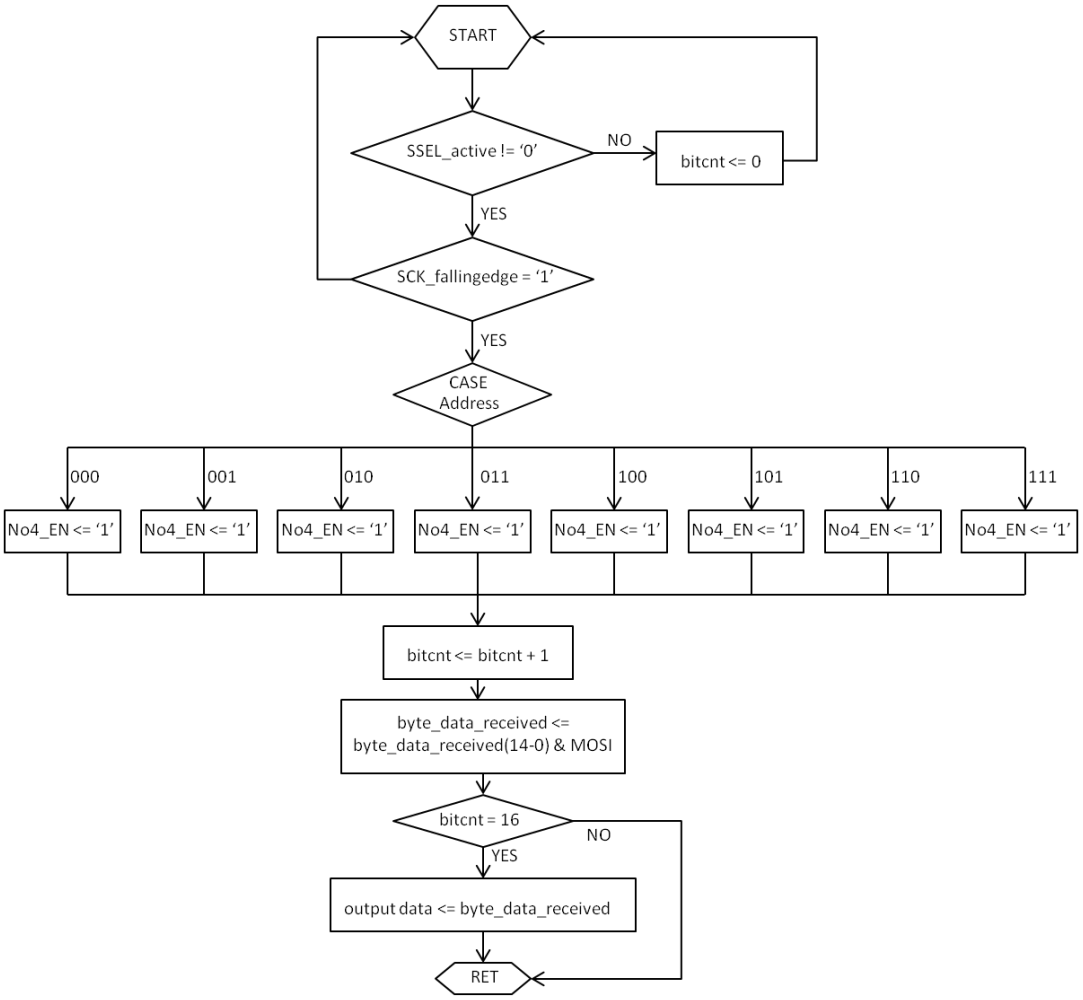


Figure 32 –Design flow chart of SPI Slave module

The module waits until the chip select signal (SSEL_active) goes active then starts to work. It starts to read data from the MOSI port at the falling edge of the SCK. A counter "bitcnt" is used to count the number of bit that are already read from the MOSI port. This module is design to read 16 bit data as a data packet from the MOSI port. When 16-bit data have been read from the MOSI port, the received 16-bit data is sent to the next module.

4.13 Serial Communication Interface (SCI)

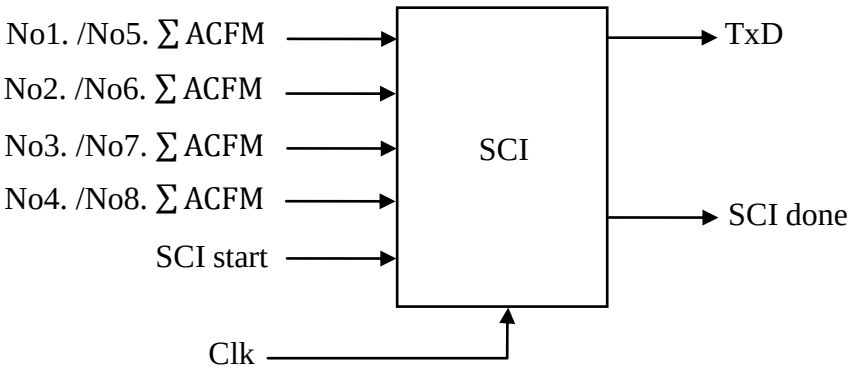


Figure 33 –Block diagram of SCI

Figure 33 shows the block diagram of one of the two SCI modules.

As discussed above, two SCIs are used in this project to ensure communication accuracy. They take responsibility for each four paths respectively. The SCI module is used to send the final ACFM value of the four paths from the FPGA device to the PC for further offline visual display. It consists of 6 input ports, system clock (Clk), four ∑ACFM value (16-bit data) from the four paths and a control signal (SCI start). Two outputs are TxD (serial bit stream) and SCI transmission finish signal (SCI done). The module starts to read data from the four input data ports when the "SCI start" signal equals to '1' which indicates starting the communication.

In this module, the baud rate that is used in the SCI communication is done by using the FPGA high-speed clock and dividing it down to generate a "tick" to generate the baud rate that is chosen in the SCI communication. In this application, it is assumed that the serial interface can tolerate a few percentages of error in the baud frequency generator.

5 Simulation Results (Testing and Verification)

Based on the modularity specialty of VHDL design in FPGA, simulation could be executed in each module. Sufficient simulation could effectively avoid system failure when the system

is put into industrial use. By changing variables in the simulation, predictions may be made about the behaviour of the system. Using the simulator could virtually investigate the behaviour of the system under study. As mentioned above, the simulator ModelSim is used to execute the simulation. Before the simulation begins, the required simulation libraries need to be compiled. Having set up the stable simulation environment, the simulation work can begin. Considering the input data from the DSP, a simulated data generator is needed to imitate the behaviour of the DSP.

The ACFM algorithm is also realised in Matlab. Having put the simulated input data into Matlab, the calculated results can be used as criterion in the verification.

In the next section, the simulated data generator will be introduced.

5.1 Simulated data generator

The work implemented on the FPGA is the core part for the ACFM system realisation, but not the only part. So before the whole system is built up, FPGA needs to verify itself. The simulated data generator is necessary. It works to imitate the output data from the front part, the DSP interface. The block diagram of it is shown below in *Figure 34***.**

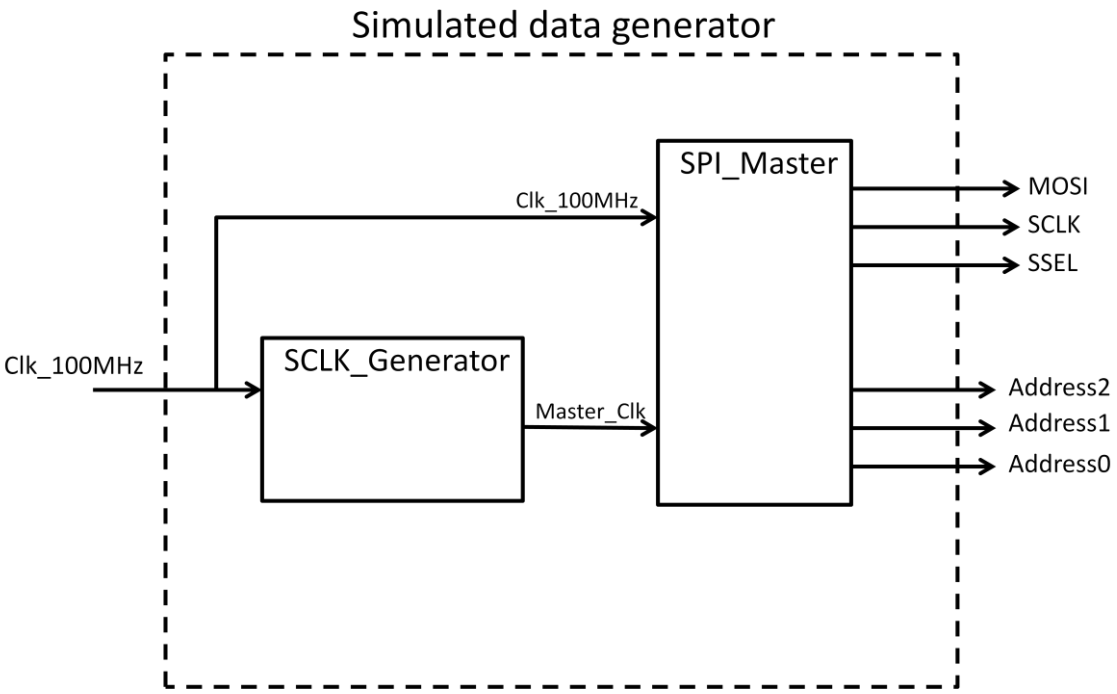


Figure 34 –Block diagram of simulated data generator

As described above, the inputs of the SPI Slave module in the top-level of the FPGA system are all provided in the diagram, the eight-path data stream (MOSI) and communication clock (SCLK), chip select signal (SSEL), and three path address signals (Address2, Address1, Address0) . With the data generator, the simulation could work exactly as the real situation.

Having built up the simulated data generator, the simulation input data has sorted out. The simulation work is generally divided into components simulation and system simulation. For the components simulation, the low-level modules which are discussed in the methodology section will be simulated. The simulation process that is involved in each model is from low-level to top-level entity. Let us begin with the components simulation.

5.2 Components simulation

In the following sections, the master path simulation and slave paths simulation will be executed separately, beginning with the master path simulation.

5.2.1 The master path simulation

Having implanted the top-level design, Xilinx ISE can generate the RTL schematic as the designer expects. The RTL schematic of the master path top-level entity is shown in *Figure 35* below.

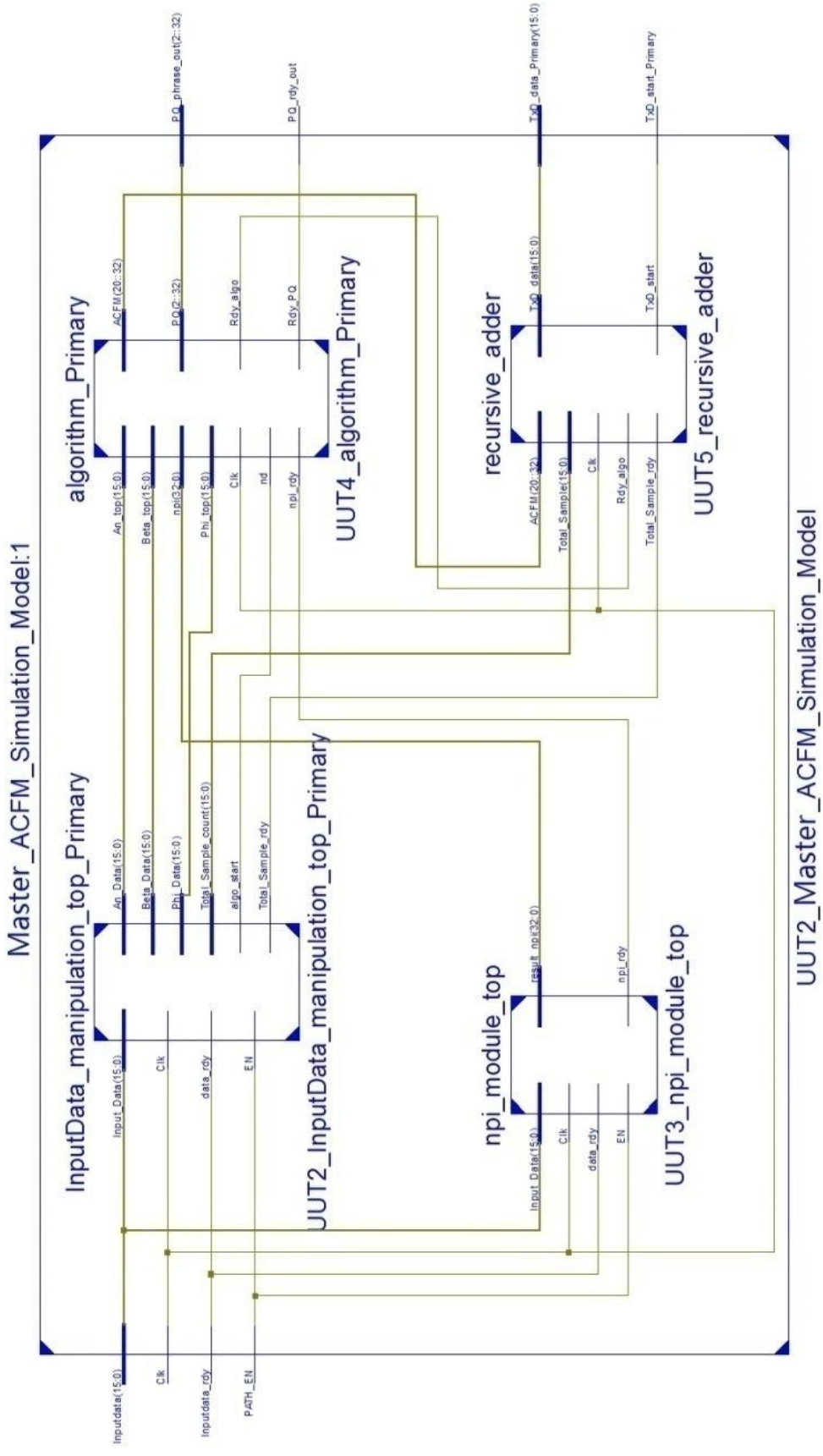


Figure 35 –Master path Simulation Module RTL Schematic

From the diagram above, it can be observed that the top-level entity consists of input data manipulation primary module, $\frac{1}{N\pi}$ calculation module, algorithm module, and recursive adder module.

As discussed in the methodology section, the master path consists of a combination of all the modules in the one-single path system. This simulation is intended to verify the behaviour of the top-level entity rather than low level entities. A simulated data generator is used to mimic the data transmitted from the DSP and simulate the behaviour of the DSP. The connection between is shown in *Figure 36* below.

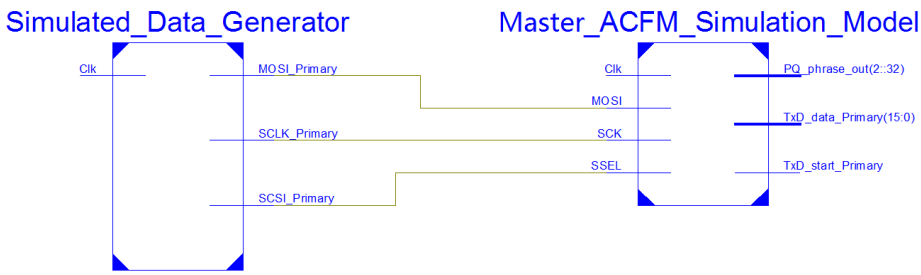


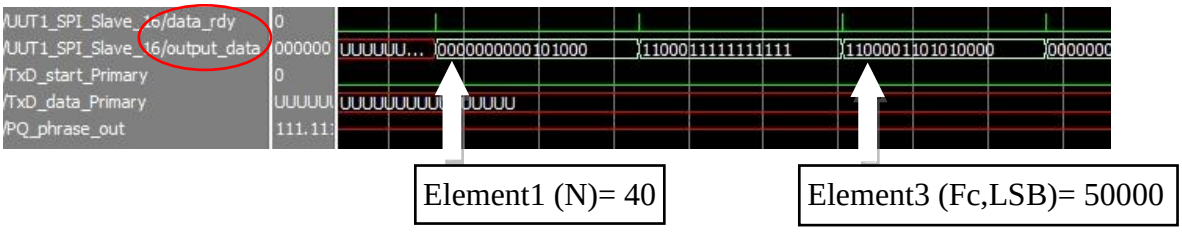
Figure 36 –RTL Schematic of the master path simulation structure

Table 1 shows the input combination that is used in the simulation. These example data are provided by TSC Company, which are collected using ACFM probe of the designed settings.

From the table it can be seen that two sets of data are used for simulation. Each of them is one frame of converted digital ACFM data and has 46 elements. The frame consists of 6 header data (n_h = 6) and 40 samples (N = 40).

The simulation waveform is shown below in *Figure 37* and *Figure 38*.

Frame1



Frame2

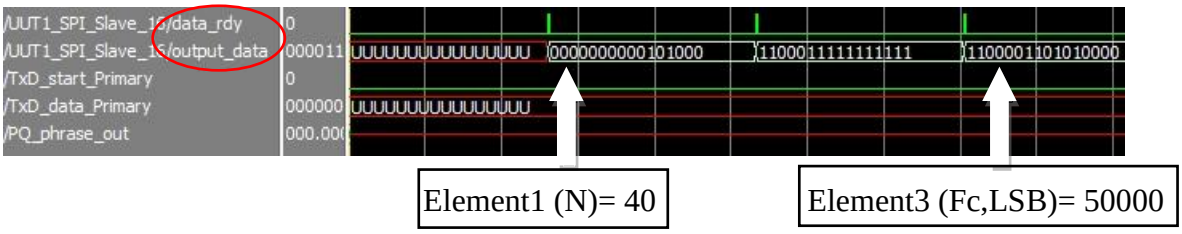
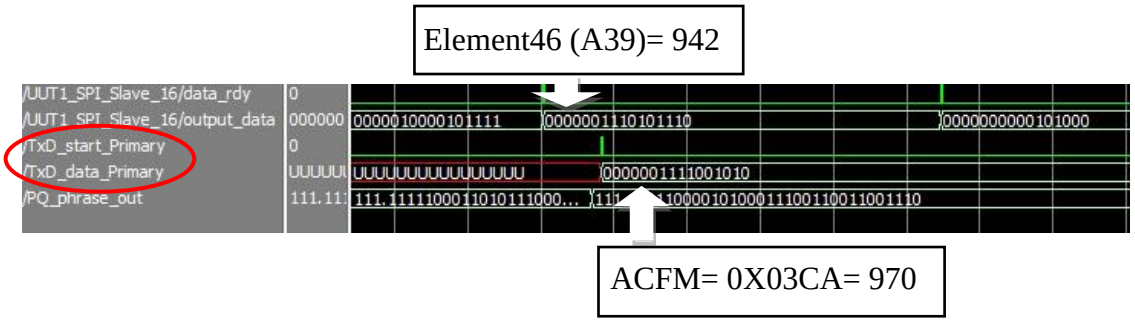


Figure 37 –Waveform of the input data in the master path simulation

Frame1



Frame2

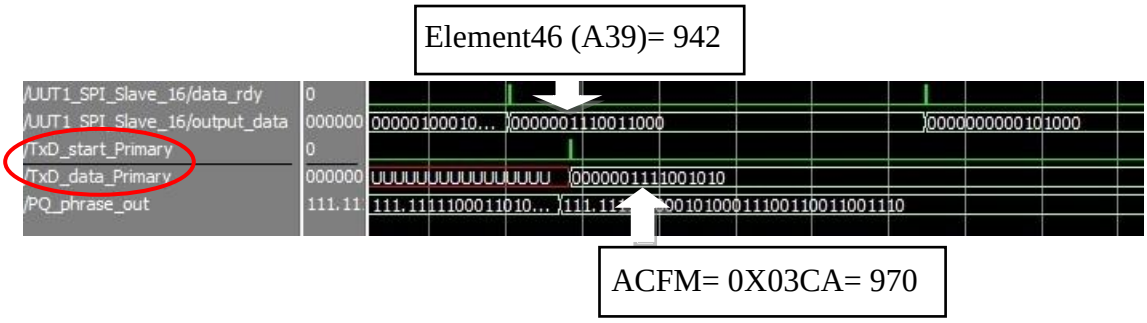


Figure 38 –Waveform of the output result in the master path simulation

In *Figure 37*, the first three elements in Frame 1 and Frame 2 are shown in the signal "output_data" column. It can be seen that in Frame 1, the first element (N) is 40 and the third element (Fc, LSB) is 50000, exactly the same as the value in the table. Frame 2 is the same.

