Design of a Decimation Filter and Testing of a High Speed $\Delta\Sigma$ Modulator

A THESIS

submitted by

SHAIK HUSSAINVALI

for the award of the degree of

MASTER OF TECHNOLOGY

DEPARTMENT OF Electrical Engineering
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

May 2012

THESIS CERTIFICATE

This is to certify that the thesis titled **Design of a Decimation Filter and Testing of a High Speed $\Delta\Sigma$ Modulator** , submitted by **Shaik Hussainvali**, to the Indian Institute of Technology Madras, for the award of the degree of **Master of Technology**, is a bona fide record of the work done by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Nagendra Krishnapura
Advisor,
Associate Professor,
Dept. of Electrical Engineering,
IIT-Madras, 600 036

Place: Chennai
Date: 25 May 2012

ACKNOWLEDGEMENTS

First of all, I would like to express deepest sense of gratitude to my advisor, Dr.Nagendra Krishnapura, whose energy, enthusiasm and constant support inspired me from the beginning till the end of the project. I am indebted to him for his ideas, valuable guidance and the encouragement throughout.

I would like to thank various faculty members of IIT Madras from whom I have benefited as a student. I especially thank Dr. Shanthi Pavan and Dr. Aniruddhan S. whose stimulating and inspiring lectures helped me to work in Analog and Mixed-Signal design.

I would like to thank Vikas and Debasish for their help during testing of Delta-Sigma modulator. I would like to thank Praveen for his help during the course of the project. I would also like to thank lab supporting staff Mrs. Sumathy and Mrs. Janaki for their help and support.

I would like to thank Sudhir, Venky, Dileep, Ajay, Rafi, Ram Prasath and my friends in the DSDL lab for their support and the friendly atmosphere in lab. I would also like to thank my brother and sister, who have given constant moral support remotely. Finally, I dedicate this thesis to my family members and friends who make my any little success meaningful.

ABSTRACT

This project involves the testing and design of a Decimation filter for a High speed $\Delta\Sigma$ Modulator compensated for more than unity feedback delay. The chip tested as part of this thesis contains a fast feedback path to compensate for delays which are greater than one clock cycle. This led to realize very high sampling speeds on 0.18μm technology. The ADC tested was a 4-bit modulator with sampling speed of 750 MHz. The chip consumes $47.6mW$ of power from a 1.8 V supply. The measured dynamic range is 82 dB for OSR of 25 and is 80 dB for OSR of 10. Decimation filter is designed to work at 1 GHz and Decimation by a factor 25 is implemented in a cascade of two stages, each stage decimating by a factor 5. Polyphase decomposition reduces the maximum operating frequency to 200 MHz so that the entire design can be implemented with 0.18 μm standard cell library. Appropriate grouping of partial products reduces the number of explicit additions/multiplications and minimizes the power consumption. The design is implemented in 0.18 μm CMOS process from UMC. It occupies an area of $\mathbf{790 \times 740}$. The design consumes a total power of about 17.06 mW from a 1.8 V supply. The simulated SNR for the Decimation filter is 93.1543 dB when fed from the output of an ideal $\Delta\Sigma$ modulator.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Introduction

Continuous time(CT) $\Delta\Sigma$ modulators have found increasing use in modern electronic components. They are high speed and consume low power. CT time $\Delta\Sigma$ modulators have an inherent anti aliasing property thus saving the need of an explicit high order anti-alias filter which saves both power and area.Another important property that they possess is noise shaping. CT $\Delta\Sigma$ modulators filter out the quantization noise in the signal band thus making the quantizer look like having more number of bits(hence better quantization) for quantization. Hence we can as well design a simple 1 bit quantizer and make it look like 4 or 5 bit quantizer by the help of the noise shaping property. Also in a given technology the sampling frequency of a CT $\Delta\Sigma$ modulators is limited by the excess loop delay(ELD) present in the loop because of the quantizer and the DAC. Most of modulators have ELD compensation but they are mostly valid for ELD less than one clock cycle [3][7]. This again limits the sampling frequency. Thus to realize higher sampling frequency we need to design the modulator which can compensate for much higher ELD. The chip [10] which is tested as part of this thesis work employs one such technique which can compensate for ELD between one to 2 clock cycle delays [11]. This led to a very high sampling frequency for a 4-bit $\Delta\Sigma$ modulator designed on 0.18μm technology. The chip with an $OSR = 25$ was tested to work at 750 MHz and the DR was 82 dB and the SNR_{max} was 68.7 dB. The bandwidth is 15 MHz/37.5 MHz and it consumes 47.6 mW from a 1.8 V supply. Also a new high speed data acquisition technique to capture the $\Delta\Sigma$ modulator data directly from the field programmable gate array(FPGA) is also discussed along with its potential applications for testing.

Once the quantization noise is shaped out of the passband as shown in Figure 1.1, a low pass filter(LPF) is required to remove the out of band components and a decimator is needed to bring the sampling rate back to the Nyquist rate from the oversampled rate. Both these tasks are performed by a decimation filter, which as its name suggests, low pass filters and decimates (brings down the sampling rate) the input signal.

The aim of this project is to test and to design a low power decimation filter for a high speed $\Delta\Sigma$ modulator.The decimation filter is designed to work at 1 GHz. Since the low pass filtering and the decimation are performed on the output signal of the $\Delta\Sigma$ modulator, the entire decimation filter system works in the digital domain.

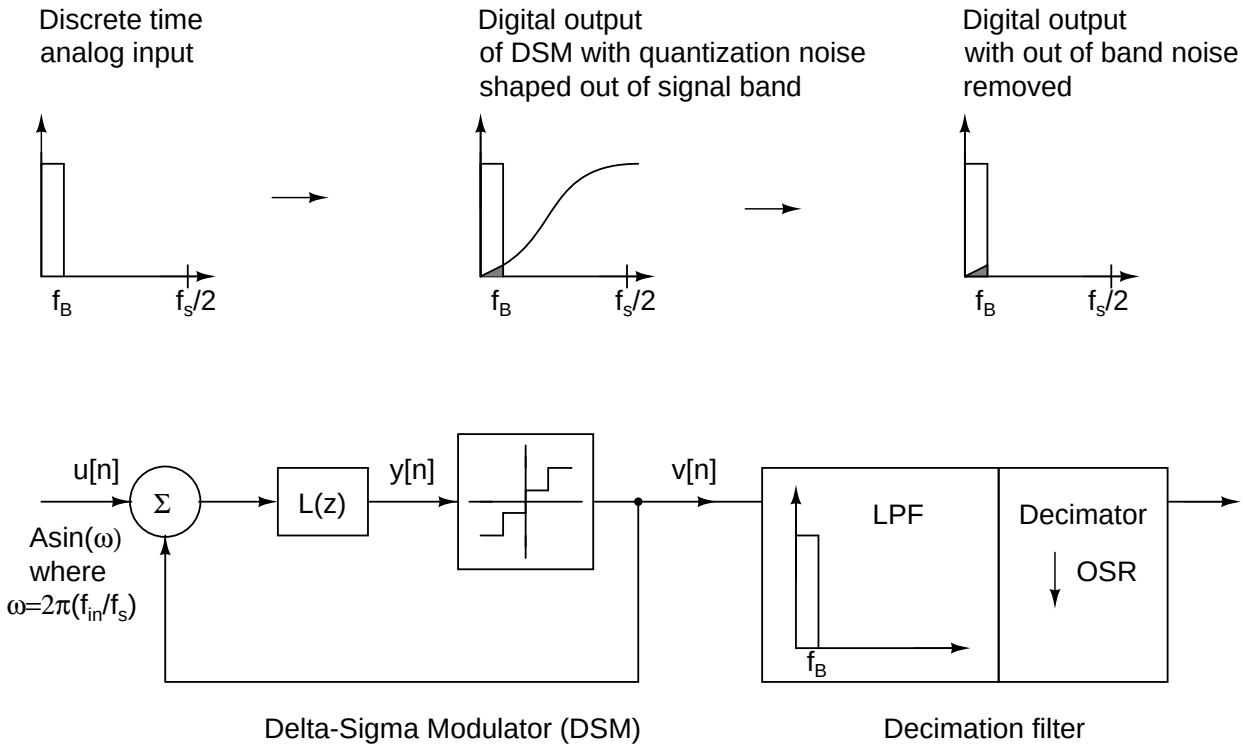


Figure 1.1: Block diagram of decimation filter for a $\Delta\Sigma$ modulator. f_B: maximum inband frequency, OSR: Oversampling ratio.

1.2 Organization

Chapter 2 discusses the test setup used to test the chip along with the measurement results. Finally a new data acquisition technique using Chipscope tool thus replacing the need of a logic analyzer is described.

Chapter 3 introduces the basics of decimation filter design. Specifications of the $\Delta\Sigma$ modulator and decimation filter are considered and a final two stage design of decimation filter is derived.

Chapter 4 gives information about the digital synthesis of the decimator using standard cells and CAD tools. The simulation results are given in this chapter.

Chapter 5 discusses the complete architecture of the $\Delta\Sigma$ analog to digital converter(ADC) after combining the Decimation filter to DSM. The simulation results are given in this chapter.

Chapter 6 concludes the thesis

The details of the chip which was tested can be found in [10] and also Appendix A, which discusses the effect of excess loop delay on a delta sigma modulator and also discusses commonly used techniques to compensate them. Appendix A further discusses the compensation technique used in the chip being tested and also gives an overview of the chip architecture.

CHAPTER 2

Test setup and measurement results

2.1 Test chip

A snapshot of the fabricated test chip is shown in Fig. 2.1. The chip has 48 pins and occupies an area of $36.7\,mm^2$.

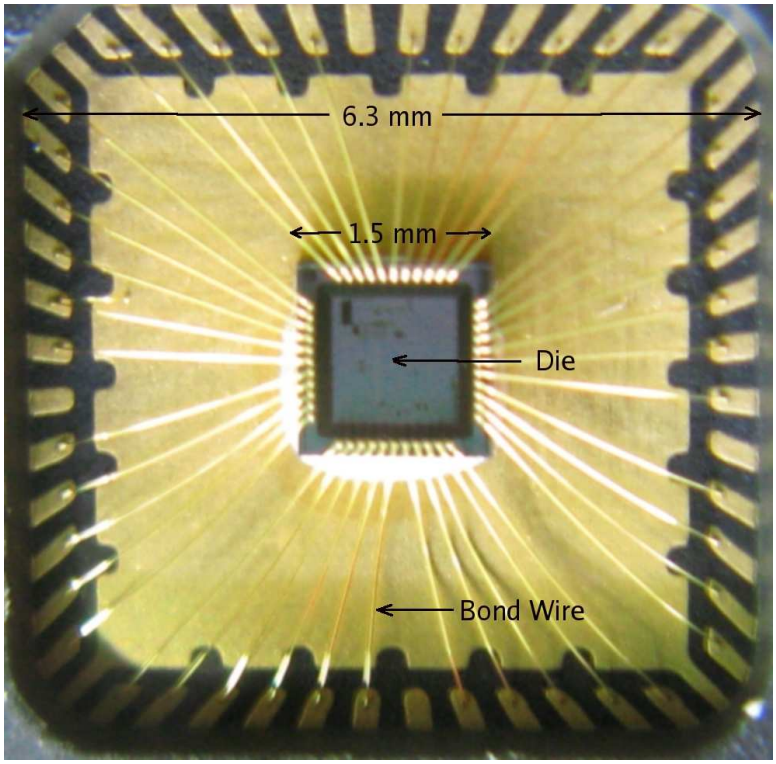


Figure 2.1: Chip used for testing, [10]

The pin details of the test chip is shown in the Figure 2.2. Out of the 48 pins, 4 pins are not used and remaining 44 pins are allocated as shown in Table 2.1. Appendix B shows the description of each pin.

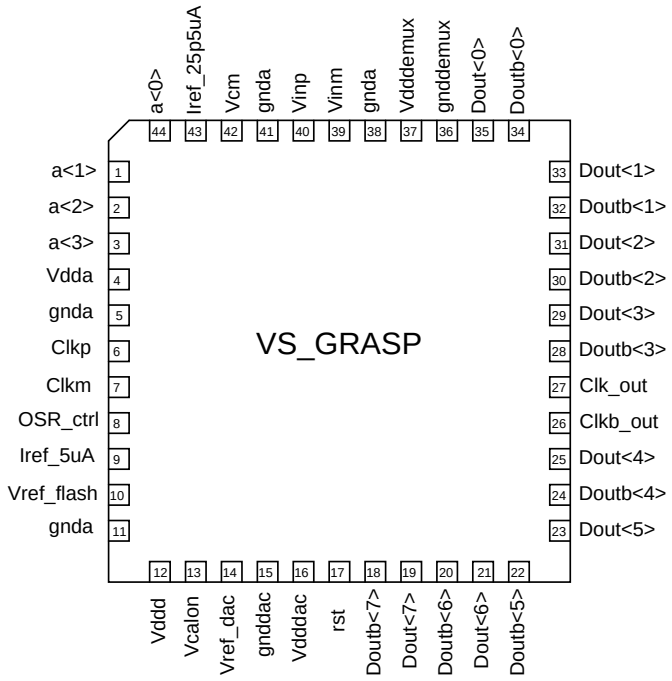


Figure 2.2: Pin details of the chip, [10]

Table 2.1: Pin allocation in the test chip, [10]

Signal	Number of pins used
Input signal	2
Input clock	2
Output digital data	18
Power supplies	10
References	5
Control signals	7

2.2 Test setup

The top level block diagram of the test setup is shown in Figure 2.3 . A signal source(Agilent 33250A) drives an input tone to a passband filter which suppresses the harmonics and the wideband noise of the signal source. A balun transformer (North-Hills 100 kHz-20 MHz balun) converts this spectrally purified tone into a differential signal that serves as the input to the test chip. The clock signal is generated by a signal source (Agilent E4422B), which can generate low-jitter sine wave at 800 MHz. An RF transformer (Mini-Circuits ADT1-1WT) is used to convert this clock to differential clock, which is

fed to the test chip.

The modulator outputs are then passed through low voltage differential signaling(LVDS) buffers and are written on a First in,First out(FIFO) at $f_s/2$. The FIFO is implemented on an FPGA(Xilinx Virtex 5 board) and it stores the 8bit ADC outputs on its FIFO. This data stored in the FIFO is then read out to a Logic Analyzer at a much lower rate. A FIFO is used between test chip and logic analyzer because the logic analyzer available to us can operate at a maximum speed of 200 MHz. So, the FPGA captures the data at a high speed and then transfers it at a lower speed to the logic analyzer. From the logic analyzer, deserialized data is transferred to a PC for post-processing.

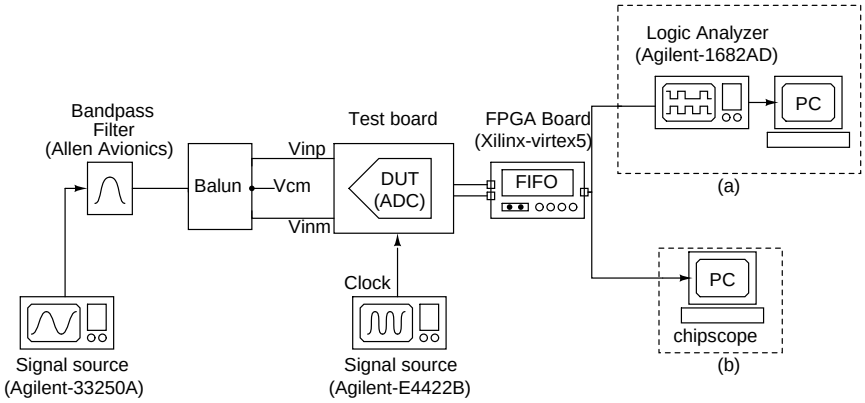


Figure 2.3: Test setup using either (a)logic analyzer (b)Chipscope

The logic analyzer is a costly and scarce instrument hence a new way of capturing data was used by the help of a tool called Chipscope provided by Xilinx. Details of test setup using chipscope will be discussed in the last section of this chapter. Figure 2.4 shows a snapshot of the test setup.

2.3 Test board

Various test boards were used for testing. From the earlier test results it was suspected that the packaging of the chip has an effect on the output as the chip packaged by SPEL was not running at 800 MHz whereas the chip packaged by EUROPRACTICE

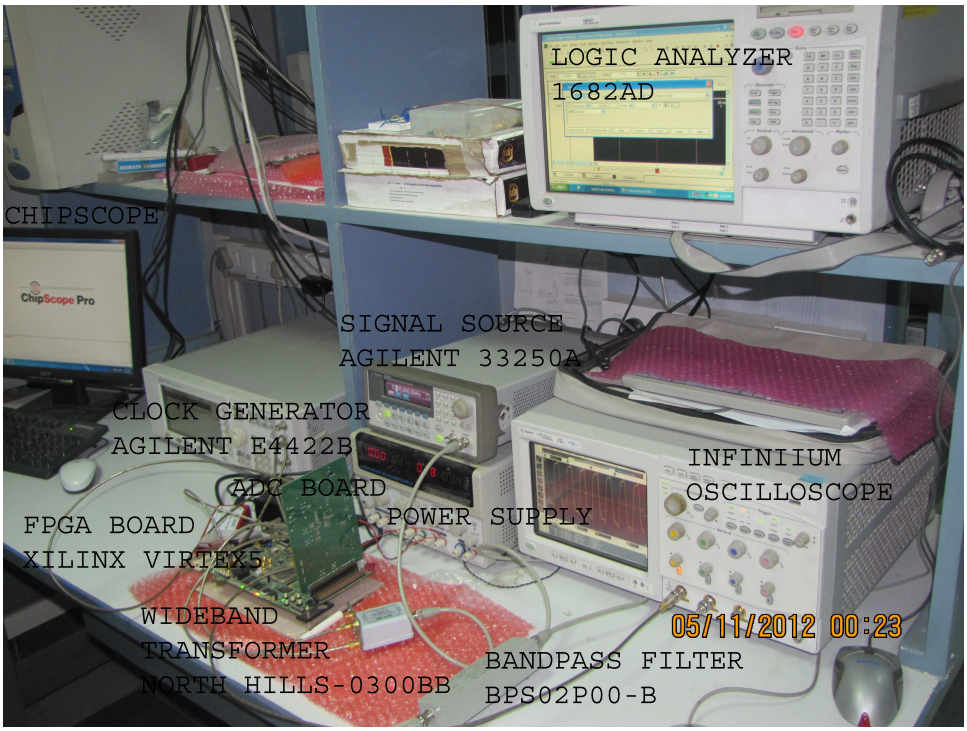


Figure 2.4: Snapshot of test setup

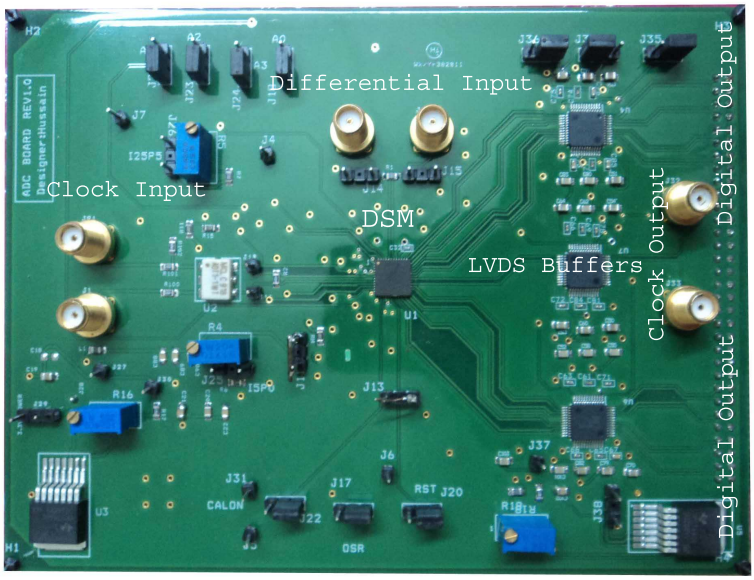


Figure 2.5: Board used for the testing

was running at 800 MHz. So another printed circuit board was designed which contains DSM and LVDS buffers on the single board instead of having on two different boards as used in the earlier work. The snapshot of the test board with labels is shown in the Figure 2.5. The measurement results presented in the next section contains readings taken from this board. The dimension of the board is 14.5 cmX11.5 cm.

2.4 Measurement results

All the measurements are done at $f_s = 750\ MHz$ instead of 800 MHz, as the modulator was becoming unstable at 800 MHz. The chip consumes $47.6mW$ of power from a 1.8V supply. Figure 2.6 shows the idle channel response of the modulator.

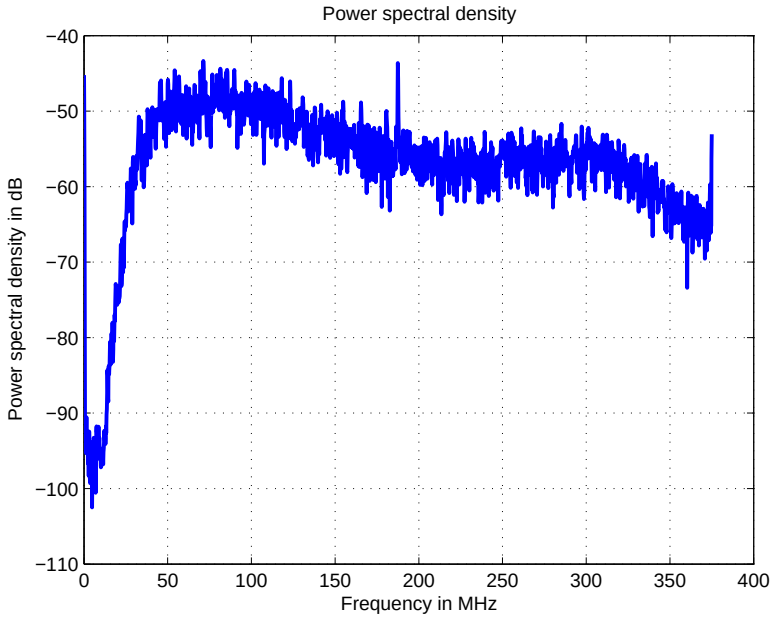


Figure 2.6: Idle channel response for OSR of 25

It has tones at $f_s/4$ due to the input reference substractor used in the flash, which is operating at $f_s/4$. Idle channel response of the modulator for different sampling rates and their in-band noise values are as shown in figures. The modulator becomes unstable at $f_s = 800\ MHz$.

