ADSL / VDSL Line Simulation

A Feasibility Study and Initial Design

Andrew Wilkinson

awilkinson@totalise.co.uk

http://www.users.totalise.co.uk/~awilkinson/index.html

MSc Telecommunications

August 1999

Dr. Phil Lane

UNIVERSITY COLLEGE LONDON

Abstract

This dissertation describes an MSc project to investigate the feasibility of building a digital hardware line simulator for use as ADSL and VDSL test environments during modem development. The simulation requirement interms of crosstalk, noise and the physical line effects of twisted copper access pairs at DSL frequencies are determined. A simulation model using both time and frequency domains is brought forward with suitable DFT parameters calculated. Modelling of the physical line's attenuation and phase shift through frequency domain filtering is developed. Two practical ADSL line simulator implementation paths using DSP and FPGA devices are evaluated according to five performance and development metrics. Initial high level and revised logic level designs are undertaken using Xilinx FPGAs and FFT modules. Functionality is tested and performance evaluated using Xilinx Foundation software. Finally, future adaptability to VDSL line simulation is discussed. 44 references included.

Preface

The aim of this MSc project was to investigate the possibility of constructing a compact PC controlled hardware line simulator suitable for replacing the traditional twisted copper pair access cable as a test environment for ADSL and VDSL modems under design. The dissertation covers a wide range of subjects, ranging from investigation of the copper pair at DSL transmission frequencies, through to an in depth low-level initial design based on Xilinx FPGAs.

The main areas of the dissertation cover:

- The behaviour of the twisted copper pair at DSL frequencies in chapters 2 and 3.
- Simulation requirements and suitable signal processing methods in chapters 4 and 5.
- ANSL DMT ADSL signal modelling using Matlab in chapter 6.
- High level designs for ADSL simulators comparing DSP and FPGA solutions in chapter 7.
- A revised high level design using FPGA processing blocks in chapter 8.
- In depth low-level logic circuitry and functionality testing of the revised design using the latest Xilinx FPGAs in chapter 9.
- Possible extension to VDSL line simulation in chapter 10.

The low-level logic designs in chapter 9 assume the reader is conversant in designing Xilinx FPGAs using the Foundation design environment and includes complex logic designs without step by step explanation of their operation which is secondary to the aim of the project and dissertation. Included on the CD ROM are all Foundation schematic, simulation and waveform files associated with all the circuits of chapter 9.

In addition to the project design files for use in the Xilinx Foundation environment, the CD ROM also includes full data sheets in PDF format of all devices mentioned, as well as pertinent DSL tutorials from various sources and the latest Xilinx Foundation upgrade files necessary for the design. In order to compile and run simulations on the project designs, a minimum of 128 Mbytes of RAM is required, otherwise excessive hard disk activity occurs with the associated extended processing times.

Contents

List of Figures

Chapter 2

Chapter 3

Chapter 4

Chapter 5

Chapter 6

Chapter 7

Chapter 8

Chapter 9

Chapter 10

List of Tables

Chapter 3

Chapter 4

Chapter 7

Chapter 8

Chapter 9

Chapter 10

Chapter 1

Introduction

Only a few years ago, even the most enlightened telecommunication's engineer might ask:

"Is there a need for a hardware simulator to model a fixed access twisted copper pair?"

For voice, fixed line access is a mature technology. Ever since telecommunication networks employed switching, subscribers have been connected to the network operator's closest switch via an access network, usually consisting of a single twisted copper or aluminium pair of wires. In the UK alone, the incumbent telco, BT, has over 36 million subscriber pairs. It is therefore reasonable to question if there is a need for a hardware system to simulate the behaviour of a twisted copper pair. Such a widely deployed, mature technology, should be fully understood? Yes and no. The key to this question is bandwidth. Access technology over the last century has provided a perfectly adequate 4 kHz low bandwidth service dedicated to analogue voice conveyance. Since the mid 1980s, personal computers have increasingly pervaded residential homes. Initially, home computing was informationally isolated from the outside world, with data transfer only through removable magnetic media. The 1990s witnessed a major change in the way in which home computers were used. The widespread growth of the 'Internet' and particularly the development of Graphical User Interfaces (GUIs) such as Netscape and Microsoft's Internet Explorer fuelled a revolution in the way home users viewed their PCs. These developments projected the PC from a useful and entertaining novelty almost to the point of a 'must have' gateway to the outside world.

The increase in CPU speeds and the reduction in volatile and permanent memory storage costs have stretched the PC's capacity from manipulation of essentially text-based information to the arena of 'multimedia'. Whereas text-based data transfers can be satisfied by low bit rate transmission, multimedia applications such as real time video transfer and other interactive services require high bit rates for an acceptable Quality of Service (QOS). Even non-real time multimedia data transfers of items such as pictures require high transmission bit rates to give acceptable download times for end users. Consider a large e-mail of several hundred words, approximately 1 kilobyte in size. Using a standard

56 kbit/s modem, the e-mail could be downloaded within a second. Now consider the same e-mail with an enclosure of a single high quality 1 Megabyte bit map image. Using the same modem, the e-mail could take more than two and a half minutes to download!

The existing access network provided the logical choice to connect PCs to the outside world, commonly referred to as connecting to the Internet. However, this choice was driven mainly by cost and convenience – a computer engineer would never have opted for a 4 kHz band limited analogue drop cable to the nearest digital switching unit given a blank piece of paper! Although not ideal, by using multilevel Quadrature Amplitude Modulation (QAM) signalling, modem engineers have managed to squeeze transmission rates upto a maximum of 56 kbit/s through the analogue access network with no modifications to its structure. 56 kbit/s provides fast enough transmission for most text based applications, but not multimedia, hence methods were needed to provide higher bit rates.

One solution for providing home users with greater data rates is to digitise the access loop. The Integrated Services Digital Network offers data rates of upto 144 kbit/s, with 128 kbits/s reliably available to the end user. ISDN is a great improvement over analogue modem systems, especially since it offers a reliable service - the data rate offered by all ISDN lines is always 128 kbits/s, whereas analogue modems rarely achieve their maximal throughput and what is provided is unreliable in that it may vary during the holding time of a data connection. Although an improvement, 128 kbits/s isn't really enough for multimedia applications and is an expensive solution considering the gain in bit rate.

Both ISDN and analogue QAM modems use baseband transmission. DSL modems employ passband transmission by modulating data onto carrier frequencies much higher than the spectrum occupied by analogue voice signals. The expansion of the Power Spectral Density (PSD) mask over baseband transmission introduces new behaviour which DSL modems must accommodate. Classically, twisted copper access pairs used in voice telephony are fully characterised by their transmission performance and signalling resistance. The transmission performance, measured in dB, is the maximum signal attenuation before the level of extraneous noise becomes unacceptable to the customer. On residential lines a figure of 10 dB is a common upper limit. The signalling resistance is the resistance seen by signalling (dial tone, ringing etc). The maximum allowable signalling resistance is of the order of a kilo Ohm. Over the broader utilised spectrum, the line's frequency response exhibits much greater variation and with higher frequency operation, the effects of crosstalk from other high bit rate lines becomes very significant and prevents the line from being viewed in isolation.

The susceptibility to crosstalk at higher frequencies is the core driver for a DSL line simulator. DSL modem designers such as Fujitsu have difficulty in recreating a realistic test line for evaluating their equipment. To test baseband modems operating to a few tens of kHz, coiled reels of access cables are used which provide an adequate test environment because the signals are affected mainly by the lines low frequency transmission performance. When differential transmission with twisted copper pairs is employed the crosstalk from neighbouring access pairs within the same bundle is negligible. For voice band QAM modems, the transmission rate is predominately limited by noise and insertion loss. Essentially the line under test can be viewed in isolation from similar neighbouring call carrying lines. Coiled bundles of access pairs cannot be used for testing DSL modems as the close coiling of the same line induces self crosstalk, so designers are forced to use long runs of stretched out cable in a fashion similar to real life to avoid self self-crosstalk due to the signal under test. Additionally, crosstalk from

neighbouring DSL lines is significant and is often the limiting factor to the modem's achievable bit rate. With a physical test line, a designer must inject interfering DSL signals along other copper pairs within the same access bundle in order to recreate a realistic test environment. On a real physical access bundle, this is both difficult and expensive to do and time consuming to alter the test environment.

A compact PC based line simulator that simulates both the physical line characteristics throughout the transmission spectrum and the effect of crosstalk from neighbouring DSL lines within the same theoretical access bundle is therefore very desirable. Significantly, once the test environment is under PC control it can be instantly reconfigurable both interms of the physical line characteristics and crosstalk.

The aim of this project is to investigate both the technical and commercial feasibility of making a line simulator for use with Very high bit rate Digital Subscriber Line (VDSL) and Asymmetrical Digital Subscriber Line (ADSL) modems.

Chapter 2

The Twisted Copper Pair

Twisted pair channels are noisy, lossy and prone to crosstalk. Both ADSL and VDSL modulation schemes are designed to overcome or reduce the effects of these impairments. Before any simulator design can be considered, a thorough understanding of the transmission environment in which ADSL and VDSL modems operate is essential. Due to the extended PSD of both modulation schemes into the MHz region, both physical line characteristics and crosstalk susceptibility are considerably different to that experienced for voice band communication on the same channel. This chapter will briefly look at physical line characteristics of twisted copper pairs into the MHz region and the nature of crosstalk specific to ADSL and VDSL systems.

2.1 Physical Line Characteristics

It is important to realise that access loops between a LEX and a customer's premises generally do not consist of a single long uniform conductive pair, but rather of a series of spliced lengths of pairs increasing in diameter towards the subscriber[1]. Due to this, the insertion loss of the access loop is generally not monotonically increasing with reach towards the subscriber and is also subject to abrupt splicing losses in addition to the cumulative losses upto that point. The nature of the access loop must be considered for future work in developing models for software to drive a hardware simulator.

Any transmission line can be described in terms of its physical characteristics of resistance, inductance, capacitance and admittance, known as the primary RLCG parameters. For the lowpass voice spectrum of 4 kHz, RLCG parameters are constant for normal gauge access pairs such as 24 and 26 gauge wire. At higher frequencies above a hundred kHz or so, the pair's resistance, inductance, capacitance and admittance vary significantly with frequency

2.1.1 RLCG Characteristics at Extended Bandwidths

Figure 2.1 shows the typical RLCG characteristics of a 1000 thousand foot (1 kft) twisted copper pair of plastic insulation coated (PIC) 24 and 26 gauge wire, generated by Matlab using cubic spline interpolation to published data[2]. The variation in resistance, inductance and admittance is clearly seen at frequencies above 100 kHz. Although primary RLCG parameters give an insight into the basic behaviour of a twisted copper pair at higher frequencies, two more useful transmission line parameters are defined. The propagation constant and characteristic impedance of the line enable the voltage and current at any point along the line and hence the insertion loss and phase shift due to any length of twisted copper pair to be determined.

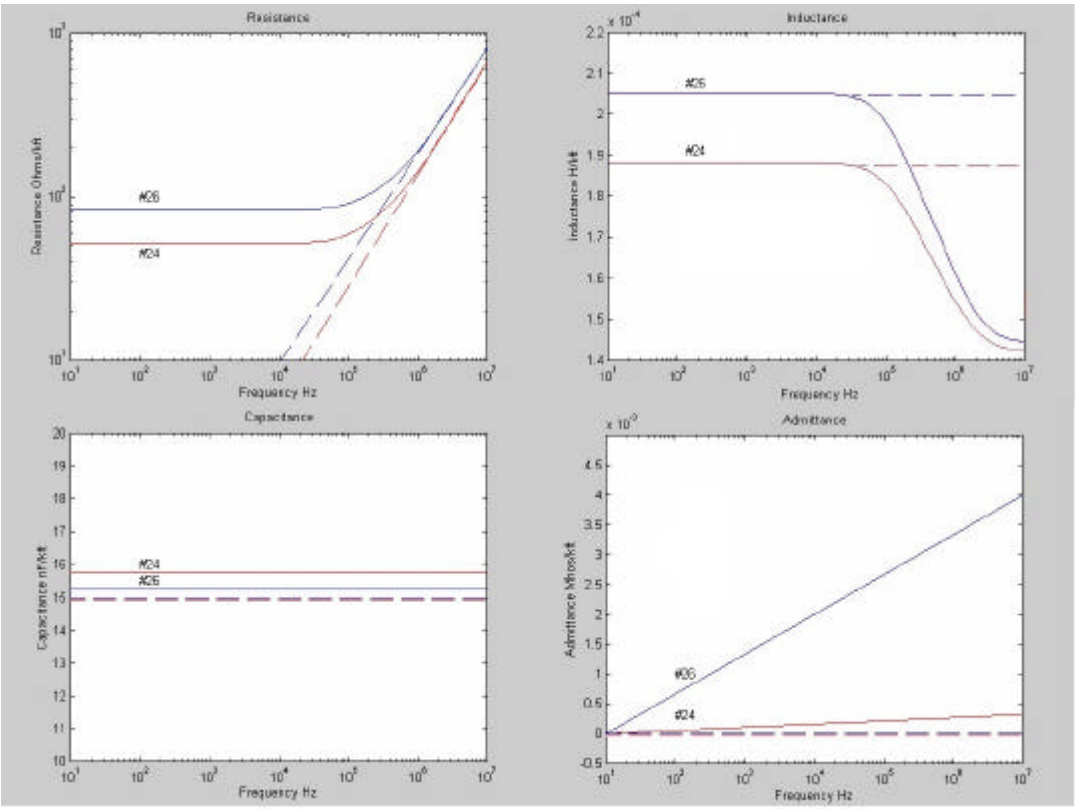


Figure 2.1 Primary PIC twisted pair parameters

2.1.2 Propagation Constant and Characteristic Impedance

Any termination design will attempt to match the terminating impedance with the line impedance for maximum possible power transfer to the receiver. For a matched line and termination impedance, the voltage at any point ‘x’ from the source is given by

$$V(f,x) = V_0(f)e^{-x\gamma f}$$

In terms of a transfer function from source to termination,

$$H(f) = \frac{V(f,x)}{V_0(f)} = e^{-x\gamma f}$$

The γ term is the propagation constant and is related to the primary line characteristics by

$$\gamma = \sqrt{(R + j\omega L)(G + j\omega C)}$$

From the primary line parameters, below 100 Mhz $j\omega C >> G$, therefore the propagation constant can be approximated to

$$\gamma = j\omega\sqrt{LC}\sqrt{\left(\frac{R}{j\omega L} + 1\right)}$$

Since $j\omega L > R$ in the spectrum of interest,

$$\sqrt{\left(\frac{R}{j\omega L} + 1\right)} \approx \left(\frac{R}{j\omega L} + 1\right)$$

Thus

$$\gamma = R\sqrt{\frac{C}{L}} + j\omega\sqrt{LC}$$

The resistance increases proportionally to the root of frequency due to the skin effect and the line capacitance is constant. Although the pair's inductance varies approximately 30% between 100 kHz and 10 MHz, if as a first approximation it is assumed to be constant, the frequency dependence of the propagation constant is given by:

$$\gamma = \alpha_0 f^{1/2} + \beta_0 f$$

Where the constants α_0 and β_0 are related to the primary line parameters. C, L and R.

2.1.3 Insertion Loss

The main reason for deriving a theoretical expression for the characteristic impedance in terms of the propagation constant is to determine what behaviour a DSL line simulator must simulate due to the physical line effects. The line's transfer function can be considered in terms of magnitude and phase, allowing the insertion loss in decibels and the associated phase shift for sinusoids throughout the relevant DSL spectrum to de determined. Over a typical range of access spans, figures 2.2 and 2.3

show the analytic insertion loss and associated phase shift derived above, for 26 gauge twisted copper pairs up 10 MHz. All graphs were generated by Matlab using the primary RLCG parameters shown in figure 2.1 (see appendix1 for further details).

As will be discussed later, the simulator's required DFT resolution is critically dependent on the range of the variation of insertion loss and phase over the DSL bandwidth.

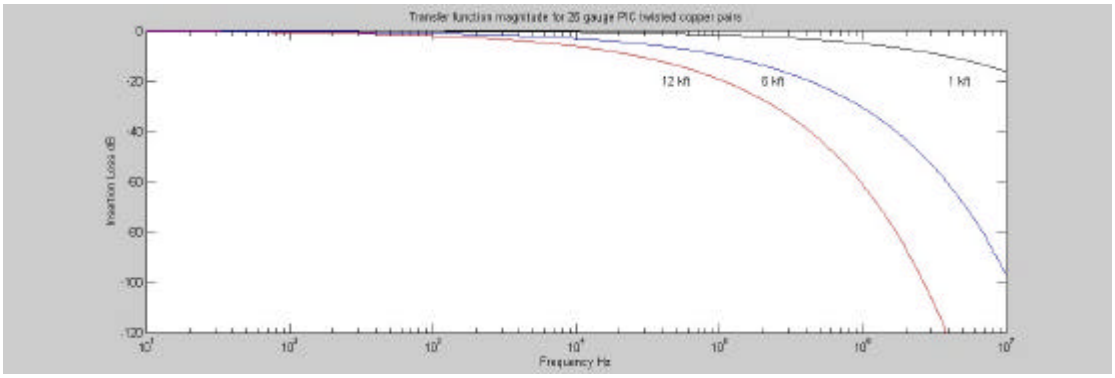


Figure 2.2 Insertion loss for 26 gauge PIC twisted copper pair

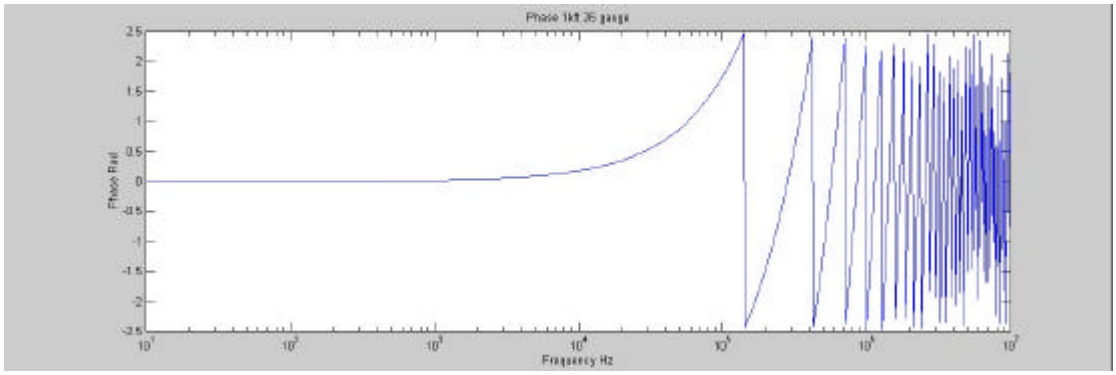


Figure 2.3 Phase shift for 26 gauge PIC twisted copper pair

As can be seen from the bode plot of phase, there is zero phase shift over the voice band, but above 100 kHz the phase response changes rapidly. Figure 2.4 shows linear plots of phase response over the ADSL bandwidth of 1 MHz for the proposed extremes of ADSL system reach.

Although these graphs are derived from approximations, experimental work[3] indicates the analytic results adequately model the access line's insertion loss and phase response over the ADSL bandwidth.

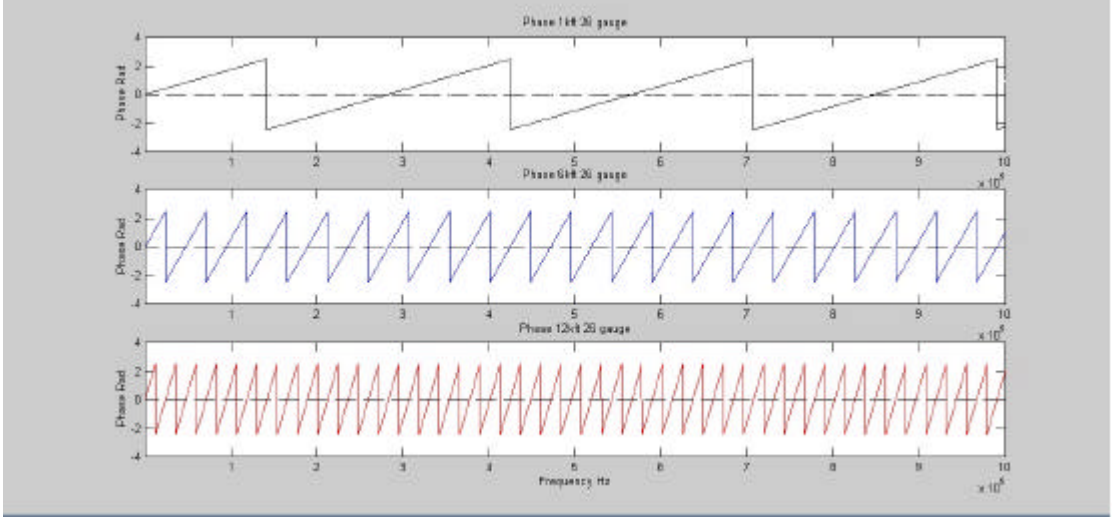


Figure 2.4 Phase shift for 26 gauge PIC twisted copper pair over ADSL bandwidth

2.2 Crosstalk in ADSL and VDSL Systems

Alexander Graham Bell patented the first use of twisted copper pairs in 1881 to alleviate the effect of crosstalk between neighbouring wires within an access bundle. Until this time, multiple conductors with a single shared ground line were common. Unacceptable levels of crosstalk in access bundles of only a few hundred metres in length were often encountered. Later, this type of scheme was termed common mode signal transmission where the driving signal is applied between the common ground and conductor. As such, any nearby signal carrying conductor, called a disturbing source, electro-magnetically induces voltages in other signal carrying wires differently to that induced in the physically different and spatially separated common ground. Clearly at the access termination, the earpiece, a potential difference will exist between the ground and conductor due to the disturbing signal – crosstalk. The twisted copper pair operates in differential mode where only a difference in potential between the two conductors causes a current to flow. There is essentially no difference in the em field from the disturber, but the two conductors are now subject to the same disturbance so they will experience an identical common change in their potential relative to some absolute ground but not to each other. At the earpiece termination there is no difference in potential across the pair, so no power is dissipated in the earpiece due to the disturber. The disturbing crosstalk has been rejected by the differential operation of the pair of wires. In other words, twisted copper pairs give rise to common mode rejection of disturbing signals. Figures 2.5 and 2.6 show common mode and differential operation of transmission lines.

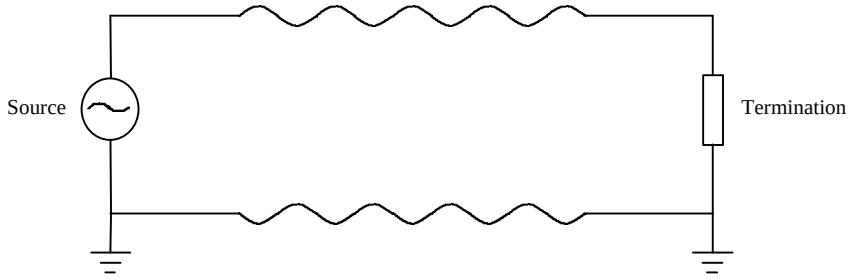


Figure 2.5 Common mode operation

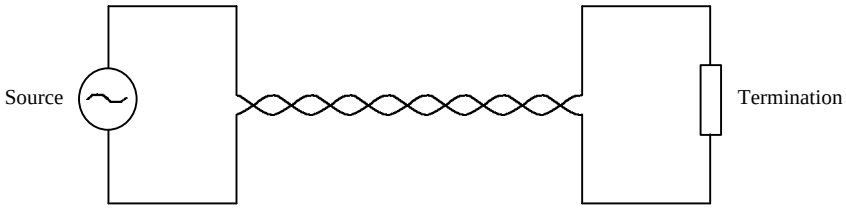


Figure 2.6 Differential mode operation

If twisted pairs reject common mode signals doesn't that imply crosstalk is eliminated at any frequency? Unfortunately due to the phenomena of capacitive and inductive inbalance[4] between different pairs along the length of the access bundle, differential crosstalk still occurs in twisted copper pairs.

Although the actual method through which crosstalk is induced is more important when considering how to generate model crosstalk for use in a simulator rather than directly in the design of the simulator hardware itself, some basic understanding is required inorder to adequately provision a simulator to copy crosstalk induction on twisted copper pairs. In any discussion on crosstalk, several definitions of different types of crosstalk are made which include:

- Self crosstalk from similar access signals (e.g. ADSL crosstalk to ADSL lines)
- Foreign crosstalk from different access signal (e.g. ISDN crosstalk to VDSL lines)

Both self and foreign crosstalk can be caused by sources located at

- the same end of an access bundle, termed Near End CrossTalk (NEXT)
- the opposite end of an access bundle, termed Far End CrossTalk (FEXT)

Figures 2.7 and 2.8 show NEXT and FEXT.

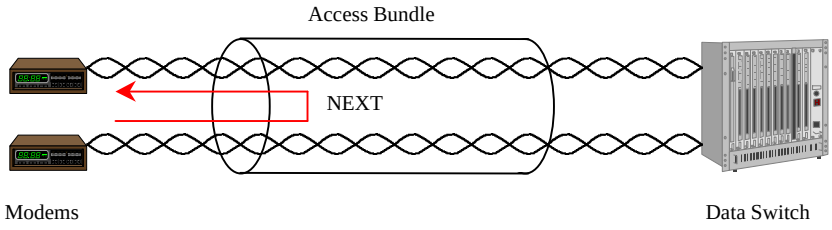


Figure 2.7 Near End CrossTalk

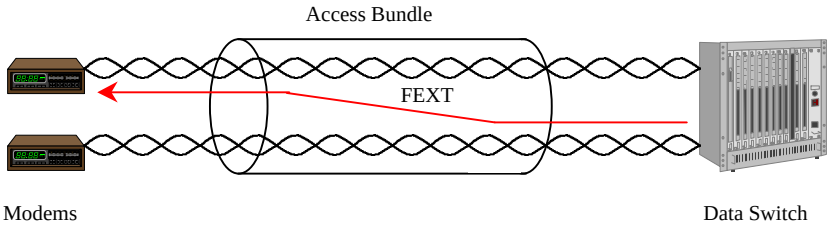


Figure 2.8 Far End CrossTalk

The effect of crosstalk on different modulation schemes is dependent mainly on the PSD relationship between the disturbed and disturbing signals[5], discussed further in subsequent chapters. To clarify this, consider the case of mythical ZDSL and YDSL disturbers inducing crosstalk in a QDSL signal which have the PSDs shown in figure 2.9.

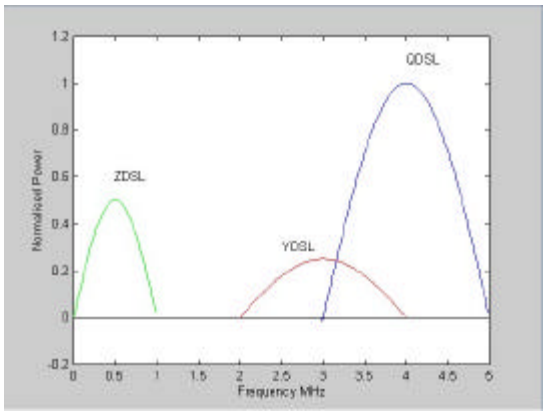


Figure 2.9 Mythical DSL power spectra

The PSDs of the QDSL and ZDSL signals don't overlap, therefore in the absence of non-linear intermodulation distortion and perfect out of band noise rejection, crosstalk from the ZDSL disturber will have no affect on the Bit Error Rate (BER) of the QDSL signal after demodulation. This is in contrast to the PSDs of the QDSL and YDSL signals, which show significant overlap. Crosstalk from a YDSL signal will have a significant impact on the BER of the QDSL signal.

2.2.1 Additive Nature of Crosstalk

Regardless of the type of crosstalk that a signal is subject to (e.g. Self NEXT, Foreign FEXT etc), the induced crosstalk power in a signal carrying pair is, like noise, fundamentally additive. In the absence of non-linear effects, a monotone disturber at frequency f_1 will induce crosstalk noise in other pairs at the same frequency f_1. This is an important property of crosstalk, which will be invoked to allow its simulation in the hardware design.

Although more relevant to further work on the development of software models to drive the hardware simulator, when modelling the effect of crosstalk from multiple disturbers, one must take a statistical approach. Crosstalk models exist which predict both NEXT and FEXT for a single disturber. These can be extended statistically to model multiple disturbers, but not by simply summing of the effect from each individual disturber[6]. For example, to model the effect of two similar disturbers one cannot just double the results from a model of one disturber

2.3 Guassian, Coloured and Impulsive Noise

As with all telecommunication systems, Additive Guassian White Noise (AGWN) and impulsive noise from sources such as switching equipment is present in DSL systems. In the extended spectra of DSL signals other important noise sources are present due to high frequency em radiation. Noise from bandlimited radio communication systems often falls within DSL spectra. A common example is Amateur Radio transmission, which has a PSD within the Discrete Multi-Tone (DMT) ADSL PSD. Such noise of often termed coloured noise and as with crosstalk, AGWN and impulsive noise are additive.

References

[1] American National Standards Institute, Asymmetric Digital Subscriber Line Metallic Interface Standard, T1.413, August 1995.

[2] Walter Y. Chen, “DSL”, Macmillan, 1998, p54-58.

[3] Harold Hughes, “Telecommunications Cables”, Wiley, 1997, p114-120.

[4] Denis J. Rauschmayer, “ADSL / VDSL Principles”, Macmillan, 1999, p44-53.

[5] Denis J. Rauschmayer, “ADSL / VDSL Principles”, Macmillan, 1999, p89-130.

[6] Walter Y. Chen, “DSL”, Macmillan, 1998, p64-67.

Chapter 3

ADSL / VDSL Technologies

Both ADSL and VDSL are designed to operate on a subscriber's existing twisted copper access pair terminating at a Local Exchange Building (LEX) in conjunction with the existing Plain Old Telephone Service (POTS). Various types of both ADSL and VDSL exist or are under development, although only DMT ADSL has been standardised[1]. Dual operation alongside the POTS is achieved through frequency division multiplexing which supports the legacy unmodulated baseband voice service upto 4 kHz and modulated data transmission in a higher passband. Filters known as POTS splitters separate the two signals at both the customer's premises and LEX. ADSL offers asymmetrical data rates of upto 6 Mbps downstream and 768 kbps upstream, whilst VDSL will offer either symmetrical or asymmetrical transmission to a combined limit of 55 Mbps. The maximum achievable data rate for both systems is limited by reach and SNR. This chapter attempts to describe both DSL systems, but only to a level that the salient points related to a line simulator are brought out, not exhaustively. Where further detail is sought, reference texts are identified.

3.1 ADSL

ADSL is aimed at providing data rates from 1.5 to 6 Mbps downstream and upto 768 kbps upstream[2]. For a given data rate the maximum separation between the customer and LEX is limited by attenuation and SNR due to crosstalk and other noise. Simply, higher data rate lines have a shorter reach compared with lower rate lines as have lines with thinner conductors compared with thicker ones. As will be discussed shortly, two different ADSL modulation schemes exist which have slightly different performance objectives, but Table 3.1 gives an indication of approximate maximum and minimum ADSL performance for uniform, untapped access lines[3].

AWG	Maximum Reach	Maximum Rate
24	18 kft @ 1.5 Mbps	6.1 Mbps @ 12 kft
26	15 kft @ 1.5 Mbps	6.1 Mbps @ 9 kft

Table 3.1 Downstream ADSL performance figures

3.1.1 ADSL General Architecture

Practical deployment of ADSL to a new subscriber has been kept as simple as possible. No access line modifications are necessary, although a DSL service provider generally surveys the line to determine its length and condition and to determine the maximum possible data rate that it will support. A recent variation of ADSL is DSL Lite[4] which is designed to make the installation of a POTS splitter at the customer's premises obsolete allowing an ADSL Lite modem to be plugged into the existing phone socket as simply as a common 56 k PC analogue modem. DSL Lite, although not yet standardised, will use the same modulation scheme as DMT ADSL, but offer a reduced data rate.

Figure 3.1 shows the end to end model for full ADSL systems, for ADSL Lite, the customer's POTS splitter would be removed.

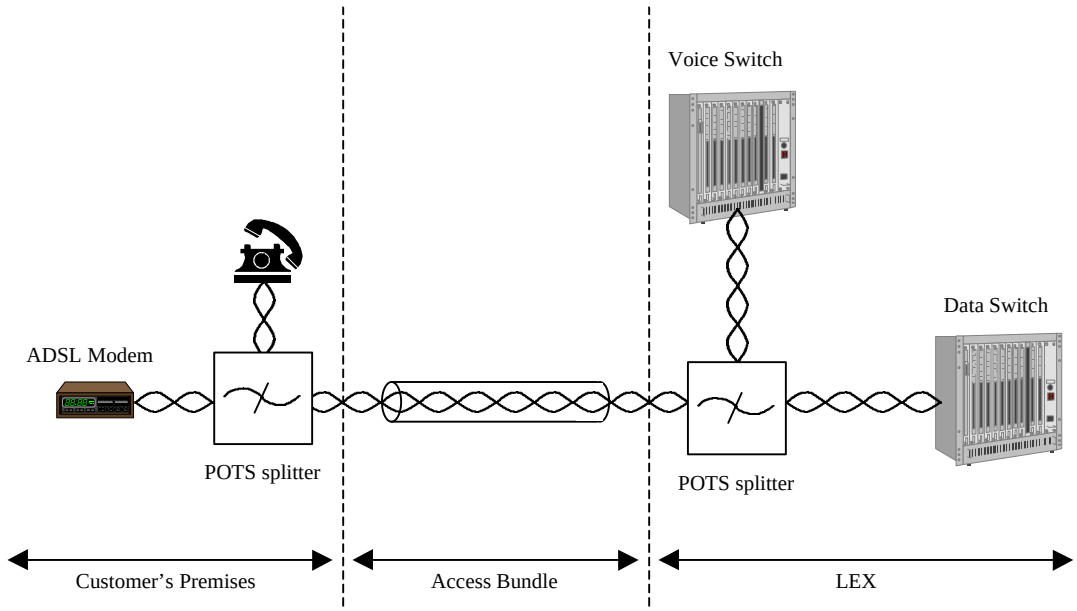


Figure 3.1 ADSL end to end model

3.1.2 DMT Modulation

The DMT ADSL modulation scheme is currently the only open standardised version of the access technology. ANSI have standardised all aspects of the service through the T1.413 series of documents. Full duplex operation is achieved either by Frequency Division Multiplex (FDM) or Echo Cancellation (EC), but both methods utilise the same basic multiple carrier QAM scheme.

QAM basically modulates data onto two orthogonal sine and cosine carriers of the same frequency. In its simplest form, 4 QAM, there are four possible combinations of the two carriers' amplitudes in the modulation scheme. 4 QAM modulates two binary bits per transmitted symbol (the symbol is transmitted through the combination of the two carriers for a duration called the symbol period). A succession of symbols in time is modulated by changing the amplitude of the two carriers according to the data bits' mapping onto the QAM constellation. The nature of the transition from one point to another in the constellation defines the bandwidth required by the modulation scheme. An instantaneous change from one point in the constellation to another produced by driving the modulator with a square wave would require an infinite transmission bandwidth. A modulating signal in the form of the Nyquist signalling waveform (a sinc function) gives the minimum required transmission bandwidth of half the symbol rate (one quarter the binary bit rate with 4 QAM).

Figure 3.2 shows the modulation of sine and cosine sinusoidal carriers for a constant input symbol.

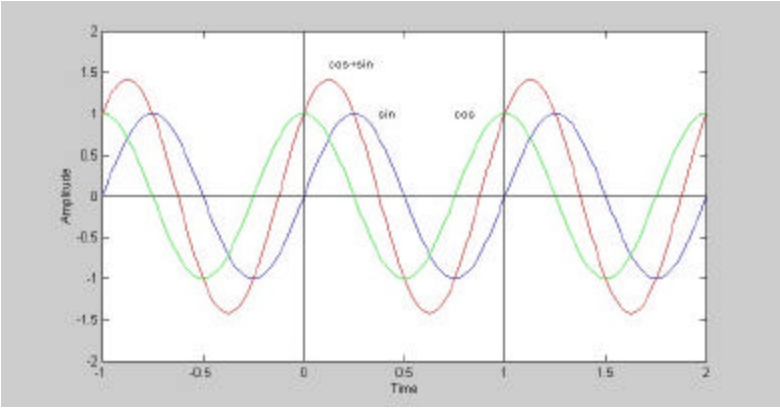


Figure 3.2 QAM modulation of a sine and cosine carrier

The information in figure 3.2 can also be represented by a constellation diagram which shows the magnitude of the carriers for each symbol and which binary bits are mapped to those symbols. If two binary zeros were mapped to the waveform in figure 3.2 the constellation diagram shown in figure 3.3 would result.

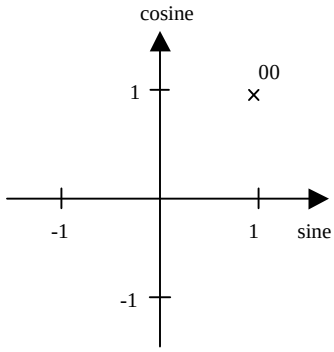


Figure 3.3 QAM constellation for a sine + cosine symbol

A complete 4 QAM constellation with the binary bit to sine and cosine amplitude mappings listed in table 3.2 is drawn in figure 3.4

Bits	Sine Amplitude	Cosine Amplitude
00	+1	+1
01	+1	-1
10	-1	-1
11	-1	+1

Table 3.2 4 QAM bit to carrier amplitude mapping

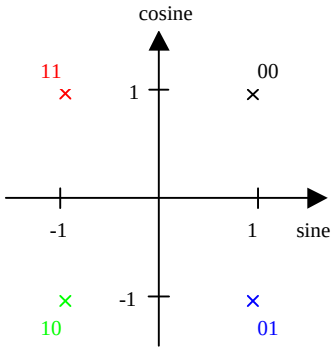


Figure 3.4 Complete 4 QAM constellation

DMT ADSL effectively modulates data onto 256 different sine and cosine carrier pairs. Since each carrier has a different centre frequency the technique is called Discrete MultiTone modulation. ANSI term each subcarrier pair as a ‘bin’ into which data is ‘loaded’ according to its constellation mapping. Each bin, i, is loaded with n_i data bits, with n depending on the measured SNR at the subcarrier frequency of the bin. A bin with a high SNR will be heavily loaded with data (i.e. use a large QAM constellation), whereas a bin with a low SNR will be lightly loaded or completely unloaded (i.e. use a small or null constellation). Figure 3.5 shows a conceptual 3 tone DMT modulation scheme with bins 1

and 3 loaded with 4 and 2 data bits per symbol respectively and bin 2 with none possibly due to intense coloured noise in its bandwidth.