In *Figure 38*, the output result waveforms of Frame 1 and Frame 2 are shown in the "TxD_data_Primary" column. The simulated ACFM result of Frame 1 is 970 and 970 for Frame 2.

5.2.2 Matlab verification and TSC expected result

In order to verify if the simulated ACFM result is correct, the results provided by TSC company and MATLAB are used as criteria.

The ACFM algorithm is described in *Equation 5*. The important part of the codes in Matlab is shown below in *Figure 39*.

```
%%%%%%%%%%%%%%%%%%%% Frame 1 %%%%%%%%%%%%%%%%%%%
% an=[834,770,736,758,788,888,998,1162,1361,1540,1765,1988,2234,2495,2692,2939,3163,3343,3522,3612,3745,3777,3832,3824,3750,369
%%%%%%%%%%%%%%%%%%%% Frame 2 %%%%%%%%%%%%%%%%%%%
% an=[832,763,714,761,754,867,974,1129,1325,1510,1726,1969,2167,2457,2659,2935,3132,3294,3492,3598,3741,3794,3801,3864,3760,370
%%%%%%%%%%%%%%%%%%%% Frame 3 %%%%%%%%%%%%%%%%%%%
% an=[841,766,725,738,807,891,998,1141,1333,1544,1773,1995,2213,2462,2726,2950,3141,3317,3494,3638,3739,3782,3816,3818,3761,369
%%%%%%%%%%%%%%%%%%%% Frame 4 %%%%%%%%%%%%%%%%%%%
an=[857,773,769,783,771,893,986,1162,1339,1478,1746,1952,2212,2473,2640,2872,3119,3277,3501,3592,3687,3760,3807,3829,3794,3680,

Fs=2000000;
Fc=50000;

a=pi*260/180;
N=40;

P=4*cos(a)/(pi*N);
Q=4*sin(a)/(pi*N);
R=2*pi*Fc/Fs;

A=0;
PQ=0;
phase=0;
for i=1:N
    ACFMi=an(i)*(P*sin(R*(i-1))+Q*cos(R*(i-1)));
    A=A+(an(i)*(P*sin(R*(i-1))+Q*cos(R*(i-1))));
end
```

Figure 39 –ACFM algorithm realisation in Matlab

As shown in the figure, the parameters are set to the value listed in the desired system design section (Section 4.3). The ACFM results are stored in the array "A".

The table below is the summary of the ACFM results of the two frames in the three different methods.

Table 2 – The ACFM results in the three methods

	Frame 1	**Frame 2**
ACFM result in FPGA simulation	970	970
ACFM result from TSC	968	968
ACFM result from Matlab	968.04	968.45

From the figures in the table, it can be observed that there is a slight difference between the ACFM result in the FPGA simulation and the criteria. This is due to the number of bit that are used to represent the data operation and also the limitation of the trigonometric module that only accepts data input with 16-bit representation. The input radian value could not be represented more accurately than the fractions in 16-bit input format. Hence, this causes some variation in the final solution. However, the variation of the cracking trough appearing in the curve is the focusing point in this design, and the accuracy is still high when it is compared with the criteria (result from TSC and Matlab simulation). Therefore it can be concluded that the behaviour of the master path top-level entity is performed accurately.

5.2.3 The slave paths simulation

The RTL schematic of the slave path, No.2 path, top-level entity is shown in *Figure 40* below.

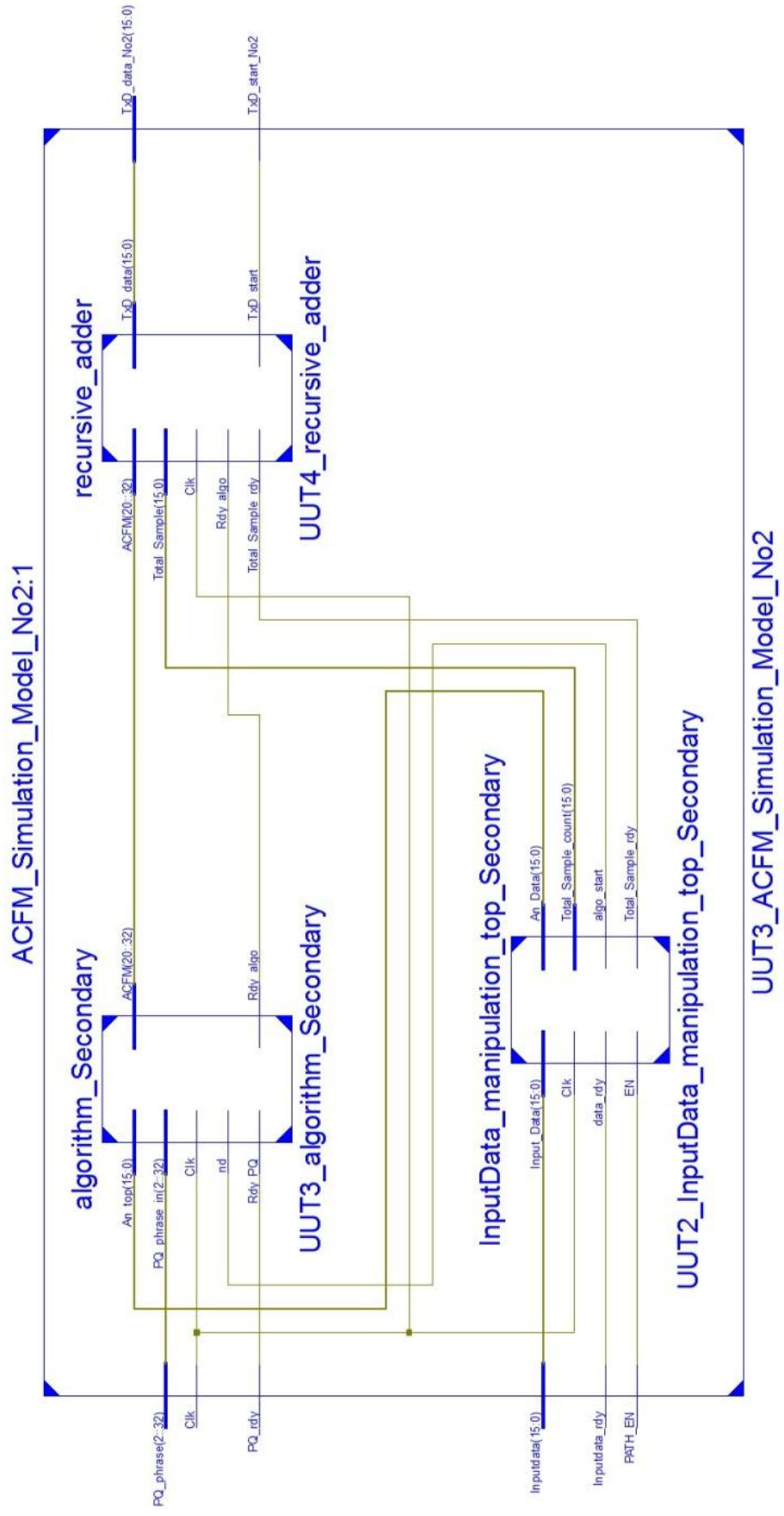


Figure 40 –The slave path Simulation Module RTL Schematic

From the diagram above, it can be observed that the slave path top-level module consists of three low-level modules, which are input data manipulation secondary module, algorithm module, and recursive adder module. The function of each module is the same as the master path. The lower-level structure is simpler and time consumption is shorter, because the design thought of the slave paths is to simplify the complexity in order to save the space in the FPGA and make the processing more efficient and productive.

The following simulation is used to simulate the three low-level modules. The simulation process that is involved in each model is from low-level to top-level entity. Let us start with the simulation of input data manipulation secondary module.

5.2.3.1 Input Data Manipulation Secondary Module

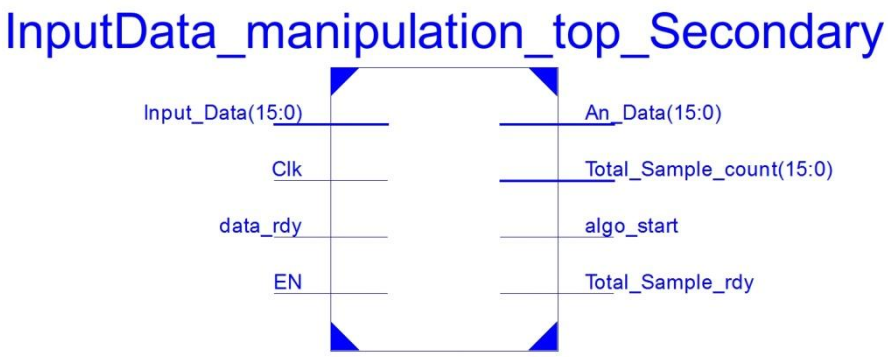


Figure 41 –Input Data Manipulation Secondary Module RTL Schematic

Figure 41 shows the RTL schematic of the input data manipulation secondary module in the slave paths. The working principle of this module has been discussed in the methodology section. The top-level entity of Input Data Manipulation Secondary Module consists of two main modules, which are An manipulation module, and Algorithm start control module. The following simulation starts with An manipulation module.

5.2.3.1.1 An manipulation module

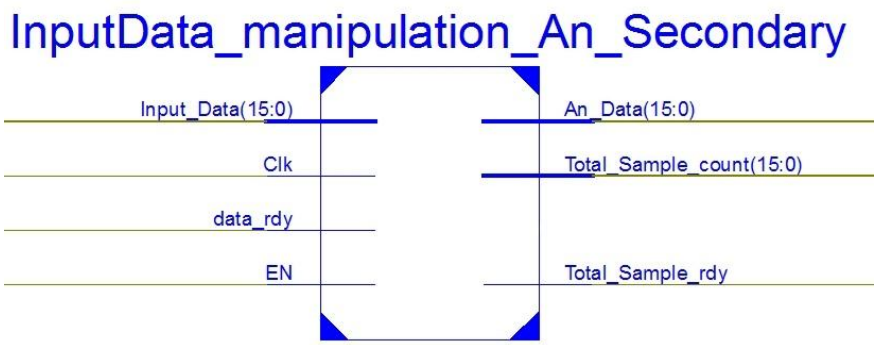


Figure 42 –An manipulation secondary module RTL Schematic

Figure 42 shows the RTL schematic of the input data manipulation secondary module in the slave paths. The working principle of this module has been discussed in the methodology section.

This simulation is to verify the behaviour of the An manipulation module by observing the input and output signals. The simulated input data is shown below in Table **3.**

Table 3 – Simulated input data

Header	**Element0**	**N**	40
	Element1	β	260
	Element2	**Fc, LSB**	50000
	Element3	**Fc,MSB**	0
	Element4	**Fs,LSB**	33950
	Element5	**Fs,MSB**	30
The first three samples	**Element6**	**A0**	832
	Element7	**A1**	763
	Element8	**A2**	714

The simulated input data is the part of one frame, but contains the 6 byte header and three samples, which are A0, A1 and A2.

The simulation waveform is shown below in *Figure 43* and *Figure 44*.

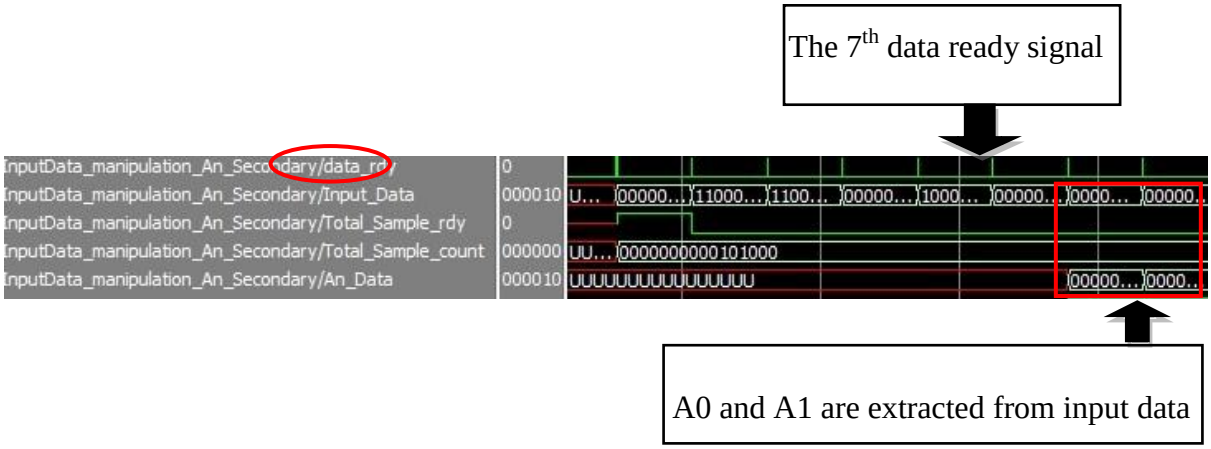


Figure 43 –Waveform of the input data in An manipulation module

From *Figure 43* it can be observed that the output "An_Data" waits until the 7th data ready arrives then extracts the input data and sends it to the followings. Referring to the data format in this project, the header length is six, and the forty samples start from the 7th element. So the behaviour of this module performs well. The data extracted in "An_Data" output signal is shown in *Figure 44*.

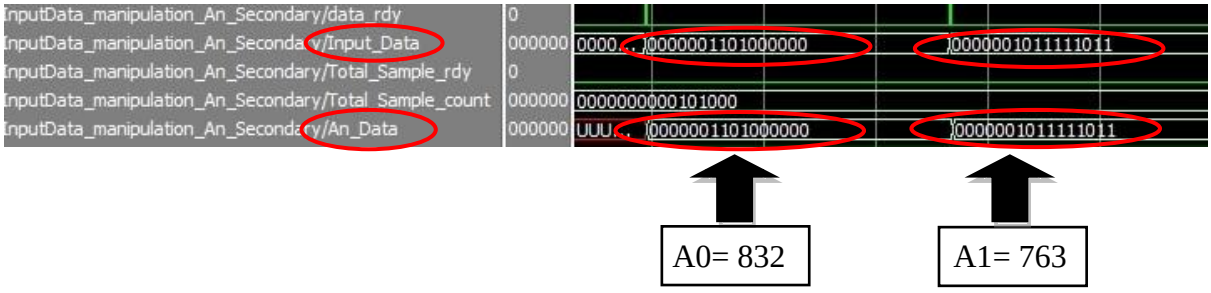


Figure 44 –Waveform of output A0 and A1in An manipulation module

From the waveform it can be seen that the first value of "An_Data" is 832, the same value as A0 in Table 3. The second value is 763, the same as the value of A1 in Table 3, therefore it could be concluded that this module could correctly extract the samples (An) from the input data.

5.2.3.1.2 Algorithm start control module

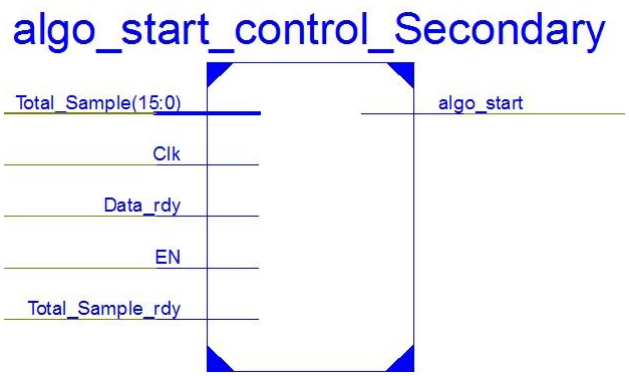


Figure 45 –Algorithm start control secondary module RTL Schematic

Figure 45 shows the RTL schematic of the algorithm start control module in the slave paths. The working principle of this module has been discussed in the methodology section.

This simulation is to verify the behaviour of the algorithm start control module by observing the input and output signals.

The Frame 1 listed in Table 1 is chosen to be the simulated input data in the simulation. The simulation waveform is shown below in *Figure 46*.

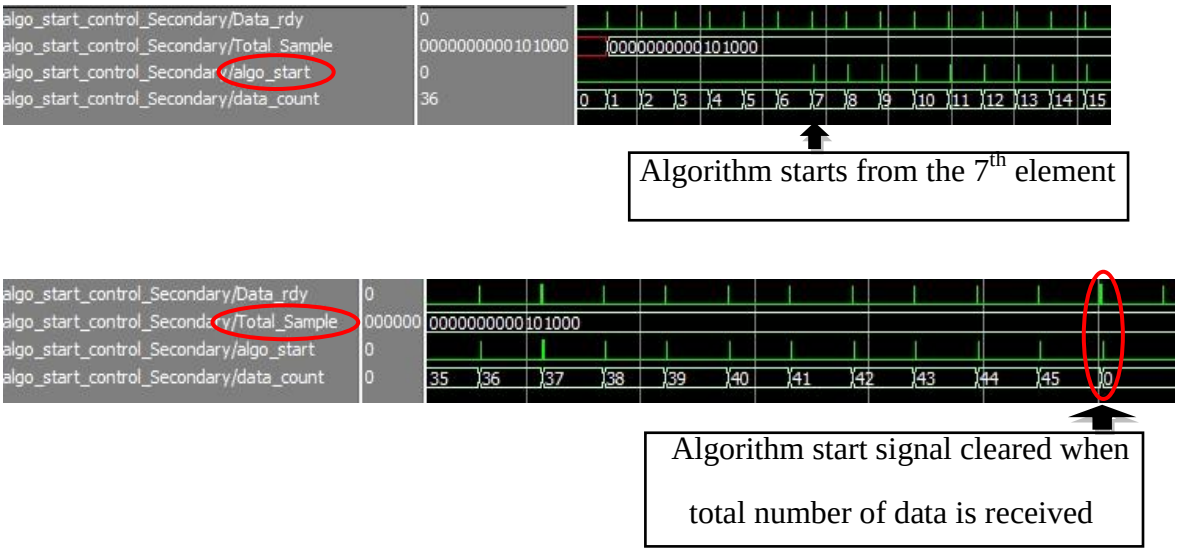


Figure 46 –Waveform of algorithm start signal in algorithm start control module

From the graph, it can be seen that the "algo_start" signal starts to change from the 7th element, and it changes to active when each sample arrives. When the total number of data has been received, the "algo_start" signal stays inactive. It can therefore be concluded that the behaviour of the algorithm start control module is correct.