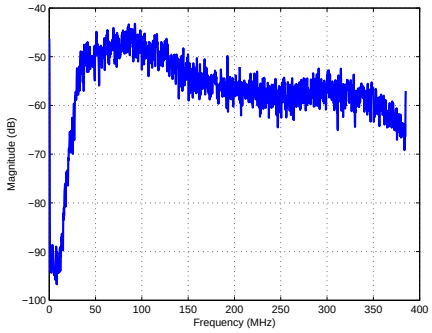


Figure 2.7: $f_s = 770\,MHz$, In-band noise = -71.52dB

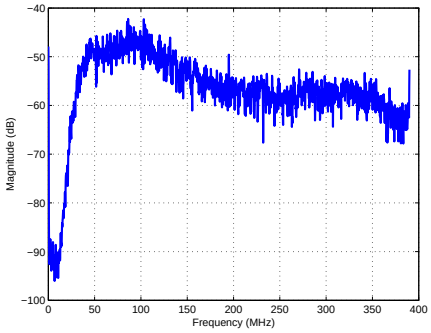


Figure 2.8: $f_s = 780\,MHz$, In-band noise = -71.29dB

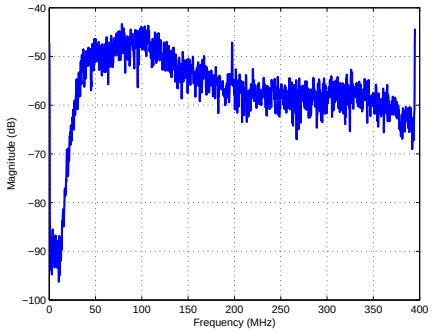


Figure 2.9: $f_s = 790\,MHz$, In-band noise = -69.12dB

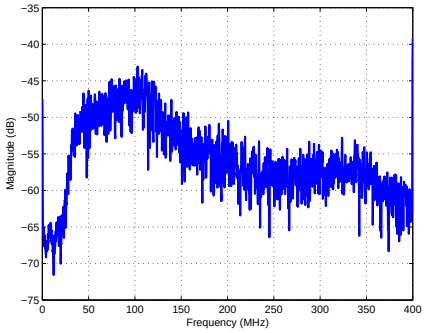


Figure 2.10: $f_s = 800\,MHz$, In-band noise = -47.92dB

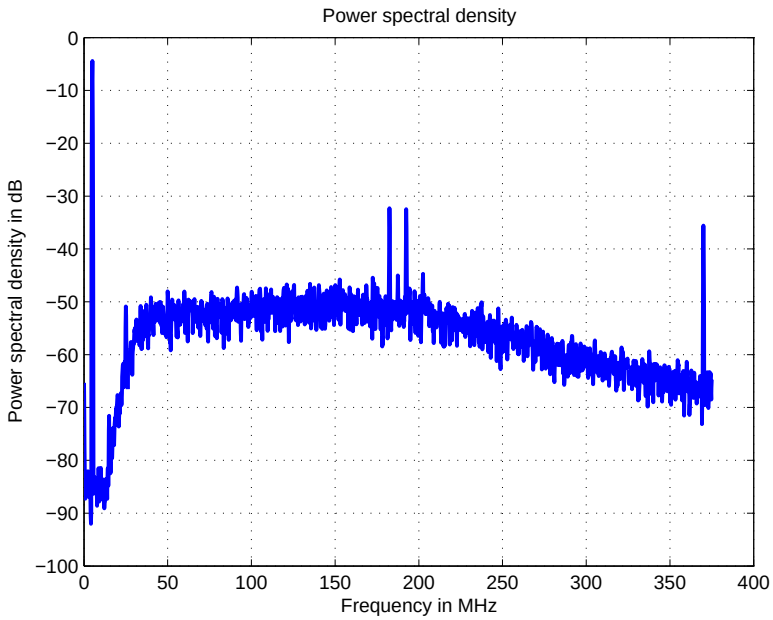


Figure 2.11: Output PSD for OSR of 25 and input of 5MHz

For an input of -1.6 dBFS the output of the modulator is shown in Figure 2.11, the SQNR is 68.7 dB and SNDR 63 dB.

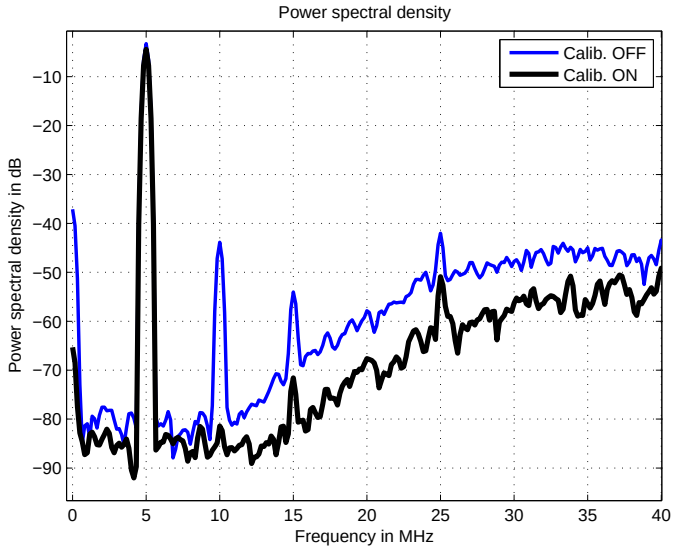


Figure 2.12: PSD with and without calibration

The DAC calibration scheme used in the design significantly reduces the distortion at the output and when compared to a case with calibration turned on the plots are as shown in Figure 2.12. The harmonics with calibration on are about 77 dB below the input tone.

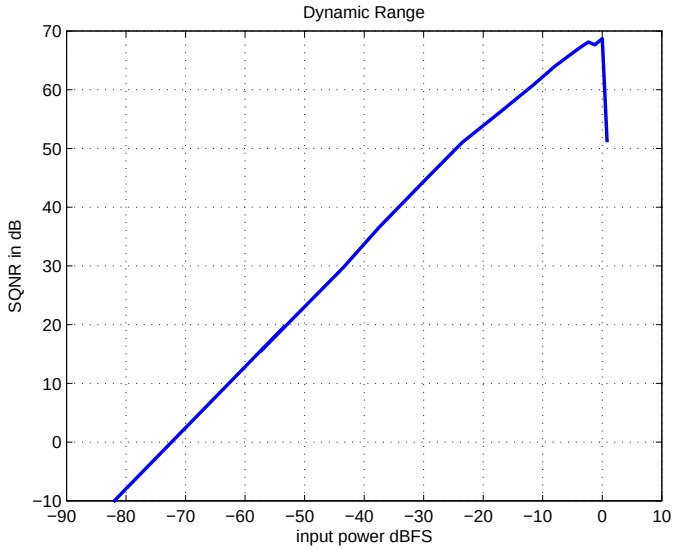


Figure 2.13: SQNR vs input amplitude for OSR= 25 and input of 5MHz

The plot of SQNR for different input amplitudes is shown in Figure 2.13. The measured dynamic range is 82 dB for OSR of 25.

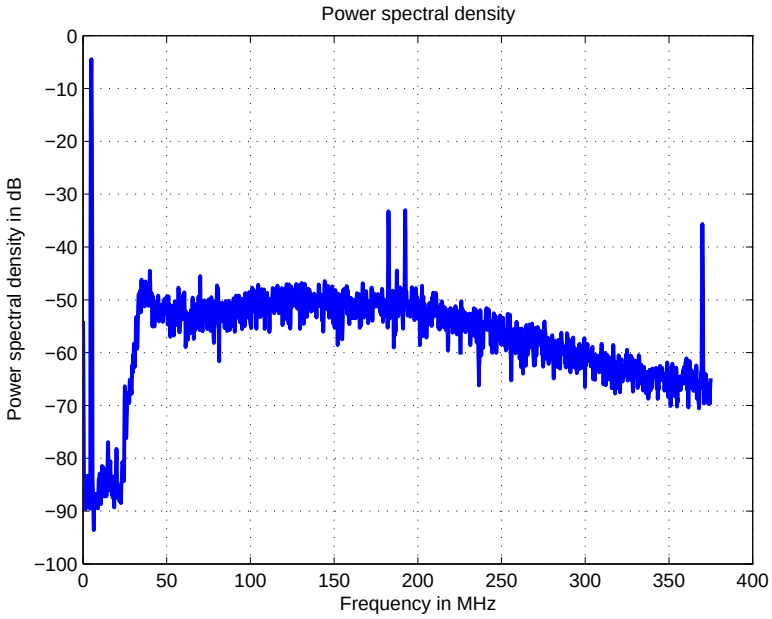


Figure 2.14: Output PSD for OSR of 10 and input of 5MHz

For the case with OSR of 10 the plot for a 5MHz input at -1.6dBFS is shown in Figure 2.14. The SQNR and SNDR are 67.27 dB and 65.73 dB respectively .

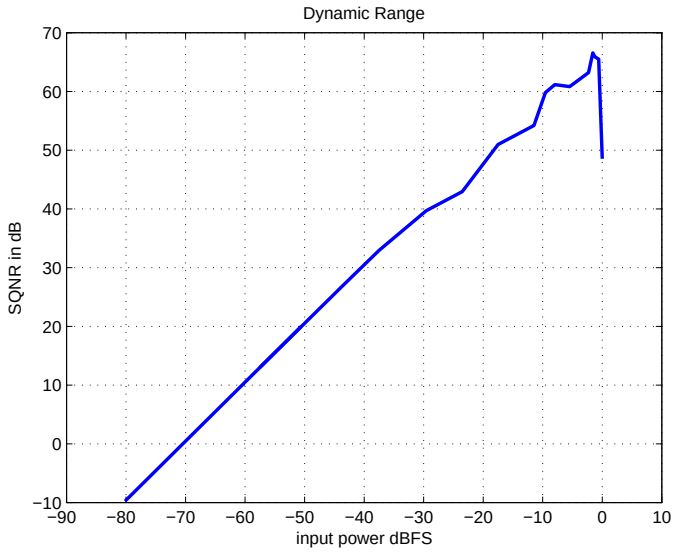


Figure 2.15: SQNR vs input amplitude for OSR of 10 and input of 5MHz

Also the plot of SQNR over different amplitudes is shown in Figure 2.15. The measured dynamic range is 80 dB.

2.5 Using Chipscope for high speed data acquisition

In the earlier section, we discussed how we were using the logic analyzer along with the FPGA board to capture the data from the ADC. Since logic analyzers are scarce and costly instruments so there was a need to come up with a way to directly capture the data from the FPGA itself. Also capturing data into the logic analyzer and then transferring it to the main machine was also a time consuming process. By using a tool called Chipscope [1] which is provided as part of the Xilinx ISE package, it is possible to capture data at high speed on the FPGA itself. It can also capture data at very high speeds(the speed of the design on the FPGA) which is much higher than the maximum sampling rate possible via a commercially available logic analyzer.

2.5.1 Chipscope overview

ChipScope is an embedded, software based logic analyzer that allows monitoring the status of selected signals in a design in order to detect possible design errors. It samples the signals needed to be analyzed and stores them on the FPGA FIFO. These are then read to the Chipscope graphical user interface (GUI) on your host computer. The Xilinx ChipScope tools package has several modules that you can add to your Verilog design to capture input and output directly from the FPGA hardware. These are:

- ICON(Integrated CONtroller): All of the ChipScope cores use the Joint Test Action Group(JTAG) Boundary Scan port to communicate with the host computer via a JTAG downloading cable (either a parallel or a USB cable). The ICON core provides a communications path between the JTAG Boundary Scan port of the FPGA device and the other ChipScope cores (ILA, VIO). The ICON core can communicate with up to 15 ILA and/or VIO cores at any given time. However,

individual cores cannot share their control ports with any other core. Therefore, the ICON core needs a distinct control port for every ILA and VIO core.

- VIO(Virtual Input/Output): A module that can monitor and drive signals in your design in real-time. You can think of them as virtual push-buttons (for input) and LEDs (for output). These can be used for debugging purposes, or they can incorporated into your design as a permanent I/O interface. Unlike the ILA core, no storage resources are required.
- ILA(Integrated Logic Analyzer): A module that lets you view and trigger on signals in your hardware design. Think of it as a digital oscilloscope (like ModelSim's waveform viewer) that you can place in your design to aid in debugging.

2.5.2 The ChipScope Analyzer Tool

The ChipScope Analyzer tool interfaces directly to the ICON, ILA, and VIO cores. By inserting an ICON and an ILA into your design and connecting them properly, you can monitor any or all of the signals in your design. Even nicer is that ChipScope provides you with a convenient software based interface for controlling the ILA, including setting the triggering options and viewing the waveforms. With this tool, the user can configure the FPGA device, set up trigger conditions and view the design signals in graphical form. The analyzer provides an intuitive interface to determine the functionality of the design. The ChipScope Analyzer tool consists of two distinct applications: the server and the client. The server is a console application that connects to the JTAG chain of the FPGA device using a JTAG download cable. The client is a GUI application that allows interacting with the devices in the JTAG chain and the cores that are found in those devices. The server and client applications can be running on the same computer (local mode) or on different computers (remote mode). Remote mode is useful when the user needs to debug a system that is in a different location or to share a single system resource with other design team members. If the user desires to debug a system that is connected directly to its local computer via a JTAG download cable, there is no need to start the server manually.

2.5.3 Using Chipscope in the design

To start using chipscope we first need to create all the modules that we need for analyzing and capturing the data. Then after creating these modules we add them to the top level of our verilog design file which we then burn on the FPGA(a Virtex 5 board was used for all the measurements). We will first of all need the Chipscope ICON core(which is necessary for all the designs). Then since we only want to capture and analyze data hence we need an ILA core and we don't need any VIO core. The number of ILA cores depends on the number of signals we want to capture which range from a few bits in some designs to hundreds of bits in some other designs. Each ILA core has the capability to sample a 32 bit signal [1]. It is like storing 32 bit samples in a FIFO and each FIFO acts like one ILA core.

For our delta sigma modulator we need to capture only 8 output digitized bits. We also need to capture one read signal which is the output from our top level as it will act as the trigger signal for our chipscope module.

Once we have finalized the specifications we start generating the cores [1]

- First, we start generating the cores by the help of Xilinx Core-generator tool present in the Xilinx ISE tool. We then start creating an ICON core. After selecting the correct device we select the number of ports which is one for our case(as we will be needing just one ILA core). We then click next.
- We now generate the ILA core in a similar fashion. We select the number of bits to be analyzed which is 9 bits(8 outputs along with a read signal trigger). Then we set the number of trigger bits which we set to all the same 9 bits which are to be analyzed(we will have the option of selecting which bit to trigger on in the chipscope GUI). We also set the box which says that data is same as trigger.
- Finally, after creating all the cores we now instantiate all of them at the top level design in a way similar to how we instantiate a simple verilog module. We map the data pins of the ILA core(which will be captured by Chipscope) to the signals we want to sample in our top level design(to map analyzer signals to signals in other sub modules we need to bring them out as pins to the top level). We also need to map the clock of the ICON core. This is the clock on which all the data will be sampled by the ILA core. After the mapping and instantiation, the design can be synthesized and implemented using the ISE synthesis tool. During

the synthesis process, these cores will act as ”black boxes”, their behavior will be provided during the implementation process in the form of netlists. Before synthesizing and implementing the design, create and add to the project an UCF file to specify the assignment of signals to the pins of FPGA device and generate the bit file that needs to be burned on the FPGA.

After generating the configuration file, the FPGA device on the development board can be configured either with the iMPACT tool or with the ChipScope Analyzer tool. In this example design, the ChipScope Analyzer tool will be used to configure the FPGA device. After that, it will be possible to set the trigger condition, to capture the values of selected signals, and to display the signal waveforms in graphical form. To do this we start the Chipscope analyzer software provided by the Xilinx ISE toolkit. In the console we select Xilinx USB cable to connect the machine to the FPGA via its USB port. After the console detects all the chipscope modules in our FPGA, then all the signals that were declared in the ILA module will appear. We then open the trigger module and select the 9th bit or the read signal bit as our trigger and set its trigger condition to *1*, such that the system samples the FPGA signals when the read switch is turned on at the sampling clock as specified in the ICON core [1].

After doing all the above we save the whole project. Next time we just open the project click on the trigger button and then set the read switch on the FPGA board to start capturing data. The captured data can then be exported to matlab readable file.

CHAPTER 3

Decimation Filter Design and Architecture

3.1 Fundamentals of Decimation filter

There are many applications where the signal at a given sampling rate needs to be converted to another signal with a different sampling rate. Discrete-time systems with unequal sampling rates at various parts of the system are called *multirate* systems. The process of reducing the sampling rate is generally called *decimation*, and the multirate structure used for decimation is called *decimator*. Accompanying a decimator is usually a low pass filter, which along with the decimator makes a *decimation filter*. In this chapter, the basic theory of decimation filters and the block level implementation of decimation filter [6] is discussed.

3.2 Cascade equivalence of decimation filters

A complex multirate system is formed by an interconnection of the up-sampler, the down-sampler, and components of an LTI digital filter. In most applications, these two appear in cascade as shown in Figure 3.1. For efficient computational results a particularly useful cascade equivalence property for our design is shown in Figure 3.1 and the equivalence is proved [6] by the manipulation of frequency domain properties of the signals.

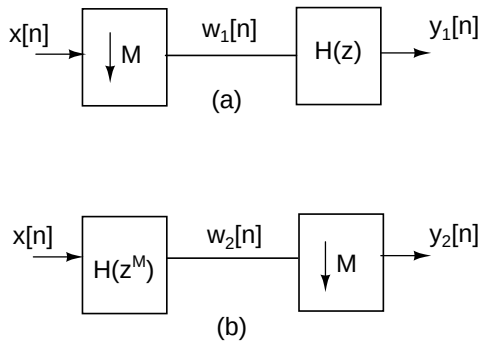


Figure 3.1: Cascade equivalence of decimation filter, [6]

3.3 Specifications of $\Delta\Sigma$ Modulator

As we have stated in chapter 1, the goal of this project is to design a decimation filter for a $\Delta\Sigma$ modulator. So before we set about designing the decimation filter, the specifications of the $\Delta\Sigma$ modulator which has already been designed, taped out and tested has to be stated and understood. The output of the $\Delta\Sigma$ modulator which has to be filtered and decimated should be generated from a model since it is this same output that forms the input to the required decimation filter.

The $\Delta\Sigma$ modulator for which the decimation filter has to be designed has a sampling frequency, f_s = 1 GHz and OSR 25 with a 4-bit output. The general block diagram of a $\Delta\Sigma$ modulator and its decimation filter shown in Figure 1.1 is redrawn here in Figure 3.2 for our particular case. The $\Delta\Sigma$ modulator has an $OSR = 25$ and gives a 4-bit output. The Over-Sampling Ratio or $OSR = 25$ means that the sampling rate is 25 times the Nyquist frequency. So to bring the system back to Nyquist rate after decimation, we have to decimate by a factor 25. The maximum frequency component present in the input signal or the input bandwidth, f_B can be calculated as shown below.

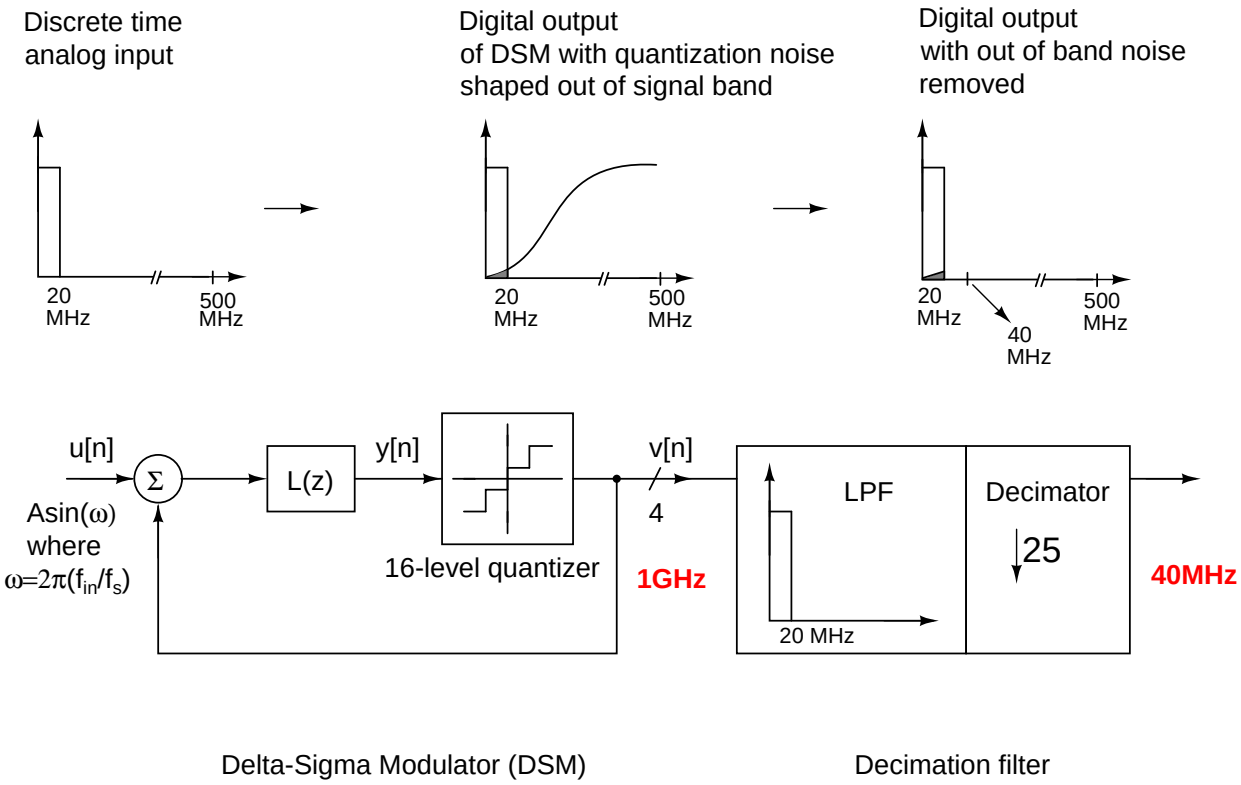


Figure 3.2: Block diagram of a decimation filter for the $\Delta\Sigma$ modulator

Since $f_s = OSR\times$ Nyquist rate,

$$f_s = \text{OSR} \times 2f_B \tag{3.1}$$

$$\implies f_B = \frac{f_s}{2 \times \text{OSR}} = \frac{1\,\text{GHz}}{50} = 20\,\text{MHz} \tag{3.2}$$

After the $\Delta\Sigma$ modulation, the quantization noise will be shaped out of the signal band and thus the decimation to Nyquist rate should be accompanied by low pass filtering which removes the out of band noise components. Since the input bandwidth is 20 MHz, we essentially need a structure with a low pass filter with passband 20 MHz and a decimator with decimating factor 25. The output of the decimation filter will be a digital signal with sampling rate 1 GHz/25 =40 MHz, which is the Nyquist rate.