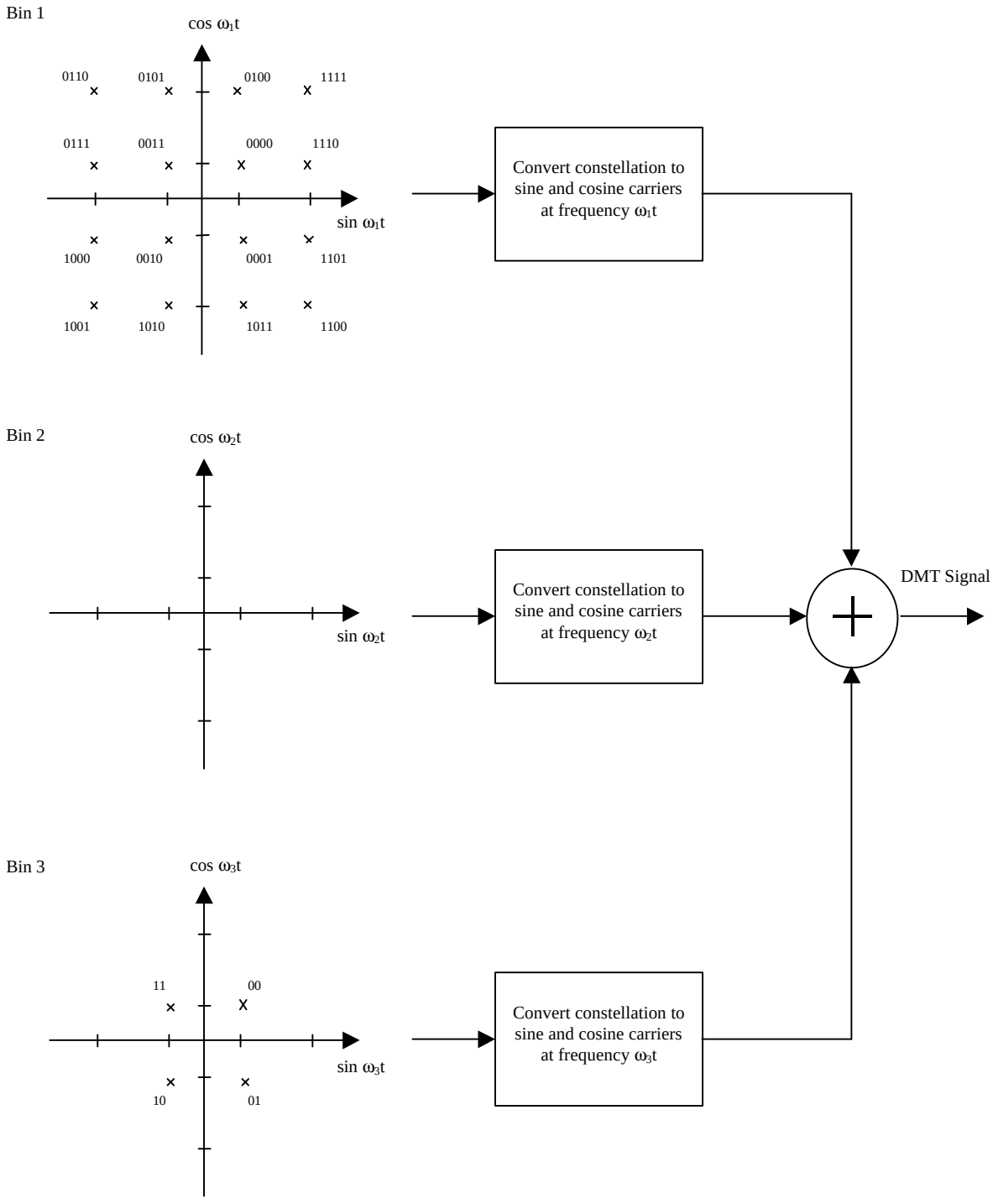


Figure 3.5 3 bin DMT modulation

In practical systems, this brute force method of summing up a set of separately generated QAM signals isn't adopted, rather each constellation output is placed along with its complex conjugate, in a vector on which an inverse Fourier transform is performed[5]. This produces a real-valued time domain sequence

that, after D/A conversion and filtering, gives a signal equivalent to that produced by the summation of QAM signals.

The simplified model of a 3 bin DMT scheme illustrates the rate adaptive nature of DMT ADSL. The data rate on each carrier pair is software controllable, which enables

- Redistribution of data loading into different bins to overcome specific bandlimited interfering signals.
- Service provider controlled tiering of data rates using software reconfigurable modems.
- FDM of the up and downstreams, thus eliminating self NEXT.

All these features reduce modem costs and widens service provision to a multi rate access system.

3.1.2.1 ANSI T1.413 DMT Specification[1]

The T1.403 standardisation for downstream DMT ADSL is for 256 carriers separated by exactly 4.3125 kHz. The first subcarrier is at 4.1325 kHz, with the 256th at 1.104 Mhz (called the Nyquist tone). The standard limits the data loading on each subcarrier to 15-bits, equivalent to a QAM constellation of 2^{15} (32 768) points, with a symbol rate of 4 kBaud (250 μs symbol period). For the upstream, there are 32 subcarriers with the same spacing, the first at 4.3125 kHz and the last at 138 kHz. The maximum upstream subcarrier loading is the same as the downstream. Figure 3.6 shows the streams' subcarrier spacing and PSDs if all subcarrier channels were equally loaded equally (i.e. the equivalent of 256 QAM signals with the same sized constellation - a theoretical situation solely to show the nature of the PDSs).

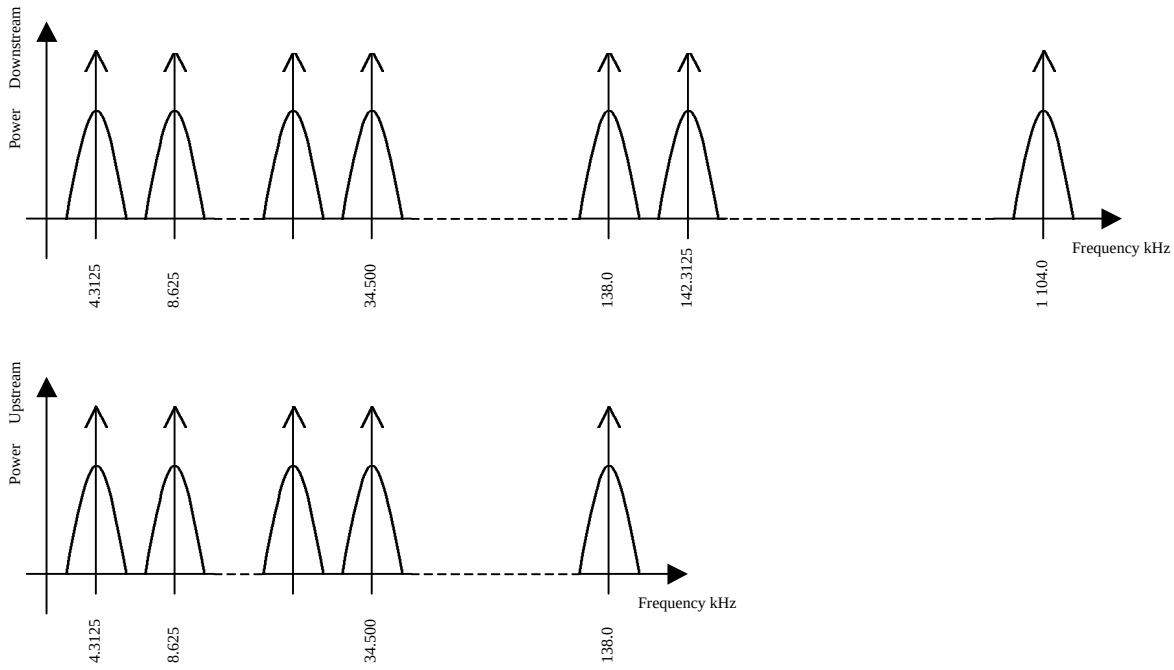


Figure 3.6 ANSI T1.413 carrier spacing and PSDs

Although the actual structure of the ADSL data framing and practical modulation scheme are very complex[6], the design of a line simulator is independent of such higher layer functionality as the ADSL signal is viewed at the physical layer.

3.1.2.2 Frequency Division Multiplexed DMT ADSL

FDM in ADSL systems is achieved by band limiting and spectrally separating upstream and downstream transmissions. The rate adaptable nature of DMT provides a simple way of implementing FDM by simply using null constellations for the downstream modulation bins where the upstream transmission spectrum is located. The same approach is used to give the FDM of the voice spectrum with the two ADSL signals. Upstream – downstream FDM is achieved by using null constellations for bins 8 to 32 in the downstream DMT transmission. Voice – ADSL FDM is through null constellations for bins 1 to 7 on both data streams. Overall, the PSD mask indicating the power spectra for all three signals is shown in figure 3.7.

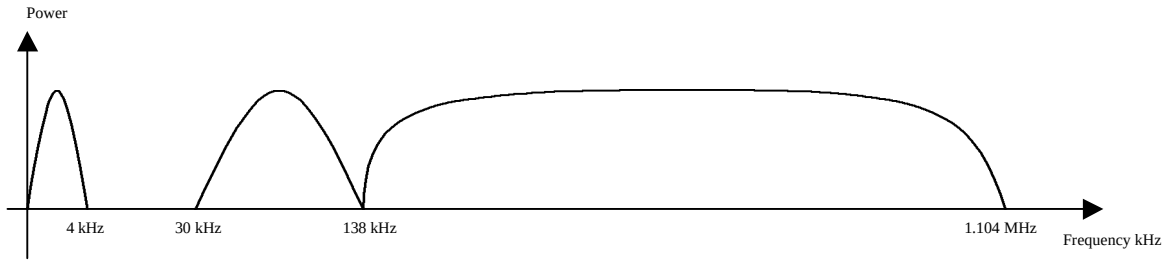


Figure 3.7 PSD mask for ANSI T1.413 FDM DMT ADSL

3.1.2.3 ADSL Lite

The PSD mask for ADSL Lite is spectrally limited by using null constellations for the higher frequency downstream subcarriers. The service is unstandardised, but uses the same subcarrier spacing as full ADSL.

3.1.2.4 DMT Echo Cancelled ADSL

An alternative to using FDM for duplex operation is to allow the data stream spectra to overlap then remove any unwanted echo at the co-located receiver due to self NEXT using cancellation techniques[7]. This is possible as a transceiver has knowledge of what it is currently transmitting, therefore it is able to remove by subtraction any attenuated copy of its own transmission arriving at its receiver. Echo cancelled DMT ADSL systems offer higher data rates of upto 8 Mbps, but have not been standardised. It is interesting to note that by giving the downstream transmission an extra 26 usable subcarriers, the maximum data rate is increased to just over 8 Mbps. This represents an increase in data rate of 33% for

only a 10% increase in occupied spectrum. This seemingly contradictory result is due to much higher SNRs at lower frequencies. In DMT ADSL systems, high frequency subcarriers tend to be lightly loaded whereas lower frequency carriers tend to be heavily loaded.

3.1.3 CAP Modulation

ADSL using CAP modulation has been developed by the company Globespan. No standarisation has been completed although several ad-hoc committees have made proposals[8,9]. Since CAP ADSL is a proprietary technology it is not precisely defined. Proponents of DMT ADSL have produced many papers championing their preferred solution at the expense of CAP[3,5]. Most of these papers have been written by companies after testing competitors' CAP modems and as such, one must view their findings with a pinch of salt.

CAP and QAM systems are very closely related. The term 'carrier-less' arises due to the primary difference between CAP modulation and true QAM. QAM modulators actually generate and mix two orthogonal sine and cosine carriers using analogue electronics. In contrast, CAP modulation is a digital process, where impulses are filtered by two filters with responses describing Hilbert transform pairs and then summed. Both techniques modulate data according to symbol to 'carrier' amplitude mapping constellations.

The ANSI proposal for CAP implementation[10] has a high symbol rate of 1088 kBaud with a maximum 256 point constellation. In comparison to DMT ADSL that transmits multiple, long 250 μs symbols, CAP ADSL transmits a single, short 0.91 μs symbol. Of greatest interest in relation to a line simulator is the PSD mask for CAP systems, shown in figure 3.8.

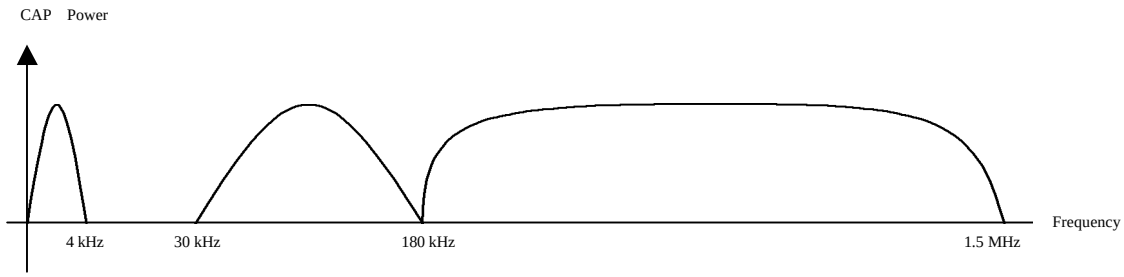


Figure 3.8 PSD mask for CAP ADSL

Whether both systems will survive is questionable. The overwhelming support for DMT is as equally accreditable to the reported merits of DMT over CAP as to the number of manufacturers producing DMT modems and chipsets compared with those producing CAP modems. Justifiably, manufacturers of DMT modems are concerned at the prospect of the possible wide deployment of DMT and CAP systems side by side in the same access bundle due to their overlapping spectra, shown in figure 3.9. T1.413 downstream DMT ADSL, which was designed specifically to eliminate self NEXT through FDM, would be adversely affected by CAP signals, especially since the lower DMT subcarriers are

heavily loaded with data. Whatever the technical merits of DMT and CAP, market forces are likely to pick the winning technology.

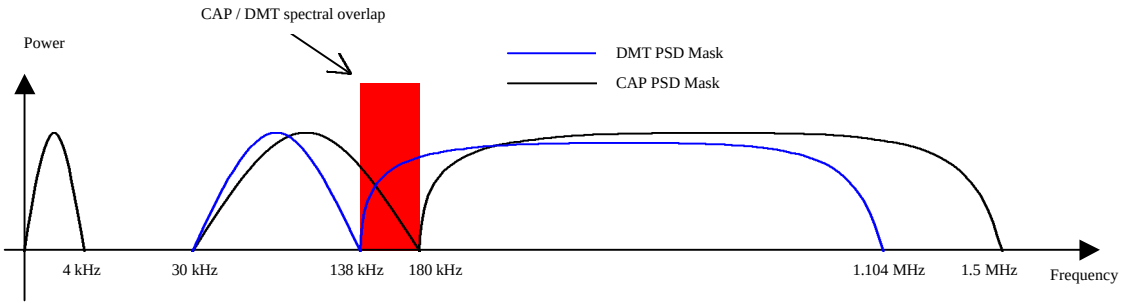


Figure 3.9 DMT / CAP power spectra comparison

3.2 VDSL

VDSL systems are designed to offer very high data rates upto 55 Mbps over distances of a few thousand feet. VDSL modems will be required to function with increased crosstalk and external interference due to the extended operational spectrum. Some typical performance targets proposed by ANSI[10] are illustrated in table 3.3.

Reach	Data Rate Downstream (max)	Data Rate Upstream (max)
1 000 ft	51.84 Mbps	2.3 Mbps
3 000 ft	27.6 Mbps	27.6 Mbps
4 500 ft	13.8 Mbps	13.8 Mbps

Table 3.3 VDSL performance targets

VDSL deployment is envisaged as a final drop to the customer in conjunction with fibre to the curb. Feasibility studies conducted by several companies have indicated CAP or QAM as the only commercially viable modulation techniques. Although DMT may well be preferable to CAP/QAM, DSP functionality has only recently reached a level advanced enough to implement ADSL DMT. As shown in figure 3.10, VDSL bandwidths are almost twenty times higher than for ADSL. DMT modulation using similar IDFT techniques to ADSL would require substantial parallel processing and is economically unattractive.

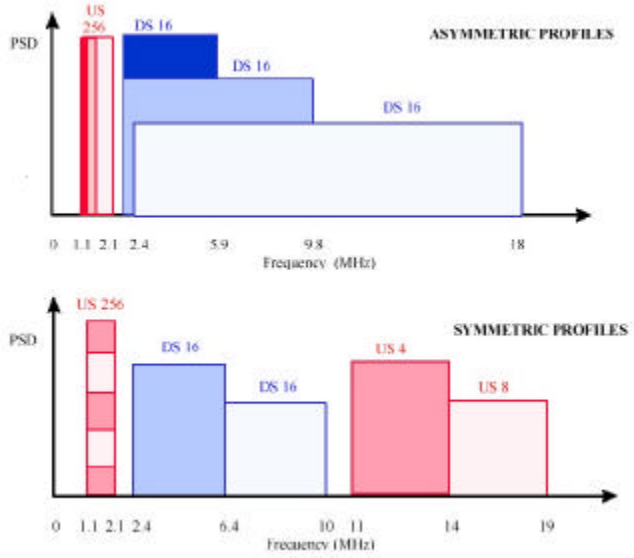


Figure 3.10 VDSL spectral spreads

3.3 ADSL and VDSL Data Rates and Reach

Although not directly relevant to the line simulator, an overall picture of the relationship between the probable deployment scenarios of ADSL and VDSL[11] is shown in figure 3.11.

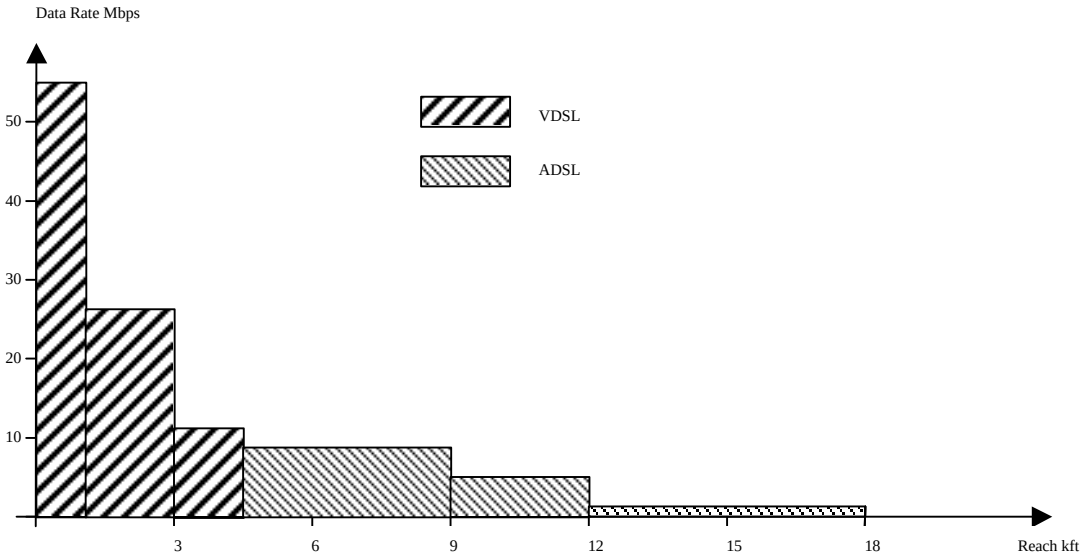


Figure 3.11 VDSL data rates and reach

References

[1] American National Standards Institute, ANSI T1.413-95, Asymmetric Digital Subscriber Line (ADSL) Metallic Interface, 1995.

[2] "The DSL Source Book", Paradyne Corp, February 1999, p24.

[3] Rupert Baines, "Discrete Multitone (DMT) vs. Carrierless Amplitude / Phase (CAP) Line Codes", Analog Devices, May 1997, p1-2 (Performance and Trials).

[4] "Consumer Installable ADSL: An In-Depth Look at G.Lite Technology", Orckit Communications Ltd, December 1998.

[5] "CAP vs DMT", Aware Ltd, White Paper, March 1999.

[6] "How does ADSL work?", DSL Knowledge Center, Orckit Communications Ltd, 1998.

[7] M. Ho, J. Cioffi, and J. Bingham, "An Echo Cancellation Method for DMT with DSLs", Amanti and Stanford University T1E1 Contribution, T1E1.4/92-201, December 1992.

[8] American National Standards Institute, contribution to standards review, T1E1.4/96-170R1, April 1999.

[9] M. Darveau et al, NorTel Inc, "QA / CAP RADSL Interference into DMT-ADSL", American National Standards Institute, contribution to standards review, T1E1 Ad Hoc/97-166, 1997.

[10] V. Freidman et al, "VDSL Draft Specifications", American National Standards Institute Document T1E1.4/98-054R1 (June 1998).

[12] P.Chow, J. Tu, and, J. Cioffi, "Performance Evaluation of a Multichannel Transceiver System for ADSL and VDSL", IEEE JSAC, Vol. 9, no. 6, August 1991.

Chapter 4

Line Simulator Requirements

Chapters 2 and 3 described the twisted copper pair environment at extended frequencies and listed the PSDs expected for ADSL and VDSL systems. The copper line gives rise the physical impairments of insertion loss and phase shift. Multiple xDSL lines within the same access bundle cause both self and foreign NEXT and FEXT. As with all electronics, additive impulsive and Guassian white noise is incident on the line. Due to the extended operational spectra, especially of VDSL, RF passband noise, often referred to as coloured noise, will impact on the systems overall BER. The line simulator must be able to accurately copy this behaviour throughout the relevant spectrum, but without inducing further signal impairments.

4.1 Physical Line Characteristics

A twisted copper line's insertion loss and phase response are functions of both frequency and reach. Chapter 2 developed theoretical expressions for both impairments and gave an indication of the predicted spread of their values using experimental data for the primary RLCG line parameters. Close examination is necessary to quantitatively describe the necessary simulator behaviour for both ADSL and VDSL CAP/QAM modulation schemes.

4.1.1 Insertion Loss

Figure 2.2, chapter 2, shows the predicted maximum insertion loss for lines to a maximum length of 12 kft, operating at frequencies to 10 MHz. Table 4.1 overleaf summaries the extremes of insertion loss.

Reach	Maximum Insertion Loss (dB)		
	@ 1 MHz (DMT ADSL BW)	@ 1.5 MHz (CAP ADSL) BW	@ 20 MHz (VDSL BW)
1 kft	5	7	≈ 30
6 kft	30	35	≈ 120
12 kft	62	87	Not Applicable

Table 4.1 Maximum insertion losses

For a uniform homogenous access line, the insertion loss is monotonically increasing. Pre-empting chapter 5, any digital simulation will in some manner quantise the frequency spectrum of the signal being modified by calculating a set of frequency values (called frequency samples) from a set of time domain samples of the signal. The discrete frequency representation will be modified according to the line's behaviour, termed frequency filtering. Whatever time-frequency transform is used, the size of the finite frequency quantisation interval will introduce new distortion when the insertion loss is simulated because the real line's insertion loss varies over the bandwidth of the frequency quantisation interval. The spacing of the calculated frequency samples is termed the transform resolution, with a single interval called a cell. Discrete transforms have a single complex data point for each transform cell. Sinusoids of different frequencies within a single cell cannot be distinguished[1], therefore a discrete transform can only model the insertion loss as a set of stepped discrete values. From this it is apparent the frequency resolution of the chosen transform must be small enough so that the line's behaviour can be approximated as flat over that region without introducing too much error. For example, consider a transform with just 16 frequency intervals to simulate the 20 MHz bandwidth of a VDSL signal over a 6 kft line. The transform's frequency resolution is 1.25 MHz (20 x 10^6/16). Figure 4.1 reproduces the insertion loss for a 6 kft 26 gauge twisted copper pair with an overlay grid spaced at the frequency samples of the 16 cell transform (shown upto 10 MHz). The red stepped line shows the discrete frequency filter's approximation to the real line's continuous response.

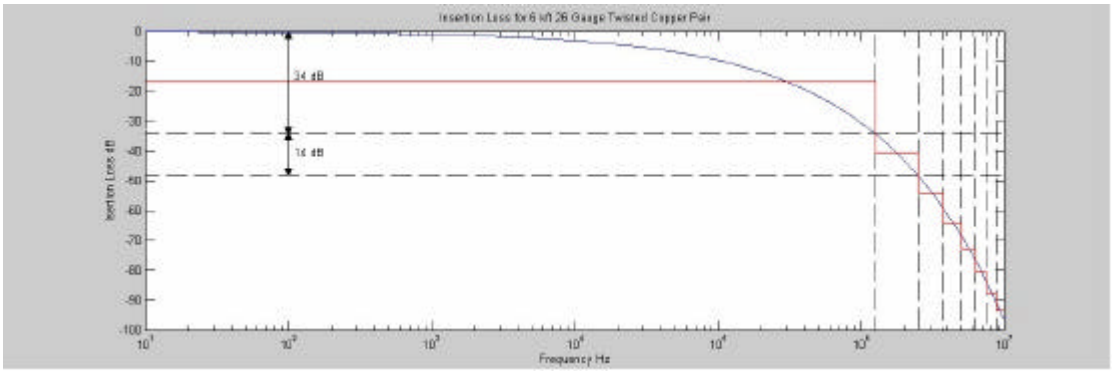


Figure 4.1 Insertion loss range

Over 20 MHz, a 6 kft line's insertion loss varies by 120 dB (extrapolated beyond 10 MHz). The average variation of insertion loss is 7.5 dB (120/16) across each transform cell. The greatest variation

occurs in the cell with the lowest frequency interval between 0 to 1.25 MHz which, from figure 4.1, is approximately 34 dB (The variation in the next interval is smaller at 14 dB). Clearly a transform with such a large large resolution compared to the actual variation in insertion loss wouldn't capture the line's behaviour at the lower frequencies. However, since DSL systems work along side the existing POTs, there is a 30 kHz guard band from DC, unoccupied by DSL signals, which doesn't need simulating. For VDSL, which operates between approximately 1.1 and 20 MHz, the maximum variation of insertion loss to be simulated occurs in the transform interval from 1.25 to 2.5 MHz, not the first interval from DC to 1.25 MHz. This interval will be termed the first relevant frequency interval and is the 14 dB variation shown in figure 4.1.

Table 4.2 lists the mean and maximum variation of insertion loss across the first relevant transform interval (not from DC), for DMT ADSL and VDSL systems for various power of two [note 1] frequency transform resolutions. Maximum losses were determined using the Matlab version 5 function `[x,y]=ginput(1)` which allows the user to specify graphical points on the response curves of figure 2.2 using a mouse and crosshairs (see appendix 2). An overlay grid was first placed on the response curve of the insertion loss, thereby indicating the transform's first relevant frequency interval. As an example, the insertion loss curve for a 12 kft twisted copper pair is shown in figure 4.2. For illustration, the overlay grid marks the first relevant frequency intervals above 30 kHz (the lowest loaded carrier for DMT ADSL), for transforms with resolutions of 32, 256 and 1024 frequency points over the part of the ADSL bandwidth between 10 kHz and 1MHz.

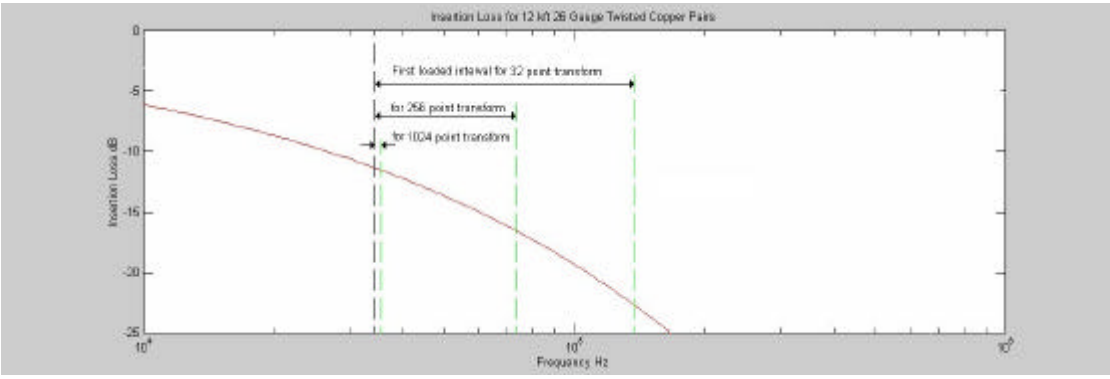


Figure 4.2. Insertion loss for 12 kft twisted copper pair with overlay grid showing the first relevant frequency transform interval

Clearly, with the small transform, the real copper pair exhibits substantial variation of insertion loss across the transform's resolution interval compared to the larger transforms.

4.1.2 Phase Shift

As shown from figure 2.4, chapter 2, the phase response of a twisted copper pair varies from $-\pi$ to π radians many times across DSL spectrums for even the shortest length of twisted copper lines. The frequency over which the phase repeats is known as the full phase rotational bandwidth and is of prime

Note 1: Chapter 5 shows that any DFT capable of computation within the time sampling window must be a Fast Fourier Transform where the input vector length is formed to be a power of two (or four).

No' of points	ADSL DMT (1.1 MHz BW)							VDSL CAP / QAM (20 MHz BW)				
	Resolution kHz	Mean Interval Loss (Across the entire DSL spectrum) dB			**Max Interval Loss (Across the first relevant transform interval) dB**			**Resolution kHz**	Mean Interval Loss (Across the entire DSL spectrum) dB		**Max Interval Loss (Across the first relevant transform interval) dB**	
		1 kft	6 kft	12 kft	**1 kft**	**6 kft**	**12 kft**		1 kft	6 kft	**1 kft**	**6 kft**
16	**68.8**	0.313	0.938	3.88	**1.33**	**7.97**	**16.1**	**1 250**	0.938	7.50	**2.32**	**14.10**
32	**34.4**	0.156	0.469	1.94	**0.35**	**2.32**	**4.75**	**625**	0.469	3.75	**1.27**	**7.63**
64	**17.2**	0.078	0.234	0.97	**0.21**	**1.36**	**2.64**	**313**	0.234	1.88	**0.67**	**4.26**
128	**8.6**	0.039	0.117	0.48	**0.11**	**0.67**	**1.36**	**156**	0.117	0.94	**0.34**	**2.28**
256	**4.3**	0.020	0.059	0.24	**0.06**	**0.34**	**0.69**	**78**	0.059	0.47	**0.18**	**1.06**
512	**2.1**	0.010	0.029	0.12	**0.03**	**0.18**	**0.35**	**39**	0.029	0.23	**0.09**	**0.59**
1024	**1.1**	0.005	0.015	0.06	**0.01**	**0.09**	**0.18**	**20**	0.015	0.12	**0.04**	**0.26**

Table 4.2 Insertion loss for various frequency transform resolutions

importance to the simulator design. The resolution of the discrete frequency transform used in the simulator must be small enough so that the phase change exhibited on the real line across the resolution bandwidth may be approximated as flat without introducing excessive distortion in the signal. Unlike the insertion loss, which has a maximum variation across the transform's first frequency interval, the rate of change of phase shift is constant across the spectrum resulting in an equal phase change across all of the transform cells. The full phase rotation bandwidths for 26 gauge 1, 6 and 12 kft twisted copper pairs are shown in detail in figure 4.3 and summarised in table 4.3.

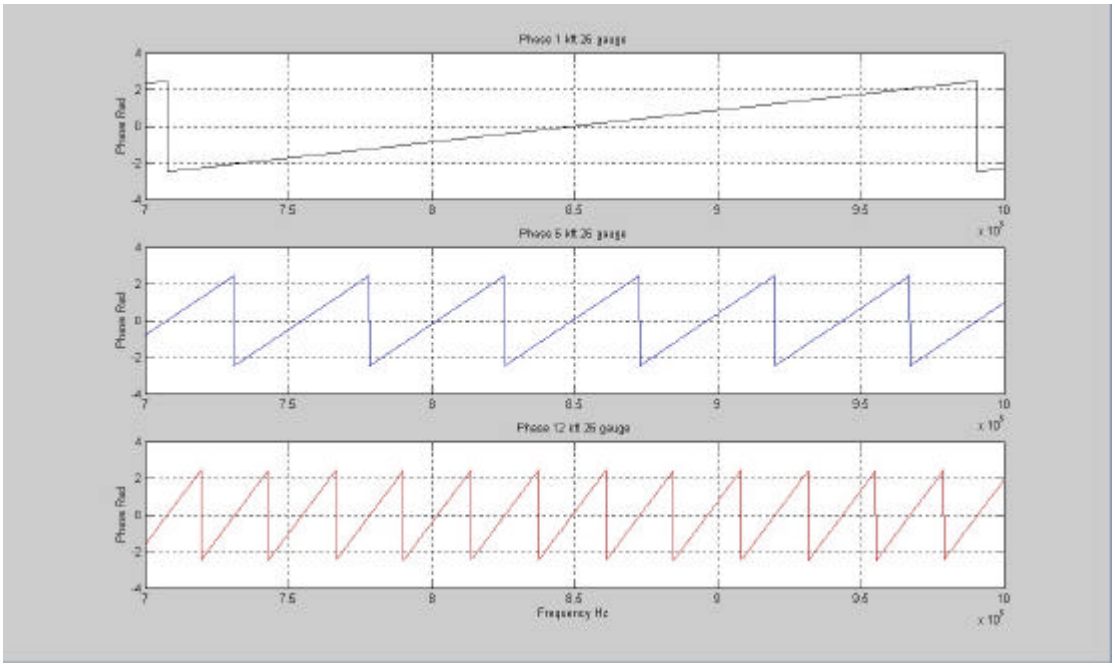


Figure 4.3. Detailed phase response for 26 gauge pairs

Reach	Full Phase Rotation Bandwidth (-π to π)	Phase Variation per kHz (rads)
1 kft	≈ 275 kHz	0.023
6 kft	≈ 46 kHz	0.136
12 kft	≈ 23 kHz	0.273

Table 4.3. Full phase rotation bandwidths for 26 gauge pairs

4.1.3 Acceptable Phase and Magnitude Variations

Before deciding on a minimum transform resolution the question as to what are acceptable variations in magnitude and phase across the transform cell arises? This is a very difficult question to answer.

Ideally the effect of increasing variations across the interval should be investigated with a prototype simulator with variable transform resolution simulating an access line between two xDSL modems. The minimum transform resolution would be determined according the simulator's experimental performance compared with an actual physical line for identical xDSL modem pairs. As an initial design, a maximum cell variation of 10%, allowing a maximum insertion loss variation of 0.4 dB (10log1.1) and phase variation of 0.628 radians will be used.

4.2 Crosstalk

The Analogue Front Ends[2] (AFEs) of DSL receivers and transmitters incorporate bandpass filtering to strictly limit the transmission bandwidth. In the absence of inter-modulation distortion, signal power falling outside the filtered bandwidth will be removed. A simulator therefore only needs to simulate crosstalk inband of its own PSD mask. The particular DSL modulation scheme defines the required simulation bandwidth, so therefore also specifies the required crosstalk simulation frequency range.

The required transform frequency resolution for crosstalk simulation is also dependent on the particular modulation scheme. For example, a simulator with a transform resolution of 10 kHz, won't have the required resolution to simulate crosstalk to individual DMT ADSL subcarriers spaced at 4.3125 kHz.

4.3 AWGN, Impulsive and Coloured Noise

Following the same argument as for crosstalk, the line simulator's bandwidth for modelling AWGN, impulsive and coloured noise will be fully specified by the requirements of the DSL signal's bandwidth.

4.4 ADSL Line Simulator Requirements

With specified PSDs and the simulation requirements of maximum insertion loss and phase shift variation over a transform cell, frequency transform parameters can be determined. Different specific simulation requirements exist for each distinct DSL access method so each will require different transform.

4.4.1 DMT ADSL

DMT ADSL modulates downstream data onto 256 subcarriers spaced at 4.3125 kHz intervals, the first at 4.3125 and the last at 1 104 kHz. A DMT ADSL line simulator must be capable of modifying each

individual subcarrier component, therefore there must be at least one frequency transform data point for each subcarrier. The upstream is similar, but limited to 32 carriers extending to 138 kHz.

4.4.1.1 Transform Requirements due to Insertion Loss

From table 4.2, with an insertion variation limit of 0.4 dB and access line lengths of 1, 6 and 12 kft, the required minimum transform resolution, number of frequency points and observable bandwidth are shown in table 4.4 for the downstream channel. The upstream transform requirements are satisfied by those for the downstream because the DMT spacing is the same, but over a smaller spectrum and with the same guard band of 30 kHz.

Reach	Minimum Transform Resolution (Downstream)	Minimum Number of Frequency Cells over Min' Observable Bandwidth (Downstream)	Minimum Observable Bandwidth (Downstream)
1 kft	4.3125 kHz	256	1.104 MHz
6 kft		256	
12 kft		512	

Table 4.4 DMT ADSL simulator transform requirements considering a maximum 10 % insertion loss variation across the first relevant transform cell

4.4.1.2 Transform Requirements due to Phase Shift

The transform requirements due to the maximum phase shift variation limit of 0.628 radians are determined from table 4.3 and listed in table 4.5 below, again for power of two transforms. As with the insertion loss, due to the rapid phase change with frequency for the long 12 kft lines (shown in the bottom plot of figure 4.3), the ADSL spectrum must be divided into an excessive 512 cells to achieve the specified limit of 10% phase variation across the resolution interval.

Reach	Minimum Transform Resolution (Downstream)	Minimum Number of Frequency Cells over Min' Observable Bandwidth (Downstream)	Minimum Observable Bandwidth (Downstream)
1 kft	17.25 kHz	64	1.104 MHz
6 kft	4.3125 kHz	256	
12 kft	2.16 kHz	512	

Table 4.5 DMT ADSL simulator transform requirements considering phase shift variation

4.4.1.3 Overall Transform Requirements

In order to satisfy the requirements for both loss and phase response and crosstalk, the higher resolution transform of the two tables and that given for crosstalk by DMT spacing must be used for each line type. Ideally the simulator should use a frequency transform with 512 samples to cover the entire range of specified reaches for DMT ADSL lines. However, as will be shown in chapter 5, a transform with 256 frequency samples within the 1.104 MHz bandwidth is more easily implemented. If a 256 cell frequency transform is used for long lines upto 12 kft long, the maximum interval insertion loss variation would be 0.69 dB (17%) over the first relevant frequency interval and the phase shift interval variation would be 1.18 radians (19%).

The frequency transform parameters in tables 4.4 and 4.5 satisfy both FDM and EC DMT ADSL.

4.4.2 CAP ADSL

CAP ADSL systems modulate data onto just one carrier in each direction, so the transform requirements are not dependent on a subcarrier spacing. The required transform resolution due to insertion loss and phase variation is the same as for the DMT simulator. CAP ADSL bandwidth is 50 % larger than that for DMT ADSL. Chapter 5 will show that a 1024 point DFT [note 2], with frequency sample spacing at the DMT subcarriers has a maximum observable frequency of 2.2 MHz. This indicates that the same transform hardware can be used to simulate CAP as well as DMT systems. However, a problem could occur due to the DMT simulator's AFEs which will incorporate lowpass filters with a 1.1MHz corner frequency. This would stop a very important part of the CAP spectrum from entering the simulator's frequency transform block. A solution would be to use the same transform block, but different AFEs for the two schemes.

4.5 VDSL Line Simulator Requirements

VDSL transmission spectrums are much larger than those for ADSL. In addition, whereas the DMT ADSL bandwidth is fixed regardless of the data rate, VDSL bandwidths are dependent on the data rate, shown previously in figure 3.10. Therefore the simulator requirements depend on the VDSL modem's designed data rate.

For a full rate service with a 20 MHz bandwidth and a short 1 kft access line, the minimum transform resolution considering the insertion loss is 39 kHz. This corresponds to 512 cells across the 20 MHz spectrum. The rate of phase variation is independent of frequency, but dependent on line length. From table 4.3, a 1 kft line requires a resolution of 27 kHz (the same for a 1 kft ADSL line, table 4.5) to limit the phase variation to 10% across the transform cells. From this requirement, the minimum number of transform cells is 741.

Note 2: When the term point is used in the context of a DFT, it refers to the combined number of positive and negative frequency points, so a 1024 point DFT divides its observable spectrum into only 512 positive frequency intervals or cells

4.6 Summary of Transform Requirements

Summarising, the required frequency tranforms for the three DSL modulation schemes under consideration are shown in table 4.6.

	Minimum Obervable Bandwidth	Minimum No. of Cells over the Minimum Observable Bandwidth
ADSL DMT	1.1 MHz	256
ADSL CAP	1.5 MHz	256 over 1.1 MHz
VDSL CAP / QAM	20 MHz	741

Table 4.6 Overall ADSL and VDSL simulator transform requirements

References

[1] E. Oran Brigham, “The Fast Fourier Transform and Its Applications”, Prentice Hall, 1988, p170.

[2] Denis J. Rauschmayer, “ADSL / VDSL Principles”, Macmillan, 1999, p208.

Chapter 5

Signal Processing Fundamentals

Signals can be described and modified in different domains such as time and frequency. Some signals are more easily modelled in one domain than in another. For example, impulsive noise due to switching is readily described by a rapidly rising and falling pulse in the time domain. The same noise may be modelled in the frequency domain in terms of its spectral components, but is less easy to do. The choice facing the designer of a line simulator is in which domain is it easiest to model and modify signals associated with DSL transmission? It is not a simple choice, as 'ease' encompasses many areas such as accuracy, complexity, versatility and cost to name but a few. To complicate the choice further, different categories of signal may be more suited to one domain rather than another, giving rise to a model described in more than one domain. This chapter will consider the two basic domains of time and frequency and develop signal processing principles for specific modelling implementations.

5.1 The Modelling Requirement

As discussed in previous chapters, a DSL line simulator must model the physical effects of the access line in addition to noise and crosstalk. The end to end description is of a transmitted signal from a DSL modem, x(t), modified by a physical line described by its impulse response, h(t), arriving at a receiving modem with the addition of noise, n(t), and crosstalk, c(t). In the time domain, x(t) is convolved[1] with h(t) and summed with n(t) and c(t). In the frequency domain, x(t)'s Fourier transform, X(f), is multiplied by h(t)'s Fourier transform, H(f), and summed with the noise and crosstalk transforms, N(f) and C(f). Figure 5.1 shows these two alternative descriptions.