5.2.3.1.3 Input Data Manipulation Secondary Module (Top-level)

Previous simulations are done on each module separately. The top-level entity, the Input Data Manipulation Secondary Module, performs the signal processing involved in the combination of all these modules. As mentioned above, the simulation is executed from the low-level entity to the top-level entity. Doing the simulation in this way can ensure that the modules involved still perform the correct behaviour after they combine together.

This module is used to organise the data that it receives. It receives the input from the SPI module and assigns the received input to the total number of data in each frame "N" and various values sampled from the track "An".

This simulation is to verify the behaviour of the Input Data Manipulation Secondary Module by observing the input and output signals.

The Frame 2 listed in Table 1 is used as the simulated input data in the simulation. The waveform is shown in the graph below in *Figure 47, Figure 48, Figure 49*.

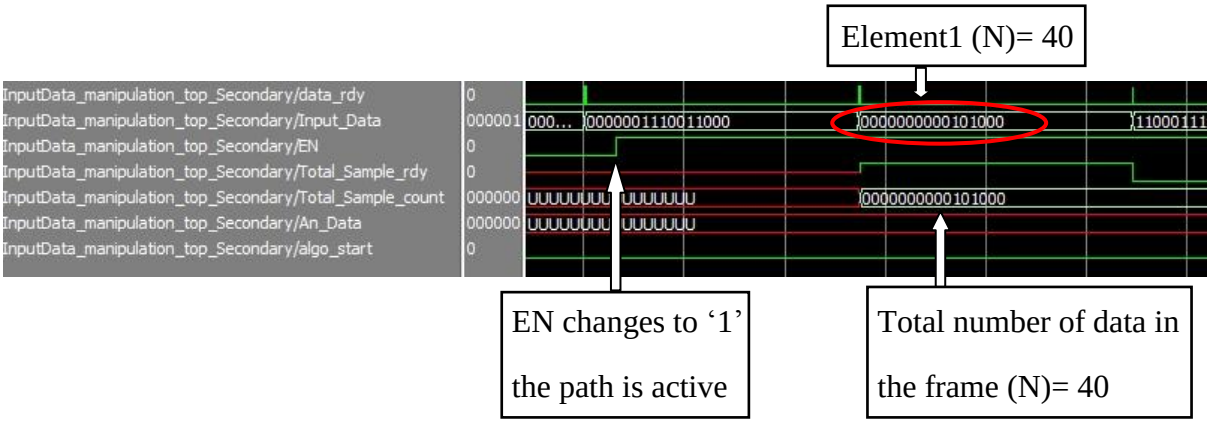


Figure 47 –Waveform of Input Data Manipulation Secondary Module simulation (1)

When the "EN" signal changes to '1', the path is active and the module starts to work. The first input data is the elment1 in Frame 2, which is 40, exactly the same as the value in the table. The total sample number is extracted and transmitted to the output "Total_Sample_count".

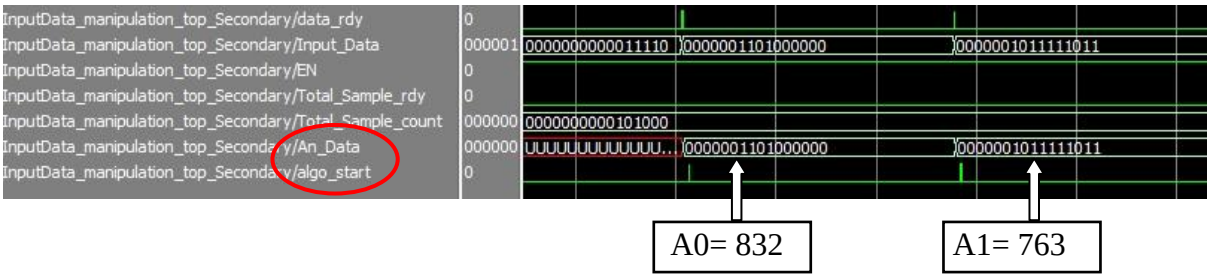


Figure 48 –Waveform of Input Data Manipulation Secondary Module simulation (2)

After the header of the frame has transmitted, the samples (An) start to be extracted and transmitted to the output "An_Data", as well as the algorithm start signal. From the waveform it can be seen that the value of A0 and A1 are exactly the same as the value listed in Table 1.

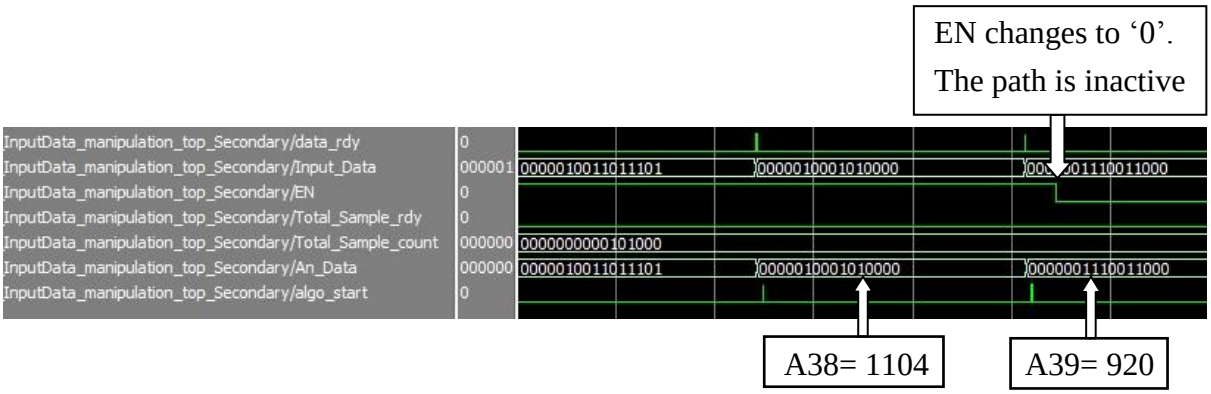


Figure 49 –Waveform of Input Data Manipulation Secondary Module simulation (3)

When the “EN” signal changes to ‘0’, the path is inactive. All the samples in Frame 2 are received. The value of A_{38} and A_{39}, the last two samples in Frame 2, is exactly the same as the value listed in Table 1. Therefore, it can be concluded that the behaviour of the top-level entity of Input Data Manipulation Secondary Module is correct.

5.2.3.2 Algorithm Secondary Module

This module is the core in the path. It is used to calculate the parameters received to get the interim ACFM value of each sample An.

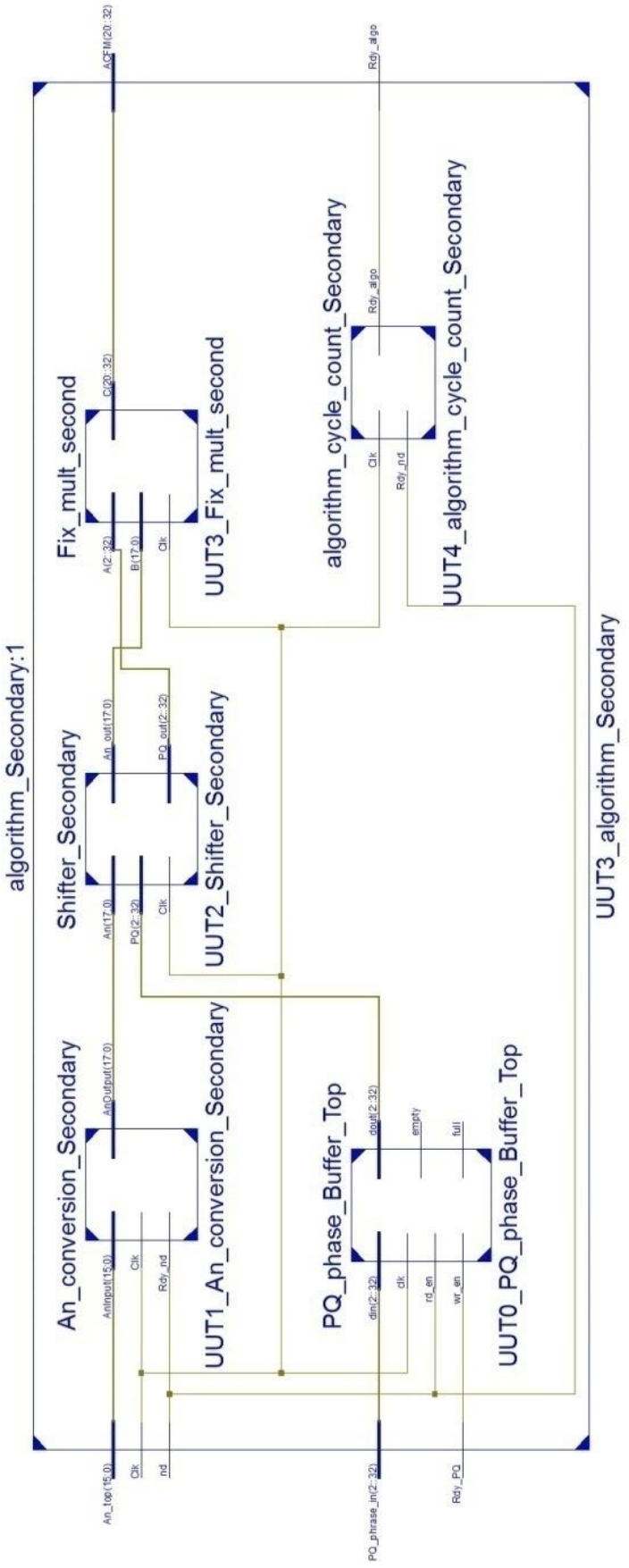


Figure 50 –Algorithm secondary module internal RTL Schematic

Figure 50 shows the internal RTL schematic of the algorithm module in the slave paths. It can evidently be seen that the complexity is greatly simplified compared with the algorithm module in the master path.

The algorithm module in the slave paths consists of five modules. They are all shown in the diagram. The relevant modules are the PQ_phase Buffer and algorithm cycle count secondary module. These two lower-level modules both play important roles. The former stores a series of PQ_phase calculated from the master path, which arrives earlier than the data for the slave paths. The latter does the synchronisation work. The following simulations are on these two modules. Let us begin with the PQ_phase Buffer simulation.

5.2.3.2.1 PQ_phase Buffer

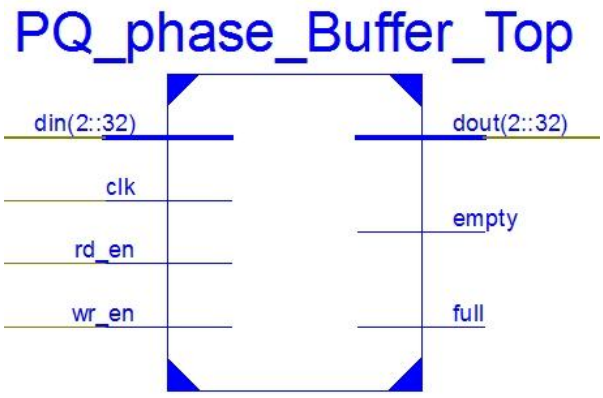


Figure 51 –PQ_phase Buffer RTL Schematic

Figure 51 shows the RTL schematic of the PQ_phase buffer in the slave paths. The PQ_phase Buffer is a FIFO (First In First Out) and the working principle has already been discussed in the methodology section. The FIFO is generated by using Xilinx IP Core generator. The internal structure cannot seen, but the simulation on the ports can be executed as normal.

This simulation is to verify the behaviour of the PQ_phase Buffer by observing the input and output signals.

With the correct simulation of the input manipulation module above, the simulation on the PQ_phase Buffer still uses the Frame 2 listed in Table 1 as the simulated input data in the simulation. The waveform is shown in the graph below in *Figure 52*, *Figure 53*, and *Figure 54*.

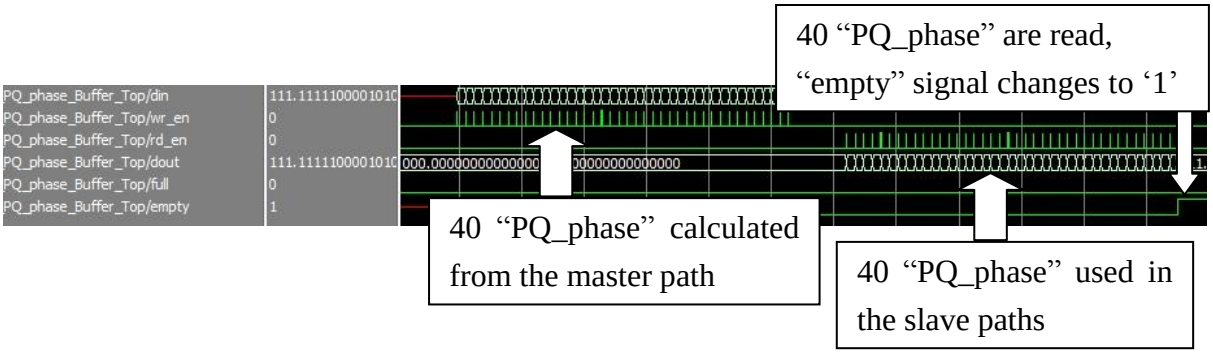


Figure 52 –Waveform of PQ_phase Buffer simulation (1)

The waveform above demonstrates the working principle of the module. When the master path is working, the PQ_phase Buffer stores the 40 “PQ_phase” calculated from the master path, the process of which is indicated by “wr_en” signal. When the slave path (No.2 path in this case) starts working, the 40 “PQ_phase” are taken to be used in the following calculation. After all the 40 “PQ_phase” are read out, the “empty” signal changes to ‘1’, which indicates that no data are stored in the FIFO. Having verified the working principle, the order of the stored data needs to be verified.

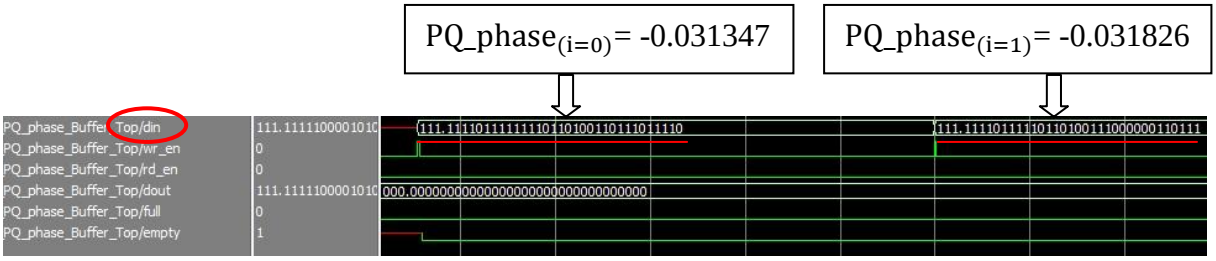


Figure 53 –Waveform of PQ_phase Buffer simulation (2)

The data stored in the “din” vector is sfixed format data, which has 3-bit integer and 32-bit fractions.

$$\text{PQ\_phase}_{(i=0)} = (111.11101111111110110100110111011110)_2 = -0.031347$$

$$\text{PQ\_phase}_{(i=1)} = (111.11101111111110110100110111011110)_2 = -0.031826$$

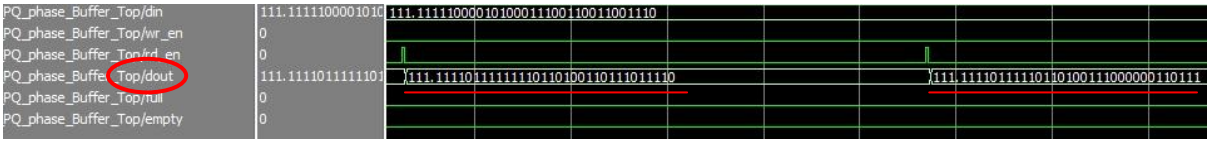


Figure 54 –Waveform of PQ_phase Buffer simulation (3)

The order of the output data from the "dout" vector is exactly the same as the data stored in the "din" vector. Therefore it can be concluded that the behaviour of the PQ_phase Buffer is performing correctly.

5.2.3.2.2 Algorithm cycle count secondary module

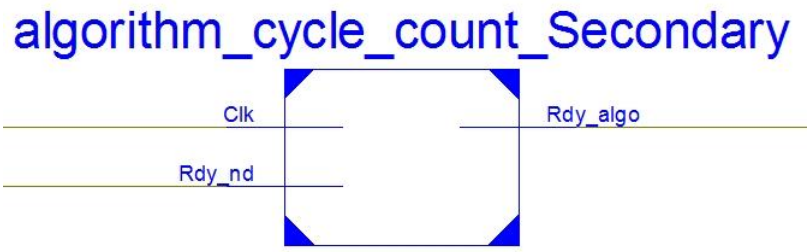


Figure 55 –Algorithm cycle count secondary RTL Schematic

Figure 55 shows the RTL schematic of the algorithm cycle count module in the slave paths. This module is to generate the synchronised algorithm finish signal. The working principle of this module has been discussed in the methodology section.

This simulation is to verify the behaviour of the algorithm cycle count secondary module by observing the input and output signals.

In order to verify the working principle shown in *Figure 20* in the methodology section, the simulation is done on this module.

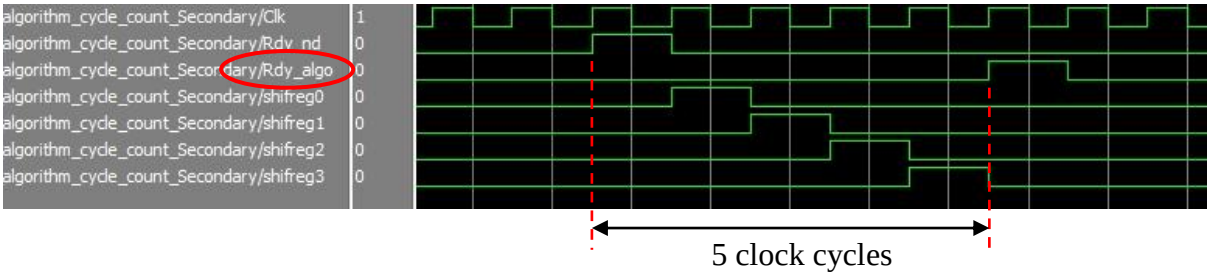


Figure 56 –Waveform of algorithm cycle count secondary module

It can be observed that when the "Rdy_nd" signal changes to '1', the shifting process starts. After 5 clock cycles, the output "Rdy_algo" responds. It indicates that the algorithm process in the slave paths takes a short time period, which is 5 clock cycles. To summarise, the behavior of the algorithm cycle count secondary module is correct.