The noise-shaping of the modulator is determined by the Noise-Transfer Function (NTF) of the $\Delta\Sigma$ modulator. The NTF of the $\Delta\Sigma$ modulator is given in (3.3) and the

magnitude response of this NTF is shown in Figure 3.3.

$$NTF(z) = \frac{(1 + 1.352z^{-1})(1 - 1.998z^{-1} + z^{-2})(1 - 1.988z^{-1} + z^{-2})}{(1 - 1.204z^{-1} + 0.3771z^{-2})(1 - 1.43z^{-1} + 0.6585z^{-2})} \tag{3.3}$$

To generate the input signal for the decimation filter, the $\Delta\Sigma$ modulator with the NTF

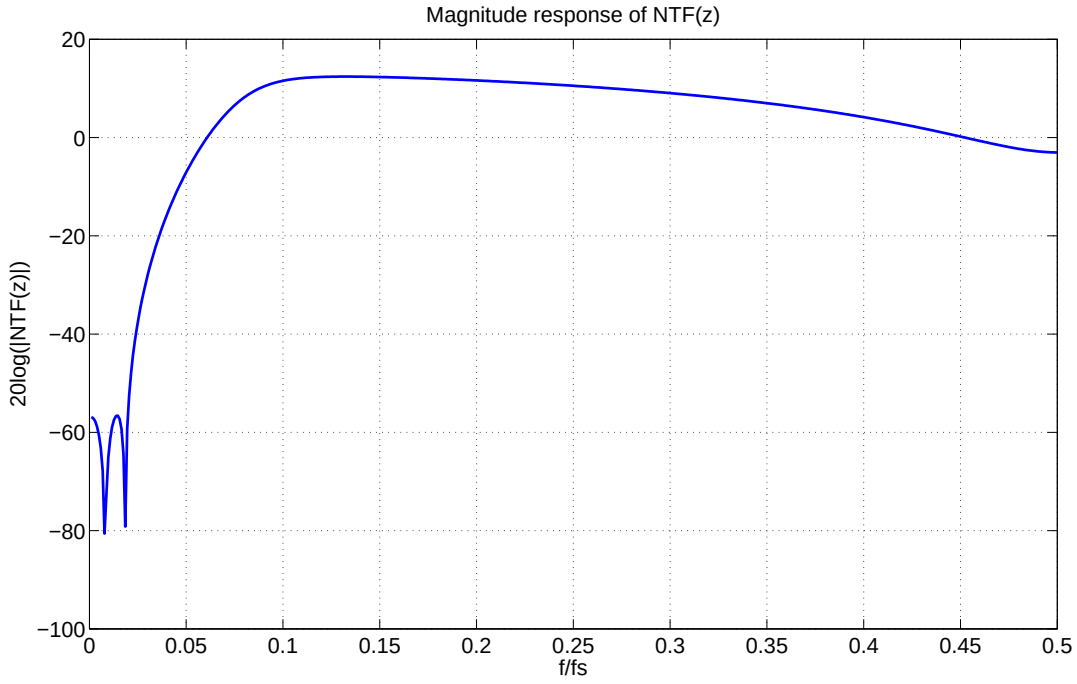


Figure 3.3: Magnitude response of the NTF of the $\Delta\Sigma$ modulator.

defined in (3.3), OSR 25 and a 16-level quantizer is modelled and simulated in MATLAB using Schreier's Delta Sigma Toolbox [9]. An input signal $A\sin(2\pi(f_{in}/f_s)n)$ is given to the modulator with the value of A chosen to be slightly less than the Maximum Stable Amplitude (MSA) of the modulator. Considering that the OSR is 25, which means the decimation factor is also 25, it is better to have the number of input points and FFT bins of the form 5^n. Thus a 5^7-point FFT of the output of the $\Delta\Sigma$ modulator is shown in Figure 3.4.

For this design of the $\Delta\Sigma$ modulator, the maximum SNR after decimation and low pass filtering can be calculated. Assuming an ideal low pass filter with sharp response,

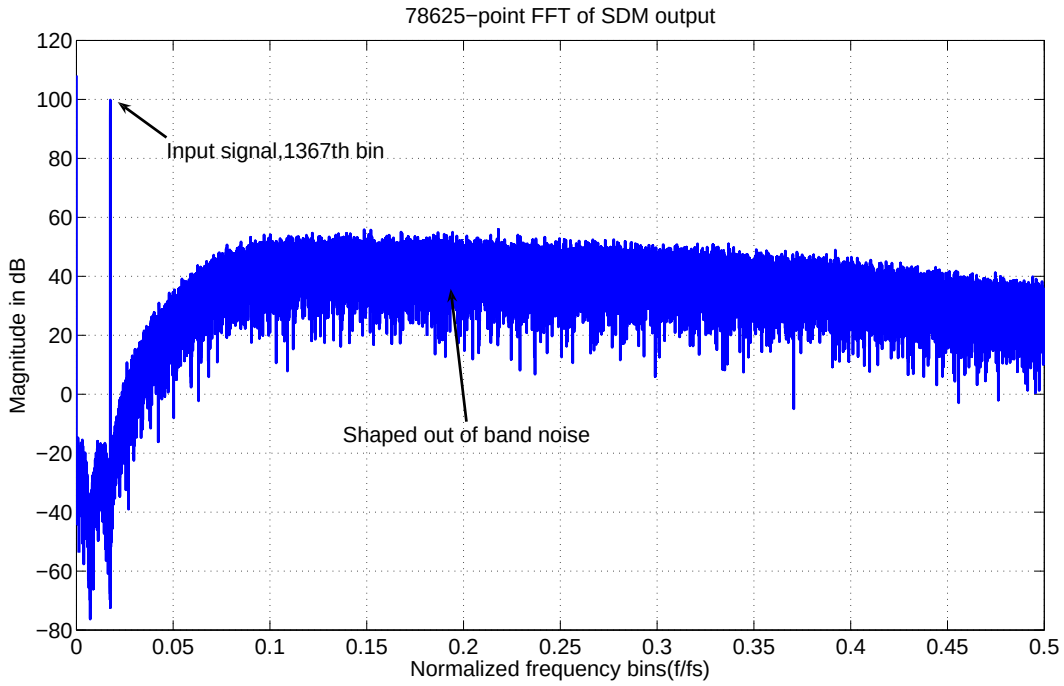


Figure 3.4: Output of the $\Delta\Sigma$ modulator or equivalently the input of the Decimation filter.

SNR can be calculated using the *simulateSNR* command in the Delta-Sigma toolbox [9]. The maximum possible SNR we can get after decimation filter is calculated to be 96 dB.

3.4 Decimation filter design

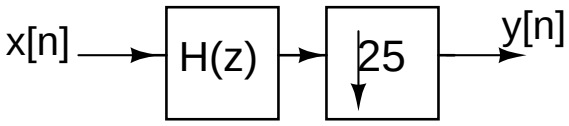


Figure 3.5: Single stage implementation.

The simplest design approach for decimation filter is single stage filter shown in Figure 3.5. In this a digital LPF is designed to meet the decimation filter specifications and it is followed by a down-sampler. Since the signal is oversampled by a large factor, the ratio of signal bandwidth(f_B) to half the oversampled rate, relative bandwidth, is very

less. Filtering at oversampled rate and then down sampling to the nyquist rate requires a very sharp filter working at the oversampled rate, which explains in the Figure 3.6. Therefore the required order of the filter is very high.

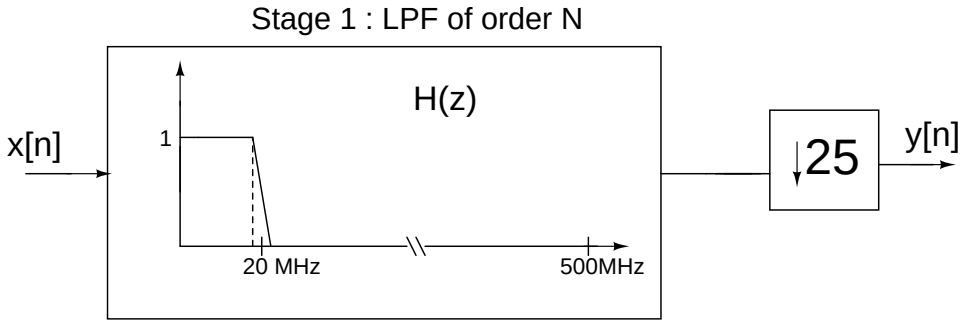


Figure 3.6: Single stage decimation filter.

A simple Kaiser's optimal FIR filter order estimation[3] is used to find the required filter order. The approximate order of the filter that meets the required specifications is $N \approx 240$. Power dissipation of digital filters is directly proportional to their frequency of operation. In single stage design entire digital LPF operates at oversampling rate, which increases the power dissipation. The disadvantages of single stage design approach are

- Excessively large filter orders because of narrow transition bandwidths.
- High power dissipation as the entire filter operates at fs.
- Inefficient use of hardware is because one out of every M computed samples is passed to output.

3.4.1 Multi-stage decimation

The disadvantages of single stage can be overcome by multistage design. Filtering the alias band noise in multiple stages and down sampling by an appropriate factor in each stage ensures efficient decimation. The decimator is designed as a cascade of two or more stages such that the overall decimation ratio is product of decimation ratios of each stage. This is called multistage decimation.

Since the required decimation factor 25 can be written as 5×5, it is possible to implement the decimation filter in two stages as we can see in Figure 3.7. Due to myriad reasons, it is conventional to use a sinc filter in the first stage of a multi-stage decimation filter[2][5]. Thus in this design also we will use a sinc filter for the first stage.

The entire decimation filter has now become a two stage system with a sinc filter in the first stage and an FIR filter in the second. For both stages, decimating factor is 5 (Figure 3.7).

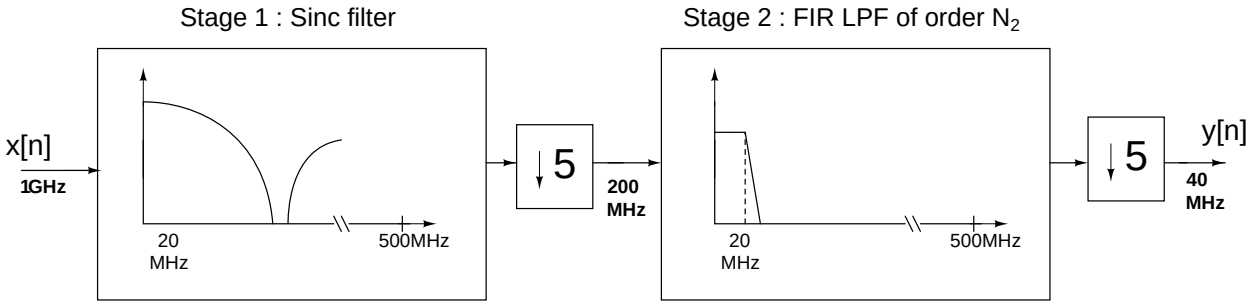


Figure 3.7: Block diagram of the two stage decimation filter.

As we will see later on, one of the main reasons to use a sinc filter in first stage, the simple architecture, breaks down due to the high speed requirement(1 GHz) of this design. Thus we are actually free to use any FIR filter for the first stage.

3.5 Sinc filter characteristics

Sinc filter is proven to be very efficient for the first stage in a multi-stage decimator and can be used to decimate to a rate four times the Nyquist rate[2][5]. Since OSR is 25, the fact that a sinc filter can be used to decimate to four times the Nyquist rate means that the decimation factor in first stage, M_1 can be as high as $25/4 = 6.25$. i.e. M_1 should be ≤ 6.25. Since both $M_1 = M_2 = 5$ in our case, this condition is met. Thus a sinc filter can be used as the first stage FIR filter in the two stage implementation scheme

and it will decimate the sampling rate to five times the Nyquist rate. A general sinc filter $H_s(z)$ of order K with $(r \times M)$ taps is of the form

$$H_s(z) = \left(\frac{1}{rM} \left(\frac{1 - z^{-rM}}{1 - z^{-1}} \right) \right)^K \tag{3.4}$$

where the nulls are at $\pm l(f_s/rM)$ where l = 1, 2, . . rM-1 as shown in Figure 3.8. Note that

$$\left(\frac{1 - z^{-rM}}{1 - z^{-1}} \right) = 1 + z^{-1} + z^{-2} \ldots z^{-(rM-1)} \tag{3.5}$$

Thus a sinc filter of order K is simply K moving average filters in cascade.

Now consider the input signal of bandwidth 20 MHz in Figure 3.8. We know that when decimated by a factor of 5, noise at the frequencies in the range $(200 - 20)$ MHz $< f < (200 + 20)$ MHz, $(400 - 20)$ MHz $< f < (400 + 20)$ MHz will alias into the passband (± 20 MHz of f_s/M,$2f_s/M \ldots$ because signal bandwidth is only 20 MHz).

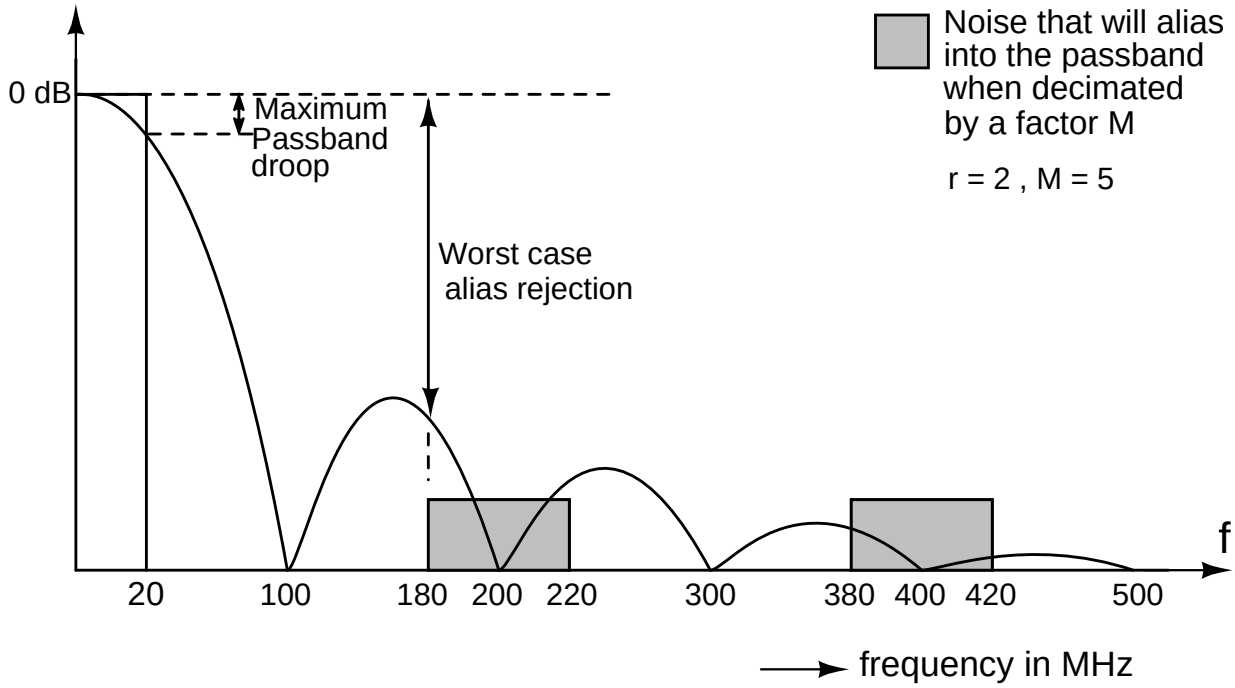


Figure 3.8: Frequency response of sinc filter showing noise aliasing to the passband. Diagram is not to scale.

Table 3.1: Variation of maximum droop and worst alias attenuation with the value of r (K=4, M=5), [6].

Value of r	Maximum passband droop at f_B	Worst alias rejection at $(f_s/M - f_B)$
1	-0.5507 dB	-75.04 dB
2	-2.2950 dB	-76.79 dB
3	-5.2840 dB	-79.78 dB
4	-9.6600 dB	-84.15 dB

The decimation factor, M of the decimator which follows the sinc filter is fixed to be 5. Value of r has to be chosen after considering the passband droop of sinc filter into account. Droop is the deviation of the sinc filter response from the ideal value of unity (0 dB) in the passband. The maximum passband droop occurs at $f_B = 20$ MHz. The worst case alias rejection is the attenuation at the frequency $(200 - 20)$ MHz = 180 MHz. Passband droops and worst case alias rejection for different values of r (for $K = 4$, $M = 5$) are shown in Table 3.1. The plots of fourth order sinc filter frequency responses for $r = 1$, $M = 5$ (5 taps) and $r = 2$, $M = 5$ (10 taps) are compared in Figure 3.9.

Usually a maximum passband droop of 3dB or above is difficult to correct using an equalizer[4, p. 15]. So from Table 3.1, it is clear that we can choose r to be one or two. The value of r is chosen to be 2 since this will greatly relax the requirements of the second stage FIR filter. For $r = 2$, passband droop is about 2.3 dB as shown in Figure 3.10. However if a lower droop is required, a five tap filter has to be chosen. Alias rejection and droop increase as K increases. A fourth order sinc filter is found to be sufficient to meet the SNR requirement in the final stage. So K is chosen to be 4. Thus the final transfer function of the sinc filter becomes

$$H_s(z) = \left(\frac{1}{10} \left(\frac{1 - z^{-10}}{1 - z^{-1}} \right) \right)^4 \tag{3.6}$$

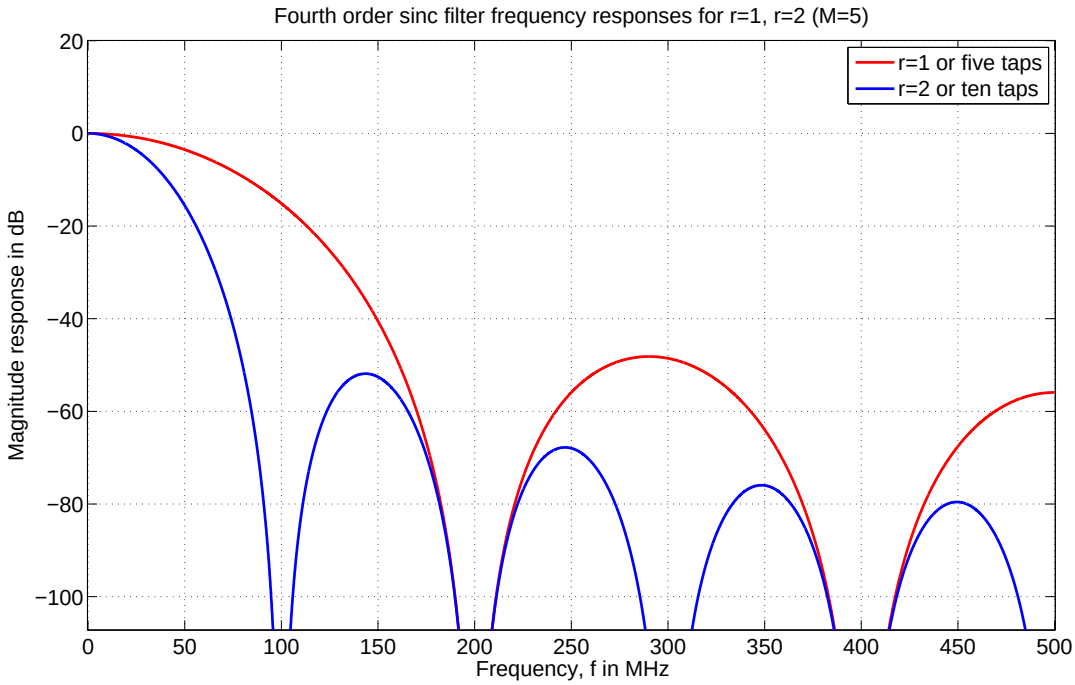


Figure 3.9: Frequency responses of sinc filter with $r = 1$ and $r = 2$. For both responses, $K = 4$, $M = 5$. Note that for the 10-tap filter alias rejection is better but passband droop is higher.

For $z = e^{j\omega}$,

$$H_s(e^{jw}) = \left(\frac{1}{10} \left(\frac{\sin(10\omega/2)}{\sin(\omega/2)} \right) \right)^4 \tag{3.7}$$

The frequency response of the sinc filter given in (3.7) is plotted in Figure 3.9. The maximum passband droop and worst case alias rejection for the chosen sinc filter are shown in Figure 3.10.