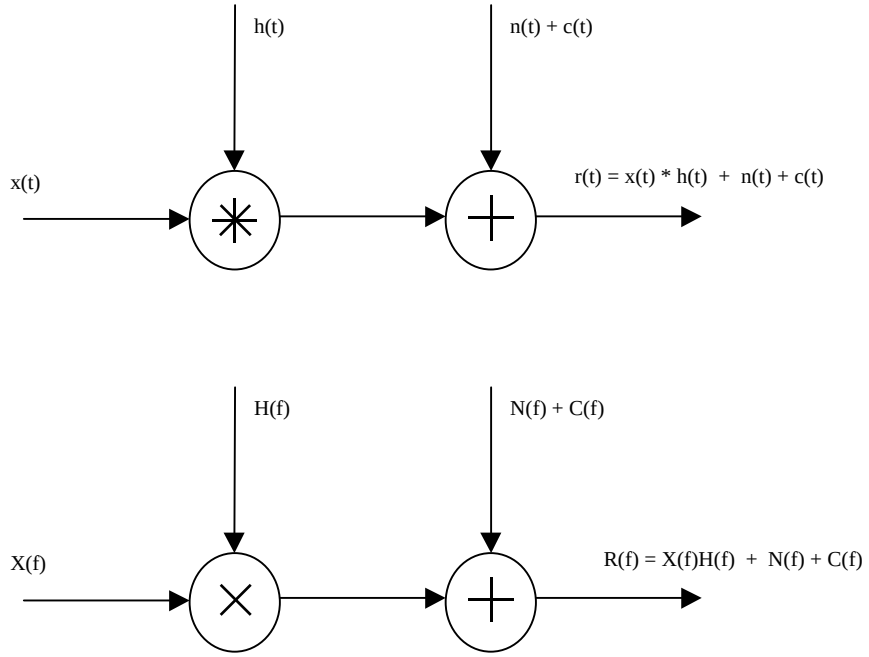


Figure 5.1 Time and frequency domain description of transmission process

5.2 Signal Domains

5.2.1 Physical Line Effects

Shown only the insertion loss of figure 2.2, it would be reasonable to conclude the response is one of a monotonic filter such as a Butterworth design. If this were the case, simulating the response in the time domain using a digital filter would be the obvious choice. However, if one also considers the phase response of figure 2.3, a designer would be challenged to reproduce it with a time domain digital filter. A linear phase response FIR filter could be used to model the insertion loss, but the phase response would be very difficult to recreate with a time domain digital filter.

An alternative to filtering in the time domain, which instead of convolving the signal and impulse response of the filter, is to multiply the signal and filter transfer function in the frequency domain. This is equivalent due the transform property of the Fourier transform; convolution in one domain is the same as multiplication in the other. Filtering in the frequency domain, is actually the preferred method for channel equalisation in practical receivers[2] where a Frequency Domain Equalisation (FEQ) filter multiplies the frequency samples given by a DFT of the incoming signal with the inverse frequency response of the channel it is connected to. Frequency domain filtering has also been employed extensively in the field of medical CT and MRI[3] scanning where processing is non-real time and requires tight control of phase response. The major problem with frequency domain filtering in the

telecommunication's arena is the requirement of real time processing. To filter in the frequency domain, a discrete frequency representation of the sampled time domain signal must be computed, then modified for each frequency sample then transformed back to the time domain. If the two transforms can be achieved in real time, frequency domain filtering provides a very powerful tool[4].

Figure 5.2 shows how a single tone's magnitude and phase can be modified using frequency domain multiplication, with an attenuation factor of 'B' and phase shift of 90°.

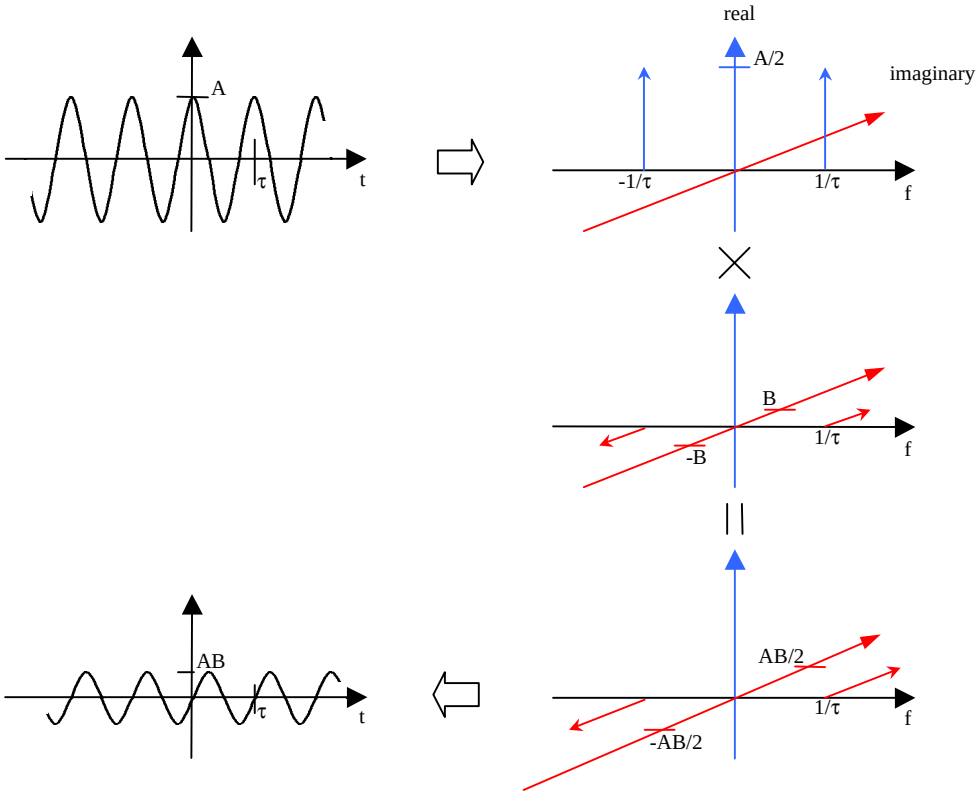


Figure 5.2 Magnitude and controlled 90° phase shift of a single tone

To produce the 90° phase shift, the Fourier transform of the signal is multiplied by a purely complex conjugate pair. The attenuation of the tone is given by the magnitude of the multiplying complex number.

Figure 5.3 shows a monotone attenuated and shifted by an arbitrary phase through multiplication with a complex conjugate pair with non-zero real and imaginary parts. The resultant phase shift of a tone at f_1 due to the filter's transfer function H(f) is given by

$$\mathrm{j}_{f_1} = \tan^{-1}\left(\frac{\mathrm{Im}(H(f_1))}{\mathrm{Re}(H(f_1))}\right)$$

The behaviour of a filter across a band of frequencies can be produced by multiplying each frequency component of a signal by different complex conjugate pairs. Clearly the control of the phase response

makes frequency filtering an ideal choice to model the phase response of an access line at the extended DSL frequencies.

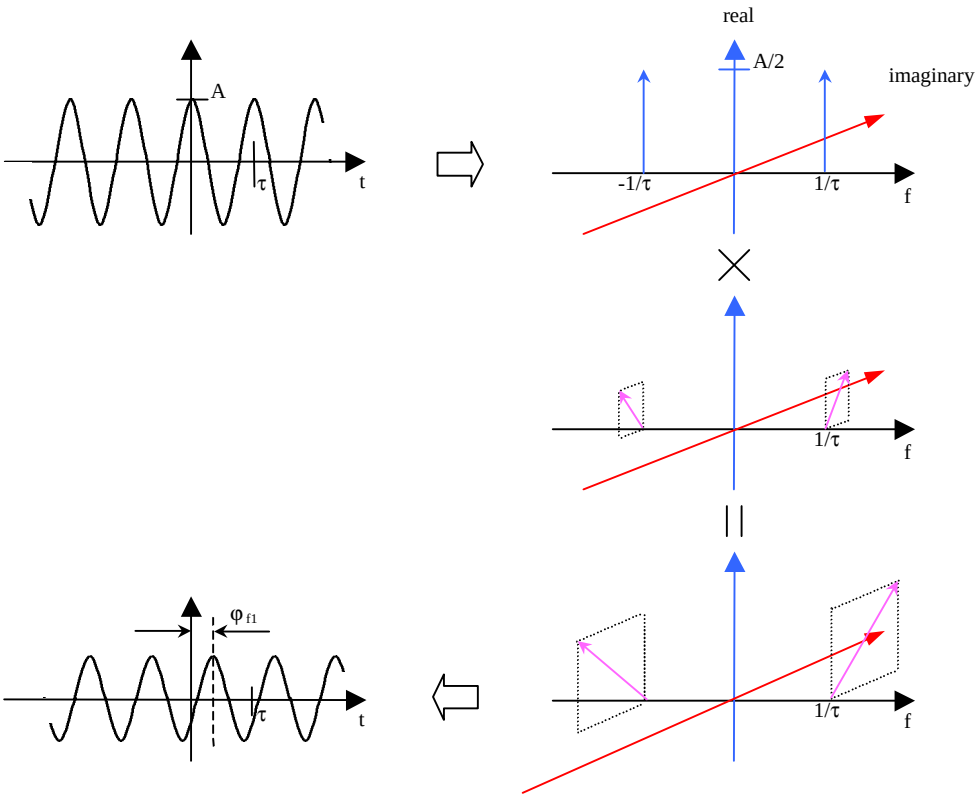


Figure 5.3 Magnitude and controlled arbitrary phase shift of a single tone

5.2.2 Crosstalk

DSL signals are characterised by their PSD which is a frequency domain description. Crosstalk, which is dependent on the PSDs of the disturbing and disturbed signals, can therefore easily be modelled in the frequency domain by the scaled addition of the disturber's PSD to the disturbed signal's PSD. With discrete transforms this corresponds to adding a scaled disturbing signal's DFT to the disturbed signal's DFT.

Self crosstalk to a DSL signal can be simply simulated by the addition of a delayed and attenuated copy of the DFT of the same signal. However, since the simulator will first sample the time waveform, self crosstalk can just as easily be produced in the time domain by the addition of a delayed and attenuated copy of the time sampled signal instead of the DFT of the signal under simulation.

Although both approaches are relatively straight forward, modelling in the frequency domain enables foreign crosstalk to be modelled more easily as knowledge of other DSL signals' PSD functions is generally known, whereas a time domain approach would require generation of line codes. Figure 5.4 shows both frequency and time domain self and foreign crosstalk modelling methods.

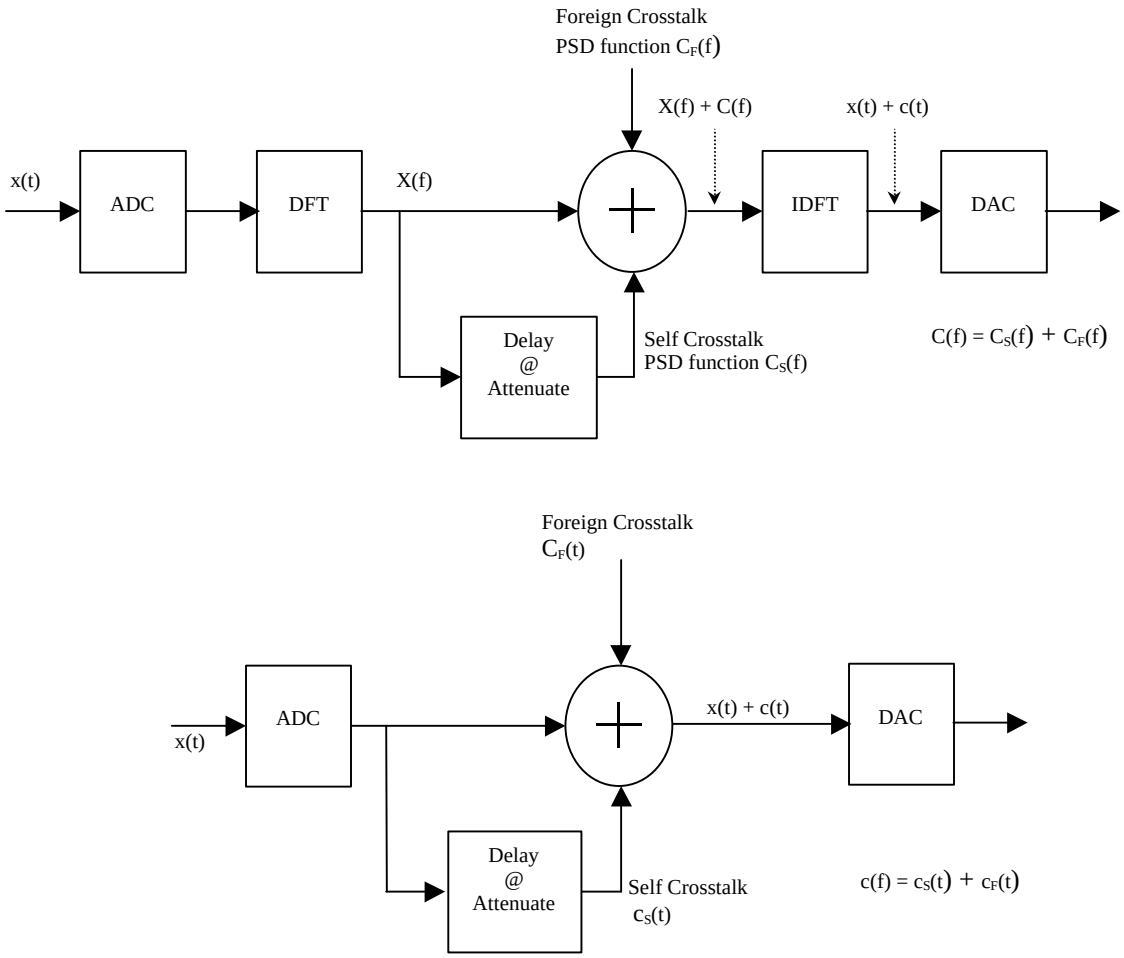


Figure 5.4 Frequency and time domain crosstalk modelling

5.2.3 Noise

5.2.3.1 Additive Gaussian White Noise

AGWN can be modelled in both time and frequency domains. In the frequency domain, AGWN can be generated as random sinusoids with random phase through additions of single complex conjugate points to the DSL signal's DFT.

5.2.3.2 Coloured Noise

By its very definition, coloured noise which is noise occupying a specific spectral band, is described in the frequency domain by its PSD function. Since coloured noise is a set of different frequency sinusoids, it could be modelled in the time domain, but would require additions to all time samples whereas in the frequency domain modelling requires additions to only the few frequency samples its occupies.

Modelled in the time domain[5], coloured or bandpass noise can be viewed as a sum of two sinusoids

$$n(t) = n_1(t)\cos\omega t + n_2(t)\sin\omega t$$

Alternatively it can also be viewed as a single sinusoid with randomly fluctuating amplitude and phase

$$n(t) = r(t)\cos[\omega t + \phi_n(t)]$$

5.2.3.3 Impulsive Noise

Impulsive noise is generally produced by switched currents ranging from semiconductor switching to motor contact arcing. Figure 5.5 shows a typical time waveform of impulse induced noise.

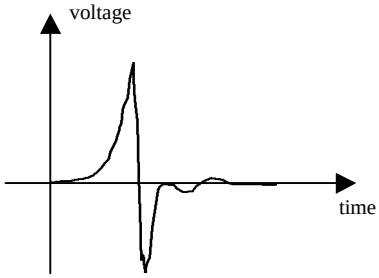


Figure 5.5 Impulsive noise waveform

Whilst repetitive impulsive noise bursts may occupy a fairly well defined spectral mask, random, irregular bursts of impulsive noise are more readily modelled in the time domain. Since both domains are useful in modelling impulsive noise, the combination of both time and frequency techniques shown in figure 5.6 gives maximum flexibility.

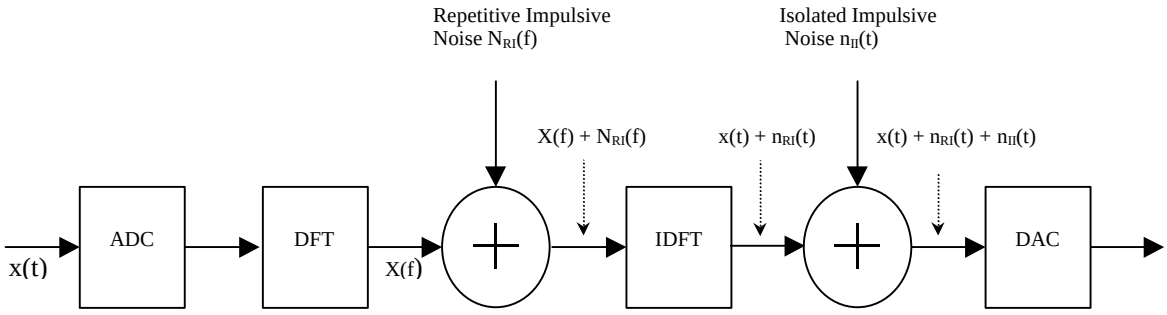


Figure 5.6 Impulsive noise modelling

5.2.4 Overall Simulation Model

Combining the modelling approaches of the previous sections, conceptually the physical line, crosstalk and noise environment can be simulated using the method shown in figure 5.7.

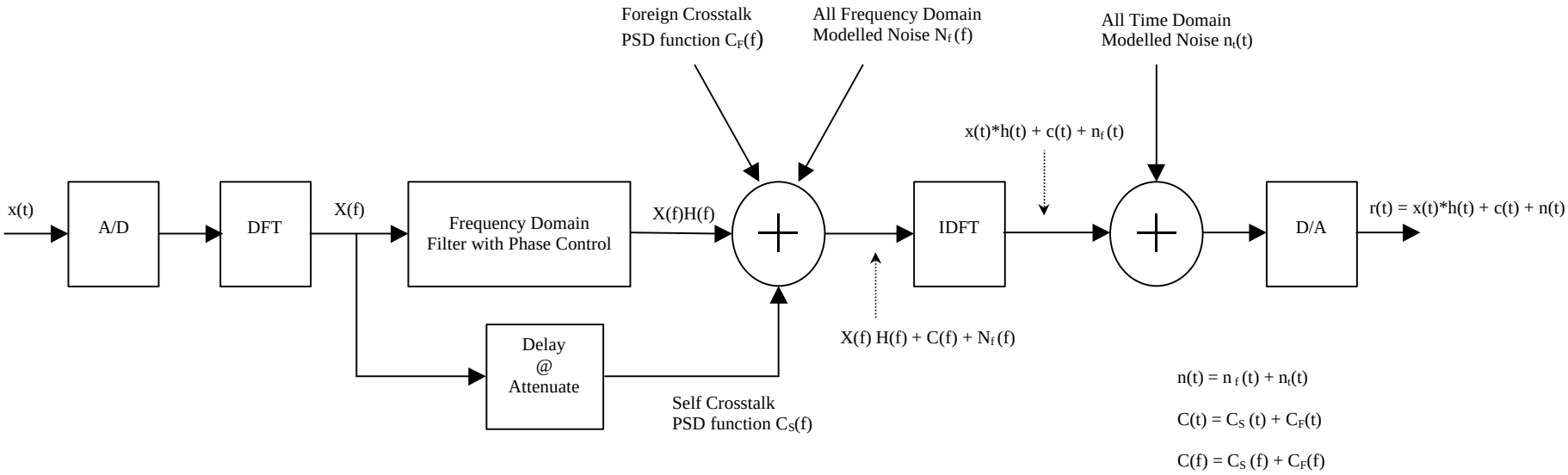


Figure 5.7 Overall simulation method

5.3 Discrete Fourier Transform

The Discrete Fourier Transform[6] (DFT) is the discrete equivalent of a signal's frequency domain description given by the continuous Fourier transform and is defined as

$$X\left(\frac{n}{NT_s}\right)=\sum_{k=0}^{N-1}x(kT_s)e^{-j2\pi nk/N} \qquad n=0,1,...,N-1$$

Application of this equation gives frequency components from DC to (N-1)/NT_s Hz in steps of the transform resolution, 1/NT_s . The components from 1/2T_s to (N-1)/NT_s are actually negative frequency components[7]. In addition, if the discrete transform is scaled the same as the continuous version, the DFT can be written

$$X\left(\frac{n}{NT_s}\right)=T_s\sum_{k=0}^{N-1}x(kT_s)e^{-j2\pi nk/N} \qquad n=0,\pm 1,\pm 1...,\pm N/2$$

Computed using the basic equation above, the computation can handle any number of time sample points, N, but tends to be computationally intensive and thus slow. Many algorithms have been developed dramatically reducing the computation time at the expense of setting conditions on the input vector's length. Speed optimised algorithms exploit the computational repetition in the full DFT that can occur for constrained data lengths. For example, the original Fast Fourier Transform (FFT) developed by J. Cooley and J. Tukey[8], the radix 2 FFT, requires an input vector of length N = 2^n. Multiplications comprise the large majority of the machine code instructions performed to compute a DFT using a digital processor. The radix 2 FFT algorithm's increase in computational speed over the basic DFT can be appreciated by considering the number of multiplications required by the two algorithms for a given time vector, shown in figure 5.8.

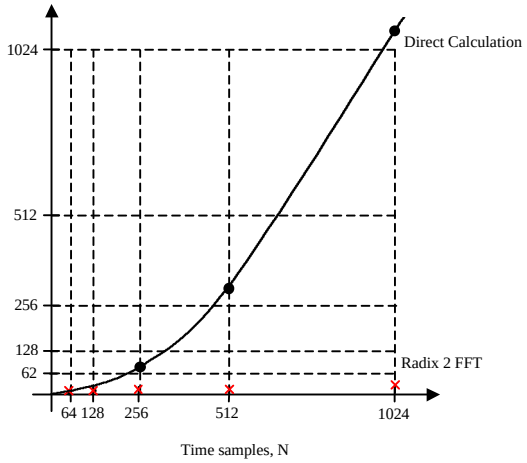


Figure 5.8 DFT and radix 2 FFT multiplication comparison

Other fast algorithms are commonly radix 4, requiring input data to be of length N = 4^n. Clearly if a data vector is conditioned for use with a radix 4 FFT it will also be suitable for a radix 2 algorithm.

5.3.1 DMT ADSL Fast Fourier Transform

Recalling the requirements set out in chapter 4 for the simulator's frequency transform, a DFT with 1024 points sampled at 4.416 MSPS has 512 cells each of width 4.31265 kHz and a maximum observable frequency of 2.208 MHz. Over the 1.1 MHz DMT bandwidth this gives 256 frequency samples. A 1024 point transform can be computed using either a 2 or 4 radix transform (2^{10} =1024 and 4^5 = 1024).

5.3.2 CAP ADSL Fast Fourier Transform

Since the DFT described for DMT ADSL has a maximum observable frequency of 2.208 MHz, it is also suitable for a CAP ADSL simulator that has the same resolution requirements and maximum observable frequency of 1.5 MHz. Sampling at just over 2 MSPS gives a 500 kHz guard band to avoid the effects of aliasing which occur when sampling at exactly twice the Nyquist limit is employed due to the physical impossibility of realising a brick wall filter necessary to finally recover the signal.

5.3.3 VDSL Fast Fourier Transform

A full VDSL simulator must have a minimum observable frequency of at least 20 MHz. The Nyquist sampling limit is therefore 40MSPS, although with the addition of a 5 MHz guard band, sampling at 50 MSPS is the practical minimum.

From chapter 4, section 5, short lines of just 1 kft (300 m), required 741 cells across the full rate 20 MHz bandwidth, which was constrained by the phase variation limit of 10%, not insertion loss variation across the first relevant frequency interval. Since there is a 5 MHz guard band, the transform must have a minimum of 926 cells across the 25 MHz bandwidth (741 x 25/20). This corresponds to a 2048 point DFT for a radix 2 FFT (24.4 kHz resolution) or 4096 points if a radix 4 algorithm is employed to compute the discrete transform (12.2 kHz resolution).

5.4 ADC and DAC Conversion

Before any DFT can be performed, the analogue signal from a DSL modem must be sampled in the time domain. Conversely, after taking the Inverse DFT (IDFT) of the modified signal, digital to analogue conversion must be performed. Practically all conversion must be to a finite quantisation interval described by a limited number of bits to be processed by any digital circuitry.

For a simulator, the ADC and DAC resolution must be at least as good as that used in the DSL modems themselves or else the simulator's sampling will distort the signal far more than the sampling within the modems. The sampling precision required in modems must be fine enough so as not to limit the

channel capacity further due to its presence. Practical fast ADC and DAC conversion is achieved using flash converters which due to their architecture have a limited number of quantisation bits and hence precision.

5.4.1 Theoretical ADC and DAC Precision

Extensive work by Dr. Walter Chen[9,10] on analogue front end precision requirements in DSL modems has been conducted on which the sampling precision for the simulator has been based. The author identifies three fundamental factors determining the required DAC resolution for passband QAM based transmitters

- The constellation size
- The desired signal-to-noise ratio or error rate
- The peak to average voltage ratio of digital shaping filters

The author draws particular attenuation to the effect of the digital filtering and the maximum transmission error rate on the required conversion precision. For example, a passband transmitter with a constellation of 16 points requires 4-bit quantisation ($2^4 = 16$) without digital filtering, which increases to 6 bits with filtering and a BER limit of between 10^{-6} and 10^{-8}.

For ADSL with a maximum constellation size of 32 768 points the minimum DAC resolution is 14 bits, but practically should be 15 to 16 bits to minimise the effect of quantisation noise compared with other transmission impairments[11]. The same author proposes a minimum 16-bit ADC resolution for lines upto 16 kft (3 miles) long.

5.4.2 Practical ADC and DAC Precision

As previously mentioned, a line simulator's sampling resolution must be at least as good as that used in the modems it will be connected to. Modem manufacturers however are very unlikely to divulge what is inside their equipment, but an indication of the conversion precision employed can be gained from considering recently released DSL AFE chip-sets that perform filtering and signal conversion. One such device, the ADSL KeyWave™ AFE manufactured by Fujitsu[12], is designed for both full and G.Lite applications and contains 15-bit ADC and DAC conversion blocks, sampling at some factor of the 17.664 MHz reference clock, probably 8.832 MSPS (17.664/2) or 4.416 MSPS (17.664/4). The exact reference clock division to the converters can only be predicted from the advance data sheet as the device isn't finalised yet, but will almost certainly be some factor of two.

References

[1] Walter Chen, “DSL”, Macmillan, 1998, p.107.

[2] Denis J. Rauschmayer, “ADSL / VDSL Principles”, Macmillan, 1999, p208, figure 7.2.

[3] O. Helenon, M. Laval-Jeantet, J. Frija , “Artifacts on lung CT scans: removal with Fourier filtration”, Radiology, 171(2), May 1989, pp.572-4.

[4] Edward Cunningham, “Digital Filtering, An Introduction”, Wiley, 1995, p346.

[5] John O’Reilly, Communication Systems Modelling Lecture Course, University College London, November 1998.

[6] E. Oran Brigham, “The Fast Fourier Transform and Its Applications”, Prentice Hall, 1988, p97.

[7] E. Oran Brigham, “The Fast Fourier Transform and Its Applications”, Prentice Hall, 1988, p169.

[8] J. Cooley and J Tukey, “An Algorithm for Machine Calculation of Complex Fourier Series”, Math. Computation, April 1965, Vol. 19, pp.297-301.

[9] W. Chen, “A Calculation of the Required A/D Precision for ADSL”, Bellcore T1E1 Contribution, T1E1.4/92-082, May 1992.

[10] Walter Chen, “DSL”, Macmillan, 1998, Chapter 7 (Analog Front-End Precision).

[11] N. Al-Dhahir and J. Coiffi, “On the Uniform ADC Bit Precision and Chip Level Computation for a Gaussian Signal”, IEEE Trans. On Signal Processing, vol. 44, no. 2 , February 1998, pp.434-438.

[12] Product Preview, KeyWave™ AFE, Fujitsu Microelectronics UK limited, December 1998.

Chapter 6

Matlab ADSL Signal Models and Transform Verification

The work described in this chapter arose for three main reasons:

- Matlab ADSL signal models could be used to examine the effect of different sets of frequency multiplication and addition vectors when developing the simulator's software drivers.
- To enable basic verification of the transform and sampling scheme for DMT ADSL signals.
- To satisfy the author's 'bit pushing' inclinations and intrigue at the physical nature of DMT QAM based modulation.

Seven Matlab functions have been written and the associated m-files included on the CD ROM, which enable multilevel QAM signals to be visualised, sampled and then filtered to recover pseudo analogue waveforms. Online help is included with all functions and can be accessed with the usual Matlab `help` keyword.

6.1 QAM Signal Generation

As described more fully in chapter 3, QAM fundamentally consists of the sum of sinusoid and cosinusoidal carriers of the same frequency whose amplitudes are discrete functions of the data words

to be transmitted. Data is encoded onto each carrier pair by mapping an m-bit binary word onto a unique pair of carrier amplitudes, forming a symbol. To enable a set of different words to be transmitted, the amplitudes of the two carriers are changed from the old to new values during the time between sampling epochs. The nature of the transitions determine the transmission bandwidth required. An instantaneous change from one set of amplitudes to another would require an infinite bandwidth, whereas if the transition is in the form of a raised cosine or Nyquist signalling function, the bandwidth required is equal to the data symbol rate. For example, with ANSI DMT ADSL, the symbol rate is 4.3125 kHz and each carrier group occupies a 4.3125 kHz bandwidth. Conceptually, DMT is simply the sum of a set of different orthogonal carrier pairs, each modulated according to its own data input.

6.1.1 Baseband QAM

The Matlab function `genbbnew` generates a time vector describing the baseband signalling envelope of a user defined series of data symbols (i.e. a time vector describing the amplitudes and transitions between of a set of carrier amplitudes at the sampling epochs). To illustrate this, consider the following two vectors which contain the cosine and sine carrier amplitudes at a contiguous group of sampling epochs:

```
cossym = [+1 +3 +3 –1 –3  +1 +1 –1]
sinsym = [+1 –1 –3 –1 –1  +3 –1 –3]
```

These two vectors contain the amplitudes of the two carriers only at the time sampling epochs and are illustrated in figure 6.1

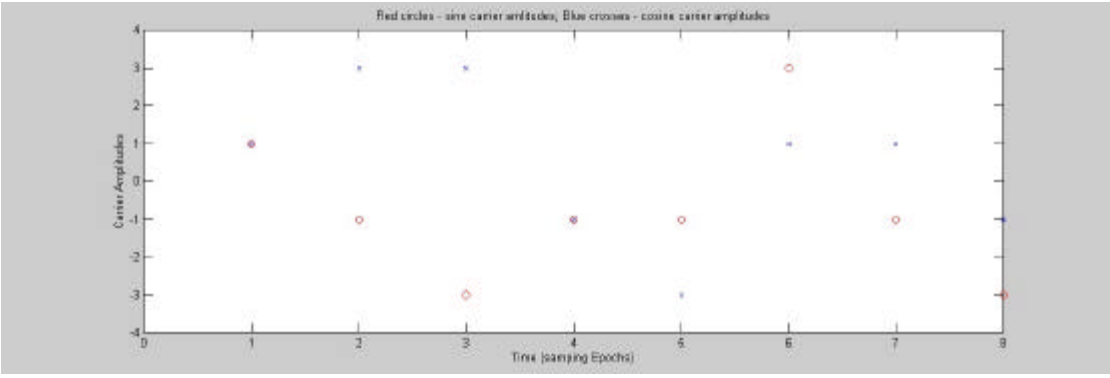


Figure 6.1 Illustrative cosine and sine carrier amplitudes for QAM

For ANSI DMT ADSL, the transitions between the points shown are of the form a raised cosine. The Matlab function

```
[tout, ycosbb, ysinbb] = genbbnew(cossym, sinsym, M, ts, tr);
```

returns two pseudo continuous time vectors `ycosbb` and `ysinbb` with points spaced `tr` seconds apart according to the two sets of carrier amplitudes `cossym` and `sinsym` using the raised cosine function to give the intermediate transition points. The top plot of figure 6.2 shows the baseband envelopes produced when `ts=1`, `tr=0.01`, `M=4` and the previous symbol vectors.

6.1.2 Passband QAM

The two carrier envelopes modulate the cosine and sine carriers by multiplication to produce the two orthogonal passband signals. The final QAM waveform is given by the addition of the two modulated carriers. The function

```
[tout, ybp, ycosbp, ycosbb, ysinbb, cossymb, sinsymb,]
            = qamdef(cossym, sinsym, M, ts, carrierfreq, tr);
```

generates the baseband envelopes using `genbbnew`, then multiplies with the two carriers of a user defined frequency, to produce the two orthogonal signals `ycosbp` and `ysinbp` and finally the QAM signal `ybp`.

The vectors returned using the same parameters as before and a carrier frequency at four times the symbol rate were used to plot figure 6.2 below.

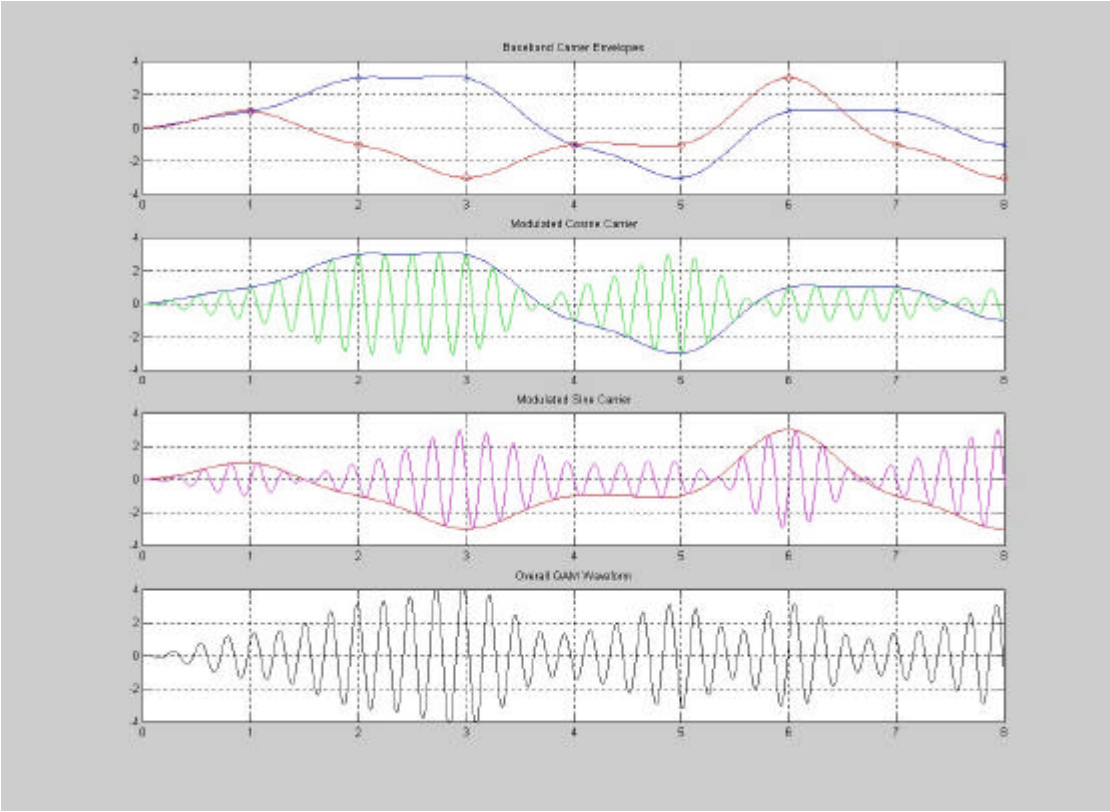


Figure 6.2 QAM signals produced by the Matlab function `qamdef`

It should be noted that although the piecewise approach used to give baseband envelopes, carriers and then passband signals by multiplication is perfectly valid, in practical DMT transmitters the multiple QAM signals are actually produced digitally by means of an inverse DFT, described more fully in section 3.1.2 and associated references.

6.1.3 Extension to DMT

Functions to produce a composite DMT signal, which is simply the sum of multiple individual QAM waveforms with different carrier frequencies, can easily be written through repeated execution of the `qamdef` function and a final vector summation.

6.2 Sampling

The functions above produce pseudo continuous descriptions of QAM signals. The term pseudo continuous is used to indicate that although no digital representation of a signal can be truly continuous (i.e. have an infinite number of points), the number of actual points is far greater than the number of points which would produced by the simulator's sampling of the real ADSL signal. For example, if the simulator sampled at four times the highest carrier frequency, the simulator's internal time data vector representing that pair of carriers would have just four points per carrier cycle. The functions `sampstep` and `sampfreq` both replicate the action of the simulator's ADC time sampling. The first allows the user to define the sampling step in seconds, the second as a sampling rate. Figure 6.3 shows plots of the vectors produced by sampling the final QAM signal of figure 6.2 with a sampling frequency of four times the carrier frequency (illustrated only over the first 3 symbols for clarity). It should be borne in mind when viewing the plot that the QAM signal is a composite addition of modulated cosine and sine carriers, therefore the number and position of samples within subsequent complete oscillations seems to vary. This is deceptive, as it is infact the QAM waveform which is varying compared to a simple sinusoid due to its very nature of modulation.

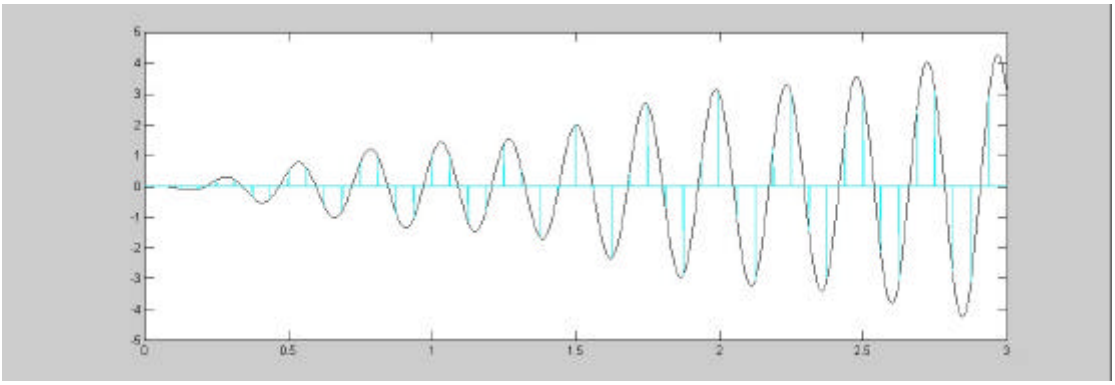


Figure 6.3 Sampled QAM signal produced by the Matlab function `sampfreq`

6.3 Filtering

The samples from the IDFT are converted to an analogue signal in the simulator by the action of the DAC and output low pass filter. Using Matlab, filtering of the sampled discrete time vector of the form shown in figure 6.3 can be achieved via convolution with a pseudo continuous filter impulse response. The function `brckfilt` performs the function of a perfect brickwall filter by convolved with the sampled time vector resulting from the IDFT. The function allows the user to enter the bandwidth of the filter.

6.4 Basic Transform Verification

Using the custom and standard signal processing Matlab functions, it is possible to generate signals for each of the 256 ADSL DMT subtones, sample, perform a DFT then IDFT and finally filter and compare with the original pseudo analogue signal. Physically, this is the same as A/D conversion, discrete Fourier transformation, multiplication by unity coefficients and addition with zero components in the frequency manipulation block, inverse discrete Fourier transformation and finally lowpass filtering on the hardware simulator board.

Various different sets of symbols, symbol periods, carrier frequencies and sampling rates were tested as described above. Comparison of all the resulting waveforms showed that the original psuedo continuous signal was perfectly recovered after sampling, transformation and filtering for sampling rates greater or equal to twice the carrier frequency. In addition when sampling was carried out below the Nyquist limit, the recovered and original waveforms were distinctly different due to aliasing. Mathematically this is not difficult, although somewhat laborious, to prove as follows:

1. Form a continuous equation in the time domain describing the QAM waveform using raised cosine signalling.
2. Sample this waveform by multiplication with a repetitive time impulse function with non-zero impulses at the desired sampling rate over a user defined time sampling window.
3. Perform a DFT
4. Perform an IDFT, which will give the same data vector as the original time samples because the DFT and IDFT are inverse operations with both being one to one functions.
5. Filter the IDFT results through repeated convolution in the time domain of all IDFT output samples with the filter's continuous impulse response.

This proof is by no means new, but the procedure of verification with Matlab is comforting in that the processing employed doesn't fundamentally alter the input signal and therefore in the perfect case introduces no changes to the signal during processing (zero quantisation noise, constrained periodic input signal, brickwall filter response).

6.5 Extension to ‘Simulator’ Simulation

One of the main motivations in writing the custom m-files was to produce a set of functions which would allow the entire signal processing operation from input to output to be modelled in Matlab to aid in the development of the future software driver. Such a model can be used from the command line or a GUI such as Simulink employed using modified functions as block operations. Such as model would enable the effect of different frequency and time manipulation vectors, sampling rates, quantisation precision and filtering schemes to be studied before a hardware board is finally constructed.

Chapter 7

Initial ADSL Line Simulator Design

The initial ADSL line simulator design was based solely on signal manipulation in the frequency domain. At the heart of the design are a FFT, manipulation block and an IFFT. Two implementation paths using Digital Signal Processors (DSP) and Field Programmable Gate Arrays (FPGA) were evaluated in terms of five metrics. In order to give an easily re-configurable environment, parameters describing the access line's response, noise and crosstalk environment must be continuously downloadable during a simulation run, either from a pre-computed data store or generated real time through a PC control interface. A detailed high level design using FPGAs was completed identifying peripheral signal conversion, logic and memory components.