5.2.3.2.3 Algorithm Secondary Module (Top-level)

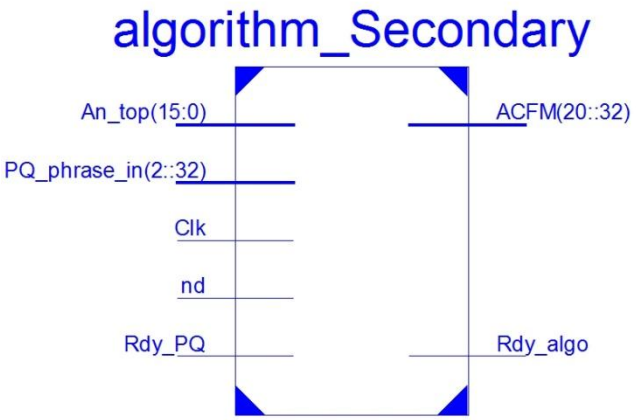


Figure 57 –Algorithm secondary module RTL Schematic

Figure 57 shows the RTL schematic of the algorithm module in the slave paths. This module is used to calculate the parameters received to get the interim ACFM value of each sample An and then send them in order to the following recursive adder module to get the final ACFM result for every frame. The lower-level of the algorithm module has been done. This simulation is to verify the behaviour of the top-level Algorithm Secondary Module by observing the input and output signals.

The Frame 2 listed in Table 1 is used as the simulated input data in the simulation. The input data and the expected results are shown below in Table 4.

Table 4 – Simulated data and expected results in Algorithm Secondary Module simulation

	A_i	$PQ\_phase_{(i)}$	$ACFM_{A_i}$
i=0	832	-0.031347	5.2664
i=1	763	-0.031826	7.5428
…	…	…	…
i=39	920	-0.030097	2.4077

The expected $PQ\_phase_{(i)}$ and $ACFM_{A_i}$ value are listed in the table. The following step is to verify if the simulation results are the same as the expected results. The simulation waveforms are shown below in *Figure 58*, *Figure 59*, and *Figure 60*.

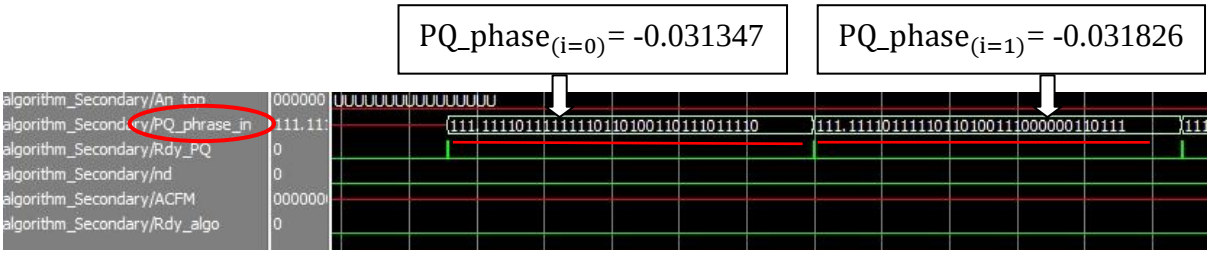


Figure 58 –Waveform of Algorithm Secondary Module (1)

The PQ_phase arrives first, because the data process starts from the master path. The PQ_phase is calculated from it. The value of $PQ\_phase_{(i=0)}$ and $PQ\_phase_{(i=1)}$ have been verified in the PQ_phase Buffer simulation. The "PQ_phase_in" input is correctly responding.

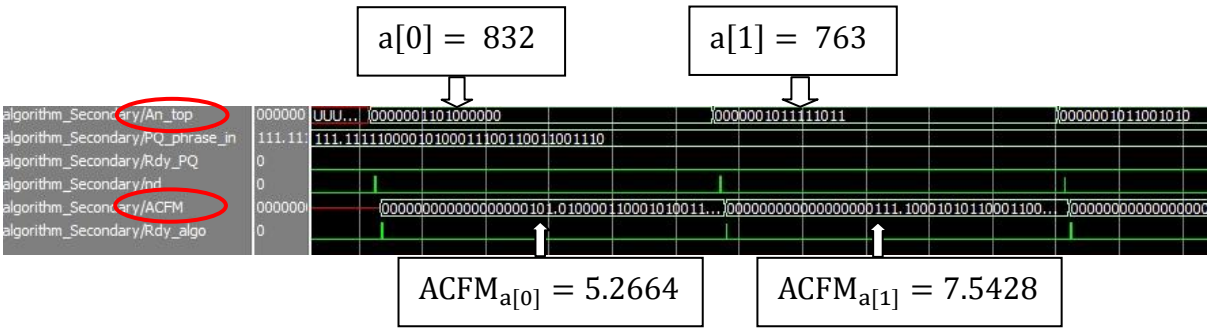


Figure 59 –Waveform of Algorithm Secondary Module (2)

The output "ACFM" is sfixed format data, which has 21-bit integer and 32-bit fractions.

$$\mathrm{ACFM}_{\mathrm{A}_0} = (000000000000000000101.010000110001010011\ \ldots)_2 = 5.2664$$

$$\mathrm{ACFM}_{\mathrm{A}_1} = (000000000000000000111.100010101100011100\ \ldots)_2 = 7.5428$$

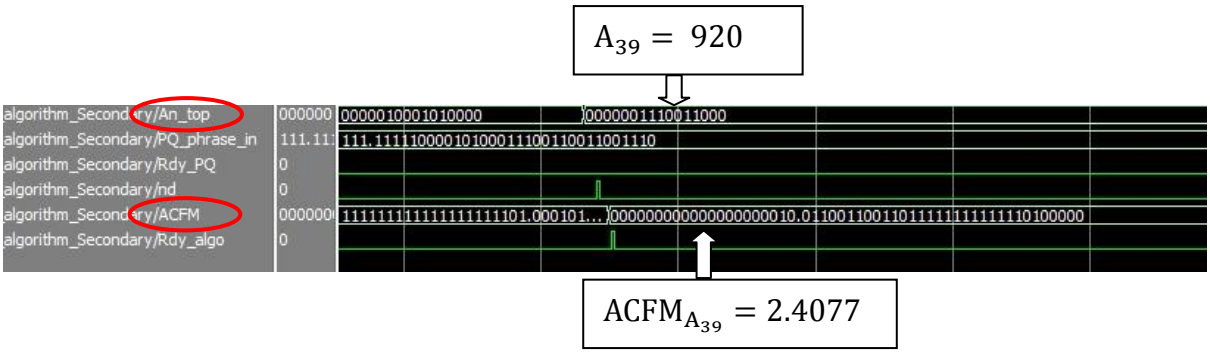


Figure 60 –Waveform of Algorithm Secondary Module (3)

The last sample a[39] and the ACFM value of it are shown in the waveform. It can be seen that the value are exactly the same as the expected results. All the other samples and the ACFM values of them have been verified, and they are all correct. To sum up, the behaviour of the Algorithm Secondary Module is correct.

5.2.4 Recursive Adder Module

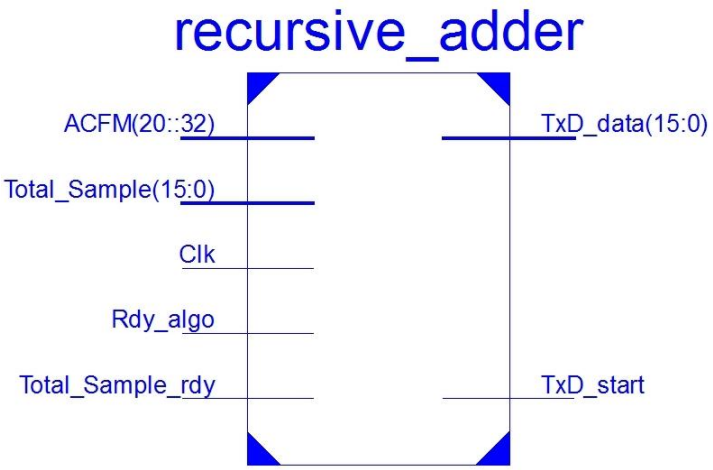


Figure 61 –Recursive adder module RTL Schematic

Recursive adder is used to perform the addition on every single ACFM value that is produced by the algorithm module in each path. The working principle of this module has been discussed in the methodology section.

This simulation is to verify the behaviour of recursive adder by observing the input and output signals.

The Frame 2 listed in Table 1 is used as the simulated input data in the simulation. The input data and the expected results are shown below in Table 5.

Table 5 – Simulated data and expected results in recursive adder simulation

	A_i	$PQ\_phase_{(i)}$	$ACFM_{A_i}$
i=0	832	-0.031347	5.2664
i=1	763	-0.031826	7.5428
…	…	…	…
i=39	920	-0.030097	2.4077
		$\sum ACFM_{A_i}$	968.45

The expected $PQ\_phase_{(i)}$ and $ACFM_{A_i}$ value are listed in the table. The following step is to verify if the simulation results are the same as the expected results. The simulation waveforms are shown below in *Figure 62*.

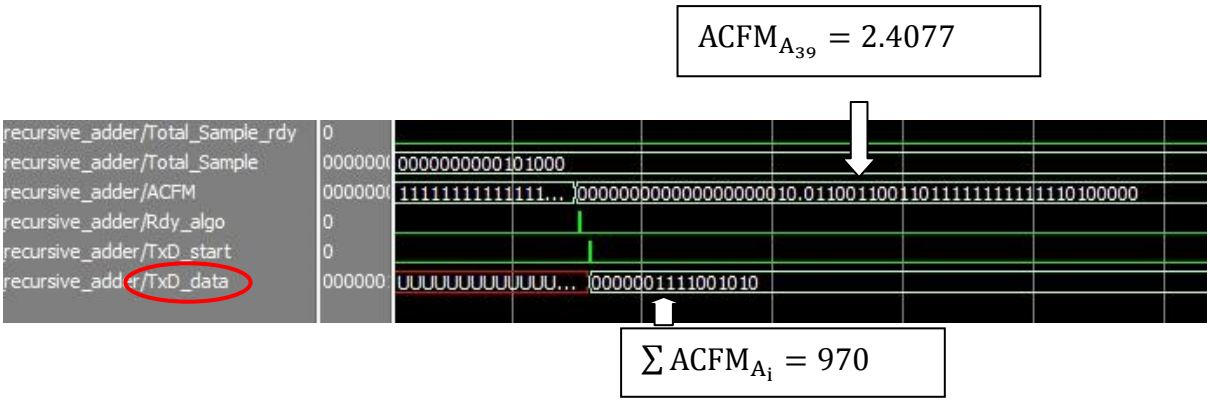


Figure 62 –Waveform of recursive adder module

The output signal “TxD_data” is the signed 16-bit ACFM result. It can be seen from the waveform that the value is 970, which is in the accuracy tolerance compared with the expected result. All the other samples and the ACFM values of them have been verified and they are all correct. When the ACFM result is ready, the transmission start signal “TxD_start” changes to ‘1’. To sum up, the behaviour of the recursive adder module is correct.

5.2.5 Synchronize Buffer

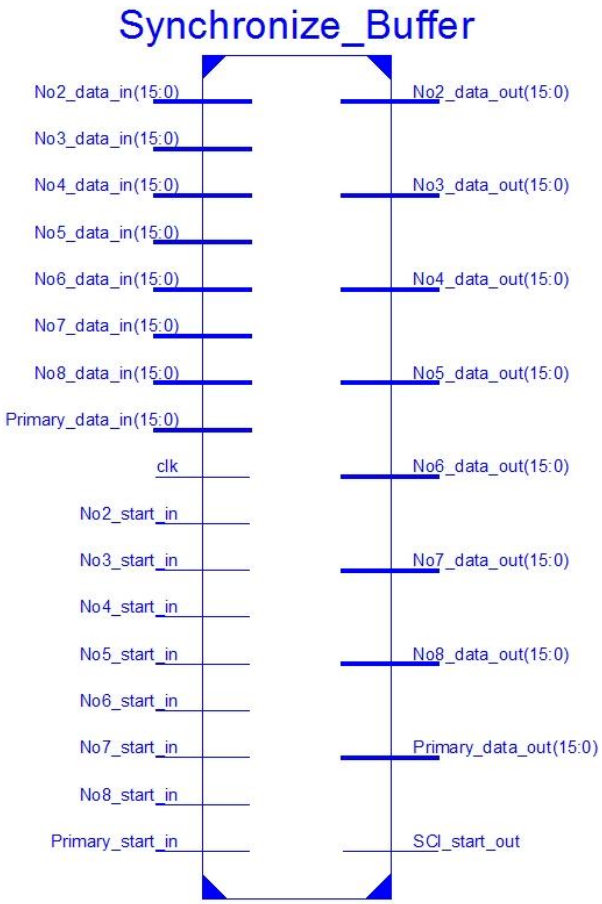


Figure 63 –Synchronize Buffer RTL Schematic

Synchronize Buffer works to buffer the ACFM results for all the eight paths, and guarantee the results can arrive at the SCI module at the same time. The working principle of this module has been discussed in the methodology section.

This simulation is to verify the behaviour of Synchronize Buffer by observing the input and output signals. The previous simulations are all done. This simulation is to put all the eight paths together to view the inputs and outputs.

The simulation waveform is shown below in *Figure 64*.

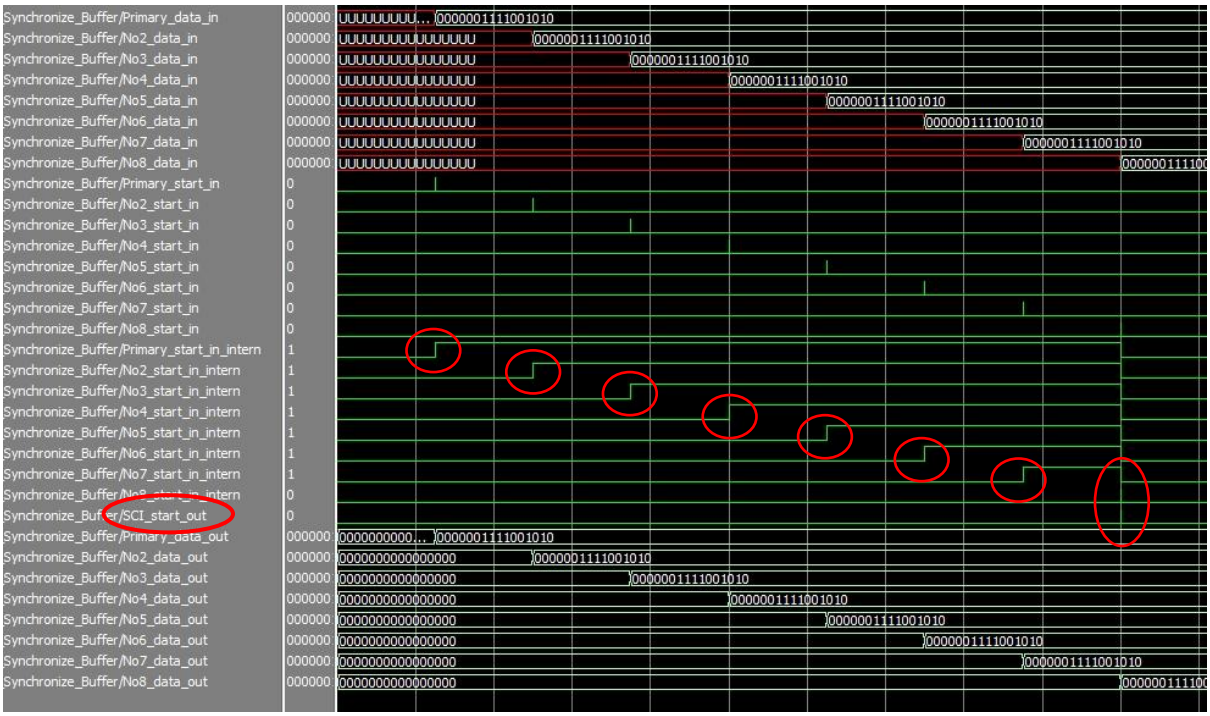


Figure 64 –Waveform of Synchronize Buffer

From the waveform, it can be seen that the SCI starting transmission signal “SCI_start_out” changes to ‘1’ when “No8_start_in” changes to ‘1’, which indicates the data process in the No8 path has finished. The internal signals, which are to latch the control signals from the eight paths, have all changed to ‘1’, which means the data processes in all the 8 paths have finished, therefore the “SCI_start_out” should change to ‘1’. To sum up, the behaviour of Synchronize Buffer is correct.

5.2.6 Serial Peripheral Interface (SPI) slave module

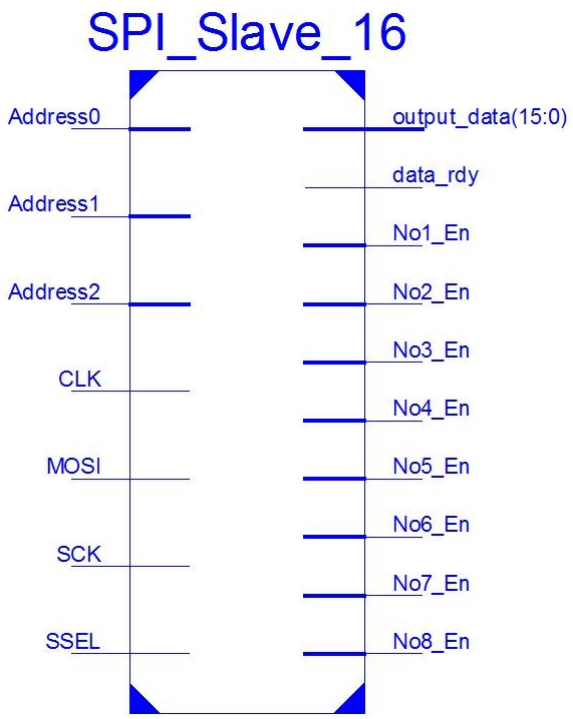


Figure 65 –SPI Slave module RTL Schematic

The SPI module acts as slave mode and is used to receive data from DSP processor. It consists of seven input signals and ten output signals. The inputs are eight-path data stream (MOSI), SPI communication clock (SCK), chip select signal (SSEL), system clock (Clk) and three path address lines (Address2, Address1, Address0). The outputs are control signal (Data_rdy), 16-bit output data (output_data) and eight paths enable signals (from No1_EN to No8_EN). The working principle of this module has been discussed in the methodology section.

This simulation is to verify the behaviour of the SPI module by observing the input and output signals.

The Frame 2 listed in Table 1 is used as the simulated input data in the simulation. The working principle is shown in the waveform below.