As we have mentioned before, the input to the sinc filter is the output from the DSM. Since we have designed the sinc filter for a 4-bit unsigned input, the output from the DSM has to be scaled and offset. The output of DSM corresponding to an input signal $A\sin(2\pi(f_{in}/f_s)n)$ is given as the input to the sinc filter. The output signal after the sinc filtering and decimation is shown in Figure 3.11.

The SNR after sinc decimation is calculated to be 54.69dB. This 14-bit output forms

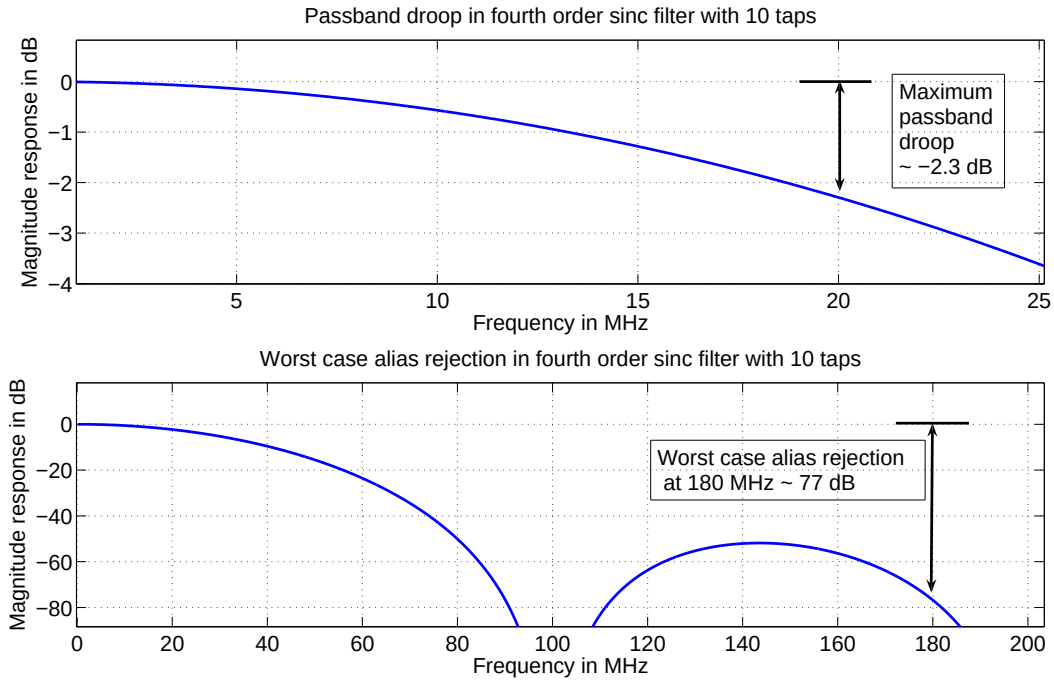


Figure 3.10: Maximum passband droop and worst case alias rejection for sinc filter with $K = 4$, $r = 2$, $M = 5$ given in (3.6).

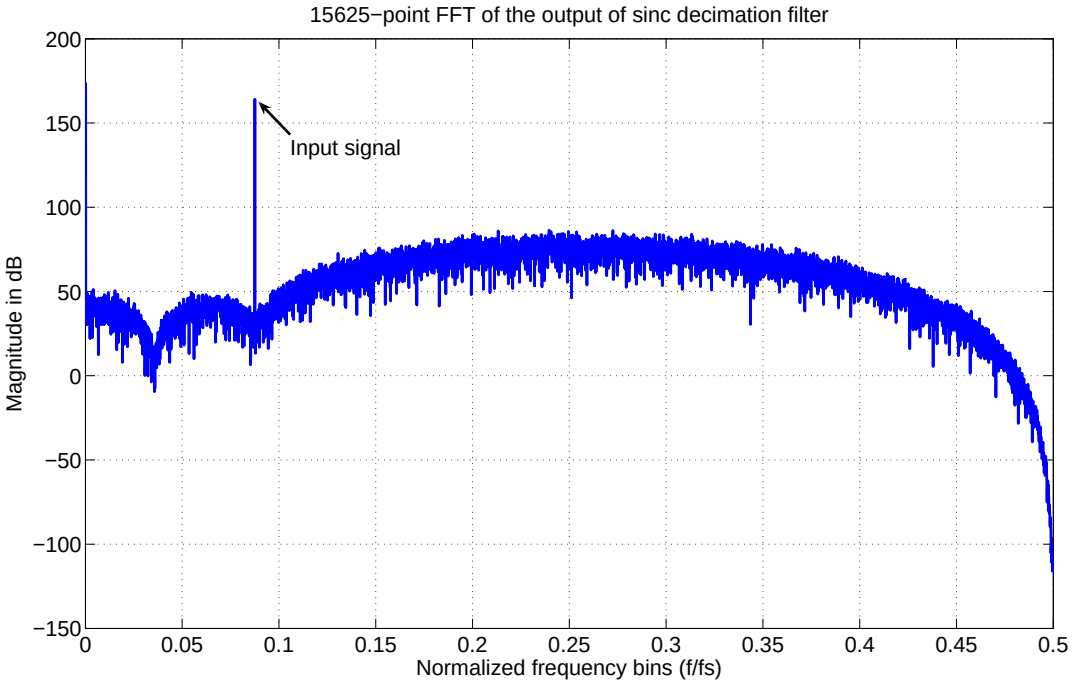


Figure 3.11: 5^6-point FFT of the output of the fourth order 10-tap sinc decimation filter. Normalized frequency 1 after decimation corresponds to 200 MHz. Thus the input signal which is 1367^{th} bin is ≈ 17.5 MHz.

the input to the second stage of the decimation filter.

3.6 FIR filter characteristics

The ideal FIR filter response should be as shown in Figure 3.7. It is required that after the second stage FIR filter, the decimation filter should meet the specification of 93 dB SNR at the output. The input of the FIR filter is the output of the first stage sinc filter. This output is derived for an input sinusoidal signal of frequency ≈ 17.5 MHz (1367^{th} bin) and is plotted in Figure 3.11. This signal when passed through the stage 2 FIR filter and decimator should give a signal of 93 dB SNR. Considering all this, we can set about to determine the FIR filter coefficients.

3.6.1 Determining the FIR filter coefficients

FIR filter coefficients are determined using Parks-McClellan optimal FIR filter design algorithm. This can be done in MATLAB using two commands *firpmord* and *firpm*. This command uses the Remez exchange algorithm and Chebyshev approximation theory to design filters with an optimal fit between the desired and actual frequency responses. The filters are optimal because the maximum error between the desired frequency response and the actual frequency response is minimized.

Our aim is to use the Parks-McClellan algorithm to obtain an FIR filter $H_{pc}(z)$ of minimum order which meets the SNR requirement. The basic filter parameters used to design the filter $H_{pc}(z)$ are given in Table 3.2. Since the order of an FIR filter increases with lesser passband ripple, steeper transition and higher stopband attenuation, the values in Table 3.2 are found using trial and error so that the order of $H_{pc}(z)$ is as low as possible while still meeting the SNR requirement at the output.

MATLAB commands *firpmord* and *firpm* returns an FIR filter of order 60. The

Table 3.2: Parameters for the FIR filter $H_{pc}(z)$

Passband ripple, δ_p	< 1 dB
Stopband attenuation, δ_s	50 dB
Passband edge, f_p	18.25 MHz
Stopband edge, f_s	24 MHz
Sampling frequency, f_s	200 MHz

frequency response of the FIR filter, $H_{pc}(z)$ is shown in Figure 3.12 and Figure 3.13.

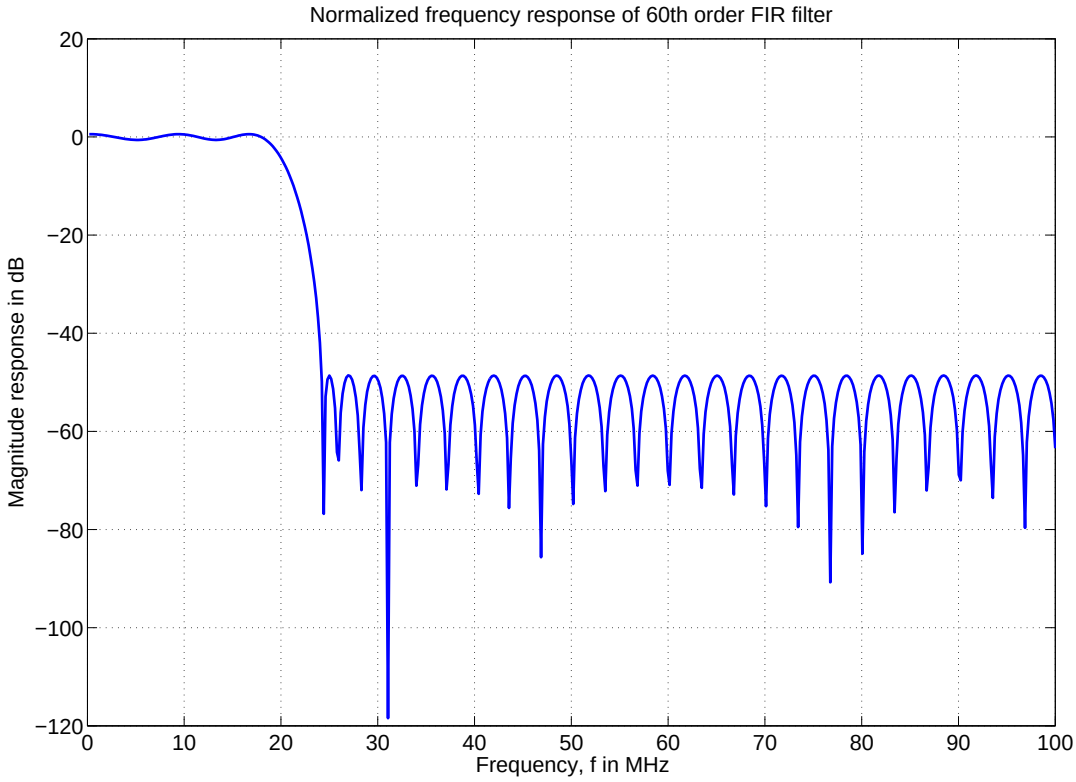


Figure 3.12: Normalized frequency response of the FIR filter $H_{pc}(z)$ obtained using MATLAB.

From Figure 3.13, it can be seen that the stop band attenuation is around 50 dB, transition band is between 18 MHz and 24 MHz. A closer look of the passband frequency response of both FIR filter in second stage and sinc filter in first stage given in Figure 3.14 reveals that the passband ripple of FIR filter along with the passband droop of the sinc filter deteriorates the response for frequencies closer to 20 MHz.

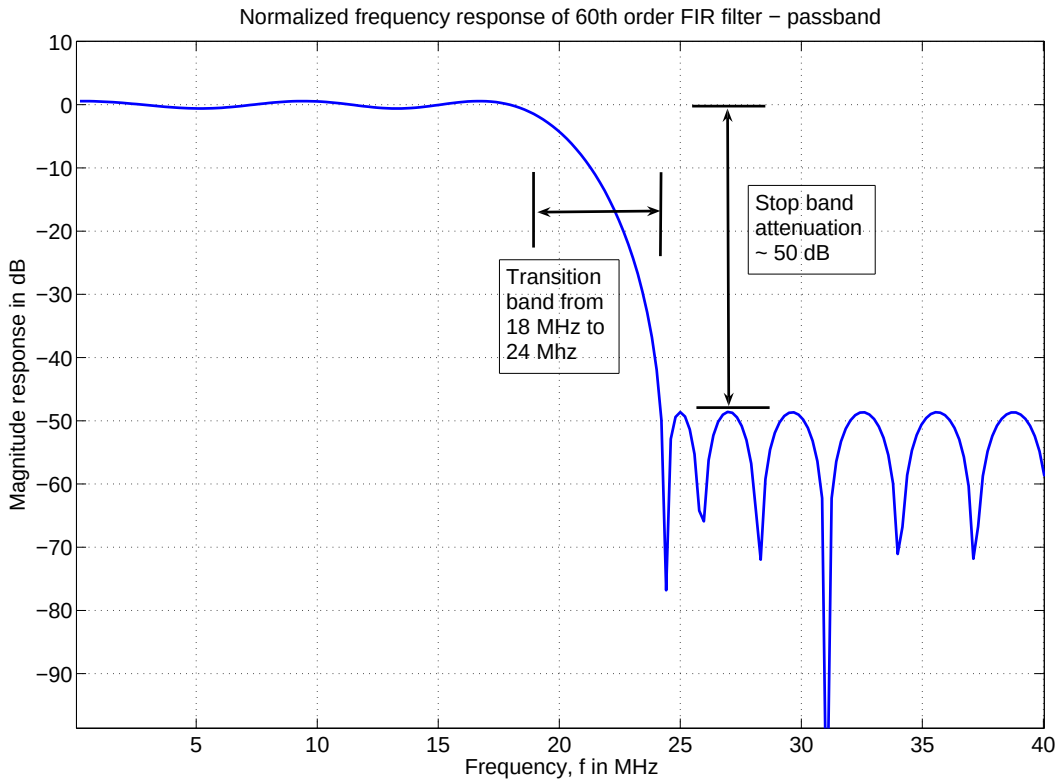


Figure 3.13: Normalized frequency response of the FIR filter $H_{pc}(z)$- a closer look. Note that the stop band attenuation is around 50 dB and the transition band is from 18 MHz to 24 MHz.

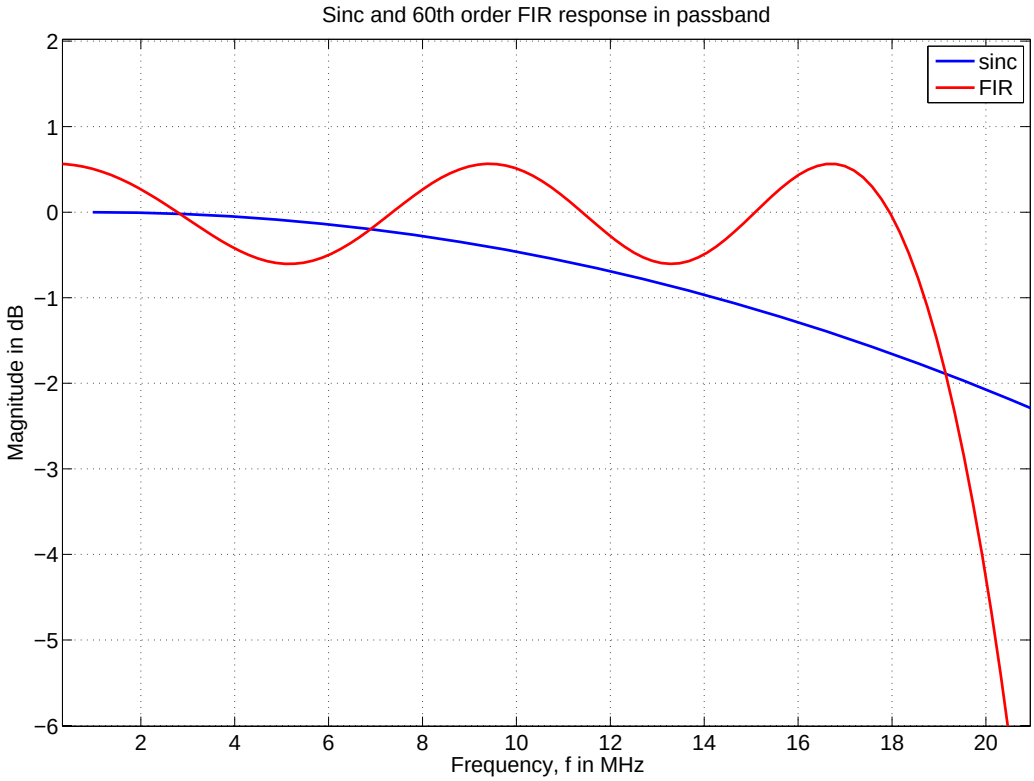


Figure 3.14: Droop of 4^{th} order 10-tap sinc filter and FIR ripple within passband 20 MHz. Note that the FIR filter ripple is within 1 dB.

3.6.2 Sinc and FIR filter frequency responses

To put the entire decimation filter in perspective, the normalized frequency responses of both the FIR filter and the sinc filter are plotted in the same graph in Figure 3.15. Note that FIR filter transition is much more steeper than that of the sinc filter. The output of the decimation filter after the second stage FIR filtering and decimation for an input sinusoidal signal of frequency ≈ 17.5 MHz(1367^{th} bin) is shown in Figure 3.17.

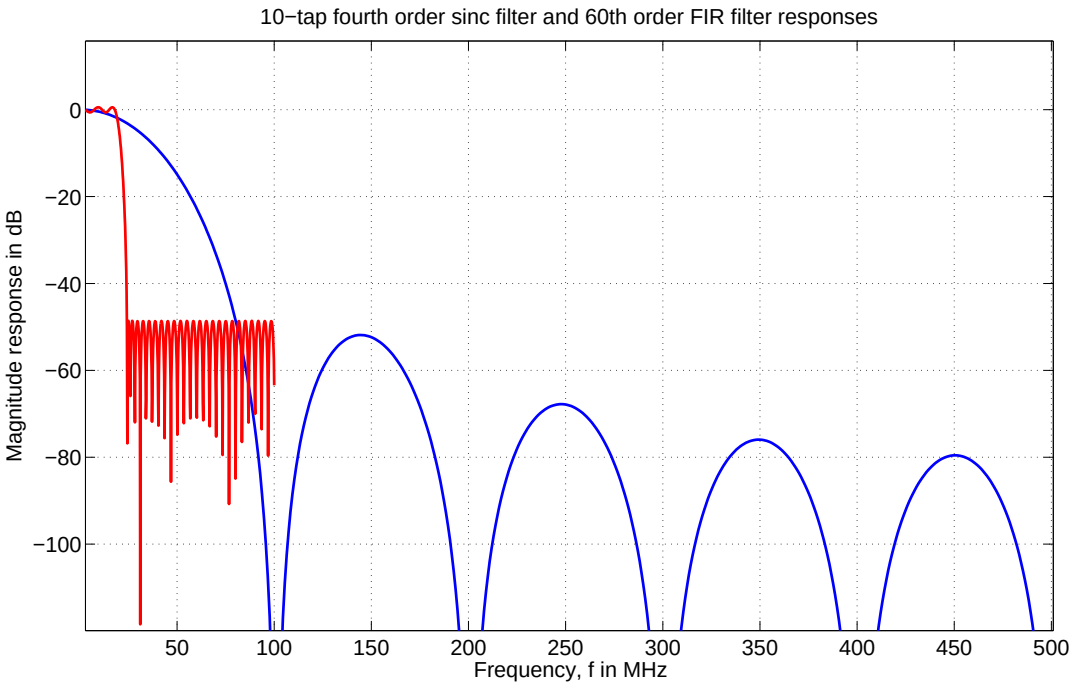


Figure 3.15: Frequency response of the 60^{th} order FIR filter $H_{pc}(z)$ and the 4^{th} order 10-tap sinc filter super imposed. Note that half the sampling rate is 100 MHz and 500 MHz for FIR and sinc respectively.

3.6.3 64^{th} order FIR filter

Note that the sinc filter in (3.6) can be written as

$$H_s(z) = \left(\frac{1}{10} \left(\frac{1 - z^{-10}}{1 - z^{-1}} \right) \right)^4 \tag{3.8}$$

$$= \left(\frac{1}{10} \left(\frac{(1 - z^{-5})(1 + z^{-5})}{1 - z^{-1}} \right) \right)^4 \tag{3.9}$$

This factorization of $(1-z^{-10})$ into $(1-z^{-5})(1+z^{-5})$ helps us to exploit the cascade equivalence discussed in section 3.2 and the factor $(1 + z^{-5})^4$ can thus be pushed to the lower frequency side(200 MHz) as shown in Figure 3.16 to become $(1 + z^{-1})^4$. This leaves only a scaled fourth order sinc filter with 5 taps operating at 1 GHz.

The 60^{th} order FIR filter obtained using Parks-McClellan algorithm, $H_{pc}(z)$ can be multiplied with the factor $(1+z^{-1})^4$ pushed from the sinc stage. This makes the order of the final FIR filter, $H_f(z)$ in (3.10) 64. This is illustrated in Figure 3.16. The combined frequency response of two stages of the decimation filter is a fourth order sinc filter with 10 taps in cascade with 60^{th} order $H_{pc}(z)$ or equivalently a fourth order sinc filter with 5 taps in cascade with 64^{th} order $H_f(z)$.

$$H_f(z) = H_{pc}(z) \times (1 + z^{-1})^4 \tag{3.10}$$

$$= \sum_{k=0}^{k=63} hf_k z^{-k} \tag{3.11}$$

If we choose to implement $(1 + z^{-1})^4$ and $H_{pc}(z)$ separately, $(1 + z^{-1})^4$ has to operate at 200 MHz, on the other hand, if $H_{pc}(z)$ and $(1 + z^{-1})^4$ are clubbed together to form a single filter $H_f(z)$, all the computations can be done at a lower rate of 40 MHz as we will see in the later sections. Thus it is advantageous to implement $(1 + z^{-1})^4$ and $H_{pc}(z)$ as a single FIR filter $H_f(z)$.