7.1 Line Simulator Block Functionality

Figure 7.1 shows the basic block functionality of the initial simulator design. As previously mentioned, signal manipulation for line response, noise and crosstalk, is performed solely in the frequency domain through frequency filtering and spectral component addition respectively.

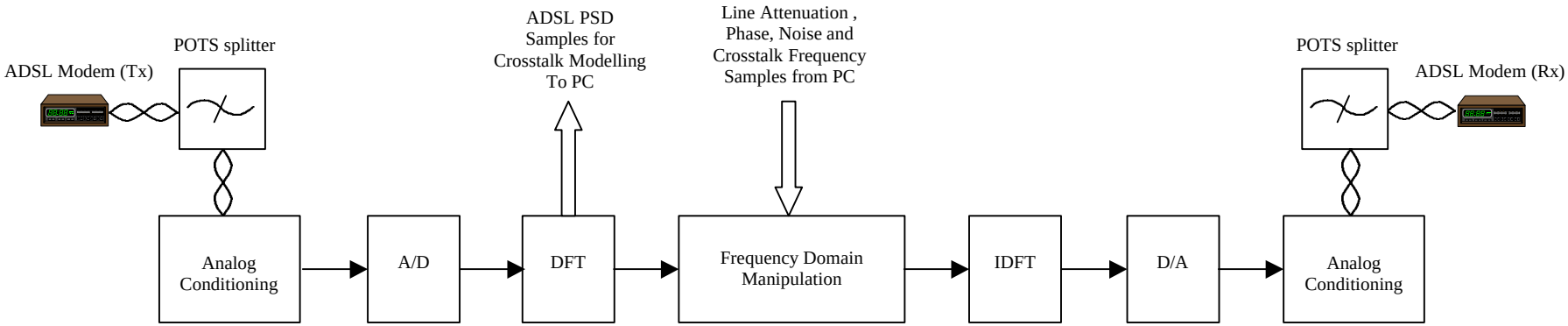


Figure 7.1 ADSL line simulator functional block diagram

7.2 Frequency Filter and FFT Implementation

Frequency filtering operates on a block processing principle. Repeatedly over each sampling window, N consecutive time samples are processed to give a block of N computed frequency samples (positive and negative). The frequency samples in each block must be multiplied by N complex discrete filter response samples. The resulting manipulated block of frequency components is then transformed back to the time domain using an IFFT. Clearly, for real-time simulation, each time a block 'enters' the FFT, a manipulated block must 'leave' the IFFT to prevent a build up of blocks requiring manipulation. If all three processes are carried out within the period of one sampling window, the overall signal will be subject to just one sampling window delay. However, such a scheme is both computationally expensive and unnecessary. The three processes of FFT, manipulation and IFFT may be pipelined so that each takes one time sampling window to compute. The result is a signal delayed by three sampling windows.

The FFT algorithm for an ADSL line simulator must repeatedly compute a 16-bit 1024 point DFT within one sampling window. At 4.416 MSPS, the sampling window, T_w, of a 1024 point DFT is given by:

$$\begin{aligned} T_w &= NT_s \\ &= \frac{N}{f_s} \\ &= \frac{1024}{4.416 \text{ x } 10^6} = 232\mu s \end{aligned}$$

Before any peripheral circuitry can be designed, the practical FFT implementation must be chosen. The peripheral circuitry for signal conversion, control logic and PC interfacing will essentially be designed around individual FFT solutions.

Three fundamental hardware approaches to performing the FFT were considered: hardware specific transform chips, DSPs and FPGAs. Most hardware FFTs are designed to operate in the audio spectrum and none were found offering the required performance of a 16-bit 1024 point DFT computed within the 232 μs sampling window.

In order to evaluate the suitability of using a DSP or FPGA to perform the FFT, manipulation and IFFT, the following five metrics were considered:

1. 1024 point 16-bit FFT / IFFT execution time
2. Chip packaging
3. Prototype and production costs
4. Versatility to implement new simulator functionality with minimal hardware redesign
5. Future adaptability to VDSL line simulation

The first metric, speed of execution, is obviously the most important for either a university or industrial based project, but the following four would rate differently in terms of importance within different development teams. For example, chip packaging and prototyping costs are very important to a four

month MSc project conducted in a setting with no established development hardware, whereas production cost and future adaptability to VDSL line simulation may be rated more highly in a commercial organisation with on-going DSL modem design programs.

7.2.1 Implementation Using DSPs

DSPs are basically microprocessors with architectures specifically designed for the repeated arithmetic operations commonly encountered in performing signal processing functions such as filtering, correlation and FFTs. As with common microprocessors, the CPU executes machine code instructions sequentially although some degree of parallelism may be incorporated. Complex mathematical functions are written either in machine code directly for specific processors, or in a higher level language such as C which is processor independent, then compiled down to machine code for a specific DSP architecture. Both machine code libraries specific to individual DSP architectures and higher-level software libraries exist for almost every signal processing function imaginable.

7.2.1.1 Speed Metric

From the outset, the decision to use a FFT library function or to write one's own code must be taken. The execution time of an existing function written in C will vary according to the target DSP architecture and compiler used. However, benchmark performance figures are generally available with individual functions on at least one host architecture. Although this will vary on different platforms, the performance figures give a good initial figure for execution time. Machine code library function execution times are fairly easy to determine as these are processor specific and will be published with the number of machine cycles required to run the entire algorithm. With knowledge of the processor clock period, a simple calculation gives the execution time. Although existing functions are not necessarily speed optimised, the prospective improvement through writing speed optimised code is unlikely to be dramatic and without extensive effort may even give reduced performance. Therefore for assessment of the speed metric, published benchmark performance figures are used.

Table 7.1 lists the execution times of 1024 point 16-bit radix 4 FFT algorithms on a representative selection of the latest DSP platforms[1,2,3].

DSP	Software Type	Clock Speed (MHz)	Execution Time (μs)
TMS320C6202	Machine Code	250	53
TMS320C6701	Machine Code	167	108
TigerSHARC	C	250	41
DSP56600	Machine Code	60	287

Table 7.1 FFT execution times on various DSP architectures

The first three entries in table 7.1 represent the latest DSP products, either at the product preview or sample distribution stage. The final entry for the Motorola DSP56600 is a current production device, released four years ago. The difference in processor clock speeds between the latest and established DSPs is quite dramatic and shows DSPs have only recently reached performance levels advanced enough to execute the 1024 point FFT identified for ADSL line simulation within 232 µs on a single device.

Both the Analog Devices TigerSHARC and the Texas Instruments TMS320C6202 DSPs are fast enough to implement both the FFT and IFFT functions within one 232 µs sampling period. Using the TMS320C6202, the 1024 complex (vectored) frequency filter multiplications each require 56 cycles, a total of 229 µs. In contrast, 1024 complex additions require just 2064 cycles or 8 µs. Clearly, to implement the transformations and signal manipulation algorithm would require two DSPs. Alternatively a hybrid solution with a single DSP to perform the transforms and a small FPGA for the multiplications and additions could be used, shown in figure 7.2.

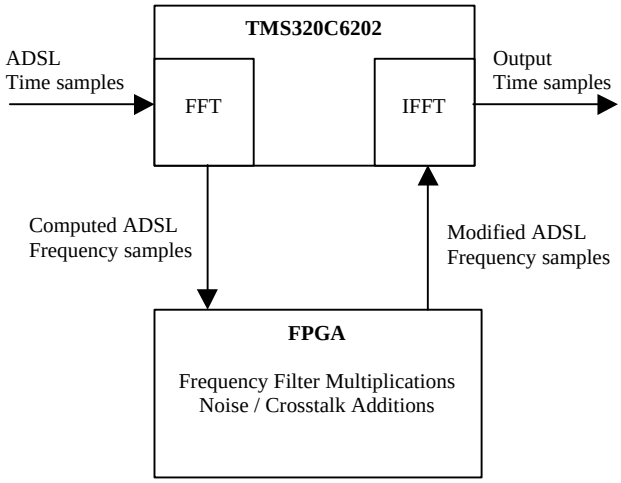


Figure 7.2 Hybrid DSP / FPGA solution

7.2.1.2 Chip Packaging

Table 7.2 shows the packaging options for the four DSPs identified above.

DSP	Package
TMS320C62xx	352 / 348 pin BGA
TMS320C67xx	452 pin BGA
TigerSHARC	400 pin BGA
DSP56600	144 pin QFP / BGA

Table 7.2 DSP packaging options

7.2.1.3 Prototyping and Production Costs

Since DSP solutions are fundamentally software based, some form of software emulation is required to verify code before functionality testing is performed on a development board. Both comprehensive emulation software and development kits cost several thousands of pounds each.

As is seen from table 7.2, the three DSPs capable of meeting the speed requirements are packaged in very large devices and are not really convenient for prototyping in a university environment with extremely limited resources.

Unit costs are difficult to determine as silicon suppliers tend to avoid single quantity supply, preferring instead to offer evaluation samples of devices, then minimum shipments of several tens of units. Samples may possibly be obtained via a large industrial sponsor such as Fujitsu. Currently the TMS320C6202 is quoted at $655 per unit with a minimum shipment of 200 units[4].

7.2.1.4 Versatility

One of the prime advantages of using a DSP to perform the FFT / IFFT functions is the ease with which they can be reprogrammed to give new functionality. Even with the hybrid approach of figure 7.2, on The TMS320C6202 there would be approximately 125 μs of 'free' processor time available each time sample window for added functionality.

7.2.1.5 Future Adaptability to VDSL Line Simulation

A quick calculation of the sampling window for the FFT described in chapter 5 of a 2048 point radix 2 FFT, with time samples at 50 MSPS gives

$$T_w = \frac{N}{f_s}$$
$$= \frac{2048}{4.416 \text{ x } 10^6} = 41\text{ms}$$

From the DSP benchmarks, a 2048 point radix 2 FFT would take 183 μs to execute on the TMS320C6202. In terms of the required speed increase to execute the function within the given 41 μs, a quadrupling of processor speed is required. A more efficient 4096 point radix 4 FFT requires 251 μs to perform. With a sampling window of 82 μs (4096 samples instead of 2048), real time processing requires a three-fold increase in computation speed. Considering the increase in clock speeds over the last four years (from the TMS320C32-60 to the TMS320C6202 and from the SHARC to TigerSHARC DSPs) a VDSL line simulator will probably be feasible using one DSP to perform the FFT and another for the IFFT within a few years.

Even if the packaging of new faster DSPs is different to the latest chips from the same manufacturer, the DSP solution does offer a logical proving ground towards a VDSL simulator.

7.2.2 Implementation Using FPGAs

7.2.2.1 Speed Metric

Unlike DSPs, FPGAs don't operate on a predefined maximum device clock. Instead, performance is determined by logic and path delays within a device. As such, it is impossible to accurately predict the performance of a new design without physically placing and routing it inside a target device. The design of a complex block such as a large FFT would take an experienced designer many man-hours and would be a highly iterative process to minimise logic and routing delays. However, the interest in the possible use of FPGAs to perform FFT functions was spurned by the 'Core' program piloted by the largest FPGA supplier, Xilinx.

The Core program provides many pre-designed and verified functional blocks with guaranteed execution times on specific target devices. In complex logic designs, an externally driven clock signal is introduced for synchronous operation so the performance of a particular Core design is given in terms of a maximum clocking frequency and number of clock cycles required. The minimum clock period is determined by the maximum critical net and logic delay within the Core design. Appendix 4 includes a list of available Cores and appendix 5 details three Cores of interest: 1024 point FFTs, parallel multipliers and registered adders.

With reference to the FFT Core data sheets in appendix 5, a 1024 point, 16-bit FFT can be performed in 17408 clock cycles. Further into the data sheet, an approximate period of 60 ns is given for external read access timing for the XC4013E-3 device. From the timing diagrams there are two clock periods for each memory read cycle, giving a clock period of approximately 30 ns (33 MHz) and total FFT execution time of 522 μs. Targeted at this device, the FFT Core isn't fast enough for real-time processing in the line simulator.

The XC4013E-3 is a fairly old device. Xilinx rate their chips with speed grades because, as previously mentioned, clock speeds only apply to synchronous designs and are different for each design. Two newer, improved versions of the XC4000 FPGA series have since been introduced, the XC4000XL series and in 1998 the XC4000XLA series. The XLA grades are claimed to have significant improvements in maximum clock speeds over the older 'E' grades. Figures 7.3 and 7.4 show extracts from Xilinx literature promoting the E and XLA XC4000 series.

The XC4000XLA speed grade saw the introduction of resources for a special 'Fast Clock' in addition to the standard 'Global Clock' resources present in the E grades. Comparison of the Global Clock of the E-3 and XLA speed grades show a predicted increase in chip speed of approximately 138% ((133 – 56) / 56). If this improvement is seen in targeting the FFT Core to the fastest XLA device, the execution time should be reduced from 522 μs to just 222 μs, within the target of the time sampling window.

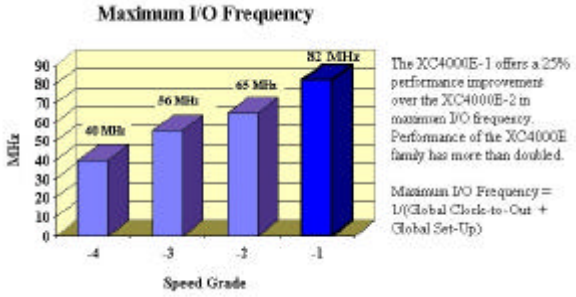


Figure 7.3 XC4000E chip speeds

Figure 7.4 XC4000XLA chip speeds

Cores also exist for parallel multipliers and scaled adders to implement the signal manipulation. Multiplication using FPGAs is implemented using lookup tables, whereas digital processors use repeated addition. With reference to appendix 5, targeted at XC4000E-1 devices, 16-bit by 16-bit multiplication requires just 5 cycles of the 58 MHz clock, or 86 ns. The multiplication Core deals with scalars, not vectors, therefore separate multiplications are needed for both the real and imaginary parts of each of the computed 1024 ADSL frequency samples. In the XC4000E-1 series, the total of 2048 multiplications would take 176 μs, but targeted at the fastest XC4000XLA speed grade the execution time should fall to about 108 μs (62% speed improvement). The Core multiplies two 16-bit numbers giving a 32-bit result. Obviously since the physical line is always attenuating, the simulator must multiply by numbers less than one. This can be achieved by ignoring the lower 16 bits of the result, shown in figure 7.5.

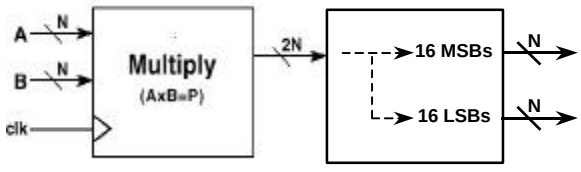


Figure 7.5 Multiplication by coefficients < 1

If the frequency sample to be attenuated is 16-bits wide, then the most significant 16-bits of the result of a multiplication by 2^{16} is equivalent to a unity multiplication. To attenuate by 2 use a multiplying coefficient of $2^{16}/2$ and use the 16 most significant bits of the result only.

The addition Core data sheet in appendix 5 doesn't include performance figures. However, addition should be at least as quick as multiplication when implemented using the CLBs of an FPGA. A ball park figure from the net and logic delay of an N-bit adder[5] of 4.5 + 0.35N ns, gives a processing time of approximately 21 μs for 2048 additions (real and imaginary parts) for the E-3 speed grade.

Combining the processing times for the signal multiplications and additions, using the fastest XLA grade device both functions can be performed within approximately 130 μs, well within one time sampling window.

7.2.2.2 Chip Packaging

Each Core design requires a minimum number of CLBs, listed in the relevant data sheet. The smallest device required for the FFT Core is the XC4013, which is available in 160 pin QFP packaging. For the manipulation block, the 16-bit area optimised multiplier Core requires 213 CLBs, while the 16-bit adder needs just 9. The total of 222 CLBs are available in the smallest XLA device, the XC4013XLA, which contains 576 CLBs.

Commercial, zero insertion force, multiple extraction cycle 160 pin QFP sockets are available with convenient pin layout which would enable prototyping within a university environment.

7.2.2.3 Prototyping and Production Costs

A total of three XC4013XLA devices are required to implement the FFT, IFFT and manipulation blocks using Cores. The XC4013XLA-07PQ160C is quoted at £59.99 from MicroCall. In addition to hardware, the Foundation software design suite is required to generate Cores and finally place and route in target devices. A single university licensed copy of the full package costs £373 (Normally $7995).

Both prototyping and production costs for an FPGA implementation are a fraction of that for a DSP solution.

7.2.2.4 Versatility

Xilinx FPGAs are programmed by a serial bit stream stored in a PROM during the initialisation period immediately after power up. The internal CLB configuration and hence the functionality of the device is determined by the content of this bit stream. Therefore, within the limitations of the physical layout of a FPGA based simulator board, new functionality can be programmed to an existing FPGA device if sufficient CLB resources are available on that device.

The FFT / IFFT Cores each utilise approximately 90% of the two XC4013 devices (532 out of 576 CLBs), with the multiplier and adder using 39% of a similar device, so some scope is present for revised functionality based on a simulator board with three FPGA devices without hardware modification. Combining the spare resources from all three FPGAs after their Cores have been placed and routed, approximately 442 CLBs are free for implementing other control and support logic which may also be required on the simulator board.

7.2.2.5 Future Adaptability to VDSL Line Simulation

A simulator design based on the 1024 point FFT Core is less adaptable to future use for VDSL line simulation because larger 2048 and 4096 point FFT Cores don't exist. In addition to this requiring the in-house development of larger FFT functions, considerably more CLBs would be needed for their implementation. However, the fundamental signal processing approach could be proved using FPGAs for the simulation of ADSL lines then a larger VDSL simulator built.

In terms of speed, the VDSL simulation requirement of a 2048 point FFT computed in just 41 µs implemented using FPGAs requires a projected 10 fold speed increase, considerably more than the three fold increase required of a DSP solution.

7.2.3 Preferred Implementation

Overall, mainly due to ease of prototyping and cost, the FPGA solution to FFT, IFFT and signal manipulation is preferred over the hybrid DSP-FPGA solution.

7.3 Overall Line Simulator Design

Once the decision to use FPGAs to implement the simulator's fundamental functions is taken, an overall design around these blocks can be made. Extensive reference is made to the FFT Core data sheet in appendix 5. Figure 7.6 shows the FFT Core Interface from the data sheet.

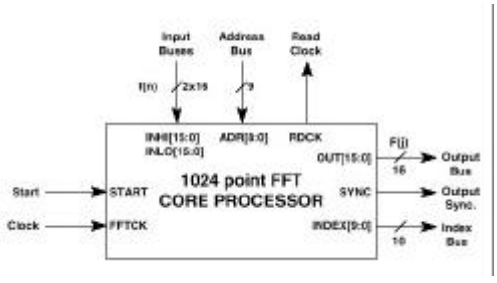


Figure 7.6 1024 point FFT interface pinout

A block schematic diagram of the complete simulator design is shown in figure 7.7.

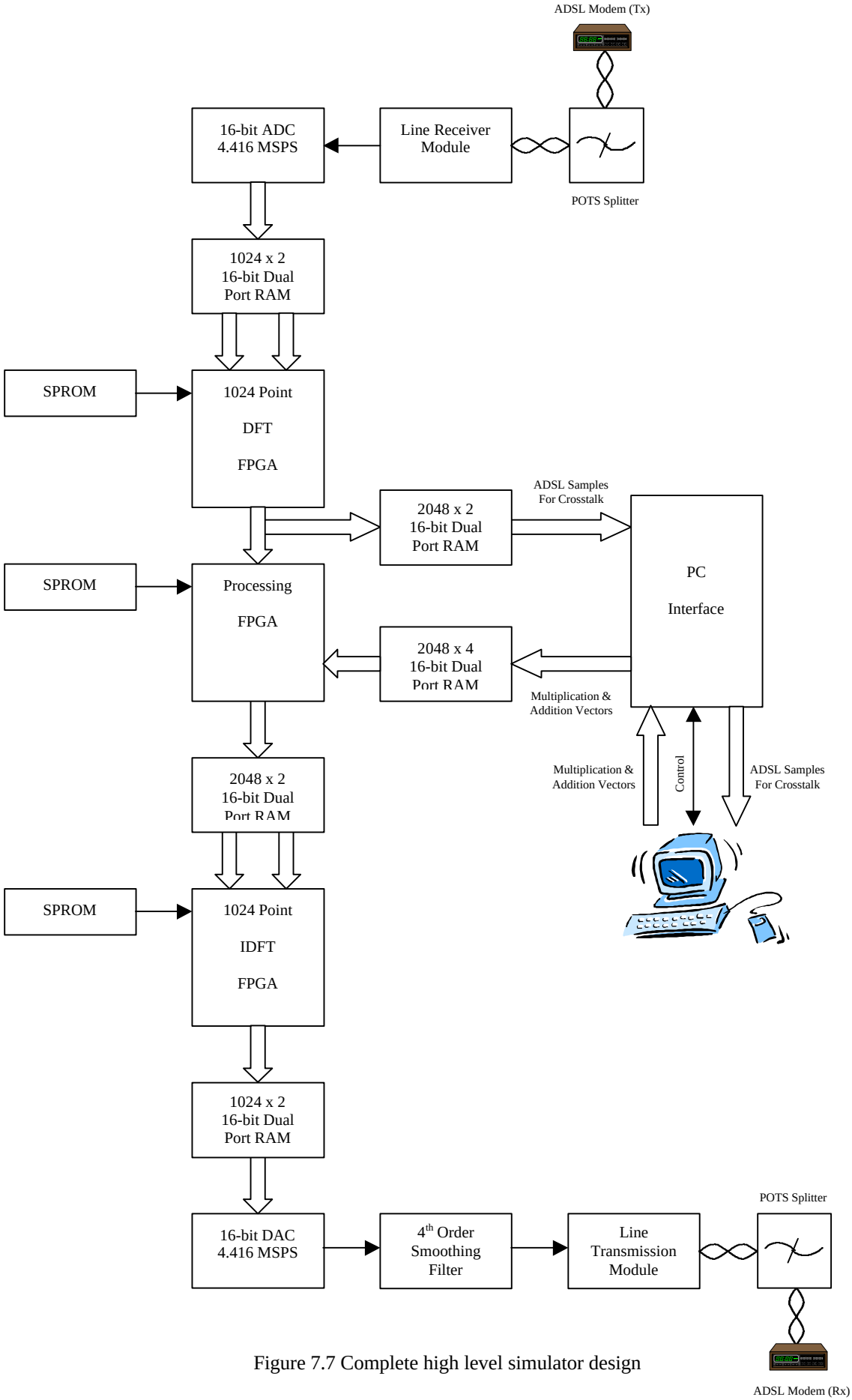


Figure 7.7 Complete high level simulator design

7.3.1 FFT Core Input Data Conditioning

The basic FFT Core is used to perform both the DFT on blocks of 1024 ADSL time samples and the IDFT on blocks of 1024 complex manipulated frequency samples. A new transformation as done each time sample window on a new set of data samples. Since the same FFT Core design is used for both DFT and IDFT functions, they will be referred to as the FFT and (I)FFT Core respectively.

7.3.1.1 ADC – FFT Interface

The FFT Core requires input data split into two blocks. For the DFT, the first block, called the LOBLOCK, consists of time samples t(0) to t(N/2 – 1) and the second, the HIBLOCK, consists of time samples t(N/2) to t(N –1). The FFT Core reads one data word from each block every two FFTCKs. In addition, the core requires un-interrupted access to both memory blocks. Because the ADC produces data at a different rate (once every ADCLK) and in a sequential order, each time windowed group of 1024 time samples from the ADC must be buffered and then read in the order and at the rate required by the FFT Core during the next window period. This can be achieved by loading the time samples from the ADC into two blocks of dual port RAM organised into two pages. Whilst data from the converter is being loaded into one page on one side of the dual port RAM, data from the previous sampling window in the other page will be read by the FFT Core from the opposite side. During the next sampling window, data from the ADC is loaded into the second page whilst the core performs the DFT on the 1024 time samples stored in the first page. This process repeats continuously. Conceptually this arrangement is shown in figure 7.8.

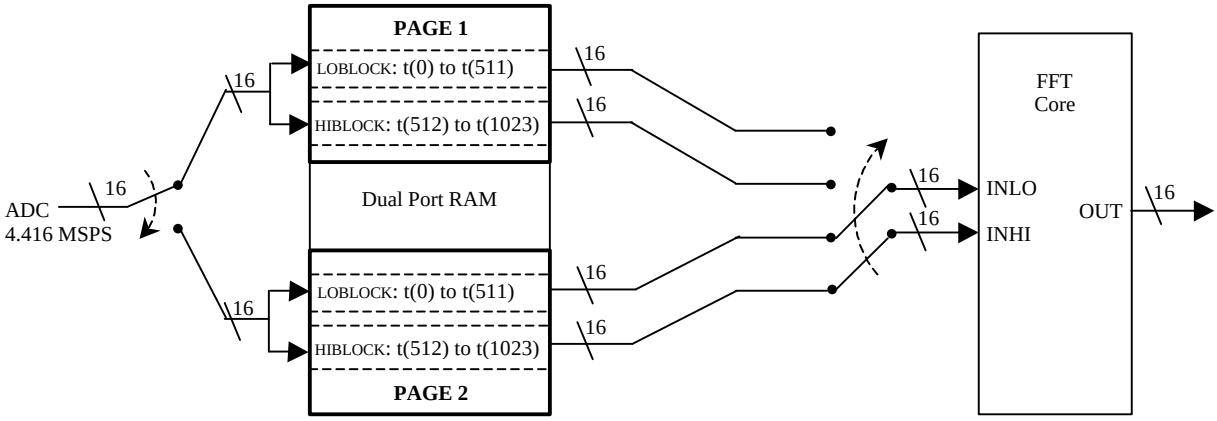


Figure 7.8 Conceptual memory arrangement for time sample input to the FFT

Each block of each page consists of 512 16-bit wide locations. Practically the block structure can be implemented using just two 1 kByte dual port RAM chips, one for each block. Shown in figure 7.9, each block contains 2 pages, page 1 extends from address location 0 to 511 with page 2 from location 512 to 1023.

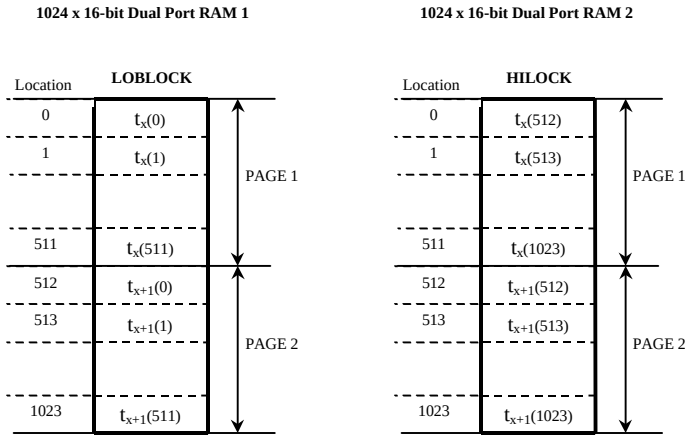


Figure 7.9 Practical memory block and page structure

A single 9-bit parallel output counter can be used to address both pages at the same time with appropriate block and page selection control, shown in figure 7.10 with associated digital timing in figure 7.11. The FFTSTART signal for the FFT Core is generated on the rising edge of the BLOCKSELECT signal which occurs every 1024 ADCLKs (one time sampling window). All the supporting logic shown in figure 7.10 can easily fit into the spare capacity of the FPGAs after the Cores have been placed and routed.

7.3.1.2 IFFT – DAC Interface

The ADC samples at 4.416 MSPS, which is also the rate at which samples must be converted by the DAC. Since the FFT Core reads its input data at a different rate and in a different order to which the ADC produced it, memory buffering of the time samples is required. In the same way, the order and rate of data output from the (I)FFT Core is not sequential and not at 4.416 MSPS, so memory buffering is also required between the (I)FFT and DAC.

After multiplication and addition, the modified 1024 frequency samples must be stored in a similar two block arrangement to that for the DFT shown in figure 7.9 to allow the (I)FFT Core un-interrupted access to its own input data. Here, the LOBLOCK consists of the manipulated complex frequency samples, f(0) to f(N/2 – 1) and the HIBLOCK the samples f(N/2) to f(N – 1). During each window period, data output from the (I)FFT Core is written to one side of the dual port RAM, whilst data written to memory from the (I)FFT Core in the previous window is read out to the DAC from the other side at 4.416 MSPS and in a sequential order. However, whereas the input to the FFT Core is real valued only, its output is complex with both real and imaginary parts. The output from the manipulation block is also complex, therefore the storage space required for the (I)FFT Core input is double that compared with the FFT Core.

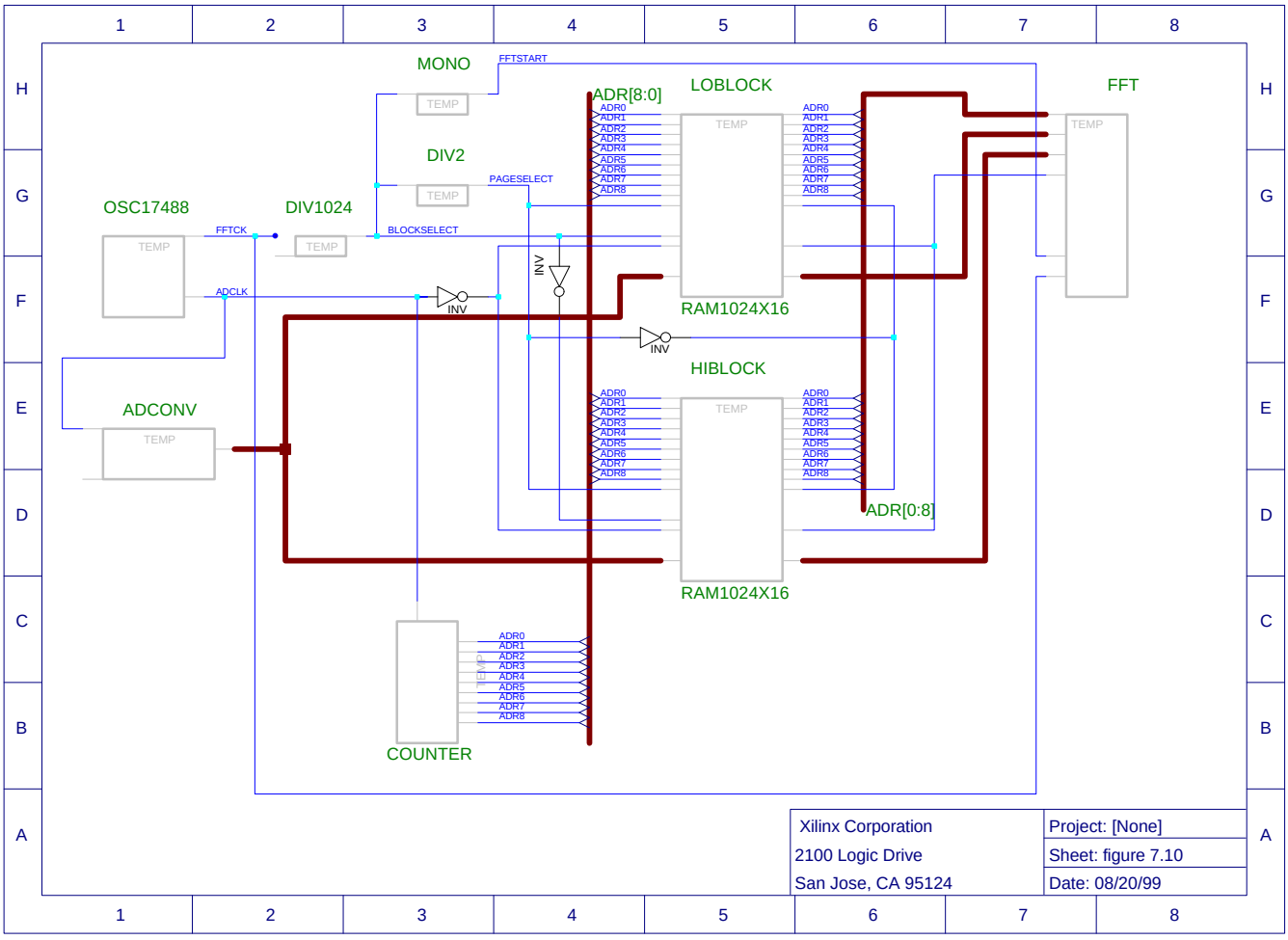


Figure 7.10 Circuit diagram for ADC and FFT Core interface

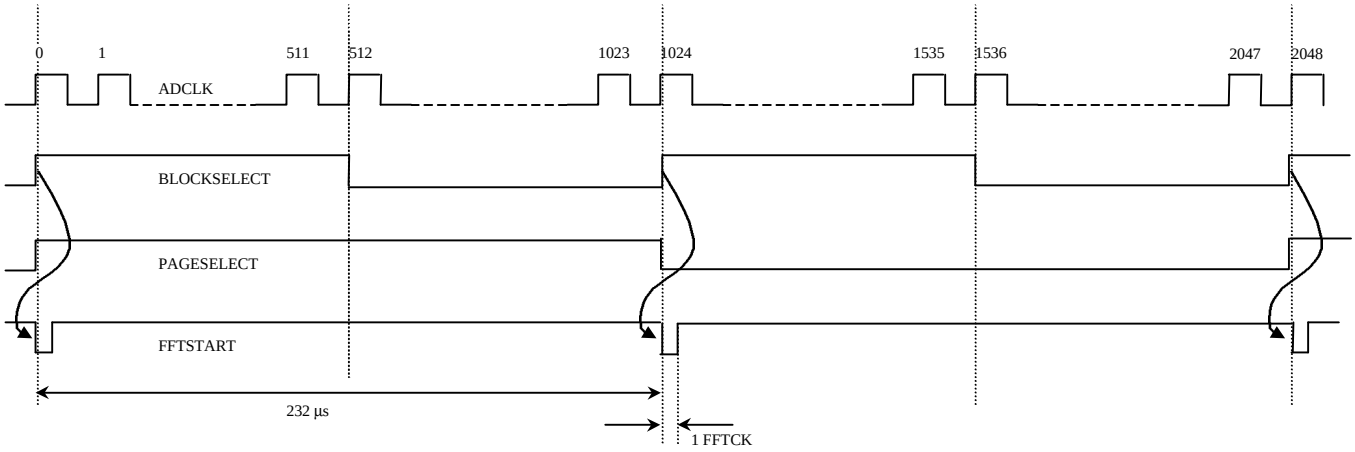


Figure 7.11 Digital timing diagram for ADC and input to FFT Core

7.3.2 FPGA Initialisation

During an initialisation period, serial data is transferred from SPROMs to the FPGAs to configure the devices. Xilinx manufacture specific SPROMs for different sized FPGAs.

7.3.3 PC Interface

Both a complex multiplication vector representing the physical line's frequency response and complex addition vectors for noise and crosstalk simulation must be supplied to the processing FPGA during a simulation run from a controlling PC. The 16-bit 2048 word addition vector (1024 real and 1024 imaginary) must be updated every time sampling window, whereas the same sized multiplication vector requires loading only once at the start of a simulation run. When a new line is to be simulated, a new multiplication vector must be downloaded describing the new frequency response. As previously described, an efficient method to generate self crosstalk is to use a delayed and attenuated copy of the actual ADSL signal's DFT itself. To allow complete control of self crosstalk, the complex 1024 computed ADSL frequency samples from the FFT Core should be uploaded to the PC during each time sampling window, then processed and downloaded as a constituent part of a later addition vector. Ideally a well defined interface should exist between the simulator board and controlling PC.

Shown previously in figure 7.7, dual port RAM can be used as a store and buffer between the PC interface and the DFT and processing FPGAs. The RAM used to store both the copy of the ADSL's DFT and processing addition vector should operate on a split page mode similar to that described previously to allow simultaneous read and write operations for consecutive data blocks from opposite sides of the dual port RAM. The multiplication vector could be stored in single port RAM as only one download per simulation run is required, but since dual port RAM will be used exclusively elsewhere, using an identical device will simplify timing and other circuitry.

Much of the device level logic will be very similar in nature to that shown in figure 7.10. To avoid excessive numbers of circuit level diagrams, only figure 7.7, the high level block diagram of the complete design is included.

7.3.4 Line Receiver and Transmitter Modules

The output from an ADSL modem will be at voltage levels required for twisted pair transmission. The final part of a transmitting modem contains an AFE that matches the line and power driver's impedance. The first part of a receiving modem will also contain an AFE matching its impedance to that of the transmission line's and operate at a suitable sensitivity. The input of the line simulator should be consistent with that normally seen by a transmitting modem and its output consistent with that seen by a receiving modem. This requires matching both impedance and signal levels.

Specific line drivers such as the Analog Devices AD816 differential driver[6], designed for use with ADSL modems, incorporate both transmitter and receiving electronics in a single device and are suitable for use as the simulator's line drivers.

7.3.5 ADC and DAC Converters

Both ADC and DAC must generate and convert parallel 16-bit data vectors. Two Analog Devices converters, the AD9240-EB DAC at £135 and AD768-EB ADC at £104 are both available on fully populated evaluation boards from the supplier SEI Millenium. The use of evaluation boards obviously reduces the design time, as board layout and interface circuitry is already optimised by the manufacturer.

7.3.6 DAC Output Filtering

Between the output from the DAC and line driver, filtering is necessary. From the scant available details of ADSL modems and AFEs, typically these filters are fourth order with cutoff frequencies of approximately 1.2 MHz[7,8]. Because receiving ADSL modems incorporate high order anti-aliasing filters with similar performance, there is no problem in limiting the spectral output from the simulator to 1.2 MHz with smoothing filters as anything above this is effectively removed by the receiving AFE before any A/D conversion takes place in the receiving ADSL modem.

For simplicity, a fourth order analogue Butterworth filler implemented using fast op-amps will be used to filter the DAC's output in the line simulator.

References

[1] Texas Instruments, TMS320C6701 Data Sheet SPRS067, May 1998.
Texas Instruments, TMS320C67x Single Precision Floating Point Assembly Benchmarks, May 1998.
Texas Instruments, TMS320C6202 Data Sheet SPRS072A, January 1998.
Texas Instruments, TMS320C62x Assembly Benchmarks, May 1998.

[2] Analog Devices, "A New Architecture for the Digital Convergence Infrastructure", TigerSHARC DSP Product Preview and Benchmarks, June 1999.

[3] Motorola, DSP56600 DSP Benchmarks, November 1996.

[4] SEI Macro Group Ltd, Component Quotation, 26th March 1999.

[5] "Estimating the Preformance of XC4000E Adders and Counters", Xilinx Application Note XAPP 018, Xilinx, July, 1996.

[6] Analog Devices, AD816 500 mA Differential Driver & Dual Low Noise (VF) Amplifiers, Data Sheet, 1996.

[7] ST Microelectronics, STLC60135 TOSCA ADSL DMT Transceiver, Data Sheet, May 1998.

[8] Fujitsu Microelectronics UK Ltd, MB86626 Keywave ADSL AFE, Data Sheet, December 1998.

‘Oliver wasn’t as altogether happy as the lucky pig that accidentally got locked into the malt house of the local brewery.’

Charles Dickens

Chapter 8

Revised ADSL Simulator Design

Although the words in Charles Dickens’s book Oliver Twist were written over one hundred years ago, they deftly conveyed my feelings when on delivery of the Foundation and Core Generator software I discovered that the FFT Core didn’t actually exist!

This chapter details the revised design of the line simulator using a Xilinx ‘Reference Design’ for a 1024 point FFT in place of the Core design. In addition to new memory interface circuitry between FPGA blocks, a time manipulation FPGA after the IFFT and a new complete AFE at the final design stage from Fujitsu including ADC, DAC and filtering are incorporated into the design.

8.1 Overall Revised Simulator Design

Figure 8.1 shows a block diagram of the overall revised line simulator design incorporating additional time domain manipulation and integrated AFEs. The revised design builds on the transform principles developed originally, with the only major change being the inclusion of the time manipulation block to simulate time domain modeled noise components.

In contrast to the FFT Core which split its real 1024 input data points into two blocks (t(0) to t(511) and then t(512) to t(1023)), the FFT Reference Design splits the real and imaginary components of both input and output vectors into two separate continuous blocks.