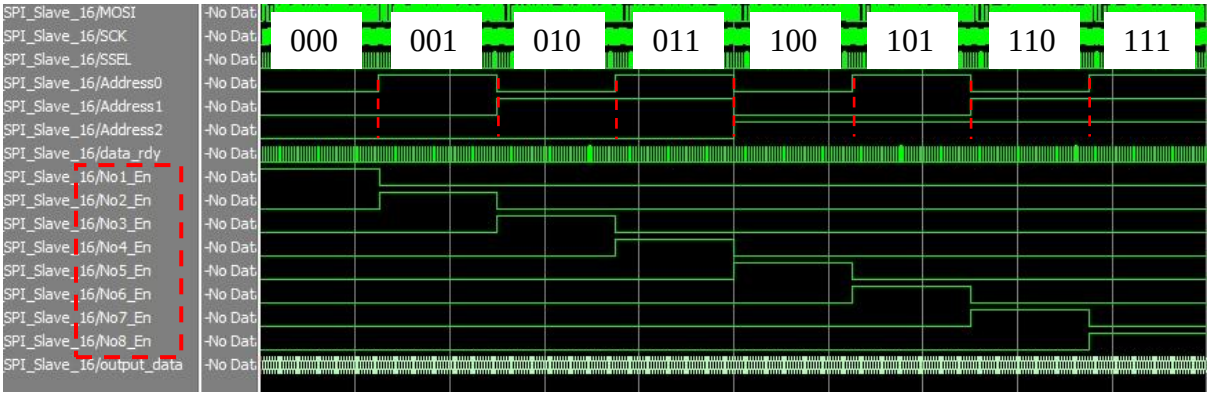


Figure 66 –Waveform of Serial Peripheral Interface (SPI) (1)

The waveform above shows the how the SPI module assigns the input data to the eight paths. The 3-bit address line indicates the address the data belongs to. When the address equals "000", the No1 path (the master path) enable signal "No1_En" changes to '1' (active). Once all the data in one frame have been received, the signal "No1_En" changes back to '0'. The situation is the same in the other seven paths. In the view of the transmission of each data, the waveform is shown below in *Figure 67*.

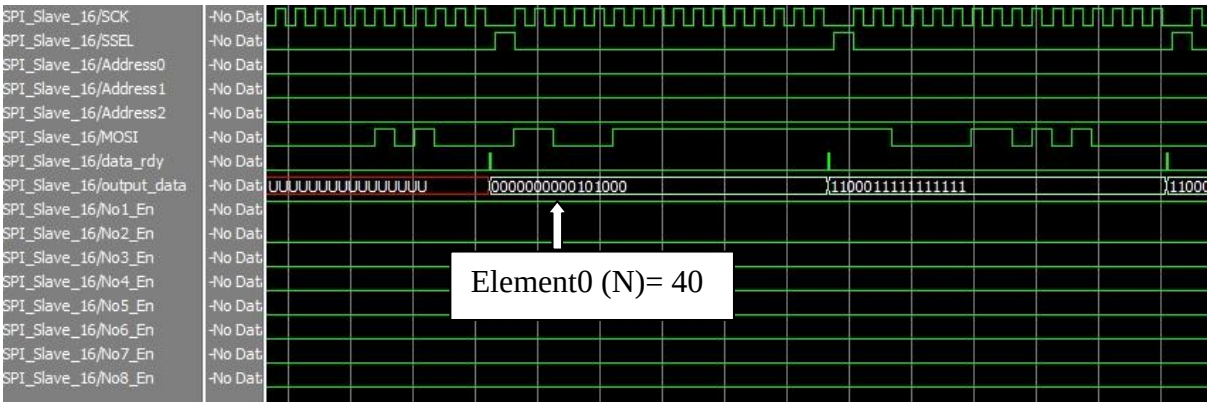


Figure 67 –Waveform of Serial Peripheral Interface (SPI) (2)

From the simulation waveform above, it can be seen that the chip select signal (SSEL) stays low along the data transfer across the SPI bus. The first output data is element1 in Frame 2, which is the total number of data in one frame. The "data_rdy" control signal is toggled to high when the data is ready at the output port. To sum up, the behavior of the SPI module is correct.

5.2.7 Serial Communication Interface (SCI)

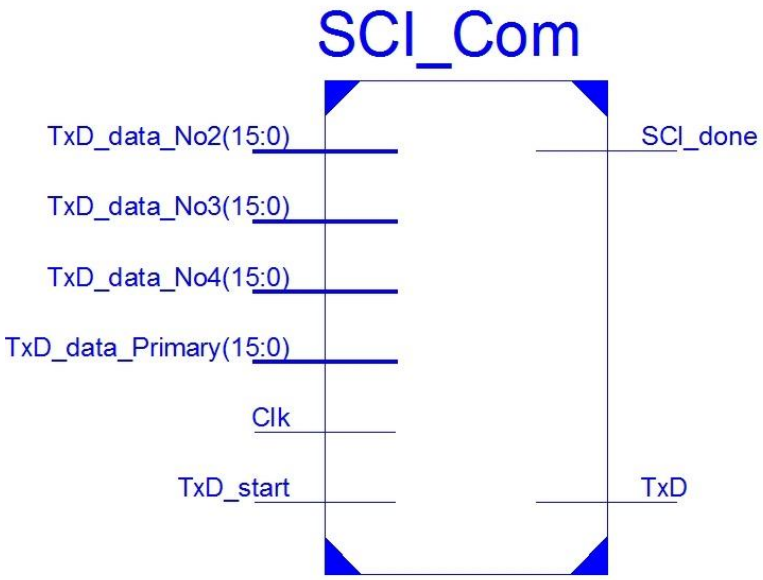


Figure 68 –SCI module RTL Schematic

Two SCIs are used in this project. Each of them performs the same behaviour. They take responsibility for each of the four paths respectively. The SCI module is used to send the final ACFM value of the four paths from the FPGA device to the PC for further offline visual display. It consists of 6 input ports, system clock (Clk), four TxD_data (16-bit data) from the four paths and a control signal (TxD_start), and 2 outputs are TxD (serial bit stream) and SCI transmission finish signal. The module starts to read data from the four input data ports when the TxD_start signal equals '1', which indicates starting the communication.

This simulation is to verify the behaviour of the SCI module by observing the input and output signals.

The simulated data for the four paths inputs are set the same, which is "0000000100000000", in order to view the transmission protocol in the simulation waveform. The result is shown below in *Figure 69*.

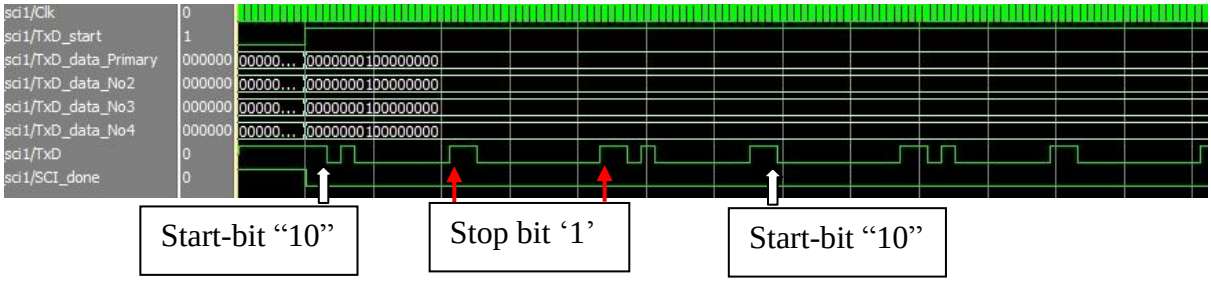


Figure 69 –Waveform of Serial Communication Interface (SCI) (1)

It can be observed from the waveform that the start bits used in the transmission are “10” and the stop bit used is “1”. That means the header of each 16-bit ACFM result is “10” and the footer is ‘0’. The real data is the 16 bits between the header and footer. Also, it can be seen that the upper byte is transferred first, followed by the lower byte. From the output signal “TxD” in the waveform, it can be seen that the upper byte “00000001” is transferred first, followed by the lower byte “00000000”. Having made clear the transmission rules, the order of the data from the four paths transmission needs to be verified.

The simulated input data is listed in Table 6 below.

Table 6 – Simulated data in SCI module simulation

TxD_data from the master path	0000000000000001
TxD_data from the No2 path	0000000000000010
TxD_data from the No3 path	0000000000000011
TxD_data from the No4 path	0000000000000100

And the simulation waveform is shown in *Figure 70* below.

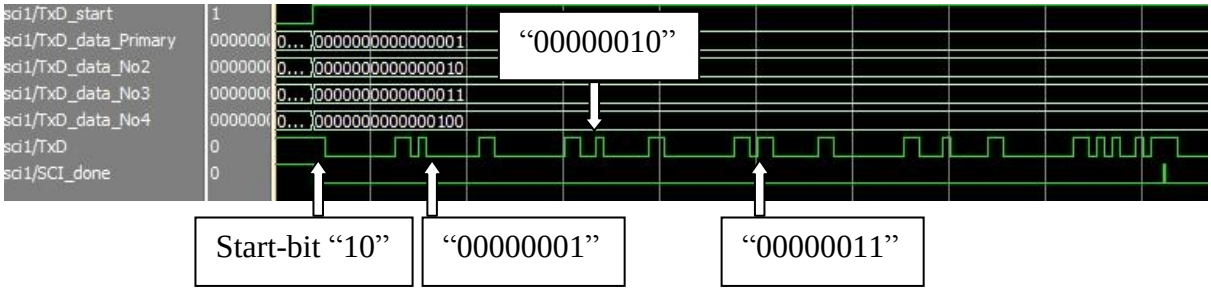


Figure 70 –Waveform of Serial Communication Interface (SCI) (2)

From the waveform, it can be concluded that the "TxD_data" of the four paths are transferred in order. For the master path, the upper byte "00000000" is transferred first followed by the lower byte "00000001". Then for the No2 path, the upper byte "00000000" is transferred first followed by the lower byte "00000010". To sum up, the behaviour of the SCI module is performing correctly.

5.3 System testing (Top Level Entity)

The master path and the slave paths have been simulated above, as well as their behaviour individually verified. The top-level structure of the eight paths system should then be considered.

Having implanted the top-level structure of the eight paths system, Xilinx ISE could generate the RTL schematic as the designer expects. The RTL schematic of the eight paths top-level entity is shown in *Figure 71* below.

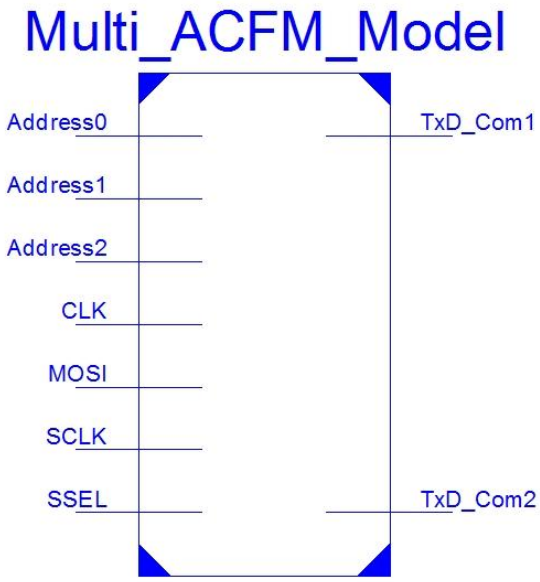


Figure 71 –Multiple ACFM model RTL Schematic

As already discussed, the inputs from the DSP are the chip select signal (SSEL), the serial data signal (MOSI) and SPI communication clock signal (SCLK), as well as three address wire signals (Address0, Address1, Address2). The working principle has been discussed in the methodology section. The internal structure of the eight paths system is shown below in *Figure 72*.

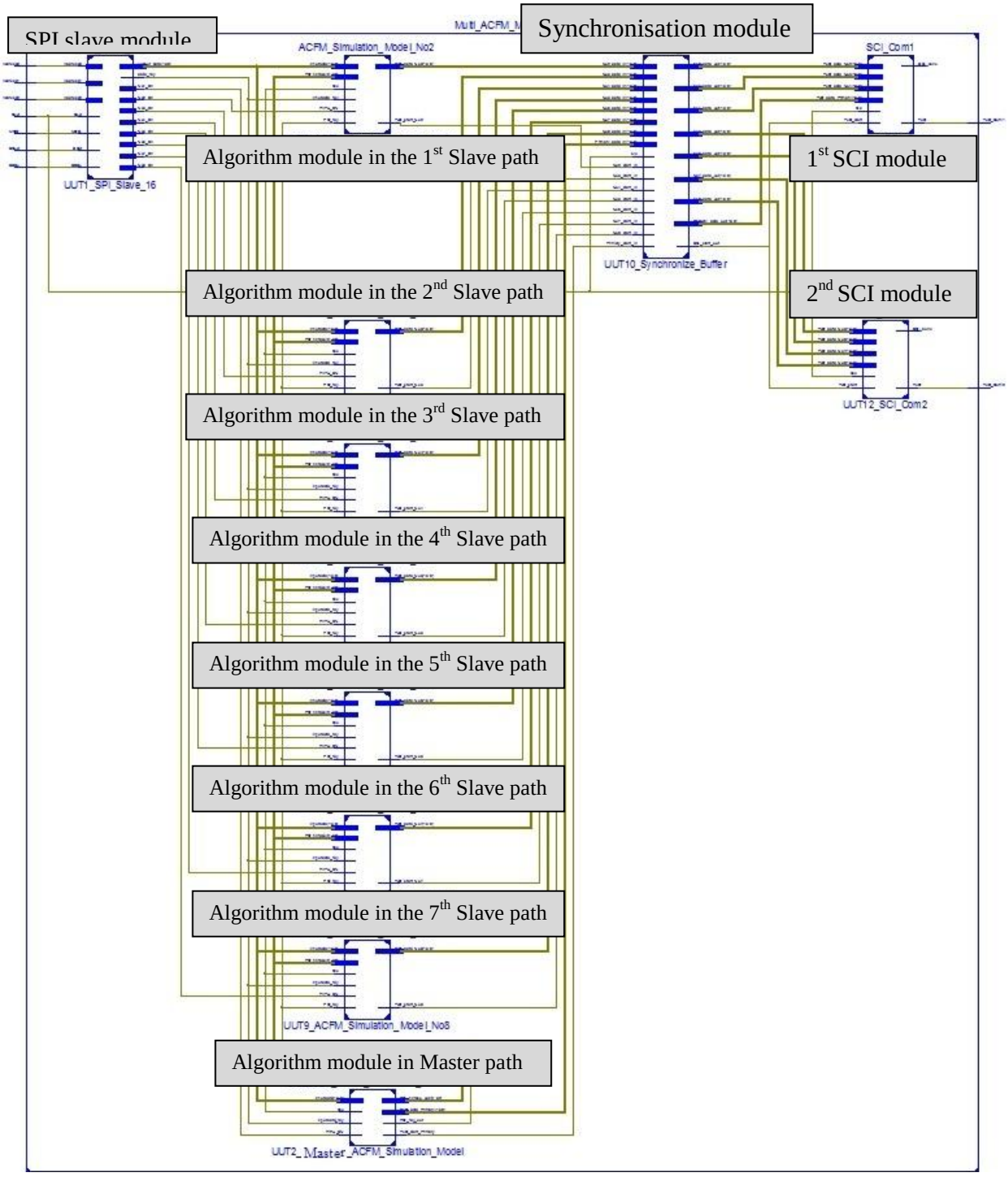


Figure 72 –Eight paths system RTL Schematic

The structure is too complex to display. Although the details cannot be clearly viewed from the diagram above, it can still be observed that the top-level entity consists of four parts: SPI slave, algorithm realisation in the eight paths, synchronisation and two SCI modules.

In order to simulate the behaviour of receiving data from the DSP, a simulated data generator is used to mimic the data transmitted from the DSP and simulate the behaviour of the DSP. The connection between is shown in *Figure 73* below.

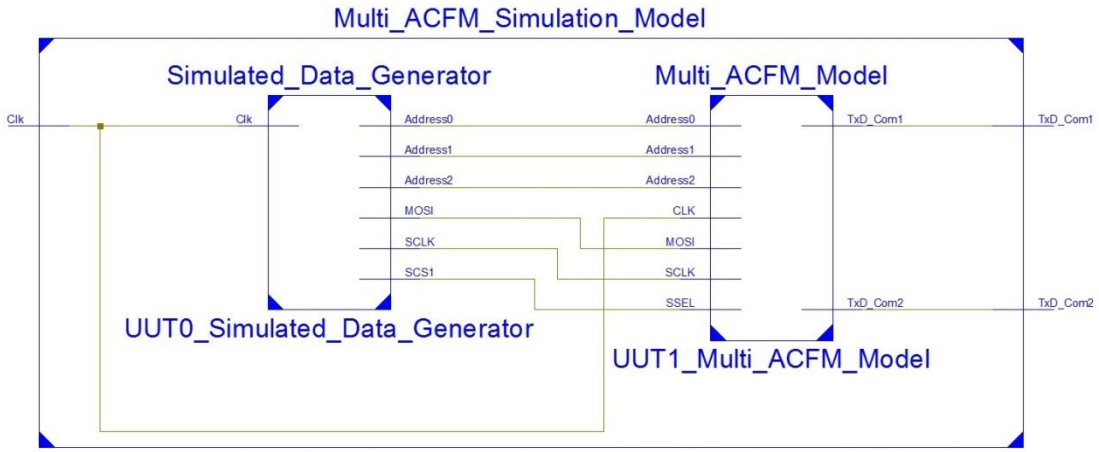


Figure 73 –RTL Schematic of the master path simulation structure

The Frame 1 and Frame 2 listed in Table 1 are used as the simulated input data in the simulation. In order to verify if the simulated ACFM result is correct, the results provided by TSC company and MATLAB are used as criteria. The criteria and the expected results are shown in Table 2 and the simulation waveform is shown below in *Figure 74*.

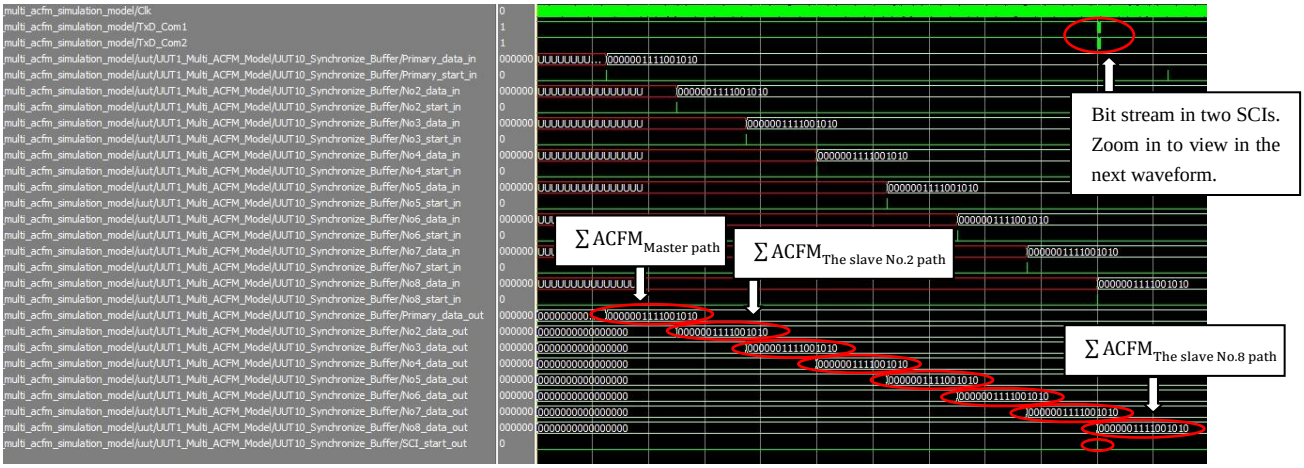


Figure 74 –Waveform of system testing (1)

The ∑ACFM results shown in *Figure 74* are the calculation results of the Frame 1 and Frame 2. The Frame 1 is used as the input sent to the master path, while the Frame 2 is used as the

inputs sent to the seven slave paths. From the waveform it can be seen that the results are the same between the eight paths. The results are:

$$\text{Result}_{\text{Master path}} = \text{B"0000001111001010"} = \text{D"970"}$$

$$\text{Result}_{\text{Slave paths}} = \text{B"0000001111001010"} = \text{D"970"}$$

Comparing the results with the criteria in Table 2, it can be concluded that the results are correct. The SCI bit stream will then be zoomed in to view the behaviour, shown in the waveform below in *Figure 75*.