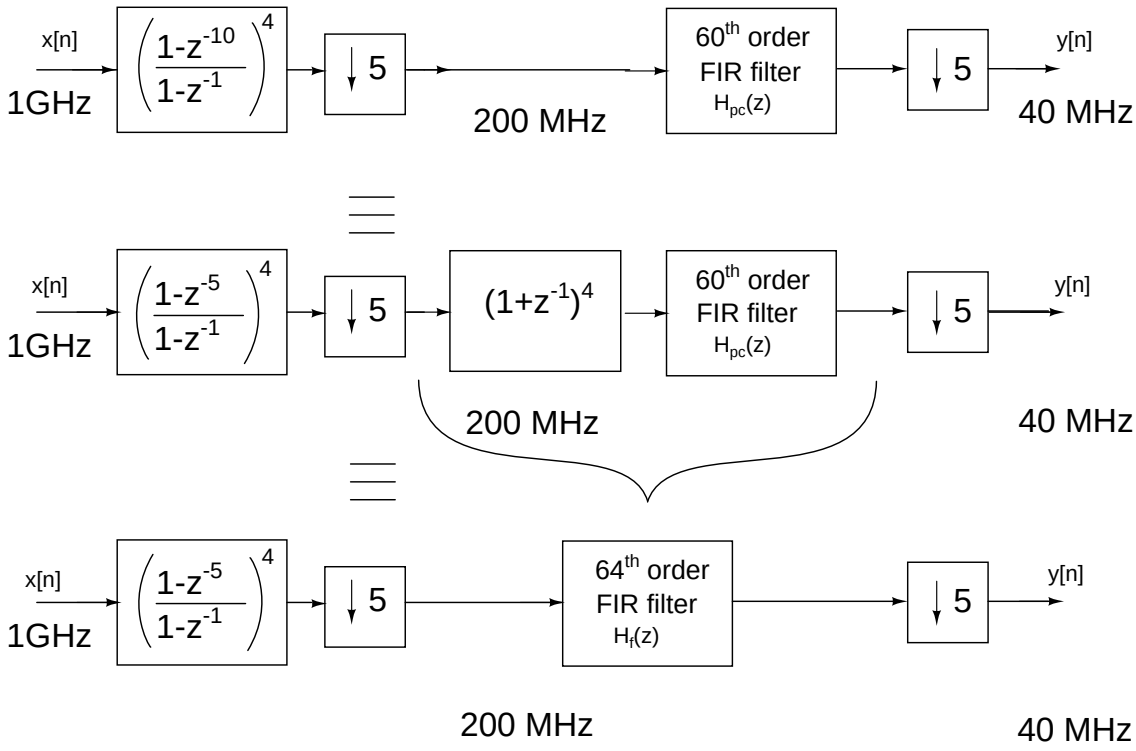


Figure 3.16: The factor $(1 + z^{-1})^4$ pushed to the low frequency side of the sinc filter is combined with the 60^{th} order FIR filter to get a 64^{th} order FIR filter [6]

3.6.4 Finite precision of the FIR filter coefficients

Since we are implementing the combined 64^{th} order FIR filter $H_f(z)$, the coefficients are obtained by performing the multiplication $H_{pc}(z) \times (1+z^{-1})^4$ shown in (3.10). The 64 coefficients in (3.11) given in Table 3.3 are then scaled such that $|hf_k| \leq 1$. i.e. the scaling factor $F_1 = 1/(\max(|hf_k|))$.

$$hf'_k = hf_k \times F_1 \tag{3.12}$$

$$= hf_k \times \frac{1}{max(|hf_k|)} \tag{3.13}$$

$$= hf_k \times \frac{1}{3.0035} \tag{3.14}$$

Even though MATLAB gives filter coefficients to 64-bit precision, fortunately it is not necessary to do 64-bit multiplications. A finite precision much lower than 64 bits is actually sufficient to meet the SNR requirement. To find out the required number of

bits to represent the FIR filter coefficients,

1. Scale the coefficients hf'_k again by a scaling factor $F_2 = 2^n$.
2. Round $(hf'_k \times 2^n)$ to the nearest integer.
3. Calculate SNR.

Repeat this process for $n = 1, 2, \ldots$. After a few iterations it is found that the SNR criterion of the decimation filter is met only if $n \geq 10$. Thus the value of n is chosen to be 10 or in other words, the FIR filter coefficients can be represented in at most 10 bits. Scaling of the filter coefficients is summarized below.

$$hf''_k = hf'_k \times F_2 \tag{3.15}$$

$$= hf_k \times \frac{2^{10}}{3.0035} \tag{3.16}$$

$$= hf_k \times 340.9366 \tag{3.17}$$

The new scaled coefficients are given in Table 3.3. Note that the division by max($|hf_k|$) while scaling makes sure that the largest FIR filter coefficient after scaling will be the largest possible number that can be represented in 10 bits (i.e. 1024). The FFT of the decimation filter output with 10-bit precision FIR filter coefficients is given in Figure 3.17. SNR calculated from this FFT for an input signal at 1367^{th} bin is 93 dB.

3.7 Decimation filter using polyphase decomposition

In section 3.2, the cascade equivalence property of decimation filters is shown. The property literally means that a filter $H(z^M)$ before a decimator with decimation factor M is equivalent to a filter $H(z)$ if it comes after the decimator. This property is significant due to the fact that a filter after decimator works at a lower frequency and thus has less

Table 3.3: Finite precision FIR filter coefficients. Only 32 filter coefficients are shown because of the symmetry $h_k = h_{63-k}$, $k = 0, 1, \ldots 63$, [6]

hf_k	Value of hf_k	hf_k''	(hf_k'')	Expansion	N
hf_0	0.003453128632	1.1773	1	1	1
hf_1	0.017459621385	5.9526	6	(2^2+2)	2
hf_2	0.039133310612	13.3419	13	(2^3+2^2+1)	3
hf_3	0.053315357077	18.1771	18	(2^4+2)	2
hf_4	0.049194153841	16.7720	17	(2^4+1)	2
hf_5	0.022826019450	7.7822	8	(2^3)	1
hf_6	-0.026726821644	-9.1121	-9	$-(2^3+1)$	2
hf_7	-0.091379259526	-31.1545	-31	$-(2^4+2^3+2^2+2+1)$	5
hf_8	-0.152588887407	-52.0231	-52	$-(2^5+2^4+2^2)$	3
hf_9	-0.187513355588	-63.9302	-64	$-(2^6)$	1
hf_{10}	-0.177614450723	-60.5553	-61	$-(2^5+2^4+2^3+2^2+1)$	5
hf_{11}	-0.117651625377	-40.1117	-40	$-(2^5+2^3)$	2
hf_{12}	-0.021138111557	-7.2068	-7	$-(2^2+2+1)$	3
hf_{13}	0.080997947545	27.6152	28	$(2^4+2^3+2^2)$	3
hf_{14}	0.149548722242	50.9866	51	(2^5+2^4+2+1)	4
hf_{15}	0.151726477066	51.7291	52	$(2^5+2^4+2^2)$	3
hf_{16}	0.076271160792	26.0036	26	(2^4+2^3+2)	3
hf_{17}	-0.057406812101	-19.5721	-20	$-(2^4+2^2)$	2
hf_{18}	-0.201111667924	-68.5663	-69	$-(2^6+2^2+1)$	3
hf_{19}	-0.292186843342	-99.6172	-100	$-(2^6+2^5+2^2)$	3
hf_{20}	-0.276829023603	-94.3811	-94	$-(2^6+2^4+2^3+2^2+2)$	5
hf_{21}	-0.134665635427	-45.9124	-46	$-(2^5+2^3+2^2+2)$	4
hf_{22}	0.104952878889	35.7823	36	(2^5+2^2)	2
hf_{23}	0.362737737201	123.6706	124	$(2^6+2^5+2^4+2^3+2^2)$	5
hf_{24}	0.528036189800	180.0269	180	$(2^7+2^5+2^4+2^2)$	4
hf_{25}	0.492153423628	167.7931	168	$(2^7+2^5+2^3)$	3
hf_{26}	0.186382819150	63.5447	64	(2^6)	1
hf_{27}	-0.388185540468	-132.3467	-132	$-(2^7+2^4)$	2
hf_{28}	-1.150402802168	-392.2144	-392	$-(2^8+2^7+2^3)$	3
hf_{29}	-1.953771295922	-666.1122	-666	$-(2^9+2^7+2^4+2^3+2)$	5
hf_{30}	-2.623248549904	-894.3615	-894	$-(2^9+2^8+2^6+2^5+2^4+2^3+2^2+2)$	8
hf_{31}	-3.003490924493	-1024.0000	-1024	-2^{10}	1
				Total =	96

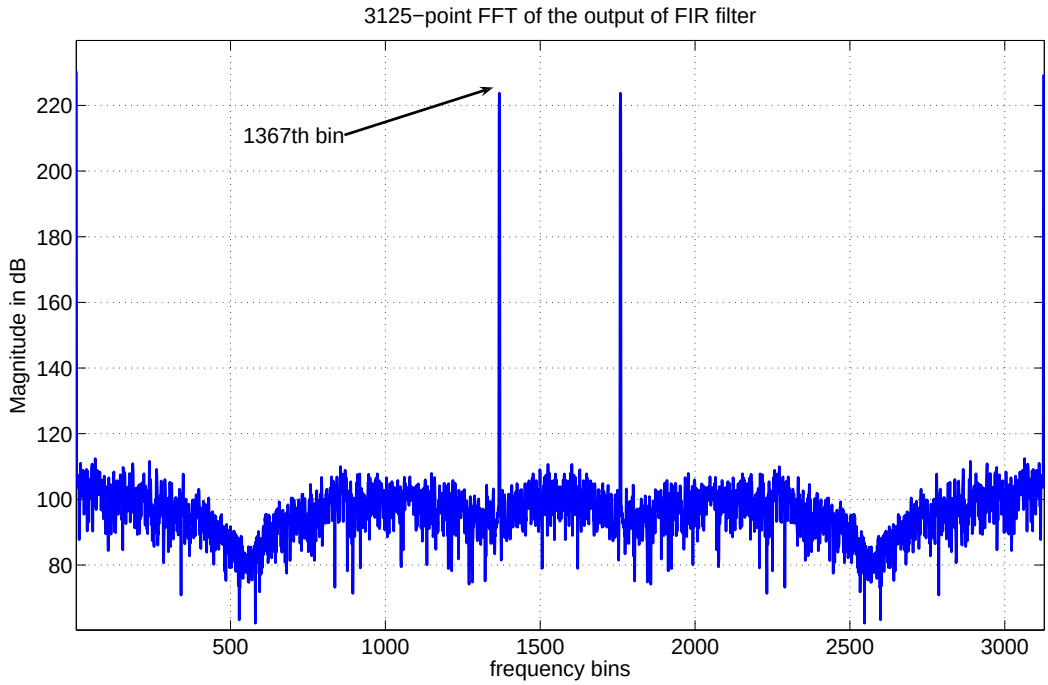


Figure 3.17: FFT of the output signal after FIR filter and decimator

stringent timing constraints and lower power consumption. A practical application of the cascade equivalence can be best seen in polyphase decomposition which will be discussed in this section.

3.7.1 Polyphase decomposition of sinc filter

The transfer function of the sinc filter with 10 taps obtained in (3.6) can be expanded as

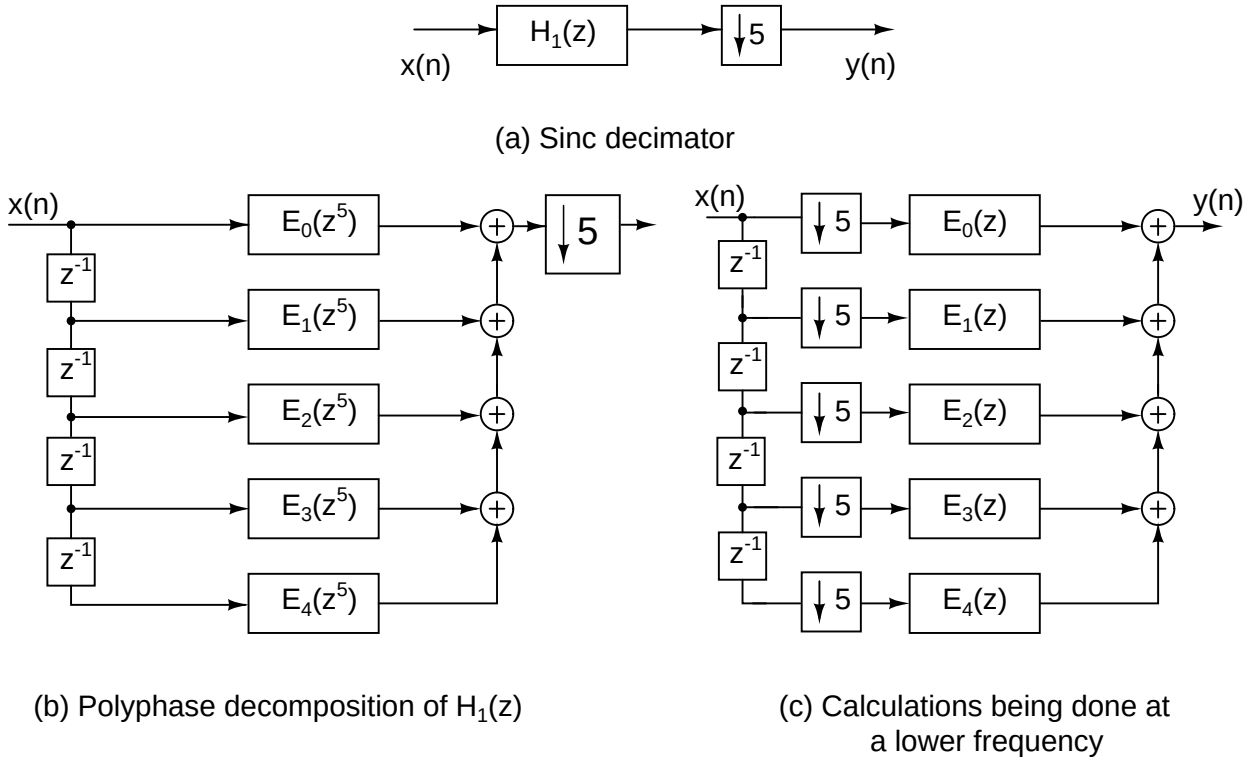
$$H_s(z) = \left(\frac{1-z^{10}}{1-z^{-1}}\right)^4 \tag{3.18}$$

$$= \left(\frac{(1-z^{-5})(1+z^{-5})}{1-z^{-1}}\right)^4 \tag{3.19}$$

$$= (1+z^{-1}+z^{-2}+z^{-3}+z^{-4})^4(1+z^{-5})^4 \tag{3.20}$$

$$= H_1(z)(1+z^{-5})^4 \tag{3.21}$$

As we have seen, the factor $(1 + z^{-5})^4$ can be pushed to the low frequency side of the decimator to give $(1 + z^{-1})^4$ (Figure 3.16) which is later be combined with the FIR filter in the second stage of the decimation filter. By moving even a part of the sinc filter computation to the low frequency side, we can considerably decrease power consumption. Now the transfer function to be implemented in the first stage (at 1 GHz) is only $H_1(z)$. This can be expanded as shown below.



(a) Sinc decimator

(b) Polyphase decomposition of $H_1(z)$

(c) Calculations being done at a lower frequency

Figure 3.18: Polyphase decomposition of sinc filter, [6]

$$H_1(z) = (1 + z^{-1} + z^{-2} + z^{-3} + z^{-4})^4 = \sum_{n=0}^{16} h(n) z^{-n} \tag{3.22}$$

$$= 1 + 4z^{-1} + 10z^{-2} + 20z^{-3} + 35z^{-4} + 52z^{-5} + 68z^{-6} + 80z^{-7} + 85z^{-8} + 80z^{-9}$$
$$+68z^{-10} + 52z^{-11} + 35z^{-12} + 20z^{-13} + 10z^{-14} + 4z^{-15} + z^{-16} \tag{3.23}$$

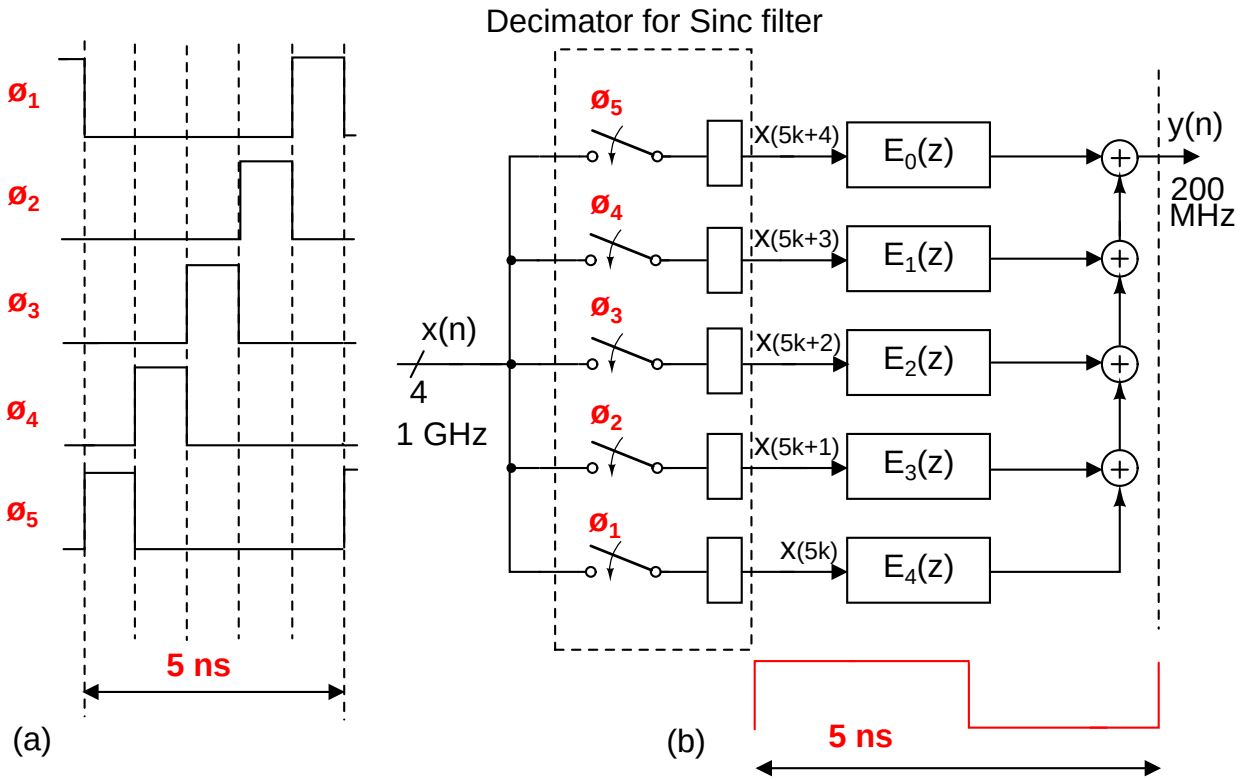


Figure 3.19: Polyphase decomposition of sinc filter. The filter is split into a decimator and five FIR filters, [6]

Which can be written as

$$\begin{aligned} H_1(z) = \quad & (1 + 52z^{-5} + 68z^{-10} + 4z^{-15}) \\ & + z^{-1}(4 + 68z^{-5} + 52z^{-10} + z^{-15}) \\ & + z^{-2}(10 + 80z^{-5} + 35z^{-10}) \\ & + z^{-3}(20 + 85z^{-5} + 20z^{-10}) \\ & + z^{-4}(35 + 80z^{-5} + 10z^{-10}) \end{aligned} \tag{3.24}$$

That is

$$H_1(z) = E_0(z^5) + z^{-1}E_1(z^5) + z^{-2}E_2(z^5) + z^{-3}E_4(z^5) + z^{-4}E_4(z^5) \tag{3.25}$$

where

$$E_0(z) = (1 + 52z^{-1} + 68z^{-2} + 4z^{-3}) \tag{3.26}$$

$$E_1(z) = (4 + 68z^{-1} + 52z^{-2} + z^{-3}) \tag{3.27}$$

$$E_2(z) = (10 + 80z^{-1} + 35z^{-2}) \tag{3.28}$$

$$E_3(z) = (20 + 85z^{-1} + 20z^{-2}) \tag{3.29}$$

$$E_4(z) = (35 + 80z^{-1} + 10z^{-2}) \tag{3.30}$$

The above decomposition of the sinc filter $H_1(z)$ into $E_0(z)$ to $E_4(z)$ is called polyphase decomposition (Figure 3.18 and Figure 3.19). Note that all the multiplications with filter coefficients now need to be carried out only at 200 MHz. This greatly relaxes the timing constraints and lowers the power consumption. The actual circuit implementation of this architecture is described in detail in [6].

3.7.2 Polyphase decomposition of FIR filter

The transfer function of the 64^{th} order FIR filter obtained in Table 3.3 with 10-bit precision coefficients can be written as

$$H_f(z) = \sum_{k=0}^{k=63} h_k z^{-k} \tag{3.31}$$

where h_k, k = 0,1..63 are the coefficients. Just like in the case of sinc filter, the FIR filter can also be split into five filters as shown in (3.32). Figure 3.20 and Figure 3.21 illustrate the decomposition. Note that this is quite similar to the decomposition done for sinc filter case.

$$H_f(z) = G_0(z^5) + z^{-1}G_1(z^5) + z^{-2}G_2(z^5) + z^{-3}G_4(z^5) + z^{-4}G_4(z^5) \tag{3.32}$$

The forms of $G_k(z)$ are given in Table 3.4. The polyphase decomposition of the FIR filter $H_f(z)$ into $G_0(z)$ to $G_4(z)$ facilitates computations to be carried out in a lower frequency (40 MHz) instead of 200 MHz (Figure 3.20, Figure 3.21). It is assumed that the five streams of 14-bit data $y(5k)$ to $y(5k+4)$ will appear at the positive edge of a 25 ns clock and will stay available throughout the period.