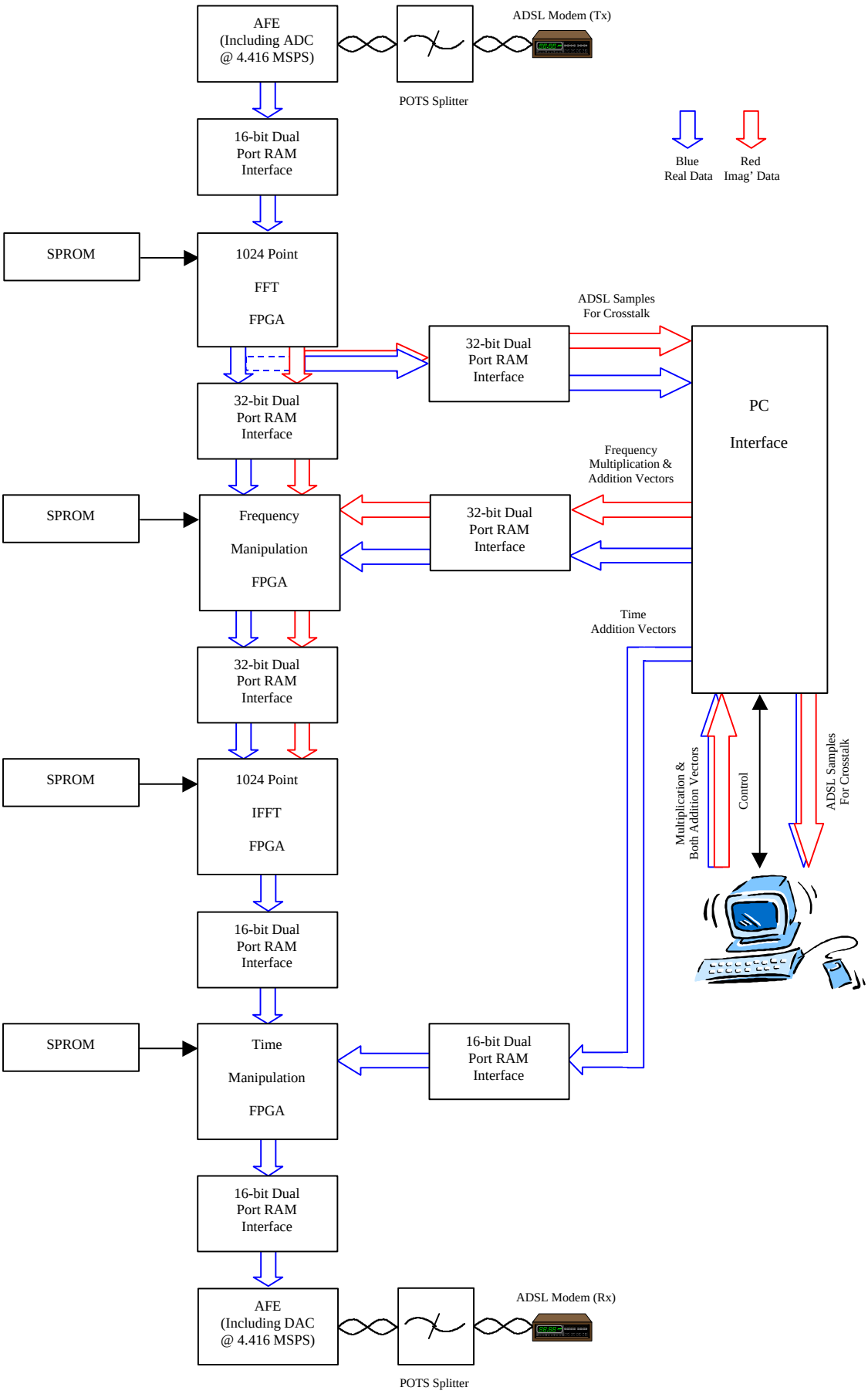


Figure 8.1 Revised ADSL line simulator block diagram

8.2 Xilinx Reference FFT Design

The full data sheet for the 1024 point FFT Reference Design from Xilinx may be viewed on the enclosed CD. The most salient first four pages are included in appendix 6. The most notable differences between the new Reference Design and the original FFT Core are as follows

- Radix 4 operation.
- Real and imaginary input and output data vector separation.
- The design requires two banks of 1 k-byte 32-bit external scratch pad RAM.
- Increased processing speed, reducing block processing to within 100 µs.
- Considerably increased CLB count, targeting XC4062 or larger devices.

Figure 8.2 shows the FFT Reference Design pin-out diagram.

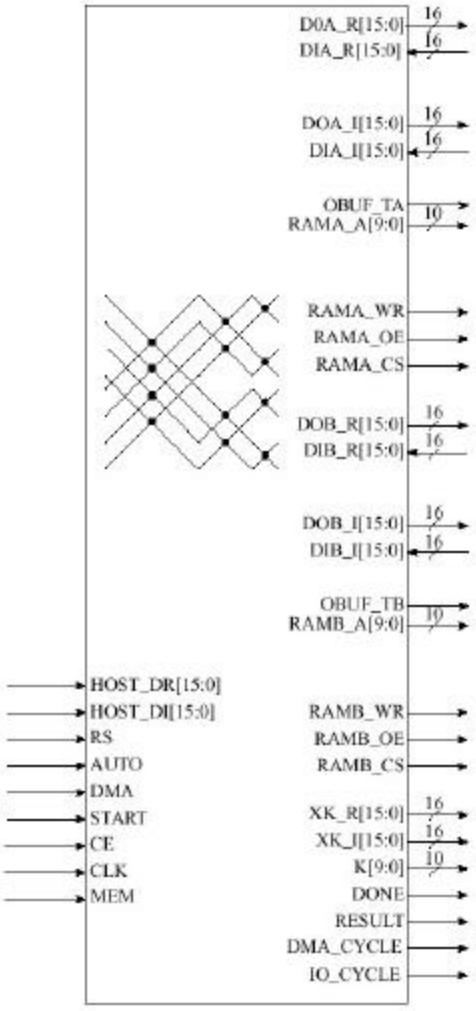


Figure 8.2 FFT Reference Design pin-out diagram

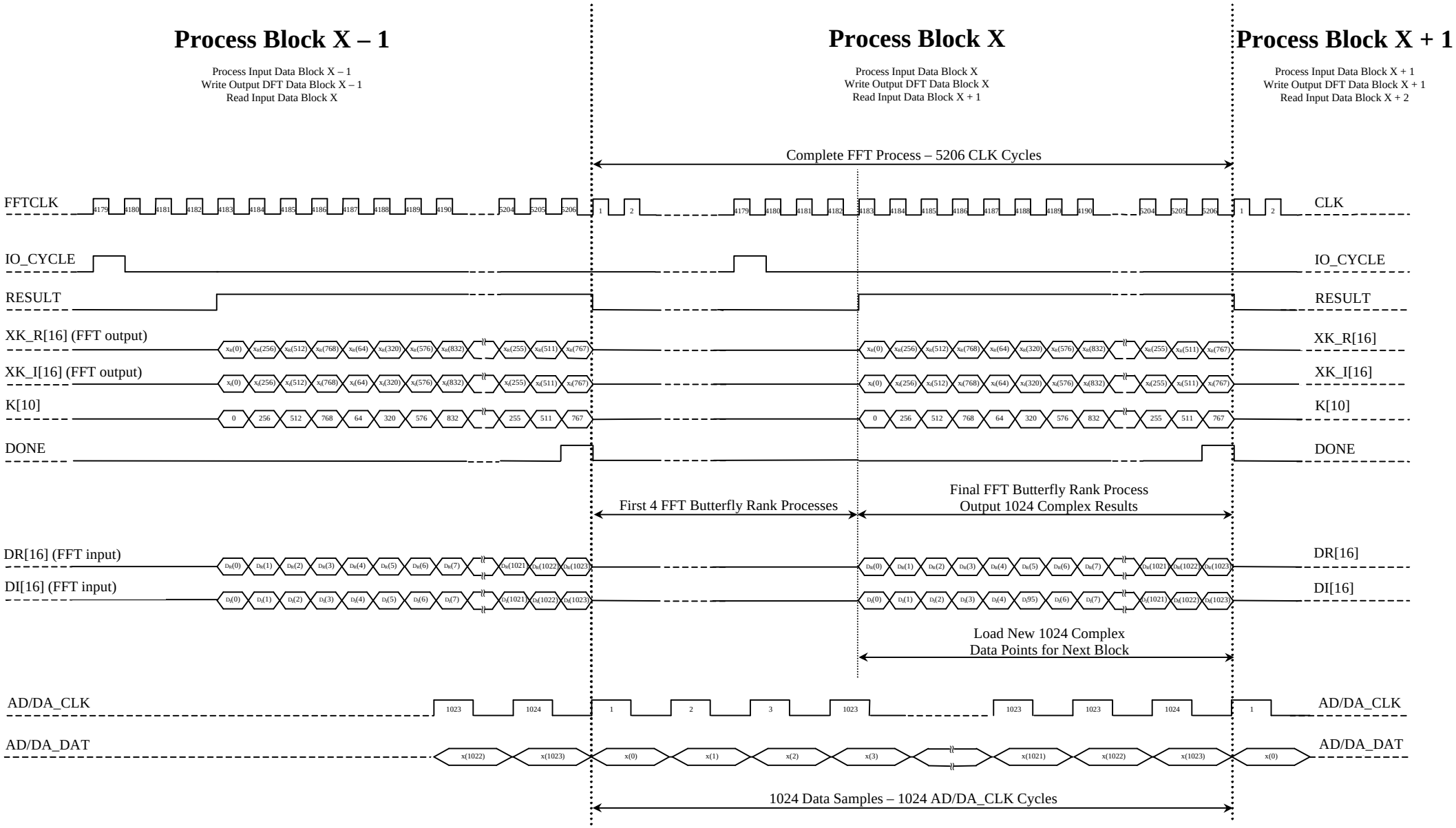


Figure 8.3 Timing diagram for FPGA FFT I/O, ADC and DACs

8.2.1 FFT Reference Design I/O and Control Timing

Figure 8.3 summarises the published I/O and control timing for the FFT Reference Design. The main points impacting the supporting circuitry are

- The FFT process consists of five Butterfly rank operations.
- The total FFT process requires 5206 FFTCLKS: 5 x 1024 Butterfly rank clock cycles, 17 clock cycles before each rank and one more at the start of each FFT operation.
- During the final rank process, the result is output whilst at the same time, data for the next FFT block operation is read into one of the scratch pad RAM blocks, alternating between blocks A and B on subsequent FFT transforms due to the odd number of Butterfly ranks.
- The real and imaginary components of each of the 1024 samples are accessed or output at the same time, thus requiring only one address bus for each block of scratchpad RAM and output data buses.
- The FFT doesn't address the input RAM, the latter is required to produce its data in chronological order and to the specified timing during the final Butterfly rank process, whereas the FFT actively addresses the data components of the output results through the use of a de-scrambling index bus, K.
- Periphery devices are notified of the FFT's need for new input data or its intention to write new results through the control strobes: DMA_CYCLE, IO_CYCLE, RESULT and DONE.
- The FFTCLK theoretically operates at upto 68 MHz, giving a clock period of 15 ns.

8.3 Internal Memory Interfaces

In the initial simulator design, the difference between time sample ordering and generation rate at the DAC's output and the required ordering and input rate to the FFT Core gave rise to the need for memory buffering between the two, operating on a pipelined block processing approach. Implementation using paged dual port RAM interfaces was developed. A similar situation occurs using the Reference Design. Conceptually, the full ordering and rate conversion scheme is shown in figure 8.4. Each page of dual port RAM contains 1024 16-bit locations.

8.3.1 Memory Interfaces - Justification

Fast dual port RAM is expensive, so should only be used where necessary. The following five sub-sections explain why dual port memory buffering is necessary between the ADC, FFT, frequency manipulation FPGA, IFFT, time manipulation FPGA and DAC. Reference to the top of figure 8.3 should be made for the control logic timing for the FFT Reference Design and the bottom of the same figure for general parallel flash A/D and D/A conversion processes when the FFT processing time and time sampling window are exactly the same.

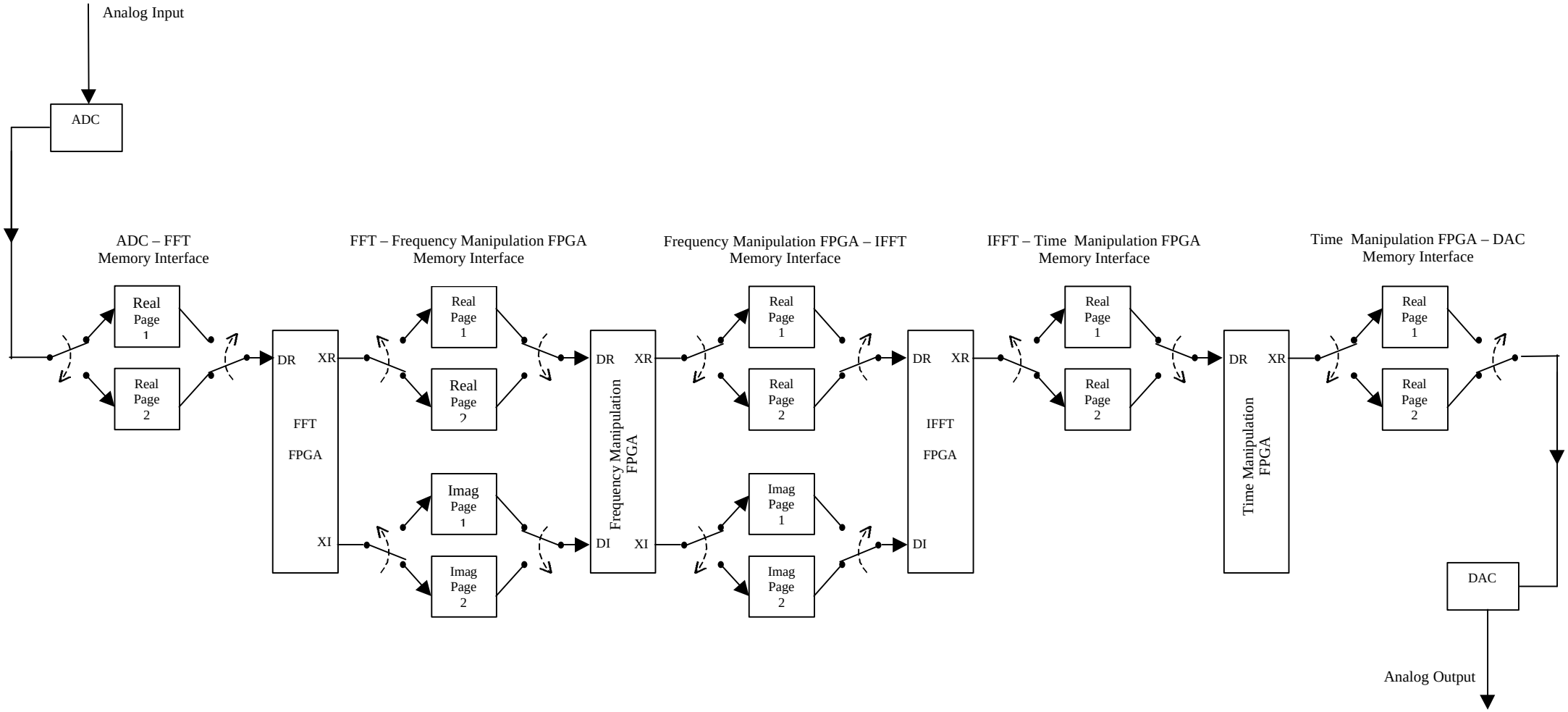


Figure 8.4 Conceptual memory page structure for ADC, DAC, FFT, IFFT and manipulation FPGAs
(All data lines 16-bit wide)

8.3.1.1 ADC – FFT FPGA Interface

From the ADC, time samples are output in chronological order (x(0), x(1),…,x(1023)) and generated at a rate of 1 sample every ADCLK (4.416 MHz), bottom of figure 8.3. The FFT Reference Design requires its 1024 point input data vector split into real and imaginary components then also reads them in chronological order, one sample every FFTCLK (≈50 MHz), top of figure 8.3. The real – imaginary split isn't a problem between the ADC and FFT FPGA as the time samples are purely real, but the difference in ADC write and FFT read rates requires sample rate conversion through a memory interface with pipelined block processing.

8.3.1.2 FFT – Frequency Manipulation FPGA Interface

Unlike common radix 2 bit reversed output ordering, the real and imaginary radix 4 FFT output frequency samples are in the digit reversed order[1] of the form

$$A[10] = [a_1, a_0, a_3, a_2, a_5, a_4, a_7, a_6, a_9, a_8] \qquad a_i \in \{1, 0\}, \quad i = 0, 1, ..., 9$$

This order is shown in figure 8.3. These samples are read by the processing FPGA and, if not for the mismatch in inter-sample FFT write time and manipulation processing delay, could be manipulated in the order they are produced by the FFT FPGA. The low level designs of chapter 9 will show that 16-bit multiplication and addition takes approximately 114 ns using Core multipliers and adders (8 PRIMARY_CLKs). However, frequency samples from the FFT FPGA are written once every 29 ns (1 35 MHz FFTCLK) during the last 1024 FFTCLKs of the complete 5206 cycle FFT operation, figure 8.3. Therefore, the frequency samples cannot be manipulated at the rate the FFT FPGA writes them, so memory buffering for pipelined operation is also required between the FFT and manipulation FPGAs.

8.3.1.3 Frequency Manipulation – IFFT FPGA Interface

The (I)FFT Reference Design requires chronologically ordered input vectors, (X'(0), X'(1),…,X'(1023)) and reads one sample per FFTCLK during the final 1024 IFFT cycles, figure 8.3. Because the rate at which the manipulation FPGA can write modified frequency samples is limited by the total manipulation delay of 114 ns, pipelined block processing is required for the manipulation process and IFFT, with a similar paged dual port RAM solution between the two FPGAs.

8.3.1.4 IFFT – Time Manipulation FPGA Interface

The output from the IFFT will be purely real (unless multiplication of the positive and negative frequency samples is by non complex conjugate pairs which would result in complex time samples which can't be converted to a physical time signal by the DAC anyway) and in the radix 4 order. The output from the IFFT includes a sample index bus (K in figure 8.3) specifically to de-scramble the

IFFT output vector, so reordering should occur immediately after the IFFT, justifying the IFFT time manipulation memory interface.

8.3.1.5 Time Manipulation FPGA – DAC Interface

The DAC requires chronologically ordered samples, spaced by 1 DACLK (equal to the ADCLK). Re-ordering has already been accomplished and sample output rate conditioning could also be done through careful timing via the previous memory interface, but without a memory buffer implementing the block processing approach between the time manipulation FPGA and DAC, the length of time that samples can spend being processed in the FPGA will be constrained by the DAC data input requirement of evenly spaced samples at the rate of one per DACLK. In future design developments, depending on what new operations are configured for the time manipulation FPGA, the time processing delay may increase from that of simple addition. In order to allow maximum flexibility, a memory buffer is included between the time manipulation FPGA and DAC, allowing the whole 232 μs block processing window to be available for manipulation operations on the 1024 real time samples.

8.3.2 Practical Memory Implementation

Practically, the page structure for each of the five interfaces can be implemented in a variety of ways. Since the real and imaginary components of each vector are always read or written in the same order and at the same time by any of the four FPGAs (e.g. for the dc frequency component, the FFT FPGA writes $X_R(0)$ and $X_I(0)$ at the same time, figure 8.3), only one address bus for each side of each memory interface is required for both real and imaginary memory blocks. One complex component can be stored in the least significant 16-bits and the other in the most significant 16-bits of each 32-bit memory location. Table 8.1 shows some possible RAM implementations. Figure 8.5 shows the page structure for complex data vectors using a single 2048 location, 32-bit dual port memory chip and figure 8.6 for a single 2048 location, 16-bit memory chip for the real valued data vectors.

ADC – FFT Interface **IFFT – Time Manipulation Interface** **Time Manipulation – DAC Interface** **(real data only)**	**FFT – Frequency Manipulation Interface** **Frequency Manipulation – IFFT Interface** **(real and imaginary data)**
2 x 1024 locations, 16-bit	2 x 1024 locations, 16-bit (for real data) 2 x 1024 locations, 16-bit (for imag data)
	2 x 1024 locations, 32-bit (16 MSBs for real data) / (16 LSBs for imag data)
1 x 2048 locations, 16-bit	1 x 2048 locations, 32-bit (16 MSBs for real data) / (16 LSBs for imag data)

Table 8.1 Memory interface physical RAM implementations

2048 x 32-bit Dual Port RAM Chip

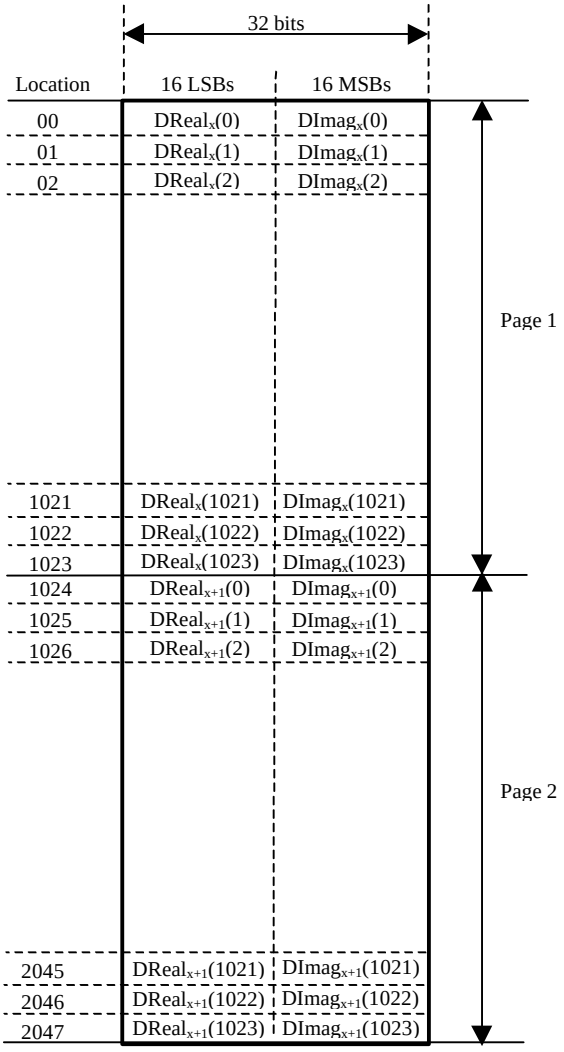


Figure 8.5 32-bit memory interface page structure for real and imaginary data vectors

2048 x 16-bit Dual Port RAM Chip

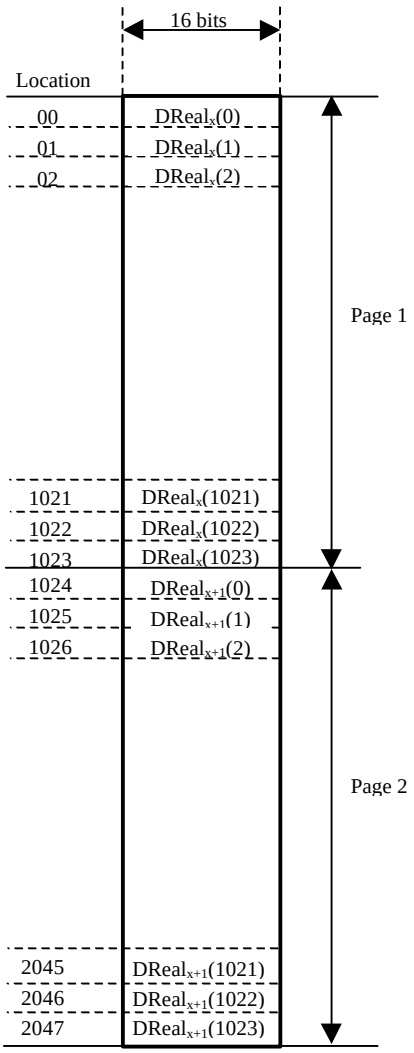


Figure 8.6 16-bit memory interface page structure for real valued data vectors

8.3.3 Memory Interfaces for PC I/O

The I/O requirements for PC control are similar to the initial design, with an extra set of 1024 point real addition vectors for the time domain manipulation required from the PC every 232 µs sampling period window. In order to allow data to be read by the FPGAs during the current processing time window whilst the PC downloads values for the next, dual port RAM buffering offering paged operation similar to that previously described can be used. The memory interfaces for PC I/O also provide sample rate transfer control to both manipulation FPGAs, simplifying the download timing demands on the PC. With paged memory buffering, the PC only has to download the full set of 512 complex conjugate multiplication coefficients, 512 complex conjugate frequency addition components and 1024 real time addition components within each 232 µs sampling window. The PC doesn't have to download the data

to satisfy the specific timing requirements of each of the manipulation blocks, nor does it have to upload the crosstalk samples from the FFT block according to their rate of production.

The great advantage of incorporating these PC I/O memory interfaces is to allow future work to define the timing for the PC interface independently of the simulator's internal processing operations. Figure 8.1 also shows all the necessary components for PC I/O, which can be practically implemented as shown in figure 8.5 and 8.6 for the complex and real data vectors.

8.3.4 Page Addressing for Memory Interfaces

In chapter 7, memory page selection was accomplished very neatly through simply deriving a PAGESELECT signal from the ADC clock. For two 1024 location memory pages (2048 location RAM chip), the ADCLK should be divided by 2048 and this signal used to drive the most significant address pin of one side of the 2048 location RAM chip with the other side's most significant address pin being driven by its logical inverse, shown below in figure 8.7 and 8.8. With this configuration, the opposite sides of the dual port RAM are always being written to, or read from, different pages in the paged storage space. Chapter 9 will present a low level design capable providing synchronous ADC, DAC, FFT, IFFT and PAGESELECT clocks from a single master oscillator.

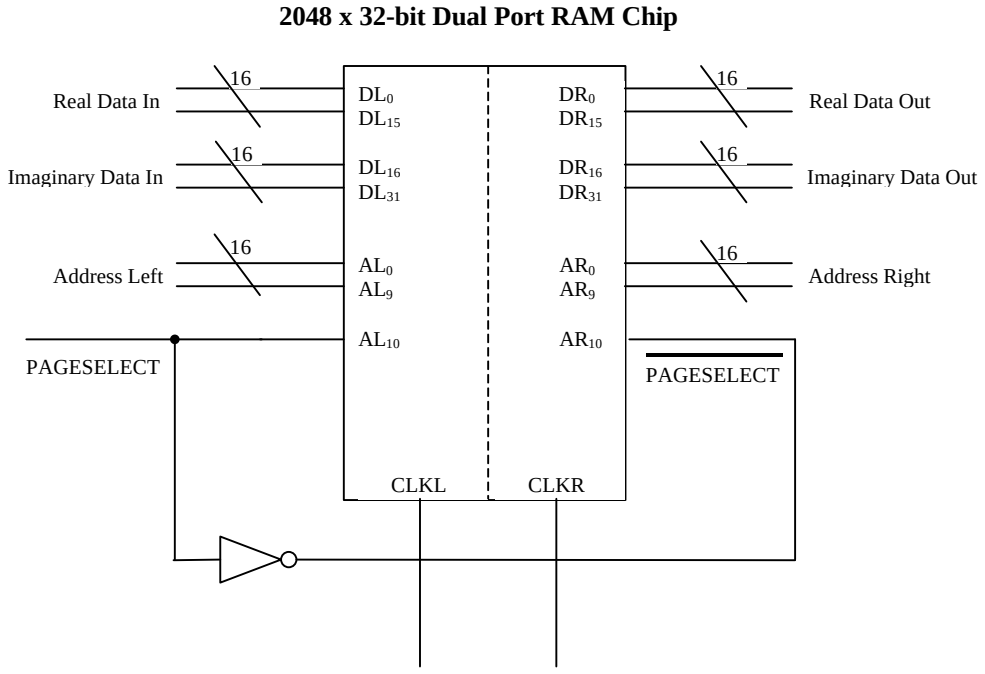


Figure 8.7 Page addressing for 2 k-word dual port RAM

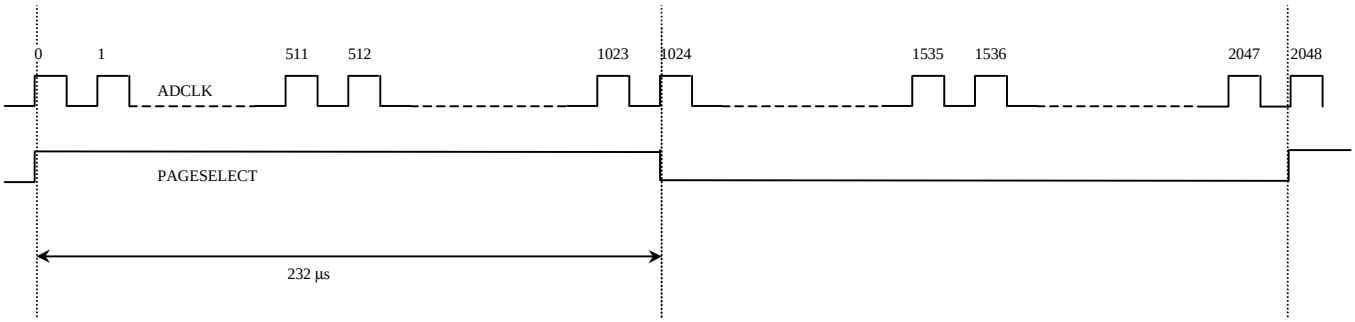


Figure 8.8 Digital timing diagram for memory page selection from the ADC clock

8.4 Processing Operations' Timing

Each of the six processing operations (A/D conversion, FFT, frequency manipulation, IFFT, time manipulation, D/A conversion) will be driven by a different frequency clock and will each require a different number of cycles of that clock to complete. That is, the FFT and IFFT operations will be driven by a clock at around 50 MHz and require 5206 cycles, the A/D and D/A conversion operations will be driven by a 4.416 MHz clock and require 1024 cycles whereas the manipulation processes will be driven by clocks at around 100 MHz and require a different number of cycles to complete depending on the internal logic design. Once memory interfaces are placed between each of the six processing blocks, the blocks can operate separately within each time sampling window period. The only constraint on each operation is that it must be completed before the start of the next time sampling window. This greatly eases logic design as each block can be designed to function independently of the others' internal timing. Figure 8.9 shows how each operation can take different periods to complete, but all within the fixed 232 µs time sampling window and also shows the passage of blocks of time samples through the simulator's six functional operations. Clearly, a time sample taken by the ADC at time T_0 will be leave the simulator from the DAC 5 sampling windows later.

As mentioned, the operational isolation by memory interfaces allows each functional block to be driven at its own internal rate, allowing future development of the manipulation blocks without impacting on the conversion and transform blocks. A single external high frequency master clock can be divided to drive the AFEs at 17.664MHz, the FFT and IFFTs at around 50 MHz (depending on final logic delays within the placed FPGA). And the manipulation blocks' clocks at around 100 MHz, again depending on the net delays within the FPGAs they are implemented in. All clocks should be chosen to be multiple of the AFE's 17.664 MHz clock to allow simple generation by division of the master.

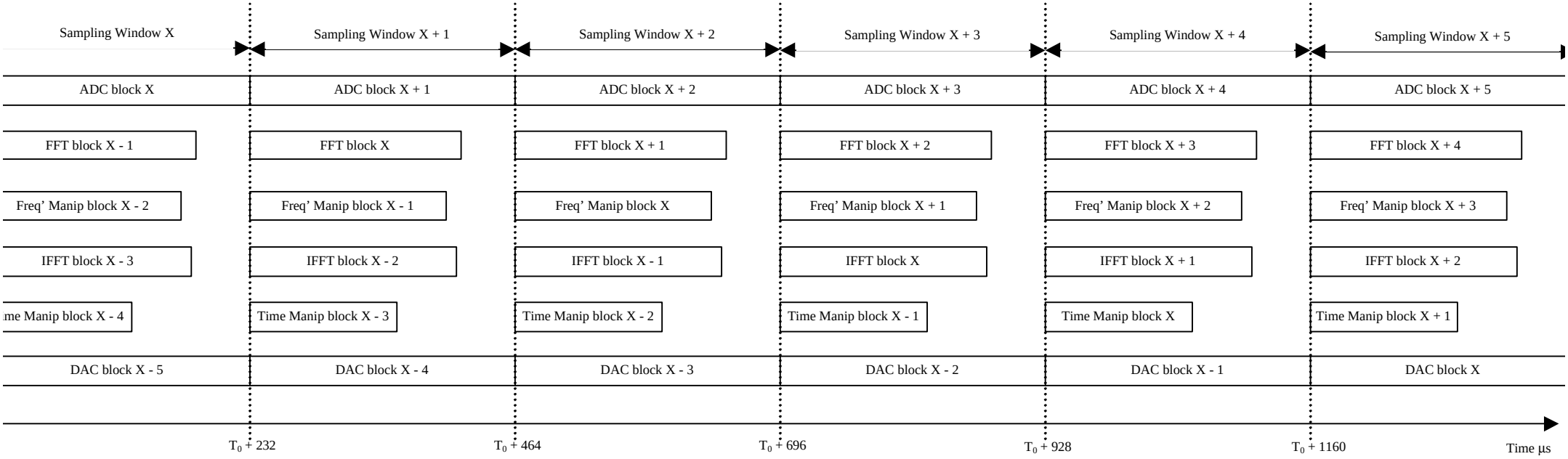


Figure 8.9 Independent operation timing for all six operations over a period of 6 time sampling windows

8.5 Analogue Front Ends

Shown in appendix 3 is the advance product preview of the Fujitsu KeyWave AFE. At present the full data sheet is commercially confidential, but the device is known to be suitable for use in the line simulator. The pin-out is shown below in figure 8.10.

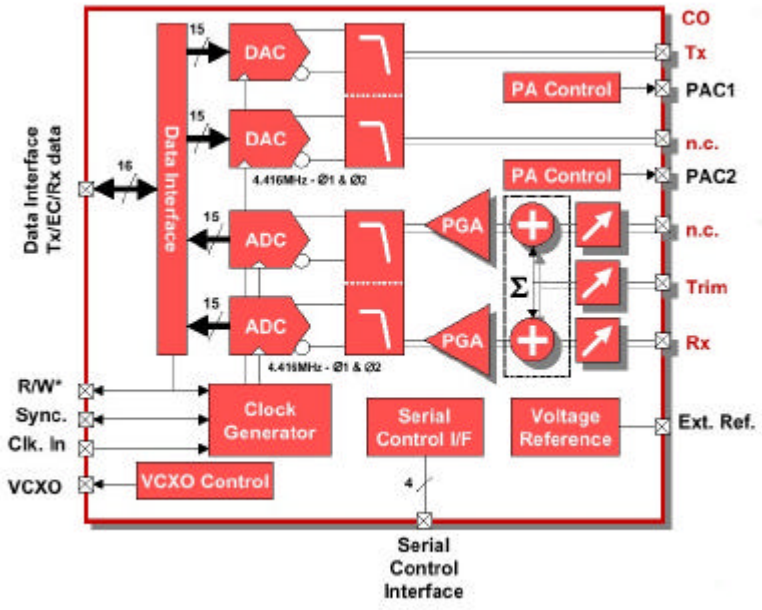


Figure 8.10 KeyWave pinout diagram

From the product preview, it is apparent that the IC:

- Is designed as a complete solution for both receivers and transmitters.
- Contains programmable low pass filters with cutoff frequencies upto 1.2 MHz.
- Has dual internal 16 bit 4.416 MSPS ADCs and DACs which can be configured to sample at 4.416 or 8.832 MSPS.
- Has an active rising edge read – write strobe for writing and reading to and from external memory
- Requires a 17.664 MHz external clock signal for timing.
- Only requires an external power line driver and hybrid transformer for connection to a twisted copper pair.

The Keywave AFE is ideal for both the receiver and transmitter front ends of the line simulator.

References

[1] E. Oran Brigham, “The Fast Fourier Transform and Its Applications”, Prentice Hall, 1988, p140.

Chapter 9

Detailed Low Level Design and Functional Testing

This chapter deals with the low level logic design of the four operations implemented with FPGAs: the FFT, frequency manipulation block, IFFT and the time manipulation block. Circuitry to implement all the required functionality is developed including memory addressing and clock division with the exception of start-up and SPROM initialisation circuitry. Where possible, logic functionality is checked and post device place and route performance determined. All designs were completed, verified and timed using the Xilinx Foundation 1.5i and Core Generator software. In addition some consideration is given to the required memory interface chips with suitable components identified.

The designs presented are in no way the only viable solution. Indeed experienced FPGA logic designers would be expected to arrive at considerably different and better solutions. Instead they should be viewed as an initial starting point for further development.

9.1 FFT FPGA

This FPGA performs a FFT on the 1024 time samples stored in the ADC – FFT memory interface each time sampling window. The 1024 DFT frequency sample results are written to the FFT – frequency

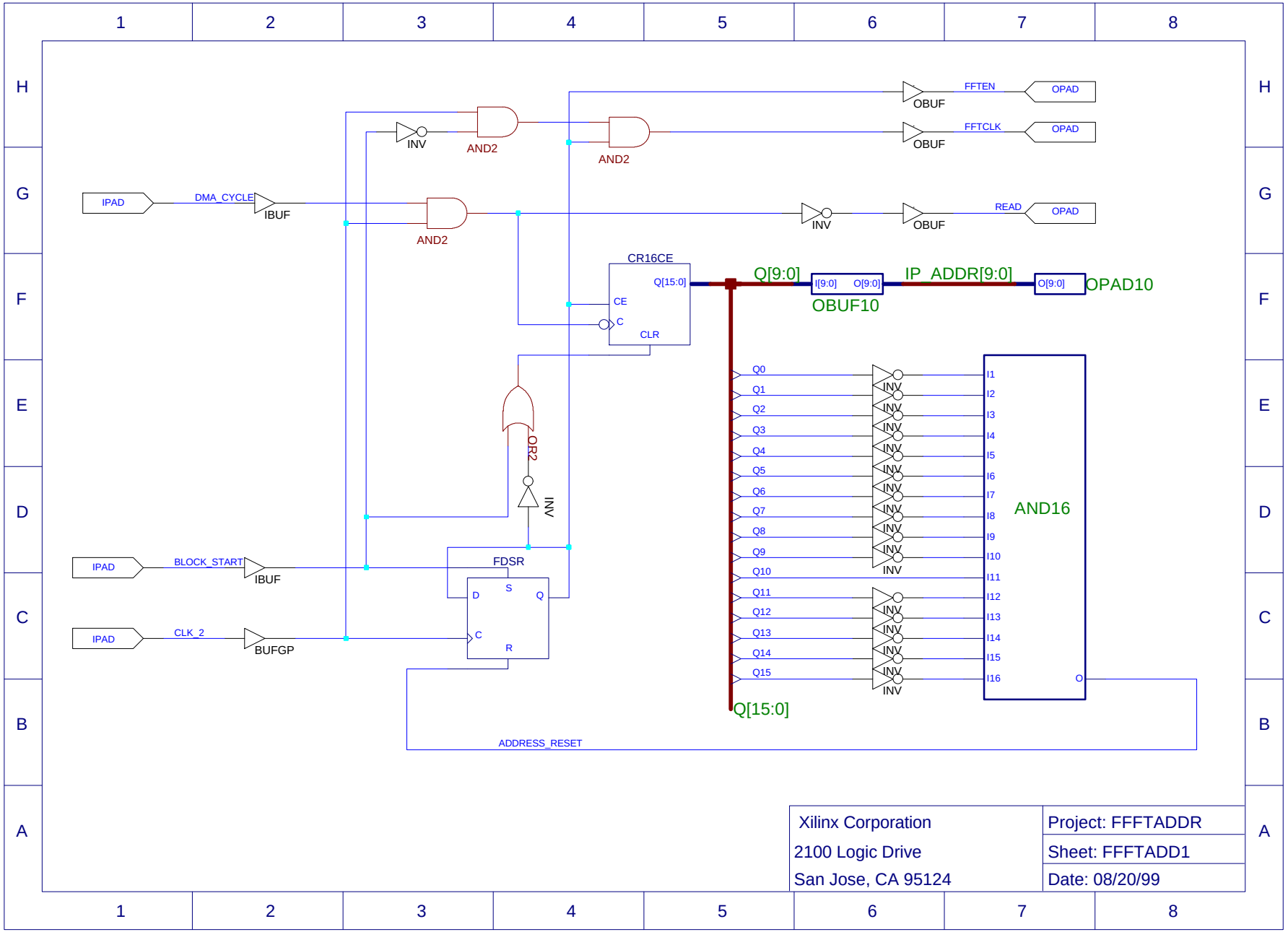


Figure 9.1 FFT additional control circuitry for real time data input from the ADC - FFT memory interface

manipulation FPGA memory interface.

All FFT FPGA schematic, simulation and associated files are located in the `:\Designs\Xilinx\Fft \fftref \fftaddr` and `\fftmem` subdirectories on the enclosed CD ROM.