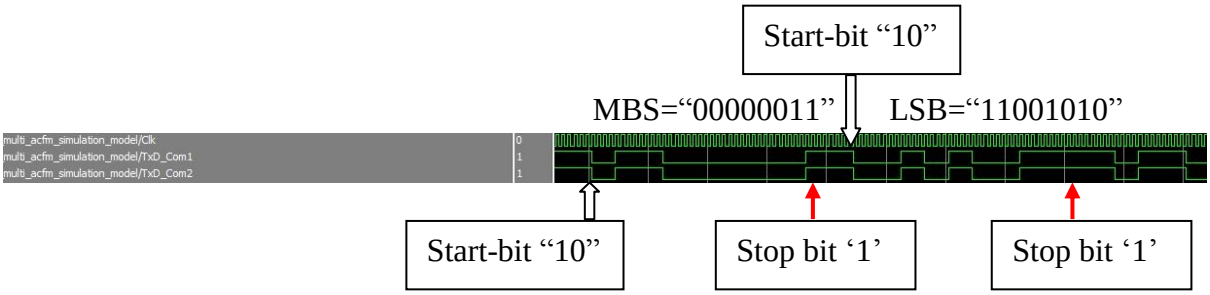


Figure 75 –Waveform of system testing (2)

As discussed in the SCI simulation part, the start bits used in the transmission are "10" and the stop bit used is "1". The real data is the 16 bits between the header and footer. Also, the upper byte is transferred first, followed by the lower byte. From the output signal "TxD_Com1" and "TxD_Com2" in the waveform, it can be seen that the upper byte (MSB) "00000011" is transferred first, followed by the lower byte (LSB) "11001010". Combining the two bytes together, it makes the 16-bit $\sum$ACFM result. To sum up, the behaviour of the eight paths top-level system is performing correctly.

From the $\sum$ACFM results of all eight paths, it can be observed that there is still a slight difference between the ACFM result in the FPGA simulation and the criteria. As discussed above in the master path simulation part, the reason is mainly due to the fixed input data format of the trigonometric module. It only accepts data input with 16-bit representation and 3-bit integer following with 13-bit fraction. It causes some variation in the final solution of

the master path. Although the trigonometric module is not used in the slave paths, the "PQ_phase", which involves the trigonometric module, is calculated from the master path. Hence, all eight paths have the same variation and the 16- bit representation can produce a high accuracy answer for all eight paths.

Having verified the behaviour of the top-level system in FPGA, the synthesise process and the realisation of the hardware should then be considered. In the following section, the synthesis performance and the hardware process will be discussed.

6 Synthesis results

The following section is to discuss the synthesis performance of the eight paths system in FPGA and the hardware implementation on FPGA Spartan-6 chip. Then the verification of the behaviour of ATLYS FPGA Board will be executed last.

6.1 Synthesis performance

The synthesis performance can be observed from the synthesis report. Next, let us discuss it. *Figure 76* below shows the device utilisation summary.

Device Utilization Summary				
Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	11,678	54,576	21%	
Number used as Flip Flops	11,678			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	9,563	27,288	35%	
Number used as logic	8,297	27,288	30%	
Number using O6 output only	5,679			
Number using O5 output only	783			
Number using O5 and O6	1,835			
Number used as ROM	0			
Number used as Memory	809	6,408	12%	
Number used as Dual Port RAM	672			
Number using O6 output only	672			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used as Single Port RAM	0			
Number used as Shift Register	137			
Number using O6 output only	88			
Number using O5 output only	1			
Number using O5 and O6	48			
Number used exclusively as route-thrus	457			
Number with same-slice register load	414			
Number with same-slice carry load	42			
Number with other load	1			
Number of RAMB16BWERs	0	116	0%	
Number of RAMB8BWERs	0	232	0%	
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%	
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%	
Number of BUFG/BUFGMUXs	2	16	12%	
Number used as BUFGs	2			
Number used as BUFGMUX	0			
Number of DSP48A1s	40	58	68%	
Average Fanout of Non-Clock Nets	2.97			

Figure 76 –Device Utilisation Summary

From *Figure 76* it can be observed that this design uses about 11,600 registers out of 54,500 that are available in the Spartan-6 (XC6SLX45) FPGA chip. The utilisation is about 21%. Besides that, the design uses about 9,500 look up tables (LUTs) out of 27,200 that are available in the chip. The utilisation is about 35%. In FPGA, most of the operations are implemented using LUTs. Lastly, but most importantly, the design uses 40 DSP48 modules out of the 58 that are available in the chip. The utilisation is about 68%. This percentage is taken up by the complicated mathematical calculations mainly in the master path, such as division and trigonometric calculation. Considering the utilisation of these indexes, the FPGA chip is around half used, which is appropriate for routing as well as further expanding in the future.

6.2 Efficiency in use of FPGA

In this section, a comparison is made between methods to explain how efficiently the FPGA works. In this project, the eight paths system is implemented by dividing the paths into one master path and seven slave paths, in which way much space can be saved, as expected.

Meanwhile, a simple way to implement this system is to copy the one-single path system eight times, which could theoretically work. Therefore, the comparison is between the master and slave paths system and the simple eight paths system.

Firstly, let us view the device utilisation summary (DUS) of the one-single path system. The summary is shown below in *Figure 77*.

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Total Number Slice Registers	8,824	11,776	74%	
Number used as Flip Flops	8,823			
Number used as Latches	1			
Number of 4 input LUTs	7,401	11,776	62%	
Number of occupied Slices	5,886	5,888	99%	
Number of Slices containing only related logic	5,886	5,886	100%	
Number of Slices containing unrelated logic	0	5,886	0%	
Total Number of 4 input LUTs	8,021	11,776	68%	
Number used as logic	7,199			
Number used as a route-thru	620			
Number used as Shift registers	202			
Number of bonded IOBs	9	372	2%	
IOB Flip Flops	1			
Number of BUFGMUXs	2	24	8%	
Number of MULT18X18SIOs	20	20	100%	
Average Fanout of Non-Clock Nets	2.44			

Figure 77 –DUS of one-single path system(Ma, 2010-2011)

The FPGA chip used in the one-single path system is Spartan-3A XC3S700AN. From *Figure 77*, it can be observed that this design used about 8,000 registers out of 10,000 that are available in the Spartan-3A XC3S700AN FPGA chip. The utilisation is about 74%. In addition, the design also used about 8,000 look up tables (LUTs) out of 10,000 that are available in this device. The utilisation is about 66%. Based on the utilisation in the one-single path system, the expected simple eight paths system utility should be eight times greater, which is 64,000 registers and 64,000 LUTs. Therefore it is evident that the Spartan-3A XC3S700AN chip is not enough for the eight paths system. Comparing the expected utilisation of the simple eight paths system and the master and slave paths system, the space saved is listed below in Table 7.

Table 7 – Comparison between simple eight paths and master & slave paths system

		Simple eight paths system	Mater &slave paths system
Spartan-3A XC3S700AN	Registers usage	64,000/10,000 (640%)	11,600/10,000 (116%)
	LUTs usage	64,000/10,000 (640%)	9,500/10,000 (95%)
	MULT18X18SIOs (trigonometric function) usage	160/20 (800%)	40/20 (200%)
Spartan-6 XC6SLX45	Registers usage	64,000/54,500 (117%)	11,600/54,500 (21%)
	LUTs usage	64,000/27,200 (235%)	9,500/27,200 (35%)
	DSP48A1s (trigonometric function) usage	160/58 (285%)	40/58 (68%)

The table above summarises the utilisation comparison between the simple eight paths system and the master and slave paths system. It can be observed that much space is saved in the latter system. The registers and LUTs used in the simple eight paths system are six times as large as the utilisation in the master and slave paths system. Because of this, even the Spartan-6 (XC6SLX45) is not big enough for the simple sight paths system but can cope with the smarter master and slave paths system. Meanwhile, the registers available in the Spartan-3A XC3S700AN FPGA chip are not enough for the master and slave paths system, so the FPGA chip is changed into Spartan-6 (XC6SLX45) in this project. The most important factor taken into consideration for this choice is the advanced DSP cells internal, of which the trigonometric function needs and consumes a great deal.

Having made clear the synthesis performance of the system, the hardware implementation can be executed in order to get the real data out from the FPGA board- ATLYS.

6.3 Hardware implementation

In this section, the hardware implementation will be discussed. In order to ensure the specific FPGA device can perform the designed operations, the implementation constrain file (".ucf" file) is needed to specify in the FPGA project. This is because the synthesis tool, place and route tools need to take care of the timing issue during the synthesis process. The main function of the ".ucf" file is to tell the synthesis tool about which pins of the FPGA chip correspond to the output signals defined in the programme. In this project, 6 IO pins are used to perform the SPI communication with the DSP; the communication clock (SCK), chip select signal (SSEL), and three path address signals (Address2, Address1, Address0) from the DSP SPI Master. In addition, 2 IO pins are used to perform the SCI communication with the PC; the bit stream from SCI1 (TxD_Com1) and the bit stream from SCI2 (TxD_Com2).

To download the program file into the Spartan-6 (XC6SLX45) FPGA chip on the ATLYS board, we used Adept, the software developed by Digilent Inc., the supplier of ATLYS board. The user interface of this software is shown below in *Figure 78*.

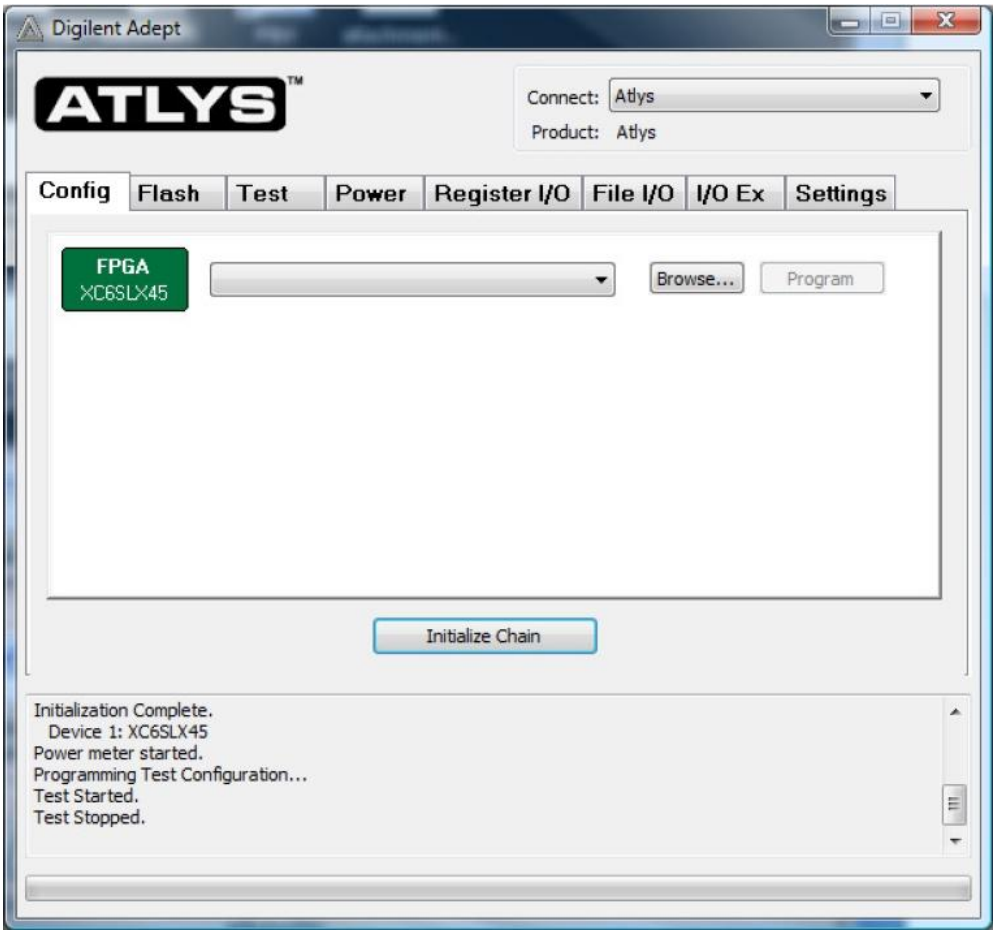


Figure 78 –User Interface of Adept Software

Once the designer plugs the ATLS board in, the software will recognise and display it. After initialising the chain, the designer is able to import the program file and then program it into the FPGA chip on the board. The Adept software also integrates the self testing, power adjustment, I/O expanding, etc.

6.4 Verification of ATLYS

To verify the behaviour of the ATLYS board, the output ports TxD_Com1 and TxD_Com2 should be inspected. The serial communication function in Matlab is used to capture the bit stream transferred from the ATLYS board.

The important part of the codes in Matlab is shown below in *Figure 79*.

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Vbaudrate=[9600, 19200, 28800, 38400, 57600, 115200, 128000, 230400, 460800];
s =serial('COM2');
s.InputBufferSize=1024;
s.timeout=8.0;
s.BaudRate=460800;
s.parity='none';
s.StopBits=1;
s.Terminator='LF';
s.FlowControl='hardware';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

'COM5'

Figure 79 –Serial communication realisation in Matlab

As shown in the figure, the output "TxD_Com1" is assigned as "Com2" in the device manager in Windows7 OS. The output "TxD_Com2" is assigned as "Com5". Therefore, when testing "TxD_Com2", the code modification is to change "COM2" to "COM5".

In the simulated data generator, two frames of healthy rail section are used, which are named Frame 3 and Frame 4. They are shown in Table 8 below, and are used as the inputs and the frames are sent cyclically.

Table 8 – Simulated two frames from DSP

	Frame 3	Frame 4
Element	**Clip 3**	**Clip 4**
N	40	40
β	260	260
Fc, LSB	50000	50000
Fc,MSB	0	0
Fs,LSB	33950	33950
Fs,MSB	30	30
A0	841	857
A1	766	773
A2	725	769
A3	738	783
A4	807	771
A5	891	893
A6	998	986
A7	1141	1162
A8	1333	1339
A9	1544	1478
A10	1773	1746
A11	1995	1952
A12	2213	2212
A13	2462	2473
A14	2726	2640
A15	2950	2872
A16	3141	3119

	Frame 3	Frame 4
Element	**Clip 3**	**Clip 4**
A17	3317	3277
A18	3494	3501
A19	3638	3592
A20	3739	3687
A21	3782	3760
A22	3816	3807
A23	3818	3829
A24	3761	3794
A25	3693	3680
A26	3558	3559
A27	3384	3418
A28	3218	3290
A29	3039	3071
A30	2816	2821
A31	2568	2581
A32	2329	2377
A33	2072	2143
A34	1865	1911
A35	1630	1632
A36	1407	1448
A37	1214	1258
A38	1061	1119
A39	939	988

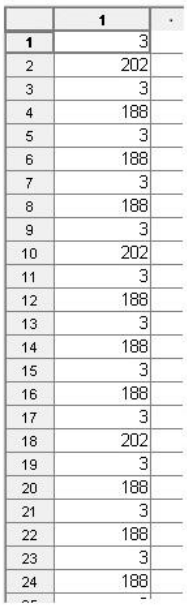
In order to verify whether the simulated ACFM results are correct, the results provided by TSC company and MATLAB are used as criteria. The ACFM results of the two frames in these two methods are shown in Table 9 below.

Table 9 – The expected ACFM results in Frame 3 and Frame 4

	Frame 3	**Frame 4**
ACFM result from TSC	968	954
ACFM result from Matlab	967.57	954.25

In this testing, Frame 3 is provided to the master path and Frame 4 is provided to the other seven slave paths. In this way, different conditions on the whole track surface are simulated.

The data received from "TxD_Com1" is shown in the Matlab workspace in *Figure 80* and data received from "TxD_Com2" is shown in *Figure 81*.



	1
1	3
2	202
3	3
4	188
5	3
6	188
7	3
8	188
9	3
10	202
11	3
12	188
13	3
14	188
15	3
16	188
17	3
18	202
19	3
20	188
21	3
22	188
23	3
24	188

Figure 80 –Bit stream from "COM2" captured in Matlab

	1
1	3
2	188
3	3
4	188
5	3
6	188
7	3
8	188
9	3
10	188
11	3
12	188
13	3
14	188
15	3
16	188
17	3
18	188
19	3
20	188
21	3
22	188
23	3
24	188

Figure 81 –Bit stream from "COM5" captured in Matlab

As previously discussed, the output signal "TxD_Com1" transfers the bit stream from the first four paths (the master path and No.2 to No.4 slave paths), while the output signal "TxD_Com2" transfers the bit stream from the following four paths (No.5 to No.8 slave paths). The received 16-bit data from the master path, which is the 2 bytes shown in the figure, is "3" and "202". Transforming the two bytes into binary representation:

$$\mathrm{D"3"} = \mathrm{B"00000011"}$$

$$\mathrm{D"202"} = \mathrm{B"11001010"}$$

So the 16-bit data is:

$$\mathrm{B"00000011\_11001010"} = \mathrm{D"970"}$$

Referring to the received 16-bit data from the seven slave paths, the 2 bytes are:

$$\mathrm{D"3"} = \mathrm{B"00000011"}$$

$$\mathrm{D"188"} = \mathrm{B"10111100"}$$

So the 16-bit data is:

$$B"00000011\_10111100" = D"956"$$

From the figures in Table 9, it can be observed that there is slight difference between the ACFM result in the FPGA simulation and the criteria. As discussed before, this is due to the number of bit that are used to represent the data operation and also the limitations of the trigonometric module that only accepts data input with 16-bit representation. The input radian value could not be represented more accurately than the fractions in the 16-bit input format. However, the accuracy is still high when it is compared with the criteria. Therefore it can be concluded that the FPGA board, the ATLYS board, is working accurately.

Having synthesised and verified the behaviour of the FPGA board, the parallel processing chain for the eight ACFM paths has been built and the mandatory requirements (discussed in Section 4.2) of the FPGA part that the system has implemented is listed below:

Mandatory features

1) Process NDT data from 8 channels.
2) 12-bit quantisation for the ADC, and 16-bit data processing in FPGA.
3) An eight channel parallel ACFM algorithm implementation in FPGA.
4) The baud rate could reach 1 Mbps.
5) The input interface of FPGA could fit into an eight DSP chip output interface.
6) Synchronisation of 8 probes' data processing chain.