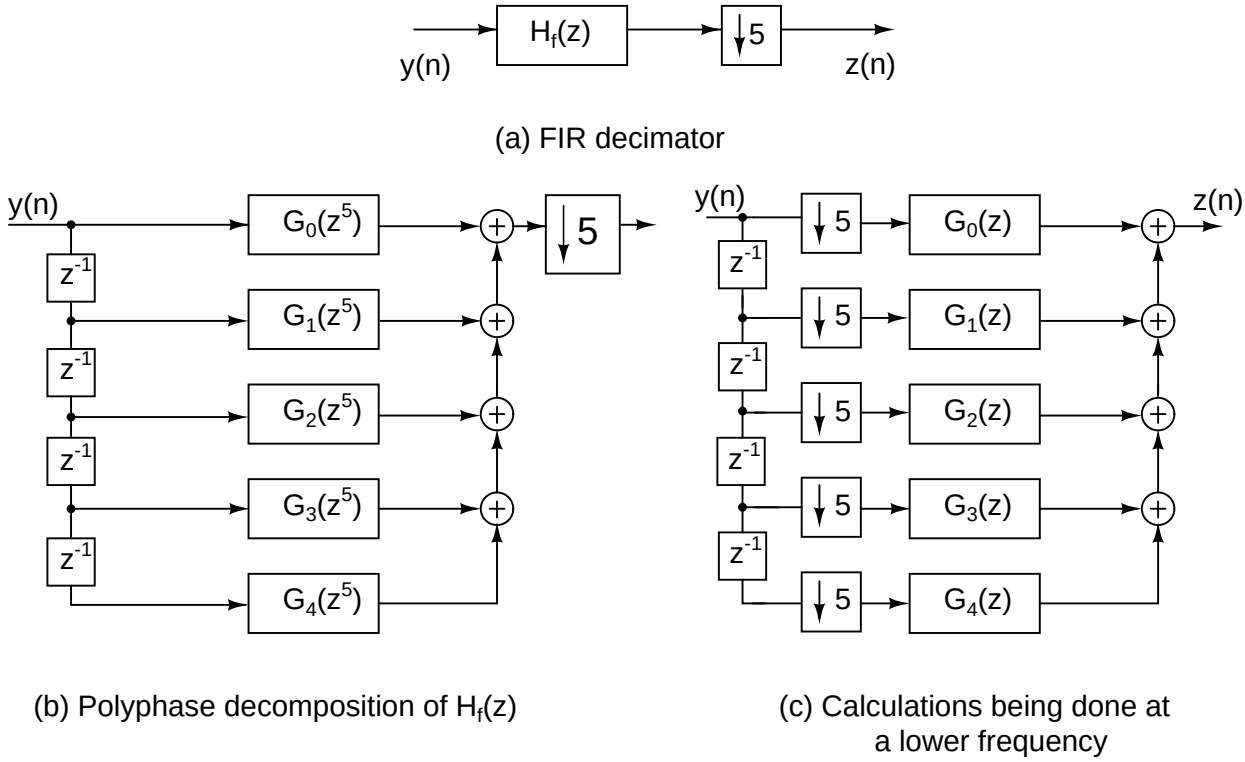


Figure 3.20: Polyphase decomposition of FIR the filter, [6]

3.8 The complete sinc filter

The complete sinc filter implementation is shown in Figure 3.22. We cannot hope to meet the timing constraint if we use conventional n-bit adders to add up the partial products. A suitable method to add up a large number of vectors is to use the Wallace tree structure[12]. The output from the wallace tree, $\overline{sum}$ and $\overline{carry}$ is added together using the vector merging Kogge-Stone adder. The Decimator block that splits the input

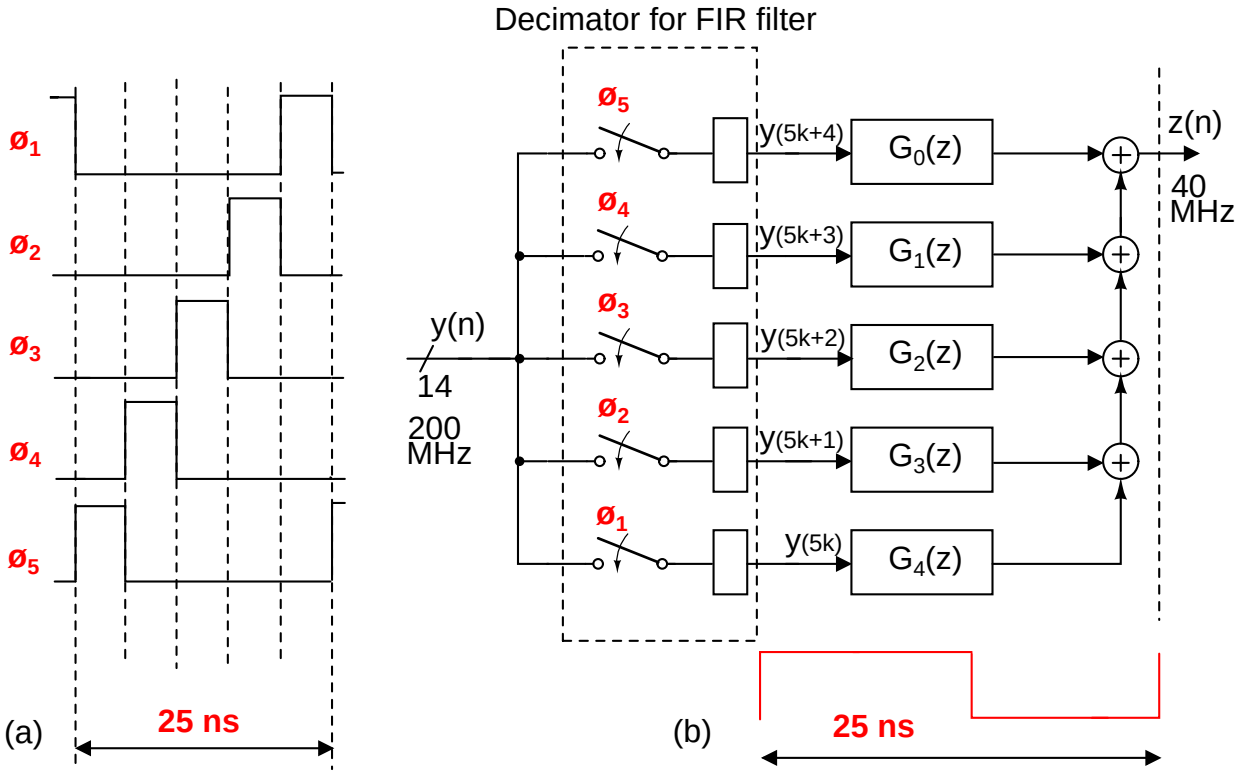


Figure 3.21: Polyphase decomposition of FIR filter, [6]

Table 3.4: $G_k(z)$ definitions

$G_k(z)$	Definition
$G_0(z)$	$h_0 + h_5z^{-1} + h_{10}z^{-2} + h_{15}z^{-3} + h_{20}z^{-4} + h_{25}z^{-5} + h_{30}z^{-6} + h_{28}z^{-7} + h_{23}z^{-8} + h_{18}z^{-9} + h_{13}z^{-10} + h_8z^{-11} + h_3z^{-12}$
$G_1(z)$	$h_1 + h_6z^{-1} + h_{11}z^{-2} + h_{16}z^{-3} + h_{21}z^{-4} + h_{26}z^{-5} + h_{31}z^{-6} + h_{27}z^{-7} + h_{22}z^{-8} + h_{17}z^{-9} + h_{12}z^{-10} + h_7z^{-11} + h_2z^{-12}$
$G_2(z)$	$h_2 + h_7z^{-1} + h_{12}z^{-2} + h_{17}z^{-3} + h_{22}z^{-4} + h_{27}z^{-5} + h_{31}z^{-6} + h_{26}z^{-7} + h_{21}z^{-8} + h_{16}z^{-9} + h_{11}z^{-10} + h_6z^{-11} + h_1z^{-12}$
$G_3(z)$	$h_3 + h_8z^{-1} + h_{13}z^{-2} + h_{18}z^{-3} + h_{23}z^{-4} + h_{28}z^{-5} + h_{30}z^{-6} + h_{25}z^{-7} + h_{20z^{-8}} + h_{15z^{-9}} + h_{10}z^{-10} + h_5z^{-11} + h_0z^{-12}$
$G_4(z)$	$h_4 + h_9z^{-1} + h_{14}z^{-2} + h_{19}z^{-3} + h_{24}z^{-4} + h_{29}z^{-5} + h_{29} + z^{-6}h_{24}z^{-7} + h_{19}z^{-8} + h_{14}z^{-9} + h_9z^{-10} + h_4z^{-11}$

data at 1 GHz into five streams is discussed in the later section.

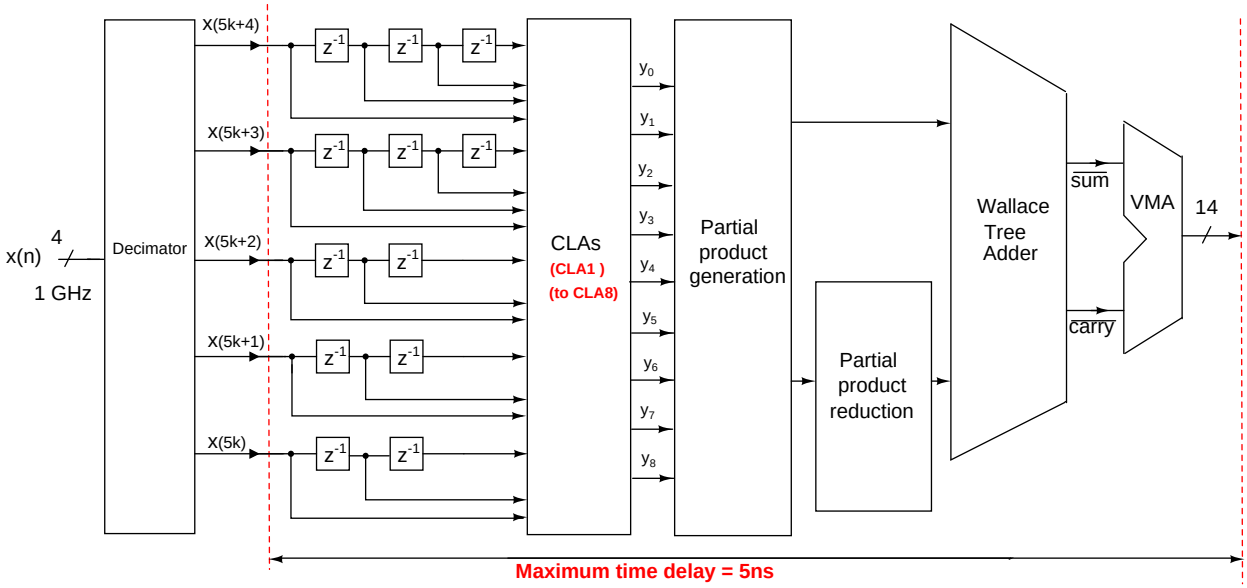


Figure 3.22: Complete sinc filter schematic. CLA: Carry Look ahead Adder, VMA: Vector Merging Adder, [6].

3.9 The complete FIR filter

The finite precision FIR filter coefficients are obtained in Table 3.3. The input to the FIR filter is the 14-bit output of the sinc filter. Unlike in the sinc filter, where all partial products are positive (because all filter coefficients of sinc filter are positive and the 4-bit input from the $\Delta\Sigma$ is DC offset and scaled to be positive), in the case of FIR filter, there are negative filter coefficients too. This makes the partial products negative even though the 14-bit input to the FIR filter is an unsigned positive integer. Table 3.3 also lists the expansion and the number of partial products(N) each coefficient will yield. To make matters simple, we follow the following strategy. Group all the positive partial products and negative partial products separately. Treat them as positive numbers and add up to get two sums. Then finally subtract the sum of negative partial products from that of positive partial products. Refer Figure 3.23.

Note that for the final subtraction, the sum of positive partial products has to be added to the 2's complement of the sum of negative partial products. This is done by inverting the sum and carry vector from the negative partial product wallace tree and also by giving a C_{in} value one.

3.10 Decimator design

Throughout our discussion, we have assumed that the five parallel streams of data $x(5k)$ to $x(5k+4)$ or $y(5k)$ to $y(5k+4)$ are available at the positive edge of a clock and remains available for one time period which is 5 ns and 25 ns respectively for the sinc and FIR filter. Here, we discuss this decimating process where a single input stream at a particular rate is split into five streams.

3.10.1 Basic decimator circuit

If we have a clock divider that gives one pulse of clk5 for every five pulses of clk1, then the basic structure of a decimator is quite simple as shown in Figure 3.24. For the case of the sinc filter, five phase shifted clocks are used to choose the five consecutive inputs. At the end of five cycles, a common clock registers the five streams together. Detailed timing diagram is shown in Figure 3.24(b). It can be seen that a particular input is ready at least 0.5 ns (with the negative edge of the master clock, clk1) before the corresponding phase shifted clock registers it with a positive clock edge.

Same argument holds for the second stage decimator before the FIR filter.

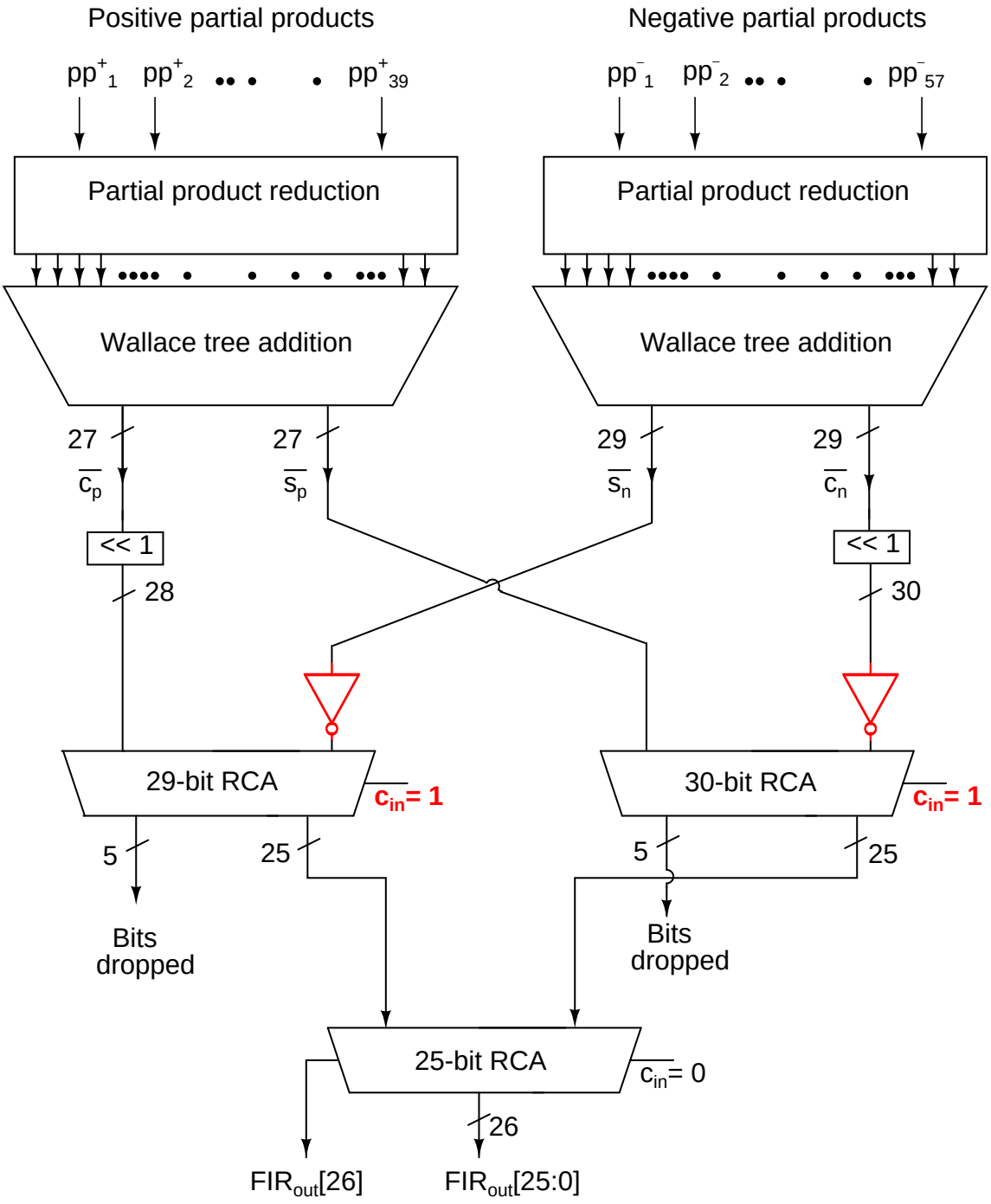


Figure 3.23: Complete FIR filter schematic, [6]

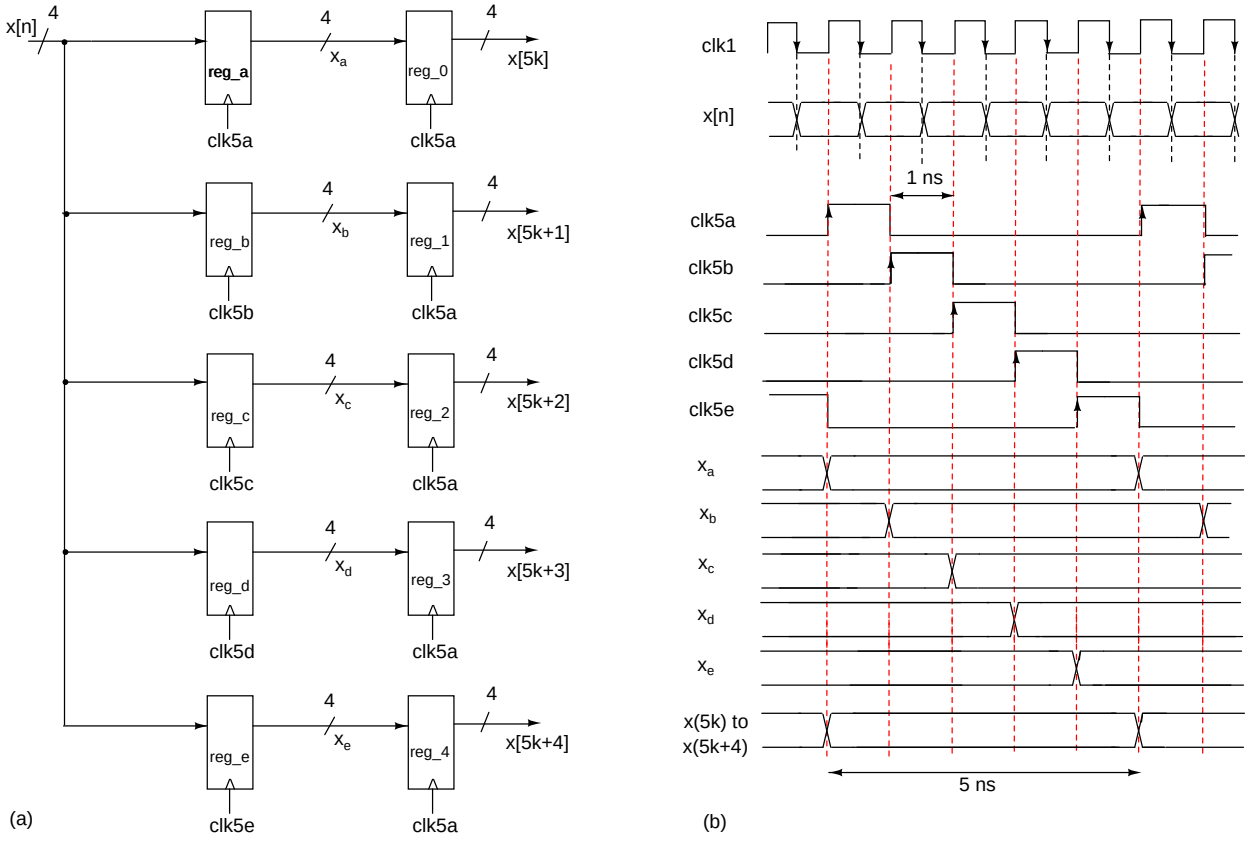


Figure 3.24: (a)Low power decimator structure (b)Timing diagram of the power efficient decimator. clk1 has time period 1 ns and clk5a to clk5e are five phase shifted clocks with time period 5 ns. For the FIR filter decimator, the timing diagram is same except that clk1 is at 200 MHz and clk5a to clk5e are at 40 MHz, [6].

3.11 Clock dividers

To divide a clock by 5, the simplest technique is to use a counting state machine that counts from 0 to 4 and then repeats. This creates five states each of which gives rise to the five phase shifted clocks required for the decimator. This is a simple method and requires only three registers working at 1 GHz.

CHAPTER 4

Synthesis and Simulation Results

4.1 Synthesis of the Decimation filter

The design and implementation details of the decimation filter in two stages using polyphase decomposition are discussed in the previous chapter. The preliminary design involves testing the decimator in MATLAB to ensure that the SNR performance is satisfactory. The actual circuit is modelled using the Hardware Description Language, Verilog and functional verification is done in Modelsim by writing appropriate test benches. Once this ideal circuit gives the same SNR obtained in Matlab for the same input stream, we can go ahead to the final steps of synthesizing and routing the circuit.