9.1.1 Peripheral Circuit Requirements

As mentioned in chapter 8, the FFT Reference Design doesn't actively address its input memory. Logic is required to provide a read strobe and the address bus to the memory interface between the ADC and FFT, which serves the same purpose as the host data server shown in the Reference Design data sheet (on CD ROM). Page selection for the memory interfaces is external to the FFT FPGA design.

9.1.2 Logic Design

The FFT Reference Design produces a strobe called DMA_CYCLE when new input data is required during the last Butterfly rank process, which lasts for the duration of the transfer then goes low. This signal can simply be gated with the FFTCLK to produce a read strobe, which will be active only when the DMA_CYCLE is asserted high by the FFT. The rising edge of the DMA_CYCLE signal can also initiate a binary counter for the address bus, which is reset on a count of 1024. Figure 9.1 shows circuitry to produce the necessary address bus and control signaling for time sample input data from the ADC – FFT memory interface.

In addition to the input control timing circuitry, the FFT block requires its scratchpad RAM data and address buses to be synchronously registered. The data buses must also be converted to bi-directional operation for interfacing to external tri-state RAM. Figure 9.2 showing the complete FFT FPGA design, includes two macros TRI_BUFF_32 and OFDX10 to achieve this.

9.1.3 FFT FPGA Design Verification

After the problems encountered with the Core FFT, it seemed wise to verify the operation of the Reference Design before finalising its use within the simulator. Operational testing can be carried out in three parts:

- Firstly, verify that the control signal timing corresponds to that shown on the published data sheet.
- Secondly, verify that the FFT transform function is correct through the transform of a 1024 point real input data vectors and comparing the results with those generated for the same input vectors by a numerical package such as Matlab.
- Thirdly, verify timing and addressing for the input RAM circuitry.

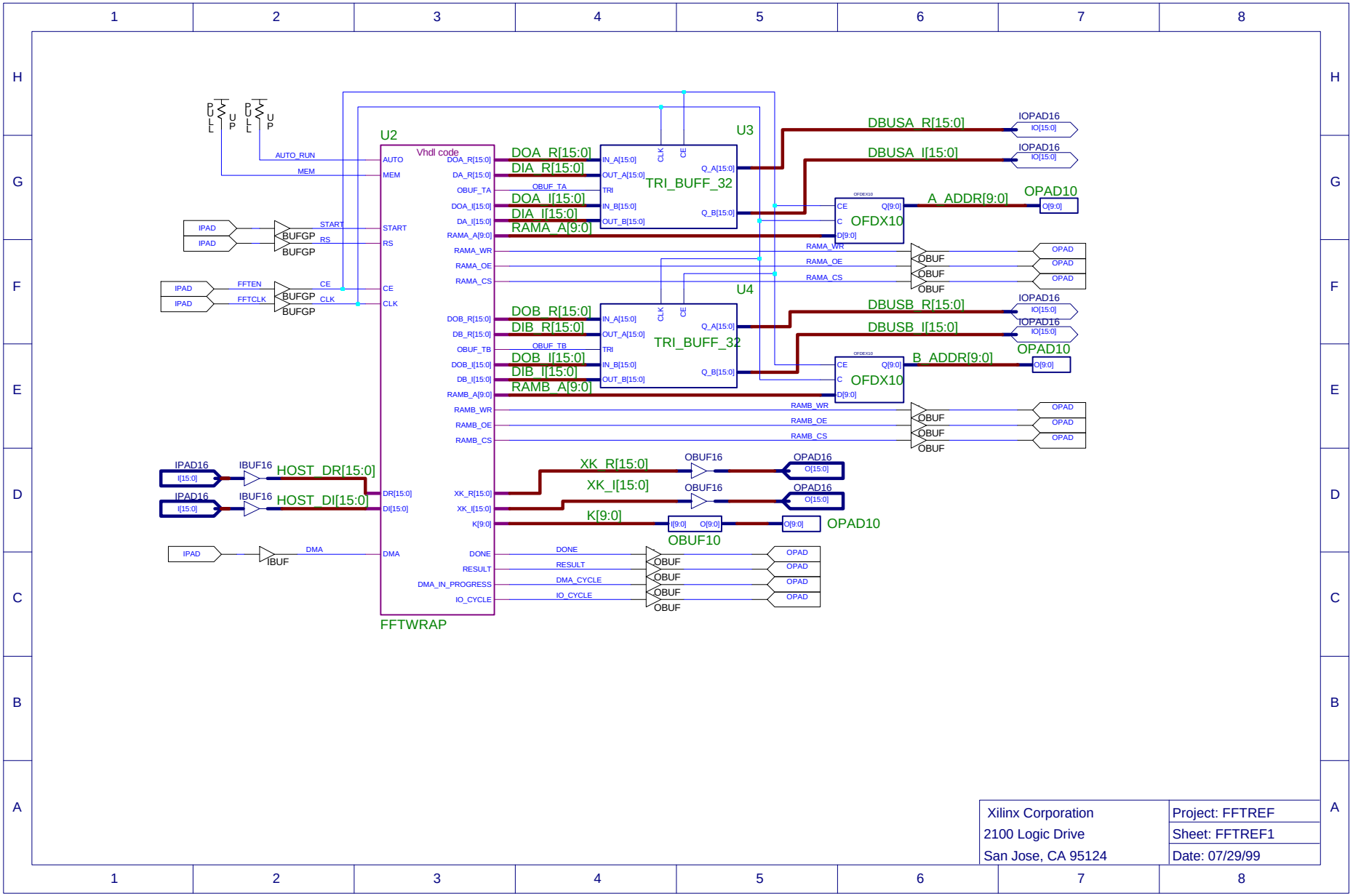


Figure 9.2 Complete FFT transform engine design

Unfortunately, the FFT Reference Design doesn't produce a behavioural model for the function, so only gate level simulation after the generation of a supplementary EDIF file is possible. This approach is extremely time consuming due to the very large design. From circuit modification to completion of a basic single 1024 point FFT simulation without scratchpad RAM, a processing time in excess of four hours is encountered using a PII 300 MHz processor. In addition, no suitable memory blocks of 1024 x 32-bit RAM for scratchpad use don't exist in Xilinx libraries, so FFT transform functionality can't be cleanly tested without custom design of suitable memory blocks for inclusion in a schematic design. Because of the large size of such memory blocks (2 x 1024 by 32-bits), the simulation time would be enormous. An alternative to inserting scratchpad RAM into the schematic design is to verify the FFT transform through a complex simulation command file which mimics the behaviour of the RAM. The command file must record temporary process data output from the FFT Butterfly ranks on complex data buses DBUSA and DBUSB (figure 9.2), then re-inject it into the FFT engine as the scratchpad RAM would do when addressed by the FFT engine in an actual physical circuit. There is no easy way to do this with the Xilinx Foundation software other than by observing the address and data buses of the scratchpad RAM through each Butterfly rank process and then copying the values into the command file for re-injection to the FFT engine in the following rank process, carefully noting the read/write ordering from the address buses. This is extremely laborious and took over a day to write the command file required to test a single complete 1024 point FFT transform.

9.1.3.1 FFT Reference Design Control Timing

Control timing verification is by far the easier of the three verification steps. The command file `fftref_control_verification.cmd` in the `fftref` directory on the CD contains a command file written to simulate an initial data vector load into the FFT engine, then a complete 5206 clock cycle 1024 point, followed by another 5206 cycles to verify the repetitive nature of the control timing for a second FFT operation. The simulation can be re-run or the results viewed from the file `fft_ct_v.tve` as simulation file takes about an hour to complete. Close examination of the control strobes and address buses confirm the published control timing and addressing.

9.1.3.2 FFT Reference Design Discrete Fourier Transform

As previously mentioned, transform functional verification is not easy in the Foundation design environment. The first method of including scratchpad RAM into a design schematic with the FFT Reference Design was initially attempted. Custom 1024 by 32-bit RAM was constructed from primitive library components and is included in the `fftmem` subdirectory on the CD, but is far too large and numerous to print! Unfortunately, this approach wasn't successful due to compilation faults when running the simulation software – this may be due to the shear size of the combined FFT block and memory as the Foundation software is written for designs which can fit into a single FPGA.

The second method of writing a complex simulation command file to mimic the action of the scratchpad RAM was reluctantly embarked upon. Due to the time consuming nature of this approach, only one transform has been tested; that of a simple sampled sinusoidal signal. Ideally the transform for

many different input sample vectors should be simulated and each result compared with that generated by Matlab for the same input vector before the functional integrity of the transform can be assured.

The simulation command file `fftref_transform_verification.cmd` contains the script to simulate the FFT on a sampled sinusoid, with the scratchpad RAM data written by the FFT engine during the transform stored in the files `A_BR1.dat`, `B_BR2.dat`, `A_BR3.dat`, `B_BR4.dat` and `A_BR5.dat`. The associated waveforms are stored in the file `fft_tf_v.tve`. Again, the simulation takes about an hour to complete. The Excel file `fftref_transform_verification.xls` tabulates the data written to the scratchpad RAM by the FFT engine during the five Butterfly rank processes in the order in which it is read back into the FFT, as well as the final transform result.

The simulation is for the FFT of a real set of 1024 time samples from a sinusoid with 128 cycles within the 1024 point sampling window, of the form

$$x(kT_S) = [0,\ 0.7071,\ 1,\ 0.7071,\ 0,\ -0.7071,\ -1,\ -0.7071,\ 0,\ 0.7071,\ 1,\ 0.7071,\ 0,\ -0.7071,\ -1,\ -0.7071\ ....]$$

where the sinusoid time samples shown repeat 128 times within the time sampling window. The DFT of such a signal should be purely imaginary with just two non-zero components of 0.5i at frequency sample +128 and –0.5i at sample 896 (equivalent to –128 with a 1024 point transform). **Examination of the FFT output vectors on data buses XK_R and XK_I from the simulation waveform shows that the FFT Reference Design's result doesn't match the correct values for the DFT of an input vector describing the sampled sinusoid**. Instead the FFT Reference Design seems to produce a set of 32 non-zero imaginary frequency samples of +0.5i at frequency samples 192 through 207 and –0.5i at samples 384 through 399. This is somewhat difficult to explain. The result, which contains only imaginary components, is consistent with the correct DFT for a real, odd time function such as the sine wave. Even the nature of the imaginary result seems indicative of a DFT having occurred (only two imaginary magnitudes of 0.5i and 0i), but the placing and sixteen times replication of the non-zero samples are incorrect.

Contact with the Design's author, Dr. Chris Dick of Xilinx, hasn't resolved the problem. It seems he hasn't done a complete simulation using the Foundation tools for the reasons iterated concerning scratchpad memory requirements and behavioural models. Dr. Dick has performed a transform using a Viewlogic simulation tool called 'Viewsim', unavailable to myself. A copy of the e-mail from Dr. Dick is included in appendix 7 and worth reading. The author suggests a new FFT design targeted at the Virtex series of devices instead of continuing with the XC4000 Reference Design. The Virtex FPGAs really are enormous, with upto 6 144 CLBs and 130 kbits of RAM compared to the largest XC4085XLA device containing 3 316 CLBs and no additional RAM. Virtex gate counts exceed 1 million compared to just over 100 000 for the XC4082XLA. The use of Virtex devices would solve all the simulation problems as a single design could also include scratchpad RAM. Unfortunately, the Virtex FFT design hasn't been released to date, but potentially provides a good transform solution.

9.1.3.3 Input RAM Addressing and Control

The command file `fft_input_address_verification.cmd` in the `fftaddr` subdirectory

simulates two complete time sampling windows. The address and timing waveforms may be viewed are stored from the file `fftaddr.tve`.

9.1.4 FFT Performance

Although actual transform functional verification hasn't been achieved, performance has been evaluated both before and after FPGA device placement and routing. Pre-routing performance is determined from the logic delay and estimated path delays within the design. Post routing performance includes the actual net delays encountered when the design has actually been placed within a target FPGA. Using the fastest XC4082XLA-07HQ240 device the maximum FFT clock frequencies are:

Timing Analysis	Maximum Clock Frequency
Pre place and route	69 MHz
Post place and route	42.3 MHz

Table 9.1 Pre and post place and route FFT performance

The large difference in pre and post performance is due to the absence of a User Constraints File (UCF) during the Foundation place and route operation. UCF files tell the Foundation design compiler which paths need to be timed 'together', which can be ignored and which are most critical. It is somewhat surprising that no UCF file is supplied with the FFT Reference Design. Because the design is only shipped as overall FFT macro and doesn't include the basic files (the net level files) from which the overall block is constructed, it is difficult to write a UCF file without guessing which constraints to apply to the paths identified in the Foundation timing reports. However, with trial and error, the post place and route clock frequency should approach the limit of 69 MHz. Even at 42 MHz, the complete 1024 point FFT execution time is just 125 µs (5205 x 24 ns), clearly within the 232 µs time sampling window. If future development for VDSL line simulation use the FFT Reference Design, an optimal UCF to maximise the FFT clock frequency at 69 MHz will be required giving an execution time of 75 µs.

9.1.5 IFFT FPGA

As the functional integrity of the FFT Reference Design hasn't been proved, the design of the IFFT module has been left to further work. The internal structure of the IFFT logic will be very similar to that for the FFT module taking into account appropriate FFT / IFFT scaling.

9.2 Frequency Manipulation FPGA

Each time sampling window, the frequency manipulation FPGA multiplies the 1024 16-bit complex DFT frequency samples with 1024 16-bit complex coefficients then adds the result to another 1024 16-bit complex numbers, downloaded from the controlling PC in the previous sampling window.

All schematic, simulation and other files for the frequency manipulation FPGA are located in the `:\Designs\Xilinx\frqmanip` directory on the enclosed CD.

9.2.1 Peripheral Circuit Requirements

As shown in figures 8.3 and 8.4, the 1024 DFT frequency samples are stored in sequential order in paged dual port RAM. The complex samples are separated into real and imaginary parts, each complex component occupying 16-bits of a 32-bit wide memory word. The manipulation block reads and modifies these samples then writes them to its output memory interface. The manipulation block must provide addressing for the input and output RAM from which the frequency samples are read from and written to. In addition, the block must address the PC memory interfaces holding the multiplication and addition vectors downloaded in the previous sampling window from the PC. Memory page selection is external to the manipulation block.

The three memory interfaces containing input data for the frequency manipulation process must be sequentially addressed and strobed with a read pulse. The memory interface to which the results are to be written must be similarly addressed and provided with a suitably timed write strobe after each manipulation result is available. Figure 9.3 shows the address and read/write strobe timing for frequency manipulation process.

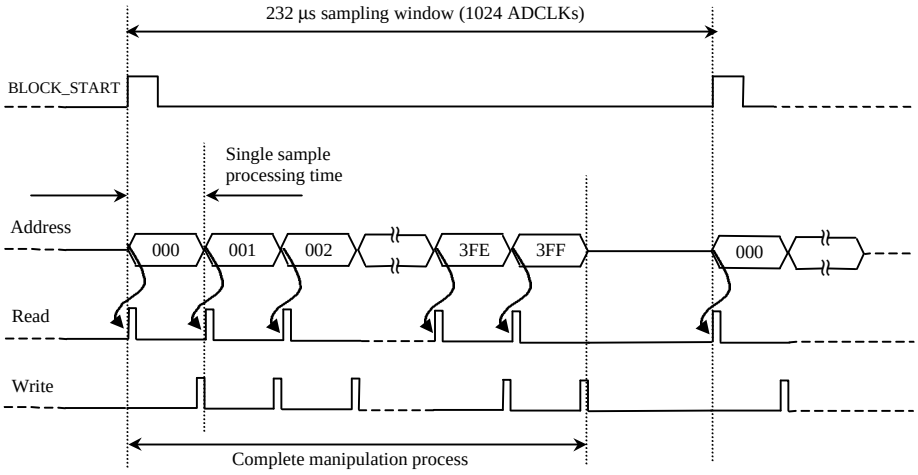


Figure 9.3 Frequency manipulation addressing and read / write strobe

9.2.2 Logic Design

9.2.2.1 Arithmetic Functions

The two arithmetic functions of addition and multiplication can be implemented using the Cores described in appendix 5. Two sets of 16-bit operation can be combined in parallel to allow the separate real and imaginary complex components to be manipulated at the same time, shown in figure 9.4. The eight data buses are as follows:

A_LO – Real DFT samples (input)
B_LO – Real multiplying coefficients (input)
C_LO – Real addition components (input)
A_HI – Imaginary DFT samples (input)
B_HI – Imaginary multiplying coefficients (input)
C_HI – Imaginary addition components (input)
RESULT_LO – Manipulated real DFT sample (output)
RESULT_HI – Manipulated imaginary DFT sample (output)

Attenuation is achieved through the method of ignoring the 16 LSBs of the result of the 16-bit multiplication of A and B. No account of carries produced by the addition process is included because this should never actually occur in practical use.

Each multiplication requires five clock cycles, with additions just one. Therefore, the result write strobe must occur at least six cycles after the read strobe.

9.2.2.2 Timing Functions

Each sample manipulation requires at least six clock cycles to perform. However, if eight cycles are allocated to each operation, the read and write strobe signals are easily derived by dividing the manipulation clock by eight (2^3). Each new 1024 sample frequency manipulation process is initiated by the assertion of the global BLOCK_START timing strobe, figure 9.3. After this strobe, the manipulation timing logic must produce 8 x 1024 (8192) clock cycles to complete the operation before the start of the next sampling window.

The required number of clock cycles are produced by gating the external clock with a control signal which is set high on the falling edge of the BLOCK_START signal and reset low after the 8192 cycles have been completed. A simple 16-bit library binary counter is used to count the number of cycles, with bits 4 to 13 acting as the address bus (effectively the clock divided by eight and counted). A 16-bit AND gate is used to generate the ADDRESS_RESET pulse after 8192 cycles have occurred. The read strobe is produced at the start of every eight clock cycles simply by dividing the manipulation clock by eight, and the write edge derived from the output from the third bit of the binary counter. Figure 9.5 shows logic to produce these timing signals.

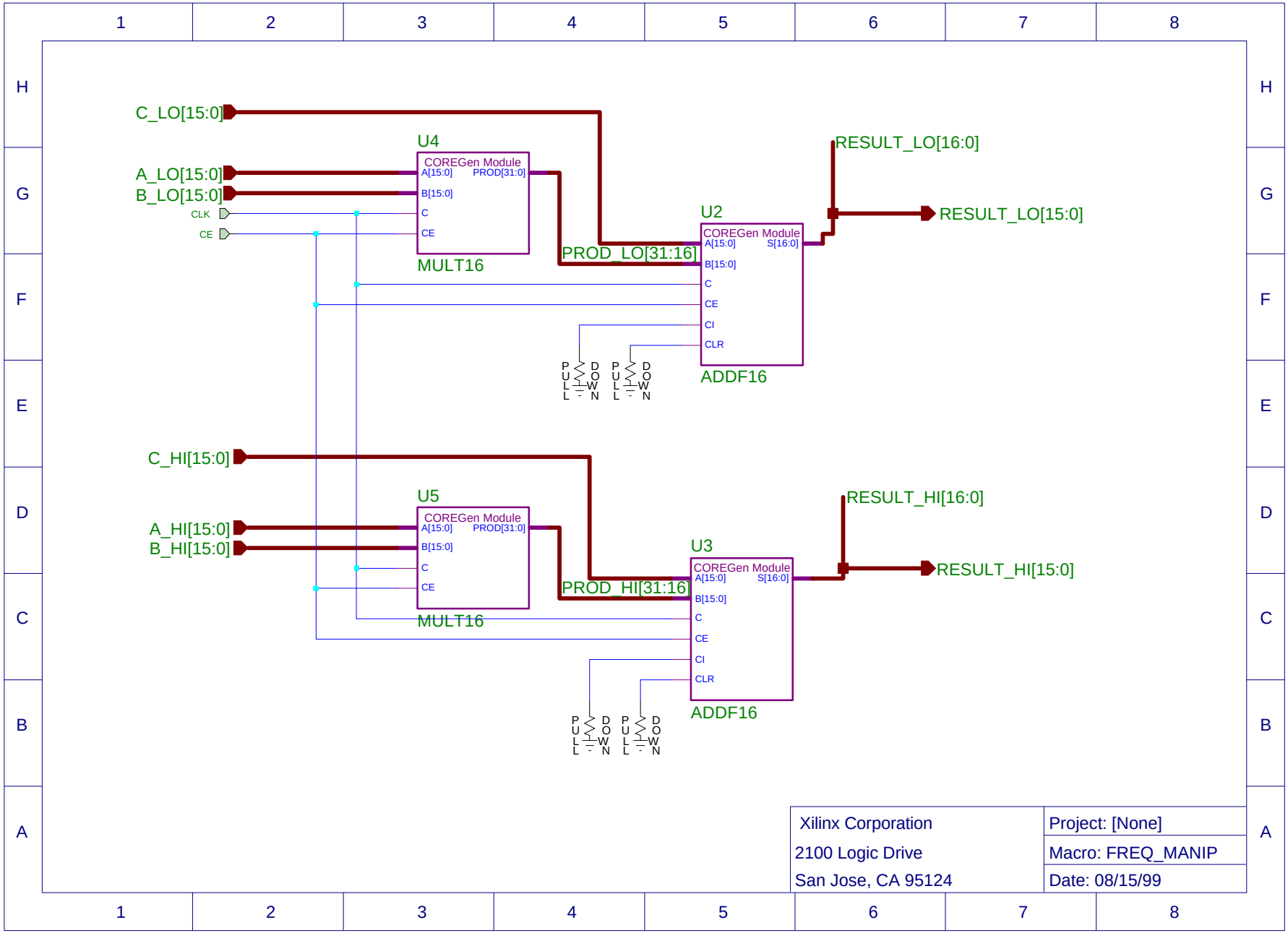


Figure 9.4 Frequency manipulation arithmetic functions

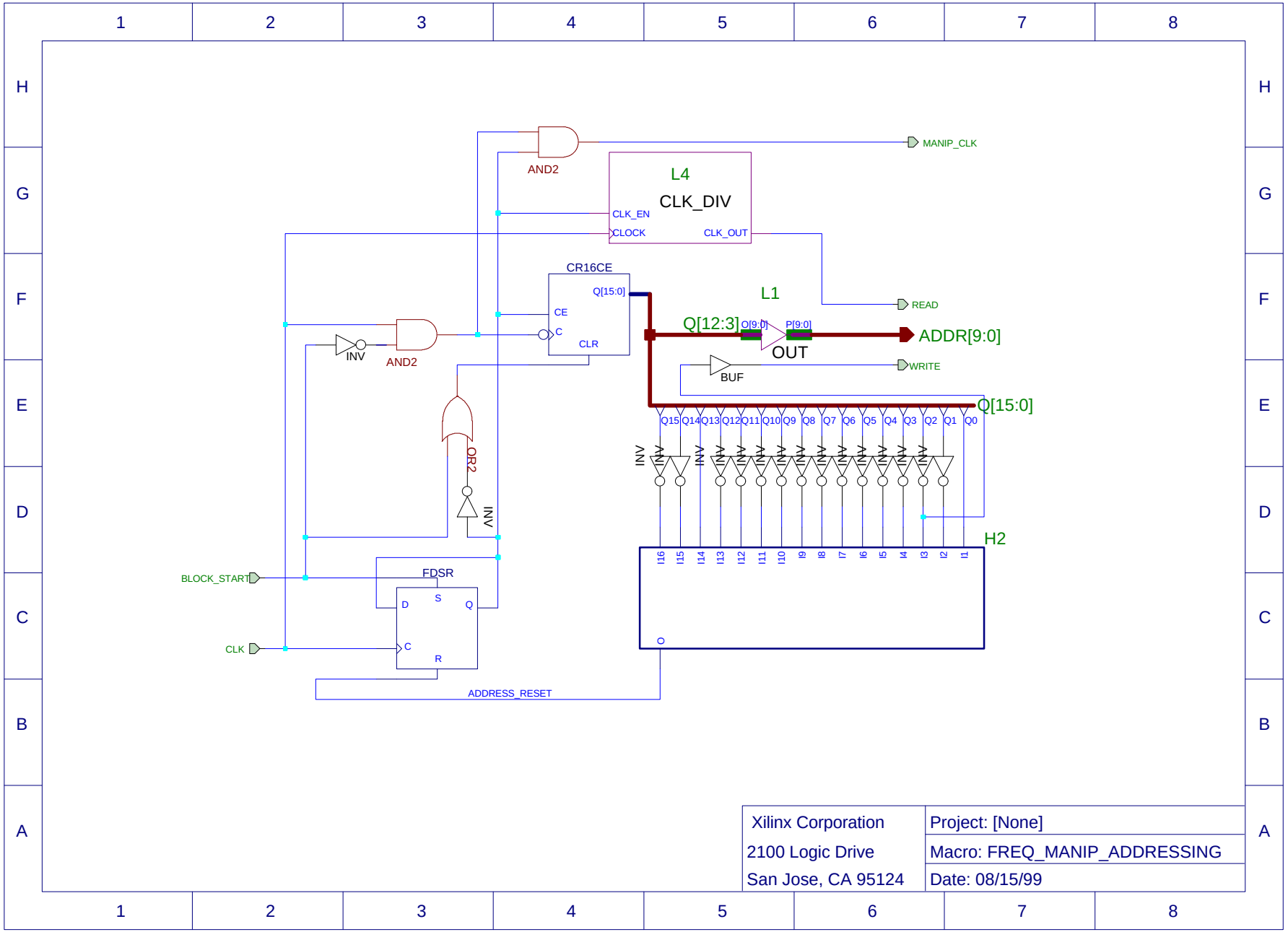


Figure 9.5 Frequency manipulation addressing and timing functions

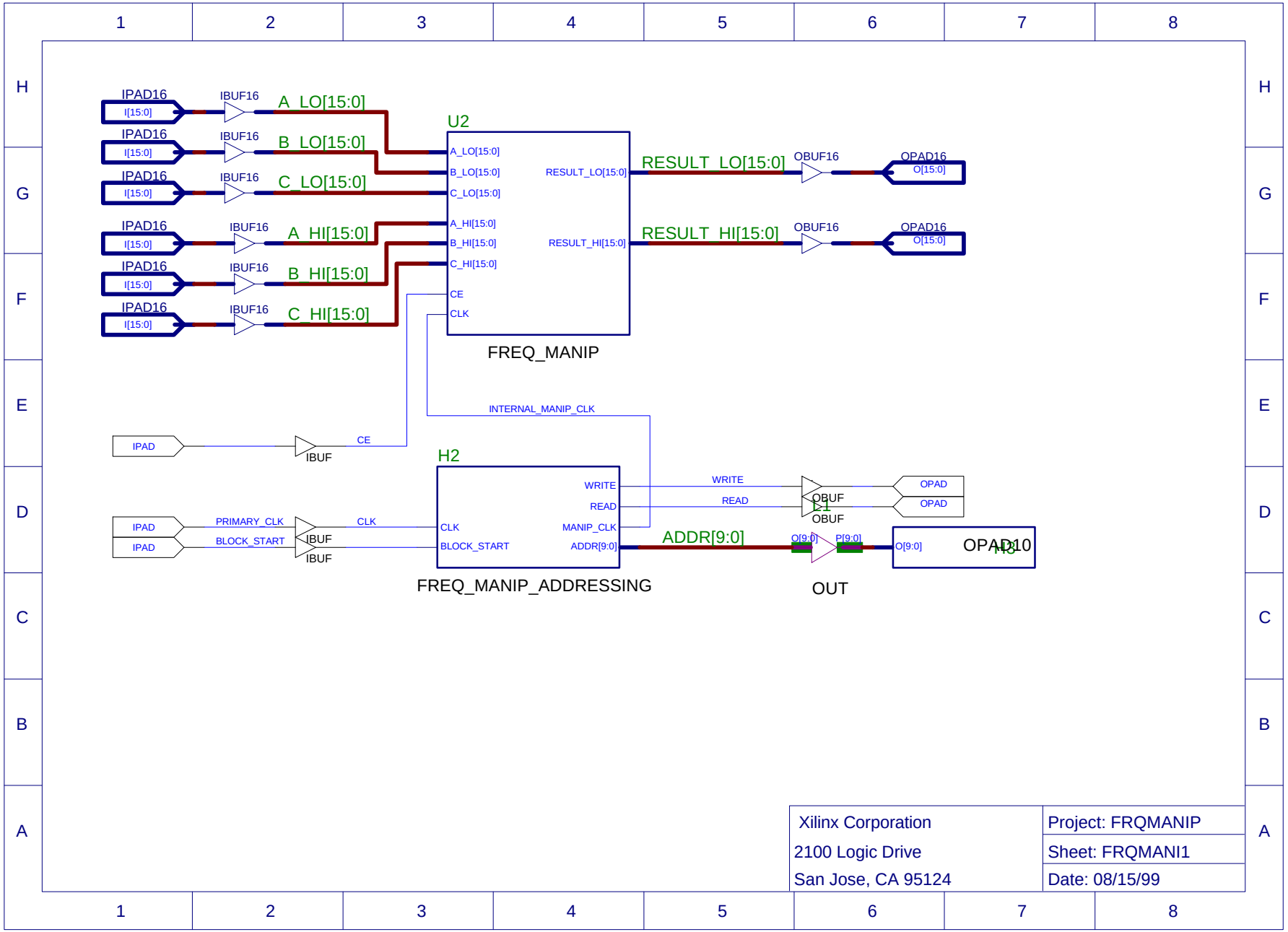


Figure 9.6 Frequency manipulation complete circuit

Figure 9.6 shows the combination of arithmetic and timing functions into a single unit.

9.2.3 Functionality Testing and FPGA Placement

Complete functionality testing has been undertaken on the overall frequency manipulation block. On the CD there is a command test file `frqmanip_verfication.cmd` which can either be re-run, or the result's waveform viewed directly from the file `frqmanip.tve`. The waveform file shows the exact read/write and address timing.

The complete frequency manipulation block can be placed in the smallest XC4013XLA device and uses only 13% of the CLBs. This would allow the time manipulation function to also be placed within the same physical FPGA to reduce costs. In addition, other board logic can easily be placed in the same device as long as one with enough I/O pins is chosen.

9.2.4 Performance

Using the XLA-07 speed grade, post placement timing analysis suggests a maximum clock frequency of 125 MHz, in which case each manipulation of 1024 samples would be completed in approximately 65 µs. Bearing in mind the external clock generation of all components, a manipulation clock at 70.656 MHz (2^2 x 17.664) will give a complete process execution time of just 116 µs, well within the 232 µs the sampling window limit.

9.3 Time Manipulation FPGA

Each time sampling window, the time manipulation operation adds 1024 real numbers downloaded from the PC during the previous sampling window and stored in one of the PC memory interfaces, to the 1024 real time samples produced by the IFFT operation, again in the previous sampling window and stored in the IFFT – time manipulation FPGA memory interface.

All schematic, simulation and other files for the time (and frequency) manipulation FPGA are located in the `:\Designs\Xilinx\allmanip` directory on the enclosed CD.

9.3.1 Peripheral Circuit Requirements

The processed time samples from the IFFT and addition components from the PC are stored in sequential order in their respective memory interfaces. The time manipulation block must provide addressing and read/write strobes exactly the same as for the frequency manipulation block.

9.3.2 Logic Design

9.3.2.1 Arithmetic Functions

The arithmetic functions of the time manipulation block are just a simplified version of those for the frequency manipulation block, incorporating only a single 16-bit adder. Figure 9.7 shows the circuitry for the time manipulation adder.

9.3.2.2 Timing Functions

Because there are spare CLBs in the frequency manipulation FPGA, the time manipulation circuitry can be placed within the same FPGA. Although using a Core adder each addition operation takes only one clock cycle, the timing circuitry of the frequency manipulation block can be used to address the time manipulation block's memory interfaces, thus eliminating the need for additional circuitry. Figure 9.8 shows the combination of frequency and time manipulation blocks.

9.3.3 Functionality Testing

Complete functionality testing has been undertaken on the overall time and frequency manipulation blocks. The command test file `allmanip_verification.cmd` can be re-run, or the results viewed directly from the file `allmanip.tve`.

9.4 Clock and Page Select Circuitry

9.4.1 Clock Signals

The various components of the simulator board require different clock frequencies. Three global clocks, external to the FPGAs and continuously running, are defined as follows:

- PRIMARY_CLK 70.656 MHz
- CLK_2 35.328 MHz (PRIMARY_CLK / 2)
- CLK_3 17.664 MHz (PRIMARY_CLK / 4)

These three clocks will be internally gated and divided as required within each FPGA to provide the necessary clocking for each functional block. Figure 9.9 shows the derivation of the three global clocks from an external oscillator, reproduced from the CD directory `:\Designs\Xilinx\Clocks`.