As discussed in the system description, the eight ACFM paths system is comprised of eight ACFM probes, a DSP development board, an FPGA device and a PC. The DSP takes the responsibility for real time data acquisition, while the PC is used to display the results of the eight paths. Considering when cracks occur on the track surface, the change in the ACFM results is significant and quite obvious to see from the changing curve. It can hardly be seen from the static figures captured in Matlab, but can be detected in the changing curve. Also,

given the high speed transferring data flow, the capture work realized by displaying the numeric figures is not easy for the user to view the change. In order to enhance the performance of the PC display, visualisation software is developed to present the curve of the eight paths results in real time. In the next section the changing curves will be displayed on the PC screen. The visualisation software will be discussed in detail.

7 Visualisation on PC

Visualisation software is developed to present the curve of the eight paths results in real time. The software used in the visual display on the PC is the NI LabWindows CVI 2009 free 30 days trial version. The aim of the programme is to receive the bit stream from two serial ports and display them on the screen in order.

Specifically, data from “TxD_Com1” (Window7 OS assigned it “COM2”) are the ACFM results of the master path and the No.2 to No.4 slave paths, whilst data from “TxD_Com2” (Window7 OS assigned it “COM5”) are the ACFM values of No.5 to No.8 slave paths. For each Com, ten bytes make up one package in the programme. The format of the package is shown below in *Figure 82*.

(0H）	（0L）	（1H）	（1L）	（2H）	（2L）	（3H）	（3L）	0x00	0XA5

Figure 82 –Format of the package in the visualisation software

The high byte of each path is sent first and the low byte follows. The stop byte of the package is “0XA5”. It is the identifier of the interrupt function in the COM callback function. The software is told that one package has been received when “0XA5” arrives. The working principle is shown below in the design flow chart in *Figure 83*.

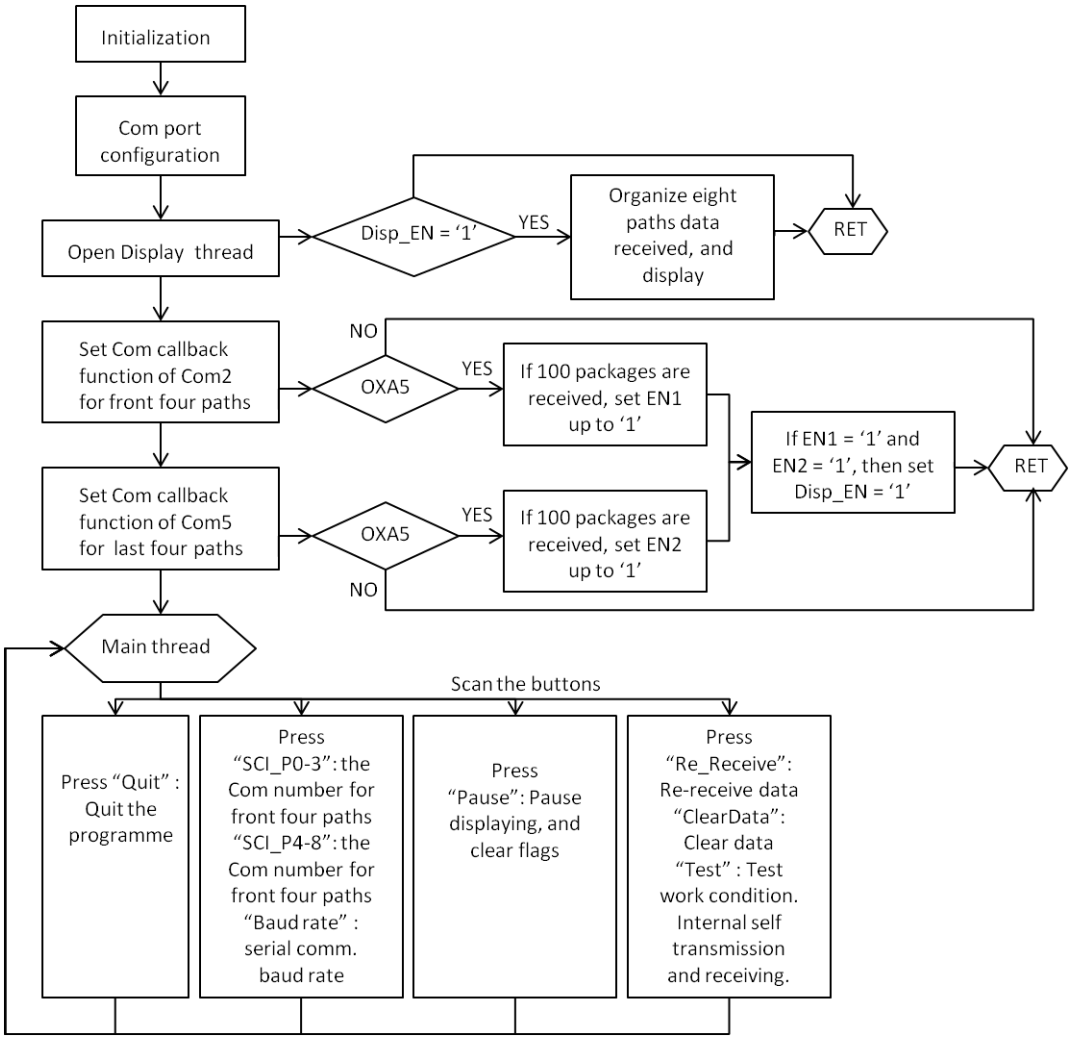


Figure 83 –Design flow chart of visual display programme

As shown in the flow chart, the programme keeps scanning the buttons. The user interface is shown below in *Figure 84*.

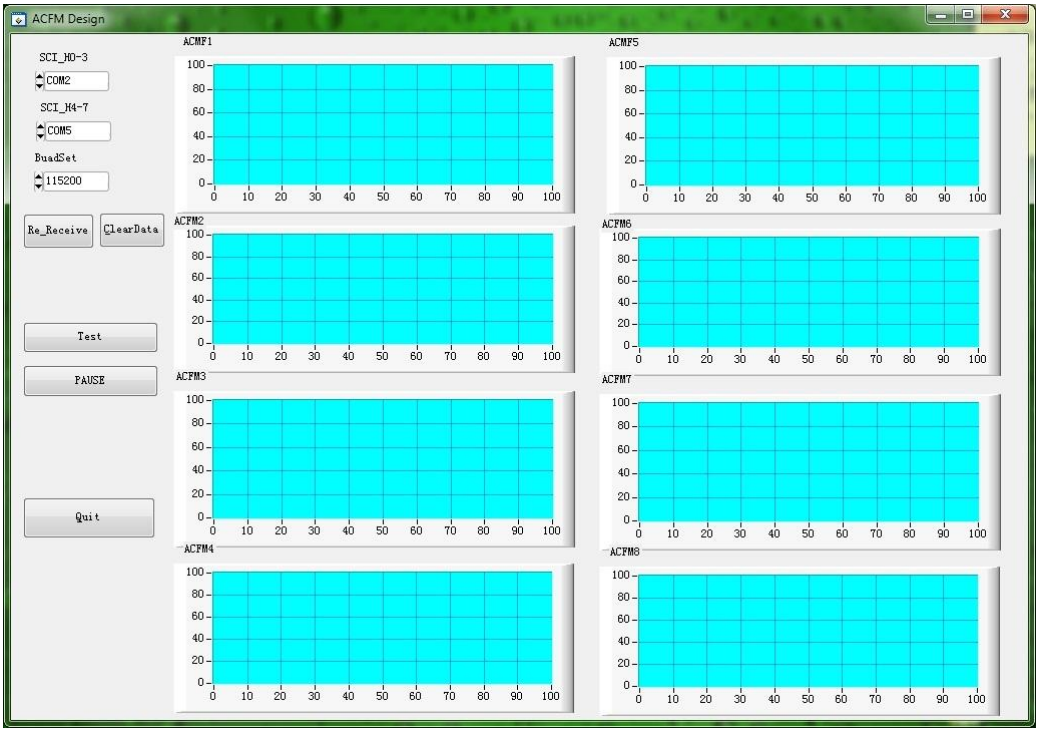


Figure 84 –User interface of visual display programme

In the com port configuration step, the serial communication protocol is set. Because the output signal "TxD_Com1" is assigned as COM2 and "TxD_Com2" is assigned as COM5, the configuration here is as shown above. 2-bit starting bits and a 1-bit stop bit are combined with 8-bit data. The format is shown below. Therefore, each byte actually consists of 10 bits' data.

Start bit	8-bit data	Stop bit

Figure 85 –Format of 11-bit frame

It can be seen in the chart that a display thread is called every 100 packages. As demonstrated in *Figure 85*, one package contains 10 bytes, so SCI transmitting one package needs:

$$10 \times 11 \times \frac{1}{460800} = 0.2\text{ms}$$

However, to perform a display one time requires 20 ms. Therefore the display unit is chosen every 100 packages, which meets the requirement of being greater than the display time. And

based on the 50 kHz input frames, and eight channels, the resolution is 500 Hz. The defects information can be inspected in the slow speed inspection train and particularly manual operation. If this was deployed on a high speed, the visualisation would change correspondingly.

Having described the visualisation software, the software can be connected to the FPGA board to view the curve of the eight paths' ACFM results.

7.1 FPGA & Visual display

In this test, the inputs are still generated by the simulated data generator and the input data is the same as the frames listed in Table 8. The frames are sent cyclically. Frame 3 is provided to the master path and Frame 4 is provided to the other seven slave paths. It is done in this way to simulate the different conditions on the whole track surface.

The curves of the eight paths are shown below in *Figure 86*.

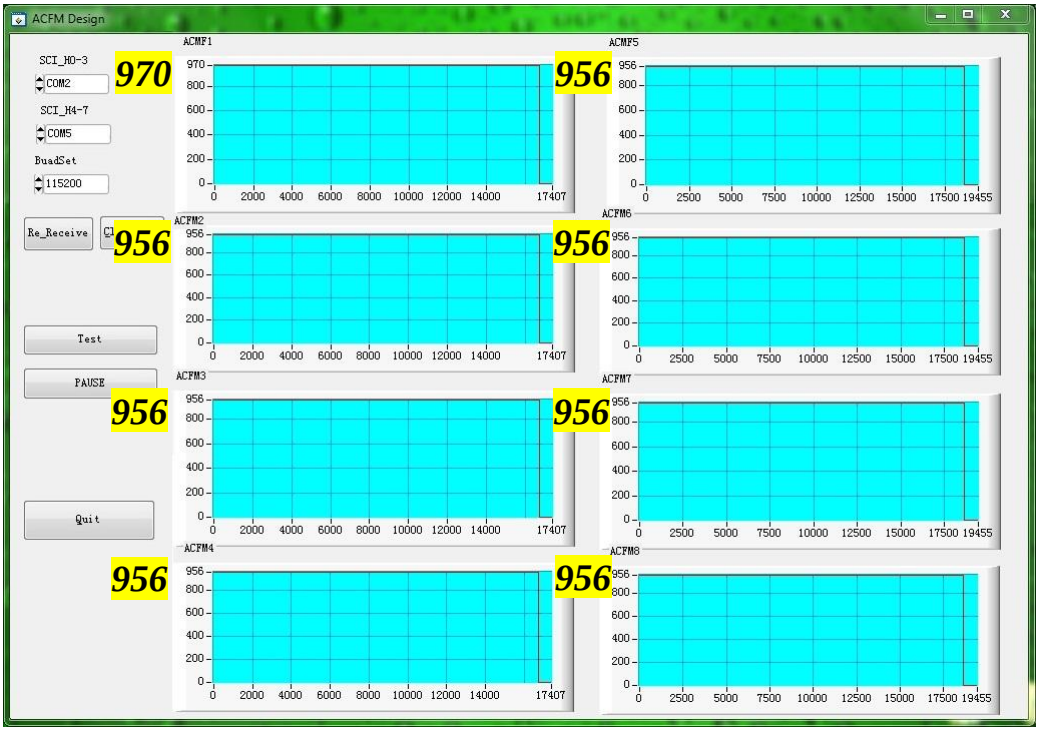


Figure 86 –Visual display of the eight paths' ACFM results

The ACFM results of the eight paths are shown above in *Figure 86*. The result of the master path is “970” and the results from the seven slave paths are “956”, as expected. If the value changes, it can be easily seen from the curves.

In the following test, the simulated crack inputs are set in the simulated data generator to imitate the crack condition on the track. The changing wave is shown below in *Figure 87*.

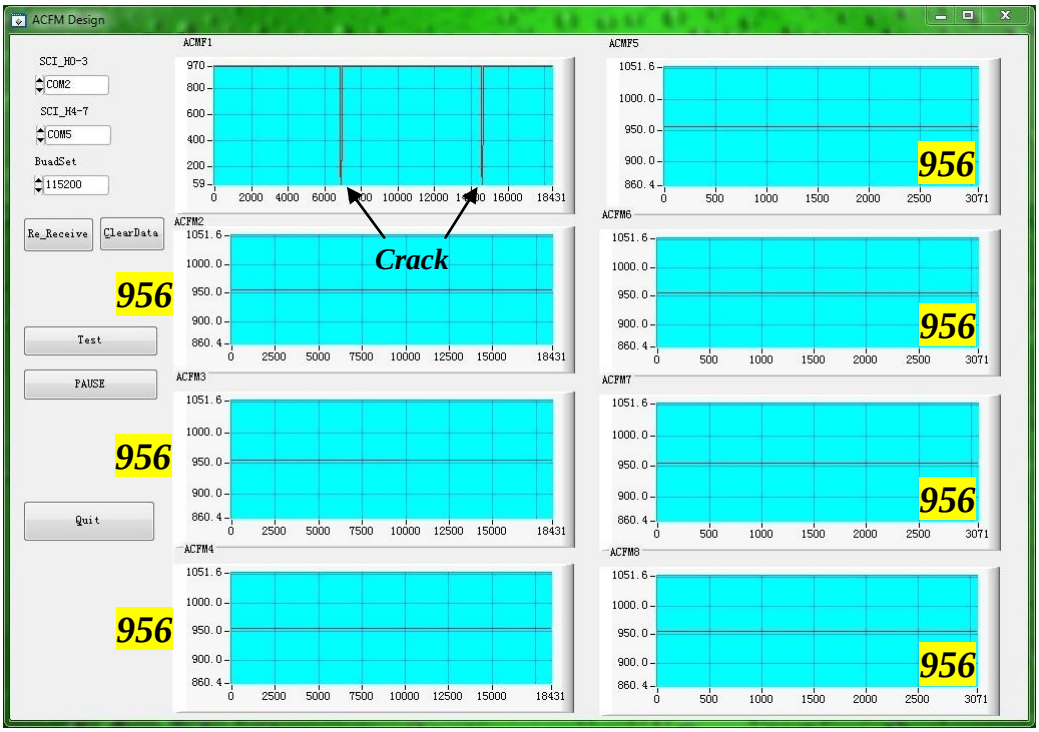


Figure 87 –Visual display of the eight paths' ACFM results with crack

As the curves show in the graph, the simulated crack occurs in the position of the probe which connects to the master processing path. Therefore there is a sharp impulse in the curve. Because the simulated data is sent cyclically, the impulse will be generated periodically as well.

Having developed and tested the visualisation software, the processing chain for the eight ACFM paths as well as the visualisation work on the PC have been realised. The system has implemented the mandatory requirements of the visualisation part which is:

Mandatory features

1) 8 channels' results visualisation.

However, the FPGA cannot communicate with the ACFM probes directly. It needs the DSP to play the role of real data acquisition. In order to expand the system a little further to verify the behaviour of it, the following task is to program the DSP to realise the eight paths data acquisition.

8 DSP for ACFM use

As an example simulated input method, only one DSP board is used in this test. The advantage of using this method is the simple structure compared with the eight DSP boards structure. The working principle of the one DSP board system will then be discussed.

The DSP board used is a DSP X28335 Kit Board, and the program environment is CCStudio version3.3. The function can generally be divided into ADC part and DAC part. The former part is used to sample analogue data collected from the track and send them to the FPGA, while the latter part is used to generate a carrier wave and send it to the ACFM probes. The detailed design is shown in the flow chart below in *Figure 88*.

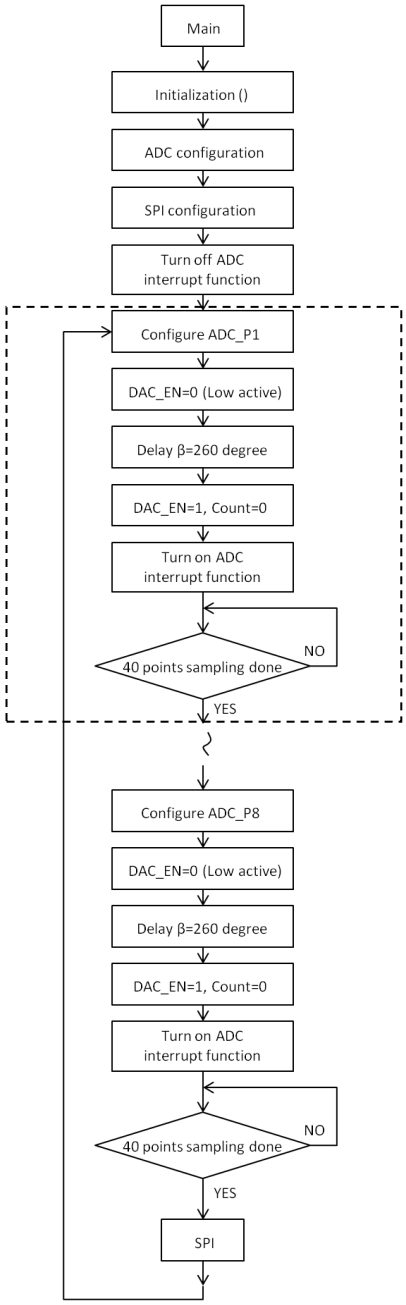


Figure 88 –Design flow chart of DSP programme

All the designs are in the main function, and the most important part in this design is in the dashed block. It is the synchronisation of the sampling and carrier wave in order to guarantee the sampling always begins at the initial phase in each period. The working principle is shown below in *Figure 89.*

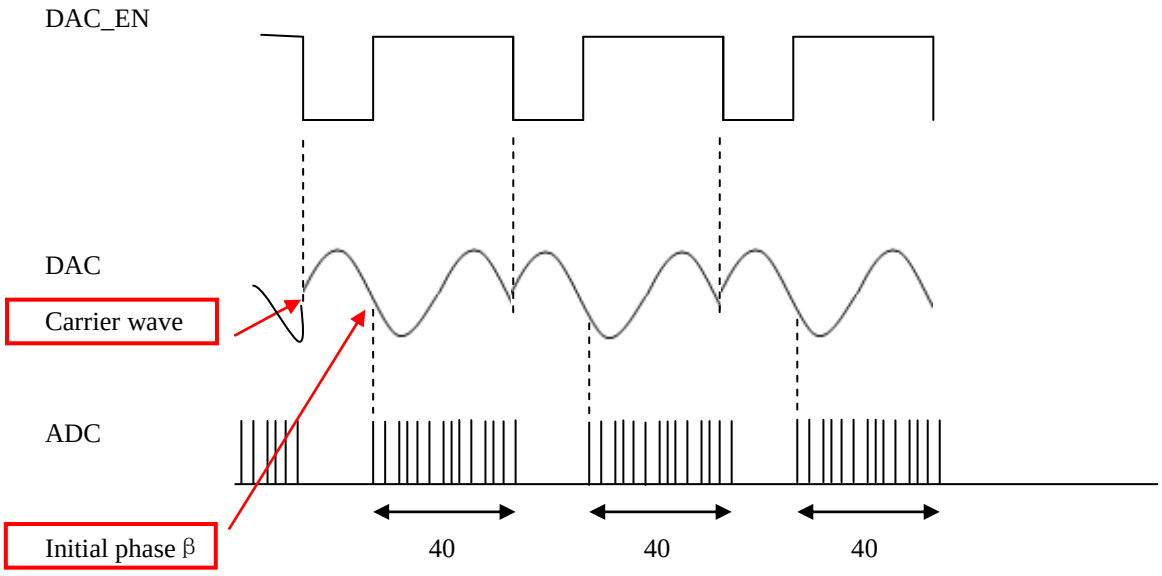


Figure 89 –Waveform of ADC sync. with DAC working principle

When the signal DAC_EN changes to zero, at the falling edge, the carrier wave is reset to the initial phase, which is set as zero ensuring the phase of carrier wave is the same at the beginning of each element sampling. In the meantime, the ADC waits until the sampling initial phase is reached then starts to sample. And after having sampled 40 points, the ADC stops working until the next DAC_EN signal falling edge arrives.