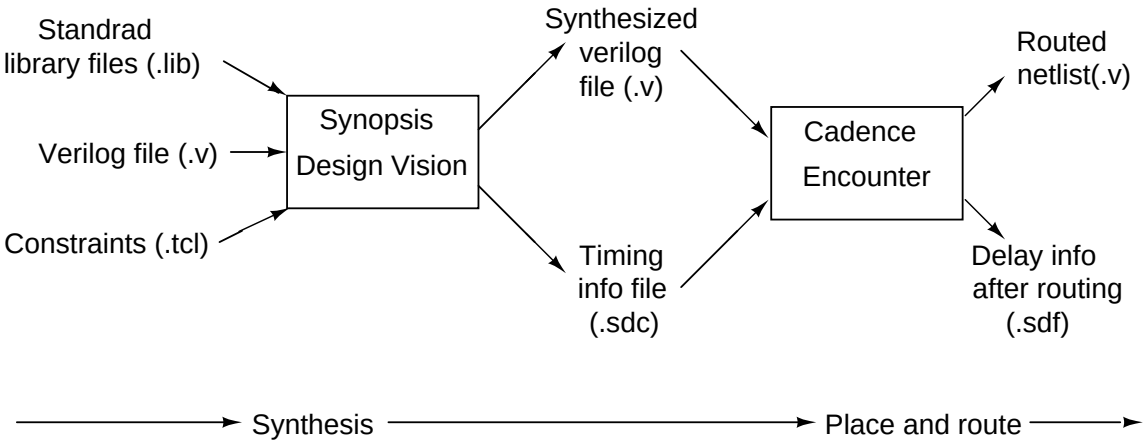


Figure 4.1: Block diagram of the design flow

The digital circuitry behaviorally modelled in verilog is to be mapped into actual circuit components based on standard cell libraries and constraints. This procedure

Table 4.1: Power report of the decimation filter(10 tap sinc filter + 60^{th} order FIR filter) obtained using Synopsis design vision(Typical case)

Module	Clock	Power	% Power (mW)
Sinc filter	200 MHz (5 ns)	3.537	20.7
FIR filter	40 MHz (25 ns)	9.566	56.1
Decimator for sinc	1 GHz (1 ns)	1.044	6.1
Decimator for FIR	200 MHz (5 ns)	0.723	4.2
CLK dividers	1 GHz (1 ns)	1.365	8
Total		17.06	100%

is called synthesis and is performed using the Synopsis synthesizing tool *Design Vision*. Design vision gives the synthesized verilog file as output by reading three inputs (Figure 4.1)

1. Verilog code for the circuit
2. Standard library files
3. Constraints

The verilog code is a behavioral description of the circuit. The standard cell libraries from UMC provide standard building blocks of the circuit in CMOS 180 nm technology. The constraints are given to the design vision using a .tcl script file. There are mainly three constraints for the decimation filter circuit.

1. Meet all the timing conditions i.e. the delay should be less than 5 ns and 25 ns for the first and second stage of decimation filter respectively.
2. Minimize the area of the circuit
3. Minimize the power consumption of the circuit

Once the three input files (verilog, libraries and constraints) are ready, design vision compiles the input and creates a synthesized verilog file which is now based on standard building blocks. Design vision also gives a timing information file (.sdc). These files can be read using *Cadence encounter* to place and route (Figure 4.1). In Place and Route the CAD tool performs three major tasks

Table 4.2: Decimation filter design summary

Technology	$0.18\,\mu$m CMOS
Supply Voltage	1.8 V
Total power	17.06 mW
Area	790×740
Passband droop	-2.4 dB at 20 MHz
Simulated SNR	93.1543 dB

- Place the layout of standard cells according to the floor plan
- Clock tree synthesis (CTS)
- Route the design to meet the timing

After placement and routing, the routed netlist can be saved as a verilog file and the timing information after routing can be extracted into a .sdf file both of which can be used for the post-route simulations.

4.2 Simulation Results

The final simulation results are summarized in Table 4.1 and 4.2. From Table 4.1, it can be seen that most of the power is dissipated in the second stage FIR filter. After simulations, it is found that the FIR filter has a huge positive slack. This means that the power consumption of the FIR filter alone can be reduced much further by decreasing the power supply voltage for FIR filter block alone. The total power consumed by the entire decimation filter is 17.06 mW. Thus polyphase implementation is proved to be very useful in high speed low power decimation filters. The low power of the decimation filter is attributed mainly to the decrease in the operating frequency of the circuit in polyphase. The price paid for the low power are the higher area in polyphase compared to the CIC implementation of sinc and the increased design complexity in both the sinc and FIR filters.

Amoeba view of the final layout is given in Figure 4.2.

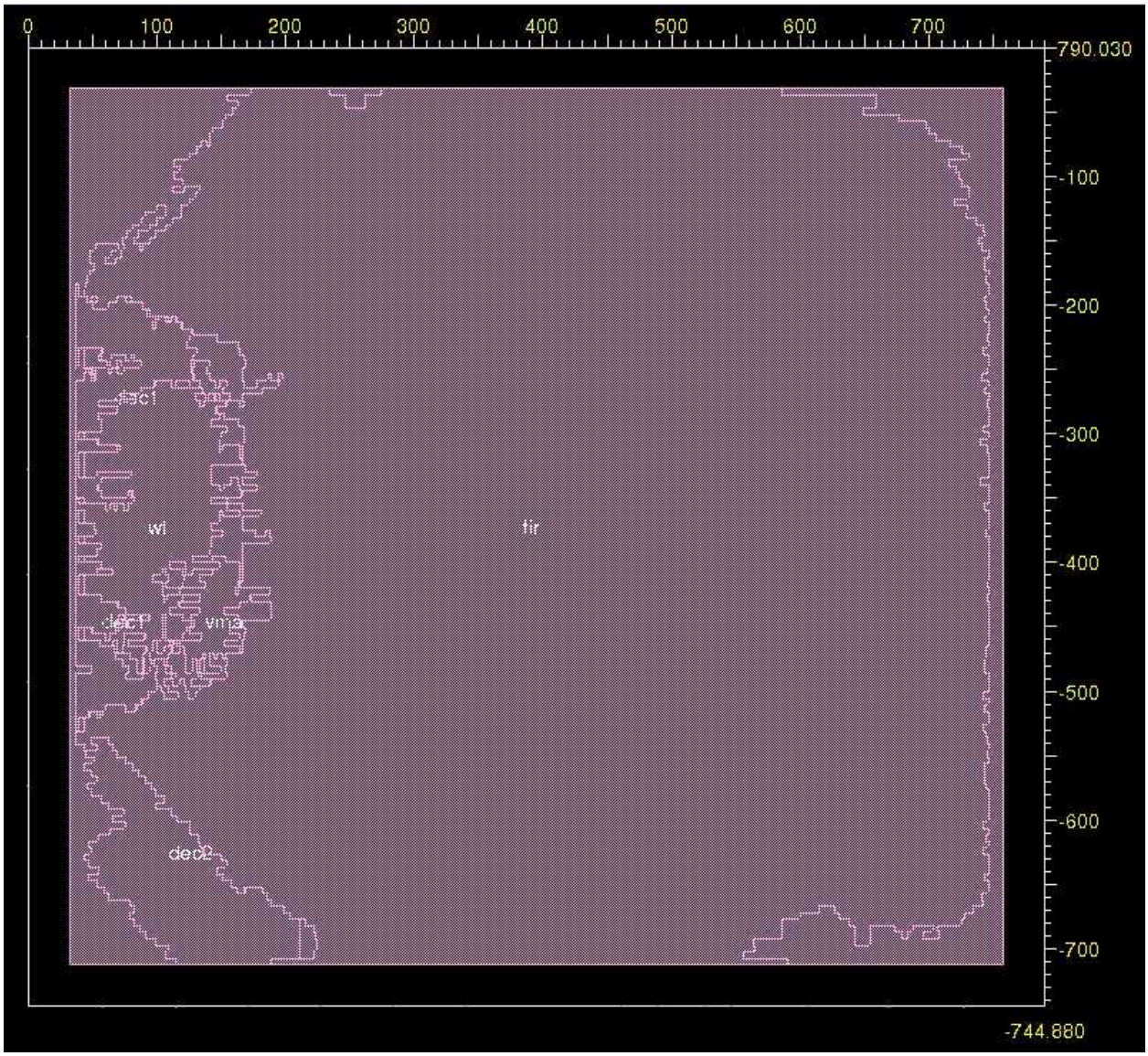


Figure 4.2: Amoeba view of the circuit after layout in encounter

CHAPTER 5

$\Delta\Sigma$ ADC Architecture and Results

5.1 Architecture

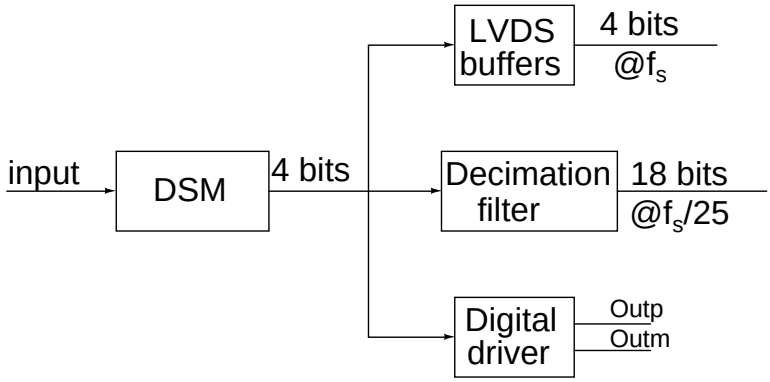


Figure 5.1: Block diagram of ADC

Figure 5.1 shows the complete top level block diagram of the $\Delta\Sigma$ analog to digital converter after combining the decimation filter that explained in the earlier chapter to the $\Delta\Sigma$ modulator that has been tested as a part of this thesis work. Description of the $\Delta\Sigma$ modulator is given in Appendix A. Out of 26 bits of decimation filter output the most significant 18 bits are taken as the output of the analog to digital converter. Along with the decimator output of the ADC, we have taken the output of the delta sigma modulator, so that the functionality and performance of the DSM can be measured. Besides we have a provision to collect the output differential signal through a digital driver circuit, which can be viewed using an oscilloscope. The digital driver circuit is explained in the next section. The chip has 76 pins and the pin details of the ADC is shown in the Figure 5.2. Appendix C shows the description of each pin.

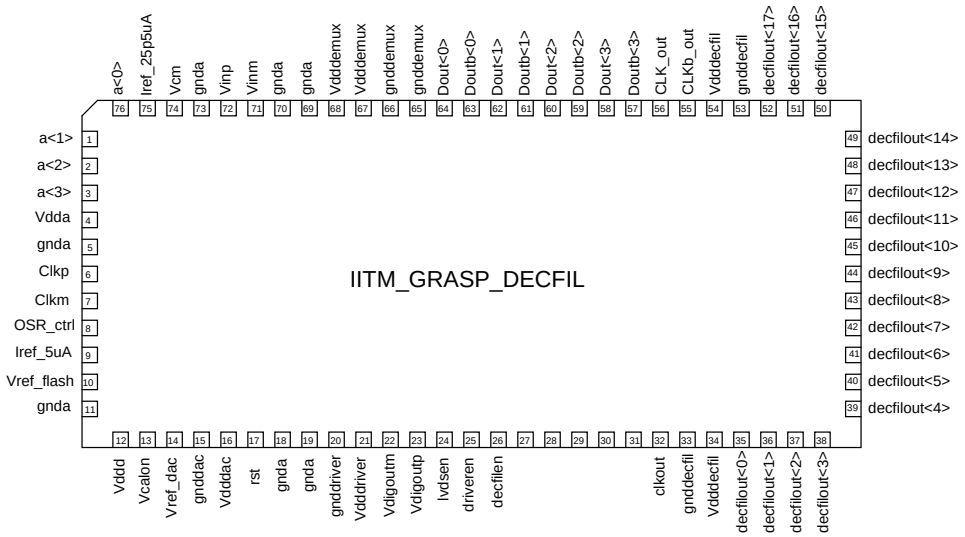


Figure 5.2: Pin details of the ADC

5.2 Digital driver

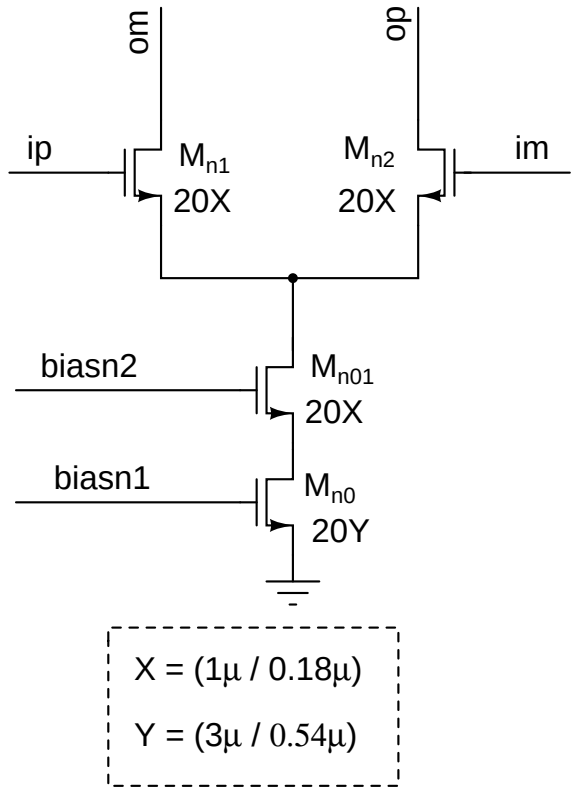


Figure 5.3: Schematic of unit DAC cell

A unit cell differential pair current steering DAC cell is as shown in Figure 5.3, which consists of 600 μA cascoded current source, steered using the NMOS differential

pair of current switches. The current value is chosen to drive proper current to the load. Since we are using rail to rail drives, the switch transistors may go into triode region, there by losing the advantage of cascode transistor for the tail current source. The switch Transistor sizes are chosen larger to make the overdrive voltage smaller so that, the switches act in a better way. Because of the larger sizes of the transistors the tail node capacitance increases, which increases the current in the capacitor due to variations in the tail node voltage. So the cascode transistor is chosen to be of smaller size, which reduces the parasitic capacitance. The cascode transistor protects the tail current from any voltage variation and also reduces modulation of current source by the output voltage. The bias voltages are generated using a low voltage cascode current mirror as shown in Figure 5.4.

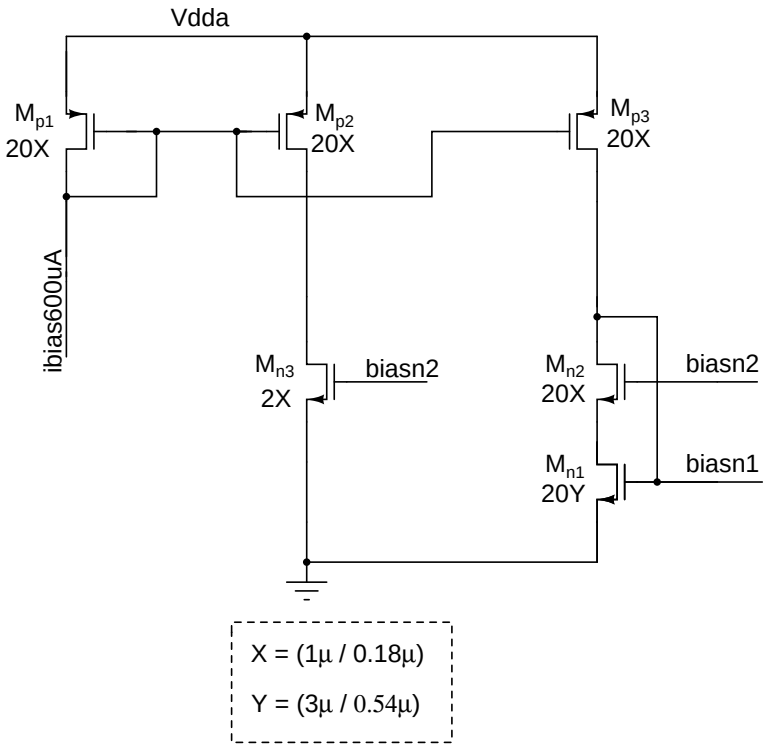


Figure 5.4: Bias generation circuit for DAC cells

CHAPTER 6

Conclusion

6.1 Testing

The high speed CT $\Delta\Sigma$ modulator was tested to work at 750 MHz and the tested chip had a dynamic range of 82 dB and SNR of 68.7 dB for a bandwidth of 15 MHz and a dynamic range of 80 dB and SNR of 67.3 dB for a bandwidth of 37.5MHz. The modulator was becoming unstable at 800 MHz. During the course of testing it was also observed that the chips packaged by EUROPRACTICE were also not running at 800 MHz.

6.2 Decimation filter

The total power consumed by the entire decimation filter is 17.06 mW. The continuous time $\Delta\Sigma$ modulator for which the decimation filter is designed consumes about 50 mW power. Low power decimation filter described in this thesis can be used to turn the continuous time $\Delta\Sigma$ modulator into a complete Analog to Digital Converter system with only 38% additional power. The simulated SNR of the Decimator output is 93.1543 dB

APPENDIX A

Excess loop delay compensation and the chip architecture

The maximum frequency of operation in a CT $\Delta\Sigma$ modulator is limited by the delay in the quantizer and the feedback DAC. The delay in the quantizer is caused because it takes a finite amount of time for the comparator in the Flash ADC to differentiate the inputs, this is known as regeneration time which acts like a loop delay. Any extra errors due to non-linearity of the quantizer is taken care by the loops filter at it is shaped out of the signal band (these errors can be considered like an extra error to the quantization error and hence get shaped out). The DAC circuit converts the digitized output of the quantizer to an analog signal which is then fedback to the input. But mismatch between the DAC elements causes an error which appears directly at the input [8]. Since the loop gain of the signal in the signal bandwidth is unity hence the error due to DAC also appears at the output. To prevent this different techniques are used for example DWA(data weighted averaging). Introducing these extra circuitry reduces the DAC error at the expense of extra delay(the DAC clock has to be delayed). All these factors mainly contribute to the ELD(excess loop delay).

Due to excess loop delay present in the loop the DAC will have to be clocked at a time greater than the delay($t > \tau_d$). But when the DAC pulse is delayed by some time say τ_d then the equivalent DT loop filter order increases by one [3]. This seriously affects the noise shaping and increases the inband noise and also reduces the maximum stable amplitude. If this delay becomes too large then the modulator will become unstable. Figure A.1 shows the effect of ELD on noise shaped idle channel output of the modulator. The next section will discuss about common ELD compensation tech-

niques. After that we will discuss about the ELD technique used in the chip which is being tested as part of this thesis work.

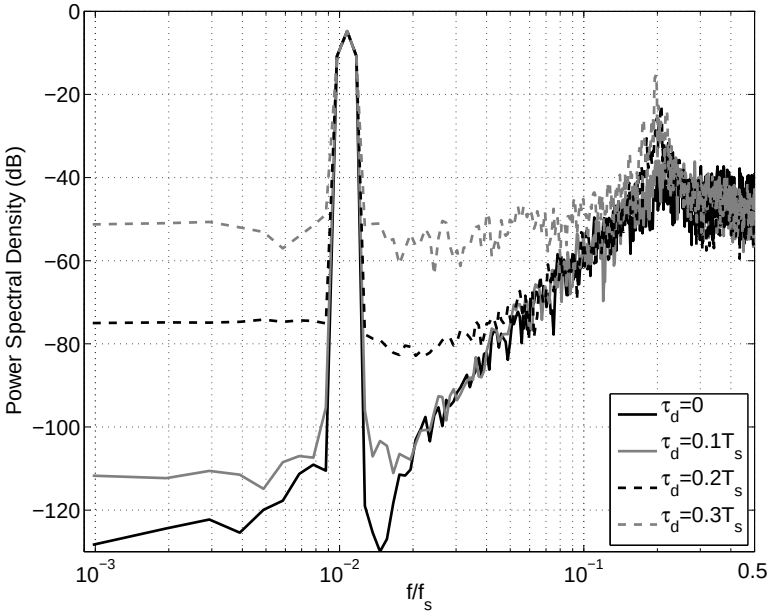


Figure A.1: Effect of ELD on Noise Shaping, [10]

A.1 Commonly used compensation techniques

One of the techniques used for compensating ELD is by introducing one clock cycle delay in the feedback path and then using another DAC to provide the missing sample. Due to introduction of one clock cycle delay the second sample of the impulse response of the main CT loop is zero. The extra new compensation path(with less delay) introduced compensates for this sample without affecting the other samples produced by the main DAC [13]. Hence the samples of the impulse response of the overall feedback loop becomes equivalent to that of the DT modulator hence compensating the ELD. The schematic in the Figure A.2 shows this.

Another more commonly used technique is to directly add the scaled DAC output to the output of the loop filter. The scaling constant depends on the delay of the feedback

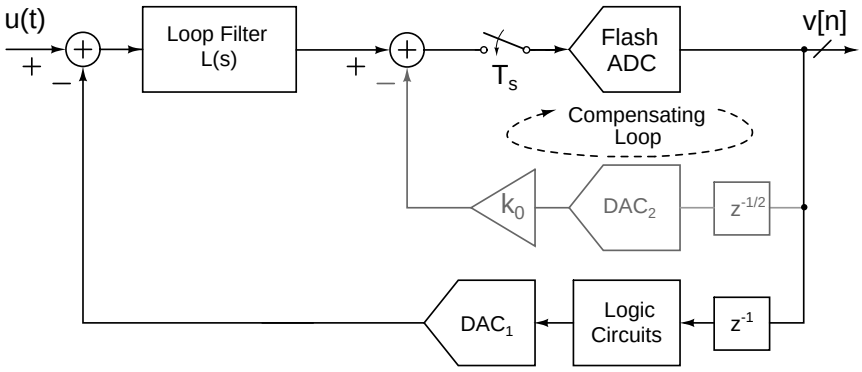


Figure A.2: ELD compensation using an extra DAC, [10]

path. This technique is also shown in Figure A.3

The scaling constant is realized by adding a differentiator after the output of the 1^{st} loop filter. This is done by passing the output from the first loop filter integrator to the virtual node of the summing opamp using a capacitor.

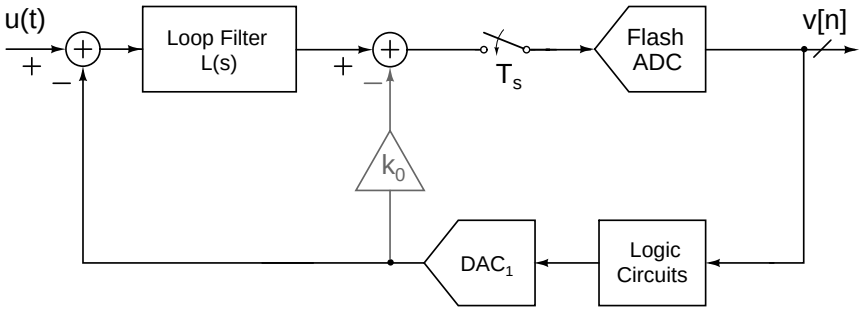


Figure A.3: ELD compensation using a differentiating path in the loop filter, [10]

A.1.1 Limitations of commonly used ELD techniques

The techniques proposed above have a lot of limitations. The most important being the excess loop delay has to be less than one clock cycle. This limits the maximum sampling speed of the modulator. Also the technique in which we introduced an extra DAC, the DAC loads the output of the quantizer and takes a larger area.