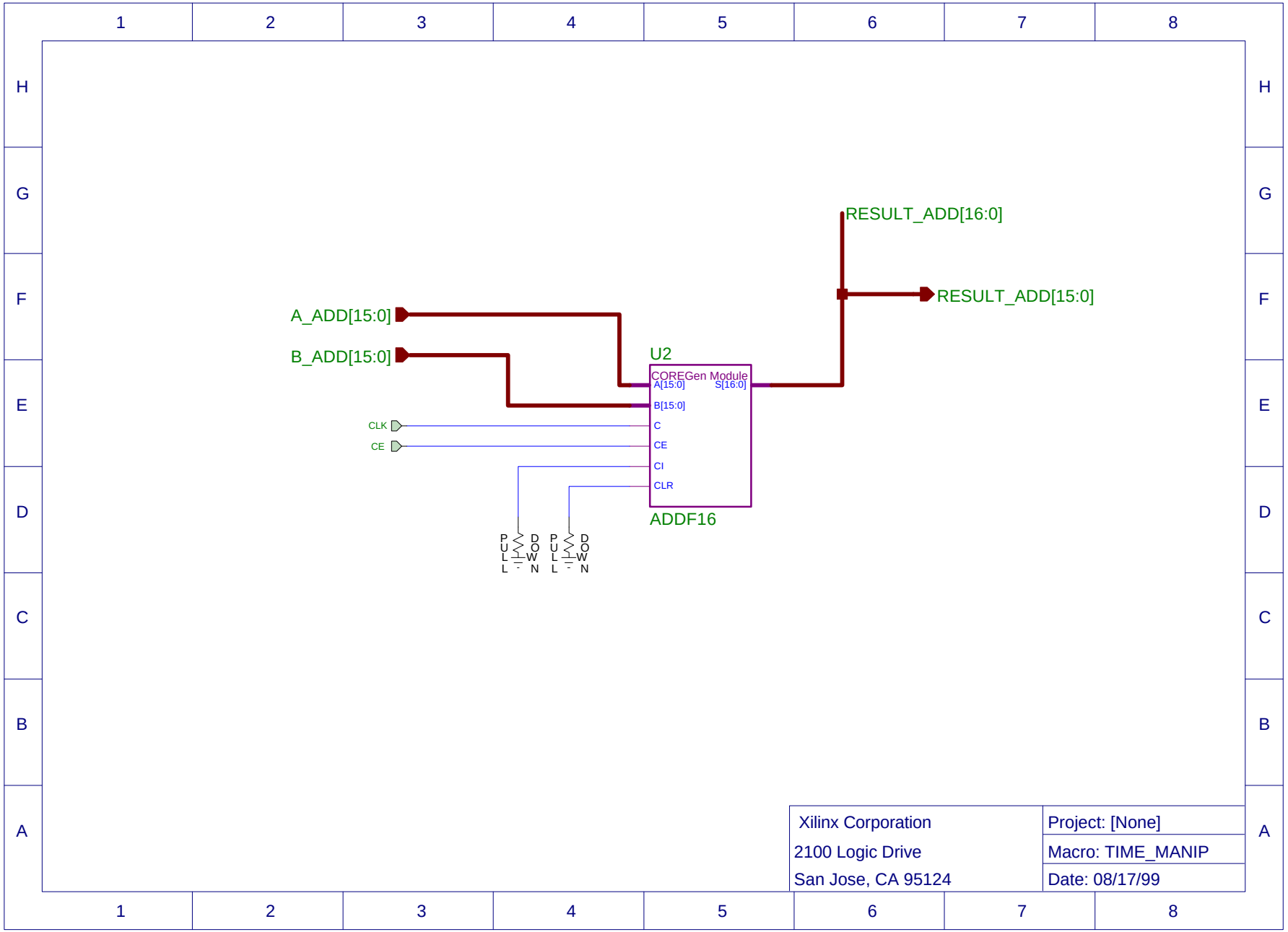


Figure 9.7 Time manipulation arithmetic functions

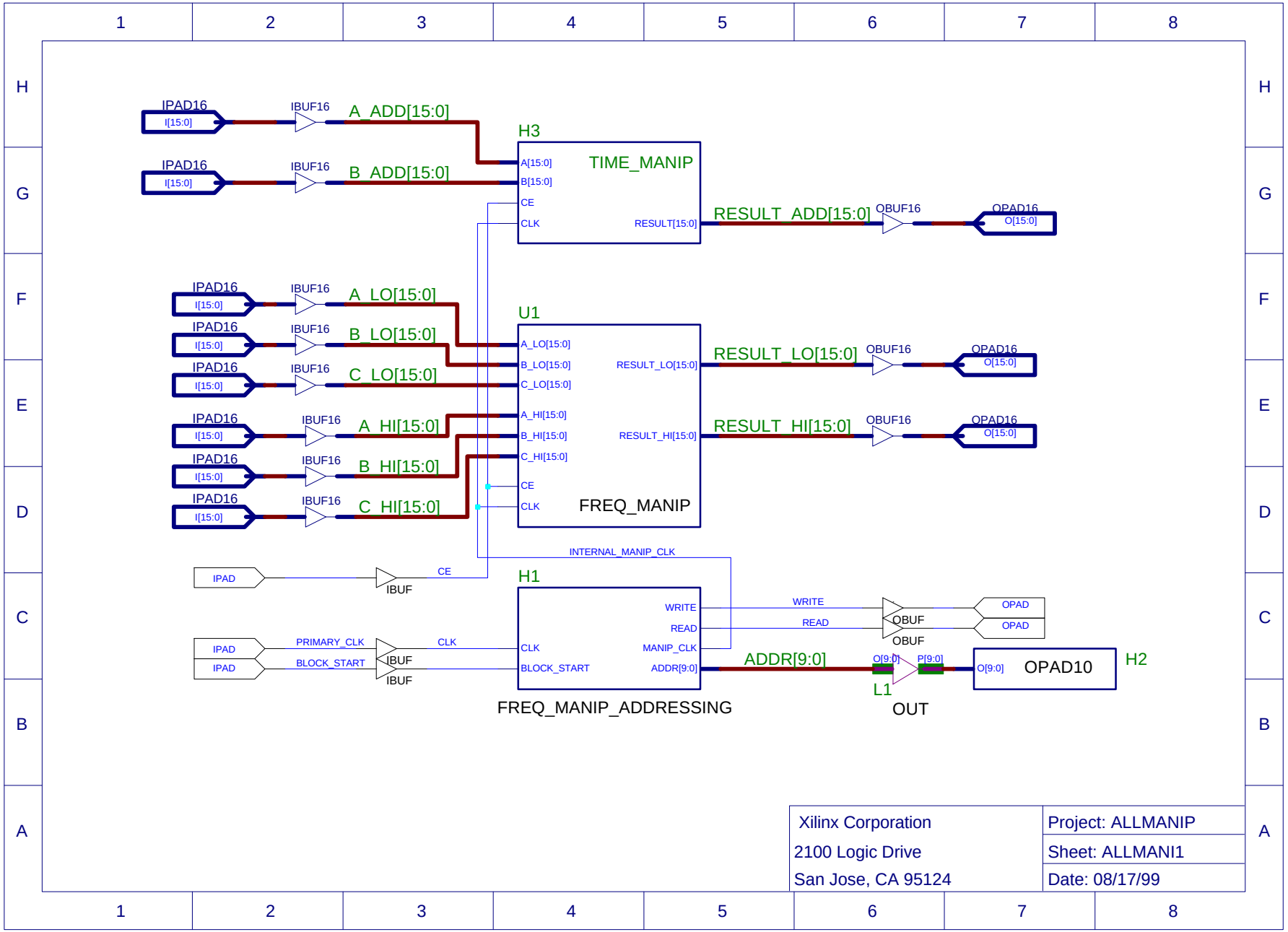


Figure 9.8 Time and frequency manipulation complete circuit

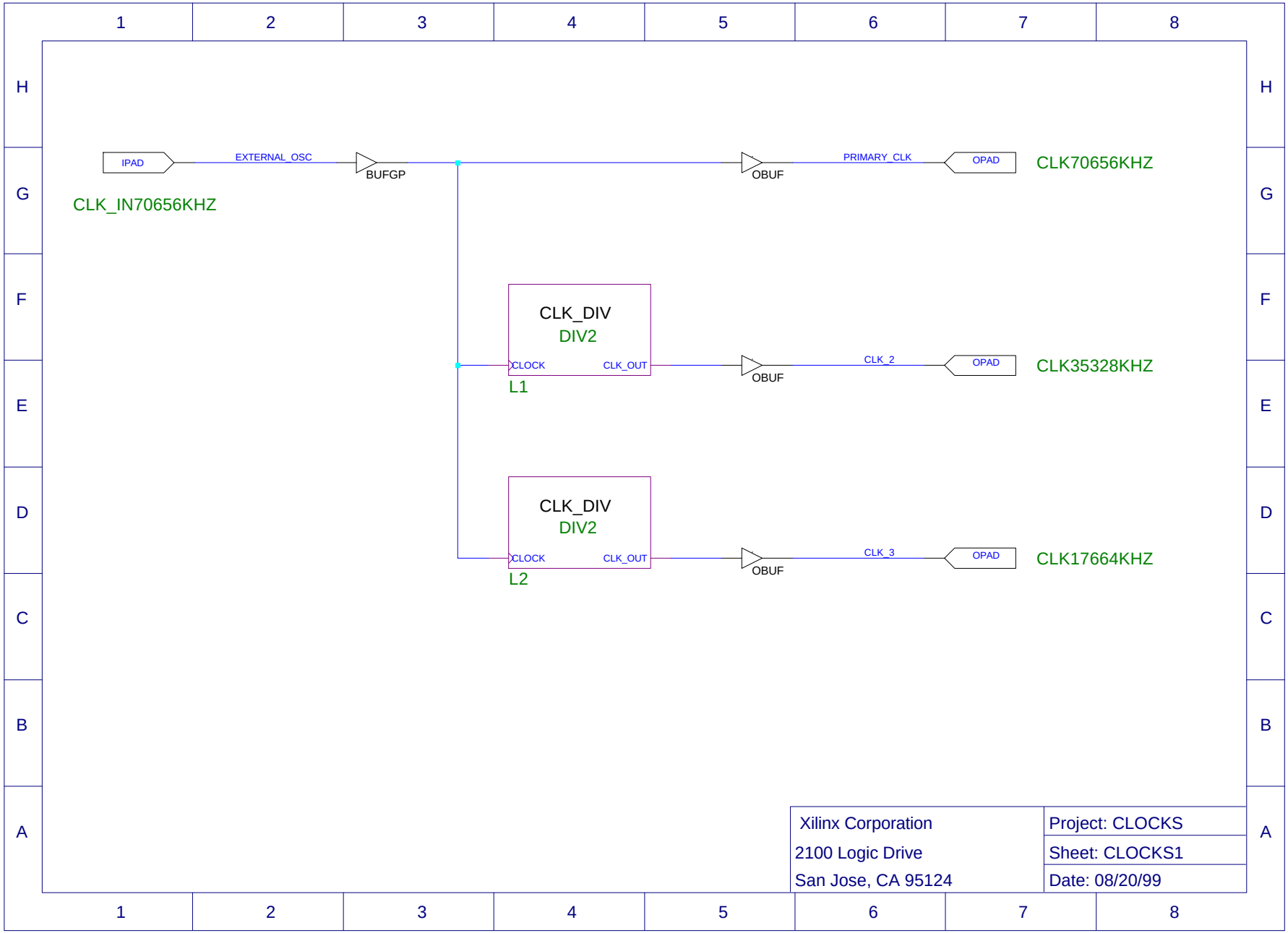


Figure 9.9 External global clock generation

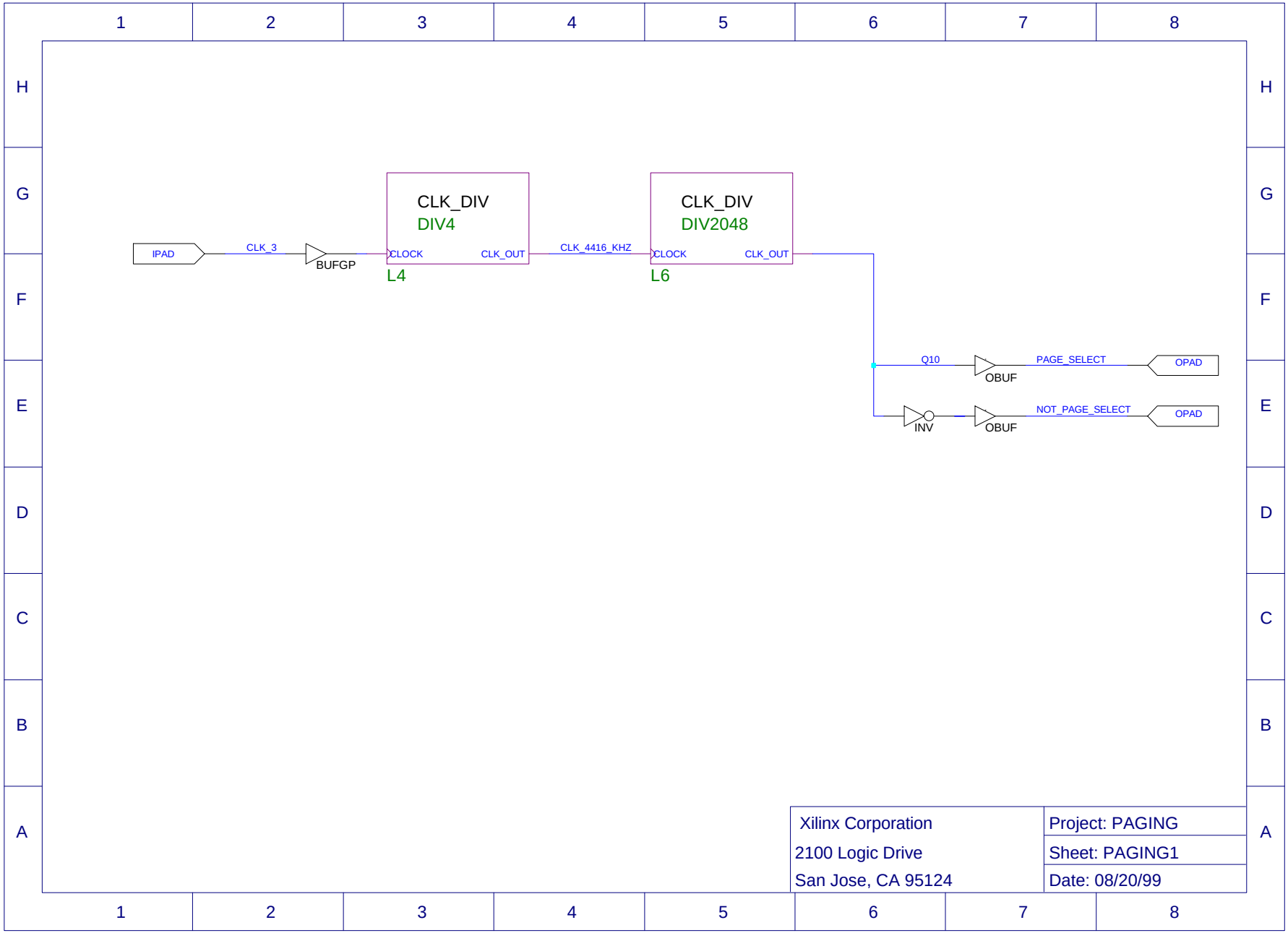


Figure 9.10 Memory interfacce paging circuitry

9.4.2 Memory Interface Paging

As described in chapter 8, memory interface page selection is easily achieved by dividing the 4.416 MHz ADC clock by 2048. With the use of AFEs instead of basic ADCs, the A/D conversion process clock is 17.664 not 4.416 MHz, therefore the page selection circuitry shown in figure 9.10 and stored in the directory `:\Designs\Xilinx\Paging`, initially divides the global CLK_3 by four. The simulation command file `page_select_verification.cmd` generates the waveforms in the file `page_sel.tve`.

The appropriate PAGE_SELECT and NOT_PAGE_SELECT signals must be connected to each side of all memory interfaces, those between simulator functional blocks and PC interfaces, as shown in figures 8.1 and 8.4.

9.5 Memory Chips

Two types of memory components are needed for the simulator board: dual port RAM for memory interfaces and ordinary single port scratchpad RAM for the FFT Reference Design. The main criteria for component selection are of course access time, data word width and storage depth.

9.5.1 Interface Memory

One of the main functions of the memory interfaces is rate conversion between writing and reading from opposite sides of the RAM. Dual port RAM chips can be strobed either by a single clock on which data transfer occurs on both sides, or each side can be driven by separate clocks working asynchronously. The simulator memory must be of the latter type to allow reading and writing at different rates.

One potential problem with dual port RAM is the possibility of peripheral circuitry on opposite sides of the device addressing the same location at the same time. More expensive chips can handle this situation allowing access to the same location either during the same clock cycle through a priority mechanism or by delaying one side’s access by temporarily buffering data until the next clock cycle. More basic chips have no mechanism for access conflicts. One advantage of the page structure is that locations can’t be simultaneously read to and written from at the same time as the peripheral circuitry on opposite sides of the RAM are always working on separate pages of the storage space. This simplifies memory selection for the interfaces reducing device selection solely to access time and capacity considerations.

The simulator’s memory interfaces can be classed into two types, those storing complex data and those for real data only. The former requires 32-bit, whereas the latter requires just 16-bit wide RAM. Regardless of the complex or real nature of the data, each interface requires a total of 2048 locations, equivalent to two pages each of 1024 locations. Recourse to the FFT Reference Design’s timing

diagrams show that read and write accesses occur in single FFT clock cycles. Running on the global clock, CLK_2, the FFT interface memory must have access times better than 29 ns. The manipulation operations running at the PRIMARY_CLK require interface access times better than 14 ns

The largest manufacturer of dual port RAM is Cypress Semiconductor, producing devices of both 16 and 32-bit widths. Table 9.2 identifies two suitable devices for the real data 16-bit and complex data 32-bit interfaces.

Device	Data Width and Depth	Access Time
cy7c09389v[1]	16 bits x 16 k	7.5 – 12 ns
cy7c09579v[2]	32 bits x 8 k	5 – 8 ns

Table 9.2 Suitable memory interface devices

9.5.2 FFT / IFFT Scratchpad RAM

The FFT scratchpad RAM requires 32-bit by 1024 location single port memory, operating at the FFT clocking rate. Any synchronous RAM with access times better than the FFT clocking period (29 ns at 35 MHz) will suffice. Since so many manufacturers produce suitable RAM, device selection is until board construction.

9.6 Low Level Design Summary

The FFT Reference Design with registered address and tristate data buses for scratchpad RAM use, associated input RAM address bus and control circuitry and internal clock gating can all be placed in a single 240 pin QFP XC4062XLA-07HQ240 device. Overall clocking at between 42 and 69 MHz is projected from design simulation within the Xilinx Foundation software environment. For the ADSL simulator, the global 35 MHz CLK_2 clock is used to give a complete 5206 cycle execution time of 149 μs. Transform results appear to be incorrect, with further testing required before a definitive conclusion can be drawn. The IFFT block hasn't been designed because of the doubt concerning the Reference Design's integrity, but will be virtually identical to that for the FFT.

Both time and frequency manipulation blocks with all associated addressing and control signals for I/O memory interface data transfer have been completed. Functional verification confirms the arithmetic integrity of both operations. Both blocks fit into a single XC4013XLA-07HQ240 device. Operation at upto 125 MHz is possible, but reduced to the 70 MHz PRIMARY_CLK rate for the ADSL simulator. A complete set of 1024 frequency manipulation complex additions and multiplications require 9 632 cycles giving an execution time of 137.6 μs. The time manipulation circuit runs from the frequency

manipulation clock to reduce logic overhead and therefore takes the same time to execute a complete set of 1024 real additions.

Paging and global external clock generation circuitry providing page selection and three free-running clock signals at 70.656, 35.328 and 17.664 MHz, can all fit within the XC4013 FPGA used for the manipulation circuits.

References

[1] Cypress Semiconductor, CY7C09569V FLEx36™ Dual-Port Static RAM, Data Sheet, February 13, 1999.

[2] Cypress Semiconductor, CY7C09389V Synchronous Dual-Port Static RAM, Data Sheet, November 23,1998.

Chapter 10

Conclusions

This chapter presents an overview of the initial research into DSL simulator requirements, appropriate signal processing techniques, implementation options and two specific designs. Finally, a brief discussion of the simulator's extension to full and bandwidth limited VDSL systems is presented.

10.1 The Modelling Requirement

A simulator emulating a twisted copper access pair at DSL frequencies must adequately model the line's physical impairments of insertion loss and phase shift, and also the effects of noise and crosstalk from other access lines within the same bundle.

10.1.1 Physical Line Effects

Insertion loss and phase shift are functions of conductor diameter, access reach and frequency. Theoretical development of a line's attenuation and its effect on a signal's phase from the four primary line characteristics of inductance, capacitance, inductance and resistance lead to transfer function expressions for insertion loss and phase response or shift. Theoretical results closely match those experienced on real DSL lines and can be used as a basis for determining the physical line modelling requirements.

Over DMT ADSL bandwidths and reach, total insertion losses of upto 62 dB can be expected. For VDSL systems operating at upto 20 times the frequency, but at greatly reduced reaches, losses as high

as 120 dB occur. For a given length of twisted copper pair, the rate of change of insertion loss with frequency is greatest at lower frequencies. For example on a 6 kft 26 gauge PIC line, the insertion loss changes by 34 dB between DC and 1.25 MHz and by just 14 dB between 1.25 and 2.5 MHz (figure 4.1).

Unlike insertion loss, phase shift changes linearly with frequency (figure 4.3). Over even the shortest reach, the phase undergoes many rotations over both the ADSL and VDSL bandwidths. The full phase rotation bandwidth for short 1 kft lines is approximately 275 kHz and just 23 kHz for longer 12 kft pairs (table 4.3).

Although only a reproduction of material in chapter 3, the complex nature of the twisted copper pair's response over 10 MHz is shown below as its nature is crucial to the simulation signal processing approach taken.

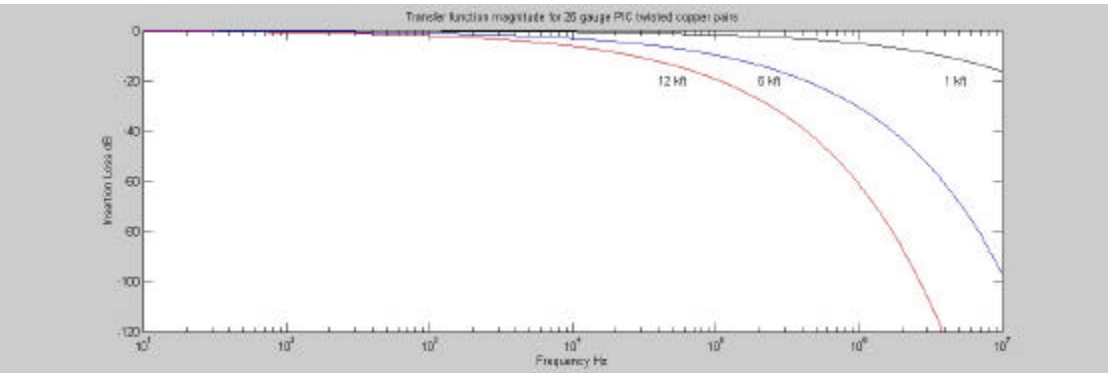


Figure 10.1 Insertion loss for 26 gauge PIC twisted copper pair

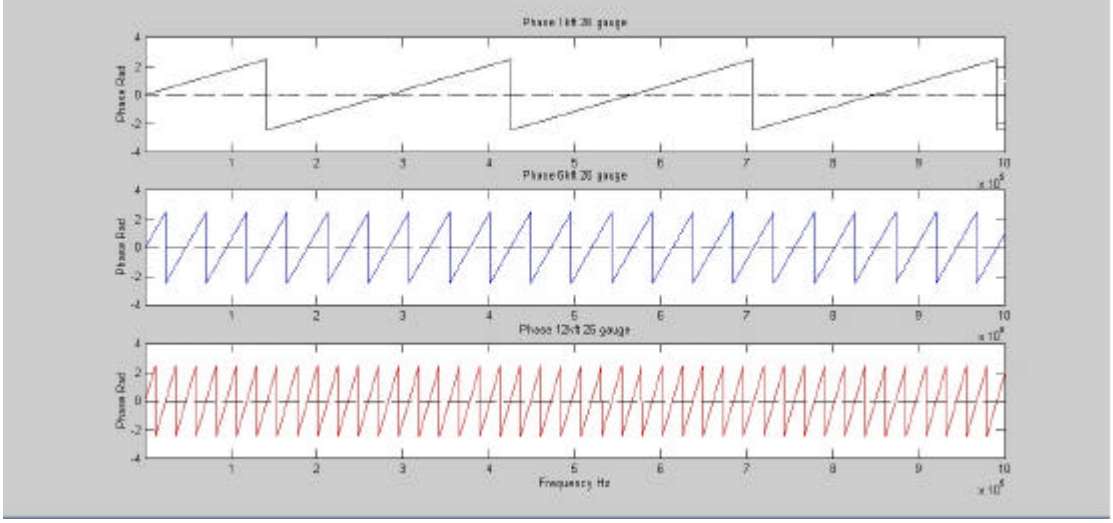


Figure 10.2 Phase shift for 26 gauge PIC twisted copper pair over ADSL bandwidth

10.1.2 Crosstalk and Noise

At extended DSL transmission frequencies, crosstalk from other access lines becomes significant. Crosstalk can be classified into two types: self and foreign. Self crosstalk is an induced signal from and disturber of the same access type whereas foreign crosstalk is due to a disturber of a different access type. For example, self ADSL crosstalk occurs between two ADSL lines within the same bundle whereas foreign crosstalk occurs between ADSL and ISDN lines.

DSL modem receivers incorporate sharp bandpass filtering, so in the absence of non-linear intermodulation distortion, only crosstalk within the DSL transmission bandwidth of the modem under test is significant because any out of band crosstalk is rejected by the receiver.

In addition to crosstalk type, the nature of its induction is also classified as Near End CrossTalk (NEXT) or Far End CrossTalk (FEXT). However, this is irrelevant to the line simulator hardware as NEXT and FEXT crosstalk incident on a receiver can't be distinguished (ignoring the case of 'self' induced self NEXT whereby a receiver could distinguish the difference because of the correlation between the signal it is transmitting and that which it is receiving due to crosstalk). Rather, the software to drive the hardware would take into account the nature of NEXT and FEXT in generating a realistic crosstalk model.

Both AGWN and coloured bandpass noise are incident on DSL lines. For the same reason as crosstalk, only noise occupying the relevant DSL spectrum need be simulated.

10.2 Signal Processing Methods

Signal modification in the frequency domain provides possibly the only method of simulating the physical line's response due to the tight control required over phase, shown in figure 10.2. The method of signal manipulation in the frequency domain, known as frequency domain filtering, has long been practised in the field of medical CT and MRI scanning where processing is non-real time. The principle of multiplying each DFT signal sample with a complex filter coefficient at that frequency is both simple to understand and elegant. The main reason for its limited use in the telecommunication's arena is the need to compute a frequency domain description of the signal using a DFT before any manipulation can take place and then to transform the result back to the time domain with an IDFT. Once time – frequency and the reverse transformations can be computed in real time (i.e. in the same time it takes to sample the N points of the transform, known as the time sampling window), frequency filtering offers a very powerful tool.

In addition to modelling the line's physical response, noise and crosstalk that are both fundamentally additive in nature, can be modelled in the frequency domain by the simple addition of complex components to each DFT sample.

Initially simulation by signal manipulation solely in the frequency domain was developed. However, some noise, especially impulsive noise is more readily described in the time domain through addition to time samples. The revised simulator design includes an additional time domain manipulation block.

Generation of crosstalk in a hybrid time and frequency domain simulator can be achieved by either the addition of suitable DSL line code time samples at the time manipulation stage or through the addition of suitably modified discretised PSD masks in the frequency domain. The latter method can be used to simply simulate self crosstalk through the addition of an attenuated and delayed copy of the actual ADSL signal from the modem under test. Foreign crosstalk can be simulated through the addition of the theoretical or measured PSD of a foreign disturbing signal, a simpler approach than the generation and addition of specific line codes in the time domain.

10.3 DFT Requirements

The requirements for the DFT used within the simulator consist of the maximum transform frequency resolution or cell size, the amplitude precision (i.e. quantisation interval) and the transform computation time for real time simulator operation.

The frequency transform resolution determines the accuracy to which the discrete model of the line and transmission process reflects the actual continuous line. For the physical line effects, an initial figure of a maximum 10% difference between the continuous line's response and the discrete model's constant response over any transform cell was considered.

DSL systems incorporate frequency division multiplexing to allow POTS to co-exist alongside data transmission at higher frequencies. This coupled with the strict bandpass filtering of the modem receivers, allows the line's behaviour at frequencies below the lowest frequency used by the DSL modulation scheme to be ignored. ADSL's FDM with the POTS is fortuitous as concerns the size of the required DFT, as the greatest variation of insertion loss occurs in the 30 kHz band reserved POTS (figure 4.1). VDSL's FDM is with both POTS and ADSL, allowing the spectral band from DC to about 1.2 MHz to be ignored. The FDM nature of DSL transmission gives rise to the concept of the first relevant transform interval for simulation as being the first DFT cell within the DSL PSD, not from DC (section 4.1.1 and table 4.2).

ADSL simulator systems require at least 256 transform cells over the 1.1 MHz bandwidth giving a maximum transform cell size of 4.3125 kHz due to insertion loss (table 4.2) and coincidentally the same considering phase shift (table 4.5, strictly for 6 kft lines only). VDSL systems with a greater bandwidth, but first relevant transform interval from just above the 1.1 MHz ADSL spectrum, require at least 741 transform cells over the 20 MHz transmission bandwidth.

In addition to the physical line effects, the DFT cell size's effect on crosstalk modelling must also be considered. For DMT systems, in order to model crosstalk and noise within a particular transmission bin, the discrete representation of the signal must have at least one sample across that bin's local bandwidth. ANSI DMT ADSL, the only developed or proposed DMT system, has 256 subcarriers

spaced at 4.3125 kHz, therefore the DFT employed must have at least 256 samples across the 1.1 MHz ADSL bandwidth and spaced at the 4.3125 kHz DMT frequencies. The same arguments don't apply to single 'carrier' CAP ADSL and CAP/QAM VDSL systems. In the absence of stricter needs, the insertion loss and phase shift requirements for ADSL were assumed sufficient to model crosstalk and noise for these other two systems. Overall, the most stringent requirement arising from insertion loss, phase shift and crosstalk modelling requirements must be implemented. For DMT ADSL systems, all three maximum transform cell requirements were equal at 4.3125 kHz.

In addition to the minimum Nyquist sampling rate of twice the required maximum observable frequency, sampling guard bands are required, thus increasing the necessary A/D conversion rate. For real time simulation, Fast Fourier Transforms are required. When radix 2 or 4 FFT algorithms are used and sufficient sampling guard bands included, the simulator transform parameters in table 10.1 result.

	Sampling Rate	Transform Size
DMT ADSL	4.416 MSPS	1024 point (radix 2 or 4)
CAP ADSL	4.416 MSPS	1024 point (radix 2 or 4)
VDSL	50 MSPS	2048 point (radix 2) 4098 point (radix 4)

Table 10.1 FFT Transform Parameters

Quantisation precision must be at least as good as that used in the modems attached to the simulator. A minimum of 15 to 16-bits is required.

10.4 Implementation Paths

Of the three methods of implementing the FFT and IFFT processes considered, DSPs and FPGAs were found viable. After extensive research into performance, ADSL simulation using either technology was thought to be possible. VDSL simulation is currently not feasible without extensive parallel processing techniques. Implementation with DSPs is possibly the most versatile interms of the ease of functional evolution without hardware redesign and extension to VDSL simulation. However, the development and production costs and suitability of development within the university environment favour the FPGA implementation route.

FFT modules for both solutions exist in the form of software code for DSP platforms and Reference logic designs for FPGAs, avoiding the need for complex and time consuming customised transform engine design.

10.5 ADSL Line Simulation Using FPGAs

A simulator board based around Xilinx FPGAs using the FFT Reference Design to provide both the FFT and IFFT functions and with separate time and frequency manipulation blocks requires three separate FPGA processing devices. With three FPGAs, sufficient unused logic capacity exists to implement all necessary memory addressing, read / write and control strobing, memory interface paging and local and global clock circuitry. Dual port RAM memory interfaces between each functional block allows both independent operation within the confines of each time sampling window limit and a pipelined block processing approach. Designing individual blocks is greatly simplified by this timing independence, as internal timing considerations are local to the individual operations themselves. In addition, although there is a question mark over the validity of the results produced by the FFT Reference Design, different replacement FFT and IFFT engines can easily be used in the modular design with memory interfacing. The use of ADCs and DACs within ADSL AFEs simplifies the analogue and conversion side of the design.

Functionality testing of the manipulation blocks confirmed arithmetic operation and process execution times within the 232 μs time sampling window. Functionality testing of the FFT Reference design resulted in an invalid result for a real valued odd sampled sinusoidal input vector. Timing analysis confirmed the transformation process is also completed well within the sample window limit.

Each transform block fits into XC4062XLA-07HQ240 FPGAs with all manipulation circuitry into the smaller XC4013XLA-07HQ240 FPGA. Chip costs are £498 for the two XC4062XLA devices and £60 for the XC4013XLA from MicroCall Ltd.

10.6 Further Work

The major question remaining before a prototype ADSL simulator can be built is whether the FFT engine does indeed produce invalid results. The use of ready made Xilinx FFT designs has been problematic to say the least and it is difficult to see how extensive testing can be carried out on the transform operation because of the lack of a complete, ‘clean’ FFT module that includes the necessary scratchpad RAM to simulate within the Foundation environment. However, if the new Virtex targeted FFT materialises and includes behavioural models, this is probably the best path to follow as it is doubtful whether much support for the XC4000 FFT Reference Design can be gained.

If neither the current FFT Reference Design or new Virtex design proves suitable, there is always the possibility of designing a purpose built transform engine.

In addition to finalising the FFT engines, the PC interface needs to be developed and well documented to allow separate development of the associated hardware. The software driver for the board needs to be written, a major task requiring close collaboration with companies developing new modems to determine the required simulation functionality. An accurate line simulation through appropriate

multiplication and addition vectors for the physical line, crosstalk and noise, will require access to a physical access line for experimental test measurements.

10.7 Towards a VDSL Line Simulator

The main hurdle to an economically viable solution to VDSL line simulation is speed of FFT / IFFT processing. Without an approximate trebling of DSP processing speed or ten-fold increase in FPGA speeds, currently the only method of implementing the simulator would be to use FFT processes which take several time sampling windows to compute, and have several FFT engines working on subsequent separate windows of data concurrently. In this way, if the FFT took say four sample windows to complete, a total of four processing engines would be required. This is perhaps not surprising when one considers that although DMT is by far the more versatile modulation scheme compared to QAM / CAP, signal processing has only recently reached the level where DMT functionality can be viably incorporated into consumer electronic ADSL modems.

Towards the end of the project some discussion with Fujitsu brought forward the idea of a limited bandwidth VDSL simulator. Tentative figures for modems operating upto frequencies of the order of 5 to 6 MHz at 1 kft were suggested. Sampling at 13 MSPS (6 MHz observable spectrum with a 0.5 Mhz guard band), a 1024 sample window is 79 µs long. If the FFT Reference design is found to be correct, targeted at the fastest XLA device, FFT processing times of 75 µs are predicted by the Foundation place and route software, clearly such a simulator is feasible now.

Appendices

Appendix 1

Calculation of Magnitude and Phase Characteristics of 26 Gauge PIC Twisted Copper Pairs

From The transfer function of the twisted copper pair

$$H(f) = \frac{V(f,x)}{V_0(f)} = e^{-x\gamma f}$$

The real and imaginary parts of the propagation constants and by separated

$$\begin{aligned} H(f) &= e^{-x(\alpha + j\beta)f} \\ &= e^{-\alpha x} e^{-j\beta x} \\ &= e^{-\alpha_0 x\sqrt{f}} e^{-j\beta_0 x f} \end{aligned}$$

Since the imaginary exponential term has unity magnitude, it only contributes to the phase response of the line, hence:

$$\begin{aligned} |H(f)| &= \left|e^{-\alpha_0 x f}\right| \\ \phi(H(f)) &= \langle \beta_0 x \end{aligned}$$

The constants α_0 and β_0 are related to the primary line characteristics by

$$\begin{aligned} \alpha_0 &= k\sqrt{\frac{C}{L}} \\ \beta_0 &= 2\pi\sqrt{LC} \end{aligned}$$

The constant k is directly proportional to the resistivity of the access pair due to the skin effect above about 100 kHz and can be determined from experimental measurements of resistance made to determine the primary R characteristic. From the first graph in figure 2.1, k can be calculated either from the gradient or evaluation at a point where the measured resistance is proportional to the root of frequency (over 100 kHz). Using the later method, at 100 kHz, R ≈ 30 Ω, therefore since

$$R(f) = k\sqrt{f}$$
$$k = \frac{R(f)}{\sqrt{f}}$$
$$= \frac{30}{\sqrt{100\,000}}$$
$$k = 0.135978$$

For 26 gauge twisted copper pair, the graphs of the primary line characteristics give,

$$C = 15.25\,nF$$
$$L = 0.205\,mH$$

The transfer function magnitude is therefore

$$\left|H(f)\right| = e^{-0.001173\sqrt{f}\,x}$$

In decibels, the insertion loss for a length of x kft is given by

$$Loss(db) = 10\log e^{-0.001173\sqrt{f}\,x}$$

The phase response in radians for the same line is

$$\mathrm{f}\,(H(f)) = (\mathrm{p}/2)\tan^{-1}\left(\frac{\cos(0.0000111 f x)}{\sin(0.0000111 f x)}\right)$$

Appendix 2

Matlab Version 5 '`ginput`' Function

» version

ans =

5.0.0.4069

» help ginput

GINPUT Graphical input from mouse.
[X,Y] = GINPUT(N) gets N points from the current axes and returns
the X- and Y-coordinates in length N vectors X and Y. The cursor
can be positioned using a mouse (or by using the Arrow Keys on some
systems). Data points are entered by pressing a mouse button
or any key on the keyboard except carriage return, which terminates
the input before N points are entered.

[X,Y] = GINPUT gathers an unlimited number of points until the
return key is pressed.

[X,Y,BUTTON] = GINPUT(N) returns a third result, BUTTON, that
contains a vector of integers specifying which mouse button was
used (1,2,3 from left) or ASCII numbers if a key on the keyboard
was used.

Appendix 3

Fujitsu ADSL AFE Product Preview

Product Preview

FML/MS/ADSLAFE/PV/4078

MB86626

December 1998

KeyWave™AFE ADSL Analog Front End

Version 1.4

Features

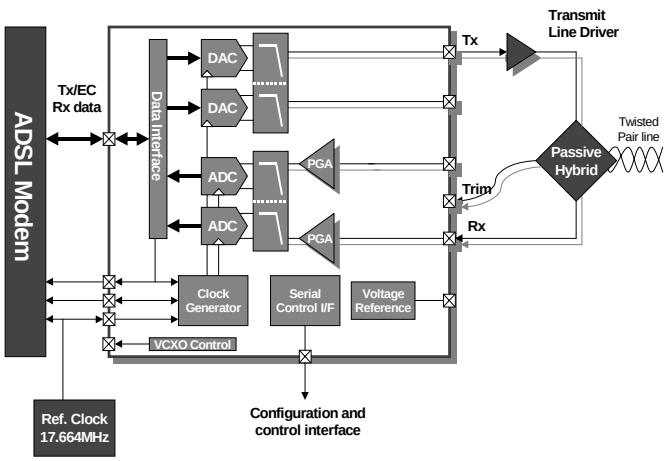
- Integrates all active circuits except Transmit line driver
- Programmable for G.dmt or G.lite (2 channel at CO, or 1 plus Analog modem function at RT)
- Low power, 3.3V operation - from 235mW/ch (2 channel CO G.lite) to 525mW (RT G.dmt)
- Integrated 15-bit A/D & D/A converters
- 0 to +38 dB AGC range
- Supports analog and digital echo cancellation
- 0.35µm CMOS technology with Triple Well
- Industrial temperature range (-40 °C to +85 °C)
- Plastic package 80-pin LQFP
- Compatible with POTS and ISDN

Description

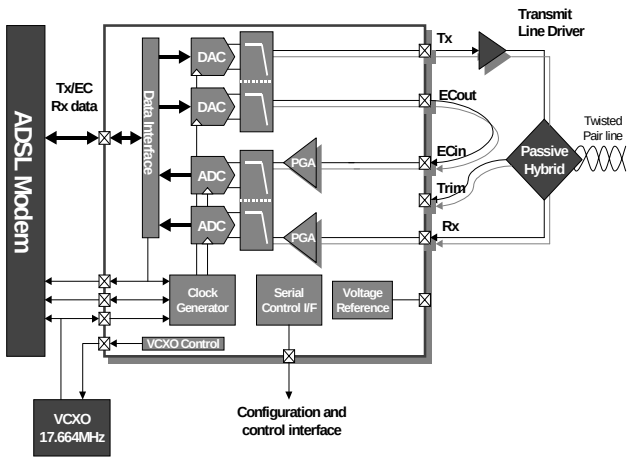
The Fujitsu MB86626 is an analog front end for ADSL modems. The device integrates high resolution analog to digital converters (ADC) and digital to analog converters (DAC), and combined with active filtering significantly reduces the requirements placed on external components. The architecture supports both analog and digital echo cancellation (EC). The MB86626 KeyWave AFE is ideal for cost sensitive Remote Terminal equipment (RT) and power sensitive Central Office equipment (CO). Flexible configuration is incorporated to address future ADSL derivatives (e.g. G.lite, splitterless, Universal ADSL).

Typical Applications

Full Rate ADSL (G.dmt)
Central Office



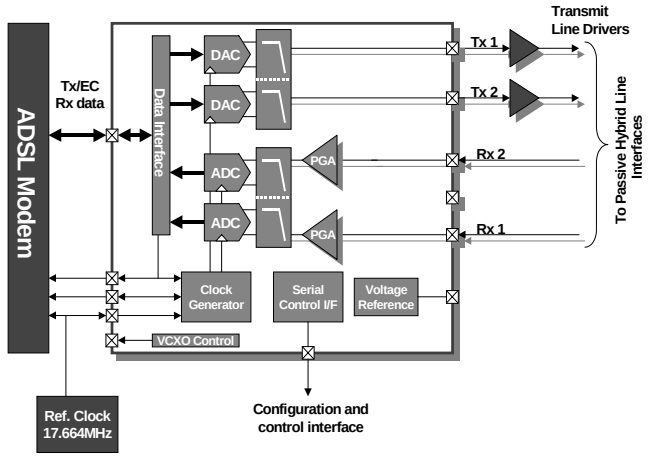
Remote Terminal



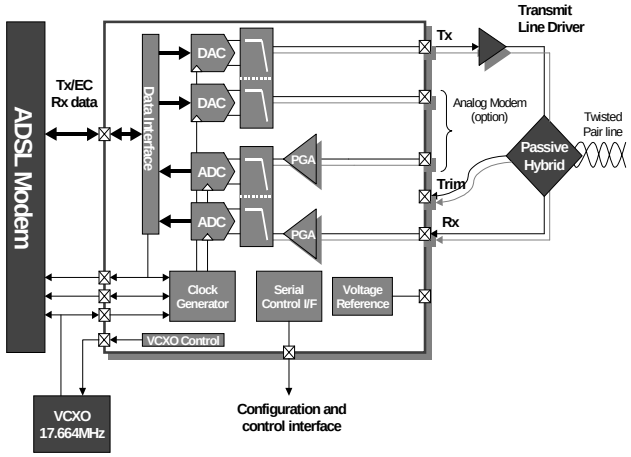
This Product Preview is intended to introduce a new product concept being considered by Fujitsu. The document is subject to change and does not represent a commitment by Fujitsu to develop the product.