Having made clear the working principle of the DSP system, the test and verification should be considered.

8.1 DSP & FPGA

Before putting the data from the DSP into the processing chain in the FPGA, the output data of the DSP should be tested and verified.

The test is to build a straight path in the FPGA from the front SPI module to the last SCI module. The display method uses the visualisation software in LabWindows CVI and the serial communication programme in Matlab.

The 8-pin Pmod connector is used to perform the communication between the DSP, FPGA and PC. Two pins out from the 8-pin Pmod connector are used to send serial data to the PC.

Besides that, the other 6 IOs are used to perform the SPI communication between the DSP board and the FPGA.

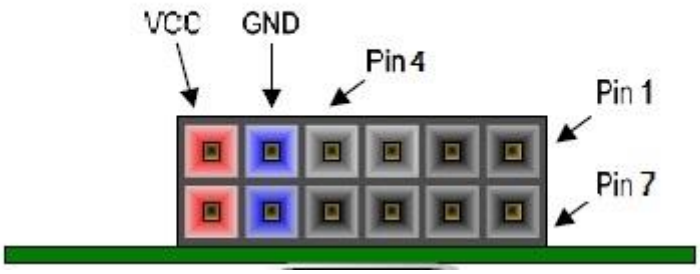


Figure 90 –The Pmod connector service condition

Figure 90 above demonstrates the service condition of the Pmod connector. The assignment of each pin is listed below:

- Pin1: TxD_Com1
- Pin2: TxD_Com2
- Pin3: MOSI (bit stream through SPI)
- Pin4: SCLK (SPI communication clock 6.25MHz)
- Pin5: SSEL (Chip select signal, falling edge enable)
- Pin6: Address0
- Pin7: Address1
- Pin8: Address2

Having done the connection work, the simulation of the DSP behaviour can start. The mimic 16-bit input data is “55AA” in hexadecimal representation and is sent cyclically.

The data through the straight path is captured by Matlab in the PC. The result is shown below in *Figure 91*.

	1
1	85
2	170
3	85
4	170
5	85
6	170
7	85
8	170
9	85
10	170
11	85
12	170
13	85
14	170
15	85
16	170
17	85
18	170
19	85
20	170
21	85
22	170
23	85
24	170

	1
1	85
2	170
3	85
4	170
5	85
6	170
7	85
8	170
9	85
10	170
11	85
12	170
13	85
14	170
15	85
16	170
17	85
18	170
19	85
20	170
21	85
22	170
23	85
24	170

*Figure 91 –Bit stream from "*TxD_Com1*"&"* TxD_Com2*" captured in Matlab*

The received 16-bit, which is 2 bytes shown in the figure, is "85" and "170". Transforming the two bytes into binary representation:

$$D"85" = B"01010101" = H"55"$$

$$D"170" = B"10101010" = H"AA"$$

So the 16-bit data is:

$$B"01010101\_10101010" = H"55AA"$$

The data captured by Matlab is exactly the same as the mimic data in the DSP. Therefore it can be concluded that the FPGA board, the ATLYS board, is working correctly.

8.2 DSP & FPGA & Visual display

Having tested and verified the behaviour of the DSP is correct, the output of the DSP can be put into the processing chain in the FPGA to verify the working condition of the whole system.

In this test, the inputs are still generated by the simulated data generator. The input data is the same as the frames listed in Table **8**. The frames are sent cyclically. The simulated crack input, which mimics the crack condition on the track, is provided to the master path and Frame 4 is provided to the other seven slave paths. This is to simulate the different conditions on the whole track surface.

In the following test, the simulated crack inputs are set in the simulated data generator to imitate the crack condition on the track. The changing wave exactly the same shown in *Figure 87*.

The programme in the DSP could accommodate all the other parameter configurations. However, parameter modification will result in setting changes for the whole system, such as the carrier frequency, sample frequency, etc. So if modification is needed in a future project, the system could easily accommodate a new system design.

The requirements of the system realised in this part is:

Mandatory features

1) The ADC in each channel runs at 1 MSPS.

Preferred features

1) The ADC in each channel runs at 2 MSPS.
2) The structure of the DSP system could be simplified from 8 DSPs to 1 DSP.
3) The system could be able to accommodate other parameter configurations, such as carrier frequencies, sample frequency, sampling starting phase, etc.

The system has therefore implemented both the mandatory requirements and the preferred requirements of the DSP part listed at the beginning of the dissertation.

9 Conclusion

In this project, the system is divided into three parts regarding the equipment: DSP device, FPGA device and PC equipment. Based on their functions, the whole system also can be divided into three parts: data acquisition, data processing and data visualisation. In this project, all these three parts are implemented. The mandatory and preferred requirements of the system have been realised.

The eight paths parallel processing chain of the ACFM data has been developed on an FPGA device. The ACFM data in the eight paths are collected from eight parallel positioned ACFM probes. In order to make the system more efficient and productive, one of the eight paths is set as the master path to calculate the constants in the parameters configuration; the master path delivers the constants to the other seven slave paths through the bus. Designing the system in this way can save a great deal of components in the FPGA chip, and thus achieve cost benefits. Based on this structure, synchronisation work is necessary in the system design. After the data processing in the FPGA device, the final ACFM results can be achieved.

The single DSP structure has also been realised. The ADC sample rate is set at 2 MSPS. 40 samples (12-bit) are acquired from the carrier wave, the frequency of which is 50kHz. Based on Nyquist theory, this number of samples could well characterise the reflected carrier wave which indicates the real condition on the track surface. Putting the header which contains the parameter configuration in the front, the 46 elements frame is formed. The communication between the DSP and the FPGA device is through an SPI interface, and the communication speed is set at 6.25 Mbps. As discussed in Section 2.4, the communication speed of SPI is one of the key parameters to achieve high speed specialty of the system. If the data from the eight paths which connects to eight ACFM probes are parallel transferred, with this speed, the time gap between each time sampling is on a millisecond level. Assuming the inspection train

is running at 100 km/h, the distance between the two adjacent sampling times is shorter than 1 mm, which more than satisfies real industry requirements. Even if the eight paths would expand, this part of the system is competent.

Another feature implemented to achieve high speed specialty of the system is serial communication through the RS-232 between the FPGA and PC. The visualisation software has been developed on the PC. The software is designed to provide a visual way for users to view the change of output results from the FPGA device, and find out where the cracks are located. The baud rate of the SCI communication between the FPGA and the PC is set at 1 Mbit/s, and realised by using two serial ports transmitting bit flow at speed of 460800 bps. In this project, one DSP with one SPI interface transferring at 6.25 Mbps is used to achieve a 1 Mbit/s baud rate, because the SPI communication for each path is one eighth of 6.25 Mbps. RS-232 communication is a widely used serial communication protocol, and most developing environments have the RS-232 function integrated. The normal baud rate used is from 300 bps to 115200 bps, but no specific maximum. In this project, the highest baud rate that could be called in the visualisation software is 460800 bps. And by assigning the SCI transmission work to two serial ports, the 1 Mbps baud rate could be achieved. If the transmission speed needs further enhancement, the RS-485 could be considered, which could achieve up to 35 Mbps. But as not many developing environments have RS-485 integrated, the developer may need to build a function in the terminal to receive the bit flow accordingly.

At this stage, experiments for this system are still in the lab. Once it is applied on the inspection trains, the system could allow a high speed automatic track inspection with an ACFM probes array located at a constant lift-off (distance between probe and track). Related researches in other major areas, such as mechanical and civil engineering, are also being carried out. Some of them are trying to solve how to weaken the lift-off vibration effect

during high speed inspection using ACFM probes. The final realisation of this goal still needs further work, but initial results are promising.

During the development of the system, many problems occurred, such as code debugging, timing issues, synchronisation work between the master and slave paths, etc. One problem worthy of mention concerns the SPI design. The SPI slave module is designed to capture data in the rising edge of the SCLK. However, it could not capture the data correctly as the signal on the MOSI does not settle down at the rising edge of the SCLK. In order to solve this problem, the SPI slave module was modified to capture the data on the falling edge of the SCLK. This is because the signal at the MOSI pin is much more stable during the falling edge of the SCLK.

A great deal of knowledge has been gained and techniques learned from this project, including familiarity with VHDL language, Xilinx FPGA chip programming, DSP programming and debugging techniques, in particular the scientific method of dividing a complex system into small parts to debug separately. A future improvement of the system would be to integrate the function realised in the DSP part into the FPGA. The structure could then be simpler and the internal processing speed of the FPGA much faster. Also, the performance of the visualisation on the PC could be enhanced by integrating the curves of the eight probes into one curve using an appropriate operator. Identification of the cracks would then be more accurate and much more user friendly. If all this work were to be implemented, high speed automatic rail inspection could be put into industrial use in the near future.

10 References

Anami, T. (2004). 'Preventive rail grinding strategy adopted on Shinkansen lines of JR East'. *Rail Eng. Int.* vol. 2. pp. 10-12.

Berkeley Design Technology, I. (October 2002). 'Comparing FPGAs and DSPs for Embedded Signal Processing'. Stanford University. Berkeley, California.

Board, T.i.I. (2007-2008). 'National Rail Trends'. *Office of Rail Regulation*. July. p. 245.

Bray, D.E. (2000). 'Historical review of technology development in NDE'. In Proceedings of the 15thWorldConference on. Roma.

Brizuela, J., Ibanez, A. and Fritsch, C. (2010). 'NDE system for railway wheel inspection in a standard FPGA'. *Journal of Systems Architecture*. August. pp. 616-622.

Brizuela, J., Ibanez, A. and Fritsch, C. (2010). 'NDE system for railway wheel inspection in a standard FPGA'. *Journal of Systems Architecture*. pp. 616-622.

Burd, J. (April 2005). 'Magnetic flux leakage old and new, magnetics in non-destructive testing'. One day seminar proceedings of magnetics in non-destructive testing. 8-13.

Cannon, D.F. (2003). 'An international cross reference of rail defects, 2nd edition'. *UIC Rail Defect Management Report*. June.

Clark, R. (2004). 'Rail flaw detection: overview and needs for future developments'. *NDT&E International*. vol. 37. pp. 111-118.

Clark, R. (2004). 'Rail flaw detection: overview and needs for future developments'. *NDT&E Int.* no. 37. pp. 111-118.

Dover, W.D., Charlesworth, F.D.W., Taylor, K.A., Collins, R. and Michael, D.H. (1981). *The use of A-C Field Measurement to Determine the Shape and Size of a Crack in a Metal*. 1st edition. George Birnbaum and George Free: Eddy-Current Characterization of Materials and Structures.

Drury, J. and Pearson, N. (April 2005). 'Corrosion detection in ferrite steels using magnetic flux leakage'. One day seminar proceedings of magnetics in non-destructive testing. 14-19.

Ferreira, L. and Murray, M. (1997). *Modelling rail track deterioration and maintenance: current practices and future needs.* 173rd edition. Not known: Transp. Rev.

Gao, L.-n. and Teng, L. (2008). 'Spaceborne digital signal processing system design based on FPGA'. *IEEE*.

Grant, P.M. (1993). 'Digital signal processing part1: digital filter and the DFT'. *Electronics and communication engineering journal*. February.

Grassie, S.L. (2005). 'Rail corrugation: advances in measurement, understanding and treatment'. *Wear*. no. 258. pp. 1224-1324.

Grassie, S.L. (2005). 'Rolling Contact Fatigue on the British railway system: treatment'. *Wear*. no. 258. pp. 1310-1318.

Herring, A. (1877). *Mode of detecting defects in railroad*.

Howitt, M. (2002). 'Bombardier brings ACFM into the rail industry'. *Insight*. vol. 44. no. 6. June.

Ireland, R.C. and Torres, C.R. (2005). 'Limitations of the Circumference MFL technique in the NDE of pipelines'. Magnetics in Non-destructive Testing Seminar. London, UK.

Jones, N.B. and Watson, J.D.M. (1990). 'Digital signal processing: principles, devices and applications'. *IET*.

Junger, M., Thomas, H.M., Krull, R. and Rühe, S. (2004). 'The potential of eddy current technology regarding railroad inspection and its implementation'. the 16thWorld Conference onNon-DestructiveTesting. Montreal.

Kube, K. (2005). 'Sperry trucks track troubles deep inside rails'. *TrainsMag.* February. pp. 20-21.

Lewis, A., Micheal, D., Lugg, M. and Collins, R. (1988). 'Thin-skin electromagnetic fields around surface-breaking'. *J Appl Phys*. vol. 64. no. 8. pp. 3777-3784.

Li, Y., Tian, G.Y. and Ward, S. (2006). 'Numerical simulation on magnetic flux leakage evaluation at high speed'. *NDT&E International*. vol. 39. pp. 367-373.

Lugg, M. and Topp, D. (2006). 'Recent developments and applications of the ACFM insoection method and ACSM stress measurement method'. In the Proceedings of ECNDT 2006. Berlin, Germany.

M Ph Papaelias, C.R.C.L.D.B.B.M.L. (2009). 'High-speed inspection of rolling contact fatigue in rails using ACFM sensors'. *The British Institute of Non-Destructive Testing (Insight)*. no. 7. July.

Ma, K. (2010-2011). 'DEVELOPMENT OF AN FPGA (FIELD PROGRAMMABLE GATE ARRAY) PROCESSING CHAIN FOR HIGH-SPEED NDT (NON-DESRUCTIVE TESTING) SYSTEMS'. *Msc final project report (intenal EECE/University of Birmingham, available upon request).*

Mandayam, S., Udpa, L., Udpa, S.S. and Lord, W. (1996). 'Invariance transformations for magnetic flux leakage signal'. *IEEE Trans. Magn.* vol. 32. pp. 1577-1589.

Nair, P.P. (2008). 'Image and video processing using FPGA technology for medical application'. *IET*. pp. 1-7.

Nicholson, G.L. and Davis, C.L. (2012). 'Modelling of the response of an ACFM sensor to rail and railwheel RCF cracks'. *NDT&E International*. no. 46. December. pp. 107-114.

Nicholson, G.L., Rowshandel, H., Hao, X.J. and Davis, C.L. (2013). 'Measurement and modelling of ACFM response to multiple RCF cracks in rail and wheels'. *Ironmaking & Steelmaking*. vol. 40. no. 2. pp. 87-91.

Office of rail regulation; (2006). 'Train Derailment at Hatfield: A final Report by the Independent Investigation Board'. July. p. 245.

Papaelias, M., Márquez, G.F.P., Muñoz, C.J.M. and Roberts, C. (2008). 'A B-Spline approach to alternating current field measurement for railroad inspection'. The International Conference of Industrial Engineering and Engineering Management. Singapore. 3.

Papaelias, M., Robert, C. and Davis, C.L. (2008). 'A review on non-destructive evaluation of rails'. *state-of-the- art and future development*. May.

Papaelias, M., Robert, C., Davis, C.L. and Lugg, M.C. (2009). 'High-speed inspection of rail using ACFM techniques'. *NDT&E International*. pp. 328-335.

Papaelias, M., Roberts, C., Davis, C.L., Blakeley, B. and Lugg, M. (2009). 'High-speed inspection of rolling contact fatigue in rails using ACFM sensors'. *Insight-Non-Destructive Testing and Condition Monitoring*. vol. 51. no. 7. pp. 366-369.

Papaelias, M.P., Roberts, C., Davis, C.L., Blakeley, B. and Lugg, M. (2010). 'Further developments in high-speed detection of rail rolling contact fatigue using ACFM techniques'. *Insight-Non-Destructive Testing and Condition Monitoring*. vol. 52. no. 7. July. pp. 358-360.

Petcher, P.A., Burrows, S.E. and Dixon, S. (2013). 'Shear horizontal (SH) ultrasound wave propagation around smooth corners'. *Ultrasonics*. pp. 997-1004.

Pohl, R., Krull, R. and Meierhofer, R. (2006). 'A new eddy current instrument in a grinding train'. ECNDT 2006. Berlin, Germany.

Rosula, S.R., Carlos, M.O., Jose, C.N.M., Noel, S.P., Raphael, A.G. and Jovilyn Therese, B.F. (2008). 'FPGA-based digital signal processing trainer'. *IEEE*.

Schöch, W. (2004). 'Combating rail surface fatigue in Europe by head check grinding'. *Rail Eng. Int.* vol. 1. pp. 6-8.

Schöch, W. and Heyder, R. (2003). 'Rail surface fatigue and grinding: exploring the interaction'. the 6th International Conference on contact mechanics and wear of rail/wheel systems. Gothenburg.

Schöch, W., Heyder, R. and Grohmann, H.D. (2006). 'Contact geometry and surface fatigue guidelines for appropriate rail maintenance'. the 7th International Conference on contact mechanics and wear of rail/wheel systems. Brisbane.

Thomas, H.-M., Heckel, T. and Hanspach, G. (2006). 'Advantage of a combined ultrasonic and eddy current examination for railway inspection trains'. In: the Proceedings of ECNDT 2006. Berlin, Germany.

Thomas, H.-M., Junger, M., Hintze, H., Krull, R. and Rühe, S. (2000). 'Pioneering inspection of railroad rails with eddy currents'. the 15th World Conference on Non-Destructive Testing. Rome.

Topp, D. and Smith, M. (2005). 'Application of the ACFM inspection method to rail and rail vehicles'. In the proceedings of ECNDT 2005. Barcelona, Spain.