Let us now consider the technique in which we used a direct path to add a sample directly from the output of the DAC. As the loop delay increases the contribution from

this direct path will keep increasing such that the overall NTF looks exactly similar to the ideal case without any delay [7]. For large delays it becomes very difficult to realize large gains keeping the delay low. Also as soon the delay in the feedback path becomes greater than one clock cycle then the second sample of the impulse response of the feedback path will be zero. Multiplying it with a constant will not be able to make the second sample non zero. This increases the order of the NTF and hence degrades the noise attenuation in the signal band and causes more peaking compared to ideal case.

The only way to compensate this excess delay of one clock cycle is to introduce an extra feedback path which is independent of this delay.

A.2 Compensating for more than unity cycle excess loop delay

The technique used in the chip [10] which is being tested, employs a local feedback circuit with lesser delay in order to compensate for the ELD of greater than unity clock cycle [11]. Figure A.4 shows a additional feedback loop to compensate for the missing second sample of the original loop response. The delay of this additional loop is significantly less (lesser than unity clock cycle) as it uses only a single sample and hold circuit. To make the response of the modulator similar to the ideal case we first find out the sampled response of the loop. To do this we break the loop and apply a impulse at the input of the quantizer. The samples of the output are then compared with the ideal response. The second sample is adjusted by the help of the scaling coefficient in the local feedback path. The remaining samples are adjusted by changing the loop filter coefficients such that the remaining response is equivalent to that of the ideal case. We now calculate the new NTF due to the introduction of this loop. We break the loop at point X at the start of the quantizer and find the transfer function from the error introduced in the quantizer to the output [10]. It is also easily seen from the figure that the

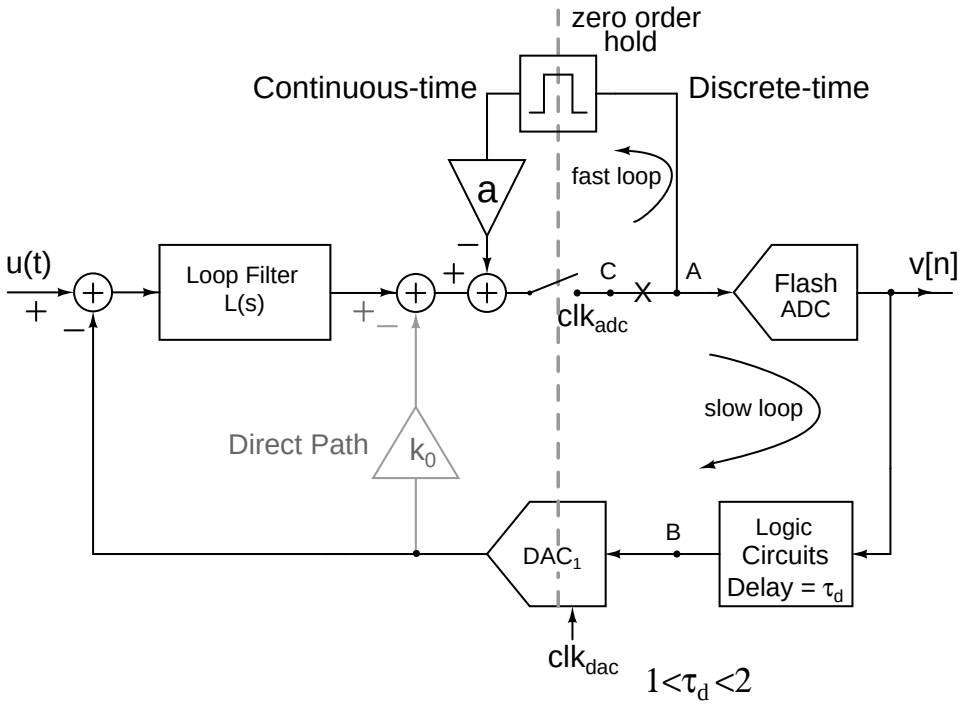


Figure A.4: ELD compensation using an extra fast feedback path, [10]

transfer function of the fast extra feedback loop is $1/(1+az^{-1})$. We can model the flash ADC by adding a quantization error e_q. Hence the overall transfer function looks like

$$e_q(z) - V(z)[L_d(z)z^{-2} + k_0 z^{-2}](\frac{1}{1+az^{-1}}) = V(z) \tag{A.1}$$

$$\frac{V(Z)}{e_q(z)} = \frac{1+az^{-1}}{1+az^{-1}+(k_0+L_d(z))z^{-2}} \tag{A.2}$$

Since the coefficient of the loop filter were adjusted to match the samples hence the denominator remains the same as the ideal case. We now see an extra zero due to the compensation loop. Hence the new NTF in terms of ideal NTF is written as

$$NTF_{new} = (1 + az^{-1})NTF_{ideal}(z) \tag{A.3}$$

The NTF of the compensated loop for a fourth order $\Delta\Sigma$ compared to the ideal case when the ELD is 1.5 looks as shown in Figure A.5. As we can see from the figure for $OSR = 25$ there is an 7.5 dB increase in the quantization noise. Also by simulation it was seen that for higher OBG the performance becomes similar to that of the ideal case with a order less [10]. Hence to get a performance of a ideal 3rd order loop filter using this compensation technique we will have to design it for 4th order.

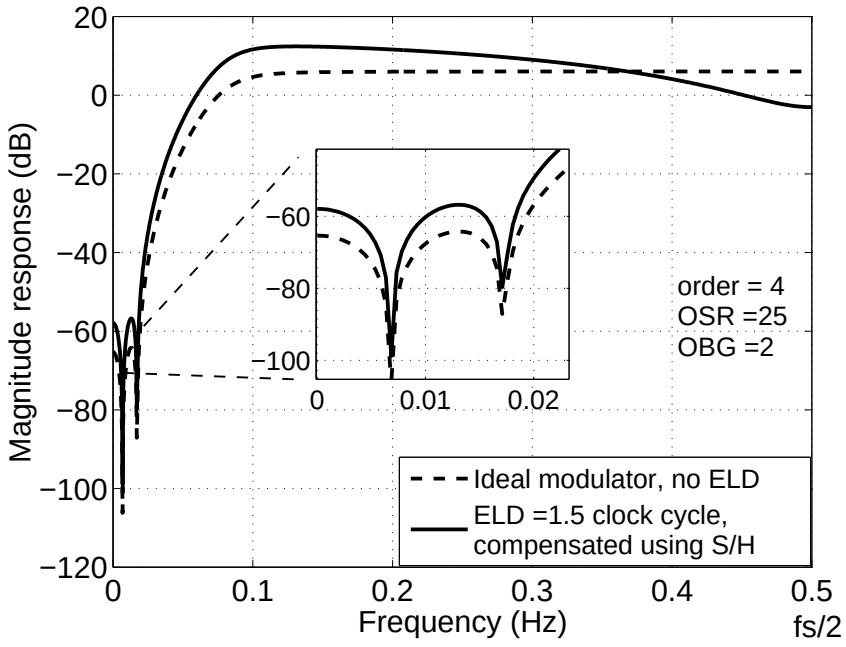


Figure A.5: Effect of the extra zero on the NTF, [10]

A.3 Architecture of the $\Delta\Sigma$ modulator

A.3.1 Loop filter

This section will briefly describe the overall architecture of the chip. More details regarding the chip can be found in [10]. The chip which was tested as part of this thesis work was designed for an OBG of 2 and an OSR of 25. A forth order loop filter with optimized zeros was used. The over all modulator looks as shown in Figure A.6

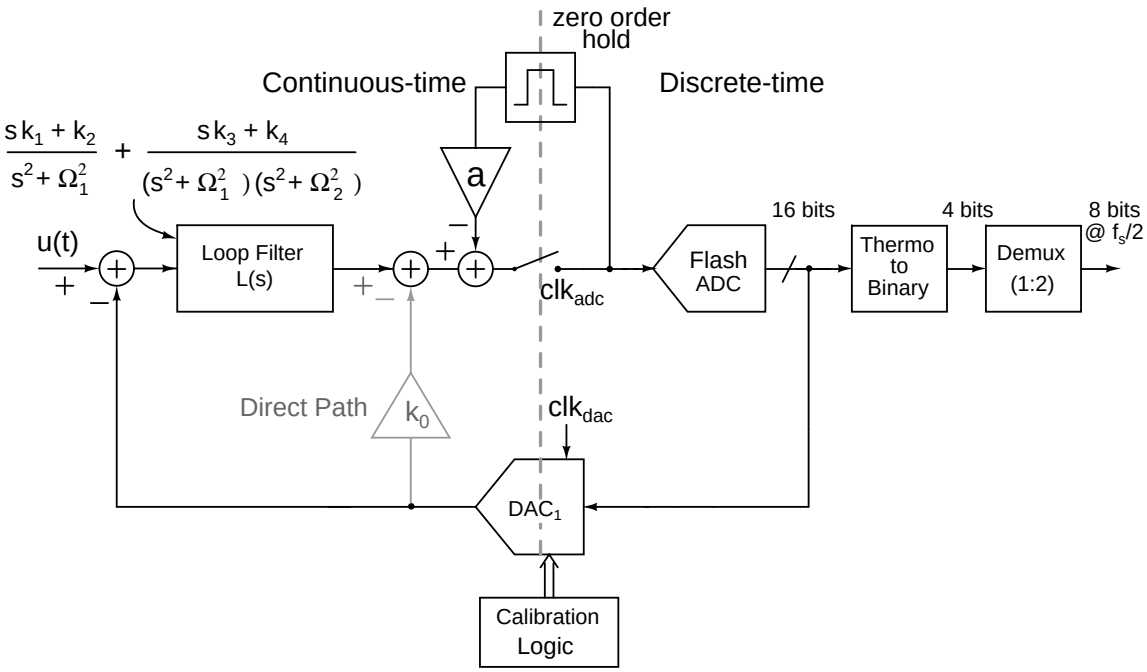


Figure A.6: Block diagram of the ADC, [10]

The loop filter chosen was a cascade of 4 integrators with feedforward summation(CIFF) [10]. To realize the integrators of the loop filter and a Active RC integrator was used. Active RC integrators were chosen over G_mC because of their higher linearity. The NTF for the modulator can be derived from the ideal case. Using the Schreier $\Delta\Sigma$ toolbox [9] in Matlab we know that the NTF of an ideal system for OSR of 25 and

OBG of 2 is

$$NTF_{ideal} = \frac{(z^2 - 1.98z + 1)(z^2 - 1.98z + 1)}{(z^2 - 1.204z + 0.3771)(z^2 - 1.43z + 0.6585)} \tag{A.4}$$

$$NTF_{new} = (1 + az^{-1})NTF_{ideal}(z) \tag{A.5}$$

The first opamp for the loop filter was designed using a feedforward compensated architecture as it has high bandwidth and consumes low power(at the expense of signal swing). The other opamps(2nd,3rd and 4th) were designed using as simple 2 stage miller compensated opamps as these opamps are less crucial compared to the 1^{st} opamp.

A.3.2 Fast feedback ELD compensating path

To realize the fast feedback path we need two simple components one of them is the sample and hold part and the other is the constant multiplication part. The sample and hold is realized using a 2 stage track and hold circuit. That is while one stage is tracking the other holds and vice versa. The switches used on the track and hold circuit is bootstrap switch to reduce the distortion. For constant scaling part we use a G_m cell which is a simple voltage controlled current source. For a scaling constant of a_0 the G_m of the cell is kept at a_0/R where R is the feedback resistance(from the output to virtual ground) of the summing opamp used after the loop filter.

A.3.3 Flash ADC

The 4 bit quantizer is realized using a flash ADC. Flash ADC architecture offers high speed at the expense of more power and area; since speed is critical here so a flash ADC

was used. In this architecture the input is compared will all the possible quantization levels at the same instant. So for a 4bit ADC we will need $2^4 - 1 = 15$ comparators to compare the input with the quantization levels. The output of these comparators will be an array of 1's and 0's which is called a thermometer code. This then has to be converted to binary using a separate circuitry outside the modulator, the DAC uses the thermometer code itself. The reference levels are generated using resistive ladders and a differential version of the circuit is implemented.

A.3.4 4 bit DAC

The DAC converts the digital outputs of the quantizer to analog signals which can be compared with the input signal in a CT modulator. The DAC topology used in [10] is a complementary current steering DAC which feeds the current to the loop filter. This architecture is used because the total noise PSD is 2 times that of current steering sources. Whereas in the more commonly used DAC architecture using NMOS current steering and PMOS current source the total noise PSD is 4 times that of the current source noise.

We know that mismatch in the DAC cells can seriously affect the performance of the modulator as any kind of error introduced due to the DAC comes directly at the output. To reduce mismatch various techniques like dynamic element matching or data weighted averaging can be used so that the mismatch between the DAC cells on an average is minimized [8]. In the current modulator the DAC uses a *calibration scheme* to reduce the mismatch. The basic idea is to charge the gate capacitance of the NMOS to a value such that it draws the reference current when it is switched on. So in one cycle we connect the reference current source to the M1 and let the capacitor charge to the reference value and when we want to use it we disconnect the reference current source and connect the NMOS as shown in Figure A.7

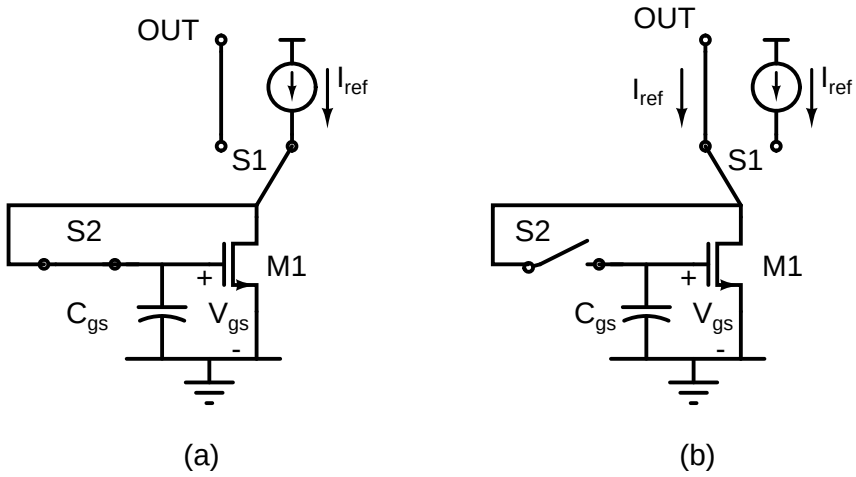


Figure A.7: Calibration of a DAC cell (a)Calibration Phase (b)Output phase, [10]

A similar technique is used for this chip in which each DAC cell has two reference currents sources and two reference current transistors. This is because when one of the current sources is being used by the DAC the other gets calibrated. Once the calibration of the other cell is complete the two sources switch their functionality and now the other source enters the calibration phase . Also since we have say N DAC cells not all the N transistors (which are not connected to the DAC output) in these cells are being calibrated, only one transistor is calibrated and then the others in the other cells are calibrated in a sequence(to generated this sequence we require some extra switching circuitry). So among the N DAC cells each of the transistors connected to the calibration circuit is calibrated in a sequence once by one and inside each DAC cell as soon as once of the current source gets calibrated it gets connected to DAC output and the other transistor connects to the calibration circuit as it will then be calibrated in the next cycle after the calibration of the other DAC cells are complete.

APPENDIX B

Pin details of the $\Delta\Sigma$ modulator chip

Table B.1: Functionality of each pin, [10]

Pin	Name	Functionality
1,2,3,44	$a < 1 >$, a$< 2 >$, a$< 3 >$, a$< 4 >$	Capacitor tuning bits
4	Vdda	1.8 V supply voltage for the loop filter and biasing
5, 11, 15, 36, 38, 41	gnda	Ground
6, 7	Clkp, Clkm	Differential input clocks
8	OSR_ctrl	Control to choose between OSR =25 (low), OSR =10 (high)
9	Iref_5uA	Reference current source $5\mu A$(sinking current)
10	Vref_flash	Control for changing input swing of flash ADC. Nominal value =0.9 V
12	Vddd	1.8 V supply voltage for the flash ADC
13	Vcalon	Control for making calibration on (high) and off (low).
14	Vref_dac	Control for DAC unit cell current. Nominal value =0.9 V
16	Vdddac	1.8 V supply voltage for the DAC
17	rst	Reset the digital calibration logic (Reset when high)
18-25 and 28-35	Doutb $< 7 >$ to Dout $< 0 >$	Deserialized differential output LVDS data stream
26, 27	Clk_out, Clkb_out	Differential output LVDS clocks
37	Vdd_demux	1.8 V supply voltage for the Demux and IO pads
39, 40	Vinm, Vinp	Differential input signals
42	Vcm	Common mode reference voltage of 0.9 V
43	Iref_25p25 μA	Reference current source of value $25.5\mu A$(source current)

APPENDIX C

Pin details of the $\Delta\Sigma$ modulator with Decimation filter

Table C.1: Functionality of each pin.

Pin	**Name**	**Functionality**
1,2,3,76	$a < 1 >$, a$< 2 >$, a$< 3 >$, a$< 0 >$	Capacitor tuning bits
4	Vdda	1.8 V supply voltage for the loop filter and biasing
5, 11, 15, 18, 19, 20, 33, 53, 65, 66, 69, 70, 73	gnda	Ground
6, 7	Clkp, Clkm	Differential input clocks
8	$\mathrm{OSR\_ctrl}$	Control to choose between OSR =25 (low), OSR =10 (high)
9	$\mathrm{Iref\_5uA}$	Reference current source $5\mu A$(sinking current)
10	$\mathrm{Vref\_flash}$	Control for changing input swing of flash ADC. Nominal value =0.9 V
12	Vddd	1.8 V supply voltage for the flash ADC
13	Vcalon	Control for making calibration on (high) and off (low).
14	$\mathrm{Vref\_dac}$	Control for DAC unit cell current. Nominal value =0.9 V
16	Vdddac	1.8 V supply voltage for the DAC
17	rst	Reset the digital calibration logic (Reset when high)
21	Vdddriver	1.8 V supply voltage for the Digital driver
22, 23	Vdigoutm, Vdigoutp	Differential output signals from Digital driver

24	lvdsen	Control for lvds buffers to enable (high) and disable (low).
25	driveren	Control for Digital driver to enable (high) and disable (low).
26	decfilen	Control for Decimation filter to enable (high) and disable (low).
27-31	-	Unused pins
32	Clkout	Output clock of Decimation filter
34, 54	Vdddecfil	1.8 V supply voltage for the Decimation filter
35-52	$\mathrm{decfilout} < 0 >$ to $\mathrm{decfilout} < 17 >$	Output data stream of Decimation filter
55, 56	$\mathrm{Clkb\_out}$, $\mathrm{Clk\_out}$	Differential output LVDS clocks
57-64	$\mathrm{Doutb} < 3 >$ to $\mathrm{Dout} < 0 >$	Deserialized differential output LVDS data stream
67, 68	$\mathrm{Vdd\_demux}$	1.8 V supply voltage for the Demux and IO pads
71, 72	Vinm, Vinp	Differential input signals
74	Vcm	Common mode reference voltage of 0.9 V
75	$\mathrm{Iref\_25p25}\,\mu\mathrm{A}$	Reference current source of value $25.5\mu A$(source current)

REFERENCES

[1] (). [Online].Available: www.stanford.edu/ phartke/chipscope_tutorial.pdf.

[2] **Candy, J. C.**, Decimation for sigma delta modulation. *In IEEE Trans. on Communications*, volume COM-34. 1986.

[3] **Cherry, J.** and **W. Snelgrove** (1999). Excess loop delay in continuous-time delta-sigma modulators. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, **46**(4), 376–389.

[4] **Doppalapudi, N. R.** (2006). *Decimator, Interpolator, and DEM Techniques for Oversampling $\Delta\Sigma$ Data converters*. Master's thesis, IIT Madras.

[5] **Hogenauer, E. B.**, An economical class of digital filters for decimation and interpolation. *In IEEE Trans. on Acoust., Speech and Signal Processing*, volume ASSP-29. 1981.

[6] **Jishnu, C.**, *20MHz Bandwidth Decimation Filter for a fourth order 1GS/s $\Delta\Sigma$ Modulator*. IIT Madras, 2010.

[7] **Pavan, S.** (November 2008). Excess loop delay compensation in continuous-time delta-sigma modulators. *IEEE Transactions on Circuits and Systems II: Express briefs*, **55, no.11**, 1119–1123.

[8] **Pavan, S.** and **N. Krishnapura**, Oversampling Analog-to-Digital Converter Design. *In 21st International Conference on VLSI Design 2008, VLSID.*. 2008.

[9] **Schreier, R.** (). $\Delta\,\Sigma$ Toolbox. *[Online]. Available: http://www.math- works.com*.

[10] **Singh, V.**, *Design of a High Speed High Resolution Continuous-Time Delta Sigma Modulator*. IIT Madras, 2010.

[11] **Vikas Singh, Nagendra Krishnapura, Shanthi Pavan** (2010). Compensating for Quantizer Delay in Excess of One Clock Cycle in Continuous-Time $\Delta\Sigma$ Modulators. *IEEE Transactions on Circuits and Systems*, **43**(2), 351–360.

[12] **Wallace, C. S.**, A suggestion for a fast multipler. *In IEEE Trans. Electron. Comput.*, volume EC-13. 1964.

[13] **Yan, S.** and **E. Sanchez-Sinencio** (2004). A continuous-time $\Sigma\,\Delta$ modulator with 88-dB dynamic range and 1.1-MHz signal bandwidth. *IEEE Journal of Solid-State Circuits*, **39**(1), 75–86.