MB86626 KeyWave™AFE

ADSL (G.lite)
Central Office (2 lines)



Remote Terminal



Worldwide Headquarters

Japan

Fujitsu Limited

Tel: +81 44 754 3753
Fax: +81 44 754 3329

1015 Kamikodanaka 4-1-1
Nakahara-ku
Kawasaki-shi
Kanagawa-ken 211-88
Japan

http://www.fujitsu.co.jp/

Asia

Fujitsu Microelectronics Asia Pte Limited

Tel: +65 281 0770
Fax: +65 281 0220

151 Lorong Chuan
#05-08 New Tech Park
Singapore 556741

http://www.fmap.com.sg/

USA

Fujitsu Microelectronics Inc

Tel: +1 408 922 9000
Fax: +1 408 922 9179

3545 North First Street
San José CA 95134-1804
USA

Tel: +1 800 866 8608
Fax: +1 408 922 9179

Customer Response Center
Mon-Fri: 7am-5pm (PST)

http://www.fujitsumicro.com/

Europe

Fujitsu Mikroelektronik GmbH

Tel: +49 6103 6900
Fax: +49 6103 690122

Am Siebenstein 6-10
D-63303 Dreieich-Buchschlag
Germany

http://www.fujitsu-ede.com/

Appendix 4

Xilinx Cores

Standard Bus Interfaces

Peripheral Component Interconnect Bus

Function	CORE Solution	Source	Page
PCI Master & Slave Interfaces 2.0	LogiCORE	Xilinx	2-11
PCI Master & Slave Interfaces 1.2.0	LogiCORE	Xilinx	2-19
PCI32 Spartan Master & Slave Interfaces	LogiCORE	Xilinx	2-29
Synthesizable PCI Bridge Design Example	LogiCORE	Xilinx	2-35

PC-Card Bus (PCMCIA)

Function	CORE Solution	Source	Page
PCMCIA Fax/Modem Macro	AllianceCORE	Mobile Media Research	3-15
PCMCIA Library R 1.2	AllianceCORE	Mobile Media Research	3-19
PCMCIA Prototyping Card	AllianceCORE	Mobile Media Research	3-23
PCMCIA Card Debugger/Exerciser	AllianceCORE	Mobile Media Research	3-25
PCMCIA CIS Generator 1.2	AllianceCORE	Mobile Media Research	3-27

Universal Serial Bus

Function	CORE Solution	Source	Page
Low-Speed USB Function Controller	AllianceCORE	Mentor	3-29
Full-Speed USB Function Controller	AllianceCORE	Mentor	3-33
3-Port USB Hub Controller	AllianceCORE	Mentor	3-37
USB Function Evaluation Board	AllianceCORE	Mentor	3-41
USB Hub Evaluation Board	AllianceCORE	Mentor	3-43
USB Simulation Model	AllianceCORE	Mentor	3-45

Other Standard Bus Products

Function	CORE Solution	Source	Page
ISA Plug and Play Interface	Reference Design	Xilinx	5-3
XF-TWSI Two-Wire Serial Interface (IIC)	AllianceCORE	Memec Design Services	3-9

Digital Signal Processing

Correlators

Function	CORE Solution	Source	Page
One Dimensional RAM-Based Correlator	LogiCORE	Xilinx	2-45
One Dimensional ROM-Based Correlator	LogiCORE	Xilinx	2-41

Filters

Function	CORE Solution	Source	Page
Comb Filter	LogiCORE	Xilinx	2-49
16-Tap, 8-Bit FIR Filter	Reference Design	Xilinx	5-3
Serial Distributed Arithmetic FIR Filter	LogiCORE	Xilinx	2-51
Dual-Channel Serial Distributed Arithmetic FIR Filter	LogiCORE	Xilinx	2-55
Parallel Distributed Arithmetic FIR Filter	LogiCORE	Xilinx	2-59

Transforms

Function	CORE Solution	Source	Page
DFT Core (Real Data In, Complex Data Out)	LogiCORE	Xilinx	2-65
FFT Core (1024 Points)	LogiCORE	Xilinx	2-69

DSP Building Blocks

Function	CORE Solution	Source	Page
SDA FIR Control Logic	LogiCORE	Xilinx	2-73
Sine/Cosine	LogiCORE	Xilinx	2-75
Non-Symmetric, 16-Deep Time Skew Buffer	LogiCORE	Xilinx	2-77
Non-Symmetric, 32-Deep Time Skew Buffer	LogiCORE	Xilinx	2-81
Symmetric, 16-Deep Time Skew Buffer	LogiCORE	Xilinx	2-85

Communications and Networking

Asynchronous Transfer Mode

Function	CORE Solution	Source	Page
Cell Assembler (CC-201)	AllianceCORE	CoreEl Microsystems	3-49
Cell Delineation (CC-200)	AllianceCORE	CoreEl Microsystems	3-53
CRC10 Generator and Verifier (CC-130)	AllianceCORE	CoreEl Microsystems	3-57
CRC32 Generator and Verifier (CC-131)	AllianceCORE	CoreEl Microsystems	3-61
UTOPIA Slave (CC-141)	AllianceCORE	CoreEl Microsystems	3-85

Forward Error Correction

Function	CORE Solution	Source	Page
Reed-Solomon Decoder	AllianceCORE	Integrated Silicon Systems	3-71
Reed-Solomon Encoder	AllianceCORE	Integrated Silicon Systems	3-77
Viterbi Decoder	AllianceCORE	CAST	3-91

Telecommunications

Function	CORE Solution	Source	Page
HDLC Protocol Core	AllianceCORE	Integrated Silicon Systems	3-65
MT1F T1 Framer	AllianceCORE	Virtual IP Group	3-81

Base-Level Functions

Basic Elements

Function	CORE Solution	Source	Page
Clock Divider	LogiBLOX	Xilinx	4-4
Comparator	LogiBLOX	Xilinx	4-4
Constant	LogiCORE	Xilinx	2-91
Constant	LogiBLOX	Xilinx	4-4
Counter	LogiBLOX	Xilinx	4-4
Loadable Binary Counter	Reference Design	Xilinx	5-4
Ultra-Fast Synchronous Counter	Reference Design	Xilinx	5-4
Accelerating Loadable Counters	Reference Design	Xilinx	5-4
Data Register	LogiBLOX	Xilinx	4-4
Decoder	LogiBLOX	Xilinx	4-4
Frequency/Phase Comparator for PLL	Reference Design	Xilinx	5-5
Simple Gates	LogiBLOX	Xilinx	4-4
Harmonic Frequency Synthesizer and FSK Modulator	Reference Design	Xilinx	5-5
Input/Output	LogiBLOX	Xilinx	4-4
Multiplexer	LogiBLOX	Xilinx	4-4
Multiplexers, Barrel Shifters	Reference Design	Xilinx	5-3
Two Input MUX	LogiCORE	Xilinx	2-93
Three Input MUX	LogiCORE	Xilinx	2-95
Four Input MUX	LogiCORE	Xilinx	2-97
Parallel to Serial Converter	LogiCORE	Xilinx	2-99
Pulse-Width Modulation	Reference Design	Xilinx	5-5

Basic Elements

Function	CORE Solution	Source	Page
Register	LogiCORE	Xilinx	2-101
Serial Code Conversion between BCD and Binary	Reference Design	Xilinx	5-5
Shift Register	LogiBLOX	Xilinx	4-4
Tristate	LogiBLOX	Xilinx	4-4

Math Functions

Function	CORE Solution	Source	Page
1's and 2's Complement	LogiCORE	Xilinx	2-103
Accumulator	LogiBLOX	Xilinx	4-4
Scaled by 1/2 Accumulator	LogiCORE	Xilinx	2-105
Adder/Subtracter	LogiBLOX	Xilinx	4-4
Adders, Subtractors, Accumulators	Reference Design	Xilinx	5-3
Registered Adder	LogiCORE	Xilinx	2-107
Registered Loadable Adder	LogiCORE	Xilinx	2-109
Registered Scaled Adder	LogiCORE	Xilinx	2-111
Registered Serial Adder	LogiCORE	Xilinx	2-113
Integrator	LogiCORE	Xilinx	2-115
Constant Coefficient Multiplier	LogiCORE	Xilinx	2-117
Constant Coefficient Multiplier (Pipelined)	LogiCORE	Xilinx	2-119
Parallel Multipliers, Area Optimized	LogiCORE	Xilinx	2-121
Parallel Multipliers, Performance Optimized	LogiCORE	Xilinx	2-125
Square Root	LogiCORE	Xilinx	2-127
Registered Subtracter	LogiCORE	Xilinx	2-129
Registered Loadable Subtracter	LogiCORE	Xilinx	2-131

Memories

Function	CORE Solution	Source	Page
Delay Element	LogiCORE	Xilinx	2-133
FIFOs in XC4000 RAM	Reference Design	Xilinx	5-4
Register-Based FIFO	Reference Design	Xilinx	5-4
Synchronous FIFO	LogiCORE	Xilinx	2-135
16-Word Deep Registered Look-Up Table	LogiCORE	Xilinx	2-139
32-Word Deep Registered Look-Up Table	LogiCORE	Xilinx	2-141
Registered Synchronous RAM	LogiCORE	Xilinx	2-143
Registered ROM	LogiCORE	Xilinx	2-145
ROM, RAM, Synch-RAM, Dual Port RAM	LogiBLOX	Xilinx	4-4
4Mb Virtual SPROM	Reference Design	Xilinx	5-5

Processor Products

Function	CORE Solution	Source	Page
Dynamic Microcontroller	Reference Design	Xilinx	5-4
TX400 Series RISC CPU Cores	AllianceCORE	T7L Technology Inc.	3-97
RISC CPU Core Design Base Board	AllianceCORE	T7L Technology Inc.	3-103
Scalable Development Platform Integrated Software	AllianceCORE	T7L Technology Inc.	3-107
V8 uRISC 8-bit RISC Microprocessor	AllianceCORE	VAutomation	3-109
IntelliCore™ Prototyping System	AllianceCORE	VAutomation	3-115

Processor Peripherals

Function	CORE Solution	Source	Page
C2910A Microprogram Controller	AllianceCORE	CAST	3-129
Configuring FPGAs over a Processor Bus	Reference Design	Xilinx	5-5
M8237 DMA Controller	AllianceCORE	Virtual IP Group	3-133
M8254 Programmable Timer	AllianceCORE	Virtual IP Group	3-143
M8255 Programmable Peripheral Interface	AllianceCORE	Virtual IP Group	3-147
XF8255 Programmable Peripheral Interface	AllianceCORE	Memec Design Services	3-151
XF8256 Multifunction Microprocessor Support Controller	AllianceCORE	Memec Design Services	3-155
M8259 Programmable Interrupt Controller	AllianceCORE	Virtual IP Group	3-159
XF8279 Programmable Keyboard Display Interface	AllianceCORE	Memec Design Services	3-163
XF9128 Video Terminal Logic Controller	AllianceCORE	Memec Design Services	3-167
DRAM Controller	AllianceCORE	NMI Electronics	3-173

UARTs

Function	CORE Solution	Source	Page
XF-8250 Asynchronous Communications Element	AllianceCORE	Memec Design Services	3-137
M16450 - Universal Asynchronous Receiver/ Transmitter	AllianceCORE	Virtual IP Group	3-121
M16550A - UART With RAM	AllianceCORE	Virtual IP Group	3-125

Generic Development Tools

Function	CORE Solution	Source	Page
GVA-100 DSP Prototyping Platform	AllianceCORE	GV & Associates	3-179
MDS FPGA Development Module	AllianceCORE	Memec Design Services	3-183
Microprocessor-Based Core Evaluation Card	AllianceCORE	NMI Electronics	3-185
Xilinx CORE Generator	LogiCORE	Xilinx	2-7

Appendix 5

Xilinx Core Data Sheets

1. 16 bit, 1024 Point FFT
2. Area Optimised Parallel Multiplier
3. Registered Adder

LogiCORE™ FFT Core (1024 Points)

February 8, 1998 | Product Specification – PRELIMINARY

XILINX®

Xilinx Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
Fax: +1 408-559-7114
E-mail: dsp@xilinx.com
URL: www.xilinx.com

FFT technology developed by Rice Electronics.

Features

- 2's complement, fixed-point arithmetic
- Real-valued input data (15 bit)
- Complex output data (16 bit ~86 dB available output SNR)
- Transform size (N) = 1024
- No programming required
- No "twiddle factor" memory required (internal to Core)
- Process real-time sampling rates ~2Mhz
- Simplified interface (nominal support logic required)
- Synchronous design, optimized for XC4000E, EX, and XL families of FPGAs
- Drop-in modules for the XC4000E, EX, and XL families
- High performance and density guaranteed through Relational Placed Macro (RPM) mapping and placement technology
- Available in Xilinx CORE Generator

Applications

- Communications (high speed modems, transmulti-plexers)
- Instrumentation (medical, scientific, test)
- Multi-media (signal compression/decompression)
- Military (radar, EW, ELINT, ESM)

General Description

The 1024-point Fast Fourier Transform (FFT) Core is a functionally complete processor. The design requires a 1024-point external data memory and nominal interface support. The Core targets the Xilinx XC4000E, EX, and XL FPGA product series.

The Core renders the following transform for a real-valued input vector, f(n):

$$F(j) = \sum_{n=1}^{N-1} f(n) e^{\frac{-2\pi i j n}{N}}$$

for j=0 to (N/2-1)

where:

F(j) = output (frequency domain) coefficients
f(n) = input (time domain) sequence
N = length of input sequence (transform size)

The Core accepts a real-valued input sequence f(n) and produces a complex output sequence F(j). Due to the real-valued nature of f(n), only the lower half of the set of F(j) is generated. The upper half of the F(j) is the complex conjugate of the lower half, and therefore represents redundant information.

General specifications of the FFT Core are listed in Table 1.

Functional Description

The 1024-point FFT Core is a functionally complete processor requiring minimal external control. Only an External Memory (1k word) is required for Core operation. The External Memory holds the input vector f(n) to be transformed.

The Core itself requires no initialization, and may be activated whenever valid data is present in External Memory. When in operation, the Core must have exclusive access to External Memory. The Core performs "read-only" accesses to the Memory (no write operations).

Table 1: 1024-Point FFT Parameters

Core Name	N = Size of Transform	P = Clock Periods[1]	Clock Speed[2]	Execution Time[3]	Core Size[4]
1024 FFT	1024 points	17408			532

1. P = number of clock periods for transform execution
2. Maximum clock speed based on XC4000E-3 series
3. Execution Time = P/(Clock Speed in Mhz)
4. Approximately 70% utilization of F/G function generators for XC4013 device

The 1024-point FFT Core requires no external storage of constants. "Twiddle-factors" are generated internally to the Core and require no user programming.

The Core possesses physically separate input and output interfaces.

The input interface has separate data and address buses for accessing External Memory. While External Memory consists of 1024 words, only 9 address bits are required from the Core. This is due to the simultaneous access of two memory words on every read cycle, as explained below.

The output interface presents output coefficients F(j) at a constant rate. This interface provides physically separate buses for output coefficients and index information. The index identifies the specific output coefficient F(j).

Pinout

DATA INPUT

The input interface includes dual 16-bit unidirectional data buses to the Core (INHI[15:0], INLO[15:0]) and a 9-bit address bus (ADR[8:0]) from the Core. These buses support read-only operations from External Memory during FFT processing.

DATA OUTPUT

Output consists of the 16-bit unidirectional Output bus (OUT[15:0]), the Index bus (INDEX[9:0]), and an output Synchronization signal (SYNC). Output coefficients F(j) are presented on the Output bus. The corresponding value of j appears on the Index bus.

The Index bus identifies the component (imaginary or real) and the coefficient number (j), associated with the data on the Output bus.

TIMING INPUTS

Timing inputs consist of a START signal and a continuous clock (FFTCK). A pre-defined number of clock pulses is required for execution of the transform (see Table 1).

Figure 1 illustrates the 1024-point FFT Core interface signals. The format of the interface signals is summarized in Table 2.

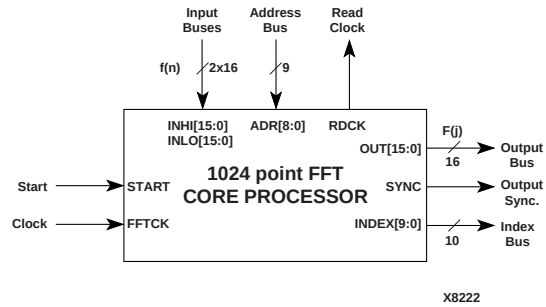


Figure 1: 1024-point FFT Core Interface

Table 2: Core Signal Pinout

Signal	Signal Direction	Description
START	Input	Logic level dictates operational state: 0=Reset State (Dormant), 1=Execution State
FFTCK	Input	Continuous clock (see Table 1 for max.frequency)
INHI[14:0] INLO[14:0]	Input	One sign bit + 14 magnitude bits; 2's complement notation. Dual unidirectional input buses for f(n)
ADR[8:0]	Input	Unsigned 9-bit bus. Specifies read address to External Memory
RDCK	Output	Continuous clock from Core. Synchronizes access to External Memory. Derived from FFTCK (+2)
OUT[15:0]	Output	One sign bit + 15 magnitude bits; 2's complement notation. Unidirectional output bus for F(j)
SYNC	Output	Positive pulse indicates presence of new output component (pulse width = one clock period)
INDEX[9:0]	Output	Unsigned 10-bit bus. Identifies F(j) value on OUT bus. Index MSB indicates imaginary or real component: 0=Imaginary, 1=Real. Remainder of Index indicates j

Timing and Control

When the START signal is low, the Core is "reset". This prepares the Core for execution of a new transform. START must go low for a minimum of one FFTCK period for reset to occur.

While START is low, the Core interfaces are inactive. FFT processing begins when START goes high.

Input Interface

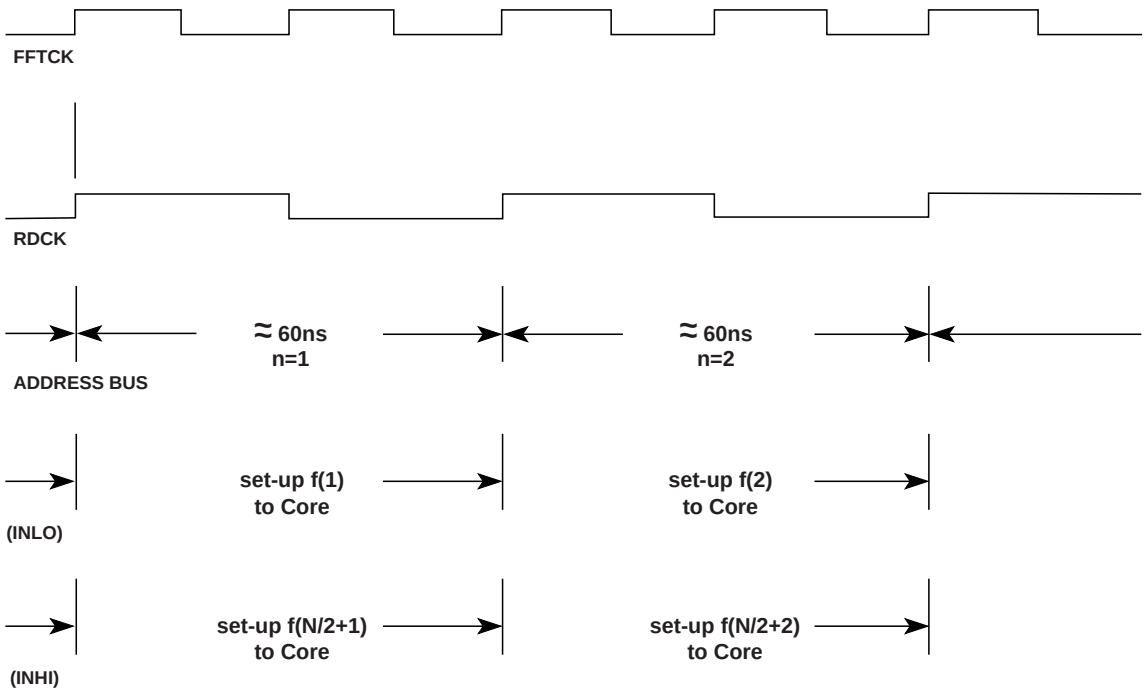
Input Interface Timing

The input interface requires exclusive (uninterrupted) access to the External Memory during FFT processing. This interface consists of an address bus, dual input data buses, and a continuous clock (RDCK).

RDCK is produced by the Core and can be used to synchronize External Memory to the Core. RDCK is derived from the Core input clock (FFTCK) and is half the FFTCK frequency.

The address bus from the Core changes on the rising edge of RDCK. The data buses to the Core must be stable by the next rising edge of RDCK. As seen in Figure 2, the time allocated for memory access is one RDCK (2 FFTCKs). This is equivalent to ~60ns at maximum clock speeds.

At the Core interface, the address and data buses terminate (respectively) at the output and input of FD type registers (XC4000 library primitives). Consequently, the Core contributes minimal logic delay in the memory access path. Accordingly, most of the RDCK period is available for delay through External Memory and associated I/O buffers.



Address bus represents index on input sequence f(n). INLO and INHI (input data buses) must respond with associated data samples within 1 RDCK period.

Continuous timing sequence is maintained for duration of FFT process.

Note: All signal transitions occur on rising clock edge.

X8224

Figure 2: Input Interface Timing

External Memory Organization

The input buffer, f(n), must be accessible in two separate halves from External Memory. The two halves must be available simultaneously on the INLO and INHI data buses. This imposes the following organizational requirements on External Memory:

The lower half of f(n) must be available at the INLO bus from the following External Memory address locations:

f(0)--->location 0
f(1)--->location 1
f(2)--->location 2
•
•
•
f(N/2 - 1)--->location (N/2 - 1)

The upper half of f(n) must be available at the INHI bus from the following External Memory address locations:

f(N/2 + 0)--->location 0
f(N/2 + 1)--->location 1
f(N/2 + 2)--->location 2
•
•
•
f(N -1)--->location (N/2 - 1)

Note: the common 9-bit ADR bus is used to simultaneously address both halves of External Memory.

Ordering Information

This macro comes free with the Xilinx CORE Generator. For additional information contact your local Xilinx sales representative, or e-mail requests to dsp@xilinx.com.

For information on Rice Electronics, contact:

Rice Electronics
PO Box 741
Florissant, MO 63032
Phone: +1 314-838-2942
Fax: +1 314-838-2942
E-mail: ricedsp@aol.com

XILINX®

Xilinx Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
Fax: +1 408-559-7114
E-mail: dsp@xilinx.com
URL: www.xilinx.com

Features

- Parallel Multipliers with parameterizable data widths
- Area optimized design
- Speed almost equal to our performance optimized version, but with fewer CLBs
- Independently adjustable input variable widths from 6 to 32 bits
- Full precision outputs
- Two's complement and magnitude data formats
- Registered inputs and outputs
- Predictable timing and reduced routing times with pre-designed drop-in modules
- Supports Foundation and Viewlogic as well as VHDL and Verilog design methodologies
- Drop-in modules for the XC4000E, EX, and XL families
- High performance and density guaranteed through Relational Placed Macro (RPM) mapping and placement technology
- Available in Xilinx CORE Generator

Functional Description

This parameterized module is a high-speed parallel implementation that multiplies an N-bit wide variable times an M-bit variable and produces an N+M bit result.

The CORE Generator accepts the parameters entered through the dialog box and creates the specific design from the values entered using a parameterized VHDL recipe. VHDL instantiation code and a schematic symbol are created along with the netlist for the design.

An area-efficient, high-speed algorithm is used to give an efficient, tightly packed design. Each stage is pipelined for maximum performance.

In addition to this area-efficient design, the CORE Generator contains a performance optimized design that yields a 10% to 20% increase in speed, but uses more CLB resources.

Pinout

Signal names for the schematic symbol are shown in Figure 1 and described in Table 1.

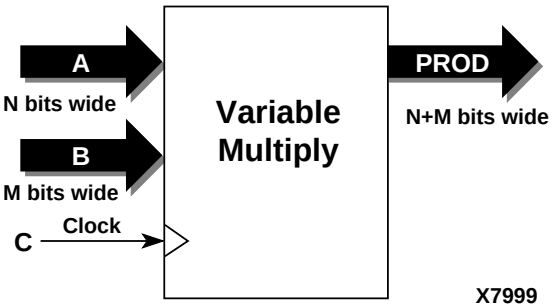


Figure 1: Core Schematic Symbol

Table 1: Core Signal Pinout

Signal	Signal Direction	Description
A	Input	Parallel Data In, N-bits wide
B	Input	Parallel Data In, M-bits wide
C	Input	Clock, processes data on the low to high transition
PROD	Output	Parallel Data Out, N + M bits wide

CORE Generator Parameters

The CORE Generator dialog box for this macro is shown in Figure 2. The parameters are as follows:

- **Component Name**: Enter a name for the component.
- **Width A**: Select an input bit width from the pull-down menu for the input A. The valid range is 6 to 32.
- **Width B:** Select an input bit width from the pull-down menu for the input B. The valid range is 6 to 32.
- **Signed or Unsigned:** Representation of the variables.

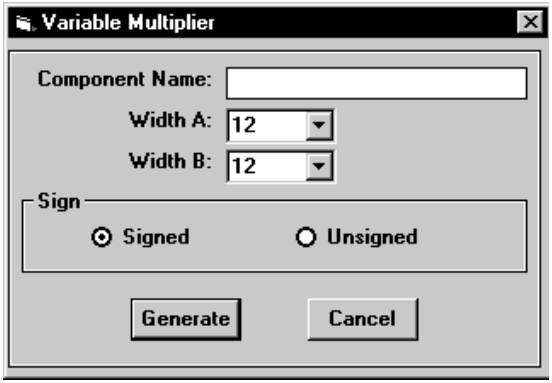


Figure 2: CORE Generator Dialog Box

Module Layout

Figure 3 shows the organization of CLBs in a 12x12 multiplier. This module fits in an array of 132 CLBs organized as a matrix of 12 rows by 11 columns. The module uses 122 CLBs. The white squares represent the unused CLBs, demonstrating the efficiency with which the matrix is utilized. The unused CLBs are available for use in other parts of the system design.

Data enters the module from the left side (A) and the bottom left side (B). The resulting product is available on the right side (O). If all 24 outputs are not connected to the next stage of the design, the Xilinx implementation software eliminates all unused CLBs.

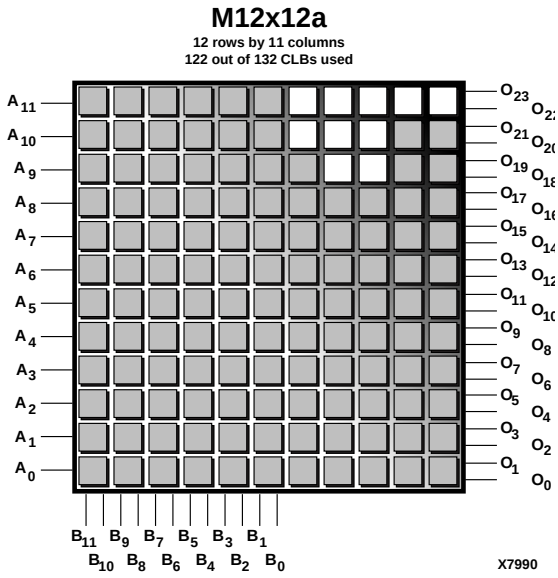


Figure 3: CLB Organization for Module

All of the area efficient series of modules can be built in a rectangular format. For example, an 8-bit by 12-bit multiplier will fit in a rectangular matrix of 8 rows by 11 columns.

Figure 4 shows a 12x12 multiplier in place in an XC4005E chip. The 4005E contains 196 CLBs, of which 122 are used by the 12x12 multiplier and 74 are available for other functions.

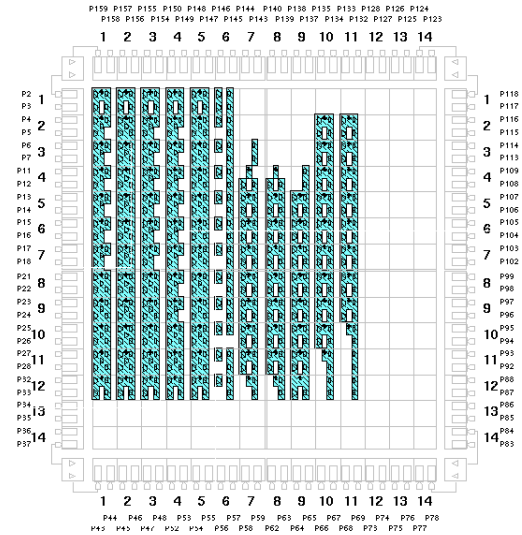


Figure 4: Xilinx Floorplanner View of XC4005E with a 12x12 multiplier

The module contains both relative placement information and timing constraints, allowing the Xilinx place and route software to consistently produce the same timing specifications. CLB relative placement information (R-LOCs) speeds up the routing place and route time by a factor of 10, thus permitting rapid design changes.

Multiplier Latency

Data is buffered on the input and output of the multiplier cores. The total latency (number of clocks required to get the first output) is a function of the width of the **B** variable only.

Table 2: Multiplier Latency

B Data Width	Latency (# Clocks)
6 to 8 bits	4
9 to 16 bits	5
17 to 31 bits	6

Core Resource Utilization

Tables 3 to 8 show the number of CLBs required for some of the available bit widths. The maximum speed is for XC4000E-1 devices.

All the variable multipliers are built to fit in a rectangular or square matrix of N rows by M (or M-1) columns.

Table 3: Bit Width versus CLB Count for A = 8 bits

A Bit Width	B Bit Width	CLB Count	4000E-1 MHz	4000E-3 MHz
8	6	44	86	54
8	8	54	83	53
8	10	86	86	53
8	12	96	78	46
8	14	111	77	45
8	16	124	74	44
8	20	180	71	43
8	24	211	68	41
8	32	282	60	35

Table 4: Bit Width versus CLB Count for A = 10 bits

A Bit Width	B Bit Width	CLB Count	4000E-1 MHz	4000E-3 MHz
10	6	53	81	51
10	8	65	77	49
10	10	100	78	47
10	12	113	77	47
10	14	133	70	42
10	16	146	72	44
10	20	209	65	39
10	24	238	62	38
10	32	330	57	34

Table 5: Bit Width versus CLB Count for A = 12 bits

A Bit Width	B Bit Width	CLB Count	4000E-1 MHz	4000E-3 MHz
12	6	61	75	47
12	8	75	72	46
12	10	116	70	42
12	12	130	67	41
12	14	153	65	40
12	16	167	65	39
12	20	238	63	38
12	24	274	59	36
12	32	371	51	31

Table 6: Bit Width versus CLB Count for A = 16 bits

A Bit Width	B Bit Width	CLB Count	4000E-1 MHz	4000E-3 MHz
16	6	79	64	41
16	8	98	63	39
16	10	148	61	37
16	12	165	59	36
16	14	192	56	35
16	16	213	58	35
16	20	299	56	34
16	24	349	51	33
16	32	473	49	31

Table 7: Bit Width versus CLB Count for A = 20 and 24 bits

A Bit Width	B Bit Width	CLB Count	4000E-1 MHz	4000E-3 MHz
20	6	98	56	36
20	8	120	55	35
20	12	202	55	34
20	16	259	50	31
20	20	358	50	30
24	6	116	50	32
24	8	142	50	32
24	12	238	46	28
24	16	305	45	28
24	20	425	45	28

Table 8: Bit Width versus CLB Count for Other Widths of A

A Bit Width	B Bit Width	CLB Count	4000E-1 MHz	4000E-3 MHz
6	6	37	96	60
9	9	94	81	49
11	11	121	74	45
13	13	161	61	38
14	14	173	62	37
15	15	203	58	36
17	17	293	55	33
18	18	308	53	32
19	19	343	52	32
32	6	151	41	27
32	8	187	41	27
32	12	309	41	26

Multiplier Trade-offs

Two different implementations of parallel multipliers trade area for speed. The area efficient designs consume about one-fourth less CLB resources than the high speed designs in the 4000E family.

The additional routing resources in the 4000EX and 4000XL families will increase the performance for the area efficient designs. In addition, both structures will benefit from the overall performance increase derived from the 4000XL .35 micron process technology.

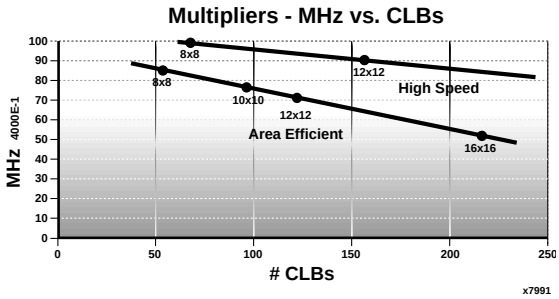


Figure 5: Trade-offs of Area Versus Speed Optimization for Multipliers

Ordering Information

This macro comes free with the Xilinx CORE Generator. For additional information contact your local Xilinx sales representative, or e-mail requests to dsp@xilinx.com.

XILINX®

Xilinx Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
Fax: +1 408-559-7114
E-mail: dsp@xilinx.com
URL: www.xilinx.com

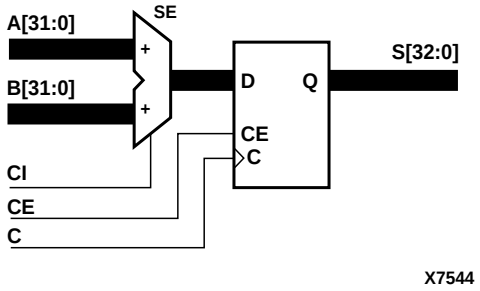


Figure 1: Core Schematic Symbol

Features

- Two data bus inputs: 2 to 32 bits wide
- Supports both 2's complement signed and magnitude-only unsigned data
- Drop-in modules for the XC4000E, EX, and XL families
- Registered output
- Clock Enable for output register
- Asynchronous Clear for output register
- Uses Fast Carry logic for high speed
- High performance and density guaranteed through Relational Placed Macro (RPM) mapping and placement technology
- Available in Xilinx CORE Generator

Functional Description

The Registered Adder module accepts two input buses, **A** and **B**, adds them, and registers the sum, **S**. The input data buses can be in either 2's complement signed or magnitude-only unsigned numbers.

Pinout

Signal names for the schematic symbol are shown in Figure 1 and described in Table 1.

Table 1: Core Signal Pinout

Signal	Signal Direction	Description
A[N-1, 0]	Input	A data input – value is added to B data
B[N-1, 0]	Input	B data input – value is added to A data
CI	Input	CARRY IN – initial carry logic input. Set this to GND if not used
CE	Input	CLOCK ENABLE – active high signal used to enable the transfer of data from the internal registers to output
C	Input	CLOCK – clocks the output register
S[N, 0]	Output	SUM DATA OUTPUT – the registered output of the adder

CORE Generator Parameters

The CORE Generator dialog box for this macro is shown in Figure 2. The parameters are as follows:

- **Component Name**: Enter a name for the output files generated for this module.
- **Data Width**: Select an input bit width from the pull-down menu. The valid range is 2-32. The same data width is applied to both the A and B inputs. The output size is automatically set to the input width plus one.
- **Sign**: Select Signed or Unsigned.

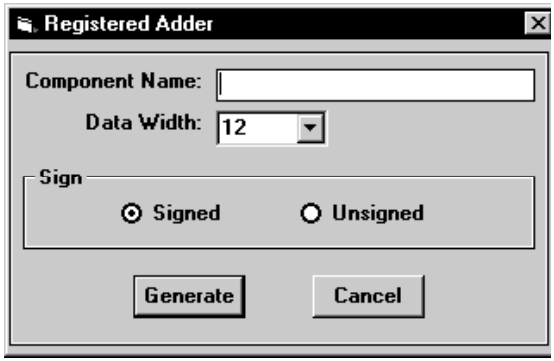


Figure 2: CORE Generator Dialog Box

Core Resource Utilization

Table 2 shows the number of CLBs required for each available bit width.

Ordering Information

This macro comes free with the Xilinx CORE Generator. For additional information contact your local Xilinx sales representative, or e-mail requests to dsp@xilinx.com.

Table 2: Bit Width versus CLB Count

Bit Width	CLB Count
2	2
3	2
4	3
5	3
6	4
7	4
8	5
9	5
10	6
11	6
12	7
13	7
14	8
15	8
16	9
17	9
18	10
19	10
20	11
21	11
22	12
23	12
24	13
25	13
26	14
27	14
28	15
29	15
30	16
31	16
32	17

Appendix 6

Xilinx 1024 Point High Performance FFT Reference Design Data Sheet (pp1-4)

(The full data sheet in PDF format is available on the enclosed CD)

Reference DESIGN

High-Performance 1024-Point Complex FFT

April 8, 1999 Application Note

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
FAX: +1 408-559-7114
Email: coregen@xilinx.com
URL: http://www.xilinx.com/ipcenter

Features

- High-performance 1024-point complex FFT
- 16-bit complex input and output data
- 2's complement arithmetic
- Integrated direct memory controller (DMAC) to simplify host and memory interfacing
- High performance and density guaranteed through Relational Placed Macro (RPM) mapping and placement technology

1 Functional Description

The xFFT1024 fast Fourier transform (FFT) Core computes a 1024-point complex FFT. The input data is a vector of 1024 complex values represented as 16-bit 2's complement numbers – 16-bits for each of the real and imaginary component of a datum.

2 Theory of Operation

The discrete Fourier transform (DFT) $X(k), k = 0, \mathrm{K}, N-1$ of a sequence $x(n), n = 0, \mathrm{K}, N-1$ is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-jnk2\pi/N} \quad k = 0, \mathrm{K}, N-1 \tag{1}$$

where N is the transform size and $j = \sqrt{-1}$. The *fast Fourier transform (FFT)* is a computationally efficient algorithm for computing a DFT.

The Xilinx 1024-point transform engine employs a Cooley-Tukey radix-4 decimation-in-frequency (DIF) FFT [1] to compute the DFT of a complex sequence. In general, this algorithm requires the calculation of columns or *ranks* of radix-4 butterflies. These radix-4 butterflies are sometimes referred to as *dragonflies.* Each processing rank consists of $N/4$ dragonflies. For $N = 1024$ there are 5 dragonfly ranks, with each rank comprising 256 dragonflies.

The FFT processor input-data for the Core is a vector of 1024 complex samples. The real and imaginary components of each sample are represented as 16-bit 2's complement numbers. The data is stored externally to the FPGA. An additional bank of scratchpad RAM is also required. The phase factors used in the FFT calculation are generated within the Core, so only the two

1024 × 32-bit banks of RAM discussed above are required to realize a complete transform processor. Like the input-data, the phase factors are kept to a precision of 16 bits.

All of the control signals required to interface the FFT module to external memory are generated by the Core. An integrated *direct memory controller (DMAC)* is also provided to allow the user to easily download data vectors for processing, and to read-back the previous transform result.

Two modes of operation are supported – *Continuous* and *One-shot.* The continuous mode of operation allows concurrent I/O and processing. As described above, 5 dragonfly ranks are performed to compute a 1024-point transform. When the Continuous mode of operation is selected, the host may download a new vector of data to the FFT engine while the 5th and final rank is being computed. Concurrent with the data load operation, the result of the current FFT is presented on the Core result databus. The FFT output samples are in *digit reversed order*, not *bit reversed order* as is the case with many conventional Cooley-Tukey radix-2 FFT algorithms.

3 Finite Word Length Considerations

3.1 Scaling

The radix-4 FFT algorithm processes an array of data by successive passes over the array. On each pass, the algorithm performs dragonflies, each dragonfly picking up four complex numbers and returning four complex numbers to the same addresses but in a different memory bank. The numbers returned to memory by the processor are larger than the numbers picked from memory. A strategy must be employed to accommodate this dynamic range expansion. A full explanation of scaling strategies and their implications is beyond the scope of this document, the reader is referred to several papers available in the open literature [2] [3] that discuss this topic.

The Xilinx 1024-point FFT Core scales dragonfly results by a factor of 4 on each processing pass. The scaling results in the final output sequence being modified by the factor $1/1024$. Formally, the output sequence $X'(k), k = 0,\mathrm{K}\ , N-1$ computed by the Core is defined in Eq. (2)

$$X'(k) = \frac{1}{1024} X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-jnk2\pi / N} \qquad k = 0,\mathrm{K}\ , N-1 \tag{2}$$

4 Pinout

The schematic symbol is shown in Figure 1.

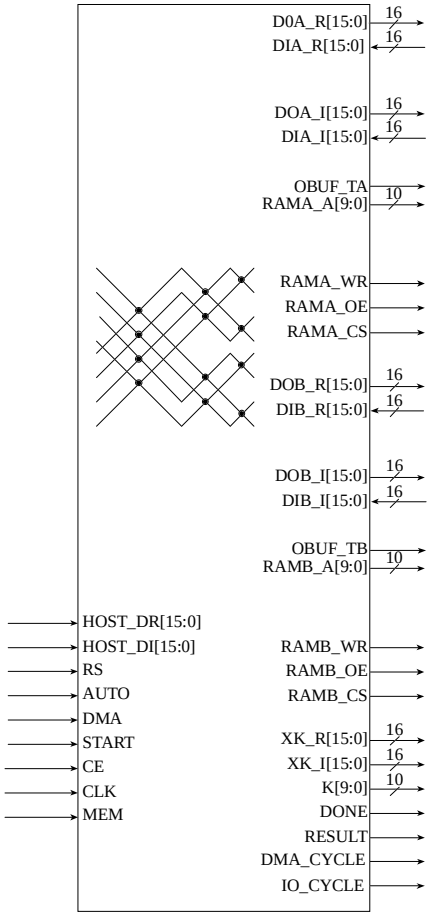


Figure 1: 1024-point FFT symbol.

Table 1 defines the module pin functionality.

5 Applications Information

Figure 2 shows the recommended method for interfacing the FFT Core to external memory. The data buses DOA_[R|I], DOB_[R|I], DIA_[R|I], DIB_[R|I], RAMA_A and RAMB_A must be registered in the FPGA IOBs. Figure 3 shows how to register the memory data bus in the FPGA IOBs using *OFDEX16* and *IFDX16* library components. Figure 4 illustrates the use of *OFDX* elements to register the address bus.

Signal	Direction	Description	Signal	Direction	Description
RS	Input	Master reset – active high	RAMA_OE	Output	RAM A output enable – active low
START	Input	Initiate FFT execution – active high	RAMB_OE	Output	RAM B output enable – active low
AUTO	Input	Defines *One-shot* or *Continuous* mode	RAMA_CS	Output	RAM A chip select – active low
DMA	Input	Direct memory access request – active high	RAMB_CS	Output	RAM B chip select – active low
CE	Input	Clock enable – active high	RAMA_WR	Output	RAM A write strobe – active rising-edge
CLK	Input	Master clock input – active positive edge	RAMB_WR	Output	RAM B write strobe – active rising-edge
HOST_DR	Input	Host databus – real	OBUF_TA	Output	Tri-state control for IOB FFs
HOST_DI	Input	Host databus – imag.	OBUF_TB	Output	Tri-state control for IOB FFs
DOA_R	I/O	RAM block A databus - real	RESULT	Output	FFT result strobe – active high
DOA_I	I/O	RAM block A databus – imag.	XK_R	Output	FFT result bus - real
DOB_R	I/O	RAM block B databus - real	XK_I	Output	FFT result bus – imag.
DOB_I	I/O	RAM block B databus – imag.	K	Output	FFT result index bus
RAMA_A	Output	RAM A address bus	DONE	Output	FFT complete strobe – active high[1]
RAMB_A	Output	RAM B address bus	DAM_CYCLE	Output	Host DMA in progress strobe – active high
MEM	Input	Defines single or dual address-space operation	IO_CYCLE	Output	Precedes activation of *STROBE* by 4 clock cycles and indicates a pending I/O operation[2]

NOTES:

1. *DONE* is active for one clock period.
2. *IO_CYCLE* is active for one clock period.

Table 1: 1024-point FFT pin definitions.

Appendix 7

Response from Dr. Chris Dick, Author of the Xilinx FFT Reference Design, Concerning Incorrect FFT Transform Results

From: Chris Dick <Chris.Dick@xilinx.com>
To: Andrew Wilkinson <awilkinson@totalise.co.uk>
Subject: Re: 1024 pint FFT Reference Design - Naming Contradiction ?
Date: 05 August 1999 19:25

Before you invest too much more time in the XC4000 FFT design, I want to make you aware that there is a 1024-point FFT for Virtex. This has not been released as yet, it is still in alpha testing. This FFT has a behv model and can easily and quickly be simlulated in a tool like Modelsim. The behv is available now, and in several weeks an edif netlist will be available.

When I simulated the XC4000 FFT I was in a Viewlogic environment. There are memory modules available in viewsim that permit the simulation of a complete transform.

Another approach I used was as follows: I have a C model of the FFT. You run your input data through this model and capture all the intermediate results from the butterflies to a file. You use this file as input to the simulator. You capture all the outputs from the simulation, and offline verify that the core produces all of the correct results. Now this is not as clean as inserting memory in the design, but I have had problems doing gate level simulations of the design that include two 1024x32 banks of memory.

I am not aware of an easy way to simulate the required storage in the Foundation simulator in a simple manner. I did try this at one point by building memories produced by the core generator. I recall that I had to construct a memory of the required size using several smaller memories. I did not have much sucess with this approach.

The underlying problem is that there is no behv. model for the XC4000 FFT and you are forced to do a gate level sim. This is of course not the case with the Virtex FFTs

Chris

--
Dr Chris Dick
Senior Systems Engineer
Xilinx Inc
2100 Logic Drive
San Jose
CA 95124